"Linkages" Code Report

Thursday, November 8, 2007 -- 5:04:47 PM

Visual Basic 6 Project Report:

--> Project Title = "Linkages"
--> Project Executable = "Linkages.dll"
--> Project Name = "Linkages"
--> Project Description = "Corridor Designer: Tools to generate and analyze wildlife habitat corridors."
--> Project Version = 1.4.739
--> Project Company Name = "Jenness Enterprises"
--> Project Resource File = "Linkages.RES"
--> Project Filename = d:\arcgis_stuff\consultation\az_linkages\vb_code\linkages.vbp

---

General Statistics:

--> 14 forms...
--> 21 classes...
--> 12 modules...
--> 37,487 lines of code
--> 1,426,883 total characters

14Forms: -- -- -- -- -- -- -- -- -- -- --

1] frm_Summarize.frm
--> File Location = d:_stuff_linkages_code_Summarize.frm
--> File Exists = true
--> Number of Lines = 4,485
--> Number of Characters = 196,629

2] frmAbout.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 596
--> Number of Characters = 26,842

3] frmBottleneck.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 1,736
--> Number of Characters = 58,454

4] frmClip.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 1,218
--> Number of Characters = 43,375

5] frmEsrIllustration.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 61
--> Number of Characters = 1,905

6] frmEsriSample.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 149
--> Number of Characters = 5,524

7] frmGraph.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 2,705
--> Number of Characters = 108,793

8] frmGraphicsShapefile.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 1,609
--> Number of Characters = 77,137

9] frmHabSuitStats.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 661
--> Number of Characters = 22,998

10] frmReport_modal.frm
--> File Location = d:_stuff_linkages_code_modal.frm
--> File Exists = true
--> Number of Lines = 466
--> Number of Characters = 16,291

11] frmSelScreen.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 1,002
--> Number of Characters = 33,121

12] frmStep2.frm
--> File Location = d:_stuff_linkages_code.frm
--> File Exists = true
--> Number of Lines = 19
--> Number of Characters = 509

13] jennessent_compareparameters.frm
--> File Location = d:_stuff_linkages_code_compareparameters.frm
--> File Exists = true
--> Number of Lines = 2,410
--> Number of Characters = 79,976

14] progress_single.frm
--> File Location = d:_stuff_linkages_code_single.frm
--> File Exists = true
--> Number of Lines = 534
--> Number of Characters = 16,806

---

21 Classes: -- -- -- -- -- -- -- -- -- -- --

1] Anchor
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 52
--> Number of Characters = 1,550

2] AnchorObject
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true

--> Number of Lines = 358
--> Number of Characters = 18,047
3] AnchorObjectList
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 101
--> Number of Characters = 2,974
4] clsBottleneckDemo
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 227
--> Number of Characters = 7,460
5] clsToolBar
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 92
--> Number of Characters = 2,977
6] clsToolBarForConference
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 104
--> Number of Characters = 3,423
7] cmdAbout
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 190
--> Number of Characters = 6,103
8] cmdBottleneck
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 225
--> Number of Characters = 7,494
9] cmdClip
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 231
--> Number of Characters = 6,636
10] cmdCompare
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 220
--> Number of Characters = 7,051
11] cmdDeleteCorGraphics
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 253
--> Number of Characters = 7,943
12] cmdHabSuitStats
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 296
--> Number of Characters = 9,855
13] cmdHistogramStats
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 301
--> Number of Characters = 9,004
14] cmdNewShapefile
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 219
--> Number of Characters = 6,238
15] cmdOpenTable
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 517
--> Number of Characters = 17,294
16] cmdSumMod
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 394
--> Number of Characters = 12,355
17] cmdTestCode
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 253
--> Number of Characters = 8,777
18] CollectionMod
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 211
--> Number of Characters = 5,276
19] Extension
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 945
--> Number of Characters = 35,789
20] toolDrawPoly
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 506
--> Number of Characters = 16,812
21] toolReturnCoords
--> File Location = d:_stuff_linkages_code.cls
--> File Exists = true
--> Number of Lines = 526
--> Number of Characters = 17,038

---

12 Modules: -- -- -- -- -- -- -- -- -- -- --

1] aml_func_mod
--> File Location = d:_stuff_linkages_code_func_mod.bas
--> File Exists = true
--> Number of Lines = 1,842
--> Number of Characters = 61,757
2] CorridorAnalysisFunctions
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 4,293
--> Number of Characters = 173,195
3] CorridorSampleData
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 1,042
--> Number of Characters = 61,993
4] ErrorHandling
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 93
--> Number of Characters = 4,750
5] GridFunctions
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 1,320
--> Number of Characters = 46,574
6] modClipFunctions
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 629
--> Number of Characters = 25,641
7] modGenFunctions
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 15
--> Number of Characters = 554
8] modHelpStrings
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 74
--> Number of Characters = 8,655

9] MyGeneralOperations
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 2,280
--> Number of Characters = 78,357
10] MyGeometricOperations
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 1,369
--> Number of Characters = 48,853
11] MyVBOperations
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 23
--> Number of Characters = 674
12] QuickSort
--> File Location = d:_stuff_linkages_code.bas
--> File Exists = true
--> Number of Lines = 635
--> Number of Characters = 17,424

---

45 References:

1] COMDLG32.OCX
2] ErrorHandlerUI 1.0 Type Library
3] ESRI 3DAnalyst Object Library
4] ESRI ArcCatalogUI Object Library
5] ESRI ArcMap Object Library
6] ESRI ArcMapUI Object Library
7] ESRI Carto Object Library
8] ESRI CartoUI Object Library
9] ESRI Catalog Object Library
10] ESRI CatalogUI Object Library
11] ESRI DataSourcesFile Object Library
12] ESRI DataSourcesGDB OBJECT Library
13] ESRI DataSourcesOleDB Object Library
14] ESRI DataSourcesRaster Object Library
15] ESRI DataSourcesRasterUI Object Library
16] ESRI Display Object Library
17] ESRI DisplayUI Object Library
18] ESRI Editor Object Library
19] ESRI EditorExt Object Library
20] ESRI Framework Object Library
21] ESRI GeoAnalyst Object Library
22] ESRI GeoDatabase Object Library
23] ESRI GeoDatabaseDistributed Object Library
24] ESRI GeoDatabaseDistributedUI Object Library
25] ESRI GeoDatabaseUI Object Library
26] ESRI Geometry Object Library
27] ESRI Geoprocessing Object Library
28] ESRI GeoprocessingUI Object Library
29] ESRI GeoReferenceUI Object Library
30] ESRI Location Object Library
31] ESRI LocationUI Object Library
32] ESRI NetworkAnalysis Object Library
33] ESRI Output Object Library
34] ESRI OutputExtensions Object Library
35] ESRI OutputExtensions User Interface Library.
36] ESRI OutputUI Object Library
37] ESRI Server Object Library
38] ESRI SpatialAnalyst Object Library
39] ESRI SpatialAnalystUI Object Library
40] ESRI System Object Library
41] ESRI SystemUI Object Library
42] ESRI SystemUtility Object Library
43] MSCOMCTL.OCX
44] OLE Automation
45] RICHTX32.OCX

Form, Module and Class Code

Form 1: frm_Summarize.frm

```
VERSION 5.00
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin VB.Form frm_Summarize
   Caption         =   "Summary Statistics:"
   ClientHeight    =   6930
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   5685
   Icon            =   "frm_Summarize.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   6930
   ScaleWidth      =   5685
   StartUpPosition =   1  'CenterOwner
   Begin VB.CommandButton cmdCancel
      Caption         =   "Close"
      Height          =   375
      Left            =   2745
      TabIndex        =   9
      Top             =   6540
      Width           =   870
   End
   Begin VB.CommandButton cmdOK
      Caption         =   "OK"
      Height          =   375
      Left            =   4545
      TabIndex        =   8
      Top             =   6540
      Width           =   870
   End
   Begin VB.CommandButton cmdHelp
      Caption         =   "Help"
      Height          =   375
      Left            =   3652
      TabIndex        =   7
      Top             =   6540
      Width           =   870
```

```
End
Begin VB.CommandButton cmdOpenTable
   Height          =   375
   Left            =   45
   Picture         =   "frm_Summarize.frx":038A
   Style           =   1  'Graphical
   TabIndex        =   5
   ToolTipText     =   "Open Attributes Table"
   Top             =   6540
   Width           =   420
End
Begin VB.CommandButton cmdSwitchSel
   Height          =   375
   Left            =   525
   Picture         =   "frm_Summarize.frx":040C
   Style           =   1  'Graphical
   TabIndex        =   6
   ToolTipText     =   "Switch Selection"
   Top             =   6540
   Width           =   420
End
Begin VB.Frame frmGeneral
   Height          =   5535
   Left            =   45
   TabIndex        =   11
   Top             =   915
   Width           =   5565
   Begin VB.ComboBox cbxLayer
      Height          =   315
      Left            =   330
      Style           =   2  'Dropdown List
      TabIndex        =   0
      Top             =   435
      Width           =   4950
   End
   Begin VB.TextBox txtOutput
      Height          =   330
      Left            =   330
      Locked          =   -1  'True
      TabIndex        =   2
      Top             =   4635
      Width           =   4440
   End
   Begin VB.CheckBox chkSumSelected
      Caption         =   "Calculate Statistics on Selected Records Only"
      Height          =   450
      Left            =   315
```

```
   TabIndex        =   4
   Top             =   5040
   Width           =   5085
End
Begin VB.CommandButton cmdGetFile
   Height          =   360
   Left            =   4875
   Picture         =   "frm_Summarize.frx":049C
   Style           =   1  'Graphical
   TabIndex        =   3
   ToolTipText     =   "Browse for Output Filename..."
   Top             =   4605
   Width           =   555
End
Begin MSComctlLib.TreeView treeFields
   Height          =   3030
   Left            =   330
   TabIndex        =   1
   Top             =   1200
   Width           =   4920
   _ExtentX        =   8678
   _ExtentY        =   5345
   _Version        =   393217
   Indentation     =   617
   LabelEdit       =   1
   LineStyle       =   1
   Style           =   7
   BorderStyle     =   1
   Appearance      =   1
End
Begin VB.Label lblLayer
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "1.  Select a data layer or table:"
   Height          =   195
   Left            =   135
   TabIndex        =   14
   Top             =   195
   Width           =   2160
End
Begin VB.Label lbl3
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "3.  Specify folder for output tables:"
   Height          =   195
   Left            =   135
   TabIndex        =   13
```

```
      Top             =   4380
      Width           =   2415
   End
   Begin VB.Label lbl2
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "2.  Choose one or more statistics to calculate from the following data fields:  "
      Height          =   195
      Left            =   135
      TabIndex        =   12
      Top             =   975
      Width           =   5340
   End
End
Begin MSComctlLib.ImageList imgImageList
   Left            =   4035
   Top             =   855
   _ExtentX        =   1005
   _ExtentY        =   1005
   BackColor       =   -2147483643
   ImageWidth      =   16
   ImageHeight     =   16
   MaskColor       =   12632256
   _Version        =   393216
   BeginProperty Images {2C247F25-8591-11D1-B16A-00C0F0283628}
      NumListImages   =   6
      BeginProperty ListImage1 {2C247F27-8591-11D1-B16A-00C0F0283628}
         Picture         =   "frm_Summarize.frx":0512
         Key             =   ""
      EndProperty
      BeginProperty ListImage2 {2C247F27-8591-11D1-B16A-00C0F0283628}
         Picture         =   "frm_Summarize.frx":0577
         Key             =   ""
      EndProperty
      BeginProperty ListImage3 {2C247F27-8591-11D1-B16A-00C0F0283628}
         Picture         =   "frm_Summarize.frx":05EF
         Key             =   ""
      EndProperty
      BeginProperty ListImage4 {2C247F27-8591-11D1-B16A-00C0F0283628}
         Picture         =   "frm_Summarize.frx":069A
         Key             =   ""
      EndProperty
      BeginProperty ListImage5 {2C247F27-8591-11D1-B16A-00C0F0283628}
         Picture         =   "frm_Summarize.frx":0776
         Key             =   ""
      EndProperty
      BeginProperty ListImage6 {2C247F27-8591-11D1-B16A-00C0F0283628}
```

```
            Picture         =   "frm_Summarize.frx":082B
            Key             =   ""
         EndProperty
      EndProperty
   End
   Begin VB.Label lblSum2
      BackStyle       =   0  'Transparent
      Caption         =   "Statistics will be displayed in a Report window, and saved in dBASE tables in folder specified below."
      Height          =   495
      Left            =   45
      TabIndex        =   15
      Top             =   540
      Width           =   4455
      WordWrap        =   -1  'True
   End
   Begin VB.Image imgCorrIcon
      Height          =   855
      Left            =   4665
      Picture         =   "frm_Summarize.frx":08D1
      Top             =   60
      Width           =   945
   End
   Begin VB.Image imgCornerBars
      Height          =   225
      Left            =   5430
      Picture         =   "frm_Summarize.frx":33D5
      Top             =   6705
      Width           =   225
   End
   Begin VB.Label lblSummarize
      BackStyle       =   0  'Transparent
      Caption         =   "Calculate statistics for one or more fields from any feature dataset, raster dataset or standalone table:"
      Height          =   465
      Left            =   45
      TabIndex        =   10
      Top             =   45
      Width           =   4455
   End
End
Attribute VB_Name = "frm_Summarize"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private m_TableLayers As Collection
```

```
Private m_TableLayerNames As Collection ' ACTUAL LAYER NAMES
Private m_CurrentSelected As IUnknown
'Private m_TableFields As ITableFields
Private m_TableFields As IUnknown
Private m_TableFieldIndices() As Long
Private m_selLayer As IUnknown
Private m_Return As IVariantArray
'Private m_SumHelp As frmSumHelp
Private m_App As IApplication
Private m_ExtensionConfig As IExtensionConfig

' STATISTICS VARIABLES
Private m_excludeString As String
Private m_ShouldExclude As Boolean
'Private m_ShouldAdvanced As Boolean
Private m_ShouldDoAll As Boolean
Private m_ReportForm As frmReport_modal
Private m_NumDecPlaces As Integer

Private Anchors As AnchorObjectList  ' Main anchor control object
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frm_Summarize.frm"


'Public Property Set Field_Array(ByVal vNewValue As ITableFields)
Public Property Set Field_Array(ByVal vNewValue As IUnknown)
  On Error GoTo ErrorHandler


32:   Set m_TableFields = vNewValue


  Exit Property
ErrorHandler:
  HandleError True, "Field_Array " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Public Property Set SelectedLayer(ByVal vNewValue As IUnknown)
  On Error GoTo ErrorHandler


44:   Set m_selLayer = vNewValue


  Exit Property
ErrorHandler:
  HandleError True, "SelectedLayer " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Property

Public Property Set ArcApp(ByVal vNewValue As IApplication)
  On Error GoTo ErrorHandler


56:   Set m_App = vNewValue


  Exit Property
ErrorHandler:
  HandleError True, "ArcApp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Set theTableLayers(ByVal theLayers As Collection)
  On Error GoTo ErrorHandler


68:   Set m_TableLayers = theLayers


  Exit Property
ErrorHandler:
  HandleError True, "theTableLayers " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set theCurrentSelected(ByVal aPossibleLayer As IUnknown)
  On Error GoTo ErrorHandler


80:   Set m_CurrentSelected = aPossibleLayer


  Exit Property
ErrorHandler:
  HandleError True, "theCurrentSelected " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Private Sub Fill_TreeView()
  On Error GoTo ErrorHandler


' IMAGELIST:
' 1) UNCHECKED
' 2) CHECKED
```

```
' 3) TEXT
' 4) NUMBER
' 5) ID
' 6) DATE

  Dim thePointNumberStats(6) As String
100:   thePointNumberStats(0) = "Minimum"
101:   thePointNumberStats(1) = "Maximum"
102:   thePointNumberStats(2) = "Mean"
103:   thePointNumberStats(3) = "Sum"
104:   thePointNumberStats(4) = "Standard Deviation"
105:   thePointNumberStats(5) = "Variance"
106:   thePointNumberStats(6) = "Histogram"

  Dim theLineNumberStats(8) As String
109:   theLineNumberStats(0) = "Minimum"
110:   theLineNumberStats(1) = "Maximum"
111:   theLineNumberStats(2) = "Sum"
112:   theLineNumberStats(3) = "Mean"
113:   theLineNumberStats(4) = "Mean_WBL"
114:   theLineNumberStats(5) = "Standard Deviation"
115:   theLineNumberStats(6) = "Standard Deviation_WBL"
116:   theLineNumberStats(7) = "Variance"
117:   theLineNumberStats(8) = "Variance_WBL"

  Dim theAreaNumberStats(8) As String
120:   theAreaNumberStats(0) = "Minimum"
121:   theAreaNumberStats(1) = "Maximum"
122:   theAreaNumberStats(2) = "Sum"
123:   theAreaNumberStats(3) = "Mean"
124:   theAreaNumberStats(4) = "Mean_WBA"
125:   theAreaNumberStats(5) = "Standard Deviation"
126:   theAreaNumberStats(6) = "Standard Deviation_WBA"
127:   theAreaNumberStats(7) = "Variance"
128:   theAreaNumberStats(8) = "Variance_WBA"

  Dim theCategoryPointStats(1) As String
131:   theCategoryPointStats(0) = "Count"
132:   theCategoryPointStats(1) = "Proportion"

  Dim theCategoryLineStats(2) As String
135:   theCategoryLineStats(0) = "Count"
136:   theCategoryLineStats(1) = "Proportion"
137:   theCategoryLineStats(2) = "Length"

  Dim theCategoryAreaStats(2) As String
140:   theCategoryAreaStats(0) = "Count"
```

```
141:   theCategoryAreaStats(1) = "Proportion"
142:   theCategoryAreaStats(2) = "Area"

  Dim theDateStats(1) As String
145:   theDateStats(0) = "First"
146:   theDateStats(1) = "Last"

  Dim theContinuousGridStats(4) As String
149:   theContinuousGridStats(0) = "Minimum"
150:   theContinuousGridStats(1) = "Maximum"
151:   theContinuousGridStats(2) = "Mean"
152:   theContinuousGridStats(3) = "Standard Deviation"
153:   theContinuousGridStats(4) = "Histogram"

  Dim theIntegerGridStringStats(2) As String
156:   theIntegerGridStringStats(0) = "Count"
157:   theIntegerGridStringStats(1) = "Proportion"
158:   theIntegerGridStringStats(2) = "Area"

  Dim theIntegerGridNumberStats(10) As String
161:   theIntegerGridNumberStats(0) = "Minimum"
162:   theIntegerGridNumberStats(1) = "Maximum"
163:   theIntegerGridNumberStats(2) = "Mean"
164:   theIntegerGridNumberStats(3) = "Median"
165:   theIntegerGridNumberStats(4) = "Mode"
166:   theIntegerGridNumberStats(5) = "Sum"
167:   theIntegerGridNumberStats(6) = "Standard Deviation"
168:   theIntegerGridNumberStats(7) = "Histogram"
169:   theIntegerGridNumberStats(8) = "Count"
170:   theIntegerGridNumberStats(9) = "Proportion"
171:   theIntegerGridNumberStats(10) = "Area"

'  Dim theCurrentField As Long
'  theCurrentField = m_TableFieldIndices(cbxFieldName.ListIndex)

  Dim aNode As Node

  Dim anIndex As Long
  Dim anIndexString As String
180:   anIndexString = CStr(cbxLayer.ListIndex)
  Dim anIndex2 As Long

  Dim theAlias As String
  Dim theType As Integer

186:   treeFields.Nodes.Clear
```

```
  ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
190:   Set psbar = m_App.StatusBar
  Dim pPro As IStepProgressor
192:   Set pPro = psbar.ProgressBar

'  MsgBox "Finished Dimensioning TreeView Variables (Fill_TreeView)..."  '
*************************************************************

196:   Screen.MousePointer = vbHourglass

'  MsgBox "Finished Changing MousePointer (Fill_TreeView)..."  ' ************************************************************

    ' REMEMBER m_TableFields IS IUnknown:  COULD BE FEATURE LAYER, RASTER LAYER OR STANDALONE TABLE
201:   Set m_TableFields = m_TableLayers.Item(anIndexString)
202:   Set m_selLayer = m_TableFields
  Dim pTableFields As ITableFields
  Dim pFeatureClass As IFeatureClass
  Dim pFeatureLayer As IFeatureLayer
  Dim pLayerFields As ILayerFields
  Dim enumShapeType As esriGeometryType

209:   If TypeOf m_TableFields Is IRasterLayer Then        ' IF RASTER LAYER
    Dim pRasterLayer As IRasterLayer
    Dim pRaster As IRaster
    Dim pRasterBand As IRasterBand
    Dim pRasterBandCollection As IRasterBandCollection
    Dim pRasterDataset As IRasterDataset
    Dim booHasTable As Boolean
    Dim pFields As IFields

218:     Set pRasterLayer = m_TableFields
219:     Set pRaster = pRasterLayer.Raster
220:     Set pRasterBandCollection = pRaster
221:     Set pRasterBand = pRasterBandCollection.Item(0)
222:     Set pRasterDataset = pRasterBand.RasterDataset
223:     pRasterBand.HasTable booHasTable

225:     If booHasTable Then                    ' USE VAT TABLE FIELDS
      Dim pTable As ITable
227:       Set pTable = pRasterBand.AttributeTable
228:       Set pFields = pTable.Fields
229:       Set pLayerFields = pRasterLayer
230:       psbar.ShowProgressBar "Examining fields in " & cbxLayer.List(cbxLayer.ListIndex) & "...", 1, _
           pLayerFields.FieldCount, 1, True
232:       For anIndex = 0 To pLayerFields.FieldCount - 1
233:         If pLayerFields.FieldInfo(anIndex).Visible = True Then     ' ONLY SHOW VISIBLE FIELDS
```

```
234:           theAlias = pLayerFields.FieldInfo(anIndex).Alias         ' LIST FIELDS BY ALIAS
235:           theType = pLayerFields.Field(anIndex).Type
236:           anIndexString = "key_" & CStr(anIndex)
237:           If theType < 4 Then           ' NUMERIC VALUES ARE LESS THAN 4
238:             Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 4)
239:             aNode.Tag = "Field"

241:             For anIndex2 = 0 To 10
242:               Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                    theIntegerGridNumberStats(anIndex2), 1)
244:               aNode.Tag = "Stat"
245:             Next anIndex2

247:           ElseIf theType = 4 Then       ' STRING VALUE
248:             Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 3)
249:             aNode.Tag = "Field"
250:             For anIndex2 = 0 To 2
251:               Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                   theIntegerGridStringStats(anIndex2), 1)
253:               aNode.Tag = "Stat"
254:             Next anIndex2
255:           ElseIf theType = 5 Then       ' DATE VALUE
256:             Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 6)
257:             aNode.Tag = "Field"
258:             For anIndex2 = 0 To 1
259:               Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                   theDateStats(anIndex2), 1)
261:               aNode.Tag = "Stat"
262:             Next anIndex2
263:           End If
264:         End If

266:         psbar.StepProgressBar
267:       Next anIndex
268:     Else                                  ' USE STANDARD STATS AND HISTOGRAM

270:       anIndexString = "key_0"

272:       theAlias = "Continuous Grid Values"
273:       Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 4)
274:       aNode.Tag = "Field"

276:       For anIndex2 = 0 To 4
277:         Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
              theContinuousGridStats(anIndex2), 1)
279:         aNode.Tag = "Stat"
280:       Next anIndex2
```

```
282:     End If

284:   ElseIf TypeOf m_TableFields Is IFeatureLayer Then        ' IF FEATURE LAYER
285:     Set pTableFields = m_TableFields
286:     Set pFeatureLayer = m_TableFields
287:     Set pFeatureClass = pFeatureLayer.FeatureClass
288:     enumShapeType = pFeatureClass.ShapeType

    Dim intGeometryDim As esriGeometryDimension
    Select Case enumShapeType
      Case 1, 2                                  ' POINT FEATURES
        intGeometryDim = esriGeometry0Dimension
      Case 3, 6, 13, 14, 15, 16                  ' LINEAR FEATURES
        intGeometryDim = esriGeometry1Dimension
      Case 4, 5, 9, 11, 18, 19, 22               ' AREAL FEATURES
        intGeometryDim = esriGeometry2Dimension
      Case Else                                  ' UNKNOWN
        intGeometryDim = esriGeometry0Dimension
300:    End Select

302:    psbar.ShowProgressBar "Examining fields in " & cbxLayer.List(cbxLayer.ListIndex) & "...", 1, _
         pTableFields.FieldCount, 1, True

305:    For anIndex = 0 To pTableFields.FieldCount - 1

307:      If pTableFields.FieldInfo(anIndex).Visible = True Then     ' ONLY SHOW VISIBLE FIELDS
308:        theAlias = pTableFields.FieldInfo(anIndex).Alias         ' LIST FIELDS BY ALIAS
309:        theType = pTableFields.Field(anIndex).Type
310:        anIndexString = "key_" & CStr(anIndex)
311:        If theType < 4 Then          ' NUMERIC VALUES ARE LESS THAN 4
312:          Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 4)
313:          aNode.Tag = "Field"
         Select Case intGeometryDim
           Case esriGeometry0Dimension         ' POINTS
316:             For anIndex2 = 0 To 6
317:               Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                    thePointNumberStats(anIndex2), 1)
319:               aNode.Tag = "Stat"
320:             Next anIndex2
           Case esriGeometry1Dimension         ' LINEAR FEATURES
322:             For anIndex2 = 0 To 8
323:               Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                    theLineNumberStats(anIndex2), 1)
325:               aNode.Tag = "Stat"
326:             Next anIndex2
           Case esriGeometry2Dimension         ' AREAL FEATURES
```

```
328:               For anIndex2 = 0 To 8
329:                 Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                      theAreaNumberStats(anIndex2), 1)
331:                 aNode.Tag = "Stat"
332:               Next anIndex2
333:             End Select
334:           ElseIf theType = 4 Then      ' STRING VALUE
335:             Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 3)
336:             aNode.Tag = "Field"
          Select Case intGeometryDim
            Case esriGeometry0Dimension         ' POINTS
339:               For anIndex2 = 0 To 1
340:                 Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                     theCategoryPointStats(anIndex2), 1)
342:                 aNode.Tag = "Stat"
343:               Next anIndex2
            Case esriGeometry1Dimension         ' LINEAR FEATURES
345:               For anIndex2 = 0 To 2
346:                 Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                     theCategoryLineStats(anIndex2), 1)
348:                 aNode.Tag = "Stat"
349:               Next anIndex2
            Case esriGeometry2Dimension         ' AREAL FEATURES
351:               For anIndex2 = 0 To 2
352:                 Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                     theCategoryAreaStats(anIndex2), 1)
354:                 aNode.Tag = "Stat"
355:               Next anIndex2
356:             End Select
357:           ElseIf theType = 5 Then      ' DATE VALUE
358:             Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 6)
359:             aNode.Tag = "Field"
360:             For anIndex2 = 0 To 1
361:               Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                    theDateStats(anIndex2), 1)
363:               aNode.Tag = "Stat"
364:             Next anIndex2
365:           End If
366:         End If

368:         psbar.StepProgressBar
369:       Next anIndex
370:     ElseIf TypeOf m_TableFields Is IStandaloneTable Then        ' IF STANDALONE TABLE

    Dim pstandalonetable As IStandaloneTable
373:       Set pTableFields = m_TableFields
374:       Set pstandalonetable = m_TableFields
```

```
376:     psbar.ShowProgressBar "Examining fields in " & cbxLayer.List(cbxLayer.ListIndex) & "...", 1, _
          pTableFields.FieldCount, 1, True

379:     For anIndex = 0 To pTableFields.FieldCount - 1

381:       If pTableFields.FieldInfo(anIndex).Visible = True Then     ' ONLY SHOW VISIBLE FIELDS
382:         theAlias = pTableFields.FieldInfo(anIndex).Alias         ' LIST FIELDS BY ALIAS
383:         theType = pTableFields.Field(anIndex).Type
384:         anIndexString = "key_" & CStr(anIndex)
385:         If theType < 4 Then          ' NUMERIC VALUES ARE LESS THAN 4
386:           Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 4)
387:           aNode.Tag = "Field"

389:           For anIndex2 = 0 To 6
390:             Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                  thePointNumberStats(anIndex2), 1)
392:             aNode.Tag = "Stat"
393:           Next anIndex2

395:         ElseIf theType = 4 Then      ' STRING VALUE
396:           Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 3)
397:           aNode.Tag = "Field"

399:           For anIndex2 = 0 To 1
400:             Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                 theCategoryPointStats(anIndex2), 1)
402:             aNode.Tag = "Stat"
403:           Next anIndex2

405:         ElseIf theType = 5 Then      ' DATE VALUE
406:           Set aNode = treeFields.Nodes.Add(, , anIndexString, theAlias, 6)
407:           aNode.Tag = "Field"
408:           For anIndex2 = 0 To 1
409:             Set aNode = treeFields.Nodes.Add(anIndexString, tvwChild, anIndexString & "_" & CStr(anIndex2), _
                  theDateStats(anIndex2), 1)
411:             aNode.Tag = "Stat"
412:           Next anIndex2
413:         End If
414:       End If

416:       psbar.StepProgressBar
417:     Next anIndex
418:   End If
419:   psbar.HideProgressBar

'  MsgBox "Finished Hiding Progress Bar (Fill_TreeView)..."  ' ********************************************************
```

```
423:   Screen.MousePointer = vbDefault

'  MsgBox "Finished Resetting Mouse Pointer (Fill_TreeView)..."  ' ************************************************************


  Exit Sub
ErrorHandler:
  HandleError False, "Fill_TreeView " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

'Private Sub cbxFieldName_Click()
'
'  Call Fill_TreeView
'
'End Sub

Private Sub cbxLayer_Click()
  On Error GoTo ErrorHandler


'  cbxFieldName.Clear

  Dim anIndex As Long
  Dim pFieldInfo As IFieldInfo
  Dim anIndex2 As Long
  Dim booHasTable As Boolean
449:   booHasTable = True
  Dim anIndexString As String
  Dim intSelIndex As Long
452:   intSelIndex = cbxLayer.ListIndex
453:   If intSelIndex = -1 Then intSelIndex = 0

455:   anIndexString = CStr(intSelIndex)

  ' REMEMBER m_TableFields IS IUnknown:  COULD BE FEATURE LAYER, RASTER LAYER OR STANDALONE TABLE
458:   Set m_TableFields = m_TableLayers.Item(anIndexString)
459:   Set m_selLayer = m_TableFields

461:   If TypeOf m_TableFields Is IRasterLayer Then        ' IF RASTER LAYER

463:     cmdSwitchSel.Enabled = False

    Dim pRasterLayer As IRasterLayer
    Dim pRaster As IRaster
    Dim pRasterBand As IRasterBand
```

```
    Dim pRasterBandCollection As IRasterBandCollection
    Dim pRasterDataset As IRasterDataset
    Dim pLayerFields As ILayerFields
    Dim pFields As IFields

473:     Set pRasterLayer = m_TableFields
474:     Set pRaster = pRasterLayer.Raster
475:     Set pRasterBandCollection = pRaster
476:     Set pRasterBand = pRasterBandCollection.Item(0)
477:     Set pRasterDataset = pRasterBand.RasterDataset
478:     pRasterBand.HasTable booHasTable

480:     If booHasTable Then                    ' USE VAT TABLE FIELDS
      Dim pTable As ITable
482:       Set pLayerFields = pRasterLayer
483:       Set pTable = pRasterBand.AttributeTable
484:       Set pFields = pTable.Fields
      ReDim m_TableFieldIndices(pFields.FieldCount)
486:       anIndex2 = -1

488:       For anIndex = 0 To pFields.FieldCount - 1
489:         If pLayerFields.Field(anIndex).Type < 6 Then ' ONLY STRINGS, NUMBERS AND DATES
490:           Set pFieldInfo = pLayerFields.FieldInfo(anIndex)

492:           If pFieldInfo.Visible Then
'            cbxFieldName.AddItem (pFieldInfo.Alias)

495:             anIndex2 = anIndex2 + 1
496:             m_TableFieldIndices(anIndex2) = anIndex
497:           End If

499:         End If
500:       Next anIndex
501:     Else                                   ' USE STANDARD STATS AND HISTOGRAM
'      cbxFieldName.AddItem ("Continuous Grid Dataset")   ' THIS SINGLE FIELD OPTION WILL BE ASSIGNED A NUMERIC TYPE OF DOUBLE
      ReDim m_TableFieldIndices(0)
504:       m_TableFieldIndices(0) = 0
505:     End If

507:   Else      ' IF FEATURE LAYER OR STANDALONE TABLE

509:     cmdSwitchSel.Enabled = True
    Dim pTableFields As ITableFields
511:     Set pTableFields = m_TableFields
    ReDim m_TableFieldIndices(pTableFields.FieldCount)

514:     anIndex2 = -1
```

```
516:     For anIndex = 0 To pTableFields.FieldCount - 1
517:       If pTableFields.Field(anIndex).Type < 6 Then ' ONLY STRINGS, NUMBERS AND DATES
518:         Set pFieldInfo = pTableFields.FieldInfo(anIndex)

520:         If pFieldInfo.Visible Then
'          cbxFieldName.AddItem (pFieldInfo.Alias)

523:           anIndex2 = anIndex2 + 1
524:           m_TableFieldIndices(anIndex2) = anIndex
525:         End If

527:       End If
528:     Next anIndex
529:   End If

  'Debug.Print cbxLayer.List(cbxLayer.ListIndex)

'  cbxFieldName.ListIndex = 0
534:   cmdOpenTable.Enabled = booHasTable

536:   Call DisplayNumSelected
537:   Call Fill_TreeView
538:   Call CheckOK

  Exit Sub
ErrorHandler:
  HandleError True, "cbxLayer_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub



Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler


551:   Me.Visible = False


  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdGetFile_Click()
```

```
  On Error GoTo ErrorHandler

564:   SetWindowPos Me.hWnd, -2, 0, 0, 0, 0, &H1 Or &H2

  Dim strDirPath As String
  Dim strUserName As String

569:   strDirPath = txtOutput.Text
570:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then strDirPath = strDirPath & "\"
571:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
572:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_App))
573:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
574:   End If

  Dim boolWorkspaceExists As Boolean
577:   boolWorkspaceExists = Not Dir$(strDirPath) = ""

579:   If Not boolWorkspaceExists Then
580:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.TempPathLocation)
581:   End If

  Dim pGxDialog As IGxDialog
584:   Set pGxDialog = New GxDialog

  Dim pGxDialogFilter As IGxObjectFilter
'  Set pGxDialogFilter = New GxFilterWorkspaces
588:   Set pGxDialogFilter = New GxFilterBasicTypes
'  pGxDialogFilter.Name = "Folders"
'  Set pGxDialogFilter = New GxFilterContainers      ' INCLUDED GRIDS AND COVERAGES

  Dim pGxObject As IGxObject
  Dim pGxSelection As IEnumGxObject

595:   With pGxDialog
596:     .AllowMultiSelect = False
597:     .StartingLocation = strDirPath
598:     .Title = "Please select (don't open!) folder to contain your dBASE Tables:"
599:     Set .ObjectFilter = pGxDialogFilter
600:   End With

  Dim theFinalString As String
'  If Not pGxDialog.DoModalOpen(0, pEnumGx) Then
'Exit Sub 'Exit if user press Cancel
'End If
'MsgBox pEnumGx.Next.FullName
```

```
607:   If (pGxDialog.DoModalOpen(Me.hWnd, pGxSelection) = True) Then
    'Set pGxObject = pGxDialog.FinalLocation
609:     Set pGxObject = pGxSelection.Next

    Dim pGxFile As IGxFile
612:     Set pGxFile = pGxObject

614:     theFinalString = pGxObject.FullName
615:     If (Right(theFinalString, 1) <> "\") And (Right(theFinalString, 1) <> "/") Then theFinalString = theFinalString & "\"

'    If aml_func_mod.ExistFileDir(theFinalString) Then
'      theFinalString = aml_func_mod.MakeUniqueFilename(theFinalString)
'
'      MsgBox "Unable to overwrite the file '" & pGxDialog.Name & "'.  The new file will be saved to " & _
'        theFinalString & "..."
'    End If

624:     txtOutput.Text = theFinalString

626:   End If


629:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2









''  Dim strMxdPathShort As String
''  strMxdPathShort = GetPath
''
''  Dim strMxdPath As String
''  strMxdPath = aml_func_mod.GetFullFileString(strMxdPathShort)
''
''  Dim strDirPath As String
''  strDirPath = aml_func_mod.ReturnDir(strMxdPath)
''
''  Dim strSuggestFileName As String
''  strSuggestFileName = strDirPath & "summary.dbf"
''
''  strSuggestFileName = aml_func_mod.MakeUniqueFilename(strSuggestFileName)
'
```

```
'  Dim strDirPath As String
'  Dim strUserName As String
'
'  strUserName = aml_func_mod.GetTheUserName
'  strDirPath = "c:\Documents and Settings\" & strUserName & "\My Documents\"
'
'  Dim boolMediaFileExists As Boolean
'  boolMediaFileExists = Not Dir$(strDirPath) = ""
'
'  If Not boolMediaFileExists Then
'    strDirPath = "c:\Documents and Settings\My Documents\"
'    boolMediaFileExists = Not Dir$(strDirPath) = ""
'  End If
'
'  If Not boolMediaFileExists Then
'    strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.TempPathLocation)
'  End If
'
'  Dim strSuggestFileName As String
'  strSuggestFileName = aml_func_mod.MakeUniqueFilename(strDirPath & "summary.dbf")
'
'  Dim pGxDialog As IGxDialog
'  Set pGxDialog = New GxDialog
'
'  Dim pGxDialogFilter As IGxObjectFilter
'  Set pGxDialogFilter = New GxFilterdBASEFiles
'
'  Dim pGxObject As IGxObject
'  Dim pGxSelection As IEnumGxObject
'
'  With pGxDialog
'    .AllowMultiSelect = False
'    .StartingLocation = strDirPath
'    .Title = "Please specify where to save your Summary table:"
'    Set .ObjectFilter = pGxDialogFilter
'    .Name = aml_func_mod.ReturnFilename(strSuggestFileName)
'  End With
'
'  Dim theFinalString As String
'
'  If (pGxDialog.DoModalSave(m_App.hWnd) = True) Then
'    Set pGxObject = pGxDialog.FinalLocation
'
'    Dim pGxFile As IGxFile
'    Set pGxFile = pGxObject
'
'    theFinalString = pGxFile.Path & "\" & pGxDialog.Name
```

```
'
'    If aml_func_mod.ExistFileDir(theFinalString) Then
'      theFinalString = aml_func_mod.MakeUniqueFilename(theFinalString)
'
'      MsgBox "Unable to overwrite the file '" & pGxDialog.Name & "'.  The new file will be saved to " & _
'        theFinalString & "..."
'    End If
'
'    txtOutput.Text = theFinalString
'
'  End If

  Exit Sub
ErrorHandler:
  HandleError True, "cmdGetFile_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4

End Sub

Private Sub cmdHelp_Click()
  On Error GoTo ErrorHandler

  Dim strPath As String
723:   strPath = App.Path & "\help"

725:   Call Linkages.MyGeneralOperations.OpenDoc("General_Statistics_Subdocument.pdf", strPath)

  Exit Sub
ErrorHandler:
  HandleError True, "cmdHelp_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Command1_Click()
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
  HandleError False, "Command1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler
```

```
'  MsgBox "Just Clicked"
  ' SAVE SELECTED WORKSPACE
  Dim ext As Linkages.Extension
748:   Set ext = m_ExtensionConfig
  Dim strWorkFolder As String
750:   strWorkFolder = txtOutput.Text
751:   If Right(strWorkFolder, 1) <> "\" And Right(strWorkFolder, 1) <> "/" Then
752:     strWorkFolder = strWorkFolder & "\"
753:   End If
754:   ext.ClipDirectoryPath = strWorkFolder

  Dim strReport As String
  Dim strSubReport As String

  Dim strLetters() As String
760:   strLetters = ReturnOutlineLetters

  Dim theTimeBegan As Date
763:   theTimeBegan = Now

  Dim pNodes As Nodes
766:   Set pNodes = treeFields.Nodes

  Dim anIndex As Long
  Dim anIndex2 As Long
  Dim anIndex3 As Long

  Dim pNode As Node
  Dim pParentNode As Node
  Dim lngParentIndex As Long
  Dim strParentKey As String
  Dim lngFieldIndex As Long

  Dim strParentFieldName As String
  Dim intParentFieldType As Integer
  Dim strParentFieldType As String
  Dim strParentFieldAlias As String
  Dim strLayerType As String
  Dim intLayerType As Integer
  Dim strShapeType As String

  ' MAKE 2 ARRAYS:
  ' 1) VARIANT ARRAY CONTAINING STRING DEFINING LAYER TYPE (Feature or Raster),
  '    LAYER SUBTYPE (Point, Polyline, Polygon; Integer vs. Continuous Raster) AND LAYER
  ' 2) WORK ORDER VARIANT ARRAY, CONTAINING VARIANT STATISTIC ARRAYS FOR EACH STATISTIC.
  '    EACH SUB-VARIANT ARRAY CONTAINS:
  '    - FIELD INDEX
```

```
  '    - FIELD NAME
  '    - FIELD ALIAS
  '    - FIELD TYPE (NUMBER, STRING, DATE)
  '    - STATISTIC NAME
  '    - WHETHER THIS STATISTIC REQUIRES WEIGHTING (I.E. MEASURING THE GEOMETRY)

  Dim pWorkOrderArray As esriSystem.IVariantArray
799:   Set pWorkOrderArray = New esriSystem.VarArray
  Dim pLayerArray As esriSystem.IVariantArray
  Dim pStatAndFieldArray As esriSystem.IVariantArray
  Dim pFieldInfoArray As esriSystem.IStringArray
  Dim pStatInfoArray As esriSystem.IStringArray
  Dim booRequiresWeighting As Boolean
  Dim booRequiresStandardStats As Boolean
  Dim strStatName As String
  Dim pWorkOrderDictionary As Linkages.CollectionMod
  Dim pWorkOrderDictArray As esriSystem.IVariantArray
  Dim booHasKey As Boolean

  Dim pNumValArray As esriSystem.IDoubleArray
  Dim dblNumValArray() As Double
  Dim pStrValArray As esriSystem.IStringArray
  Dim strStringValArray() As Double
  Dim pDateValArray As esriSystem.IVariantArray
  Dim dateDateValArray() As Double
  Dim pSizeValArray As esriSystem.IDoubleArray
  Dim dblSizeValArray() As Double

  Dim booUseSelection As Boolean
821:   booUseSelection = (chkSumSelected.Value = 1)

  Dim theValueCount As Long
  Dim pSelectionSet As ISelectionSet
  Dim pCursor As ICursor
  Dim pTable As ITable
  Dim pTableSelection As ITableSelection
  Dim theValueIndex As Long
  Dim theNumberNull As Long
  Dim pRow As IRow

  ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
834:   Set psbar = m_App.StatusBar
  Dim pPro As IStepProgressor
836:   Set pPro = psbar.ProgressBar

  ' DIMENSION ALL THE STATS VARIABLES
```

```
Dim theSum As Double
Dim theMean As Double
Dim theMedian As Double
Dim theModeString As String
Dim theMinimum As Double
Dim theMaximum As Double
Dim theRange As Double
Dim theStdErrMean As Double
Dim theVar As Double
Dim theStdDev As Double
Dim theCount As Long
Dim theMeanWBL As Double
Dim theMeanWBA As Double
Dim theStDevWBL As Double
Dim theStDevWBA As Double
Dim theVarWBL As Double
Dim theVarWBA As Double
Dim theProportion As Double
Dim theLength As Double
Dim theArea As Double
Dim theFirst As Date
Dim theLast As Date
Dim lngNumberNull As Long
Dim lngHistArray() As Long
Dim theFinalModes() As Double
Dim booFoundMode As Boolean
Dim lngNumBins As Long
Dim lngBinCount As Long
Dim dblBinInterval As Double
Dim dblRunningBinVal As Double
Dim lngMaxBinCount As Long
Dim dblBinRatio As Double
Dim strHistReport As String

Dim pPolygon As IPolygon
Dim pPolyline As IPolyline
Dim pArea As IArea
Dim pMultipoint As IMultipoint
Dim pPointCollection As IPointCollection
Dim dblValue As Double
Dim strValue As String
Dim dateValue As Date
Dim dblSize As Double
Dim dblProportion As Double
Dim pVarArray As IVariantArray

Dim pOrigField As IField
```

```
  Dim strfilename As String

  Dim pClone As IClone
  Dim pNewStringField As IField

  ' SET NUMBER OF HISTOGRAM BINS HERE
892:   lngNumBins = 9

  Dim lngLetterInd As Long

  Dim pStats As IVariantArray
  Dim pWeightStats As IDoubleArray
  Dim pFeatureLayer As IFeatureLayer
  Dim pFeatureClass As IFeatureClass
  Dim lngShapeField As Long
  Dim pTableFields As ITableFields
  Dim pStTable As IStandaloneTable

  Dim pRasterLayer As IRasterLayer
  Dim pRaster As IRaster
  Dim pRasterBand As IRasterBand
  Dim pRasterBandCollection As IRasterBandCollection
  Dim pRasterDataset As IRasterDataset
  Dim pCountDataset As IRasterDataset
  Dim pRasterStatistics As IRasterStatistics
  Dim booHasTable As Boolean

  Dim pCountRasterBand As IRasterBand
  Dim pCountRasterBandCollection As IRasterBandCollection
  Dim pCountGeoDataset As IGeoDataset
  Dim pRasterProps As IRasterProps
  Dim lngNoDataValue As Long
  Dim varNoDataValue As Variant
  Dim pCountCursor As ICursor
  Dim pCountRow As IRow
  Dim lngCountValField As Long
  Dim lngCountCountField As Long
  Dim lngCountValValue As Long
  Dim lngCountCountValue As Long
  Dim pLogicalOp As ILogicalOp
  Dim booGridHistogram As Boolean
  Dim booGridCategorical As Boolean
  Dim booGridNumeric As Boolean
  Dim dblCellSize As Double

  Dim booContMin As Boolean
  Dim booContMax As Boolean
```

```
  Dim booContMean As Boolean
  Dim booContMode As Boolean
  Dim booContMedian As Boolean
  Dim booContSD As Boolean
  Dim booContHist As Boolean
  Dim lngNewFieldCounter As Long
  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit

  Dim pNewStandaloneTable As IStandaloneTable
  Dim pTableWindow2 As ITableWindow2
  Dim pStandaloneTableCollection As IStandaloneTableCollection
  Dim pMxDoc As IMxDocument

  Dim theKeys() As String       ' KEYS ARE STRING VERSIONS OF FIELD NUMBERS
  Dim aKey As String

  Dim lngNumDecPlaces As Long
953:   lngNumDecPlaces = 3

955:   If TypeOf m_TableFields Is IFeatureLayer Then

957:     Set pFeatureLayer = m_TableFields
958:     Set pTableFields = pFeatureLayer
959:     Set pFeatureClass = pFeatureLayer.FeatureClass
960:     Set pTable = pFeatureClass
961:     lngShapeField = pFeatureClass.FindField(pFeatureClass.ShapeFieldName)

    ' IDENTIFY FEATURE CLASS SHAPE TYPE
964:     intLayerType = pFeatureClass.ShapeType
    Select Case intLayerType
      Case 1
967:         strShapeType = "Point"
      Case 2
969:         strShapeType = "Multipoint"
      Case 3
971:         strShapeType = "Polyline"
      Case 4
973:         strShapeType = "Polygon"
      Case Else
975:         strShapeType = "Unknown"
976:     End Select

'    strReport = "Statistics Report on Feature Class '" & pFeatureLayer.Name & "':" & vbCrLf & _
       "-------------------------------------------------" & vbCrLf & _
```

```
        "Shape Type = " & strShapeType & vbCrLf & _
        "Records Analyzed = zzRecsAnalyzedzz out of zzTotalRecszz" & vbCrLf & _
        "-------------------------------------------------" & vbCrLf

984:     strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fprq2\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\tx90\tx360\tx450\tx720\b\f0\fs16 Statistics Report on Feature Class
'" & _
      pFeatureLayer.Name & "'\b0\par" & vbCrLf & _
      "\pard ------------------------------------------------------------------------------\par" & vbCrLf & _
      "\b Shape Type: \b0  " & strShapeType & "\par" & vbCrLf & _
      "\b Records Analyzed: \b0  zzRecsAnalyzedzz out of zzTotalRecszz\par" & vbCrLf & _
      "------------------------------------------------------------------------------\par" & vbCrLf

    ' MAKE AND FILL LAYER INFO ARRAY
994:     Set pLayerArray = New esriSystem.VarArray
995:     pLayerArray.Add "Feature Class"
996:     pLayerArray.Add strShapeType
997:     pLayerArray.Add pFeatureLayer

999:     For anIndex = 1 To pNodes.Count     ' CHECK ALL NODES IN TREEVIEW
1000:       Set pNode = pNodes.Item(anIndex)
1001:       If pNode.Image = 2 Then           ' THEN THIS STAT IS SELECTED
1002:         Set pParentNode = pNode.Parent
1003:         strParentKey = pParentNode.Key

        ' MAKE EMPTY FIELD INFO ARRAY
1006:         Set pStatAndFieldArray = New esriSystem.VarArray

        ' STAT AND FIELD ARRAY HAS VALUES FOR EACH SELECTED NODE IN TREE VIEW.  VALUES ARE:
        ' (0) Field Index
        ' (1) Field Name
        ' (2) Field Alias
        ' (3) Field Type (Number, String, Date)
        ' (4) Name of Statistic
        ' (5) Whether this statistic requires weights (i.e. geometry lengths, counts or areas)
        ' (6) Whether this statistic requires standard statistics (i.e. mean, sum, min, max, sd, var, range, etc.)

        ' GET INFORMATION ON FIELD AND ADD TO FIELD INFO ARRAY
1018:         lngParentIndex = CLng(Right(strParentKey, Len(strParentKey) - 4))
1019:         strParentFieldName = pTableFields.Field(lngParentIndex).Name
1020:         strParentFieldAlias = pTableFields.FieldInfo(lngParentIndex).Alias
1021:         intParentFieldType = pTableFields.Field(lngParentIndex).Type
        Select Case intParentFieldType
          Case Is < 4
1024:             strParentFieldType = "Number"
          Case 4
```

```
1026:             strParentFieldType = "String"
          Case 5
1028:             strParentFieldType = "Date"
          Case Else
1030:             strParentFieldType = "Unknown"
1031:         End Select

1033:         pStatAndFieldArray.Add lngParentIndex
1034:         pStatAndFieldArray.Add strParentFieldName
1035:         pStatAndFieldArray.Add strParentFieldAlias
1036:         pStatAndFieldArray.Add strParentFieldType
1037:         strStatName = pNode.Text
1038:         pStatAndFieldArray.Add strStatName

1040:         booRequiresWeighting = _
           (strShapeType <> "Point") And _
           (Right(strStatName, 3) = "WBL" Or Right(strStatName, 3) = "WBA" Or strStatName = "Proportion" Or _
                    (strParentFieldType = "String" And strStatName = "Length") Or _
                    (strParentFieldType = "String" And strStatName = "Area"))

1046:         booRequiresStandardStats = _
          (strParentFieldType = "String") Or _
          (strParentFieldType = "Date") Or _
          (strStatName = "Minimum") Or _
          (strStatName = "Maximum") Or _
          (strStatName = "Mean") Or _
          (strStatName = "Sum") Or _
          (strStatName = "Standard Deviation") Or _
          (strStatName = "Variance") Or _
          (strStatName = "Histogram") Or _
          (strStatName = "Mode")

1058:         pStatAndFieldArray.Add booRequiresWeighting
1059:         pStatAndFieldArray.Add booRequiresStandardStats

        ' ADD STAT ARRAY TO WORK ORDER ARRAY
1062:         pWorkOrderArray.Add pStatAndFieldArray
1063:       End If     ' END CHECKING IF THIS NODE IS SELECTED
1064:     Next anIndex

    ' TURN THIS INTO A DICTIONARY (COLLECTION MOD) WITH:
    '        KEY = STRING OF FIELD INDEX
    '    ELEMENT = VARIANT ARRAY WITH 7 ITEMS:
    '            1) STRING ARRAY WITH:
    '               a) FIELD NAME
    '               b) FIELD ALIAS
    '               c) FIELD TYPE (SHOULD BE "NUMBER", "STRING", OR "DATE")
```

```
    '            2) STRING ARRAY WITH ALL REQUESTED STATISTICS
    '            3) NUMBER, STRING OR DATE ARRAY FOR DATA VALUES, DEPENDING ON FIELD TYPE
    '            4) A NUMBER ARRAY FOR SHAPE SIZES (EACH FIELD GETS ITS OWN BECAUSE THEY MIGHT HAVE NULL VALUES)
    '            5) BOOLEAN INDICATING WHETHER SHAPE SIZE ARRAY SHOULD BE FILLED
    '            6) NUMBER (LONG) OF NULL VALUES FOUND IN THIS FIELD
    '            7) BOOLEAN INDICATING WHETHER STANDARD STATISTICS SHOULD BE CALCULATED

1080:     Set pWorkOrderDictionary = New Linkages.CollectionMod
1081:     For anIndex = 0 To pWorkOrderArray.Count - 1
1082:       Set pStatAndFieldArray = pWorkOrderArray.Element(anIndex)
1083:       lngFieldIndex = pStatAndFieldArray.Element(0)
1084:       booHasKey = pWorkOrderDictionary.HasKey(CStr(lngFieldIndex))
   '   MsgBox "Index = " & CStr(anIndex) & vbCrLf & _
            "Field Name = " & pStatAndFieldArray.Element(1) & vbCrLf & _
            "Statistic = " & pStatAndFieldArray.Element(4) & vbCrLf & _
            "Already Found Key = " & CStr(booHasKey)
1089:       If Not booHasKey Then
1090:         Set pFieldInfoArray = New esriSystem.strArray
1091:         pFieldInfoArray.Add pStatAndFieldArray.Element(1)       ' FIELD NAME
1092:         pFieldInfoArray.Add pStatAndFieldArray.Element(2)       ' FIELD ALIAS
1093:         pFieldInfoArray.Add pStatAndFieldArray.Element(3)       ' FIELD TYPE ("Number", "String", "Date")

1095:         Set pStatInfoArray = New esriSystem.strArray
      '  MsgBox pStatAndFieldArray.Element(4)
1097:         pStatInfoArray.Add pStatAndFieldArray.Element(4)

1099:         Set pWorkOrderDictArray = New esriSystem.VarArray
1100:         pWorkOrderDictArray.Add pFieldInfoArray                 ' ELEMENT (0)
1101:         pWorkOrderDictArray.Add pStatInfoArray                  ' ELEMENT (1)

        ' ADD EMPTY ARRAY TO HOLD VALUES
        Select Case pStatAndFieldArray.Element(3)
          Case "Number"
1106:             Set pNumValArray = New esriSystem.DoubleArray
1107:             pWorkOrderDictArray.Add pNumValArray                ' ELEMENT (2)
          Case "String"
1109:             Set pStrValArray = New esriSystem.strArray
1110:             pWorkOrderDictArray.Add pStrValArray                ' ELEMENT (2)
          Case "Date"
1112:             Set pDateValArray = New esriSystem.VarArray
1113:             pWorkOrderDictArray.Add pDateValArray               ' ELEMENT (2)
1114:         End Select

        ' ADD EMPTY ARRAY TO HOLD LENGTH (POLYLINE), AREA (POLYGON) OR COUNT (MULTIPOINT) VALUES
1117:         Set pSizeValArray = New esriSystem.DoubleArray
1118:         pWorkOrderDictArray.Add pSizeValArray                   ' ELEMENT (3)
```

```
1120:         booRequiresWeighting = pStatAndFieldArray.Element(5)
1121:         pWorkOrderDictArray.Add booRequiresWeighting            ' ELEMENT (4)
1122:         lngNumberNull = 0
1123:         pWorkOrderDictArray.Add lngNumberNull                   ' ELEMENT (5)
1124:         booRequiresStandardStats = pStatAndFieldArray.Element(6)
1125:         pWorkOrderDictArray.Add booRequiresStandardStats        ' ELEMENT (6)

1127:         pWorkOrderDictionary.AddObject pWorkOrderDictArray, CStr(lngFieldIndex), False
1128:       Else
1129:         Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(CStr(lngFieldIndex))
1130:         Set pStatInfoArray = pWorkOrderDictArray.Element(1)
1131:         pStatInfoArray.Add pStatAndFieldArray.Element(4)

        ' ONLY RESET ELEMENT 4 IF booRequiresWeighting = True
1134:         booRequiresWeighting = pStatAndFieldArray.Element(5)
1135:         If booRequiresWeighting Then pWorkOrderDictArray.Element(4) = True

        ' ONLY RESET ELEMENT 6 IF booRequiresStandardStats = True
1138:         booRequiresStandardStats = pStatAndFieldArray.Element(6)
1139:         If booRequiresStandardStats Then pWorkOrderDictArray.Element(6) = True
1140:       End If
1141:     Next anIndex

    ' FOR DEBUGGING
'    strReport = "Stats on Layer " & pFeatureLayer.Name & vbCrLf & _
'        "Requires Weighting = " & booRequiresWeighting & vbCrLf & _
'        pWorkOrderDictionary.Count & " fields selected: " & vbCrLf
'    Dim theKeys() As String
'    theKeys = pWorkOrderDictionary.ReturnKeys
'    Dim aKey As String
'    For anIndex = LBound(theKeys) To UBound(theKeys)
'      aKey = theKeys(anIndex)
'      Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
'      Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
'      Set pStatInfoArray = pWorkOrderDictArray.Element(1)
'      strReport = strReport & "  --> Field Name = " & pFieldInfoArray.Element(0) & vbCrLf & _
'                              "  --> Field Alias = " & pFieldInfoArray.Element(1) & vbCrLf & _
'                              "  --> Field Type = " & pFieldInfoArray.Element(2) & vbCrLf
'      For anIndex2 = 0 To pStatInfoArray.Count - 1
'        strReport = strReport & "     --> Requested Statistic #" & CStr(anIndex2 + 1) & " = " & _
'            pStatInfoArray.Element(anIndex2) & vbCrLf
'      Next anIndex2
'      strReport = strReport & "Field Requires Weighting = " & CStr(pWorkOrderDictArray.Element(4)) & vbCrLf
'      strReport = strReport & "------------------------" & vbCrLf
'    Next anIndex
'
'    MsgBox strReport
```

```
    ' END DEBUGGING -----------------

1169:     If booUseSelection Then
1170:       Set pTableSelection = pFeatureLayer
1171:       Set pSelectionSet = pTableSelection.SelectionSet
1172:       pSelectionSet.Search Nothing, True, pCursor
1173:       theValueCount = pSelectionSet.Count
1174:       Set pTable = pFeatureLayer
1175:       strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTotalRecszz", _
        Linkages.aml_func_mod.InsertCommas(CStr(pTable.RowCount(Nothing))))
1177:     Else
1178:       Set pTable = pFeatureLayer
1179:       Set pCursor = pTable.Search(Nothing, True)
1180:       theValueCount = pTable.RowCount(Nothing)
1181:       strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTotalRecszz", _
        Linkages.aml_func_mod.InsertCommas(CStr(theValueCount)))
1183:     End If

1185:     strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzRecsAnalyzedzz", _
        Linkages.aml_func_mod.InsertCommas(CStr(theValueCount)))


    ' FILL ARRAYS OF VALUES
1190:     theKeys = pWorkOrderDictionary.ReturnKeys

1192:     If (theValueCount = 0) Then                 ' NO RECORDS WILL BE EXAMINED; SHOULD BE VERY RARE

1194:       MsgBox "No records could be analyzed!  Possibly feature class has no records?  Bailing out..."


1197:       MsgBox "No records in '" & pFeatureLayer.Name & "' could be analyzed!  Possibly layer has no records? " & _
          "No statistics generated for this layer...", , "Problem with " & pFeatureLayer.Name & ":"
1199:       strReport = strReport & "\b    --> Unable to generate statistics for this layer...\b0\par" & vbCrLf

1201:     Else

1203:       Set pRow = pCursor.NextRow
1204:       theValueIndex = -1
1205:       theNumberNull = 0

1207:       Screen.MousePointer = vbHourglass

1209:       psbar.ShowProgressBar "Gathering values from " & pFeatureLayer.Name, 1, _
            theValueCount, 1, True

      ' PUT VALUES INTO ESRI ARRAYS ------------------------------------------------
```

```
      ' NEED TO GET FEATURES IF USING WEIGHTED VALUES
      ' NEED TO DIMENSION pPolygon, pArea, pGeometry, pPolyline
      ' CAN AVOID THIS IF USING POINTS; MAKE SURE CODE DOESN'T TRY TO GET SIZES IF USING POINTS

1218:       Do While Not pRow Is Nothing
1219:         For anIndex = LBound(theKeys) To UBound(theKeys)
1220:           aKey = theKeys(anIndex)
1221:           lngFieldIndex = CLng(aKey)
1222:           Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
1223:           Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
1224:           booRequiresWeighting = pWorkOrderDictArray.Element(4)
1225:           lngNumberNull = pWorkOrderDictArray.Element(5)
          Select Case pFieldInfoArray.Element(2)
            Case "Number"
1228:               dblValue = pRow.Value(lngFieldIndex)
1229:               If IsNull(dblValue) Then
1230:                 lngNumberNull = lngNumberNull + 1
1231:                 pWorkOrderDictArray.Element(5) = lngNumberNull
1232:               Else
1233:                 Set pNumValArray = pWorkOrderDictArray.Element(2)
1234:                 pNumValArray.Add dblValue

              ' IF REQUIRES WEIGHTING, GET SHAPE AND MEASURE IT.  SHOULD ONLY HAVE POLYGONS, POLYLINES AND MULTIPOINTS
1237:                 If booRequiresWeighting Then
1238:                   Set pSizeValArray = pWorkOrderDictArray.Element(3)
                  Select Case strShapeType
                    Case "Multipoint"
1241:                     Set pMultipoint = pRow.Value(lngShapeField)
1242:                     Set pPointCollection = pMultipoint
1243:                     dblSize = pPointCollection.PointCount
1244:                     pSizeValArray.Add dblSize
                    Case "Polygon"
1246:                     Set pPolygon = pRow.Value(lngShapeField)
1247:                     Set pArea = pPolygon
1248:                     dblSize = pArea.Area
1249:                     pSizeValArray.Add dblSize
                    Case "Polyline"
1251:                     Set pPolyline = pRow.Value(lngShapeField)
1252:                     dblSize = pPolyline.length
1253:                     pSizeValArray.Add dblSize
1254:                   End Select
1255:                 End If
1256:               End If

            Case "String"
1259:               strValue = pRow.Value(lngFieldIndex)
'             If strValue = "" Then
```

```
'                lngNumberNull = lngNumberNull + 1
'                pWorkOrderDictArray.Element(5) = lngNumberNull
'             Else
1264:                If strValue = "" Then strValue = "<-- EMPTY STRING -->"
1265:                Set pStrValArray = pWorkOrderDictArray.Element(2)
1266:                pStrValArray.Add strValue

             ' IF REQUIRES WEIGHTING, GET SHAPE AND MEASURE IT.  SHOULD ONLY HAVE POLYGONS, POLYLINES AND MULTIPOINTS
1269:                If booRequiresWeighting Then
1270:                  Set pSizeValArray = pWorkOrderDictArray.Element(3)
               Select Case strShapeType
                 Case "Multipoint"
1273:                    Set pMultipoint = pRow.Value(lngShapeField)
1274:                    Set pPointCollection = pMultipoint
1275:                    dblSize = pPointCollection.PointCount
1276:                    pSizeValArray.Add dblSize
                 Case "Polygon"
1278:                    Set pPolygon = pRow.Value(lngShapeField)
1279:                    Set pArea = pPolygon
1280:                    dblSize = pArea.Area
1281:                    pSizeValArray.Add dblSize
                 Case "Polyline"
1283:                    Set pPolyline = pRow.Value(lngShapeField)
1284:                    dblSize = pPolyline.length
1285:                    pSizeValArray.Add dblSize
1286:                End Select
'                End If
1288:                End If

           Case "Date"
1291:                dateValue = pRow.Value(lngFieldIndex)
1292:                If IsNull(dateValue) Then
1293:                  lngNumberNull = lngNumberNull + 1
1294:                  pWorkOrderDictArray.Element(5) = lngNumberNull
1295:                Else
1296:                  Set pDateValArray = pWorkOrderDictArray.Element(2)
1297:                  pDateValArray.Add dateValue
              ' NOTE:  NO WEIGHTING OPTIONS FOR DATE FIELDS
1299:                End If
1300:            End Select

1302:          Next anIndex

1304:          psbar.StepProgressBar
1305:          Set pRow = pCursor.NextRow
1306:        Loop
      '-------------------------------------------------------------------------------------------------------------
```

```
      ' FINALLY ACTUALLY CALCULATE STATISTICS
      '-----------------------------------
      '        KEY = STRING OF FIELD INDEX
      '    ELEMENT = VARIANT ARRAY WITH 7 ITEMS:
      '            1) STRING ARRAY WITH:
      '               a) FIELD NAME
      '               b) FIELD ALIAS
      '               c) FIELD TYPE (SHOULD BE "NUMBER", "STRING", OR "DATE")
      '            2) STRING ARRAY WITH ALL REQUESTED STATISTICS
      '            3) NUMBER, STRING OR DATE ARRAY FOR DATA VALUES, DEPENDING ON FIELD TYPE
      '            4) A NUMBER ARRAY FOR SHAPE SIZES (EACH FIELD GETS ITS OWN BECAUSE THEY MIGHT HAVE NULL VALUES)
      '            5) BOOLEAN INDICATING WHETHER SHAPE SIZE ARRAY SHOULD BE FILLED
      '            6) NUMBER (LONG) OF NULL VALUES FOUND IN THIS FIELD
      '            7) BOOLEAN INDICATING WHETHER STANDARD STATISTICS SHOULD BE CALCULATED

1323:       For anIndex = LBound(theKeys) To UBound(theKeys)
1324:         aKey = theKeys(anIndex)
1325:         lngFieldIndex = CLng(aKey)
1326:         Set pOrigField = pTableFields.Field(lngFieldIndex)
1327:         Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
1328:         Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
1329:         Set pStatInfoArray = pWorkOrderDictArray.Element(1)
1330:         Set pSizeValArray = pWorkOrderDictArray.Element(3)
1331:         booRequiresWeighting = pWorkOrderDictArray.Element(4)
1332:         lngNumberNull = pWorkOrderDictArray.Element(5)
1333:         booRequiresStandardStats = pWorkOrderDictArray.Element(6)


        Select Case pFieldInfoArray.Element(2)
          ' NEED TO DO DIFFERENT THINGS FOR NUMBERS, STRINGS AND DATES
          Case "Number"   ' ------------------------------------------------------------------------------------------

1340:             strReport = strReport & _
                 "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf &
_
                 "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                 "\b Data saved to \b0  zzTableSaveTozz\par" & vbCrLf & _
                 "\b Selected Statistics: \b0\par" & vbCrLf

           ' SET STAT VARIABLES


1349:             Set pNumValArray = pWorkOrderDictArray.Element(2)
1350:             If pNumValArray.Count = 0 Then                      ' NO STATS CALCULATED FOR THIS FIELD
1351:               strReport = strReport & "\b     !!! No Stats Calculated!  Apparently no non-null values found... \b0\par" & vbCrLf
1352:               strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTableSaveTozz", "[NO TABLE GENERATED]")
1353:             ElseIf pNumValArray.Count = 1 Then                  ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
```

```
1354:               theSum = pNumValArray.Element(0)
1355:               theMean = pNumValArray.Element(0)
1356:               theMinimum = pNumValArray.Element(0)
1357:               theMaximum = pNumValArray.Element(0)
1358:               theRange = 0
1359:               theCount = 1
1360:               theStdDev = 0
1361:               theVar = 0
1362:               theMedian = pNumValArray.Element(0)
1363:               theStdErrMean = 0
1364:               theModeString = "No Mode Found"
                  ' theFinalModes
1366:               booFoundMode = False
                  ReDim lngHistArray(0)
1368:               lngHistArray(0) = pNumValArray.Element(0)
1369:               theMeanWBL = pNumValArray.Element(0)
1370:               theStdDevWBL = 0
1371:               theVarWBL = 0

1373:             Else                                                    ' ACTUAL STATS
                  ' PUT NUMBERS IN DOUBLE ARRAY
                  ReDim dblNumValArray(pNumValArray.Count - 1)
                  If booRequiresWeighting Then ReDim dblSizeValArray(pNumValArray.Count - 1, 1)
1377:               For anIndex2 = 0 To pNumValArray.Count - 1
1378:                 dblNumValArray(anIndex2) = pNumValArray.Element(anIndex2)
1379:                 If booRequiresWeighting Then
1380:                   dblSizeValArray(anIndex2, 0) = pNumValArray.Element(anIndex2)
1381:                   dblSizeValArray(anIndex2, 1) = pSizeValArray.Element(anIndex2)
1382:                 End If
1383:               Next anIndex2
1384:               Set pStats = New esriSystem.VarArray
1385:               Set pWeightStats = New esriSystem.DoubleArray
1386:               If booRequiresStandardStats Then
1387:                 Call Linkages.QuickSort.DoubleAscending(dblNumValArray, 0, UBound(dblNumValArray))
1388:                 Set pStats = Linkages.MyGeneralOperations.BasicStatsFromArray( _
                    dblNumValArray, pFieldInfoArray.Element(1), pFeatureLayer.Name, m_App, lngNumBins)
1390:                 theSum = pStats.Element(0)
1391:                 theMean = pStats.Element(1)
1392:                 theMinimum = pStats.Element(2)
1393:                 theMaximum = pStats.Element(3)
1394:                 theRange = pStats.Element(4)
1395:                 theCount = pStats.Element(5)
1396:                 theStdDev = pStats.Element(6)
1397:                 theVar = pStats.Element(7)
1398:                 theMedian = pStats.Element(8)
1399:                 theStdErrMean = pStats.Element(9)
1400:                 theModeString = pStats.Element(10)
```

```
1401:                 theFinalModes = pStats.Element(11)
1402:                 booFoundMode = pStats.Element(12)
1403:                 lngHistArray = pStats.Element(13)
1404:                 If theMaximum > 100000 Then
1405:                   lngNumDecPlaces = 0
1406:                 ElseIf (theMaximum > 1000) And (theMaximum <= 100000) Then
1407:                   lngNumDecPlaces = 2
1408:                 ElseIf (theMaximum > 10) And (theMaximum <= 1000) Then
1409:                   lngNumDecPlaces = 4
1410:                 ElseIf theMaximum <= 10 Then
1411:                   lngNumDecPlaces = pOrigField.Scale
1412:                 End If
1413:               End If
1414:               If booRequiresWeighting Then
1415:                 Set pWeightStats = Linkages.MyGeneralOperations.BasicStatsFromArray_Weighted( _
                      dblSizeValArray, pFieldInfoArray.Element(1), pFeatureLayer.Name, m_App)
1417:                 theMeanWBL = pWeightStats.Element(0)
1418:                 theStDevWBL = pWeightStats.Element(1)
1419:                 theVarWBL = pWeightStats.Element(2)
1420:               End If
1421:             End If


            ' MAKE REPORT AND TABLES
1425:             If booRequiresStandardStats And booRequiresWeighting Then
1426:               strfilename = MakeNumberDBASETable(pOrigField, pStats, pStatInfoArray, pFieldInfoArray, pWeightStats)
1427:             ElseIf (Not booRequiresStandardStats) And booRequiresWeighting Then
1428:               strfilename = MakeNumberDBASETable(pOrigField, Nothing, pStatInfoArray, pFieldInfoArray, pWeightStats)
1429:             ElseIf (booRequiresStandardStats) And (Not booRequiresWeighting) Then
1430:               strfilename = MakeNumberDBASETable(pOrigField, pStats, pStatInfoArray, pFieldInfoArray, Nothing)
1431:             End If

1433:             strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTableSaveTozz", _
                  Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\"))

1436:             lngLetterInd = -1
1437:             For anIndex2 = 0 To pStatInfoArray.Count - 1
1438:               strStatName = pStatInfoArray.Element(anIndex2)
              Select Case strStatName
                Case "Minimum"
1441:                   lngLetterInd = lngLetterInd + 1
1442:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Minimum: \b0  " & CStr(theMinimum) & "\par" &
vbCrLf
                Case "Maximum"
1444:                   lngLetterInd = lngLetterInd + 1
1445:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Maximum: \b0  " & CStr(theMaximum) & "\par" &
vbCrLf
```

```
                Case "Mean"
1447:                   lngLetterInd = lngLetterInd + 1
1448:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean: \b0  " & CStr(theMean) & "\par" & vbCrLf
                Case "Sum"
1450:                   lngLetterInd = lngLetterInd + 1
1451:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Sum: \b0  " & CStr(theSum) & "\par" & vbCrLf
                Case "Standard Deviation"
1453:                   lngLetterInd = lngLetterInd + 1
1454:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation: \b0  " & CStr(theStdDev) &
"\par" & vbCrLf
                Case "Variance"
1456:                   lngLetterInd = lngLetterInd + 1
1457:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Variance: \b0  " & CStr(theVar) & "\par" &
vbCrLf
                Case "Mean_WBL"
1459:                   lngLetterInd = lngLetterInd + 1
1460:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean_WBL: \b0  " & CStr(theMeanWBL) & "\par" &
vbCrLf
                Case "Standard Deviation_WBL"
1462:                   lngLetterInd = lngLetterInd + 1
1463:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation_WBL: \b0  " &
CStr(theStDevWBL) & "\par" & vbCrLf
                Case "Variance_WBL"
1465:                   lngLetterInd = lngLetterInd + 1
1466:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Variance_WBL: \b0  " & CStr(theVarWBL) &
"\par" & vbCrLf
                Case "Mean_WBA"
1468:                   lngLetterInd = lngLetterInd + 1
1469:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean_WBA: \b0  " & CStr(theMeanWBL) & "\par" &
vbCrLf
                Case "Standard Deviation_WBA"
1471:                   lngLetterInd = lngLetterInd + 1
1472:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation_WBA: \b0  " &
CStr(theStDevWBL) & "\par" & vbCrLf
                Case "Variance_WBA"
1474:                   lngLetterInd = lngLetterInd + 1
1475:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Variance_WBA: \b0  " & CStr(theVarWBL) &
"\par" & vbCrLf
                Case "Histogram"
1477:                   lngLetterInd = lngLetterInd + 1
1478:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Histogram:\b0\par" & vbCrLf
1479:                   If pNumValArray.Count = 1 Then
1480:                     strReport = strReport & "        !!! Single Value:  No Histogram created...\par" & vbCrLf
1481:                   Else
1482:                     strHistReport = Linkages.CorridorAnalysisFunctions.MakeHistogramData(pFieldInfoArray, lngNumBins,
theMinimum, _
                   theMaximum, lngHistArray, m_App, txtOutput.Text, lngNumDecPlaces)
```

```
1484:                     strReport = strReport & strHistReport
1485:                   End If
1486:               End Select
1487:             Next anIndex2
1488:             strReport = strReport & "==================================================\par" & vbCrLf

          Case "String"   ' ------------------------------------------------------------------------------------

1492:             strReport = strReport & _
                "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf &
_
                "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                "\b Statistics by Unique Value: \b0\par" & vbCrLf
            ' SET STAT VARIABLES
1497:               Set pStrValArray = pWorkOrderDictArray.Element(2)
1498:               Set pClone = pOrigField
1499:               Set pNewStringField = pClone.Clone

1501:               If pStrValArray.Count = 0 Then                        ' NO STATS CALCULATED FOR THIS FIELD
1502:                 strReport = strReport & "\b    !!! No Stats Calculated!  Apparently no non-null values found...\b0\par" & vbCrLf
1503:               ElseIf pStrValArray.Count = 1 Then                    ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
1504:                 Set pStats = New esriSystem.VarArray
1505:                 Set pVarArray = New esriSystem.VarArray

1507:                 strValue = pStrValArray.Element(0)
1508:                 pVarArray.Add strValue
1509:                 theCount = 1
1510:                 pVarArray.Add theCount
1511:                 dblProportion = 1
1512:                 pVarArray.Add dblProportion
1513:                 If booRequiresWeighting Then
1514:                   dblSize = pSizeValArray.Element(0)
1515:                 Else
1516:                   dblSize = 1
1517:                 End If
1518:                 pVarArray.Add dblSize
1519:                 pStats.Add pVarArray

1521:                 strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField)
1522:                 strReport = strReport & strSubReport
1523:               Else                                                  ' ACTUAL STATS
              ' STRING STAT FUNCTION TAKES STRINGS IN ESRI STRING ARRAY

1526:                 If booRequiresWeighting Then
1527:                   Set pStats = Linkages.CorridorAnalysisFunctions.StatsForStrings(pStrValArray, pSizeValArray)
1528:                 Else
1529:                   Set pStats = Linkages.CorridorAnalysisFunctions.StatsForStrings(pStrValArray, Nothing)
```

```
1530:               End If

1532:               strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField)
1533:               strReport = strReport & strSubReport

1535:             End If

          Case "Date"   ' ----------------------------------------------------------------------------------------
1538:             Set pDateValArray = pWorkOrderDictArray.Element(2)

1540:             If pDateValArray.Count = 0 Then                         ' NO STATS CALCULATED FOR THIS FIELD
1541:               strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf
& _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b [NO TABLE GENERATED]\par" & vbCrLf & _
                  " !!! No Stats Calculated!  Apparently no non-null date values found...\par" & vbCrLf & _
                  "=====================================================\par" & vbCrLf
1547:             ElseIf pDateValArray.Count = 1 Then                     ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
1548:               dateValue = pDateValArray.Element(0)
1549:               strfilename = MakeDateTable(pFieldInfoArray, dateValue, dateValue)
1550:               strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf
& _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b Data saved to \b0  " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
                  "\b Statistics: \b0\par  " & vbCrLf & _
                  "     1] Earliest Date = " & CStr(dateValue) & "\par" & vbCrLf & _
                  "     2] Latest Date = " & CStr(dateValue) & "\par" & vbCrLf & _
                  "=====================================================\par" & vbCrLf


1560:             Else                                                  ' ACTUAL STATS
            ' STRING STAT FUNCTION TAKES STRINGS IN ESRI STRING ARRAY
1562:               Set pStats = Linkages.CorridorAnalysisFunctions.StatsForDates(pDateValArray)
1563:               strfilename = MakeDateTable(pFieldInfoArray, pStats.Element(0), pStats.Element(1))
1564:               strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf
& _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b Data saved to \b0  " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
                  "\b Statistics: \b0\par  " & vbCrLf & _
                  "     1] Earliest Date = " & CStr(pStats.Element(0)) & "\par" & vbCrLf & _
                  "     2] Latest Date = " & CStr(pStats.Element(1)) & "\par" & vbCrLf & _
                  "=====================================================\par" & vbCrLf

1573:             End If
```

```
1574:             strReport = strReport & strSubReport
1575:         End Select
1576:       Next anIndex
1577:     End If




1584:   ElseIf TypeOf m_TableFields Is IStandaloneTable Then

1586:     Set pStTable = m_TableFields
1587:     Set pTableFields = pStTable
1588:     Set pTable = pStTable.Table


'    strReport = "Statistics Report on Table '" & pStTable.Name & "':" & vbCrLf & _
       "-----------------------------------------------" & vbCrLf & _
       "Records Analyzed = zzRecsAnalyzedzz out of zzTotalRecszz" & vbCrLf & _
       "-----------------------------------------------" & vbCrLf

1596:     strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fprq2\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\tx90\tx360\tx450\tx720\b\f0\fs16 Statistics Report on Table '" & _
      pStTable.Name & "'\b0\par" & vbCrLf & _
      "\pard ----------------------------------------------------------------------------\par" & vbCrLf & _
      "\b Records Analyzed: \b0  zzRecsAnalyzedzz out of zzTotalRecszz\par" & vbCrLf & _
      "----------------------------------------------------------------------------\par" & vbCrLf

1604:     For anIndex = 1 To pNodes.Count     ' CHECK ALL NODES IN TREEVIEW
1605:       Set pNode = pNodes.Item(anIndex)
1606:       If pNode.Image = 2 Then           ' THEN THIS STAT IS SELECTED
1607:         Set pParentNode = pNode.Parent
1608:         strParentKey = pParentNode.Key

        ' MAKE EMPTY FIELD INFO ARRAY
1611:         Set pStatAndFieldArray = New esriSystem.VarArray

        ' STAT AND FIELD ARRAY HAS VALUES FOR EACH SELECTED NODE IN TREE VIEW.  VALUES ARE:
        ' (0) Field Index
        ' (1) Field Name
        ' (2) Field Alias
        ' (3) Field Type (Number, String, Date)
        ' (4) Name of Statistic

        ' GET INFORMATION ON FIELD AND ADD TO FIELD INFO ARRAY
```

```
1621:         lngParentIndex = CLng(Right(strParentKey, Len(strParentKey) - 4))
1622:         strParentFieldName = pTableFields.Field(lngParentIndex).Name
1623:         strParentFieldAlias = pTableFields.FieldInfo(lngParentIndex).Alias
1624:         intParentFieldType = pTableFields.Field(lngParentIndex).Type
        Select Case intParentFieldType
          Case Is < 4
1627:             strParentFieldType = "Number"
          Case 4
1629:             strParentFieldType = "String"
          Case 5
1631:             strParentFieldType = "Date"
          Case Else
1633:             strParentFieldType = "Unknown"
1634:         End Select

1636:         pStatAndFieldArray.Add lngParentIndex
1637:         pStatAndFieldArray.Add strParentFieldName
1638:         pStatAndFieldArray.Add strParentFieldAlias
1639:         pStatAndFieldArray.Add strParentFieldType
1640:         strStatName = pNode.Text
1641:         pStatAndFieldArray.Add strStatName

        ' ADD STAT ARRAY TO WORK ORDER ARRAY
1644:         pWorkOrderArray.Add pStatAndFieldArray
1645:       End If     ' END CHECKING IF THIS NODE IS SELECTED
1646:     Next anIndex

    ' TURN THIS INTO A DICTIONARY (COLLECTION MOD) WITH:
    '        KEY = STRING OF FIELD INDEX
    '    ELEMENT = VARIANT ARRAY WITH 4 ITEMS:
    '             1) STRING ARRAY WITH:
    '                a) FIELD NAME
    '                b) FIELD ALIAS
    '                c) FIELD TYPE (SHOULD BE "NUMBER", "STRING", OR "DATE")
    '             2) STRING ARRAY WITH ALL REQUESTED STATISTICS
    '             3) NUMBER, STRING OR DATE ARRAY FOR DATA VALUES, DEPENDING ON FIELD TYPE
    '             4) NUMBER (LONG) OF NULL VALUES FOUND IN THIS FIELD

1659:     Set pWorkOrderDictionary = New Linkages.CollectionMod
1660:     For anIndex = 0 To pWorkOrderArray.Count - 1
1661:       Set pStatAndFieldArray = pWorkOrderArray.Element(anIndex)
1662:       lngFieldIndex = pStatAndFieldArray.Element(0)
1663:       booHasKey = pWorkOrderDictionary.HasKey(CStr(lngFieldIndex))
   '   MsgBox "Index = " & CStr(anIndex) & vbCrLf & _
             "Field Name = " & pStatAndFieldArray.Element(1) & vbCrLf & _
             "Statistic = " & pStatAndFieldArray.Element(4) & vbCrLf & _
             "Already Found Key = " & CStr(booHasKey)
```

```
1668:       If Not booHasKey Then
1669:         Set pFieldInfoArray = New esriSystem.strArray
1670:         pFieldInfoArray.Add pStatAndFieldArray.Element(1)       ' FIELD NAME
1671:         pFieldInfoArray.Add pStatAndFieldArray.Element(2)       ' FIELD ALIAS
1672:         pFieldInfoArray.Add pStatAndFieldArray.Element(3)       ' FIELD TYPE ("Number", "String", "Date")

1674:         Set pStatInfoArray = New esriSystem.strArray
      '  MsgBox pStatAndFieldArray.Element(4)
1676:         pStatInfoArray.Add pStatAndFieldArray.Element(4)

1678:         Set pWorkOrderDictArray = New esriSystem.VarArray
1679:         pWorkOrderDictArray.Add pFieldInfoArray                 ' ELEMENT (0)
1680:         pWorkOrderDictArray.Add pStatInfoArray                  ' ELEMENT (1)

        ' ADD EMPTY ARRAY TO HOLD VALUES
        Select Case pStatAndFieldArray.Element(3)
          Case "Number"
1685:             Set pNumValArray = New esriSystem.DoubleArray
1686:             pWorkOrderDictArray.Add pNumValArray                ' ELEMENT (2)
          Case "String"
1688:             Set pStrValArray = New esriSystem.strArray
1689:             pWorkOrderDictArray.Add pStrValArray                ' ELEMENT (2)
          Case "Date"
1691:             Set pDateValArray = New esriSystem.VarArray
1692:             pWorkOrderDictArray.Add pDateValArray               ' ELEMENT (2)
1693:         End Select

1695:         lngNumberNull = 0
1696:         pWorkOrderDictArray.Add lngNumberNull                   ' ELEMENT (3)

1698:         pWorkOrderDictionary.AddObject pWorkOrderDictArray, CStr(lngFieldIndex), False
1699:       Else
1700:         Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(CStr(lngFieldIndex))
1701:         Set pStatInfoArray = pWorkOrderDictArray.Element(1)
1702:         pStatInfoArray.Add pStatAndFieldArray.Element(4)
1703:       End If
1704:     Next anIndex

1706:     If booUseSelection Then
1707:       Set pTableSelection = pStTable
1708:       Set pSelectionSet = pTableSelection.SelectionSet
1709:       pSelectionSet.Search Nothing, True, pCursor
1710:       theValueCount = pSelectionSet.Count
1711:       Set pTable = pStTable.Table
1712:       strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTotalRecszz", _
        Linkages.aml_func_mod.InsertCommas(CStr(pTable.RowCount(Nothing))))
1714:     Else
```

```
1715:       Set pTable = pStTable.Table
1716:       Set pCursor = pTable.Search(Nothing, True)
1717:       theValueCount = pTable.RowCount(Nothing)
1718:       strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTotalRecszz", _
        Linkages.aml_func_mod.InsertCommas(CStr(theValueCount)))
1720:     End If

1722:     strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzRecsAnalyzedzz", _
        Linkages.aml_func_mod.InsertCommas(CStr(theValueCount)))

    ' FILL ARRAYS OF VALUES
1726:     theKeys = pWorkOrderDictionary.ReturnKeys

1728:     If (theValueCount = 0) Then                 ' NO RECORDS WILL BE EXAMINED; SHOULD BE VERY RARE

1730:       MsgBox "No records in '" & pStTable.Name & "' could be analyzed!  Possibly table has no records? " & _
          "No statistics generated for this table...", , "Problem with " & pStTable.Name & ":"
1732:       strReport = strReport & "   --> Unable to generate statistics for this table...\par" & vbCrLf

1734:     Else

1736:       Set pRow = pCursor.NextRow
1737:       theValueIndex = -1
1738:       theNumberNull = 0

1740:       Screen.MousePointer = vbHourglass

1742:       psbar.ShowProgressBar "Gathering values from " & pStTable.Name, 1, _
            theValueCount, 1, True

      ' PUT VALUES INTO ESRI ARRAYS ------------------------------------------------

1747:       Do While Not pRow Is Nothing
1748:         For anIndex = LBound(theKeys) To UBound(theKeys)
1749:           aKey = theKeys(anIndex)
1750:           lngFieldIndex = CLng(aKey)
1751:           Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
1752:           Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
1753:           lngNumberNull = pWorkOrderDictArray.Element(3)

          Select Case pFieldInfoArray.Element(2)

            Case "Number"
1758:               dblValue = pRow.Value(lngFieldIndex)
1759:               If IsNull(dblValue) Then
1760:                 lngNumberNull = lngNumberNull + 1
1761:                 pWorkOrderDictArray.Element(5) = lngNumberNull
```

```
1762:               Else
1763:                 Set pNumValArray = pWorkOrderDictArray.Element(2)
1764:                 pNumValArray.Add dblValue
1765:               End If

            Case "String"
1768:               strValue = pRow.Value(lngFieldIndex)
1769:               If strValue = "" Then strValue = "<-- EMPTY STRING -->"
1770:               Set pStrValArray = pWorkOrderDictArray.Element(2)
1771:               pStrValArray.Add strValue

            Case "Date"
1774:               dateValue = pRow.Value(lngFieldIndex)
1775:               If IsNull(dateValue) Then
1776:                 lngNumberNull = lngNumberNull + 1
1777:                 pWorkOrderDictArray.Element(5) = lngNumberNull
1778:               Else
1779:                 Set pDateValArray = pWorkOrderDictArray.Element(2)
1780:                 pDateValArray.Add dateValue
1781:               End If
1782:            End Select

1784:         Next anIndex

1786:         psbar.StepProgressBar
1787:         Set pRow = pCursor.NextRow
1788:       Loop
      '-------------------------------------------------------------------------------------------------------
      ' FINALLY ACTUALLY CALCULATE STATISTICS
      '-------------------------------------
      '        KEY = STRING OF FIELD INDEX
      '    ELEMENT = VARIANT ARRAY WITH 4 ITEMS:
      '            1) STRING ARRAY WITH:
      '               a) FIELD NAME
      '               b) FIELD ALIAS
      '               c) FIELD TYPE (SHOULD BE "NUMBER", "STRING", OR "DATE")
      '            2) STRING ARRAY WITH ALL REQUESTED STATISTICS
      '            3) NUMBER, STRING OR DATE ARRAY FOR DATA VALUES, DEPENDING ON FIELD TYPE
      '            4) NUMBER (LONG) OF NULL VALUES FOUND IN THIS FIELD

1802:       For anIndex = LBound(theKeys) To UBound(theKeys)
1803:         aKey = theKeys(anIndex)
1804:         lngFieldIndex = CLng(aKey)
1805:         Set pOrigField = pTableFields.Field(lngFieldIndex)
1806:         Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
1807:         Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
1808:         Set pStatInfoArray = pWorkOrderDictArray.Element(1)
```

```
1809:         lngNumberNull = pWorkOrderDictArray.Element(3)

        Select Case pFieldInfoArray.Element(2)
          ' NEED TO DO DIFFERENT THINGS FOR NUMBERS, STRINGS AND DATES
          Case "Number"   ' ------------------------------------------------------------------------------------------

1815:             strReport = strReport & _
                "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf &
_
                "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                "\b Data saved to zzTableSaveTozz\par" & vbCrLf & _
                "\b Selected Statistics: \b0\par  " & vbCrLf

            ' SET STAT VARIABLES
1822:               Set pNumValArray = pWorkOrderDictArray.Element(2)
1823:               If pNumValArray.Count = 0 Then                        ' NO STATS CALCULATED FOR THIS FIELD
1824:                 strReport = strReport & "    !!! No Stats Calculated!  Apparently no non-null values found...\par" & vbCrLf
1825:                 strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTableSaveTozz", "[NO TABLE GENERATED]")
1826:               ElseIf pNumValArray.Count = 1 Then                    ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
1827:                 theSum = pNumValArray.Element(0)
1828:                 theMean = pNumValArray.Element(0)
1829:                 theMinimum = pNumValArray.Element(0)
1830:                 theMaximum = pNumValArray.Element(0)
1831:                 theRange = 0
1832:                 theCount = 1
1833:                 theStdDev = 0
1834:                 theVar = 0
1835:                 theMedian = pNumValArray.Element(0)
1836:                 theStdErrMean = 0
1837:                 theModeString = "No Mode Found"
            ' theFinalModes
1839:               booFoundMode = False
            ReDim lngHistArray(0)
1841:               lngHistArray(0) = pNumValArray.Element(0)
1842:               theMeanWBL = pNumValArray.Element(0)
1843:               theStDevWBL = 0
1844:               theVarWBL = 0

1846:             Else                                                    ' ACTUAL STATS
            ' PUT NUMBERS IN DOUBLE ARRAY
            ReDim dblNumValArray(pNumValArray.Count - 1)
1849:               For anIndex2 = 0 To pNumValArray.Count - 1
1850:                 dblNumValArray(anIndex2) = pNumValArray.Element(anIndex2)
1851:               Next anIndex2
1852:               Set pStats = New esriSystem.VarArray

1854:               Call Linkages.QuickSort.DoubleAscending(dblNumValArray, 0, UBound(dblNumValArray))
```

```
1855:               Set pStats = Linkages.MyGeneralOperations.BasicStatsFromArray( _
                  dblNumValArray, pFieldInfoArray.Element(1), pStTable.Name, m_App, lngNumBins)
1857:               theSum = pStats.Element(0)
1858:               theMean = pStats.Element(1)
1859:               theMinimum = pStats.Element(2)
1860:               theMaximum = pStats.Element(3)
1861:               theRange = pStats.Element(4)
1862:               theCount = pStats.Element(5)
1863:               theStdDev = pStats.Element(6)
1864:               theVar = pStats.Element(7)
1865:               theMedian = pStats.Element(8)
1866:               theStdErrMean = pStats.Element(9)
1867:               theModeString = pStats.Element(10)
1868:               theFinalModes = pStats.Element(11)
1869:               booFoundMode = pStats.Element(12)
1870:               lngHistArray = pStats.Element(13)
1871:               If theMaximum > 100000 Then
1872:                 lngNumDecPlaces = 0
1873:               ElseIf (theMaximum > 1000) And (theMaximum <= 100000) Then
1874:                 lngNumDecPlaces = 2
1875:               ElseIf (theMaximum > 10) And (theMaximum <= 1000) Then
1876:                 lngNumDecPlaces = 4
1877:               ElseIf theMaximum <= 10 Then
1878:                 lngNumDecPlaces = pOrigField.Scale
1879:               End If

1881:             End If

            ' MAKE REPORT AND TABLES
1884:             strfilename = MakeNumberDBASETable(pOrigField, pStats, pStatInfoArray, pFieldInfoArray, Nothing)
1885:             strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTableSaveTozz", _
                    Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\"))
1887:             lngLetterInd = -1

1889:             For anIndex2 = 0 To pStatInfoArray.Count - 1
1890:               strStatName = pStatInfoArray.Element(anIndex2)
              Select Case strStatName
                Case "Minimum"
1893:                   lngLetterInd = lngLetterInd + 1
1894:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Minimum: \b0  " & CStr(theMinimum) & "\par" &
vbCrLf
                Case "Maximum"
1896:                   lngLetterInd = lngLetterInd + 1
1897:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Maximum: \b0  " & CStr(theMaximum) & "\par" &
vbCrLf
                Case "Mean"
1899:                   lngLetterInd = lngLetterInd + 1
```

```
1900:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean: \b0  " & CStr(theMean) & "\par" & vbCrLf
                Case "Sum"
1902:                   lngLetterInd = lngLetterInd + 1
1903:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Sum: \b0  " & CStr(theSum) & "\par" & vbCrLf
                Case "Standard Deviation"
1905:                   lngLetterInd = lngLetterInd + 1
1906:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation: \b0  " & CStr(theStdDev) &
"\par" & vbCrLf
                Case "Variance"
1908:                   lngLetterInd = lngLetterInd + 1
1909:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Variance: \b0  " & CStr(theVar) & "\par" &
vbCrLf
                Case "Mean_WBL"
1911:                   lngLetterInd = lngLetterInd + 1
1912:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean_WBL: \b0  " & CStr(theMeanWBL) & "\par" &
vbCrLf
                Case "Standard Deviation_WBL"
1914:                   lngLetterInd = lngLetterInd + 1
1915:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation_WBL: \b0  " &
CStr(theStDevWBL) & "\par" & vbCrLf
                Case "Variance_WBL"
1917:                   lngLetterInd = lngLetterInd + 1
1918:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Variance_WBL: \b0  " & CStr(theVarWBL) &
"\par" & vbCrLf
                Case "Mean_WBA"
1920:                   lngLetterInd = lngLetterInd + 1
1921:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean_WBA: \b0  " & CStr(theMeanWBL) & "\par" &
vbCrLf
                Case "Standard Deviation_WBA"
1923:                   lngLetterInd = lngLetterInd + 1
1924:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation_WBA: \b0  " &
CStr(theStDevWBL) & "\par" & vbCrLf
                Case "Variance_WBA"
1926:                   lngLetterInd = lngLetterInd + 1
1927:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Variance_WBA: \b0  " & CStr(theVarWBL) &
"\par" & vbCrLf
                Case "Histogram"
1929:                   lngLetterInd = lngLetterInd + 1
1930:                   strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Histogram: \b0\par  " & vbCrLf
1931:                   If pNumValArray.Count = 1 Then
1932:                     strReport = strReport & "        !!! Single Value:  No Histogram created...\par" & vbCrLf
1933:                   Else
1934:                     strHistReport = Linkages.CorridorAnalysisFunctions.MakeHistogramData(pFieldInfoArray, lngNumBins,
theMinimum, _
                       theMaximum, lngHistArray, m_App, txtOutput.Text, lngNumDecPlaces)
1936:                     strReport = strReport & strHistReport
1937:                   End If
```

```
1938:               End Select
1939:             Next anIndex2
1940:             strReport = strReport & "==================================================\par" & vbCrLf

          Case "String"   ' ------------------------------------------------------------------------------------------

1944:             strReport = strReport & _
                "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf &
_
                "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                "\b Statistics by Unique Value: \b0\par" & vbCrLf
            ' SET STAT VARIABLES
1949:             Set pStrValArray = pWorkOrderDictArray.Element(2)
1950:             Set pClone = pOrigField
1951:             Set pNewStringField = pClone.Clone

1953:             If pStrValArray.Count = 0 Then                        ' NO STATS CALCULATED FOR THIS FIELD
1954:               strReport = strReport & "    !!! No Stats Calculated!  Apparently no non-null values found...\par" & vbCrLf
1955:             ElseIf pStrValArray.Count = 1 Then                    ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
1956:               Set pStats = New esriSystem.VarArray
1957:               Set pVarArray = New esriSystem.VarArray

1959:               strValue = pStrValArray.Element(0)
1960:               pVarArray.Add strValue
1961:               theCount = 1
1962:               pVarArray.Add theCount
1963:               dblProportion = 1
1964:               pVarArray.Add dblProportion
1965:               dblSize = 1
1966:               pVarArray.Add dblSize
1967:               pStats.Add pVarArray

1969:               strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField)
1970:               strReport = strReport & strSubReport
1971:             Else                                                  ' ACTUAL STATS
              ' STRING STAT FUNCTION TAKES STRINGS IN ESRI STRING ARRAY

1974:               Set pStats = Linkages.CorridorAnalysisFunctions.StatsForStrings(pStrValArray, Nothing)
1975:               strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField)
1976:               strReport = strReport & strSubReport

1978:             End If

          Case "Date"   ' --------------------------------------------------------------------------------------------
1981:             Set pDateValArray = pWorkOrderDictArray.Element(2)

1983:             If pDateValArray.Count = 0 Then                       ' NO STATS CALCULATED FOR THIS FIELD
```

```
1984:               strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]" & vbCrLf & _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & vbCrLf & _
                  "   [NO TABLE GENERATED]\par" & vbCrLf & _
                  "   !!! No Stats Calculated!  Apparently no non-null date values found...\par" & vbCrLf & _
                    "==================================================\par" & vbCrLf

1991:               ElseIf pDateValArray.Count = 1 Then                    ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
1992:                 dateValue = pDateValArray.Element(0)
1993:                 strfilename = MakeDateTable(pFieldInfoArray, dateValue, dateValue)
1994:                 strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf
& _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b Data saved to \b0 " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
                  "\b Statistics: \b0\par  " & vbCrLf & _
                  "   1] Earliest Date = " & CStr(dateValue) & "\par" & vbCrLf & _
                  "   2] Latest Date = " & CStr(dateValue) & "\par" & vbCrLf & _
                  "==================================================\par" & vbCrLf


2004:               Else                                                  ' ACTUAL STATS
            ' STRING STAT FUNCTION TAKES STRINGS IN ESRI STRING ARRAY
2006:                 Set pStats = Linkages.CorridorAnalysisFunctions.StatsForDates(pDateValArray)
2007:                 strfilename = MakeDateTable(pFieldInfoArray, pStats.Element(0), pStats.Element(1))
2008:                 strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf
& _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b Data saved to\b0 " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
                  "\b Statistics: \b0\par" & vbCrLf & _
                  "   1] Earliest Date = " & CStr(pStats.Element(0)) & "\par" & vbCrLf & _
                  "   2] Latest Date = " & CStr(pStats.Element(1)) & "\par" & vbCrLf & _
                  "==================================================\par" & vbCrLf

2017:               End If
2018:               strReport = strReport & strSubReport
2019:           End Select
2020:         Next anIndex
2021:       End If




2028:   ElseIf TypeOf m_TableFields Is IRasterLayer Then
```

```
2030:     Set pRasterLayer = m_TableFields
2031:     Set pRaster = pRasterLayer.Raster
2032:     Set pRasterBandCollection = pRaster
2033:     Set pRasterBand = pRasterBandCollection.Item(0)
2034:     Set pRasterStatistics = pRasterBand.Statistics
2035:     Set pRasterDataset = pRasterBand.RasterDataset
2036:     pRasterBand.HasTable booHasTable

    ' GET CELL SIZE
2039:     dblCellSize = (Linkages.GridFunctions.ReturnCellSize(pRaster)) ^ 2

2041:     If booHasTable Then                    ' USE VAT TABLE FIELDS

2043:       Set pRasterProps = pRasterBand
2044:       lngNoDataValue = pRasterProps.NoDataValue
2045:       If IsNumeric(lngNoDataValue) Then
2046:         lngNoDataValue = CLng(varNoDataValue)
2047:       Else
2048:         lngNoDataValue = -999999
2049:       End If
2050:       Set pTable = pRasterBand.AttributeTable
2051:       Set pTableFields = pRasterLayer

      ' GET GENERAL COUNT OF NON-NULL CELL VALUES
2054:       lngCountValField = pTable.FindField("Value")
2055:       lngCountCountField = pTable.FindField("Count")
2056:       lngCountCountValue = 0
2057:       Set pCountCursor = pTable.Search(Nothing, True)
2058:       Set pCountRow = pCountCursor.NextRow
2059:       Do Until pCountRow Is Nothing
2060:         lngCountValValue = pCountRow.Value(lngCountValField)
2061:         If lngCountValValue <> lngNoDataValue Then
2062:           lngCountCountValue = lngCountCountValue + pCountRow.Value(lngCountCountField)
2063:         End If
2064:         Set pCountRow = pCountCursor.NextRow
2065:       Loop

'      strReport = "Statistics Report on Grid '" & pRasterLayer.Name & "':" & vbCrLf & _
          "-------------------------------------------------" & vbCrLf & _
          "Non-Null Grid Cell Count = " & Linkages.aml_func_mod.InsertCommas(lngCountCountValue) & _
          " grid cells" & vbCrLf & _
          "-------------------------------------------------" & vbCrLf

2073:       strReport = _
        "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fprq2\fcharset0 Arial;}}" & vbCrLf & _
        "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\tx90\tx360\tx450\tx720\b\f0\fs16 Statistics Report on Grid '" & _
```

```
        pRasterLayer.Name & "'\b0\par" & vbCrLf & _
        "\pard ----------------------------------------------------------------------------\par" & vbCrLf & _
        "\b Non-Null Grid Cell Count: \b0 " & Linkages.aml_func_mod.InsertCommas(lngCountCountValue) & _
          " grid cells\par" & vbCrLf & _
        "----------------------------------------------------------------------------\par" & vbCrLf


2083:       For anIndex = 1 To pNodes.Count     ' CHECK ALL NODES IN TREEVIEW
2084:         Set pNode = pNodes.Item(anIndex)
2085:         If pNode.Image = 2 Then           ' THEN THIS STAT IS SELECTED
2086:           Set pParentNode = pNode.Parent
2087:           strParentKey = pParentNode.Key

          ' MAKE EMPTY FIELD INFO ARRAY
2090:           Set pStatAndFieldArray = New esriSystem.VarArray

          ' STAT AND FIELD ARRAY HAS VALUES FOR EACH SELECTED NODE IN TREE VIEW.  VALUES ARE:
          ' (0) Field Index
          ' (1) Field Name
          ' (2) Field Alias
          ' (3) Field Type (Number, String, Date)
          ' (4) Name of Statistic

          ' GET INFORMATION ON FIELD AND ADD TO FIELD INFO ARRAY
2100:           lngParentIndex = CLng(Right(strParentKey, Len(strParentKey) - 4))
2101:           strParentFieldName = pTableFields.Field(lngParentIndex).Name
2102:           strParentFieldAlias = pTableFields.FieldInfo(lngParentIndex).Alias
2103:           intParentFieldType = pTableFields.Field(lngParentIndex).Type
          Select Case intParentFieldType
            Case Is < 4
2106:               strParentFieldType = "Number"
            Case 4
2108:               strParentFieldType = "String"
            Case 5
2110:               strParentFieldType = "Date"
            Case Else
2112:               strParentFieldType = "Unknown"
2113:           End Select

2115:           pStatAndFieldArray.Add lngParentIndex
2116:           pStatAndFieldArray.Add strParentFieldName
2117:           pStatAndFieldArray.Add strParentFieldAlias
2118:           pStatAndFieldArray.Add strParentFieldType
2119:           strStatName = pNode.Text
2120:           pStatAndFieldArray.Add strStatName

          ' ADD STAT ARRAY TO WORK ORDER ARRAY
```

```
2123:           pWorkOrderArray.Add pStatAndFieldArray
2124:         End If     ' END CHECKING IF THIS NODE IS SELECTED
2125:       Next anIndex

    ' TURN THIS INTO A DICTIONARY (COLLECTION MOD) WITH:
    '        KEY = STRING OF FIELD INDEX
    '    ELEMENT = VARIANT ARRAY WITH 8 ITEMS:
    '            1) STRING ARRAY WITH:
    '               a) FIELD NAME
    '               b) FIELD ALIAS
    '               c) FIELD TYPE (SHOULD BE "NUMBER", "STRING", OR "DATE")
    '            2) STRING ARRAY WITH ALL REQUESTED STATISTICS
    '            3) NUMBER, STRING OR DATE ARRAY FOR DATA VALUES, DEPENDING ON FIELD TYPE
    '            4) NUMBER (LONG) OF NULL VALUES FOUND IN THIS FIELD
    '            5) A NUMBER ARRAY FOR COUNTS OF EACH UNIQUE CELL VALUE (EACH FIELD GETS ITS OWN BECAUSE THEY MIGHT HAVE NULL VALUES)
    '            6) BOOLEAN INDICATING WHETHER HISTOGRAM SHOULD BE GENERATED
    '            7) BOOLEAN INDICATING WHETHER CATEGORICAL DATA SHOULD BE GENERATED
    '            8) BOOLEAN INDICATING WHETHER NUMERIC STATISTICS SHOULD BE GENERATED

2142:       Set pWorkOrderDictionary = New Linkages.CollectionMod
2143:       For anIndex = 0 To pWorkOrderArray.Count - 1
2144:         Set pStatAndFieldArray = pWorkOrderArray.Element(anIndex)
2145:         lngFieldIndex = pStatAndFieldArray.Element(0)
2146:         booHasKey = pWorkOrderDictionary.HasKey(CStr(lngFieldIndex))
     '   MsgBox "Index = " & CStr(anIndex) & vbCrLf & _
              "Field Name = " & pStatAndFieldArray.Element(1) & vbCrLf & _
              "Statistic = " & pStatAndFieldArray.Element(4) & vbCrLf & _
              "Already Found Key = " & CStr(booHasKey)
2151:         If Not booHasKey Then
2152:           Set pFieldInfoArray = New esriSystem.strArray
2153:           pFieldInfoArray.Add pStatAndFieldArray.Element(1)       ' FIELD NAME
2154:           pFieldInfoArray.Add pStatAndFieldArray.Element(2)       ' FIELD ALIAS
2155:           pFieldInfoArray.Add pStatAndFieldArray.Element(3)       ' FIELD TYPE ("Number", "String", "Date")

2157:           Set pStatInfoArray = New esriSystem.strArray
        '  MsgBox pStatAndFieldArray.Element(4)
2159:           pStatInfoArray.Add pStatAndFieldArray.Element(4)

2161:           Set pWorkOrderDictArray = New esriSystem.VarArray
2162:           pWorkOrderDictArray.Add pFieldInfoArray                 ' ELEMENT (0)
2163:           pWorkOrderDictArray.Add pStatInfoArray                  ' ELEMENT (1)

          ' ADD EMPTY ARRAY TO HOLD VALUES
          Select Case pStatAndFieldArray.Element(3)
            Case "Number"
2168:               Set pNumValArray = New esriSystem.DoubleArray
2169:               pWorkOrderDictArray.Add pNumValArray                ' ELEMENT (2)
```

```
          Case "String"
2171:             Set pStrValArray = New esriSystem.strArray
2172:             pWorkOrderDictArray.Add pStrValArray                ' ELEMENT (2)
          Case "Date"
2174:             Set pDateValArray = New esriSystem.VarArray
2175:             pWorkOrderDictArray.Add pDateValArray               ' ELEMENT (2)
2176:         End Select

2178:         lngNumberNull = 0
2179:         pWorkOrderDictArray.Add lngNumberNull                   ' ELEMENT (3)

        ' ADD EMPTY ARRAY TO HOLD LENGTH (POLYLINE), AREA (POLYGON) OR COUNT (MULTIPOINT) VALUES
2182:         Set pSizeValArray = New esriSystem.DoubleArray
2183:         pWorkOrderDictArray.Add pSizeValArray                   ' ELEMENT (4)

2185:         booGridHistogram = pStatAndFieldArray.Element(4) = "Histogram"
2186:         pWorkOrderDictArray.Add booGridHistogram                ' ELEMENT (5)

2188:         booGridCategorical = (pStatAndFieldArray.Element(4) = "Count") Or _
                                   (pStatAndFieldArray.Element(4) = "Proportion") Or _
                                   (pStatAndFieldArray.Element(4) = "Area")
2191:         pWorkOrderDictArray.Add booGridCategorical              ' ELEMENT (6)

2193:         booGridNumeric = (pStatAndFieldArray.Element(4) = "Minimum") Or _
                               (pStatAndFieldArray.Element(4) = "Maximum") Or _
                               (pStatAndFieldArray.Element(4) = "Mean") Or _
                               (pStatAndFieldArray.Element(4) = "Median") Or _
                               (pStatAndFieldArray.Element(4) = "Mode") Or _
                               (pStatAndFieldArray.Element(4) = "Sum") Or _
                               (pStatAndFieldArray.Element(4) = "Standard Deviation") Or _
                               (pStatAndFieldArray.Element(4) = "Histogram")
2201:         pWorkOrderDictArray.Add booGridNumeric                  ' ELEMENT (7)

2203:         pWorkOrderDictionary.AddObject pWorkOrderDictArray, CStr(lngFieldIndex), False
2204:       Else
2205:         Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(CStr(lngFieldIndex))
2206:         Set pStatInfoArray = pWorkOrderDictArray.Element(1)
2207:         pStatInfoArray.Add pStatAndFieldArray.Element(4)
2208:         If pStatAndFieldArray.Element(4) = "Histogram" Then pWorkOrderDictArray.Element(5) = True
2209:         If (pStatAndFieldArray.Element(4) = "Count") Or _
                                   (pStatAndFieldArray.Element(4) = "Proportion") Or _
                                   (pStatAndFieldArray.Element(4) = "Area") Then pWorkOrderDictArray.Element(6) = True
2212:         If (pStatAndFieldArray.Element(4) = "Minimum") Or _
                               (pStatAndFieldArray.Element(4) = "Maximum") Or _
                               (pStatAndFieldArray.Element(4) = "Mean") Or _
                               (pStatAndFieldArray.Element(4) = "Median") Or _
                               (pStatAndFieldArray.Element(4) = "Mode") Or _
```

```
                        (pStatAndFieldArray.Element(4) = "Sum") Or _
                        (pStatAndFieldArray.Element(4) = "Standard Deviation") Or _
                        (pStatAndFieldArray.Element(4) = "Histogram") Then pWorkOrderDictArray.Element(7) = True
2220:         End If
2221:       Next anIndex

      ' ALWAYS WORK THROUGH ALL RECORDS IN TABLE
2224:       Set pCursor = pTable.Search(Nothing, True)
2225:       lngCountCountField = pTable.FindField("Count")

      ' FILL ARRAYS OF VALUES
2228:       theKeys = pWorkOrderDictionary.ReturnKeys

2230:       If (pTable.RowCount(Nothing) = 0) Then                  ' NO RECORDS WILL BE EXAMINED; SHOULD BE VERY RARE

2232:         MsgBox "No records in the '" & pRasterLayer.Name & "' attribute table could be analyzed!  Possibly table has no records?
" & _
            "No statistics generated for this table...", , "Problem with " & pRasterLayer.Name & ":"
2234:         strReport = strReport & "   --> Unable to generate statistics for this dataset...\par" & vbCrLf

2236:       Else

2238:         Set pRow = pCursor.NextRow
2239:         theValueIndex = -1
2240:         theNumberNull = 0

2242:         Screen.MousePointer = vbHourglass

2244:         psbar.ShowProgressBar "Gathering values from " & pRasterLayer.Name & " attribute table...", 1, _
              theValueCount, 1, True

        ' PUT VALUES INTO ESRI ARRAYS ------------------------------------------------

2249:         Do While Not pRow Is Nothing
2250:           If lngCountCountField >= 0 Then        ' I.E. IF THERE IS A COUNT FIELD; THERE SHOULD BE IN RASTER TABLES
2251:             lngCountCountValue = pRow.Value(lngCountCountField)
2252:           Else                                   ' IF NO COUNT FIELD, WEIGHT THINGS EQUALLY
2253:             lngCountCountValue = 1
2254:           End If
2255:           For anIndex = LBound(theKeys) To UBound(theKeys)   ' KEYS ARE STRING VERSIONS OF FIELD INDEX VALUES
2256:             aKey = theKeys(anIndex)
2257:             lngFieldIndex = CLng(aKey)
2258:             Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
2259:             Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
2260:             lngNumberNull = pWorkOrderDictArray.Element(3)
2261:             booGridHistogram = pWorkOrderDictArray.Element(5)
```

```
            Select Case pFieldInfoArray.Element(2)

              Case "Number"
2266:                 dblValue = pRow.Value(lngFieldIndex)
2267:                 If IsNull(dblValue) Then
2268:                   lngNumberNull = lngNumberNull + 1
2269:                   pWorkOrderDictArray.Element(5) = lngNumberNull
2270:                 Else
2271:                   Set pNumValArray = pWorkOrderDictArray.Element(2)
2272:                   pNumValArray.Add dblValue
2273:                   Set pSizeValArray = pWorkOrderDictArray.Element(4)
2274:                   pSizeValArray.Add lngCountCountValue
2275:                 End If

              Case "String"
2278:                 strValue = pRow.Value(lngFieldIndex)
2279:                 If strValue = "" Then strValue = "<-- EMPTY STRING -->"
2280:                 Set pStrValArray = pWorkOrderDictArray.Element(2)
2281:                 pStrValArray.Add strValue
2282:                 Set pSizeValArray = pWorkOrderDictArray.Element(4)
2283:                 pSizeValArray.Add lngCountCountValue

              Case "Date"
2286:                 dateValue = pRow.Value(lngFieldIndex)
2287:                 If IsNull(dateValue) Then
2288:                   lngNumberNull = lngNumberNull + 1
2289:                   pWorkOrderDictArray.Element(5) = lngNumberNull
2290:                 Else
2291:                   Set pDateValArray = pWorkOrderDictArray.Element(2)
2292:                   pDateValArray.Add dateValue
2293:                   Set pSizeValArray = pWorkOrderDictArray.Element(4)
2294:                   pSizeValArray.Add lngCountCountValue
2295:                 End If
2296:              End Select

2298:            Next anIndex

2300:            psbar.StepProgressBar
2301:            Set pRow = pCursor.NextRow
2302:          Loop
'         '-------------------------------------------------------------------------------------------------------
'         ' FINALLY ACTUALLY CALCULATE STATISTICS
'         '-----------------------------------
       '         KEY = STRING OF FIELD INDEX
       '    ELEMENT = VARIANT ARRAY WITH 7 ITEMS:
       '              1) STRING ARRAY WITH:
       '                 a) FIELD NAME
```

```
      '               b) FIELD ALIAS
      '               c) FIELD TYPE (SHOULD BE "NUMBER", "STRING", OR "DATE")
      '            2) STRING ARRAY WITH ALL REQUESTED STATISTICS
      '            3) NUMBER, STRING OR DATE ARRAY FOR DATA VALUES, DEPENDING ON FIELD TYPE
      '            4) NUMBER (LONG) OF NULL VALUES FOUND IN THIS FIELD
      '            5) A NUMBER ARRAY FOR COUNTS OF EACH UNIQUE CELL VALUE (EACH FIELD GETS ITS OWN BECAUSE THEY MIGHT HAVE NULL
VALUES)
      '            6) BOOLEAN INDICATING WHETHER HISTOGRAM SHOULD BE GENERATED
      '            7) BOOLEAN INDICATING WHETHER CATEGORICAL DATA SHOULD BE GENERATED
      '            8) BOOLEAN INDICATING WHETHER NUMERIC STATISTICS SHOULD BE GENERATED

2320:         For anIndex = LBound(theKeys) To UBound(theKeys)
2321:           aKey = theKeys(anIndex)
2322:           lngFieldIndex = CLng(aKey)
2323:           Set pOrigField = pTableFields.Field(lngFieldIndex)
2324:           Set pClone = pOrigField
2325:           Set pNewStringField = pClone.Clone
2326:           Set pWorkOrderDictArray = pWorkOrderDictionary.GetObject(aKey)
2327:           Set pFieldInfoArray = pWorkOrderDictArray.Element(0)
2328:           Set pStatInfoArray = pWorkOrderDictArray.Element(1)
2329:           lngNumberNull = pWorkOrderDictArray.Element(3)
2330:           Set pSizeValArray = pWorkOrderDictArray.Element(4)
2331:           booGridHistogram = pWorkOrderDictArray.Element(5)
2332:           booGridCategorical = pWorkOrderDictArray.Element(6)
2333:           booGridNumeric = pWorkOrderDictArray.Element(7)

          Select Case pFieldInfoArray.Element(2)
            ' NEED TO DO DIFFERENT THINGS FOR NUMBERS, STRINGS AND DATES
            Case "Number"   ' ------------------------------------------------------------------------------------------

2339:               strReport = strReport & _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" & vbCrLf
& _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b Data saved to \b0 zzTableSaveTozz\par" & vbCrLf & _
                  "\b Selected Statistics:\b0\par" & vbCrLf

              ' SET STAT VARIABLES
2346:               Set pNumValArray = pWorkOrderDictArray.Element(2)
2347:               If pNumValArray.Count = 0 Then                        ' NO STATS CALCULATED FOR THIS FIELD
2348:                 strReport = strReport & "    !!! No Stats Calculated!  Apparently no non-null values found...\par" & vbCrLf
2349:                 strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTableSaveTozz", "[NO TABLE GENERATED]")
2350:               ElseIf pNumValArray.Count = 1 Then                    ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
2351:                 theSum = pNumValArray.Element(0)
2352:                 theMean = pNumValArray.Element(0)
2353:                 theMinimum = pNumValArray.Element(0)
2354:                 theMaximum = pNumValArray.Element(0)
```

```
2355:                 theMedian = pNumValArray.Element(0)
2356:                 theRange = 0
2357:                 theCount = 1
2358:                 theStdDev = 0
2359:                 theVar = 0
2360:                 theMedian = pNumValArray.Element(0)
2361:                 theStdErrMean = 0
2362:                 theModeString = "No Mode Found"
              ' theFinalModes
2364:                 booFoundMode = False
              ReDim lngHistArray(0)
2366:                 lngHistArray(0) = pNumValArray.Element(0)
2367:                 theCount = 1
2368:                 theProportion = 1
2369:                 theArea = dblCellSize

2371:               Else                                                ' ACTUAL STATS
              ' PUT NUMBERS IN DOUBLE ARRAY
2373:                 If booGridNumeric Then
                    ReDim dblNumValArray(pNumValArray.Count - 1)
                    ReDim dblSizeValArray(pNumValArray.Count - 1)
2376:                   For anIndex2 = 0 To pNumValArray.Count - 1
2377:                     dblNumValArray(anIndex2) = pNumValArray.Element(anIndex2)
2378:                     dblSizeValArray(anIndex2) = pSizeValArray.Element(anIndex2)
2379:                   Next anIndex2
2380:                   Set pStats = New esriSystem.VarArray

2382:                   Linkages.QuickSort.DoubleAscendingWithSizes dblNumValArray, dblSizeValArray, 0, UBound(dblNumValArray)

2384:                   Set pStats = Linkages.MyGeneralOperations.BasicStatsFromVAT(dblNumValArray, _
                      dblSizeValArray, pFieldInfoArray.Element(1), pRasterLayer.Name, m_App, _
                      lngNumBins)

2388:                   theSum = pStats.Element(0)
2389:                   theMean = pStats.Element(1)
2390:                   theMinimum = pStats.Element(2)
2391:                   theMaximum = pStats.Element(3)
2392:                   theRange = pStats.Element(4)
2393:                   theCount = pStats.Element(5)
2394:                   theStdDev = pStats.Element(6)
2395:                   theVar = pStats.Element(7)
2396:                   theMedian = pStats.Element(8)
2397:                   theStdErrMean = pStats.Element(9)
2398:                   theModeString = pStats.Element(10)
2399:                   theFinalModes = pStats.Element(11)
2400:                   booFoundMode = pStats.Element(12)
2401:                   lngHistArray = pStats.Element(13)
```

```
2402:                   If theMaximum > 100000 Then
2403:                     lngNumDecPlaces = 0
2404:                   ElseIf (theMaximum > 1000) And (theMaximum <= 100000) Then
2405:                     lngNumDecPlaces = 2
2406:                   ElseIf (theMaximum > 10) And (theMaximum <= 1000) Then
2407:                     lngNumDecPlaces = 4
2408:                   ElseIf theMaximum <= 10 Then
2409:                     lngNumDecPlaces = pOrigField.Scale
2410:                   End If
2411:                 End If
2412:               End If

             ' MAKE REPORT AND TABLES FOR NUMERIC STATISTICS
2415:               If booGridNumeric Then
2416:                 strfilename = MakeNumberDBASETable(pOrigField, pStats, pStatInfoArray, pFieldInfoArray, Nothing)
2417:                 lngLetterInd = -1
2418:                 For anIndex2 = 0 To pStatInfoArray.Count - 1
2419:                   strStatName = pStatInfoArray.Element(anIndex2)
                    Select Case strStatName
                      Case "Minimum"
2422:                         lngLetterInd = lngLetterInd + 1
2423:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Minimum: \b0  " & CStr(theMinimum) &
"\par" & vbCrLf
                      Case "Maximum"
2425:                         lngLetterInd = lngLetterInd + 1
2426:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Maximum: \b0  " & CStr(theMaximum) &
"\par" & vbCrLf
                      Case "Mean"
2428:                         lngLetterInd = lngLetterInd + 1
2429:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mean: \b0  " & CStr(theMean) & "\par" &
vbCrLf
                      Case "Sum"
2431:                         lngLetterInd = lngLetterInd + 1
2432:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Sum: \b0  " & CStr(theSum) & "\par" &
vbCrLf
                      Case "Standard Deviation"
2434:                         lngLetterInd = lngLetterInd + 1
2435:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Standard Deviation: \b0  " &
CStr(theStdDev) & "\par" & vbCrLf
                      Case "Median"
2437:                         lngLetterInd = lngLetterInd + 1
2438:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Median: \b0  " & CStr(theMedian) & "\par"
& vbCrLf
                      Case "Mode"
2440:                         lngLetterInd = lngLetterInd + 1
2441:                         strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Mode: \b0  " & CStr(theModeString) &
"\par" & vbCrLf
```

```
                  Case "Histogram"
2443:                     lngLetterInd = lngLetterInd + 1
2444:                     strReport = strReport & "  \b " & strLetters(lngLetterInd) & "] Histogram: \b0\par" & vbCrLf
2445:                     If pNumValArray.Count = 1 Then
2446:                       strReport = strReport & "        !!! Single Value:  No Histogram created...\par" & vbCrLf
2447:                     Else
2448:                       strHistReport = Linkages.CorridorAnalysisFunctions.MakeHistogramData(pFieldInfoArray, lngNumBins, _
                          theMinimum, theMaximum, lngHistArray, m_App, txtOutput.Text, lngNumDecPlaces)
2450:                       strReport = strReport & strHistReport
2451:                     End If
2452:                  End Select
2453:                Next anIndex2

2455:                Else
2456:                  strfilename = "[No numeric stats generated]"
2457:                End If
2458:                strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzTableSaveTozz", _
                     Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\"))

              ' MAKE REPORT AND TABLES FOR CATEGORICAL STATISTICS (COUNT, PROPORTION, AREA)
2462:                If booGridCategorical Then
2463:                  Set pStats = Linkages.CorridorAnalysisFunctions.StatsPropsForNumbers(pNumValArray, pSizeValArray)
2464:                  strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField, dblCellSize)

'                 For anIndex3 = 0 To pNumValArray.Count - 1
'                   strSubReport = strSubReport & "Value = " & pNumValArray.Element(anIndex3) & _
'                       ",   Count = " & pSizeValArray.Element(anIndex3) & vbCrLf
'                 Next anIndex3

2471:                  strReport = strReport & vbCrLf & strSubReport & "=====================================================\par" &
vbCrLf
2472:                Else
2473:                  strReport = strReport & vbCrLf & "=====================================================\par" & vbCrLf
2474:                End If
            Case "String"   ' ------------------------------------------------------------------------------------------

2477:                strReport = strReport & _
                     "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" &
vbCrLf & _
                     "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                     "\b Statistics by Unique Value: \b0\par" & vbCrLf
              ' SET STAT VARIABLES
2482:                Set pStrValArray = pWorkOrderDictArray.Element(2)
2483:                Set pClone = pOrigField
2484:                Set pNewStringField = pClone.Clone

2486:                If pStrValArray.Count = 0 Then                       ' NO STATS CALCULATED FOR THIS FIELD
```

```
2487:                 strReport = strReport & "    !!! No Stats Calculated!  Apparently no non-null values found...\par" & vbCrLf
2488:               ElseIf pStrValArray.Count = 1 Then                    ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
2489:                 Set pStats = New esriSystem.VarArray
2490:                 Set pVarArray = New esriSystem.VarArray

2492:                 strValue = pStrValArray.Element(0)
2493:                 pVarArray.Add strValue
2494:                 theCount = 1
2495:                 pVarArray.Add theCount
2496:                 dblProportion = 1
2497:                 pVarArray.Add dblProportion
2498:                 dblSize = 1
2499:                 pVarArray.Add dblSize
2500:                 pStats.Add pVarArray

2502:                 strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField)
2503:                 strReport = strReport & strSubReport
2504:               Else                                                  ' ACTUAL STATS
             ' STRING STAT FUNCTION TAKES STRINGS IN ESRI STRING ARRAY

2507:                 Set pStats = Linkages.CorridorAnalysisFunctions.StatsForStrings(pStrValArray, Nothing)
2508:                 strSubReport = CalcCategorySubReport(pStats, pStatInfoArray, pNewStringField, dblCellSize)
2509:                 strReport = strReport & strSubReport

2511:               End If

           Case "Date"   ' ----------------------------------------------------------------------------------------------
2514:               Set pDateValArray = pWorkOrderDictArray.Element(2)

2516:               If pDateValArray.Count = 0 Then                       ' NO STATS CALCULATED FOR THIS FIELD
2517:                 strSubReport = _
                    "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" &
vbCrLf & _
                    "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                    "\b [NO TABLE GENERATED]\par" & vbCrLf & _
                    "  --> !!! No Stats Calculated!  Apparently no non-null date values found...\par" & vbCrLf & _
                    "==================================================\par" & vbCrLf

2524:               ElseIf pDateValArray.Count = 1 Then                   ' ONE VALUE; VERY SIMPLIFIED SET OF STATS
2525:                 dateValue = pDateValArray.Element(0)
2526:                 strfilename = MakeDateTable(pFieldInfoArray, dateValue, dateValue)
2527:                 strSubReport = _
                    "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" &
vbCrLf & _
                    "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                    "\b Data saved to " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
                    "\b Statistics: \b0\par  " & vbCrLf & _
```

```
                  "   1] Earliest Date = " & CStr(dateValue) & "\par" & vbCrLf & _
                  "   2] Latest Date = " & CStr(dateValue) & "\par" & vbCrLf & _
                  "=====================================================\par" & vbCrLf

2536:             Else                                                  ' ACTUAL STATS
              ' STRING STAT FUNCTION TAKES STRINGS IN ESRI STRING ARRAY
2538:               Set pStats = Linkages.CorridorAnalysisFunctions.StatsForDates(pDateValArray)
2539:               strfilename = MakeDateTable(pFieldInfoArray, pStats.Element(0), pStats.Element(1))
2540:               strSubReport = _
                  "\b Field Name: \b0  " & pFieldInfoArray.Element(0) & "  [Alias = " & pFieldInfoArray.Element(1) & "]\par" &
vbCrLf & _
                  "\b Field Type: \b0  " & pFieldInfoArray.Element(2) & "\par" & vbCrLf & _
                  "\b Data saved to " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
                  "\b Statistics: \b0\par  " & vbCrLf & _
                  "   1] Earliest Date = " & CStr(pStats.Element(0)) & "\par" & vbCrLf & _
                  "   2] Latest Date = " & CStr(pStats.Element(1)) & "\par" & vbCrLf & _
                  "=====================================================\par" & vbCrLf

2549:             End If
2550:             strReport = strReport & strSubReport
2551:           End Select
2552:         Next anIndex
2553:       End If
2554:     Else    '  DONE WORKING WITH RASTERS WITH TABLES.  NEXT SECTION IS CONTINUOUS RASTERS

      ' GET GENERAL COUNT OF NON-NULL CELL VALUES
2557:       Set pLogicalOp = New RasterMathOps
2558:       Set pCountGeoDataset = pLogicalOp.IsNull(pRaster)
2559:       Set pCountRasterBandCollection = pCountGeoDataset
2560:       Set pCountRasterBand = pCountRasterBandCollection.Item(0)
2561:       Set pTable = pCountRasterBand.AttributeTable
2562:       lngCountValField = pTable.FindField("Value")
2563:       lngCountCountField = pTable.FindField("Count")
2564:       lngCountCountValue = -9999
2565:       Set pCountCursor = pTable.Search(Nothing, True)
2566:       Set pCountRow = pCountCursor.NextRow
2567:       Do Until pCountRow Is Nothing
2568:         lngCountValValue = pCountRow.Value(lngCountValField)
2569:         If lngCountValValue = 0 Then
2570:           lngCountCountValue = pCountRow.Value(lngCountCountField)
2571:         End If
2572:         Set pCountRow = pCountCursor.NextRow
2573:       Loop

'      strReport = "Statistics Report on Grid '" & pRasterLayer.Name & "':" & vbCrLf & _
          "-----------------------------------------------" & vbCrLf & _
          "Non-Null Grid Cell Count = " & Linkages.aml_func_mod.InsertCommas(lngCountCountValue) & _
```

```
          " grid cells" & vbCrLf & _
          "-------------------------------------------------" & vbCrLf

2581:       strReport = _
        "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fprq2\fcharset0 Arial;}}" & vbCrLf & _
        "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\tx90\tx360\tx450\tx720\b\f0\fs16 Statistics Report on Grid '" & _
        pRasterLayer.Name & "'\b0\par" & vbCrLf & _
        "\pard ------------------------------------------------------------------------------\par" & vbCrLf & _
        "\b Non-Null Grid Cell Count: \b0 " & Linkages.aml_func_mod.InsertCommas(lngCountCountValue) & _
          " grid cells\par" & vbCrLf & _
        "------------------------------------------------------------------------------\par" & vbCrLf

2590:       booContMin = False
2591:       booContMax = False
2592:       booContMean = False
2593:       booContMode = False
2594:       booContMedian = False
2595:       booContSD = False
2596:       booContHist = False
2597:       lngNewFieldCounter = 1              ' WILL AUTOMATICALLY GET A UNIQUE ID FIELD
2598:       For anIndex = 1 To pNodes.Count     ' CHECK ALL NODES IN TREEVIEW
2599:         Set pNode = pNodes.Item(anIndex)
2600:         If pNode.Image = 2 Then           ' THEN THIS STAT IS SELECTED
2601:           Set pParentNode = pNode.Parent
2602:           strParentKey = pParentNode.Key

          ' MAKE EMPTY FIELD INFO ARRAY
2605:           Set pFieldInfoArray = New esriSystem.strArray

          ' STAT AND FIELD ARRAY HAS VALUES FOR EACH SELECTED NODE IN TREE VIEW.  VALUES ARE:
          ' (0) Field Index
          ' (1) Field Name
          ' (2) Field Alias
          ' (3) Field Type (Number, String, Date)
          ' (4) Name of Statistic

          ' GET INFORMATION ON FIELD AND ADD TO FIELD INFO ARRAY
2615:           pFieldInfoArray.Add pRasterLayer.Name
2616:           strStatName = pNode.Text

2618:           If strStatName = "Minimum" Then
2619:             lngNewFieldCounter = lngNewFieldCounter + 1
2620:             booContMin = True
2621:           End If
2622:           If strStatName = "Maximum" Then
2623:             lngNewFieldCounter = lngNewFieldCounter + 1
2624:             booContMax = True
```

```
2625:           End If
2626:           If strStatName = "Mean" Then
2627:             lngNewFieldCounter = lngNewFieldCounter + 1
2628:             booContMean = True
2629:           End If
2630:           If strStatName = "Median" Then
2631:             lngNewFieldCounter = lngNewFieldCounter + 1
2632:             booContMedian = True
2633:           End If
2634:           If strStatName = "Mode" Then
2635:             lngNewFieldCounter = lngNewFieldCounter + 1
2636:             booContMode = True
2637:           End If
2638:           If strStatName = "Standard Deviation" Then
2639:             lngNewFieldCounter = lngNewFieldCounter + 1
2640:             booContSD = True
2641:           End If
2642:           If strStatName = "Histogram" Then
2643:             booContHist = True
2644:           End If

2646:         End If        ' END CHECKING IF THIS NODE IS SELECTED
2647:       Next anIndex    ' DONE WORKING THROUGH STAT OPTIONS FOR CONTINUOUS RASTERS

2649:       strReport = strReport & "\b Continuous Grid Value Statistics\b0\par" & vbCrLf & _
                        "\b Data saved to \b0 zzzFilename\par" & vbCrLf & _
                        "\b Statistics: \b0\par" & vbCrLf
2652:       anIndex = -1
2653:       If booContMin Then
2654:         anIndex = anIndex + 1
2655:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Minimum: \b0  " & CStr(pRasterStatistics.Minimum) & "\par" &
vbCrLf
2656:       End If
2657:       If booContMax Then
2658:         anIndex = anIndex + 1
2659:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Maximum: \b0  " & CStr(pRasterStatistics.Maximum) & "\par" &
vbCrLf
2660:       End If
2661:       If booContMean Then
2662:         anIndex = anIndex + 1
2663:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Mean: \b0  " & CStr(pRasterStatistics.Mean) & "\par" & vbCrLf
2664:       End If
2665:       If booContMedian Then
2666:         anIndex = anIndex + 1
2667:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Median: \b0  " & CStr(pRasterStatistics.Median) & "\par" &
vbCrLf
2668:       End If
```

```
2669:       If booContMode Then
2670:         anIndex = anIndex + 1
2671:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Mode: \b0  " & CStr(pRasterStatistics.Mode) & "\par" & vbCrLf
2672:       End If
2673:       If booContSD Then
2674:         anIndex = anIndex + 1
2675:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Standard Deviation: \b0  " &
CStr(pRasterStatistics.StandardDeviation) & "\par" & vbCrLf
2676:       End If
2677:       If booContHist Then
2678:         anIndex = anIndex + 1
2679:         If pRasterStatistics.Maximum > 100000 Then
2680:           lngNumDecPlaces = 0
2681:         ElseIf (pRasterStatistics.Maximum > 1000) And (pRasterStatistics.Maximum <= 100000) Then
2682:           lngNumDecPlaces = 2
2683:         ElseIf (pRasterStatistics.Maximum > 10) And (pRasterStatistics.Maximum <= 1000) Then
2684:           lngNumDecPlaces = 4
2685:         ElseIf pRasterStatistics.Maximum <= 10 Then
2686:           lngNumDecPlaces = 6
2687:         End If
2688:         lngHistArray = Linkages.CorridorAnalysisFunctions.GridHistogram(pRasterLayer, pRasterStatistics.Minimum, _
          pRasterStatistics.Maximum, lngNumBins, m_App)
2690:         strReport = strReport & "  \b " & strLetters(anIndex) & "] Histogram: \b0\par" & vbCrLf
2691:         If UBound(lngHistArray) = 0 Then
2692:           strReport = strReport & "        !!! Single Value:  No Histogram created...\par" & vbCrLf
2693:         Else
2694:           strHistReport = Linkages.CorridorAnalysisFunctions.MakeHistogramData(pFieldInfoArray, lngNumBins,
pRasterStatistics.Minimum, _
            pRasterStatistics.Maximum, lngHistArray, m_App, txtOutput.Text, lngNumDecPlaces)
2696:           strReport = strReport & strHistReport
2697:         End If
2698:       End If

     ' ONLY MAKE NEW TABLE IF NON-HISTOGRAM STATS SELECTED
2701:       If lngNewFieldCounter > 1 Then
2702:         Set pNewFields = New Fields
2703:         Set pNewFieldsEdit = pNewFields
2704:         pNewFieldsEdit.FieldCount = lngNewFieldCounter

       ' MAKE UNIQUE ID FIELD
2707:         Set pNewField = New Field
2708:         Set pNewFieldEdit = pNewField
2709:         pNewFieldEdit.Name = "Unique_ID"
2710:         pNewFieldEdit.Type = esriFieldTypeInteger
2711:         pNewFieldEdit.Precision = 8
2712:         Set pNewFieldsEdit.Field(0) = pNewField
```

```
2714:         anIndex = 0             ' INDEX PRESET FOR FIRST FIELD; UNIQUE_ID.  IF NO UNIQUE ID FIELD, anIndex WOULD EQUAL -1

        ' MAKE STAT FIELDS
2717:         If booContMin Then
2718:           anIndex = anIndex + 1
2719:           Set pNewField = New Field
2720:           Set pNewFieldEdit = pNewField
2721:           With pNewFieldEdit
2722:             .Type = esriFieldTypeDouble
2723:             .Name = "Minimum"
2724:             .Precision = 14
2725:             .Scale = 8
2726:           End With
2727:           Set pNewFieldsEdit.Field(anIndex) = pNewField
2728:         End If
2729:         If booContMax Then
2730:           anIndex = anIndex + 1
2731:           Set pNewField = New Field
2732:           Set pNewFieldEdit = pNewField
2733:           With pNewFieldEdit
2734:             .Type = esriFieldTypeDouble
2735:             .Name = "Maximum"
2736:             .Precision = 14
2737:             .Scale = 8
2738:           End With
2739:           Set pNewFieldsEdit.Field(anIndex) = pNewField
2740:         End If
2741:         If booContMean Then
2742:           anIndex = anIndex + 1
2743:           Set pNewField = New Field
2744:           Set pNewFieldEdit = pNewField
2745:           With pNewFieldEdit
2746:             .Type = esriFieldTypeDouble
2747:             .Name = "Mean"
2748:             .Precision = 14
2749:             .Scale = 8
2750:           End With
2751:           Set pNewFieldsEdit.Field(anIndex) = pNewField
2752:         End If
2753:         If booContMedian Then
2754:           anIndex = anIndex + 1
2755:           Set pNewField = New Field
2756:           Set pNewFieldEdit = pNewField
2757:           With pNewFieldEdit
2758:             .Type = esriFieldTypeDouble
2759:             .Name = "Median"
2760:             .Precision = 14
```

```
2761:             .Scale = 8
2762:           End With
2763:           Set pNewFieldsEdit.Field(anIndex) = pNewField
2764:         End If
2765:         If booContMode Then
2766:           anIndex = anIndex + 1
2767:           Set pNewField = New Field
2768:           Set pNewFieldEdit = pNewField
2769:           With pNewFieldEdit
2770:             .Type = esriFieldTypeDouble
2771:             .Name = "Mode"
2772:             .Precision = 14
2773:             .Scale = 8
2774:           End With
2775:           Set pNewFieldsEdit.Field(anIndex) = pNewField
2776:         End If
2777:         If booContSD Then
2778:           anIndex = anIndex + 1
2779:           Set pNewField = New Field
2780:           Set pNewFieldEdit = pNewField
2781:           With pNewFieldEdit
2782:             .Type = esriFieldTypeDouble
2783:             .Name = "St_Dev"
2784:             .Precision = 14
2785:             .Scale = 8
2786:           End With
2787:           Set pNewFieldsEdit.Field(anIndex) = pNewField
2788:         End If

2790:         strfilename = txtOutput.Text
2791:         If Right(strfilename, 1) <> "/" And Right(strfilename, 1) <> "\" Then strfilename = strfilename & "\"
2792:         strfilename = strfilename & pRasterLayer.Name & "_stats.dbf"
2793:         strfilename = Linkages.aml_func_mod.MakeUniqueFilename(strfilename)
2794:         Set pTable = Linkages.aml_func_mod.CreatedBASETable(strfilename, pNewFields)

       ' ADD DATA
2797:         Set pRow = pTable.CreateRow
2798:         pRow.Value(pTable.FindField("Unique_ID")) = 1

2800:         If booContMin Then
2801:           pRow.Value(pTable.FindField("Minimum")) = pRasterStatistics.Minimum
2802:         End If
2803:         If booContMax Then
2804:           pRow.Value(pTable.FindField("Maximum")) = pRasterStatistics.Maximum
2805:         End If
2806:         If booContMean Then
2807:           pRow.Value(pTable.FindField("Mean")) = pRasterStatistics.Mean
```

```
2808:         End If
2809:         If booContMedian Then
2810:           pRow.Value(pTable.FindField("Median")) = pRasterStatistics.Median
2811:         End If
2812:         If booContMode Then
2813:           pRow.Value(pTable.FindField("Mode")) = pRasterStatistics.Mode
2814:         End If
2815:         If booContSD Then
2816:           pRow.Value(pTable.FindField("St_Dev")) = pRasterStatistics.StandardDeviation
2817:         End If
2818:         pRow.Store
2819:         Set pNewStandaloneTable = New StandaloneTable
2820:         Set pNewStandaloneTable.Table = pTable

2822:         Set pTableWindow2 = New TableWindow

2824:         With pTableWindow2
2825:           Set .StandaloneTable = pNewStandaloneTable
2826:           Set .Application = m_App
2827:           .TableSelectionAction = esriSelectFeatures
2828:           .ShowAliasNamesInColumnHeadings = True
2829:           .ShowSelected = False
2830:           .Show True
2831:         End With
2832:         Set pMxDoc = m_App.Document
2833:         Set pStandaloneTableCollection = pMxDoc.FocusMap
2834:         pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable

2836:         pMxDoc.UpdateContents

2838:       Else
2839:         strfilename = "[No Statistics Table Created]"
2840:       End If         ' DONE MAKING CONTINUOUS RASTER STAT TABLE
2841:       strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzzFilename", _
           Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\"))
2843:     End If           ' DONE EXAMINING RASTER SECTION
2844:   End If             ' DONE CALCULATING STATISTICS


2847:   psbar.HideProgressBar
2848:   Screen.MousePointer = vbDefault

2850:   strReport = strReport & _
          "==================================================\par" & vbCrLf & _
          Linkages.MyGeneralOperations.ReturnTimeElapsedRTF(theTimeBegan, Now, 7)

2854:   strReport = strReport & "}"
```

```
  ' SHOW REPORT
  Dim frmReportForm As New Linkages.frmReport_modal
2858:   frmReportForm.txtReport.TextRTF = strReport
2859:   frmReportForm.Show vbModal

  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Function MakeDateTable(pFieldInfoArray As esriSystem.IStringArray, dateStartDate As Date, _
    dateEndDate As Date) As String
  On Error GoTo ErrorHandler

  Dim strFieldName As String
2871:   strFieldName = pFieldInfoArray.Element(0)
  Dim strfilename As String
2873:   strfilename = txtOutput.Text
2874:   If Right(strfilename, 1) <> "/" And Right(strfilename, 1) <> "\" Then strfilename = strfilename & "\"
2875:   strfilename = strfilename & strFieldName & "_stats.dbf"
2876:   strfilename = Linkages.aml_func_mod.MakeUniqueFilename(strfilename)

  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit

2883:   Set pNewFields = New Fields
2884:   Set pNewFieldsEdit = pNewFields
  ' 1) Unique_ID
  ' 2) Start_Date
  ' 3) End_Date

2889:   pNewFieldsEdit.FieldCount = 3

2891:   Set pNewField = New Field
2892:   Set pNewFieldEdit = pNewField
2893:   With pNewFieldEdit
2894:     .Name = "Unique_ID"
2895:     .Precision = 8
2896:     .Type = esriFieldTypeInteger
2897:   End With
2898:   Set pNewFieldsEdit.Field(0) = pNewField

2900:   Set pNewField = New Field
```

```
2901:   Set pNewFieldEdit = pNewField
2902:   With pNewFieldEdit
2903:     .Name = "Start_Date"
2904:     .Type = esriFieldTypeDate
2905:   End With
2906:   Set pNewFieldsEdit.Field(1) = pNewField

2908:   Set pNewField = New Field
2909:   Set pNewFieldEdit = pNewField
2910:   With pNewFieldEdit
2911:     .Name = "End_Date"
2912:     .Type = esriFieldTypeDate
2913:   End With
2914:   Set pNewFieldsEdit.Field(2) = pNewField

  Dim pTable As ITable
2917:   Set pTable = Linkages.aml_func_mod.CreatedBASETable(strfilename, pNewFields)
  Dim pRow As IRow

2920:   Set pRow = pTable.CreateRow
2921:   pRow.Value(pTable.FindField("Unique_ID")) = 1
2922:   pRow.Value(pTable.FindField("Start_Date")) = dateStartDate
2923:   pRow.Value(pTable.FindField("End_Date")) = dateEndDate
2924:   pRow.Store

  ' MAKE TABLE WINDOW AND ADD IT TO DOCUMENT
  Dim pNewStandaloneTable As IStandaloneTable
2928:   Set pNewStandaloneTable = New StandaloneTable
2929:   Set pNewStandaloneTable.Table = pTable

  Dim pTableWindow2 As ITableWindow2
2932:   Set pTableWindow2 = New TableWindow

  Dim lngLeft As Long
  Dim lngTop As Long
  Dim lngRight As Long
  Dim lngBottom As Long

2939:   With pTableWindow2
2940:     Set .StandaloneTable = pNewStandaloneTable
2941:     Set .Application = m_App
2942:     .TableSelectionAction = esriSelectFeatures
2943:     .ShowAliasNamesInColumnHeadings = True
2944:     .ShowSelected = False
2945:     .Show True
2946:   End With
```

```
  Dim pMxDoc As IMxDocument
2949:   Set pMxDoc = m_App.Document
  Dim pStandaloneTableCollection As IStandaloneTableCollection
2951:   Set pStandaloneTableCollection = pMxDoc.FocusMap
2952:   pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable

2954:   pMxDoc.UpdateContents

2956:   MakeDateTable = strfilename

  Exit Function
ErrorHandler:
  HandleError False, "MakeDateTable " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function



Private Function MakeNumberDBASETable(pOrigField As IField, pStats As esriSystem.IVariantArray, _
    pStatInfoArray As esriSystem.IStringArray, pFieldInfoArray As esriSystem.IStringArray, _
    pWeightStats As esriSystem.IDoubleArray) As String
  On Error GoTo ErrorHandler


  Dim strFieldName As String
2972:   strFieldName = pFieldInfoArray.Element(0)
  Dim strfilename As String
2974:   strfilename = txtOutput.Text
2975:   If Right(strfilename, 1) <> "/" And Right(strfilename, 1) <> "\" Then strfilename = strfilename & "\"
2976:   strfilename = strfilename & strFieldName & "_stats.dbf"
2977:   strfilename = Linkages.aml_func_mod.MakeUniqueFilename(strfilename)

  Dim theSum As Double
  Dim theMean As Double
  Dim theMinimum As Double
  Dim theMaximum As Double
  Dim theRange As Double
  Dim theCount As Double
  Dim theStdDev As Double
  Dim theVar As Double
  Dim theMedian As Double
  Dim theStdErrMean As Double
  Dim theModeString As String
  Dim theFinalModes() As Double
  Dim booFoundMode As Boolean
  Dim theMeanWBL As Double
  Dim theStDevWBL As Double
```

```
  Dim theVarWBL As Double

2996:   If Not pStats Is Nothing Then
2997:     theSum = pStats.Element(0)
2998:     theMean = pStats.Element(1)
2999:     theMinimum = pStats.Element(2)
3000:     theMaximum = pStats.Element(3)
3001:     theRange = pStats.Element(4)
3002:     theCount = pStats.Element(5)
3003:     theStdDev = pStats.Element(6)
3004:     theVar = pStats.Element(7)
3005:     theMedian = pStats.Element(8)
3006:     theStdErrMean = pStats.Element(9)
3007:     theModeString = pStats.Element(10)
3008:     theFinalModes = pStats.Element(11)
3009:     booFoundMode = pStats.Element(12)
3010:   End If
3011:   If Not pWeightStats Is Nothing Then
3012:     theMeanWBL = pWeightStats.Element(0)
3013:     theStDevWBL = pWeightStats.Element(1)
3014:     theVarWBL = pWeightStats.Element(2)
3015:   End If

  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit
  Dim strStatistic As String
  Dim lngIndex As Long

3024:   Set pNewFields = New Fields
3025:   Set pNewFieldsEdit = pNewFields

  Dim lngNewFieldCount As Long
3028:   lngNewFieldCount = 1      ' FOR UNIQUE ID FIELD

  Dim booHasHistogram As Boolean
3031:   booHasHistogram = False
3032:   For lngIndex = 0 To pStatInfoArray.Count - 1
3033:     strStatistic = pStatInfoArray.Element(lngIndex)
3034:     If strStatistic = "Minimum" Or _
       strStatistic = "Maximum" Or _
       strStatistic = "Mean" Or _
       strStatistic = "Median" Or _
       strStatistic = "Mode" Or _
       strStatistic = "Sum" Or _
       strStatistic = "Standard Deviation" Or _
```

```
       strStatistic = "Variance" Or _
       strStatistic = "Mean_WBL" Or _
       strStatistic = "Standard Deviation_WBL" Or _
       strStatistic = "Variance_WBL" Or _
       strStatistic = "Mean_WBA" Or _
       strStatistic = "Standard Deviation_WBA" Or _
       strStatistic = "Variance_WBA" Then
3048:           lngNewFieldCount = lngNewFieldCount + 1
3049:     End If
'    If strStatistic = "Histogram" Then
'      booHasHistogram = True
'      Exit For
'    End If
3054:   Next lngIndex

'  If booHasHistogram Then
'    ' SET AS ONE LESS THAN STAT INFO COUNT BECAUSE WE WON'T CREATE A HISTOGRAM TABLE HERE
'    pNewFieldsEdit.FieldCount = pStatInfoArray.Count
'  Else
'    pNewFieldsEdit.FieldCount = pStatInfoArray.Count + 1
'  End If
3062:   pNewFieldsEdit.FieldCount = lngNewFieldCount

3064:   If booHasHistogram And pStatInfoArray.Count = 1 Then
    ' IF ONLY MAKING HISTOGRAM, THEN NO REASON TO GENERATE A TABLE HERE
3066:     MakeNumberDBASETable = "[No general statistics dBASE table generated]"
    Exit Function
3068:   End If

  ' MAKE UNIQUE ID FIELD
3071:   Set pNewField = New Field
3072:   Set pNewFieldEdit = pNewField
3073:   pNewFieldEdit.Name = "Unique_ID"
3074:   pNewFieldEdit.Type = esriFieldTypeInteger
3075:   pNewFieldEdit.Precision = 8
3076:   Set pNewFieldsEdit.Field(0) = pNewField

  Dim lngFieldIndexCounter As Long
3079:   lngFieldIndexCounter = 0

3081:   For lngIndex = 0 To pStatInfoArray.Count - 1
3082:     strStatistic = pStatInfoArray.Element(lngIndex)
'    If Not strStatistic = "Histogram" Then    ' SKIP HISTOGRAM HERE
3084:     If strStatistic = "Minimum" Or _
          strStatistic = "Maximum" Or _
          strStatistic = "Mean" Or _
          strStatistic = "Median" Or _
```

```
           strStatistic = "Mode" Or _
           strStatistic = "Sum" Or _
           strStatistic = "Standard Deviation" Or _
           strStatistic = "Variance" Or _
           strStatistic = "Mean_WBL" Or _
           strStatistic = "Standard Deviation_WBL" Or _
           strStatistic = "Variance_WBL" Or _
           strStatistic = "Mean_WBA" Or _
           strStatistic = "Standard Deviation_WBA" Or _
           strStatistic = "Variance_WBA" Then
3098:       lngFieldIndexCounter = lngFieldIndexCounter + 1
3099:       Set pNewField = New Field
3100:       Set pNewFieldEdit = pNewField

      Select Case strStatistic
        Case "Minimum"
3104:           With pNewFieldEdit
3105:             .Type = esriFieldTypeDouble
3106:             .Name = "Minimum"
3107:             .Precision = 14
3108:             .Scale = 8
3109:           End With

        Case "Maximum"
3112:           With pNewFieldEdit
3113:             .Type = esriFieldTypeDouble
3114:             .Name = "Maximum"
3115:             .Precision = 14
3116:             .Scale = 8
3117:           End With

        Case "Mean"
3120:           With pNewFieldEdit
3121:             .Type = esriFieldTypeDouble
3122:             .Name = "Mean"
3123:             .Precision = 14
3124:             .Scale = 8
3125:           End With

        Case "Sum"
3128:           With pNewFieldEdit
3129:             .Type = esriFieldTypeDouble
3130:             .Name = "Sum"
3131:             .Precision = 14
3132:             .Scale = 8
3133:           End With
```

```
        Case "Standard Deviation"
3136:           With pNewFieldEdit
3137:             .Type = esriFieldTypeDouble
3138:             .Name = "St_Dev"
3139:             .Precision = 14
3140:             .Scale = 8
3141:           End With

        Case "Variance"
3144:           With pNewFieldEdit
3145:             .Type = esriFieldTypeDouble
3146:             .Name = "Variance"
3147:             .Precision = 14
3148:             .Scale = 8
3149:           End With

        Case "Mean_WBL"
3152:           With pNewFieldEdit
3153:             .Type = esriFieldTypeDouble
3154:             .Name = "Mean_WBL"
3155:             .Precision = 14
3156:             .Scale = 8
3157:           End With

        Case "Standard Deviation_WBL"
3160:           With pNewFieldEdit
3161:             .Type = esriFieldTypeDouble
3162:             .Name = "St_Dev_WBL"
3163:             .Precision = 14
3164:             .Scale = 8
3165:           End With

        Case "Variance_WBL"
3168:           With pNewFieldEdit
3169:             .Type = esriFieldTypeDouble
3170:             .Name = "Var_WBL"
3171:             .Precision = 14
3172:             .Scale = 8
3173:           End With

        Case "Mean_WBA"
3176:           With pNewFieldEdit
3177:             .Type = esriFieldTypeDouble
3178:             .Name = "Mean_WBA"
3179:             .Precision = 14
3180:             .Scale = 8
3181:           End With
```

```
        Case "Standard Deviation_WBA"
3184:           With pNewFieldEdit
3185:             .Type = esriFieldTypeDouble
3186:             .Name = "St_Dev_WBA"
3187:             .Precision = 14
3188:             .Scale = 8
3189:           End With

        Case "Variance_WBA"
3192:           With pNewFieldEdit
3193:             .Type = esriFieldTypeDouble
3194:             .Name = "Var_WBA"
3195:             .Precision = 14
3196:             .Scale = 8
3197:           End With

        Case "Median"
3200:           With pNewFieldEdit
3201:             .Type = esriFieldTypeDouble
3202:             .Name = "Median"
3203:             .Precision = 14
3204:             .Scale = 8
3205:           End With

        Case "Mode"
3208:           With pNewFieldEdit
3209:             .Type = esriFieldTypeString
3210:             .Name = "Mode_Strng"
3211:             .length = 50
3212:           End With

3214:       End Select
3215:       Set pNewFieldsEdit.Field(lngFieldIndexCounter) = pNewField
3216:     End If
3217:   Next lngIndex
  Dim pTable As ITable

'  Set pNewField = pNewFields.Field(0)
'  MsgBox pNewField.Name
'  MsgBox pNewFields.FieldCount & " fields..."

3224:   Set pTable = Linkages.aml_func_mod.CreatedBASETable(strfilename, pNewFields)

  Dim pRow As IRow
3227:   Set pRow = pTable.CreateRow
```

```
  Dim lngFieldIndex As Long

3231:   lngFieldIndex = pTable.FindField("Unique_ID")
3232:   pRow.Value(lngFieldIndex) = 1

3234:   For lngIndex = 0 To pStatInfoArray.Count - 1
3235:     strStatistic = pStatInfoArray.Element(lngIndex)

    Select Case strStatistic
      Case "Minimum"
3239:         lngFieldIndex = pTable.FindField("Minimum")
3240:         pRow.Value(lngFieldIndex) = theMinimum

      Case "Maximum"
3243:         lngFieldIndex = pTable.FindField("Maximum")
3244:         pRow.Value(lngFieldIndex) = theMaximum

      Case "Mean"
3247:         lngFieldIndex = pTable.FindField("Mean")
3248:         pRow.Value(lngFieldIndex) = theMean

      Case "Sum"
3251:         lngFieldIndex = pTable.FindField("Sum")
3252:         pRow.Value(lngFieldIndex) = theSum

      Case "Standard Deviation"
3255:         lngFieldIndex = pTable.FindField("St_Dev")
3256:         pRow.Value(lngFieldIndex) = theStdDev

      Case "Variance"
3259:         lngFieldIndex = pTable.FindField("Variance")
3260:         pRow.Value(lngFieldIndex) = theVar

      Case "Mean_WBL"
3263:         lngFieldIndex = pTable.FindField("Mean_WBL")
3264:         pRow.Value(lngFieldIndex) = theMeanWBL

      Case "Standard Deviation_WBL"
3267:         lngFieldIndex = pTable.FindField("St_Dev_WBL")
3268:         pRow.Value(lngFieldIndex) = theStDevWBL

      Case "Variance_WBL"
3271:         lngFieldIndex = pTable.FindField("Var_WBL")
3272:         pRow.Value(lngFieldIndex) = theVarWBL

      Case "Mean_WBA"
3275:         lngFieldIndex = pTable.FindField("Mean_WBA")
```

```
3276:         pRow.Value(lngFieldIndex) = theMeanWBL

      Case "Standard Deviation_WBA"
3279:         lngFieldIndex = pTable.FindField("St_Dev_WBA")
3280:         pRow.Value(lngFieldIndex) = theStDevWBL

      Case "Variance_WBA"
3283:         lngFieldIndex = pTable.FindField("Var_WBA")
3284:         pRow.Value(lngFieldIndex) = theVarWBL

      Case "Median"
3287:         lngFieldIndex = pTable.FindField("Median")
3288:         pRow.Value(lngFieldIndex) = theMedian

      Case "Mode"
3291:         lngFieldIndex = pTable.FindField("Mode_Strng")
3292:         pRow.Value(lngFieldIndex) = theModeString

3294:     End Select
3295:   Next lngIndex
3296:   pRow.Store

  Dim pNewStandaloneTable As IStandaloneTable
3299:   Set pNewStandaloneTable = New StandaloneTable
3300:   Set pNewStandaloneTable.Table = pTable

  Dim pTableWindow2 As ITableWindow2
3303:   Set pTableWindow2 = New TableWindow

  Dim lngLeft As Long
  Dim lngTop As Long
  Dim lngRight As Long
  Dim lngBottom As Long

3310:   With pTableWindow2
3311:     Set .StandaloneTable = pNewStandaloneTable
3312:     Set .Application = m_App
3313:     .TableSelectionAction = esriSelectFeatures
3314:     .ShowAliasNamesInColumnHeadings = True
3315:     .ShowSelected = False
3316:     .Show True
3317:   End With
'
'  pTableWindow2.QueryPosition lngLeft, lngTop, lngRight, lngBottom
'  pTableWindow2.PutPosition lngLeft + 4, lngTop + 4, lngRight + 4, lngBottom + 4
'
'  MsgBox "Left = " & CStr(lngLeft) & vbCrLf & "Top = " & CStr(lngTop) & vbCrLf & "Right = " & _
```

```
'      CStr(lngRight) & vbCrLf & "Bottom = " & CStr(lngBottom)

  Dim pMxDoc As IMxDocument
3326:   Set pMxDoc = m_App.Document
  Dim pStandaloneTableCollection As IStandaloneTableCollection
3328:   Set pStandaloneTableCollection = pMxDoc.FocusMap
3329:   pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable

3331:   pMxDoc.UpdateContents

3333:   MakeNumberDBASETable = strfilename

  Exit Function
ErrorHandler:
  HandleError False, "MakeNumberDBASETable " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Private Function ReturnOutlineLetters() As String()
  On Error GoTo ErrorHandler


  Dim strOutlineLetters() As String
  ReDim strOutlineLetters(25)

3347:   strOutlineLetters(0) = "a"
3348:   strOutlineLetters(1) = "b"
3349:   strOutlineLetters(2) = "c"
3350:   strOutlineLetters(3) = "d"
3351:   strOutlineLetters(4) = "e"
3352:   strOutlineLetters(5) = "f"
3353:   strOutlineLetters(6) = "g"
3354:   strOutlineLetters(7) = "h"
3355:   strOutlineLetters(8) = "i"
3356:   strOutlineLetters(9) = "j"
3357:   strOutlineLetters(10) = "k"
3358:   strOutlineLetters(11) = "l"
3359:   strOutlineLetters(12) = "m"
3360:   strOutlineLetters(13) = "n"
3361:   strOutlineLetters(14) = "o"
3362:   strOutlineLetters(15) = "p"
3363:   strOutlineLetters(16) = "q"
3364:   strOutlineLetters(17) = "r"
3365:   strOutlineLetters(18) = "s"
3366:   strOutlineLetters(19) = "t"
3367:   strOutlineLetters(20) = "u"
3368:   strOutlineLetters(21) = "v"
```

```
3369:   strOutlineLetters(22) = "w"
3370:   strOutlineLetters(23) = "x"
3371:   strOutlineLetters(24) = "y"
3372:   strOutlineLetters(25) = "z"

3374:   ReturnOutlineLetters = strOutlineLetters


  Exit Function
ErrorHandler:
  HandleError False, "ReturnOutlineLetters " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Private Function CalcCategorySubReport(pStats As esriSystem.IVariantArray, _
    pStatInfoArray As esriSystem.IStringArray, pCloneField As IField, Optional dblAreaMultiplier As Double = 1) As String
  On Error GoTo ErrorHandler

  Dim strLetters() As String
3387:   strLetters = ReturnOutlineLetters

  Dim strReport As String
3390:   strReport = ""
  Dim lngRunningCount As Long
  Dim dblRunningProp As Double
3393:   lngRunningCount = 0
3394:   dblRunningProp = 0
  Dim lngIndex As Long
  Dim lngIndex2 As Long
  Dim strStatistic As String
  Dim pVarArray As esriSystem.IVariantArray
  Dim strVal As String
  Dim lngCount As Long
  Dim dblProportion As Double
  Dim dblSize As Double
  Dim booFieldIsNumeric As Boolean
3404:   booFieldIsNumeric = Linkages.aml_func_mod.FieldIsNumeric(pCloneField)
  Dim dblVal As Double

  Dim strFieldName As String
3408:   strFieldName = pCloneField.Name
  Dim strfilename As String
3410:   strfilename = txtOutput.Text
3411:   If Right(strfilename, 1) <> "/" And Right(strfilename, 1) <> "\" Then strfilename = strfilename & "\"
3412:   strfilename = strfilename & strFieldName & "_stats.dbf"
3413:   strfilename = Linkages.aml_func_mod.MakeUniqueFilename(strfilename)
```

```
  ' MAKE TABLE
  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit

3421:   Set pNewFields = New Fields
3422:   Set pNewFieldsEdit = pNewFields
  ' 1) Unique_ID
  ' 2) [Clone of Original Field]
  ' 3)..n) Selected Statistics

  Dim lngNewFieldCount As Long
3428:   lngNewFieldCount = 2

3430:   For lngIndex = 0 To pStatInfoArray.Count - 1
3431:     strStatistic = pStatInfoArray.Element(lngIndex)
3432:     If strStatistic = "Count" Or strStatistic = "Proportion" Or strStatistic = "Length" Or strStatistic = "Area" Then
3433:       lngNewFieldCount = lngNewFieldCount + 1
3434:     End If
3435:   Next lngIndex

3437:   pNewFieldsEdit.FieldCount = lngNewFieldCount

  Dim colFieldIndexCollection As New Collection

3441:   Set pNewField = New Field
3442:   Set pNewFieldEdit = pNewField
3443:   With pNewFieldEdit
3444:     .Name = "Unique_ID"
3445:     .Precision = 8
3446:     .Type = esriFieldTypeInteger
3447:   End With
3448:   Set pNewFieldsEdit.Field(0) = pNewField

  Dim pCloneFieldEdit As IFieldEdit
3451:   Set pCloneFieldEdit = pCloneField
3452:   pCloneFieldEdit.Name = "Value"
3453:   Set pNewFieldsEdit.Field(1) = pCloneField

  Dim lngFieldCounter As Long
3456:   lngFieldCounter = 1       ' ALREADY HAS "Unique_ID" and "VALUE"
3457:   For lngIndex = 0 To pStatInfoArray.Count - 1
3458:     Set pNewField = New Field
3459:     Set pNewFieldEdit = pNewField
3460:     strStatistic = pStatInfoArray.Element(lngIndex)
    Select Case strStatistic
```

```
      Case "Count"
3463:         With pNewFieldEdit
3464:           .Name = "Count"
3465:           .Precision = 8
3466:           .Type = esriFieldTypeInteger
3467:         End With
3468:         lngFieldCounter = lngFieldCounter + 1
3469:         Set pNewFieldsEdit.Field(lngFieldCounter) = pNewField
      Case "Proportion"
3471:         With pNewFieldEdit
3472:           .Name = "Proportion"
3473:           .Precision = 16
3474:           .Scale = 12
3475:           .Type = esriFieldTypeDouble
3476:         End With
3477:         lngFieldCounter = lngFieldCounter + 1
3478:         Set pNewFieldsEdit.Field(lngFieldCounter) = pNewField
      Case "Length"
3480:         With pNewFieldEdit
3481:           .Name = "Length"
3482:           .Precision = 16
3483:           .Scale = 12
3484:           .Type = esriFieldTypeDouble
3485:         End With
3486:         lngFieldCounter = lngFieldCounter + 1
3487:         Set pNewFieldsEdit.Field(lngFieldCounter) = pNewField
      Case "Area"
3489:         With pNewFieldEdit
3490:           .Name = "Area"
3491:           .Precision = 16
3492:           .Scale = 12
3493:           .Type = esriFieldTypeDouble
3494:         End With
3495:         lngFieldCounter = lngFieldCounter + 1
3496:         Set pNewFieldsEdit.Field(lngFieldCounter) = pNewField
3497:     End Select

3499:   Next lngIndex

  Dim pTable As ITable
3502:   Set pTable = Linkages.aml_func_mod.CreatedBASETable(strfilename, pNewFields)
  Dim pRow As IRow
  Dim pRowBuffer As IRowBuffer
3505:   Set pRowBuffer = pTable.CreateRowBuffer
  Dim pRowCursor As ICursor
3507:   Set pRowCursor = pTable.Insert(True)
```

```
3509:   colFieldIndexCollection.Add pTable.FindField("Unique_ID"), "Unique_ID"
3510:   colFieldIndexCollection.Add pTable.FindField("Value"), "Value"
3511:   For lngIndex = 0 To pStatInfoArray.Count - 1
3512:     strStatistic = pStatInfoArray.Element(lngIndex)
3513:     colFieldIndexCollection.Add pTable.FindField(strStatistic), strStatistic
3514:   Next lngIndex

  Dim strPropString As String

3518:   For lngIndex = 0 To pStats.Count - 1
3519:     Set pVarArray = pStats.Element(lngIndex)
3520:     If booFieldIsNumeric Then
3521:       dblVal = pVarArray.Element(0)
3522:       strVal = CStr(dblVal)
3523:     Else
3524:       dblVal = -999  ' JUST PLACEHOLDER; dblVal SHOULD NEVER BE EXAMINED IF USING A STRING FIELD
3525:       strVal = pVarArray.Element(0)
3526:     End If

3528:     lngCount = pVarArray.Element(1)
3529:     dblProportion = pVarArray.Element(2)
3530:     dblSize = pVarArray.Element(3)
3531:     lngRunningCount = lngRunningCount + lngCount
3532:     dblRunningProp = dblRunningProp + dblProportion

3534:     If dblProportion < 0.0000000001 Then
3535:       strPropString = "< 0.0000000001"
3536:     Else
3537:       strPropString = Format(dblProportion, "0.000000000")
3538:     End If

3540:     Set pRow = pTable.CreateRowBuffer
3541:     pRow.Value(colFieldIndexCollection.Item("Unique_ID")) = lngIndex + 1
3542:     If booFieldIsNumeric Then
3543:       pRow.Value(colFieldIndexCollection.Item("Value")) = dblVal
3544:     Else
3545:       pRow.Value(colFieldIndexCollection.Item("Value")) = strVal
3546:     End If
3547:     strVal = Linkages.aml_func_mod.SubstituteString(strVal, "\", "\\")
3548:     strReport = strReport & "    " & CStr(lngIndex + 1) & "] Value = \b " & strVal & "\b0\par" & vbCrLf

    Dim lngLetterInd As Long
3551:     lngLetterInd = -1
3552:     For lngIndex2 = 0 To pStatInfoArray.Count - 1
3553:       strStatistic = pStatInfoArray.Element(lngIndex2)
      Select Case strStatistic
        Case "Count"
```

```
3556:           lngLetterInd = lngLetterInd + 1
3557:           strReport = strReport & "        " & strLetters(lngLetterInd) & "] Count = " & CStr(lngCount) & "\par" & vbCrLf
3558:           pRow.Value(colFieldIndexCollection.Item("Count")) = lngCount
      Case "Proportion"
3560:           lngLetterInd = lngLetterInd + 1
3561:           strReport = strReport & "        " & strLetters(lngLetterInd) & "] Proportion = " & _
              strPropString & "\par" & vbCrLf
3563:           pRow.Value(colFieldIndexCollection.Item("Proportion")) = dblProportion
      Case "Length"
3565:           lngLetterInd = lngLetterInd + 1
3566:           If dblSize < 1 Then
3567:             strReport = strReport & "        " & strLetters(lngLetterInd) & "] Length = " & _
              CStr(Format(dblSize, "0.000000")) & "\par" & vbCrLf
3569:           Else
3570:             strReport = strReport & "        " & strLetters(lngLetterInd) & "] Length = " & _
              CStr(Format(dblSize, "0.00")) & "\par" & vbCrLf
3572:           End If
3573:           pRow.Value(colFieldIndexCollection.Item("Length")) = dblSize
      Case "Area"
3575:           lngLetterInd = lngLetterInd + 1
3576:           If (dblSize * dblAreaMultiplier) < 1 Then
3577:             strReport = strReport & "        " & strLetters(lngLetterInd) & "] Area = " & _
              CStr(Format((dblSize * dblAreaMultiplier), "0.000000")) & "\par" & vbCrLf
3579:           Else
3580:             strReport = strReport & "        " & strLetters(lngLetterInd) & "] Area = " & _
              CStr(Format((dblSize * dblAreaMultiplier), "0.00")) & "\par" & vbCrLf
3582:           End If
3583:           pRow.Value(colFieldIndexCollection.Item("Area")) = (dblSize * dblAreaMultiplier)
3584:       End Select
3585:     Next lngIndex2
3586:     pRowCursor.InsertRow pRow

3588:   Next lngIndex
3589:   pRowCursor.Flush

3591:   strReport = "\b --- Category Statistics ------------------\b0\par" & vbCrLf & _
      "*** Data saved to " & Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "\par" & vbCrLf & _
      "*** " & Linkages.aml_func_mod.InsertCommas(pStats.Count) & " Unique values from " & _
          Linkages.aml_func_mod.InsertCommas(lngRunningCount) _
      & " total values examined.\par" _
      & vbCrLf & strReport & "==================================================\par" & vbCrLf

  ' ADD TABLE TO DOCUMENT
  Dim pNewStandaloneTable As IStandaloneTable
3600:   Set pNewStandaloneTable = New StandaloneTable
3601:   Set pNewStandaloneTable.Table = pTable
```

```
  Dim pTableWindow2 As ITableWindow2
3604:   Set pTableWindow2 = New TableWindow

3606:   With pTableWindow2
3607:     Set .StandaloneTable = pNewStandaloneTable
3608:     Set .Application = m_App
3609:     .TableSelectionAction = esriSelectFeatures
3610:     .ShowAliasNamesInColumnHeadings = True
3611:     .ShowSelected = False
3612:     .Show True
3613:   End With

  Dim pMxDoc As IMxDocument
3616:   Set pMxDoc = m_App.Document
  Dim pStandaloneTableCollection As IStandaloneTableCollection
3618:   Set pStandaloneTableCollection = pMxDoc.FocusMap
3619:   pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable

3621:   pMxDoc.UpdateContents

3623:   CalcCategorySubReport = strReport

  Exit Function
ErrorHandler:
  HandleError False, "CalcCategorySubReport " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Private Sub cmdOpenTable_Click()
  On Error GoTo ErrorHandler


  '  ORIGINAL AVENUE CODE
  ' ADAPTED FROM OpenFeatureLayerTable.bas

  Dim anIndex As Long
  Dim pFeatureLayer As IFeatureLayer
  Dim pstandalonetable As IStandaloneTable

  Dim pTableWindow As ITableWindow

  Dim pTableWindow2 As ITableWindow2           ' NEED #2 TO FIND OPEN STANDALONE TABLES
  Dim pExistingTableWindow As ITableWindow     ' FindViaStandaloneTable RETURNS A ITableWindow

3646:   If TypeOf m_selLayer Is IFeatureLayer Then

3648:     Set pFeatureLayer = m_selLayer
```

```
3649:     Set pTableWindow = New TableWindow

    ' CHECK IF TABLE IS ALREADY OPEN
3652:     Set pTableWindow = pTableWindow.FindViaFeatureLayer(pFeatureLayer, False)

3654:     If pTableWindow Is Nothing Then
      'Associate the table and a feature layer
3656:       Set pTableWindow = New TableWindow
3657:       With pTableWindow
3658:         Set .FeatureLayer = pFeatureLayer
3659:         Set .Application = m_App
3660:         .TableSelectionAction = esriSelectFeatures
3661:         .ShowAliasNamesInColumnHeadings = True
3662:         .Show True
3663:       End With
3664:     Else
3665:       pTableWindow.Show True
3666:     End If

    'Debug.Print pLayer.Name

3670:   ElseIf TypeOf m_selLayer Is IStandaloneTable Then

3672:     Set pstandalonetable = m_selLayer
3673:     Set pTableWindow2 = New TableWindow
3674:     Set pExistingTableWindow = pTableWindow2.FindViaStandaloneTable(pstandalonetable)

    ' Check if a table already exists; if not, create one
3677:     If pExistingTableWindow Is Nothing Then

3679:       With pTableWindow2
3680:         Set .StandaloneTable = pstandalonetable
3681:         Set .Application = m_App
3682:         .TableSelectionAction = esriSelectFeatures
3683:         .ShowAliasNamesInColumnHeadings = True
3684:         .ShowSelected = False
3685:         .Show True

3687:       End With
3688:     Else
3689:       pExistingTableWindow.Show True
3690:     End If
3691:   ElseIf TypeOf m_TableFields Is IRasterLayer Then        ' IF RASTER LAYER
    Dim pRasterLayer As IRasterLayer
    Dim pRaster As IRaster
    Dim pRasterBand As IRasterBand
    Dim pRasterBandCollection As IRasterBandCollection
```

```
    Dim pRasterDataset As IRasterDataset
    Dim booHasTable As Boolean

3699:     Set pRasterLayer = m_TableFields
3700:     Set pRaster = pRasterLayer.Raster
3701:     Set pRasterBandCollection = pRaster
3702:     Set pRasterBand = pRasterBandCollection.Item(0)
3703:     Set pRasterDataset = pRasterBand.RasterDataset
3704:     pRasterBand.HasTable booHasTable

3706:     If booHasTable Then                    ' USE VAT TABLE FIELDS
      Dim pTable As ITable
3708:       Set pTable = pRasterBand.AttributeTable
3709:       Set pTableWindow = New TableWindow

      ' CHECK IF TABLE IS ALREADY OPEN
3712:       Set pTableWindow = pTableWindow.FindViaTable(pTable, False)

3714:       If pTableWindow Is Nothing Then
        'Associate the table and the raster layer
3716:         Set pTableWindow = New TableWindow
3717:         With pTableWindow
3718:           Set .Table = pTable
3719:           Set .Application = m_App
3720:           .TableSelectionAction = esriSelectFeatures
3721:           .ShowAliasNamesInColumnHeadings = True
3722:           .Show True
3723:         End With
3724:       Else
3725:         pTableWindow.Show True
3726:       End If
3727:     End If
3728:   End If


  Exit Sub
ErrorHandler:
  HandleError True, "cmdOpenTable_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdSwitchSel_Click()
  On Error GoTo ErrorHandler


  Dim anIndex As Long
```

```
  Dim pMxDoc As IMxDocument
3743:   Set pMxDoc = m_App.Document

  Dim pActiveView As IActiveView
3746:   Set pActiveView = pMxDoc.ActiveView

  Dim theCurrentSelected As IUnknown
3749:   Set theCurrentSelected = m_selLayer

  Dim pFeatureSelection As IFeatureSelection
  Dim pTableSelection As ITableSelection

  'Flag the original selection
3755:   pActiveView.PartialRefresh esriViewGeoSelection, Nothing, Nothing

3757:   If TypeOf theCurrentSelected Is IFeatureLayer Then

3759:     Set pFeatureSelection = theCurrentSelected
3760:     pFeatureSelection.SelectFeatures Nothing, esriSelectionResultXOR, False
3761:     pFeatureSelection.SelectionChanged

3763:   ElseIf TypeOf theCurrentSelected Is IStandaloneTable Then

3765:     Set pTableSelection = theCurrentSelected
3766:     pTableSelection.SelectRows Nothing, esriSelectionResultXOR, False
3767:     pTableSelection.SelectionChanged

3769:   End If

  'Flag the new selection
3772:   pActiveView.PartialRefresh esriViewGeoSelection, Nothing, Nothing

3774:   Call DisplayNumSelected


  Exit Sub
ErrorHandler:
  HandleError True, "cmdSwitchSel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Click()
  On Error GoTo ErrorHandler


3786:   Call DisplayNumSelected
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "Form_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Public Function GetPath() As String
  On Error GoTo ErrorHandler


  Dim pTemplates As ITemplates
  Dim lTempCount As Long

3801:   Set pTemplates = m_App.Templates
3802:   lTempCount = pTemplates.Count

  ' The document is always the last item
3805:   GetPath = pTemplates.Item(lTempCount - 1)


  Exit Function
ErrorHandler:
  HandleError True, "GetPath " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Private Sub Form_Load()
  On Error GoTo ErrorHandler

3816:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
'  MsgBox "Form_Load Script about to start..."  ' ************************************************************

3819:    Set Anchors = New AnchorObjectList  ' Create new instance
3820:    With Anchors
3821:       With .Item(imgCorrIcon)
3822:         .SetAnchors enumSizeEnd, enumStartSize
3823:       End With
3824:       With .Item(lblSummarize)
3825:         .SetAnchors enumStartEnd, enumStartSize
3826:       End With
3827:       With .Item(lblSum2)
3828:         .SetAnchors enumStartEnd, enumStartSize
3829:       End With
3830:       With .Item(frmGeneral)
3831:          .SetAnchors enumStartEnd, enumStartEnd
3832:       End With
3833:       With .Item(treeFields)
```

```
3834:          .SetAnchors enumStartEnd, enumStartEnd
3835:       End With
3836:       With .Item(lbl3)
3837:          .SetAnchors enumStartSize, enumSizeEnd
3838:       End With
3839:       With .Item(txtOutput)
3840:          .SetAnchors enumStartEnd, enumSizeEnd
3841:       End With
3842:       With .Item(cbxLayer)
3843:          .SetAnchors enumStartEnd, enumStartSize
3844:       End With
'      With .Item(cbxFieldName)
'         .SetAnchors enumStartEnd, enumStartSize
'      End With
3848:       With .Item(cmdGetFile)
3849:          .SetAnchors enumSizeEnd, enumSizeEnd
3850:       End With
3851:       With .Item(chkSumSelected)
3852:          .SetAnchors enumStartSize, enumSizeEnd
3853:       End With
3854:       With .Item(cmdOpenTable)
3855:          .SetAnchors enumStartSize, enumSizeEnd
3856:       End With
3857:       With .Item(cmdSwitchSel)
3858:          .SetAnchors enumStartSize, enumSizeEnd
3859:       End With
3860:       With .Item(cmdHelp)
3861:          .SetAnchors enumSizeEnd, enumSizeEnd
3862:       End With
3863:       With .Item(cmdOK)
3864:          .SetAnchors enumSizeEnd, enumSizeEnd
3865:       End With
3866:       With .Item(cmdCancel)
3867:          .SetAnchors enumSizeEnd, enumSizeEnd
3868:       End With
3869:       With .Item(imgCornerBars)
3870:          .SetAnchors enumSizeEnd, enumSizeEnd
3871:       End With
3872:       .Form = Me  ' Set form reference (suggested to be last step)
3873:    End With

'  MsgBox "Anchors Finished Setting"  ' ************************************************************
  Dim newUid As New uID
3877:   newUid.Value = "Linkages.Extension"
3878:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
3880:   Set ext = m_ExtensionConfig
```

```
  ' STATS STUFF
3883:   m_excludeString = "-9999"
3884:   m_ShouldExclude = False
'  m_ShouldAdvanced = False
3886:   m_ShouldDoAll = True
3887:   If m_ReportForm Is Nothing Then
3888:     Set m_ReportForm = New Linkages.frmReport_modal
3889:   End If
3890:   m_NumDecPlaces = 2

 ' Set m_SumHelp = New frmSumHelp

3894:   Set treeFields.ImageList = imgImageList

'  MsgBox "Finished Assigning Imagelist to TreeControl..."  ' ************************************************************

'  cbxFieldName.Clear
3899:   cbxLayer.Clear

  ' CREATE COLLECTION OF NAMES, WITH INDEX VALUES THAT CORRESPOND TO ACTUAL LAYER/TABLE OBJECTS
3902:   Set m_TableLayerNames = New Collection
  Dim pFeatureLayer As IFeatureLayer
  Dim pFeatureClass As IFeatureClass
  Dim pGeometryType As esriGeometryType
  Dim pLayerForValid As ILayer
  Dim pRasterLayer As IRasterLayer
  Dim pRasterBandCollection As IRasterBandCollection
  Dim pRasterBand As IRasterBand
  Dim booHasTable As Boolean
  Dim pstandalonetable As IStandaloneTable
  Dim pStTableForValid As IStandaloneTable
  Dim pUnknown As IUnknown
  Dim intIndex As Integer
  Dim intSelectedIndex As String
  Dim strCurrentIndex As String
  Dim strLayerNames() As String
  ReDim strLayerNames(m_TableLayers.Count - 1)


'  Dim strReport As String
'  Dim anIndex As Long
'  For anIndex = 1 To m_TableLayers.Count
'    strReport = "Layer #" & CStr(anIndex) & " is Nothing = " & CStr(m_TableLayers.Item(anIndex) Is Nothing)
'  Next anIndex
'  MsgBox strReport
```

```
'  MsgBox "Finished Dimensioning Variables..."  ' ************************************************************

3930:   intSelectedIndex = -1

  Dim strShapeName As String
  Dim strListName As String

3935:   For intIndex = 0 To m_TableLayers.Count - 1

3937:     strCurrentIndex = CStr(intIndex)
3938:     Set pUnknown = m_TableLayers.Item(strCurrentIndex)

'    MsgBox intIndex & vbCrLf & "pUnknown is nothing? " & CStr(pUnknown Is Nothing) & vbCrLf & _
              "Count = " & m_TableLayers.Count & vbCrLf & "Item 1 is nothing:  " & CStr(m_TableLayers.Item(1) Is Nothing) & _
              vbCrLf & "Item 2 is nothing:  " & CStr(m_TableLayers.Item(2) Is Nothing)

3944:     If pUnknown Is m_CurrentSelected Then
3945:       intSelectedIndex = intIndex
3946:     End If

3948:     If TypeOf pUnknown Is IFeatureLayer Then

3950:       Set pFeatureLayer = pUnknown

3952:       Set pFeatureClass = pFeatureLayer.FeatureClass
3953:       pGeometryType = pFeatureClass.ShapeType
3954:       strShapeName = Linkages.aml_func_mod.ReturnShapeName(pGeometryType)

3956:       strListName = CStr(intIndex + 1) & "] " & pFeatureLayer.Name & "  (" & _
              strShapeName & " Dataset)"

3959:       m_TableLayerNames.Add pFeatureLayer.Name, strCurrentIndex
3960:       strLayerNames(intIndex) = strListName

3962:     ElseIf TypeOf pUnknown Is IStandaloneTable Then

3964:       Set pstandalonetable = pUnknown
3965:       m_TableLayerNames.Add pstandalonetable.Name, strCurrentIndex
3966:       strLayerNames(intIndex) = CStr(intIndex + 1) & "] " & pstandalonetable.Name & "  (" & _
              "Standalone Table)"

3969:     ElseIf TypeOf pUnknown Is IRasterLayer Then

3971:       Set pRasterLayer = pUnknown
3972:       m_TableLayerNames.Add pRasterLayer.Name, strCurrentIndex
3973:       strLayerNames(intIndex) = CStr(intIndex + 1) & "] " & pRasterLayer.Name & "  (" & _
              "Raster Dataset)"
```

```
3976:     End If
3977:   Next intIndex

'  MsgBox "Finished Looking for Selected Layer..."  ' ************************************************************

  ' FILL LAYER COMBOBOX WITH NAMES
3982:   For intIndex = LBound(strLayerNames) To UBound(strLayerNames)
3983:     cbxLayer.AddItem strLayerNames(intIndex)
3984:   Next intIndex

'  MsgBox "Finished Filling Layer Combobox..."  ' ************************************************************

3988:   If Not m_CurrentSelected Is Nothing Then
3989:     cbxLayer.ListIndex = intSelectedIndex
3990:   Else
3991:     cbxLayer.ListIndex = 0
3992:   End If

'  MsgBox "Finished Filling Layer Combobox..."  ' ************************************************************

'  Call cbxLayer_Click      ' APPARENTLY UNNECESSARY.  SETTING THE LISTINDEX PROPERTY > -1 FORCES CLICK EVENT


  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String
  Dim strUserName As String

4007:   strDirPath = ext.ClipDirectoryPath
4008:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
4009:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
4010:   End If
4011:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
4012:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_App))
4013:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
4014:   End If

4016:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then
4017:     strDirPath = strDirPath & "\"
4018:   End If

4020:   txtOutput.Text = strDirPath
  ' FILL UP OUTPUT FILENAME BOX
```

```
  ' FIRST TRY TO FIND USER'S "MY DOCUMENTS" FOLDER:
  ' 1) c:\Documents and Settings\[User Name]\My Documents
  ' 2) IF NO LUCK THEN c:\Documents and Settings\My Documents
  ' 3) IF NO LUCK GO TO TEMP DIRECTORY

'  Dim strDirPath As String
'  Dim strUserName As String
'
'  strUserName = aml_func_mod.GetTheUserName
'  strDirPath = "c:\Documents and Settings\" & strUserName & "\My Documents\"
'
''  MsgBox "Finished Looking for UserName MyDocuments..."  ' ************************************************************
'
'  Dim boolMediaFileExists As Boolean
'  boolMediaFileExists = Not Dir$(strDirPath) = ""
'
'  If Not boolMediaFileExists Then
'    strDirPath = "c:\Documents and Settings\My Documents\"
'    boolMediaFileExists = Not Dir$(strDirPath) = ""
''    MsgBox "Finished Looking for General MyDocuments..."  ' ************************************************************
'  End If
'
'  If Not boolMediaFileExists Then
'    strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.TempPathLocation)
''    MsgBox "Finished Looking for Temporary Path..."  ' ************************************************************
'  End If
'
'  Dim strSuggestFileName As String
'  strSuggestFileName = aml_func_mod.MakeUniqueFilename(strDirPath & "summary.dbf")
'
'  txtOutput.Text = strSuggestFileName

4054:   Call Fill_TreeView

'  MsgBox "Finished Calling Fill_TreeView..."  ' ************************************************************

4058:   Call DisplayNumSelected
4059:   Call CheckOK

'  MsgBox "Finished Calling DisplayNumSelected..."  ' ************************************************************


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  On Error GoTo ErrorHandler


4074:   Call DisplayNumSelected


  Exit Sub
ErrorHandler:
  HandleError True, "Form_MouseDown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Resize()
  On Error GoTo ErrorHandler


'  Dim theFrmHeight As Long
'  Dim theFrmWidth As Long
'  theFrmHeight = Me.Height
'  theFrmWidth = Me.Width
'
'  'Debug.Print "Height = " & m_theFrmHeight & ",   Width = " & m_theFrmWidth
'  ' IF "SIZABLE" (OPTION 2) THEN ADD 45 TO ALL HEIGHTS
'  If theFrmHeight > 4500 Then
'      frmGeneral.Height = theFrmHeight - 1425
'      treeFields.Height = theFrmHeight - 4065
'      lbl3.Top = theFrmHeight - 2580
'      txtOutput.Top = theFrmHeight - 2325
'      cmdGetFile.Top = theFrmHeight - 2385
'      chkSumSelected.Top = theFrmHeight - 1770
'      cmdOpenTable.Top = theFrmHeight - 810
'      cmdOpenStats.Top = theFrmHeight - 810
'      cmdSwitchSel.Top = theFrmHeight - 810
'      cmdHelp.Top = theFrmHeight - 810
'      cmdOK.Top = theFrmHeight - 810
'      cmdCancel.Top = theFrmHeight - 810
'  End If
'
'  If m_theFrmWidth > 5685 Then
'      frmGeneral.Width = theFrmWidth - 225
'      treeFields.Width = theFrmWidth - 855
'      txtOutput.Width = theFrmWidth - 1335
'      cbxFieldName.Width = theFrmWidth - 825
'      cmdGetFile.Left = theFrmWidth - 900
'  End If
```

```
'
'  imgCornerBars.Top = theFrmHeight - 630
'  imgCornerBars.Left = theFrmWidth - 345


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Resize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler


'  If Not m_SumHelp Is Nothing Then
'    Set m_SumHelp = Nothing
'  End If
4132:   Set m_selLayer = Nothing
4133:   Set m_TableFields = Nothing

4135:   Set m_Return = Nothing
'  Set m_SumHelp = Nothing
4137:   Set m_App = Nothing

4139:   If Not m_ReportForm Is Nothing Then
4140:     Set m_ReportForm = Nothing
4141:   End If


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub



Private Sub treeFields_NodeClick(ByVal Node As MSComctlLib.Node)
  On Error GoTo ErrorHandler


'  Debug.Print Node.Text & "   " & Node.Image & "  Tag: " & Node.Tag
4156:   If Node.Tag = "Stat" Then
4157:     If Node.Image = 1 Then
4158:       Node.Image = 2
4159:       Node.SelectedImage = 2
```

```
4160:     Else
4161:       Node.Image = 1
4162:       Node.SelectedImage = 1
4163:     End If
4164:   End If
'  Debug.Print Node.Text & "   " & Node.Image
'  treeFields.Refresh

4168:   Call CheckOK

  Exit Sub
ErrorHandler:
  HandleError True, "treeFields_NodeClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub CheckOK()
  On Error GoTo ErrorHandler


  ' ENABLE OK BUTTON IF ANY STATS ARE CHECKED
  Dim pNodes As Nodes
4181:   Set pNodes = treeFields.Nodes
  Dim anIndex As Long
  Dim pNode As Node
  Dim booStatOK As Boolean
4185:   booStatOK = False
4186:   For anIndex = 1 To pNodes.Count
4187:     Set pNode = pNodes.Item(anIndex)
4188:     If pNode.Image = 2 Then
4189:       booStatOK = True
4190:       Exit For
4191:     End If
4192:   Next anIndex

4194:   cmdOK.Enabled = booStatOK


  Exit Sub
ErrorHandler:
  HandleError False, "CheckOK " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub DisplayNumSelected()
  On Error GoTo ErrorHandler
```

```
' saguaro.ModQueryDialogDisplayNumSelected

  ' IDENTIFY CONTROLS
  Dim pFeatureLayer As IFeatureLayer
  Dim pstandalonetable As IStandaloneTable
  Dim pRasterLayer As IRasterLayer
  Dim pTableSelection As ITableSelection
  Dim pTable As ITable
  Dim pLayer As Variant

4216:   Set pLayer = m_selLayer
4217:   If TypeOf pLayer Is IRasterLayer Then
4218:     Set pRasterLayer = pLayer
4219:     Set pTable = Nothing
4220:     Set pTableSelection = Nothing
4221:     chkSumSelected.Caption = "Statistics will be calculated for entire grid..."
4222:     chkSumSelected.Value = 2
4223:     chkSumSelected.Enabled = False
    Exit Sub
4225:   ElseIf TypeOf pLayer Is IFeatureLayer Then
4226:     Set pFeatureLayer = pLayer
4227:     Set pTable = pFeatureLayer
4228:     Set pTableSelection = pFeatureLayer
4229:   Else
4230:     Set pstandalonetable = pLayer
4231:     Set pTable = pstandalonetable
4232:     Set pTableSelection = pstandalonetable
4233:   End If

  Dim lngSelCount As Long
4236:   lngSelCount = pTableSelection.SelectionSet.Count

  Dim lngDefCount As Long
4239:   lngDefCount = pTable.RowCount(Nothing)

4241:   chkSumSelected.Caption = "Calculate Statistics on Selected Records Only (" & aml_func_mod.InsertCommas(lngSelCount) & _
      " of " & aml_func_mod.InsertCommas(lngDefCount) & " selected)"

4244:   If lngSelCount = 0 Then
4245:     chkSumSelected.Value = 2
4246:     chkSumSelected.Enabled = False
4247:   Else
4248:     chkSumSelected.Value = 1
'    If chkSumSelected.Value = 2 Then
'      chkSumSelected.Value = 0
'    End If
4252:     chkSumSelected.Enabled = True
```

```
4253:   End If


  Exit Sub
ErrorHandler:
  HandleError False, "DisplayNumSelected " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 2: frmAbout.frm

```
VERSION 5.00
Object = "{3B7C8863-D78F-101B-B9B5-04021C009402}#1.2#0"; "RICHTX32.OCX"
Begin VB.Form frmAbout
   BackColor       =   &H00444444&
   Caption         =   "About Corridor Designer:"
   ClientHeight    =   5805
   ClientLeft      =   2355
   ClientTop       =   1950
   ClientWidth     =   6195
   ClipControls    =   0   'False
   Icon            =   "frmAbout.frx":0000
   LinkTopic       =   "Form2"
   LockControls    =   -1  'True
   ScaleHeight     =   5805
   ScaleWidth      =   6195
   StartUpPosition =   2  'CenterScreen
   Begin VB.CommandButton cmdManual
      Caption         =   "Open Manual"
      Height          =   345
      Left            =   4710
      TabIndex        =   2
      Top             =   5040
      Width           =   1245
   End
   Begin RichTextLib.RichTextBox rtfAbout
      Height          =   2370
      Left            =   105
      TabIndex        =   4
      Top             =   1365
      Width           =   5970
      _ExtentX        =   10530
      _ExtentY        =   4180
      _Version        =   393217
      Enabled         =   -1  'True
```

```
   ReadOnly        =   -1  'True
   ScrollBars      =   2
   TextRTF         =   $"frmAbout.frx":038A
End
Begin VB.CommandButton cmdOK
   Cancel          =   -1  'True
   Caption         =   "Close"
   Height          =   345
   Left            =   4695
   MaskColor       =   &H00DDE7EC&
   TabIndex        =   0
   Top             =   4140
   Width           =   1245
End
Begin VB.CommandButton cmdSysInfo
   Caption         =   "&System Info..."
   Height          =   345
   Left            =   4710
   TabIndex        =   1
   Top             =   4590
   Width           =   1245
End
Begin VB.Label lblEmily
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Emily Garding:  emily@CorridorDesign.org"
   BeginProperty Font
      Name            =   "Trebuchet MS"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00004000&
   Height          =   240
   Left            =   1155
   MousePointer    =   99  'Custom
   TabIndex        =   11
   Top             =   5310
   Width           =   3120
End
Begin VB.Label lblJeff
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Jeff Jenness:  jeff@CorridorDesign.org"
```

```
   BeginProperty Font
      Name            =   "Trebuchet MS"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00004000&
   Height          =   240
   Left            =   1350
   MousePointer    =   99  'Custom
   TabIndex        =   10
   Top             =   5055
   Width           =   2925
End
Begin VB.Label lblDan
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Dan Majka:  dan@CorridorDesign.org"
   BeginProperty Font
      Name            =   "Trebuchet MS"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00004000&
   Height          =   240
   Left            =   1485
   MousePointer    =   99  'Custom
   TabIndex        =   9
   Top             =   4800
   Width           =   2790
End
Begin VB.Label lblPaul
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Paul Beier:  Paul.Beier@nau.edu"
   BeginProperty Font
      Name            =   "Trebuchet MS"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
```

```
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00004000&
   Height          =   240
   Left            =   1905
   MousePointer    =   99  'Custom
   TabIndex        =   8
   Top             =   4545
   Width           =   2370
End
Begin VB.Label lblContactUs
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Contact us:"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   195
   Left            =   330
   TabIndex        =   7
   Top             =   4530
   Width           =   990
End
Begin VB.Label lblSES2
   AutoSize        =   -1  'True
   BackColor       =   &H00DDE7EC&
   BackStyle       =   0  'Transparent
   Caption         =   "www.CorridorDesign.org"
   BeginProperty Font
      Name            =   "Trebuchet MS"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00FF0000&
   Height          =   240
```

```
   Left            =   2400
   MousePointer    =   99  'Custom
   TabIndex        =   6
   ToolTipText     =   "Link to Corridor Designer web site..."
   Top             =   4275
   Width           =   1875
End
Begin VB.Label lblSES1
   AutoSize        =   -1  'True
   BackColor       =   &H00DDE7EC&
   BackStyle       =   0  'Transparent
   Caption         =   "Corridor Designer ©"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   195
   Left            =   330
   TabIndex        =   5
   Top             =   4260
   Width           =   1695
End
Begin VB.Image imgSaguaro
   Height          =   1275
   Left            =   105
   Picture         =   "frmAbout.frx":0429
   Top             =   75
   Width           =   5880
End
Begin VB.Image imgCornerBars
   Height          =   225
   Left            =   5970
   Picture         =   "frmAbout.frx":18AE5
   Top             =   5595
   Width           =   225
End
Begin VB.Label lblVersion
   AutoSize        =   -1  'True
   BackColor       =   &H00DDE7EC&
   BackStyle       =   0  'Transparent
   Caption         =   "Version:"
   BeginProperty Font
```

```
         Name            =   "MS Sans Serif"
         Size            =   8.25
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00000000&
      Height          =   195
      Left            =   330
      TabIndex        =   3
      Top             =   3990
      Width           =   705
   End
   Begin VB.Image imgBottomInset
      Height          =   1965
      Left            =   75
      Picture         =   "frmAbout.frx":18B3E
      Top             =   3780
      Width           =   4455
   End
End
Attribute VB_Name = "frmAbout"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

' Reg Key Security Options...
Const READ_CONTROL = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + _
                       KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
                       KEY_NOTIFY + KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                         ' Unicode nul terminated string
Const REG_DWORD = 4                      ' 32-bit number
```

```
Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal
ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String,
ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

Private Declare Function ShellExecute Lib "shell32.dll" Alias _
           "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As String, _
           ByVal lpFile As String, ByVal lpParameters As String, _
           ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Private Anchors As AnchorObjectList  ' Main anchor control object

Private m_lngGreen As Long


Private Sub cmdManual_Click()

  Dim strPath As String
43:   strPath = App.Path & "\help"
44:   Call Linkages.MyGeneralOperations.OpenDoc("CD_Evaluation_Tools.pdf", strPath)

End Sub

Private Sub cmdSysInfo_Click()
49:   Call StartSysInfo
End Sub

Private Sub cmdOK_Click()
53:   Unload Me
End Sub

Private Sub Form_Load()

58:    SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
59:    Set Anchors = New AnchorObjectList  ' Create new instance
60:    With Anchors
61:       With .Item(cmdOK)
62:          .SetAnchors enumSizeEnd, enumSizeEnd
63:       End With
64:       With .Item(cmdSysInfo)
```

```
65:          .SetAnchors enumSizeEnd, enumSizeEnd
66:       End With
67:       With .Item(cmdManual)
68:          .SetAnchors enumSizeEnd, enumSizeEnd
69:       End With
70:       With .Item(imgCornerBars)
71:          .SetAnchors enumSizeEnd, enumSizeEnd
72:       End With
73:       With .Item(lblVersion)
74:          .SetAnchors enumStartSize, enumSizeEnd
75:       End With
76:       With .Item(lblContactUs)
77:          .SetAnchors enumStartSize, enumSizeEnd
78:       End With
79:       With .Item(lblPaul)
80:          .SetAnchors enumStartSize, enumSizeEnd
81:       End With
82:       With .Item(lblDan)
83:          .SetAnchors enumStartSize, enumSizeEnd
84:       End With
85:       With .Item(lblJeff)
86:          .SetAnchors enumStartSize, enumSizeEnd
87:       End With
88:       With .Item(lblEmily)
89:          .SetAnchors enumStartSize, enumSizeEnd
90:       End With
91:       With .Item(lblSES1)
92:          .SetAnchors enumStartSize, enumSizeEnd
93:       End With
94:       With .Item(lblSES2)
95:          .SetAnchors enumStartSize, enumSizeEnd
96:       End With
97:       With .Item(rtfAbout)
98:          .SetAnchors enumStartEnd, enumStartEnd
99:       End With
100:       With .Item(imgSaguaro)
101:          .SetAnchors enumStartSize, enumStartSize
102:       End With
103:       With .Item(imgBottomInset)
104:          .SetAnchors enumStartSize, enumSizeEnd
105:       End With

107:       .Form = Me  ' Set form reference (suggested to be last step)
108:    End With

110:    m_lngGreen = RGB(0, 64, 0)
111:    cmdOK.BackColor = RGB(236, 231, 231)
```

```
112:    cmdSysInfo.BackColor = RGB(236, 231, 231)
113:    cmdManual.BackColor = RGB(236, 231, 231)

'   With Anchors
'      With .Item(cmdOK)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(cmdSysInfo)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(imgCornerBars)
'         .SetAnchors enumSizeEnd, enumSizeEnd
'      End With
'      With .Item(lblVersion)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(lblWebSite)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(rtfAbout)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(lglGIS)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(imgSaguaro)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      With .Item(picJenLogo)
'         .SetAnchors enumStartSize, enumStartSize
'      End With
'      .Form = Me  ' Set form reference (suggested to be last step)
'   End With

146:     Me.Caption = "About Corridor Designer"
147:     lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
  '  lblTitle.Caption = App.Title
'    rtfAbout.TextRTF = "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & _
'      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\f0\fs20 Lots of cool text describing Corridor Designer\par" & _
'      "\par" & _
'      "\b Everything you ever wanted to know about designing wildlife corridors with GIS*\b0\par" & _
'      "\par" & _
'      "*\fs16 Ok, maybe not everything. Our goal is to transfer everything we've learned about designing " & _
'      "wildlife corridors to the public to facilitate better conservation, science, and dialogue. \par}"

    Dim strRTFText As String
158:     strRTFText = "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}{\f1\fnil\fcharset2 Symbol;}}" &
```

```
vbCrLf & _
    "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\b\f0\fs20 Everything you ever wanted to know about designing wildlife
corridors with GIS*\b0\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "*\fs16 Ok, maybe not everything. Our goal is to transfer everything we've learned about designing wildlife corridors to the
public to facilitate better conservation, science, and dialogue. \par" & vbCrLf & _
    "\b\fs20\par" & vbCrLf & _
    "About CorridorDesigner\b0\par" & vbCrLf & _
    "\pard\fs16 The impetus for this project came from the interest shown to Dan Majka's response to a CONSGIS listserv question about
available software for designing wildlife corridors. We realized that although several groups have used GIS to design wildlife
linkages, the necessary conceptual steps and specific GIS methods have not been documented adequately.\fs20\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "\pard\qc\b About the Team\b0\par" & vbCrLf & _
    "\pard{\pntext\f1\'B7\tab}{\*\pn\pnlvlblt\pnf1\pnindent0{\pntxtb\'B7}}\b\fs16 Paul Beier\b0  has been researching and promoting
corridors for nearly 20 years. Paul has used these methods in collaborative, science-based efforts in southern California and Arizona,
and has supervised a Master\rquote s thesis evaluating the sensitivity of these GIS methods to uncertainty in the biological
inputs.\par" & vbCrLf & _
    "\b{\pntext\f1\'B7\tab}Dan Majka\b0  wrote the CorridorDesigner Toolbox for creating habitat and corridor analyses, designed the
webpage, and is responsible for all web content, tools documentation, and tutorials. Previously, he produced 8 linkage designs
throughout Arizona in 2005-2006 for the Arizona Missing Linkages project, and modeled the habitat distribution of 41 bird species in
Costa Rica for his MS Research at Purdue University.\par" & vbCrLf & _
    "\b{\pntext\f1\'B7\tab}Jeff Jenness\b0  wrote the CorridorDesigner corridor evaluation tools extension for ArcMap. Jeff is a GIS
programmer and wildlife biologist with over a decade of experience with universities, businesses and governmental agencies around the
world. His ArcView tools have been downloaded over 200,000 times from his website and the ESRI ArcScripts site. He also collaborates
on developing teaching tools to convey GIS concepts to high school and college students.\par" & vbCrLf & _
    "\b{\pntext\f1\'B7\tab}Emily Garding\b0  is the lead GIS Analyst for the Arizona Missing Linkages project in 2006-2007.
Previously, she spent several years as a biologist at Grand Canyon National Park combining field research, GPS, and GIS methods to
investigate carnivore movement patterns, habitat use, prey selection, and population dynamics. Her interests include preserving
habitat connectivity and getting wildlife safely across roads. For questions regarding the Arizona Missing Linkages project, contact
Emily.\fs20\par" & vbCrLf & _
    "\pard\par" & vbCrLf & _
    "\pard\qc\b Acknowledgements\b0\par" & vbCrLf & _
    "\pard\fs16 We are funded by a generous grant from the Environmental Research, Development and Education for the New Economy
(ERDENE) initiative from Northern Arizona University.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "Our approach was initially developed during 2001-2006 for South Coast Missing Linkages, a set of 16 linkage designs in southern
California (draft & final designs at scwildlands.org). Kristeen Penrod, Clint Caba\'f1ero, Wayne Spencer, and Claudia Luke made
enormous contributions to SCML and the procedures in CorridorDesigner. The designs produced by South Coast Wildlands were supported by
The Wildlands Conservancy, Resources Legacy Fund Foundation, The California Resources Agency, US Forest Service, The Nature
Conservancy, California State Parks, US National Park Service, Santa Monica Mountains Conservancy, Conservation Biology Institute, San
Diego State University Field Stations, Southern California Wetlands Recovery Project, Mountain Lion Foundation, California State Parks
Foundation, Environment Now, Anza Borrego Foundation, Summerlee Foundation, Zoological Society of San Diego, and South Coast
Wildlands.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "The Arizona Missing Linkages Project was supported by Arizona Game and Fish Department, Arizona Department of Transportation,
U.S. Fish and Wildlife Service, U.S. Forest Service, Federal Highway Administration, Bureau of Land Management, Sky Island Alliance,
Wildlands Project, and Northern Arizona University.\par" & vbCrLf & _
```

```
    "\par" & vbCrLf & _
    "Over the past 5 years, we discussed these ideas with Andrea Atkinson, Todd Bayless, Clint Caba\'f1ero, Liz Chattin, Matt Clark,
Kevin Crooks, Kathy Daly, Brett Dickson, Robert Fisher, Emily Garding, Madelyn Glickfeld, Nick Haddad, Steve Loe, Travis Longcore,
Claudia Luke, Lisa Lyren, Brad McRae, Scott Morrison, Shawn Newell, Reed Noss, Kristeen Penrod, E.J. Remson, Seth Riley, Esther Rubin,
Ray Sauvajot, Dan Silver, Jerre Stallcup, and Mike White. We especially thank the many government agents, conservationists, and
funders who conserve linkages and deserve the best possible science. \fs20\par" & vbCrLf & _
    "}"

182:     rtfAbout.TextRTF = strRTFText

184:     lblSES2.MouseIcon = LoadResPicture(102, vbResCursor)
185:     lblPaul.MouseIcon = LoadResPicture(102, vbResCursor)
186:     lblDan.MouseIcon = LoadResPicture(102, vbResCursor)
187:     lblJeff.MouseIcon = LoadResPicture(102, vbResCursor)
188:     lblEmily.MouseIcon = LoadResPicture(102, vbResCursor)

End Sub

Public Sub StartSysInfo()
    On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String

    ' Try To Get System Info Program Path\Name From Registry...
199:     If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO, SysInfoPath) Then
    ' Try To Get System Info Program Path Only From Registry...
201:     ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC, gREGVALSYSINFOLOC, SysInfoPath) Then
        ' Validate Existance Of Known 32 Bit File Version
203:         If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
204:             SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

        ' Error - File Can Not Be Found...
207:         Else
208:             GoTo SysInfoErr
209:         End If
    ' Error - Registry Entry Can Not Be Found...
211:     Else
212:         GoTo SysInfoErr
213:     End If

215:     Call Shell(SysInfoPath, vbNormalFocus)

    Exit Sub
SysInfoErr:
219:     MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub
```

```
Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As String, ByRef KeyVal As String) As Boolean
    Dim i As Long                                           ' Loop Counter
    Dim rc As Long                                          ' Return Code
    Dim hKey As Long                                        ' Handle To An Open Registry Key
    Dim hDepth As Long                                      '
    Dim KeyValType As Long                                  ' Data Type Of A Registry Key
    Dim tmpVal As String                                    ' Tempory Storage For A Registry Key Value
    Dim KeyValSize As Long                                  ' Size Of Registry Key Variable
    '------------------------------------------------------------
    ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
    '------------------------------------------------------------
233:     rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open Registry Key

235:     If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle Error...

237:     tmpVal = String$(1024, 0)                             ' Allocate Variable Space
238:     KeyValSize = 1024                                       ' Mark Variable Size

    '------------------------------------------------------------
    ' Retrieve Registry Key Value...
    '------------------------------------------------------------
243:     rc = RegQueryValueEx(hKey, SubKeyRef, 0, _
                        KeyValType, tmpVal, KeyValSize)    ' Get/Create Key Value

246:     If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle Errors

248:     If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then           ' Win95 Adds Null Terminated String...
249:         tmpVal = Left(tmpVal, KeyValSize - 1)               ' Null Found, Extract From String
250:     Else                                                    ' WinNT Does NOT Null Terminate String...
251:         tmpVal = Left(tmpVal, KeyValSize)                   ' Null Not Found, Extract String Only
252:     End If
    '------------------------------------------------------------
    ' Determine Key Value Type For Conversion...
    '------------------------------------------------------------
    Select Case KeyValType                                  ' Search Data Types...
    Case REG_SZ                                             ' String Registry Key Data Type
258:         KeyVal = tmpVal                                    ' Copy String Value
    Case REG_DWORD                                          ' Double Word Registry Key Data Type
260:         For i = Len(tmpVal) To 1 Step -1                   ' Convert Each Bit
261:             KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))   ' Build Value Char. By Char.
262:         Next
263:         KeyVal = Format$("&h" + KeyVal)                    ' Convert Double Word To String
264:     End Select

266:     GetKeyValue = True                                     ' Return Success
267:     rc = RegCloseKey(hKey)                                 ' Close Registry Key
```

```
    Exit Function                                           ' Exit

GetKeyError:      ' Cleanup After An Error Has Occured...
271:     KeyVal = ""                                            ' Set Return Val To Empty String
272:     GetKeyValue = False                                    ' Return Failure
273:     rc = RegCloseKey(hKey)                                 ' Close Registry Key
End Function

Private Sub Text1_Change()

End Sub


Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

283:   lblSES2.ForeColor = vbBlue
284:   lblPaul.ForeColor = m_lngGreen
285:   lblDan.ForeColor = m_lngGreen
286:   lblJeff.ForeColor = m_lngGreen
287:   lblEmily.ForeColor = m_lngGreen

End Sub

Private Sub imgBottomInset_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

293:   lblSES2.ForeColor = vbBlue
294:   lblPaul.ForeColor = m_lngGreen
295:   lblDan.ForeColor = m_lngGreen
296:   lblJeff.ForeColor = m_lngGreen
297:   lblEmily.ForeColor = m_lngGreen

End Sub

Private Sub imgCloseOut_Click()
302:   Unload Me
End Sub


Private Sub lblPaul_Click()
307:   Call ShellExecute(0, vbNullString, "mailto:Paul.Beier@nau.edu", vbNullString, "", 1)
End Sub

Private Sub lblPaul_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
311:   lblPaul.ForeColor = vbRed
End Sub

Private Sub lblDan_Click()
```

```
315:   Call ShellExecute(0, vbNullString, "mailto:dan@corridordesign.org", vbNullString, "", 1)
End Sub

Private Sub lblDan_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
319:   lblDan.ForeColor = vbRed
End Sub

Private Sub lblJeff_Click()
323:   Call ShellExecute(0, vbNullString, "mailto:jeff@corridordesign.org", vbNullString, "", 1)
End Sub

Private Sub lblJeff_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
327:   lblJeff.ForeColor = vbRed
End Sub

Private Sub lblEmily_Click()
331:   Call ShellExecute(0, vbNullString, "mailto:emily@corridordesign.org", vbNullString, "", 1)
End Sub

Private Sub lblEmily_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
335:   lblEmily.ForeColor = vbRed
End Sub

Private Sub lblSES2_Click()
339:   Call ShellExecute(0, vbNullString, "http://www.corridordesign.org", vbNullString, "", 1)
End Sub

Private Sub lblSES2_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
343:   lblSES2.ForeColor = vbRed
End Sub
```

Form 3: frmBottleneck.frm

```
VERSION 5.00
Object = "{3B7C8863-D78F-101B-B9B5-04021C009402}#1.2#0"; "RICHTX32.OCX"
Begin VB.Form frmBottleneck
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Describe Bottlenecks:"
   ClientHeight    =   4470
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   9945
   Icon            =   "frmBottleneck.frx":0000
   LinkTopic       =   "frmBottleneck"
   LockControls    =   -1  'True
```

```
MaxButton       =   0   'False
MinButton       =   0   'False
ScaleHeight     =   4470
ScaleWidth      =   9945
StartUpPosition =   2  'CenterScreen
Begin VB.CommandButton cmdManual
   Caption         =   "Open Manual"
   Height          =   345
   Left            =   495
   TabIndex        =   13
   Top             =   4125
   Width           =   1335
End
Begin RichTextLib.RichTextBox rtbHelp
   Height          =   2460
   Left            =   30
   TabIndex        =   12
   Top             =   1635
   Width           =   5700
   _ExtentX        =   10054
   _ExtentY        =   4339
   _Version        =   393217
   ScrollBars      =   2
   TextRTF         =   $"frmBottleneck.frx":038A
End
Begin VB.CommandButton cmdSelLink
   Caption         =   "Select"
   Height          =   255
   Left            =   4830
   TabIndex        =   5
   Top             =   1185
   Width           =   750
End
Begin VB.ComboBox cbxCorridor
   Height          =   315
   Left            =   2325
   Style           =   2  'Dropdown List
   TabIndex        =   4
   Top             =   1155
   Width           =   2400
End
Begin VB.CommandButton cmdSel2
   Caption         =   "Select"
   Height          =   255
   Left            =   4830
   TabIndex        =   3
   Top             =   720
```

```
   Width           =   750
End
Begin VB.ComboBox cbxHab2
   Height          =   315
   Left            =   1875
   Style           =   2  'Dropdown List
   TabIndex        =   2
   Top             =   690
   Width           =   2850
End
Begin VB.CommandButton cmdSel1
   Caption         =   "Select"
   Height          =   255
   Left            =   4830
   TabIndex        =   1
   Top             =   255
   Width           =   750
End
Begin VB.ComboBox cbxHab1
   Height          =   315
   Left            =   1875
   Style           =   2  'Dropdown List
   TabIndex        =   0
   Top             =   225
   Width           =   2850
End
Begin VB.CommandButton cmdHelp
   Caption         =   "Show Help >>"
   Height          =   345
   Left            =   3945
   TabIndex        =   8
   Top             =   4125
   Width           =   1275
End
Begin VB.CommandButton cmdOK
   Caption         =   "OK"
   Height          =   345
   Left            =   2835
   TabIndex        =   7
   Top             =   4125
   Width           =   1095
End
Begin VB.CommandButton cmdCancel
   Caption         =   "Cancel"
   Height          =   345
   Left            =   1845
   TabIndex        =   6
```

```
   Top             =   4125
   Width           =   975
End
Begin VB.Image imgUnCheckSpCorr
   Height          =   375
   Left            =   90
   Picture         =   "frmBottleneck.frx":0415
   Top             =   1110
   Width           =   375
End
Begin VB.Image imgUnCheckWB2
   Height          =   375
   Left            =   90
   Picture         =   "frmBottleneck.frx":0489
   Top             =   645
   Width           =   375
End
Begin VB.Image imgUnCheckWB1
   Height          =   375
   Left            =   90
   Picture         =   "frmBottleneck.frx":04FD
   Top             =   180
   Width           =   375
End
Begin VB.Image imgHelp
   Height          =   4380
   Left            =   5760
   Picture         =   "frmBottleneck.frx":0571
   Top             =   60
   Width           =   4155
End
Begin VB.Image imgCheckSpCorr
   Height          =   375
   Left            =   90
   Picture         =   "frmBottleneck.frx":3BAB5
   Top             =   1110
   Width           =   375
End
Begin VB.Image imgCheckWB2
   Height          =   375
   Left            =   90
   Picture         =   "frmBottleneck.frx":3BBCC
   Top             =   645
   Width           =   375
End
Begin VB.Image imgCheckWB1
   Height          =   375
```

```
      Left            =   90
      Picture         =   "frmBottleneck.frx":3BCE3
      Top             =   180
      Width           =   375
   End
   Begin VB.Shape boxTop
      Height          =   1560
      Left            =   45
      Top             =   60
      Width           =   5670
   End
   Begin VB.Label Label1
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Wildland Block #1:"
      Height          =   195
      Left            =   480
      TabIndex        =   11
      Top             =   255
      Width           =   1365
   End
   Begin VB.Label lbl_cbxCorridor
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Species Corridor Polygon:"
      Height          =   195
      Left            =   480
      TabIndex        =   10
      Top             =   1170
      Width           =   1830
   End
   Begin VB.Label lbl_cbxHab2
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Wildland Block #2:"
      Height          =   195
      Left            =   480
      TabIndex        =   9
      Top             =   720
      Width           =   1365
   End
End
Attribute VB_Name = "frmBottleneck"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

```
Option Explicit
' PUT IN GENERAL DECLARATIONS SECTION
Private Anchors As AnchorObjectList  ' Main anchor control object
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private m_MxDoc As esriArcMapUI.IMxDocument
Private m_pApp As IApplication
Private m_Frame As IModelessFrame
'Private m_WindowPos As IWindowPosition
Private m_ExtensionConfig As IExtensionConfig
Private m_strNameArray() As String
Private m_intNameCount As Integer
Private m_colPolygons As Collection
Private m_booHelpToggle As Boolean
Private m_IntHelpCategory As Integer
Private m_colFieldNames As Collection
Public m_lngWB1Count As Long
Public m_lngWB2Count As Long
Public m_lngCorrCount As Long
Public m_booIsDemo As Boolean

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmBottleneck.frm"

Public Property Let IsDemo(booIsDemo As Boolean)
  On Error GoTo ErrorHandler

30:   m_booIsDemo = booIsDemo

  Exit Property
ErrorHandler:
  HandleError True, "IsDemo " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Set ArcApplication(ByVal theApplication As IApplication)
  On Error GoTo ErrorHandler


41:   Set m_pApp = theApplication


  Exit Property
ErrorHandler:
  HandleError True, "ArcApplication " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Property

Public Function Frame() As IModelessFrame
  On Error GoTo ErrorHandler

52:   If m_Frame Is Nothing Then
53:     Set m_Frame = New ModelessFrame
54:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
57:   End If

59:   Set Frame = m_Frame

  Exit Function
ErrorHandler:
  HandleError True, "Frame " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Public Property Set Doc(pDoc As esriArcMapUI.IMxDocument)
  On Error GoTo ErrorHandler

69:   Set m_MxDoc = pDoc

  Exit Property
ErrorHandler:
  HandleError True, "Doc " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Private Sub cbxCorridor_Click()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
80:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
83:   lngIndex = cbxCorridor.ListIndex

85:   Set ext.PolyCorridor = Nothing

87:   If lngIndex >= 2 Then
    Dim strLayerName As String
89:     strLayerName = cbxCorridor.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
92:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
```

```
    Dim pFeatureCursor As IFeatureCursor
95:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
98:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
101:     Set pGeometry = pFeature.ShapeCopy

103:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
106:       Set pPolygon = pGeometry

108:       Set ext.PolyCorridor = pPolygon
109:       m_lngCorrCount = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
110:     End If
111:   End If
112:   cbxCorridor.BackColor = vbWhite
113:   Call UpdateCheckmarks
114:   Call UpdateSelectButtons
115:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxCorridor_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxCorridor_GotFocus()
  On Error GoTo ErrorHandler


126:   m_IntHelpCategory = 1
127:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxCorridor_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab1_Click()
  On Error GoTo ErrorHandler
136:   m_IntHelpCategory = 1
137:   Call UpdateHelpScreen
```

```
  Dim ext As Linkages.Extension
140:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
143:   lngIndex = cbxHab1.ListIndex

145:   Set ext.PolyWildland1 = Nothing

147:   If lngIndex >= 2 Then
    Dim strLayerName As String
149:     strLayerName = cbxHab1.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
152:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pFeatureCursor As IFeatureCursor
155:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
158:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
161:     Set pGeometry = pFeature.ShapeCopy

163:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
166:       Set pPolygon = pGeometry

168:       Set ext.PolyWildland1 = pPolygon
169:       m_lngWB1Count = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
170:     End If

172:   End If
173:   cbxHab1.BackColor = vbWhite
174:   Call UpdateCheckmarks
175:   Call UpdateSelectButtons
176:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab1_GotFocus()
  On Error GoTo ErrorHandler
```

```
186:   m_IntHelpCategory = 1
187:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab1_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab2_Click()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
198:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
201:   lngIndex = cbxHab2.ListIndex

203:   Set ext.PolyWildland2 = Nothing

205:   If lngIndex >= 2 Then
    Dim strLayerName As String
207:     strLayerName = cbxHab2.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
210:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pFeatureCursor As IFeatureCursor
213:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
216:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
219:     Set pGeometry = pFeature.ShapeCopy

221:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
224:       Set pPolygon = pGeometry

226:       Set ext.PolyWildland2 = pPolygon
227:       m_lngWB2Count = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
228:     End If
229:   End If
230:   cbxHab2.BackColor = vbWhite
```

```
231:   Call UpdateCheckmarks
232:   Call UpdateSelectButtons
233:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab2_GotFocus()
  On Error GoTo ErrorHandler


244:   m_IntHelpCategory = 1
245:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab2_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler

  '  ORIGINAL AVENUE CODE
  '  ' Jennessent.CompareParametersCancel
  '
  '  self.GetDialog.SetModalResult(nil)
  '  self.GetDialog.Close
260:   Unload Me

  '

  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdHelp_Click()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
273:   Set ext = m_ExtensionConfig
  Dim booHelpToggle As Boolean
```

```
275:   booHelpToggle = ext.HelpToggle1
  Dim pWindowPosition As IWindowPosition
277:   Set pWindowPosition = m_Frame

279:   If booHelpToggle = False Then
280:     booHelpToggle = True
281:     ext.HelpToggle1 = True
282:     cmdHelp.Caption = "<< Hide Help"
283:     pWindowPosition.Width = 669
284:     pWindowPosition.Height = 323
285:     cmdManual.Top = 4125
286:     cmdCancel.Top = 4125
287:     cmdOK.Top = 4125
288:     cmdHelp.Top = 4125
289:   Else
290:     booHelpToggle = False
291:     ext.HelpToggle1 = False
292:     cmdHelp.Caption = "Show Help >>"
293:     pWindowPosition.Width = 390
294:     pWindowPosition.Height = 160
295:     cmdManual.Top = 1665
296:     cmdCancel.Top = 1665
297:     cmdOK.Top = 1665
298:     cmdHelp.Top = 1665
299:   End If

301:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cmdHelp_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub SetBackColorsWhite()
  On Error GoTo ErrorHandler

311:   cbxHab1.BackColor = vbWhite
312:   cbxHab2.BackColor = vbWhite
313:   cbxCorridor.BackColor = vbWhite

  Exit Sub
ErrorHandler:
  HandleError False, "SetBackColorsWhite " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub cmdManual_Click()
  On Error GoTo ErrorHandler


  Dim strPath As String
325:   strPath = App.Path & "\help"

327:   Call Linkages.MyGeneralOperations.OpenDoc("Bottleneck_Subdocument.pdf", strPath)

  Exit Sub
ErrorHandler:
  HandleError True, "cmdManual_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler

337:   Call SetBackColorsWhite

  Dim lngIndex As Long
  Dim strLayerName As String
  Dim strLayerName2 As String
  Dim pFeatureLayer As IFeatureLayer
  Dim pFeatureLayer2 As IFeatureLayer
  Dim intHab1Index As Integer
  Dim intHab2Index As Integer
  Dim intCorrIndex As Integer
  Dim strHab1Name As String
  Dim strHab2Name As String
  Dim strCorrName As String

351:   intHab1Index = cbxHab1.ListIndex
352:   intHab2Index = cbxHab2.ListIndex
353:   intCorrIndex = cbxCorridor.ListIndex

  Dim pRelOp As IRelationalOperator
  Dim ext As Linkages.Extension
357:   Set ext = m_ExtensionConfig

  Dim pHabPoly1 As IPolygon
360:       Set pHabPoly1 = ext.PolyWildland1
  Dim pHabPoly2 As IPolygon
362:       Set pHabPoly2 = ext.PolyWildland2
  Dim pCorrPoly As IPolygon
364:       Set pCorrPoly = ext.PolyCorridor
```

```
  Dim boo1And2 As Boolean           '  SHOULD BE FALSE
  Dim boo1AndCorr As Boolean        '  SHOULD BE TRUE
  Dim boo2AndCorr As Boolean        '  SHOULD BE TRUE

  ' FIRST PUT ALL 3 POLYGONS INTO SAME SPATIAL REFERENCE AS pCorrPoly
  Dim pSpRefHab1 As ISpatialReference
372:   Set pSpRefHab1 = pHabPoly1.SpatialReference
  Dim pSpRefHab2 As ISpatialReference
374:   Set pSpRefHab2 = pHabPoly2.SpatialReference
  Dim pSpRefCorr As ISpatialReference
376:   Set pSpRefCorr = pCorrPoly.SpatialReference

378:   If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRefCorr, pSpRefHab1) Then
379:     pHabPoly1.Project pSpRefCorr
380:     Set ext.PolyWildland1 = pHabPoly1
381:   End If

383:   If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRefCorr, pSpRefHab2) Then
384:     pHabPoly2.Project pSpRefCorr
385:     Set ext.PolyWildland2 = pHabPoly2
386:   End If

'  MsgBox "pHabPoly1 is nothing? " & CStr(pHabPoly1 Is Nothing) & vbCrLf & _
      "      Length? " & CStr(pHabPoly1.length) & vbCrLf & _
      "      SR? " & pHabPoly1.SpatialReference.Name & vbCrLf & _
      "pHabPoly2 is nothing? " & CStr(pHabPoly2 Is Nothing) & vbCrLf & _
      "      Length? " & CStr(pHabPoly2.length) & vbCrLf & _
      "      SR? " & pHabPoly2.SpatialReference.Name & vbCrLf & _
      "pCorrPoly is nothing? " & CStr(pCorrPoly Is Nothing) & vbCrLf & _
      "      Length? " & CStr(pCorrPoly.length) & vbCrLf & _
      "      SR? " & pCorrPoly.SpatialReference.Name

398:   Set pRelOp = pHabPoly1
399:   boo1And2 = Not pRelOp.Disjoint(pHabPoly2) ' (pRelOp.Touches(pHabPoly2)) Or (pRelOp.Overlaps(pHabPoly2))
400:   boo1AndCorr = Not pRelOp.Disjoint(pCorrPoly) ' (pRelOp.Touches(pCorrPoly)) Or (pRelOp.Overlaps(pCorrPoly))

402:   Set pRelOp = pHabPoly2
403:   boo2AndCorr = Not pRelOp.Disjoint(pCorrPoly) ' (pRelOp.Touches(pCorrPoly)) Or (pRelOp.Overlaps(pCorrPoly))

'  MsgBox boo1And2 & vbCrLf & boo1AndCorr & vbCrLf & boo2AndCorr

  ' ERROR CHECKING CODE ----------------------------------------------------------
  ' CHECK FOR THEMES WITH MULTIPLE POLYGONS
409:   If m_lngWB1Count = 0 Then
410:     cbxHab1.BackColor = vbYellow
411:     MsgBox "Unable to find a polygon for Wildland Block #1!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
```

```
      , vbOKOnly, "Error Found in Input Data:"
414:     cbxHab1.SetFocus
    Exit Sub
416:   ElseIf m_lngWB1Count > 1 Then
417:     cbxHab1.BackColor = vbYellow
418:     lngIndex = cbxHab1.ListIndex
419:     strLayerName = cbxHab1.List(lngIndex)
420:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
421:     MsgBox "Multiple polygons (" & CStr(m_lngWB1Count) & ") found in Wildland Block #1 Layer '" & _
        pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
       "Please use the ""Select"" button to select a single " & _
       "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
425:     cbxHab1.SetFocus
    Exit Sub
427:   ElseIf m_lngWB2Count = 0 Then
428:     cbxHab2.BackColor = vbYellow
429:     MsgBox "Unable to find a polygon for Wildland Block #2!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
      , vbOKOnly, "Error Found in Input Data:"
432:     cbxHab2.SetFocus
    Exit Sub
434:   ElseIf m_lngWB2Count > 1 Then
435:     cbxHab2.BackColor = vbYellow
436:     lngIndex = cbxHab2.ListIndex
437:     strLayerName = cbxHab2.List(lngIndex)
438:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
439:     MsgBox "Multiple polygons (" & CStr(m_lngWB2Count) & ") found in Wildland Block #2 Layer '" & _
        pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
        "Please use the ""Select"" button to select a single " & _
        "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
443:     cbxHab2.SetFocus
    Exit Sub
445:   ElseIf m_lngCorrCount = 0 Then
446:     cbxCorridor.BackColor = vbYellow
447:     MsgBox "Unable to find a polygon for the Species Corridor Polygon!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
      , vbOKOnly, "Error Found in Input Data:"
450:     cbxCorridor.SetFocus
    Exit Sub
452:   ElseIf m_lngCorrCount > 1 Then
453:     cbxCorridor.BackColor = vbYellow
454:     lngIndex = cbxCorridor.ListIndex
455:     strLayerName = cbxCorridor.List(lngIndex)
456:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
457:     MsgBox "Multiple polygons (" & CStr(m_lngCorrCount) & ") found in Species Corridor Polygon Layer '" & _
        pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
        "Please use the ""Select"" button to select a single " & _
```

```
        "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
461:     cbxCorridor.SetFocus
    Exit Sub

  ' CHECK FOR SAME LAYER SELECTED
465:   ElseIf (intHab1Index > 1) And (intHab2Index > 1) And (intHab1Index = intHab2Index) Then
466:     cbxHab1.BackColor = vbYellow
467:     cbxHab2.BackColor = vbYellow
468:     lngIndex = cbxHab1.ListIndex
469:     strLayerName = cbxHab1.List(lngIndex)
470:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
471:     MsgBox "Both selected Wildland Habitat Block Layers point to the same layer! ('" & _
        pFeatureLayer.Name & "')!" & vbCrLf & vbCrLf & _
        "Please select different layers for each wildland habitat block...", _
        vbOKOnly, "Error Found in Input Data:"
475:     cbxHab1.SetFocus
    Exit Sub
477:   ElseIf (intHab1Index > 1) And (intCorrIndex > 1) And (intHab1Index = intCorrIndex) Then
478:     cbxHab1.BackColor = vbYellow
479:     cbxCorridor.BackColor = vbYellow
480:     lngIndex = cbxHab1.ListIndex
481:     strLayerName = cbxHab1.List(lngIndex)
482:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
483:     MsgBox "Both the Wildland Habitat Block #1 layer and the Species Corridor layer " & _
        "point to the same layer! ('" & _
        pFeatureLayer.Name & "')!" & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
488:     cbxHab1.SetFocus
    Exit Sub
490:   ElseIf (intHab2Index > 1) And (intCorrIndex > 1) And (intHab2Index = intCorrIndex) Then
491:     cbxHab2.BackColor = vbYellow
492:     cbxCorridor.BackColor = vbYellow
493:     lngIndex = cbxHab2.ListIndex
494:     strLayerName = cbxHab2.List(lngIndex)
495:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
496:     MsgBox "Both the Wildland Habitat Block #2 layer and the Species Corridor layer " & _
        "point to the same layer! ('" & _
        pFeatureLayer.Name & "')!" & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
501:     cbxHab2.SetFocus
    Exit Sub

  ' CHECK FOR INTERSECTING POLYGONS
505:   ElseIf boo1And2 Then
506:     cbxHab1.BackColor = vbYellow
```

```
507:     cbxHab2.BackColor = vbYellow
508:     lngIndex = cbxHab1.ListIndex
509:     If lngIndex = 1 Then
510:       strHab1Name = "Selected Graphic"
511:     Else
512:       strLayerName = cbxHab1.List(lngIndex)
513:       Set pFeatureLayer = m_colPolygons.Item(strLayerName)
514:       strHab1Name = pFeatureLayer.Name
515:     End If
516:     lngIndex = cbxHab2.ListIndex
517:     If lngIndex = 1 Then
518:       strHab1Name = "Selected Graphic"
519:     Else
520:       strLayerName2 = cbxHab2.List(lngIndex)
521:       Set pFeatureLayer2 = m_colPolygons.Item(strLayerName2)
522:       strHab2Name = pFeatureLayer2.Name
523:     End If

525:     MsgBox "Your Wildland Habitat Block #1 layer ('" & strHab1Name & "') intersects " & _
        "your Wildland Habitat Block #2 layer ('" & strHab2Name & "')!  If this is true, " & _
        "there is no need for a Species Corridor..." & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
530:     cbxHab1.SetFocus
    Exit Sub

533:   ElseIf Not boo1AndCorr Then
534:     cbxHab1.BackColor = vbYellow
535:     cbxCorridor.BackColor = vbYellow
536:     lngIndex = cbxHab1.ListIndex
537:     If lngIndex = 1 Then
538:       strHab1Name = "Selected Graphic"
539:     Else
540:       strLayerName = cbxHab1.List(lngIndex)
541:       Set pFeatureLayer = m_colPolygons.Item(strLayerName)
542:       strHab1Name = pFeatureLayer.Name
543:     End If

545:     lngIndex = cbxCorridor.ListIndex
546:     If lngIndex = 1 Then
547:       strCorrName = "Selected Graphic"
548:     Else
549:       strLayerName2 = cbxCorridor.List(lngIndex)
550:       Set pFeatureLayer2 = m_colPolygons.Item(strLayerName2)
551:       strCorrName = pFeatureLayer2.Name
552:     End If
```

```
554:     MsgBox "Your Wildland Habitat Block #1 layer ('" & strHab1Name & "') does not intersect " & _
        "your Species Corridor layer ('" & strCorrName & "')!  If this is true, " & _
        "then the species corridor polygon cannot connect your two wildland habitat blocks..." & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
559:     cbxHab1.SetFocus
    Exit Sub

562:   ElseIf Not boo2AndCorr Then
563:     cbxHab2.BackColor = vbYellow
564:     cbxCorridor.BackColor = vbYellow
565:     lngIndex = cbxHab2.ListIndex
566:     If lngIndex = 1 Then
567:       strHab2Name = "Selected Graphic"
568:     Else
569:       strLayerName = cbxHab2.List(lngIndex)
570:       Set pFeatureLayer = m_colPolygons.Item(strLayerName)
571:       strHab2Name = pFeatureLayer.Name
572:     End If

574:     lngIndex = cbxCorridor.ListIndex
575:     If lngIndex = 1 Then
576:       strCorrName = "Selected Graphic"
577:     Else
578:       strLayerName2 = cbxCorridor.List(lngIndex)
579:       Set pFeatureLayer2 = m_colPolygons.Item(strLayerName2)
580:       strCorrName = pFeatureLayer2.Name
581:     End If
582:     MsgBox "Your Wildland Habitat Block #2 layer ('" & strHab2Name & "') does not intersect " & _
        "your Species Corridor layer ('" & strCorrName & "')!  If this is true, " & _
        "then the species corridor polygon cannot connect your two wildland habitat blocks..." & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
587:     cbxHab2.SetFocus
    Exit Sub

590:   End If

592:   If m_booIsDemo Then
593:     Me.Frame.Visible = False
594:     Call Linkages.CorridorAnalysisFunctions.BottleneckAnalysisDemo(m_pApp)
595:     Me.Frame.Visible = True
596:     Unload Me
597:   Else
598:     Call CheckSavedValues
599:   End If
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub CheckSavedValues()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
610:   Set ext = m_ExtensionConfig

  Dim pArea1 As IArea
  Dim pPoly1 As IPolygon
  Dim pArea2 As IArea
  Dim pPoly2 As IPolygon
  Dim pArea3 As IArea
  Dim pPoly3 As IPolygon

619:   Set pPoly1 = ext.PolyWildland1
620:   Set pArea1 = pPoly1

622:   Set pPoly2 = ext.PolyWildland2
623:   Set pArea2 = pPoly2

625:   Set pPoly3 = ext.PolyCorridor
626:   Set pArea3 = pPoly3

'  Dim strReport As String
'  strReport = "Wildland Block #1 Area = " & CStr(pArea1.Area) & vbCrLf & _
'              "Wildland Block #2 Area = " & CStr(pArea2.Area) & vbCrLf & _
'              "Corridor Area = " & CStr(pArea3.Area)
'  MsgBox strReport


  ' GET START AND END NODES - FROM ORIGINAL TEST CODE
  Dim pStartPolygon As IPolygon
  Dim pEndPolygon As IPolygon
638:   Set pStartPolygon = ext.PolyWildland1
639:   Set pEndPolygon = ext.PolyWildland2

  Dim pClone As IClone

  ' GET CORRIDOR - FROM ORIGINAL TEST CODE
  Dim pCorPolygon As IPolygon
645:   Set pCorPolygon = ext.PolyCorridor
```

```
  ' ---------------------------------------------------------------------
'  MsgBox "<-- Code in development -->" & vbCrLf & _
        "Please check http://www.corridordesign.org for updates...", , "Function Not Implemented Yet:"

  ' PROGRESS METER STUFF ------------------------------------------------------------------
  Dim frmProgress As New frmJenProgressPercent
  Dim theTimeBegan As Date
  Dim theDetailedDescription As String

'  frmProgress.SetExpanded = ext.ProgressDialogSetExpanded
657:   frmProgress.SetExpanded = True
658:   frmProgress.SetAutoClose = ext.ProgressDialogAutoClose

660:   theTimeBegan = Now
661:   frmProgress.ShouldContinue = True
662:   frmProgress.ProgBeginTime = Now
663:   frmProgress.ProgRecCount = 0
664:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
665:   frmProgress.lblBeginTime.Caption = "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd")

667:   theDetailedDescription = "Analyzing Bottlenecks..." & vbCrLf & _
      "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd") & vbCrLf & _
      "-----------------------------------------" & vbCrLf
670:   frmProgress.txtDetails.Text = theDetailedDescription

672:   frmProgress.Frame.Caption = "Current Status:"
673:   frmProgress.Frame.Visible = True
674:   frmProgress.cmdDetails.Visible = False
675:   frmProgress.cmdStop.Visible = False

677:   Me.Frame.Visible = False

'   theProgressTimeCheck = CDate(50000)
  Dim theDescription As String
681:   theDescription = "Analyzing Bottlenecks..."
  ' PROGRESS METER STUFF ------------------------------------------------------------------

  Dim pParamDetails As esriSystem.IVariantArray
685:   Set pParamDetails = New esriSystem.VarArray

  Dim strLayerName As String
  Dim pFeatureLayer As IFeatureLayer
  Dim lngIndex As Long

  ' WILDLAND BLOCK 1
692:   If cbxHab1.ListIndex < 2 Then
693:     pParamDetails.Add "Selected Specific Polygon from Map..."
```

```
694:   Else
695:     strLayerName = cbxHab1.List(cbxHab1.ListIndex)
696:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
697:     pParamDetails.Add pFeatureLayer.Name
698:   End If

  ' WILDLAND BLOCK 2
701:   If cbxHab2.ListIndex < 2 Then
702:     pParamDetails.Add "Selected Specific Polygon from Map..."
703:   Else
704:     strLayerName = cbxHab2.List(cbxHab2.ListIndex)
705:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
706:     pParamDetails.Add pFeatureLayer.Name
707:   End If

  ' CORRIDOR POLYGON
710:   If cbxCorridor.ListIndex < 2 Then
711:     pParamDetails.Add "Selected Specific Polygon from Map..."
712:   Else
713:     strLayerName = cbxCorridor.List(cbxCorridor.ListIndex)
714:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
715:     pParamDetails.Add pFeatureLayer.Name
716:   End If

718:   ext.ProgressDialogAutoClose = (frmProgress.chkClose.Value = 1)
719:   ext.ProgressDialogSetExpanded = (frmProgress.SetExpanded)

'  Exit Sub

  ' CALL ANALYSIS MODULE
724:   Linkages.CorridorAnalysisFunctions.BottleneckAnalysis pParamDetails, m_MxDoc, m_pApp, _
      pCorPolygon, pStartPolygon, pEndPolygon, frmProgress

'  If frmProgress.chkClose.Value = 1 Then
'    Unload frmProgress
'    Set frmProgress = Nothing
'  End If

732:   Me.Frame.Visible = True

734:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError False, "CheckSavedValues " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub cmdSel1_Click()
  On Error GoTo ErrorHandler

  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
747:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
750:   Set theFormObject = ext.aSelForm

752:   If (theFormObject Is Nothing) Then
753:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
    Dim pMxDoc As IMxDocument
757:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType

773:     intKey = -1

775:     Set theMap = pMxDoc.FocusMap

    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
779:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
782:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
    Dim pEnumElement As IEnumElement

785:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)
```

```
    Dim booHasPolygon As Boolean
789:     booHasPolygon = True

    ReDim strNameArray(theMap.LayerCount)

793:     If (booHasPolygon) Then
794:       intKey = intKey + 1
795:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
796:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
797:       strNameArray(intKey) = strPolyName
798:     End If

    Dim pFeatureLayerForValid As IFeatureLayer

802:     If (theMap.LayerCount > 0) Then
803:       Set pEnumLayer = theMap.Layers(, True)
804:       pEnumLayer.Reset

806:       Set pLayer = pEnumLayer.Next
807:       Do Until pLayer Is Nothing
808:         If TypeOf pLayer Is IFeatureLayer Then
809:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
811:           If pFeatureLayerForValid.Valid Then
           ' CHECK IF POLYGON LAYER
813:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
814:             pGeometryType = pFeatureClass.ShapeType
815:             If (pGeometryType = esriGeometryPolygon) Then
816:               intKey = intKey + 1
817:               Set pFeatureLayer = pLayer
818:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
819:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
820:               strNameArray(intKey) = strPolyName
821:             End If
822:           End If
823:         End If
824:         Set pLayer = pEnumLayer.Next
825:       Loop
826:     End If

'    Dim theReport As String
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport
```

```
834:     Set frmSelect.ArcApplication = m_pApp
835:     Set frmSelect.Doc = m_MxDoc
836:     frmSelect.NameList = strNameArray
837:     frmSelect.NameCount = intKey
838:     Set frmSelect.NameCollection = colPolygonLayers
839:     frmSelect.PolygonPurpose = "Bottleneck_Wildland1"
840:     frmSelect.SearchMessage = "Wildland Block #1"

842:     frmSelect.EnableTool = False
843:     frmSelect.SetSelToolEnabled
844:     Load frmSelect

846:     frmSelect.Frame.Caption = "Select Wildland Block #1..."
847:     frmSelect.Frame.Visible = True
848:   Else
849:     Set frmSelect = theFormObject
850:   End If

852:   cbxHab1.BackColor = vbWhite

'  frmSelect.Show vbModeless
855:   frmSelect.Frame.Visible = True
856:   Me.Frame.Visible = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSel1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdSel2_Click()
  On Error GoTo ErrorHandler
865:   m_IntHelpCategory = 1
866:   Call UpdateHelpScreen

  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
871:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
874:   Set theFormObject = ext.aSelForm

876:   If (theFormObject Is Nothing) Then
877:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
```

```
    Dim pMxDoc As IMxDocument
881:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType

897:     intKey = -1

899:     Set theMap = pMxDoc.FocusMap

    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
903:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
906:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
    Dim pEnumElement As IEnumElement

909:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)

    Dim booHasPolygon As Boolean
'    booHasPolygon = False
914:     booHasPolygon = True
'
'    If (Not pEnumElement Is Nothing) Then
'      pEnumElement.Reset
'
'      Dim pElement As IElement
'      Set pElement = pEnumElement.Next
'
'      Dim pGeometry As IGeometry
'
'      Do Until pElement Is Nothing
'        Set pGeometry = pElement.Geometry
'        If TypeOf pGeometry Is IPolygon Then
```

```
'          booHasPolygon = True
'          Exit Do
'        End If
'        Set pElement = pEnumElement.Next
'      Loop
'    End If

'    MsgBox booHasPolygon

    ReDim strNameArray(theMap.LayerCount)

938:     If (booHasPolygon) Then
939:       intKey = intKey + 1
940:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
941:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
942:       strNameArray(intKey) = strPolyName
943:     End If

    Dim pFeatureLayerForValid As IFeatureLayer

947:     If (theMap.LayerCount > 0) Then
948:       Set pEnumLayer = theMap.Layers(, True)
949:       pEnumLayer.Reset

951:       Set pLayer = pEnumLayer.Next
952:       Do Until pLayer Is Nothing
953:         If TypeOf pLayer Is IFeatureLayer Then
954:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
956:           If pFeatureLayerForValid.Valid Then
            ' CHECK IF POLYGON LAYER
958:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
959:             pGeometryType = pFeatureClass.ShapeType
960:             If (pGeometryType = esriGeometryPolygon) Then
961:               intKey = intKey + 1
962:               Set pFeatureLayer = pLayer
963:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
964:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
965:               strNameArray(intKey) = strPolyName
966:             End If
967:           End If
968:         End If
969:         Set pLayer = pEnumLayer.Next
970:       Loop
971:     End If

'    Dim theReport As String
```

```
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport

979:     Set frmSelect.ArcApplication = m_pApp
980:     Set frmSelect.Doc = m_MxDoc
981:     frmSelect.NameList = strNameArray
982:     frmSelect.NameCount = intKey
983:     Set frmSelect.NameCollection = colPolygonLayers
984:     frmSelect.PolygonPurpose = "Bottleneck_Wildland2"
985:     frmSelect.SearchMessage = "Wildland Block #2"

987:     frmSelect.EnableTool = False
988:     frmSelect.SetSelToolEnabled
989:     Load frmSelect

991:     frmSelect.Frame.Caption = "Select Wildland Block #2..."
992:     frmSelect.Frame.Visible = True
993:   Else
994:     Set frmSelect = theFormObject
995:   End If

997:   cbxHab2.BackColor = vbWhite

'  frmSelect.Show vbModeless
1000:   frmSelect.Frame.Visible = True
1001:   Me.Frame.Visible = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSel2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdSelLink_Click()
  On Error GoTo ErrorHandler

1011:   m_IntHelpCategory = 1
1012:   Call UpdateHelpScreen
  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
1016:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
1019:   Set theFormObject = ext.aSelForm
```

```
1021:   If (theFormObject Is Nothing) Then
1022:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
    Dim pMxDoc As IMxDocument
1026:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType

1042:     intKey = -1

1044:     Set theMap = pMxDoc.FocusMap

    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
1048:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
1051:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
    Dim pEnumElement As IEnumElement

1054:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)

    Dim booHasPolygon As Boolean
1058:     booHasPolygon = True
'    booHasPolygon = False
'
'    If (Not pEnumElement Is Nothing) Then
'      pEnumElement.Reset
'
'      Dim pElement As IElement
'      Set pElement = pEnumElement.Next
'
```

```
'      Dim pGeometry As IGeometry
'
'      Do Until pElement Is Nothing
'        Set pGeometry = pElement.Geometry
'        If TypeOf pGeometry Is IPolygon Then
'          booHasPolygon = True
'          Exit Do
'        End If
'        Set pElement = pEnumElement.Next
'      Loop
'    End If

'    MsgBox booHasPolygon

    ReDim strNameArray(theMap.LayerCount)

1083:     If (booHasPolygon) Then
1084:       intKey = intKey + 1
1085:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
1086:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1087:       strNameArray(intKey) = strPolyName
1088:     End If

    Dim pFeatureLayerForValid As IFeatureLayer

1092:     If (theMap.LayerCount > 0) Then
1093:       Set pEnumLayer = theMap.Layers(, True)
1094:       pEnumLayer.Reset

1096:       Set pLayer = pEnumLayer.Next
1097:       Do Until pLayer Is Nothing
1098:         If TypeOf pLayer Is IFeatureLayer Then
1099:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
1101:           If pFeatureLayerForValid.Valid Then
            ' CHECK IF POLYGON LAYER
1103:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
1104:             pGeometryType = pFeatureClass.ShapeType
1105:             If (pGeometryType = esriGeometryPolygon) Then
1106:               intKey = intKey + 1
1107:               Set pFeatureLayer = pLayer
1108:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
1109:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1110:               strNameArray(intKey) = strPolyName
1111:             End If
1112:           End If
1113:         End If
```

```
1114:         Set pLayer = pEnumLayer.Next
1115:       Loop
1116:     End If

'    Dim theReport As String
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport

1124:     Set frmSelect.ArcApplication = m_pApp
1125:     Set frmSelect.Doc = m_MxDoc
1126:     frmSelect.NameList = strNameArray
1127:     frmSelect.NameCount = intKey
1128:     Set frmSelect.NameCollection = colPolygonLayers
1129:     frmSelect.PolygonPurpose = "Bottleneck_Corridor"
1130:     frmSelect.SearchMessage = "Species Corridor"

1132:     frmSelect.EnableTool = False
1133:     frmSelect.SetSelToolEnabled
1134:     Load frmSelect

1136:     frmSelect.Frame.Caption = "Select Species Corridor Polygon..."
1137:     frmSelect.Frame.Visible = True
1138:   Else
1139:     Set frmSelect = theFormObject
1140:   End If

1142:   cbxCorridor.BackColor = vbWhite
'  frmSelect.Show vbModeless
1144:   frmSelect.Frame.Visible = True
1145:   Me.Frame.Visible = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSelLink_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Sub UpdateCheckmarks()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
1156:   Set ext = m_ExtensionConfig

  Dim pHab1Polygon As IPolygon
  Dim pHab2Polygon As IPolygon
```

```
  Dim pCorPolygon As IPolygon

1162:   Set pHab1Polygon = ext.PolyWildland1
1163:   Set pHab2Polygon = ext.PolyWildland2
1164:   Set pCorPolygon = ext.PolyCorridor

1166:   imgCheckWB1.Visible = (Not ext.PolyWildland1 Is Nothing)
1167:   imgCheckWB2.Visible = (Not ext.PolyWildland2 Is Nothing)
1168:   imgCheckSpCorr.Visible = (Not ext.PolyCorridor Is Nothing)
1169:   imgUnCheckWB1.Visible = Not imgCheckWB1.Visible
1170:   imgUnCheckWB2.Visible = Not imgCheckWB2.Visible
1171:   imgUnCheckSpCorr.Visible = Not imgCheckSpCorr.Visible

  Exit Sub
ErrorHandler:
  HandleError True, "UpdateCheckmarks " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler
1180:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
  '  ORIGINAL AVENUE CODE
  '  ' Jennessent.CompareParametersOpen
  '
  '  AVUpperLeft = av.ReturnOrigin
  '  AVCenter = avUpperLeft + (av.ReturnExtent / (2@2))
  '  halfDialogWidthHeight = Self.ReturnExtent.ReturnSize / (2@2)
  '  MovePoint = AVCenter - halfDialogWidthHeight
  '  Self.MoveTo(MovePoint.GetX, MovePoint.GetY)
  '
  '  theDialog = self
  '  cmdOK = theDialog.FindByName("cmdOK")
  '  cmdCancel = theDialog.FindByName("cmdCancel")
  '
'  NOTES ON CONTROLS:
'  Remember to reset label autosize to true, if desired.
'   --> Contains Control Panels:  Make sure to cut and paste nested controls
'          if you want them to be properly nested in the Form Frame.
'   --> Contains Control Panels:  Make sure to cut and paste nested controls
'          if you want them to be properly nested in the Form Frame.
'   --> Contains Control Panels:  Make sure to cut and paste nested controls
'          if you want them to be properly nested in the Form Frame.
'   --> icon_CornerBars: Original Image = jennessent_bars.gif
1203:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
1204:   Me.Left = (Screen.Width / 3) - (Me.Width / 2)
1205:   Me.Top = (Screen.Height / 2) - (Me.Height / 2)
```

```
1207:   If m_Frame Is Nothing Then
1208:     Set m_Frame = New ModelessFrame
1209:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
1212:   End If

  ' CLEAR ANY SAVED EXTENSION POLYGONS
  Dim newUid As New uID
1216:   newUid.Value = "Linkages.Extension"
1217:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
1219:   Set ext = m_ExtensionConfig

1221:   Set ext.PolyWildland1 = Nothing
1222:   Set ext.PolyWildland2 = Nothing
1223:   Set ext.PolyCorridor = Nothing
1224:   Set ext.frmStep1 = Me

  ' CHECK HELP WINDOW STATUS
1227:   m_booHelpToggle = ext.HelpToggle1

    ' IDENTIFY POLYGON THEMES
  Dim pMxDoc As IMxDocument
1231:   Set pMxDoc = m_MxDoc

  Dim colPolygonLayers As New Collection
  Dim strNameArray() As String

  Dim theMap As IMap
  Dim pEnumLayer As IEnumLayer
  Dim pFeatureLayer As IFeatureLayer
  Dim pLayer As IUnknown
  Dim anIndex As Long
  Dim strPolyName As String
  Dim intKey As Integer

  Dim pFeatureClass As IFeatureClass
  Dim pGeometryType As esriGeometryType

1247:   intKey = 1

1249:   Set theMap = pMxDoc.FocusMap

  ReDim strNameArray(theMap.LayerCount + 2)
```

```
1253:   colPolygonLayers.Add "placeholder_FirstLine", "FirstLine"
1254:   strNameArray(0) = "FirstLine"

1256:   colPolygonLayers.Add "placeholder_SelFromView", "SelFromView"
1257:   strNameArray(1) = "SelFromView"

  Dim pFeatureLayerForValid As IFeatureLayer

1261:   If (theMap.LayerCount > 0) Then
1262:     Set pEnumLayer = theMap.Layers(, True)
1263:     pEnumLayer.Reset

1265:     Set pLayer = pEnumLayer.Next
1266:     Do Until pLayer Is Nothing
1267:       If TypeOf pLayer Is IFeatureLayer Then
1268:         Set pFeatureLayerForValid = pLayer
        ' CHECK IF FEATURE LAYER IS VALID
1270:         If pFeatureLayerForValid.Valid Then
          ' CHECK IF POLYGON LAYER
1272:           Set pFeatureClass = pFeatureLayerForValid.FeatureClass
1273:           pGeometryType = pFeatureClass.ShapeType
1274:           If (pGeometryType = esriGeometryPolygon) Then
1275:             intKey = intKey + 1
1276:             Set pFeatureLayer = pLayer
1277:             strPolyName = CStr(intKey - 1) & "] " & pFeatureLayer.Name
1278:             colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1279:             strNameArray(intKey) = strPolyName
1280:           End If
1281:         End If
1282:       End If
1283:       Set pLayer = pEnumLayer.Next
1284:     Loop
1285:   End If

1287:   m_strNameArray = strNameArray
1288:   Set m_colPolygons = colPolygonLayers

  ' FILL LISTBOXES WITH POLYGON SOURCE OPTIONS

  Dim strWB1Array() As String
  ReDim strWB1Array(intKey)
1294:   strWB1Array(0) = "Wildland Block #1 Source..."
1295:   strWB1Array(1) = "<-- Select by clicking on map -->"

  Dim strWB2Array() As String
  ReDim strWB2Array(intKey)
1299:   strWB2Array(0) = "Wildland Block #2 Source..."
```

```
1300:   strWB2Array(1) = "<-- Select by clicking on map -->"

  Dim strCorArray() As String
  ReDim strCorArray(intKey)
1304:   strCorArray(0) = "Species Corridor Source..."
1305:   strCorArray(1) = "<-- Select by clicking on map -->"


'  Dim theReport As String
'  theReport = "Upper bound of 'm_strNameArray' = " & CStr(UBound(m_strNameArray)) & vbCrLf & _
'      "Value = " & m_strNameArray(UBound(m_strNameArray))
'  For anIndex = LBound(m_strNameArray) To UBound(m_strNameArray)
'    theReport = theReport & "Index " & CStr(anIndex) & " --> " & m_strNameArray(anIndex) _
'        & vbCrLf
'  Next anIndex
'
'  MsgBox theReport

1318:   For anIndex = 2 To UBound(m_strNameArray)
1319:     If Not m_strNameArray(anIndex) = "" Then
1320:       strWB1Array(anIndex) = m_strNameArray(anIndex)
1321:       strWB2Array(anIndex) = m_strNameArray(anIndex)
1322:       strCorArray(anIndex) = m_strNameArray(anIndex)
1323:     End If
1324:   Next anIndex

1326:   cbxHab1.Clear
1327:   cbxHab2.Clear
1328:   cbxCorridor.Clear

1330:   For anIndex = 0 To UBound(strWB1Array)
1331:     cbxHab1.AddItem (strWB1Array(anIndex))
1332:     cbxHab2.AddItem (strWB2Array(anIndex))
1333:     cbxCorridor.AddItem (strCorArray(anIndex))
1334:   Next anIndex

1336:   cbxHab1.ListIndex = 0
1337:   cbxHab2.ListIndex = 0
1338:   cbxCorridor.ListIndex = 0

  ' SET CHECKMARKS AND UPDATE BUTTONS
1341:   Call UpdateCheckmarks
1342:   Call UpdateSelectButtons

  ' UPDATE HELP SCREEN POSITION
  Dim booHelpToggle As Boolean
1346:   booHelpToggle = ext.HelpToggle1
```

```
  Dim pWindowPosition As IWindowPosition
1348:   Set pWindowPosition = m_Frame

1350:   If booHelpToggle = False Then
1351:     cmdHelp.Caption = "Show Help >>"
1352:     pWindowPosition.Width = 390
1353:     pWindowPosition.Height = 160
1354:     cmdManual.Top = 1665
1355:     cmdCancel.Top = 1665
1356:     cmdOK.Top = 1665
1357:     cmdHelp.Top = 1665
1358:   Else
1359:     cmdHelp.Caption = "<< Hide Help"
1360:     pWindowPosition.Width = 669
1361:     pWindowPosition.Height = 323
1362:     cmdManual.Top = 4125
1363:     cmdCancel.Top = 4125
1364:     cmdOK.Top = 4125
1365:     cmdHelp.Top = 4125
1366:   End If

1368:   m_IntHelpCategory = 1
1369:   Call UpdateHelpScreen

  ' MAKE SURE BACK COLORS ARE WHITE
1372:   Call SetBackColorsWhite

1374:   Me.Refresh

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub UpdateHelpScreen()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
1385:   Set ext = m_ExtensionConfig
  Dim booHelpToggle As Boolean
1387:   booHelpToggle = ext.HelpToggle1

1389:   imgHelp.Visible = booHelpToggle
1390:   rtbHelp.Visible = booHelpToggle

  Dim strHelpText As String
1393:   strHelpText = Linkages.modHelpStrings.BottleneckHelp
```

```
1394:   rtbHelp.TextRTF = strHelpText

1396:   rtbHelp.SelStart = 0
1397:   rtbHelp.Locked = True

  Exit Sub
ErrorHandler:
  HandleError False, "UpdateHelpScreen " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub UpdateSelectButtons()
  On Error GoTo ErrorHandler


1408:   cmdSel1.Enabled = (cbxHab1.ListIndex = 1)
1409:   cmdSel2.Enabled = (cbxHab2.ListIndex = 1)
1410:   cmdSelLink.Enabled = (cbxCorridor.ListIndex = 1)


  Exit Sub
ErrorHandler:
  HandleError False, "UpdateSelectButtons " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler

  '  ORIGINAL AVENUE CODE
  '  ' Jennessent.CompareParametersClose
  '
  '  self.SetObjectTag(nil)
  '  self.FindByName("cmdOK").SetObjectTag(nil)
  '  self.FindByName("cmdCancel").SetObjectTag(nil)
  '
    ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
1429:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors")

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
1433:   pUID.Value = "Linkages.toolReturnCoords"

  Dim ext As Linkages.Extension
1436:   Set ext = m_ExtensionConfig
1437:   ext.EnableSelTool = False
1438:   ext.EnableDrawTool = False
```

```
1439:   Set ext.PolyCorridor = Nothing
1440:   Set ext.PolyWildland1 = Nothing
1441:   Set ext.PolyWildland2 = Nothing
1442:   Set ext.frmStep1 = Nothing

  ' UNLOAD SELECTION FORM IF IT IS OPEN
1445:   If (Not ext.aSelForm Is Nothing) Then
    Dim pSelForm As Linkages.frmSelScreen
1447:     Set pSelForm = ext.aSelForm
1448:     Unload pSelForm
1449:     Set ext.aSelForm = Nothing
1450:   End If

1452:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
1453:   If pCommand.Checked Then
1454:     Set m_pApp.CurrentTool = Nothing

    Dim pCommandItem As ICommandItem
1457:     Set pCommandItem = pCommand

1459:     pCommandItem.Refresh
1460:   End If

1462:   Set pCommand = Nothing
1463:   Set pCommandItem = Nothing
1464:   Set ext = Nothing
1465:   Set m_pApp = Nothing
1466:   Set m_MxDoc = Nothing
1467:   Set m_Frame = Nothing
1468:   Set m_ExtensionConfig = Nothing
1469:   Set m_colPolygons = Nothing
'  Set m_WindowPos = Nothing



  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub Label1_Click()
  On Error GoTo ErrorHandler

1482:   m_IntHelpCategory = 1
1483:   Call UpdateHelpScreen
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "Label1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbl_cbxCorridor_Click()
  On Error GoTo ErrorHandler


1495:   m_IntHelpCategory = 1
1496:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxCorridor_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbl_cbxHab2_Click()
  On Error GoTo ErrorHandler

1506:   m_IntHelpCategory = 1
1507:   Call UpdateHelpScreen


  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxHab2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Sub EnableOKButton()
  On Error GoTo ErrorHandler


  Dim booPolysOK As Boolean
1520:   booPolysOK = imgCheckWB1.Visible And imgCheckWB2.Visible And imgCheckSpCorr.Visible


1523:   cmdOK.Enabled = booPolysOK

'  MsgBox "optUseAll.Value = " & optUseAll.Value & vbCrLf & _
'         "(cbxPatchField.ListIndex > 1) = " & (cbxPatchField.ListIndex > 1) & vbCrLf & _
'         "(txtValue.Text <> "") = " & (txtValue.Text <> "") & vbCrLf & _
'         "((optUseAll.Value) Or ((cbxPatchField.ListIndex > 1) And (txtValue.Text <> ..))) = " & _
```

```
'         ((optUseAll.Value) Or ((cbxPatchField.ListIndex > 1) And (txtValue.Text <> ""))) & vbCrLf


  Exit Sub
ErrorHandler:
  HandleError True, "EnableOKButton " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 4: frmClip.frm

```
VERSION 5.00
Begin VB.Form frmClip
   AutoRedraw      =   -1  'True
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Clip Data to Corridor:"
   ClientHeight    =   5190
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   7920
   Icon            =   "frmClip.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5190
   ScaleWidth      =   7920
   ShowInTaskbar   =   0   'False
   StartUpPosition =   1  'CenterOwner
   Begin VB.CommandButton cmdHelp
      Caption         =   "Help"
      Height          =   345
      Left            =   6480
      TabIndex        =   11
      Top             =   3015
      Width           =   1095
   End
   Begin VB.CommandButton cmdGetWorkspace
      Height          =   360
      Left            =   7080
      Picture         =   "frmClip.frx":038A
      Style           =   1  'Graphical
      TabIndex        =   7
      ToolTipText     =   "Browse for Output Folder..."
      Top             =   4410
      Width           =   555
```

```
End
Begin VB.TextBox txtOutput
   Height          =   330
   Left            =   840
   Locked          =   -1  'True
   TabIndex        =   6
   Text            =   "Text1"
   Top             =   4425
   Width           =   5985
End
Begin VB.ListBox lbxClipLayers
   Height          =   2310
   Left            =   840
   Style           =   1  'Checkbox
   TabIndex        =   5
   Top             =   1590
   Width           =   5055
End
Begin VB.CommandButton cmdCancel
   Caption         =   "Cancel"
   Height          =   345
   Left            =   6480
   TabIndex        =   4
   Top             =   2595
   Width           =   1095
End
Begin VB.CommandButton cmdOK
   Caption         =   "OK"
   Height          =   345
   Left            =   6480
   TabIndex        =   3
   Top             =   3435
   Width           =   1095
End
Begin VB.ComboBox cbxCorridor
   Height          =   315
   Left            =   2655
   Style           =   2  'Dropdown List
   TabIndex        =   1
   Top             =   480
   Width           =   3465
End
Begin VB.CommandButton cmdSelLink
   Caption         =   "Select from Map"
   Height          =   315
   Left            =   6270
   TabIndex        =   0
```

```
   Top             =   495
   Width           =   1365
End
Begin VB.Image imgCheckPathname
   Height          =   375
   Left            =   300
   Picture         =   "frmClip.frx":0400
   Top             =   4410
   Width           =   375
End
Begin VB.Image imgUncheckPathname
   Height          =   375
   Left            =   300
   Picture         =   "frmClip.frx":0517
   Top             =   4410
   Width           =   375
End
Begin VB.Image imgCorrIcon
   Height          =   855
   Left            =   6555
   Picture         =   "frmClip.frx":058B
   Top             =   1455
   Width           =   945
End
Begin VB.Image Image3
   Appearance      =   0  'Flat
   BorderStyle     =   1  'Fixed Single
   Height          =   1110
   Left            =   105
   Top             =   4035
   Width           =   7770
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Identify Layers to Clip:"
   BeginProperty Font
      Name            =   "Tahoma"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   240
   Left            =   225
```

```
   TabIndex        =   13
   Top             =   1245
   Width           =   2220
End
Begin VB.Image Image2
   Appearance      =   0  'Flat
   BorderStyle     =   1  'Fixed Single
   Height          =   2895
   Left            =   105
   Top             =   1155
   Width           =   5985
End
Begin VB.Label lblStep1
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Identify Corridor Polygon:"
   BeginProperty Font
      Name            =   "Tahoma"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   240
   Left            =   225
   TabIndex        =   12
   Top             =   165
   Width           =   2520
End
Begin VB.Image Image1
   Appearance      =   0  'Flat
   BorderStyle     =   1  'Fixed Single
   Height          =   1110
   Left            =   105
   Top             =   60
   Width           =   7770
End
Begin VB.Image imgUncheckClipLayers
   Height          =   375
   Left            =   300
   Picture         =   "frmClip.frx":308F
   Top             =   1575
   Width           =   375
End
Begin VB.Image imgUnCheckSpCorr
```

```
   Height          =   375
   Left            =   300
   Picture         =   "frmClip.frx":3103
   Top             =   465
   Width           =   375
End
Begin VB.Label lblWorking
   AutoSize        =   -1  'True
   Caption         =   "Working..."
   BeginProperty Font
      Name            =   "Tahoma"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   285
   Left            =   2640
   TabIndex        =   10
   Top             =   2625
   Width           =   1245
End
Begin VB.Image imgIcon
   Height          =   855
   Left            =   4350
   Picture         =   "frmClip.frx":3177
   Top             =   2340
   Width           =   945
End
Begin VB.Label lblWorkspaceComment
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "(Clipped datasets will be named by their existing name appended with ""_clip"")"
   Height          =   195
   Left            =   1095
   TabIndex        =   9
   Top             =   4815
   Width           =   5445
End
Begin VB.Label lblWorkspace
   AutoSize        =   -1  'True
   Caption         =   "Specify Output Workspace:"
   BeginProperty Font
      Name            =   "Tahoma"
      Size            =   9.75
```

```
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   240
      Left            =   225
      TabIndex        =   8
      Top             =   4125
      Width           =   2655
   End
   Begin VB.Image imgCheckClipLayers
      Height          =   375
      Left            =   300
      Picture         =   "frmClip.frx":5C7B
      Top             =   1575
      Width           =   375
   End
   Begin VB.Image imgCheckSpCorr
      Height          =   375
      Left            =   300
      Picture         =   "frmClip.frx":5D92
      Top             =   465
      Width           =   375
   End
   Begin VB.Label lbl_cbxCorridor
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Species Corridor Polygon:"
      Height          =   195
      Left            =   810
      TabIndex        =   2
      Top             =   555
      Width           =   1830
   End
End
Attribute VB_Name = "frmClip"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Anchors As AnchorObjectList  ' Main anchor control object
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
```

```
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private m_MxDoc As esriArcMapUI.IMxDocument
Private m_pApp As IApplication
Private m_Frame As IModelessFrame
'Private m_WindowPos As IWindowPosition
Private m_ExtensionConfig As IExtensionConfig
Private m_strNameArray() As String
Private m_intNameCount As Integer
Private m_colPolygons As Collection
Private m_ColClipLayers As Collection
Public m_lngCorrCount As Long
Private m_IntHelpCategory As Integer

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmClip.frm"


Public Property Set ArcApplication(ByVal theApplication As IApplication)
  On Error GoTo ErrorHandler


28:   Set m_pApp = theApplication


  Exit Property
ErrorHandler:
  HandleError True, "ArcApplication " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Function Frame() As IModelessFrame
  On Error GoTo ErrorHandler

39:   If m_Frame Is Nothing Then
40:     Set m_Frame = New ModelessFrame
41:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
44:   End If

46:   Set Frame = m_Frame

  Exit Function
ErrorHandler:
  HandleError True, "Frame " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
```

```
Public Property Set Doc(pDoc As esriArcMapUI.IMxDocument)
  On Error GoTo ErrorHandler

56:   Set m_MxDoc = pDoc

  Exit Property
ErrorHandler:
  HandleError True, "Doc " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Private Sub cbxCorridor_Click()
  On Error GoTo ErrorHandler

66:   Call SetBackColorsWhite

  Dim ext As Linkages.Extension
69:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
72:   lngIndex = cbxCorridor.ListIndex

74:   Set ext.PolyCorridor = Nothing

76:   If lngIndex >= 2 Then
    Dim strLayerName As String
78:     strLayerName = cbxCorridor.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
81:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pFeatureCursor As IFeatureCursor
84:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
87:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
90:     Set pGeometry = pFeature.ShapeCopy

92:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
95:       Set pPolygon = pGeometry

97:       Set ext.PolyCorridor = pPolygon
98:       m_lngCorrCount = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
```

```
99:     End If
100:   End If

102:   Call UpdateCheckmarks

'  MsgBox cbxCorridor.ListIndex
105:   cmdSelLink.Enabled = (cbxCorridor.ListIndex = 1)

107:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxCorridor_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
118:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors")
119:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors_orig")

  Dim ext As Linkages.Extension
122:   Set ext = m_ExtensionConfig
123:   Set ext.frmClipForm = Nothing
124:   Me.Frame.Visible = False

126:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdGetWorkspace_Click()
  On Error GoTo ErrorHandler


  Dim strDirPath As String
  Dim strUserName As String

140:   strDirPath = txtOutput.Text
141:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then strDirPath = strDirPath & "\"
142:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
143:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_pApp))
```

```
144:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
145:   End If

  Dim boolWorkspaceExists As Boolean
148:   boolWorkspaceExists = Not Dir$(strDirPath) = ""

150:   If Not boolWorkspaceExists Then
151:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.TempPathLocation)
152:   End If

  Dim pGxDialog As IGxDialog
155:   Set pGxDialog = New GxDialog

  Dim pGxDialogFilter As IGxObjectFilter
'  Set pGxDialogFilter = New GxFilterWorkspaces
159:   Set pGxDialogFilter = New GxFilterBasicTypes
'  pGxDialogFilter.Name = "Folders"
'  Set pGxDialogFilter = New GxFilterContainers      ' INCLUDED GRIDS AND COVERAGES

  Dim pGxObject As IGxObject
  Dim pGxSelection As IEnumGxObject

166:   With pGxDialog
167:     .AllowMultiSelect = False
168:     .StartingLocation = strDirPath
169:     .Title = "Please select (don't open!) folder to contain your clipped datasets:"
170:     Set .ObjectFilter = pGxDialogFilter
171:   End With

  Dim theFinalString As String
'  If Not pGxDialog.DoModalOpen(0, pEnumGx) Then
'Exit Sub 'Exit if user press Cancel
'End If
'MsgBox pEnumGx.Next.FullName
178:   If (pGxDialog.DoModalOpen(Me.hWnd, pGxSelection) = True) Then
    'Set pGxObject = pGxDialog.FinalLocation
180:     Set pGxObject = pGxSelection.Next

    Dim pGxFile As IGxFile
183:     Set pGxFile = pGxObject

185:     theFinalString = pGxObject.FullName
186:     If (Right(theFinalString, 1) <> "\") And (Right(theFinalString, 1) <> "/") Then theFinalString = theFinalString & "\"

'    If aml_func_mod.ExistFileDir(theFinalString) Then
'      theFinalString = aml_func_mod.MakeUniqueFilename(theFinalString)
'
```

```
'      MsgBox "Unable to overwrite the file '" & pGxDialog.Name & "'.  The new file will be saved to " & _
'        theFinalString & "..."
'    End If

195:     txtOutput.Text = theFinalString

197:   End If

  Exit Sub
ErrorHandler:
  HandleError True, "cmdGetWorkspace_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdHelp_Click()
  On Error GoTo ErrorHandler


  Dim strPath As String
209:   strPath = App.Path & "\help"

211:   Call Linkages.MyGeneralOperations.OpenDoc("Clip_Tool_Subdocument.pdf", strPath)


  Exit Sub
ErrorHandler:
  HandleError True, "cmdHelp_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4

End Sub
Private Sub SetBackColorsWhite()
  On Error GoTo ErrorHandler

222:   cbxCorridor.BackColor = vbWhite

  Exit Sub
ErrorHandler:
  HandleError False, "SetBackColorsWhite " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler

231:   Call SetBackColorsWhite

  ' CHECK FOR SINGLE CORRIDOR POLYGON
  Dim lngIndex As Long
```

```
  Dim strLayerName As String
  Dim pFeatureLayer As IFeatureLayer

238:   If m_lngCorrCount = 0 Then
239:     cbxCorridor.BackColor = vbYellow
240:     MsgBox "Unable to find a polygon for the Species Corridor Polygon!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
      , vbOKOnly, "Error Found in Input Data:"
243:     cbxCorridor.SetFocus
    Exit Sub
245:   ElseIf m_lngCorrCount > 1 Then
246:     cbxCorridor.BackColor = vbYellow
247:     lngIndex = cbxCorridor.ListIndex
248:     strLayerName = cbxCorridor.List(lngIndex)
249:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
250:     MsgBox "Multiple polygons (" & CStr(m_lngCorrCount) & ") found in Species Corridor Polygon Layer '" & _
        pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
        "Please use the ""Select"" button to select a single " & _
        "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
254:     cbxCorridor.SetFocus
    Exit Sub
256:   End If

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
259:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors")
260:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors_orig")

  Dim ext As Linkages.Extension
263:   Set ext = m_ExtensionConfig
264:   Set ext.frmClipForm = Nothing
  Dim strWorkFolder As String
266:   strWorkFolder = txtOutput.Text
267:   If Right(strWorkFolder, 1) <> "\" And Right(strWorkFolder, 1) <> "/" Then
268:     strWorkFolder = strWorkFolder & "\"
269:   End If
270:   ext.ClipDirectoryPath = strWorkFolder

  ' MAKE DIALOG DISAPPEAR
  Dim pControl As Control
274:   For Each pControl In Me.Controls
275:    pControl.Visible = False
276:   Next pControl
277:   lblWorking.Visible = True
278:   imgIcon.Visible = True
279:   Me.Refresh
'  Me.Frame.Visible = False
'  m_pApp.RefreshWindow
```

```
'  Dim pEnvelope As IEnvelope
'  Me.Refresh
'  Set pEnvelope = m_MxDoc.ActiveView.Extent
'  m_MxDoc.ActiveView.Refresh

287:   DoEvents

  Dim pClone As IClone
  Dim pSendPolygon As IPolygon

  Dim anIndex As Integer
  Dim strName As String
  Dim pLayer As ILayer
  Dim pRasterLayer As IRasterLayer
  Dim pFeatureClass As IFeatureClass
  Dim pGeometryType As esriGeometryType
  Dim strShapeName As String

  Dim colWorkOrder As New Collection
  Dim strWorkOrder() As String
  ReDim strWorkOrder(lbxClipLayers.ListCount - 1)
  Dim intWorkIndex As Integer
304:   intWorkIndex = 0
  Dim strNewName As String

  ' GET POLYGON
  Dim pPolygon As IPolygon
309:   Set pPolygon = ext.PolyCorridor

  ' MAKE WORK ORDER COLLECTION
312:   For anIndex = 0 To lbxClipLayers.ListCount - 1
313:     If lbxClipLayers.Selected(anIndex) Then
314:       strName = lbxClipLayers.List(anIndex)
315:       Set pLayer = m_ColClipLayers.Item(strName)

317:       intWorkIndex = intWorkIndex + 1
318:       If TypeOf pLayer Is IFeatureLayer Then
319:         Set pFeatureLayer = pLayer
320:         Set pFeatureClass = pFeatureLayer.FeatureClass
321:         pGeometryType = pFeatureClass.ShapeType
322:         strShapeName = Linkages.aml_func_mod.ReturnShapeName(pGeometryType)
323:         strNewName = intWorkIndex & "] " & pLayer.Name & " (" & strShapeName & " Dataset)"
324:       Else
325:         strNewName = intWorkIndex & "] " & pLayer.Name & " (Raster Dataset)"
326:       End If

328:       colWorkOrder.Add m_ColClipLayers.Item(strName), strNewName
```

```
329:       strWorkOrder(intWorkIndex - 1) = strNewName
330:     End If
331:   Next anIndex

  Dim strReport As String
334:   strReport = "Clipped the following layers:" & vbCrLf & "-------------------------------------------------------" & vbCrLf

  Dim pClipFeatureLayer As IFeatureLayer
  Dim pClipRasterLayer As IRasterLayer
  Dim pClipLayer As ILayer
  Dim pDataset As IDataset
  Dim strClipName As String
  Dim lngFeatureCount As Long
  Dim pArea As IArea

  ' FOR RASTER CLIPS
  Dim pClipRaster As IRaster
  Dim pOrigRasterBandCollection As IRasterBandCollection
  Dim pOrigRasterBand As IRasterBand
'  Dim pOrigRasterDataset As IRasterDataset
  Dim pOrigFields As IFields
  Dim pOrigField As IField
  Dim pOrigTable As ITable

  Dim pNewRasterBandCollection As IRasterBandCollection
  Dim pNewRasterBand As IRasterBand
  Dim pNewRasterDataset As IRasterDataset
'  Dim pNewFields As IFields
  Dim pNewField As IField
  Dim pNewTable As ITable
  Dim booAddedFields As Boolean

  Dim pOrigCursor As ICursor
  Dim pOrigRow As IRow
  Dim pNewCursor As ICursor
  Dim pNewRow As IRow
  Dim pQueryFilter As IQueryFilter
366:   Set pQueryFilter = New QueryFilter
367:   pQueryFilter.AddField "Value"

  Dim lngValue As Long
  Dim lngOrigValField As Long
  Dim lngNewValField As Long
  Dim lngIndex2 As Long

  Dim pGridTableOp As IGridTableOp
375:   Set pGridTableOp = New GridTableOp
```

```
  Dim booOrigHasVAT As Boolean
  Dim booNewHasVAT As Boolean

  Dim intIndexArray() As Integer
  Dim intFieldIndex As Integer
  Dim intNewFieldIndex As Integer
382:   intNewFieldIndex = -1

384:   For anIndex = 0 To (intWorkIndex - 1)
385:     strName = strWorkOrder(anIndex)
386:     Set pClone = pPolygon
387:     Set pSendPolygon = pClone.Clone
388:     booAddedFields = False

 '   MsgBox "Index = " & CStr(anIndex) & vbCrLf & "Name = " & strName
391:     Set pLayer = colWorkOrder.Item(strName)
392:     If TypeOf pLayer Is IFeatureLayer Then  ' -------------------- CLIPPING FEATURE LAYER ----------------------
393:       Set pFeatureLayer = pLayer
394:       Set pClipFeatureLayer = Linkages.modClipFunctions.ClipFeatureLayer(pSendPolygon, pFeatureLayer, strWorkFolder, m_pApp)
395:       If pClipFeatureLayer Is Nothing Then
396:         strReport = strReport & strName & vbCrLf & "  --> No Intersecting Features Found:  No shapefile created... " & vbCrLf &
vbCrLf
397:       Else
398:         Set pClipLayer = pClipFeatureLayer
399:         Set pDataset = pClipFeatureLayer
400:         strClipName = pDataset.BrowseName
401:         pClipLayer.Name = strClipName
402:         m_MxDoc.AddLayer pClipLayer
403:         strReport = strReport & strName & vbCrLf & "  --> New Feature Count = " &
pClipFeatureLayer.FeatureClass.FeatureCount(Nothing) & _
           vbCrLf & "  --> Saved to " & pDataset.Workspace.PathName & pDataset.Name & ".shp... " & vbCrLf & vbCrLf
405:       End If
406:     Else ' ---------------------------------------------------- CLIPPING RASTER LAYER -------------------------
407:       Set pRasterLayer = pLayer
408:       Set pClipRasterLayer = Linkages.modClipFunctions.ClipRasterLayer(pSendPolygon, pRasterLayer, strWorkFolder, m_pApp)
409:       If pClipRasterLayer Is Nothing Then
410:         strReport = strReport & strName & vbCrLf & "  --> Unable to intersect raster layer:  No clipped grid created... " &
vbCrLf & vbCrLf
411:         Screen.MousePointer = vbDefault
412:       Else

414:         Set pRasterLayer = pLayer
       ' MAKE TABLE IF LAYER NEEDS IT
416:         Set pOrigRasterBandCollection = pRasterLayer.Raster
417:         Set pOrigRasterBand = pOrigRasterBandCollection.Item(0)
418:         pOrigRasterBand.HasTable booOrigHasVAT
419:         Set pNewRasterBandCollection = pClipRasterLayer.Raster
```

```
420:         Set pNewRasterBand = pNewRasterBandCollection.Item(0)
421:         Set pNewRasterDataset = pNewRasterBand
422:         pNewRasterBand.HasTable booNewHasVAT

        ' ONLY CONSIDER ADDING FIELDS TO TABLE IF BOTH ORIGINAL AND NEW GRIDS HAVE TABLES ALREADY
425:         If booOrigHasVAT And booNewHasVAT Then
426:           Set pOrigTable = pOrigRasterBand.AttributeTable
427:           Set pOrigFields = pOrigTable.Fields
428:           Set pNewTable = pNewRasterBand.AttributeTable
'           Set pNewFields = pNewTable.Fields
          ReDim intIndexArray(pOrigFields.FieldCount, 2)

432:           lngOrigValField = pOrigTable.FindField("Value")
433:           lngNewValField = pNewTable.FindField("Value")

435:           For lngIndex = 0 To pOrigFields.FieldCount - 1
'            MsgBox CStr(lngIndex) & vbCrLf & "Field Count = " & CStr(pOrigFields.FieldCount) & vbCrLf & _
                CStr(pOrigFields Is Nothing) & vbCrLf & CStr(pNewTable Is Nothing) & vbCrLf & "Original Value Index = " & _
                CStr(lngOrigValField) & vbCrLf & "New Value Index = " & CStr(lngNewValField)

440:             Set pOrigField = pOrigFields.Field(lngIndex)
            ' CHECK IF FIELD ALREADY EXISTS IN TABLE (I.E. OID, VALUE OR COUNT). ONLY ADD NEW FIELDS
442:             If pNewTable.FindField(pOrigField.Name) = -1 Then
443:               booAddedFields = True

              ' FOR DEBUGGING
'             strReport = strReport & "Original Layer = " & pLayer.Name & vbCrLf
'             For lngIndex2 = 0 To pOrigFields.FieldCount - 1
'               strReport = strReport & "--> " & pOrigTable.Fields.Field(lngIndex2).Name & vbCrLf
'             Next lngIndex2
'             strReport = vbCrLf & strReport & "New Layer: " & vbCrLf
'             For lngIndex2 = 0 To pNewTable.Fields.FieldCount - 1
'               strReport = strReport & "--> " & pNewTable.Fields.Field(lngIndex2).Name & vbCrLf
'             Next lngIndex2
'
'             MsgBox "Found New Field: [" & pOrigField.Name & "]" & vbCrLf & strReport

              ' MAKE CLONE OF FIELD
458:             Set pClone = pOrigField
459:             Set pNewField = pClone.Clone
460:             pQueryFilter.AddField pOrigField.Name

              ' ADD FIELD TO NEW RASTER VAT
463:             pGridTableOp.AddField pNewRasterDataset, pNewField
464:             intNewFieldIndex = intNewFieldIndex + 1

              ' KEEP RECORD OF INDEX VALUES OF ORIGINAL VAT FIELD AND NEW VAT FIELD.  USE THESE TO TRANSFER VALUES LATER
```

```
467:               intIndexArray(intNewFieldIndex, 0) = pOrigTable.FindField(pOrigField.Name)               ' ORIGINAL VAT
468:               intIndexArray(intNewFieldIndex, 1) = pNewTable.FindField(pNewField.Name)             ' NEW VAT
469:             End If
470:           Next lngIndex

472:           If booAddedFields Then
          ' CURSOR WILL WORK THROUGH ROWS IN NEW DATASET, AND QUERY FOR ROWS IN ORIGINAL DATASET USING VALUES FROM [VALUE] FIELD.
474:             Set pNewCursor = pGridTableOp.Update(pNewRasterDataset, Nothing, False)
475:             Set pNewRow = pNewCursor.NextRow
476:             Do While Not pNewRow Is Nothing
477:               lngValue = pNewRow.Value(lngNewValField)

479:               pQueryFilter.WhereClause = """Value"" = " & CStr(lngValue)
480:               Set pOrigCursor = pOrigTable.Search(pQueryFilter, False)      ' QUERY FOR VALUE
481:               Set pOrigRow = pOrigCursor.NextRow
482:               If Not pOrigRow Is Nothing Then                          ' SHOULD NEVER BE "NOTHING"
483:                 For lngIndex = 0 To intNewFieldIndex
484:                   pNewRow.Value(intIndexArray(lngIndex, 1)) = pOrigRow.Value(intIndexArray(lngIndex, 0))
  '               Debug.Print lngValue & ":  Transferring Value '" & CStr(pOrigRow.Value(intIndexArray(lngIndex, 0))) _
                  & " to new field [" & pNewTable.Fields.Field(intIndexArray(lngIndex, 1)).Name & "]"
487:               Next lngIndex
488:               pNewRow.Store
489:               pNewCursor.UpdateRow pNewRow
490:             End If

492:             Set pNewRow = pNewCursor.NextRow

494:           Loop ' LOOPING THROUGH ROWS OF NEW RASTER VAT
495:         End If ' DONE CHECKING IF NEW FIELDS WERE ADDED
496:       End If   ' DONE CHECKING IF BOTH ORIGINAL AND CLIPPED RASTER HAVE TABLES

       ' ADD LAYER TO MAP
499:       Set pClipLayer = pClipRasterLayer
500:       Set pDataset = pClipRasterLayer
501:       strClipName = pDataset.BrowseName
502:       pClipLayer.Name = strClipName
503:       m_MxDoc.AddLayer pClipLayer
504:       strReport = strReport & strName & vbCrLf & _
          "  --> Saved to " & pDataset.Workspace.PathName & pDataset.Name & "... " & vbCrLf & vbCrLf
506:     End If
507:   End If

509: Next anIndex

  ' IF POLYGON HAND-DRAWN OR SELECTED FROM GRAPHICS, ADD NEW CORRIDOR DESIGNER GRAPHIC TO SCREEN ILLUSTRATING CLIPPING AREA
512: If ext.CorrPolygonIsDrawn Then
```

```
    'MAKE SPATIAL ELEMENT
    Dim pElement As IElement
    Dim pPolygonElement As IPolygonElement
    Dim pSpatialReference As ISpatialReference
    Dim pGraphicElement As IGraphicElement
    Dim pElementProperties As IElementProperties
    Dim pGraCont As IGraphicsContainer

    Dim pClonePoly As IPolygon
523:     Set pClone = ext.PolyCorridor
524:     Set pClonePoly = pClone.Clone

    'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
527:     Set pGraCont = m_MxDoc.ActiveView
528:     Set pElement = New PolygonElement
529:     Set pClonePoly.SpatialReference = m_MxDoc.ActiveView.FocusMap.SpatialReference
530:     pElement.Geometry = pClonePoly
531:     Set pGraphicElement = pElement
532:     Set pSpatialReference = pClonePoly.SpatialReference
533:     Set pGraphicElement.SpatialReference = pSpatialReference
534:     Set pElementProperties = pElement
535:     pElementProperties.Name = "delete_corridors"

    ' ADD CORRIDOR DESIGNER POLYGON SYMBOL
    Dim pFillShapeElement As IFillShapeElement
539:     Set pFillShapeElement = Linkages.CorridorAnalysisFunctions.ApplyCorridorSymbol(pElement)

    ' ADD GRAPHIC TO GRAPHICS CONTAINER
542:     pGraCont.AddElement pFillShapeElement, 0

   'Draw
545:     m_MxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, pPolygon.Envelope


548:   End If


'  MsgBox strReport
  Dim frmReportForm As New frmReport_modal
553:   frmReportForm.txtReport.Text = strReport
554:   frmReportForm.Show vbModal

556:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
```

```
4
End Sub

Private Sub cmdSelLink_Click()
  On Error GoTo ErrorHandler

566:   m_IntHelpCategory = 1
'  Call UpdateHelpScreen
  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
571:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
574:   Set theFormObject = ext.aSelForm

576:   If (theFormObject Is Nothing) Then
577:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
    Dim pMxDoc As IMxDocument
581:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType

597:     intKey = -1

599:     Set theMap = pMxDoc.FocusMap

    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
603:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
606:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
```

```
    Dim pEnumElement As IEnumElement

609:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)

    Dim booHasPolygon As Boolean
613:     booHasPolygon = True
'    booHasPolygon = False
'
'    If (Not pEnumElement Is Nothing) Then
'      pEnumElement.Reset
'
'      Dim pElement As IElement
'      Set pElement = pEnumElement.Next
'
'      Dim pGeometry As IGeometry
'
'      Do Until pElement Is Nothing
'        Set pGeometry = pElement.Geometry
'        If TypeOf pGeometry Is IPolygon Then
'          booHasPolygon = True
'          Exit Do
'        End If
'        Set pElement = pEnumElement.Next
'      Loop
'    End If

'    MsgBox booHasPolygon

    ReDim strNameArray(theMap.LayerCount)

638:     If (booHasPolygon) Then
639:       intKey = intKey + 1
640:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
641:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
642:       strNameArray(intKey) = strPolyName
643:     End If

    Dim pFeatureLayerForValid As IFeatureLayer

647:     If (theMap.LayerCount > 0) Then
648:       Set pEnumLayer = theMap.Layers(, True)
649:       pEnumLayer.Reset

651:       Set pLayer = pEnumLayer.Next
652:       Do Until pLayer Is Nothing
653:         If TypeOf pLayer Is IFeatureLayer Then
```

```
654:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
656:           If pFeatureLayerForValid.Valid Then
            ' CHECK IF POLYGON LAYER
658:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
659:             pGeometryType = pFeatureClass.ShapeType
660:             If (pGeometryType = esriGeometryPolygon) Then
661:               intKey = intKey + 1
662:               Set pFeatureLayer = pLayer
663:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
664:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
665:               strNameArray(intKey) = strPolyName
666:             End If
667:           End If
668:         End If
669:         Set pLayer = pEnumLayer.Next
670:       Loop
671:     End If

'    Dim theReport As String
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport

679:     Set frmSelect.ArcApplication = m_pApp
680:     Set frmSelect.Doc = m_MxDoc
681:     frmSelect.NameList = strNameArray
682:     frmSelect.NameCount = intKey
683:     Set frmSelect.NameCollection = colPolygonLayers
684:     frmSelect.PolygonPurpose = "Clip"
685:     frmSelect.SearchMessage = "Species Corridor"

687:     frmSelect.EnableTool = False
688:     frmSelect.SetSelToolEnabled
689:     Load frmSelect

691:     frmSelect.Frame.Caption = "Select Species Corridor Polygon..."
692:     frmSelect.Frame.Visible = True
693:   Else
694:     Set frmSelect = theFormObject
695:   End If

'  frmSelect.Show vbModeless
698:   frmSelect.Frame.Visible = True
699:   Me.Frame.Visible = False
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "cmdSelLink_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Sub UpdateCheckmarks()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
709:   Set ext = m_ExtensionConfig

  Dim pCorPolygon As IPolygon
712:   Set pCorPolygon = ext.PolyCorridor
713:   imgCheckSpCorr.Visible = (Not ext.PolyCorridor Is Nothing)
714:   imgCheckClipLayers.Visible = lbxClipLayers.SelCount > 0

716:   imgUnCheckSpCorr.Visible = Not imgCheckSpCorr.Visible
717:   imgUncheckClipLayers.Visible = Not imgCheckClipLayers.Visible

  Dim strPathName As String
720:   strPathName = txtOutput.Text

722:   imgCheckPathname.Visible = Linkages.aml_func_mod.ExistFileDir(strPathName)
723:   imgUncheckPathname.Visible = Not imgCheckPathname.Visible

  Exit Sub
ErrorHandler:
  HandleError True, "UpdateCheckmarks " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler

733:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
734:   lblWorking.Visible = False
735:   imgIcon.Visible = False
736:   Me.Left = (Screen.Width / 3) - (Me.Width / 2)
737:   Me.Top = (Screen.Height / 2) - (Me.Height / 2)

739:   If m_Frame Is Nothing Then
740:     Set m_Frame = New ModelessFrame
741:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
744:   End If
```

```
  ' CLEAR ANY SAVED EXTENSION POLYGONS
  Dim newUid As New uID
748:   newUid.Value = "Linkages.Extension"
749:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
751:   Set ext = m_ExtensionConfig

753:   Set ext.PolyCorridor = Nothing

      ' IDENTIFY POLYGON THEMES
  Dim pMxDoc As IMxDocument
757:   Set pMxDoc = m_MxDoc

  Dim colPolygonLayers As New Collection
  Dim colClipLayers As New Collection
  Dim strNameArray() As String

  Dim theMap As IMap
  Dim pEnumLayer As IEnumLayer
  Dim pFeatureLayer As IFeatureLayer
  Dim pRasterLayer As IRasterLayer
  Dim pLayer As IUnknown
  Dim anIndex As Long
  Dim strPolyName As String
  Dim intKey As Integer
  Dim intClipIndex As Integer

  Dim pFeatureClass As IFeatureClass
  Dim pGeometryType As esriGeometryType

776:   intKey = 1
777:   intClipIndex = 0

779:   Set theMap = pMxDoc.FocusMap

  ReDim strNameArray(theMap.LayerCount + 2)

783:   colPolygonLayers.Add "placeholder_FirstLine", "FirstLine"
784:   strNameArray(0) = "FirstLine"

786:   colPolygonLayers.Add "placeholder_SelFromView", "SelFromView"
787:   strNameArray(1) = "SelFromView"

  Dim strShapeName As String

  Dim pFeatureLayerForValid As IFeatureLayer
```

```
  Dim strClipNames() As String
  ReDim strClipNames(theMap.LayerCount)
  Dim strClipName As String

797:   If (theMap.LayerCount > 0) Then
798:     Set pEnumLayer = theMap.Layers(, True)
799:     pEnumLayer.Reset

801:     Set pLayer = pEnumLayer.Next
802:     Do Until pLayer Is Nothing
803:       If TypeOf pLayer Is IFeatureLayer Then
804:         Set pFeatureLayerForValid = pLayer
        ' CHECK IF FEATURE LAYER IS VALID
806:         If pFeatureLayerForValid.Valid Then
          ' CHECK IF POLYGON LAYER
808:           Set pFeatureClass = pFeatureLayerForValid.FeatureClass
809:           pGeometryType = pFeatureClass.ShapeType
810:           strShapeName = Linkages.aml_func_mod.ReturnShapeName(pGeometryType)
811:           Set pFeatureLayer = pLayer
812:           If (pGeometryType = esriGeometryPolygon) Then
813:             intKey = intKey + 1
814:             strPolyName = CStr(intKey - 1) & "] " & pFeatureLayer.Name
815:             colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
816:             strNameArray(intKey) = strPolyName
817:           End If

         ' ADD TO CLIP LAYERS REGARDLESS OF SHAPE TYPE
820:           intClipIndex = intClipIndex + 1
821:           strClipName = CStr(intClipIndex) & "] " & pFeatureLayer.Name & "  (" & _
              strShapeName & " Dataset)"
823:           colClipLayers.Add pFeatureLayer, strClipName
824:           strClipNames(intClipIndex - 1) = strClipName
825:         End If
826:       ElseIf TypeOf pLayer Is IRasterLayer Then
827:         Set pRasterLayer = pLayer
828:         If pRasterLayer.Valid = True Then
829:           intClipIndex = intClipIndex + 1
830:           strClipName = CStr(intClipIndex) & "] " & pRasterLayer.Name & _
             " (Raster Dataset)"
832:           colClipLayers.Add pRasterLayer, strClipName
833:           strClipNames(intClipIndex - 1) = strClipName
834:         End If
835:       End If
836:       Set pLayer = pEnumLayer.Next
837:     Loop
838:   End If
```

```
840:   m_strNameArray = strNameArray
841:   Set m_colPolygons = colPolygonLayers

843:   Set m_ColClipLayers = colClipLayers

  ' FILL LISTBOXES WITH DATA SOURCE OPTIONS
  ' CORRIDOR POLYGON OPTIONS
  Dim strCorArray() As String
  ReDim strCorArray(intKey)
849:   strCorArray(0) = "Species Corridor Source..."
850:   strCorArray(1) = "<-- Select by clicking on map -->"

852:   If UBound(m_strNameArray) >= 2 Then
853:     For anIndex = 2 To UBound(m_strNameArray)
854:       If Not m_strNameArray(anIndex) = "" Then
855:         strCorArray(anIndex) = m_strNameArray(anIndex)
856:       End If
857:     Next anIndex
858:   End If

860:   cbxCorridor.Clear

862:   For anIndex = 0 To UBound(strCorArray)
863:     cbxCorridor.AddItem (strCorArray(anIndex))
864:   Next anIndex

866:   cbxCorridor.ListIndex = 0

  ' CLIP LAYERS
  ReDim Preserve strClipNames(intClipIndex)
870:   If intClipIndex > 0 Then
871:     For anIndex = 0 To intClipIndex - 1
872:       lbxClipLayers.AddItem strClipNames(anIndex)
873:     Next anIndex
874:   End If
875:   lbxClipLayers.ListIndex = -1
876:   cmdSelLink.Enabled = False

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '       IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String
  Dim strUserName As String

886:   strDirPath = ext.ClipDirectoryPath
```

```
887:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
888:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
889:   End If
890:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
891:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_pApp))
892:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
893:   End If

895:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then
896:     strDirPath = strDirPath & "\"
897:   End If

899:   txtOutput.Text = strDirPath

  ' MAKE SURE BACK COLORS ARE WHITE
902:   Call SetBackColorsWhite

904:   Call UpdateCheckmarks

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub
Public Sub EnableOKButton()
  On Error GoTo ErrorHandler


  Dim booPolysOK As Boolean
915:   booPolysOK = imgCheckSpCorr.Visible

917:   cmdOK.Enabled = booPolysOK And lbxClipLayers.SelCount > 0

'  MsgBox "optUseAll.Value = " & optUseAll.Value & vbCrLf & _
'         "(cbxPatchField.ListIndex > 1) = " & (cbxPatchField.ListIndex > 1) & vbCrLf & _
'         "(txtValue.Text <> "") = " & (txtValue.Text <> "") & vbCrLf & _
'         "((optUseAll.Value) Or ((cbxPatchField.ListIndex > 1) And (txtValue.Text <> ..))) = " & _
'         ((optUseAll.Value) Or ((cbxPatchField.ListIndex > 1) And (txtValue.Text <> ""))) & vbCrLf



  Exit Sub
ErrorHandler:
  HandleError True, "EnableOKButton " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)

935:   Set m_MxDoc = Nothing
936:   Set m_pApp = Nothing
937:   Set m_Frame = Nothing
938:   Set m_colPolygons = Nothing
939:   Set m_ColClipLayers = Nothing
940:   Set m_ExtensionConfig = Nothing

End Sub

Private Sub lbxClipLayers_Click()
  On Error GoTo ErrorHandler

947:   Call UpdateCheckmarks
948:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "lbxClipLayers_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 5: frmEsrilllustration.frm

```
VERSION 5.00
Begin VB.Form frmEsriIllustration
   Caption         =   "Illustration"
   ClientHeight    =   7905
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   8820
   Icon            =   "frmEsriIllustration.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   ScaleHeight     =   7905
   ScaleWidth      =   8820
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox txtInstructions
      Height          =   1170
      Left            =   15
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertical
      TabIndex        =   0
      Text            =   "frmEsriIllustration.frx":038A
      Top             =   6720
      Width           =   8805
```

```
   End
   Begin VB.Image imgGeneral
      BorderStyle     =   1  'Fixed Single
      Height          =   6660
      Left            =   15
      Picture         =   "frmEsriIllustration.frx":0390
      Top             =   30
      Width           =   8820
   End
   Begin VB.Image imgBottleneck
      BorderStyle     =   1  'Fixed Single
      Height          =   6660
      Left            =   15
      Picture         =   "frmEsriIllustration.frx":BC714
      Top             =   30
      Width           =   8805
   End
   Begin VB.Image imgPatch
      BorderStyle     =   1  'Fixed Single
      Height          =   6660
      Left            =   15
      Picture         =   "frmEsriIllustration.frx":178A98
      Top             =   30
      Width           =   8805
   End
End
Attribute VB_Name = "frmEsriIllustration"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Sub Form_Load()

5:   Me.Left = (Screen.Width / 3) - (Me.Width / 2)
6:   Me.Top = (Screen.Height / 2) - (Me.Height / 2)

End Sub
```

Form 6: frmEsriSample.frm

```
VERSION 5.00
Begin VB.Form frmEsriSample
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Describe Corridor - Output Options (Step 2 of 2):"
   ClientHeight    =   3090
```

```
ClientLeft      =   45
ClientTop       =   330
ClientWidth     =   8460
Icon            =   "frmEsriSample.frx":0000
LinkTopic       =   "Form1"
LockControls    =   -1  'True
MaxButton       =   0   'False
MinButton       =   0   'False
ScaleHeight     =   3090
ScaleWidth      =   8460
StartUpPosition =   3  'Windows Default
Begin VB.CommandButton cmdGeneral
   Caption         =   "General Statistics:"
   Height          =   750
   Left            =   5573
   TabIndex        =   4
   Top             =   1800
   Width           =   2580
End
Begin VB.CommandButton cmdBottleneck
   Caption         =   "Bottleneck Statistics:"
   Height          =   750
   Left            =   2910
   TabIndex        =   3
   Top             =   1800
   Width           =   2580
End
Begin VB.CommandButton cmdPatchConnection
   Caption         =   "Patch Connection Statistics:"
   Height          =   750
   Left            =   248
   TabIndex        =   2
   Top             =   1800
   Width           =   2580
End
Begin VB.CommandButton cmdClose
   Caption         =   "Close"
   Height          =   375
   Left            =   7410
   TabIndex        =   1
   Top             =   2700
   Width           =   1050
End
Begin VB.TextBox Text1
   Alignment       =   2  'Center
   BeginProperty Font
      Name            =   "Trebuchet MS"
```

```
         Size            =   14.25
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   1545
      Left            =   30
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      TabIndex        =   0
      Text            =   "frmEsriSample.frx":038A
      Top             =   75
      Width           =   8400
   End
   Begin VB.Shape Shape1
      Height          =   930
      Left            =   30
      Top             =   1710
      Width           =   8400
   End
End
Attribute VB_Name = "frmEsriSample"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdBottleneck_Click()

  Dim pFrmIllustration As New Linkages.frmEsriIllustration
6:   pFrmIllustration.Caption = "Illustration of Bottleneck Statistics:"

8:   pFrmIllustration.imgPatch.Visible = False
9:   pFrmIllustration.imgBottleneck.Visible = True
10:   pFrmIllustration.imgGeneral.Visible = False

12:   pFrmIllustration.txtInstructions.Text = "The Bottleneck statistics describe the corridor " & _
      "in terms of how wide it is along its entire path.  Corridor width is defined as 2 times " & _
      "the distance from the centerline of the corridor to the edge of the corridor, measured " & _
      "perpendicular to the direction of the centerline.  Corridor width is calculated along " & _
      "the entire length of the corridor.  In cases of multiple-strand corridors, width is " & _
      "defined according to the strand with the least constricting bottleneck." & vbCrLf & _
      "     Statistics include the average corridor width and the minimum corridor width (i.e. the bottleneck).  " & _
      "Tools are included to set a width threshold and calculate the total length that lies below that threshold."
```

```
21:   pFrmIllustration.Show vbModal

End Sub

Private Sub cmdClose_Click()
26:   Unload Me
End Sub

Private Sub cmdGeneral_Click()

  Dim pFrmIllustration As New Linkages.frmEsriIllustration
32:   pFrmIllustration.Caption = "Illustration of General Statistics:"

34:   pFrmIllustration.imgPatch.Visible = False
35:   pFrmIllustration.imgBottleneck.Visible = False
36:   pFrmIllustration.imgGeneral.Visible = True

38:   pFrmIllustration.txtInstructions.Text = "General statistics are available " & _
      "for any vector or raster background layers of interest.  Statistics are " & _
      "calculated for proportions of attributes that lie within the corridor, and include " & _
      "general statistics such as means, standard deviations and histograms."

43:   pFrmIllustration.Show vbModal

End Sub

Private Sub cmdPatchConnection_Click()

  Dim pFrmIllustration As New Linkages.frmEsriIllustration
50:   pFrmIllustration.Caption = "Illustration of Patch Connection Statistics:"

52:   pFrmIllustration.imgPatch.Visible = True
53:   pFrmIllustration.imgBottleneck.Visible = False
54:   pFrmIllustration.imgGeneral.Visible = False

56:   pFrmIllustration.txtInstructions.Text = "The Patch Connection statistics describe the " & _
      "corridor in terms of how far an animal must travel between patches of good habitat.  " & _
      "Statistics include the overall path that minimizes the gap size between patches, " & _
      "the 3 longest gaps between patches, and the gap length along the best path."

61:   pFrmIllustration.Show vbModal

End Sub

Private Sub Form_Load()
```

```
67:   Me.Left = (Screen.Width / 3) - (Me.Width / 2)
68:   Me.Top = (Screen.Height / 2) - (Me.Height / 2)
End Sub
```

Form 7: frmGraph.frm

```
VERSION 5.00
Object = "{3B7C8863-D78F-101B-B9B5-04021C009402}#1.2#0"; "RICHTX32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin VB.Form frmWidthGraph
   AutoRedraw      =   -1  'True
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Bottleneck Results:"
   ClientHeight    =   5730
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   8910
   Icon            =   "frmGraph.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5730
   ScaleWidth      =   8910
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows Default
   Begin MSComctlLib.Slider slidThreshold
      Height          =   2205
      Left            =   105
      TabIndex        =   0
      Top             =   60
      Width           =   300
      _ExtentX        =   529
      _ExtentY        =   3889
      _Version        =   393216
      Orientation     =   1
      LargeChange     =   1
      TickStyle       =   3
   End
   Begin VB.TextBox txtDistance
      Height          =   315
      Left            =   4950
      TabIndex        =   1
      Text            =   "Text1"
      Top             =   2820
      Width           =   1575
   End
```

```
Begin VB.CommandButton cmdPoints
   Caption         =   "Create Point Shapefile"
   Height          =   345
   Left            =   15
   TabIndex        =   2
   Top             =   5355
   Width           =   1815
End
Begin VB.CommandButton Command1
   Caption         =   "Create Segment Shapefile"
   Height          =   345
   Left            =   1860
   TabIndex        =   3
   Top             =   5355
   Width           =   2115
End
Begin VB.CommandButton cmdTables
   Caption         =   "Create Tables"
   Height          =   345
   Left            =   4005
   TabIndex        =   4
   Top             =   5355
   Width           =   1290
End
Begin VB.CommandButton cmdLayout
   Caption         =   "Add Graph to Layout"
   Height          =   345
   Left            =   5325
   TabIndex        =   5
   Top             =   5355
   Width           =   1680
End
Begin VB.CommandButton cmdClose
   Caption         =   "Close"
   Height          =   345
   Left            =   7035
   TabIndex        =   6
   Top             =   5355
   Width           =   720
End
Begin VB.CommandButton cmdMinimize
   Caption         =   "<< Minimize"
   Height          =   345
   Left            =   7785
   TabIndex        =   7
   Top             =   5355
   Width           =   1095
```

```
End
Begin RichTextLib.RichTextBox rtbStats
   Height          =   2130
   Left            =   105
   TabIndex        =   8
   Top             =   3150
   Width           =   2910
   _ExtentX        =   5133
   _ExtentY        =   3757
   _Version        =   393217
   Enabled         =   -1  'True
   ReadOnly        =   -1  'True
   ScrollBars      =   2
   TextRTF         =   $"frmGraph.frx":038A
End
Begin RichTextLib.RichTextBox rtbAboveStats
   Height          =   2130
   Left            =   3015
   TabIndex        =   9
   Top             =   3150
   Width           =   2895
   _ExtentX        =   5106
   _ExtentY        =   3757
   _Version        =   393217
   Enabled         =   -1  'True
   ReadOnly        =   -1  'True
   ScrollBars      =   2
   TextRTF         =   $"frmGraph.frx":0415
End
Begin RichTextLib.RichTextBox rtbBelowStats
   Height          =   2130
   Left            =   5910
   TabIndex        =   10
   Top             =   3150
   Width           =   2910
   _ExtentX        =   5133
   _ExtentY        =   3757
   _Version        =   393217
   Enabled         =   -1  'True
   ReadOnly        =   -1  'True
   ScrollBars      =   2
   TextRTF         =   $"frmGraph.frx":04AF
End
Begin VB.PictureBox pctBackColor
   BorderStyle     =   0  'None
   Height          =   1380
   Left            =   0
```

```
   ScaleHeight     =   1380
   ScaleWidth      =   3435
   TabIndex        =   14
   Top             =   0
   Width           =   3435
End
Begin VB.Label lblStats
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Threshold Statistics:"
   BeginProperty Font
      Name            =   "Arial"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   240
   Left            =   1785
   TabIndex        =   11
   Top             =   2850
   Width           =   1905
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Threshold = "
   BeginProperty Font
      Name            =   "Arial"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   240
   Left            =   3870
   TabIndex        =   12
   Top             =   2850
   Width           =   1080
End
Begin VB.Label lblUnits
   BackStyle       =   0  'Transparent
   Caption         =   "units"
```

```
      BeginProperty Font
         Name            =   "Arial"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   240
      Left            =   6600
      TabIndex        =   13
      Top             =   2850
      Width           =   1080
   End
End
Attribute VB_Name = "frmWidthGraph"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Declare Function GetDeviceCaps Lib "gdi32" (ByVal hdc As Long, ByVal nIndex As Long) As Long
Private Declare Function SetMapMode Lib "gdi32" (ByVal hdc As Long, ByVal nMapMode As Long) As Long
Private Declare Function SetBkMode Lib "gdi32" (ByVal hdc As Long, ByVal nBkMode As Long) As Long
Private Declare Function CloseEnhMetaFile Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function CreateEnhMetaFile Lib "gdi32" Alias "CreateEnhMetaFileA" _
      (ByVal hdcRef As Long, ByVal lpFileName As String, lpRect As RECT, ByVal lpDescription As String) As Long
Private Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function DeleteEnhMetaFile Lib "gdi32" (ByVal hemf As Long) As Long
Private Declare Function GetStockObject Lib "gdi32" (ByVal nIndex As Long) As Long
Private Declare Function DrawText Lib "user32" Alias "DrawTextA" _
      (ByVal hdc As Long, ByVal lpStr As String, ByVal nCount As Long, lpRect As RECT, ByVal wFormat As Long) As Long
Private Declare Function CreateFont Lib "gdi32" Alias "CreateFontA" _
      (ByVal h As Long, ByVal w As Long, ByVal E As Long, ByVal O As Long, ByVal w As Long, _
      ByVal i As Long, ByVal u As Long, ByVal S As Long, ByVal C As Long, ByVal OP As Long, _
      ByVal CP As Long, ByVal Q As Long, ByVal PAF As Long, ByVal F As String) As Long
Private Declare Function AngleArc& Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long, ByVal dwRadius As Long, ByVal
eStartAngle As Single, ByVal eSweepAngle As Single)
Private Declare Function MoveToEx Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long, lpPoint As POINTAPI) As Long
Private Declare Function CreatePatternBrush Lib "gdi32" (ByVal hBitmap As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As Long) As Long
Private Declare Function FillRect Lib "user32" (ByVal hdc As Long, lpRect As RECT, ByVal hBrush As Long) As Long
Private Declare Function GetClientRect Lib "user32" (ByVal hWnd As Long, lpRect As RECT) As Long
Private Declare Function DrawEdge& Lib "user32" (ByVal hdc As Long, qrc As RECT, ByVal edge As Long, ByVal grfFlags As Long)
```

```
Private Declare Function InflateRect Lib "user32" (lpRect As RECT, ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function DrawFrameControl Lib "user32" (ByVal hdc As Long, lpRect As RECT, ByVal un1 As Long, ByVal un2 As Long) As
Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
                    (ByVal hWnd As Long, _
                     ByVal wMsg As Long, _
                     ByVal wParam As Long, _
                     lParam As Any) As Long
Private Declare Function LineTo Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function CreatePen Lib "gdi32" (ByVal nPenStyle As Long, ByVal nWidth As Long, ByVal crColor As Long) As Long
Private Declare Function PolyBezier Lib "gdi32.dll" (ByVal hdc As Long, lppt As POINTAPI, ByVal cPoints As Long) As Long
Private Declare Function PolyBezierTo Lib "gdi32.dll" (ByVal hdc As Long, lppt As POINTAPI, ByVal cCount As Long) As Long
Private Declare Function PolyPolygon Lib "gdi32.dll" (ByVal hdc As Long, lpPoint As POINTAPI, lpPolyCounts As Long, ByVal nCount As
Long) As Long
Private Declare Function RectangleX Lib "gdi32" Alias "Rectangle" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As
Long, ByVal Y2 As Long) As Long
Private Declare Function CreateSolidBrush Lib "gdi32" (ByVal crColor As Long) As Long
Private Declare Function CreateHatchBrush Lib "gdi32" (ByVal nIndex As Long, ByVal crColor As Long) As Long
Private Declare Function BeginPath Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function EndPath Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function StrokePath Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function StrokeAndFillPath Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function ExtCreatePen Lib "gdi32" (ByVal dwPenStyle As Long, ByVal dwWidth As Long, ByRef lplb As LOGBRUSH, ByVal
dwStyleCount As Long, ByVal lpStyle As Long) As Long
Private Declare Function GetLastError Lib "Kernel32" () As Long
Private Declare Function GetTextExtentPoint32 Lib "gdi32.dll" Alias "GetTextExtentPoint32A" (ByVal hdc As Long, ByVal lpsz As String,
ByVal cbString As Long, lpSize As SIZE) As Long

Private Const PS_GEOMETRIC = &H10000
Private Const PS_ENDCAP_FLAT = &H200
Private Const HS_BDIAGONAL = 3               '  /////
Private Const HS_CROSS = 4                   '  +++++
Private Const HS_DIAGCROSS = 5               '  xxxxx
Private Const HS_FDIAGONAL = 2               '  \\\\\
Private Const HS_HORIZONTAL = 0              '  -----
Private Const HS_SOLID = 8
Private Const PS_JOIN_BEVEL = &H1000
Private Const BS_SOLID = 0
Private Type LOGBRUSH
        lbStyle As Long
        lbColor As Long
        lbHatch As Long
End Type
Private Const LTGRAY_BRUSH = 1
Private Const DT_NOPREFIX = &H800
Private Const DT_NOCLIP = &H100
Private Const DT_WORDBREAK = &H10
```

```
Private Const DT_EDITCONTROL = &H2000
Private Const DT_CENTER = &H1
Private Const DT_TOP = &H0
Private Const DT_LEFT = &H0
Private Const DT_RIGHT = &H2
Private Const DT_VCENTER = &H4
Private Const DT_BOTTOM = &H8
Private Const DT_SINGLELINE = &H20
Private Const DT_EXPANDTABS = &H40
Private Const DT_TABSTOP = &H80
Private Const DT_EXTERNALLEADING = &H200
Private Const DT_CALCRECT = &H400
Private Const DT_INTERNAL = &H1000

Private Const PS_SOLID = 0
Private Const PS_DASH = 1
Private Const PS_DOT = 2
Private Const PS_DASHDOT = 3
Private Const PS_DASHDOTDOT = 4
Private Const PS_NULL = 5

Const pi = 3.141578
Private Const EDGE_BUMP = &H9&
Private Const EDGE_ETCHED = &H6&
Private Const EDGE_RAISED = &H5&
Private Const EDGE_SUNKEN = &HA&
Private Const BF_ADJUST = &H2000
Private Const BF_BOTTOM = &H8
Private Const BF_BOTTOMLEFT = &H9
Private Const BF_BOTTOMRIGHT = &HC
Private Const BF_DIAGONAL = &H10
Private Const BF_FLAT = &H4000
Private Const BF_LEFT = &H1
Private Const BF_MIDDLE = &H800
Private Const BF_MONO = &H8000
Private Const BF_RECT = &HF
Private Const BF_RIGHT = &H4
Private Const BF_SOFT = &H1000
Private Const BF_TOP = &H2
Private Const BF_TOPLEFT = &H3
Private Const BF_TOPRIGHT = &H6
Private Const DFC_BUTTON = 4
Private Const DFC_CAPTION = 1
Private Const DFC_MENU = 2
Private Const DFC_SCROLL = 3
Private Const DFCS_BUTTON3STATE = &H8
Private Const DFCS_BUTTONCHECK = &H0
```

```
Private Const DFCS_BUTTONRADIO = &H4
Private Const DFCS_BUTTONRADIOIMAGE = &H1
Private Const DFCS_BUTTONRADIOMASK = &H2
Private Const DFCS_CAPTIONCLOSE = &H0
Private Const DFCS_CAPTIONHELP = &H4
Private Const DFCS_CAPTIONMAX = &H2
Private Const DFCS_CAPTIONMIN = &H1
Private Const DFCS_CAPTIONRESTORE = &H3
Private Const DFCS_CHECKED = &H400
Private Const DFCS_FLAT = &H4000
Private Const DFCS_INACTIVE = &H100
Private Const DFCS_MENUARROW = &H0
Private Const DFCS_MENUARROWRIGHT = &H4
Private Const DFCS_MENUBULLET = &H2
Private Const DFCS_MENUCHECK = &H1
Private Const DFCS_MONO = &H8000
Private Const DFCS_PUSHED = &H200
Private Const DFCS_SCROLLCOMBOBOX = &H5
Private Const DFCS_SCROLLDOWN = &H1
Private Const DFCS_SCROLLLEFT = &H2
Private Const DFCS_SCROLLRIGHT = &H3
Private Const DFCS_SCROLLSIZEGRIP = &H8
Private Const DFCS_SCROLLSIZEGRIPRIGHT = &H10
Private Const DFCS_SCROLLUP = &H0
Private Const TBM_SETTOOLTIPS = &H41D
Private Const LOGPIXELSY = 90        '  Logical pixels/inch in Y


Private Const GraphTop = 2
Private Const GraphLeft = 35
Private Const GraphBottom = 180
Private Const GraphRight = 590


Private Type POINTAPI
    X As Long
    Y As Long
End Type
Private Type RECT
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type
Private Type SIZE
    cx As Long
    cy As Long
```

```
End Type

' BELOW FROM JENNESS
Private m_pApp As IApplication
Private m_pMxDoc As IMxDocument
Private m_dblVals() As Double
Private m_dblGraphVals() As Double
Private m_dblMaxY As Double
Private m_dblMaxGraphY As Double
Private m_dblIncrement As Double
Private m_dblXIncrement As Double
Private m_dblGraphMaxY As Double
Private m_dblMaxX As Double
Private m_dblGraphMaxX As Double
Private m_lngPtCount As Long
Private m_dblYRange As Double
Private m_dblXRange As Double
Private m_dblThreshold As Double
Private m_dblMinY As Double
Private m_dblMeanY As Double
Private m_dblSDY As Double
Private m_dblMedY As Double
Private m_dblRangeY As Double
Private m_pAboveThreshold As esriSystem.IDoubleArray
Private m_pBelowThreshold As esriSystem.IDoubleArray
Private m_dblLengthAbove As Double
Private m_dblLengthBelow As Double
Private m_booIsScrolling As Boolean
Private m_booMinimizeToggle As Boolean
Private m_SpRef As ISpatialReference
Private m_UnitName As String
Private m_booCreateShapes As Boolean
Private m_Frame As IModelessFrame
Private m_pYLabels As esriSystem.IStringArray
Private m_pXLabels As esriSystem.IStringArray
Private m_strYName As String
Private m_strXName As String
Private m_dblSliderAdjustment As Double
Private m_pDataArray As esriSystem.IVariantArray
Private m_SegArray As esriSystem.IArray

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmGraph.frm"


Public Function Frame() As IModelessFrame
  On Error GoTo ErrorHandler
```

```
209:    If m_Frame Is Nothing Then
210:      Set m_Frame = New ModelessFrame
211:      m_Frame.Create Me
212:      m_Frame.Caption = Me.Caption
213:    End If
214:    Set Frame = m_Frame

  Exit Function
ErrorHandler:
  HandleError True, "Frame " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
Public Property Set ArcApplication(pApp As IApplication)
  On Error GoTo ErrorHandler

223:   Set m_pApp = pApp
224:   Set m_pMxDoc = m_pApp.Document

  Exit Property
ErrorHandler:
  HandleError True, "ArcApplication " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set SpatReference(pSpRef As ISpatialReference)
  On Error GoTo ErrorHandler

234:   Set m_SpRef = pSpRef
235:   If TypeOf pSpRef Is IProjectedCoordinateSystem Then
    Dim pPrCoSy As IProjectedCoordinateSystem
237:     Set pPrCoSy = pSpRef
238:     m_UnitName = LCase(pPrCoSy.CoordinateUnit.Name & "s")
'    MsgBox "TypeOf pSpRef Is IProjectedCoordinateSystem = TRUE" & vbCrLf & "m_UnitName has been set to " & CStr(m_UnitName)
240:   Else
241:     m_UnitName = "map units"
'    MsgBox "TypeOf pSpRef Is IProjectedCoordinateSystem = FALSE" & vbCrLf & "m_UnitName has been set to " & CStr(m_UnitName)
243:   End If
244:   lblUnits.Caption = m_UnitName

  Exit Property
ErrorHandler:
  HandleError True, "SpatReference " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set GraphNumbers(pDataArray As esriSystem.IVariantArray)
  On Error GoTo ErrorHandler
```

```
254:   Set m_pDataArray = pDataArray

  ReDim m_dblVals(pDataArray.Count - 1, 1)
  Dim lngIndex As Long
  Dim pDblArray As esriSystem.IDoubleArray
259:   For lngIndex = 0 To pDataArray.Count - 1
260:     Set pDblArray = pDataArray.Element(lngIndex)
261:     m_dblVals(lngIndex, 0) = pDblArray.Element(3)          ' DISTANCE
262:     m_dblVals(lngIndex, 1) = pDblArray.Element(2)          ' WIDTH
263:   Next lngIndex

'  Dim dblVals() As Double
'  dblVals = m_dblVals

268:   Set m_pAboveThreshold = New esriSystem.DoubleArray
269:   Set m_pBelowThreshold = New esriSystem.DoubleArray

271:   m_dblMaxY = 0
272:   m_dblMaxX = 0

  Dim dblTempX As Double
  Dim dblTempY As Double
276:   m_lngPtCount = UBound(m_dblVals, 1) + 1

'  m_dblMinY = m_dblVals(lngIndex, 1)

  Dim dblStatsArray() As Double
  ReDim dblStatsArray(UBound(m_dblVals, 1))

283:   For lngIndex = 0 To UBound(m_dblVals, 1)
284:     dblTempX = m_dblVals(lngIndex, 0)
285:     If dblTempX > m_dblMaxX Then m_dblMaxX = dblTempX
286:     dblStatsArray(lngIndex) = m_dblVals(lngIndex, 1)
287:   Next lngIndex

  ' CALC STATISTICS ON DISTANCE VALUES
290:   QuickSort.DoubleAscending dblStatsArray, 0, UBound(dblStatsArray)
  Dim pVarArray As esriSystem.IVariantArray
292:   Set pVarArray = Linkages.MyGeneralOperations.BasicStatsFromArray(dblStatsArray, "", "", m_pApp)
293:   m_dblMeanY = pVarArray.Element(1)
294:   m_dblMinY = pVarArray.Element(2)
295:   m_dblMaxY = pVarArray.Element(3)
296:   m_dblRangeY = pVarArray.Element(4)
297:   m_dblSDY = pVarArray.Element(6)
298:   m_dblMedY = pVarArray.Element(8)

  ' FIGURE OUT WHAT MAXIMUM Y SHOULD BE FOR THE GRAPH, SO THAT INCREMENTS CAN BE INTUITIVE.
```

```
301:   m_dblIncrement = m_dblMaxY / 300
  Dim dblPower As Double
303:   dblPower = 1
304:   Do Until (m_dblIncrement * dblPower) > 1
305:     dblPower = dblPower * 10
306:   Loop
307:   Do Until (m_dblIncrement * dblPower) < 10
308:     dblPower = dblPower / 10
309:   Loop
310:   m_dblIncrement = m_dblIncrement * dblPower

  ' NOW SHOULD HAVE AN INCREMENT BASED ON 1/400 OF RANGE, BUT BETWEEN 0 AND 10 AND WITH AN ADJUSTMENT VALUE TO RETURN IT TO ORIGINAL
313:   m_dblIncrement = (Int(m_dblIncrement) * 100) / dblPower
'  Debug.Print m_dblIncrement & ", " & m_dblMaxY

  Dim strFormat As String
317:   If m_dblIncrement < 1 Then
318:     strFormat = "0." & String(Len(CStr(Int(1 / m_dblIncrement))), "0")
319:   Else
320:     strFormat = "0"
321:   End If

323:   Set m_pYLabels = New esriSystem.strArray
324:   m_dblMaxGraphY = 0
325:   m_pYLabels.Add CStr(Format(0, strFormat))
326:   Do Until m_dblMaxGraphY >= m_dblMaxY
327:     m_dblMaxGraphY = m_dblMaxGraphY + m_dblIncrement
328:     m_pYLabels.Add CStr(Format(m_dblMaxGraphY, strFormat))
329:   Loop

  ' FIGURE OUT X-INCREMENT, SUCH THAT MAX X-VALUE WILL BE GREATER THAN LAST TIC MARK
332:   m_dblXIncrement = m_dblMaxX / 500
333:   dblPower = 1
334:   Do Until (m_dblXIncrement * dblPower) > 1
335:     dblPower = dblPower * 10
336:   Loop
337:   m_dblXIncrement = m_dblXIncrement * dblPower
338:   Do Until (m_dblXIncrement * dblPower) < 10
339:     dblPower = dblPower / 10
340:   Loop
341:   m_dblXIncrement = m_dblXIncrement * dblPower

343:   If m_dblXIncrement < 1 Then
344:     strFormat = "0." & String(Len(CStr(Int(1 / m_dblXIncrement))), "0")
345:   Else
346:     strFormat = "0"
347:   End If
```

```
  ' NOW SHOULD HAVE AN INCREMENT BASED ON 1/400 OF RANGE, BUT BETWEEN 0 AND 10 AND WITH AN ADJUSTMENT VALUE TO RETURN IT TO ORIGINAL
350:   m_dblXIncrement = (Int(m_dblXIncrement) * 100) / dblPower

352:   Set m_pXLabels = New esriSystem.strArray
  Dim dblInc As Double
354:   dblInc = m_dblXIncrement
355:   m_pXLabels.Add CStr(Format(0, strFormat))
356:   Do Until dblInc > m_dblMaxX

358:     If (m_UnitName = "meters") And (m_dblMaxX >= 10000) Then
359:       m_pXLabels.Add CStr(Format(dblInc / 1000, strFormat))
360:     Else
361:       m_pXLabels.Add CStr(Format(dblInc, strFormat))
362:     End If
363:     dblInc = dblInc + m_dblXIncrement
364:   Loop
365:   If (m_UnitName = "meters") And (m_dblMaxX >= 10000) Then
366:       m_strXName = "Distance (km)"
367:   ElseIf (m_UnitName = "meters") And (m_dblMaxX < 10000) Then
368:       m_strXName = "Distance (meters)"
369:   Else
370:       m_strXName = "Distance (" & m_UnitName & ")"
371:   End If
372:   m_strYName = "Width (" & m_UnitName & ")"

374:   rtbStats.TextRTF = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fs16 Bottleneck Width Statistics:\b0\par" & vbCrLf & _
      "   1] Minimum =  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMinY, "0.00"))) & " " & m_UnitName & "\par" & vbCrLf &
_
      "   2] Maximum =  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMaxY, "0.00"))) & " " & m_UnitName & "\par" & vbCrLf &
_
      "   3] Range =  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblRangeY, "0.00"))) & " " & m_UnitName & "\par" & vbCrLf &
_
      "   4] Mean =  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMeanY, "0.00"))) & " " & m_UnitName & "\par" & vbCrLf & _
      "   5] Median =  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMedY, "0.00"))) & " " & m_UnitName & "\par" & vbCrLf &
_
      "   6] St. Dev. =  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblSDY, "0.0000"))) & " " & m_UnitName & "\par" & vbCrLf
& _
      "\par" & vbCrLf & _
      "\b Length \b0 = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMaxX, "0.00"))) & " " & m_UnitName & "\par" & _
      "}"

387:   m_dblXRange = GraphRight - 15 - (GraphLeft + 58)
388:   m_dblYRange = GraphBottom - 38 - (GraphTop + 17)
```

```
  ReDim m_dblGraphVals(UBound(m_dblVals, 1), 1)
391:    For lngIndex = 0 To UBound(m_dblVals, 1)
392:      m_dblGraphVals(lngIndex, 0) = ConvertValToGraphX(m_dblVals(lngIndex, 0))
393:      m_dblGraphVals(lngIndex, 1) = ConvertValToGraphY(m_dblVals(lngIndex, 1))
'     Debug.Print CStr(lngIndex) & "]  Original:  X = " & CStr(m_dblVals(lngIndex, 0)) & ",   Y = " & CStr(m_dblVals(lngIndex, 1))
'     Debug.Print "        Converted:  X = " & CStr(m_dblGraphVals(lngIndex, 0)) & ",   Y = " & CStr(m_dblGraphVals(lngIndex, 1))
396:    Next lngIndex

398:    If m_dblMaxGraphY < 100 Then
399:      m_dblThreshold = Round((((m_dblMaxY - m_dblMinY) / 2) + m_dblMinY), 5)
400:    Else
401:      m_dblThreshold = Int(((m_dblMaxY - m_dblMinY) / 2) + m_dblMinY)
402:    End If
403:    txtDistance.Text = CStr(m_dblThreshold)
404:    slidThreshold.Left = 15 * (GraphLeft - 29)
405:    slidThreshold.Top = 15 * (GraphTop + 10)
406:    slidThreshold.Height = 15 * (GraphBottom - 40)

  ' HAVE TO SET MAX AT 1 INSTEAD OF 0.  DON'T KNOW WHY!
  ' HAVE TO TAKE NEGATIVE BECAUSE CAN'T SET VERTICAL SLIDER TO BEHAVE THE WAY YOU WOULD EXPECT, WITH MAX AT TOP AND MIN AT BOTTOM.
  ' THRESHOLD VALUE IS THEREFORE 1 UNIT HIGHER THAN THE SLIDER VALUE WOULD SUGGEST.  SLIDER CONTROL LOCATION MOVED TO ADJUST.

412:    slidThreshold.max = 1
  ' SET SLIDER POSITION DIFFERENTLY BASED ON WHETHER MAXIMUM VALUE IS < OR > 100
414:    If m_dblMaxGraphY < 100 Then
415:      slidThreshold.min = -998
416:      slidThreshold.Value = -Int((m_dblThreshold / m_dblMaxGraphY) * 1000) + 1
417:    Else
418:      slidThreshold.min = -(m_dblMaxGraphY) + 1    ' USE m_dblMaxGraphY
419:      slidThreshold.Value = -Int(m_dblThreshold) + 1
420:    End If
421:    slidThreshold.TickFrequency = 1

423:    SendMessage slidThreshold.hWnd, TBM_SETTOOLTIPS, 0, ByVal 0
424:    m_booIsScrolling = False
425:    m_booCreateShapes = True
426:    m_booMinimizeToggle = False
'   Debug.Print slidThreshold.TickFrequency

  ' MAKE GRAY BOX
  Dim nullPen As Long
  Dim greyBrush As Long

433:    nullPen = CreatePen(PS_NULL, 0, vbRed)
434:    DeleteObject SelectObject(hdc, nullPen)

436:    greyBrush = CreateSolidBrush(RGB(230, 230, 230))
```

```
437:   DeleteObject SelectObject(hdc, greyBrush)

'  RectangleX hdc, GraphLeft, GraphTop, GraphRight, GraphBottom
440:   RectangleX hdc, 4, 184, 592, 354

  ' DRAW INSET EDGE
  Dim di As Long
  Dim rc As RECT
'    di = GetClientRect(hwnd, rc)

447:   rc.Left = 4
448:   rc.Top = 184
449:   rc.Right = 592
450:   rc.Bottom = 354
451:   di = DrawEdge(hdc, rc, EDGE_SUNKEN, BF_TOPLEFT)
452:   di = DrawEdge(hdc, rc, EDGE_SUNKEN, BF_BOTTOMRIGHT)

454:   FillGraph

  Exit Property
ErrorHandler:
  HandleError True, "GraphNumbers " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Function ConvertValToGraphX(dblDistance As Double) As Double
  On Error GoTo ErrorHandler

464:   ConvertValToGraphX = GraphLeft + 58 + ((dblDistance / m_dblMaxX) * m_dblXRange)

  Exit Function
ErrorHandler:
  HandleError False, "ConvertValToGraphX " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Private Function ConvertValToGraphY(dblDistance As Double) As Double
  On Error GoTo ErrorHandler

'  ConvertValToGraphY = (GraphBottom - 38) - ((dblDistance / m_dblMaxY) * m_dblYRange)     ' USE m_dblMaxGraphY
474:   ConvertValToGraphY = (GraphBottom - 38) - ((dblDistance / m_dblMaxGraphY) * m_dblYRange)     ' USE m_dblMaxGraphY

  Exit Function
ErrorHandler:
  HandleError False, "ConvertValToGraphY " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
```

```
Public Sub FillGraph()
  On Error GoTo ErrorHandler

484:   pctBackColor.Visible = False

  ' DIMENSION PENS AND BRUSHES
  Dim bluePen As Long
  Dim redPen As Long
  Dim nullPen As Long
  Dim blackPen As Long
  Dim magentaPen As Long
  Dim blackDashPen As Long
  Dim whiteBrush As Long

  Dim lngFontWeight As Long
  Dim lngFontHeight As Long
  Dim nFont As Long, oFont As Long

  ' PREPARE THRESHOLD STATS VARIABLES
500:   m_pAboveThreshold.RemoveAll
501:   m_pBelowThreshold.RemoveAll
502:   m_dblLengthAbove = 0
503:   m_dblLengthBelow = 0
  Dim dblLength As Double
  Dim dblTruePrevX As Double
  Dim dblTrueCurrentX As Double
  Dim dblTrueShortX As Double
  Dim dblRunningLengthAbove As Double
  Dim dblRunningLengthBelow As Double
510:   dblRunningLengthAbove = 0
511:   dblRunningLengthBelow = 0

  ' DRAW WHITE BOX
514:   nullPen = CreatePen(PS_NULL, 0, vbRed)
515:   DeleteObject SelectObject(hdc, nullPen)

517:   whiteBrush = CreateSolidBrush(vbWhite)
518:   DeleteObject SelectObject(hdc, whiteBrush)

520:   RectangleX hdc, GraphLeft, GraphTop, GraphRight, GraphBottom

  ' GET THRESHOLD VALUE IN GRAPH UNITS
  Dim dblGraphThreshold As Double
524:   dblGraphThreshold = ConvertValToGraphY(m_dblThreshold)

  ' DRAW INSET EDGE
  Dim di As Long
```

```
  Dim rc As RECT
'    di = GetClientRect(hwnd, rc)

531:   rc.Left = GraphLeft - 2
532:   rc.Top = GraphTop - 2
533:   rc.Right = GraphRight + 2
534:   rc.Bottom = GraphBottom + 2
535:   di = DrawEdge(hdc, rc, EDGE_SUNKEN, BF_TOPLEFT)
536:   di = DrawEdge(hdc, rc, EDGE_SUNKEN, BF_BOTTOMRIGHT)

  ' DRAW X- AND Y-AXIS
539:   blackPen = CreatePen(PS_SOLID, 1, vbBlack)
540:   DeleteObject SelectObject(hdc, blackPen)
  Dim lpPoint As POINTAPI
  Dim strText As String
  Dim textSize As SIZE

  ' FONT FOR NUMBERS
546:   lngFontHeight = -((8 * GetDeviceCaps(hdc, LOGPIXELSY)) / 72)
547:   nFont = CreateFont(lngFontHeight, 0, 0, 0, 450, -1, 0, 0, 1, 7, 0, 0, 0, ByVal "Arial")
548:   oFont = SelectObject(hdc, nFont)

  ' X-AXIS
551:   MoveToEx hdc, GraphLeft + 55, GraphBottom - 38, lpPoint
552:   LineTo hdc, GraphRight - 15, GraphBottom - 38
553:   strText = m_pYLabels.Element(0)
554:   GetTextExtentPoint32 hdc, strText, Len(strText), textSize
555:   rc.Left = GraphLeft + 55 - textSize.cx - 2
556:   rc.Top = GraphBottom - 38 - (textSize.cy / 2)
557:   rc.Right = GraphLeft + 53
558:   rc.Bottom = GraphBottom - 38 + (textSize.cy / 2)
559:   DrawText hdc, strText, Len(strText), rc, DT_NOCLIP + DT_RIGHT

  ' Y-AXIS
562:   MoveToEx hdc, GraphLeft + 58, GraphTop + 17, lpPoint
563:   LineTo hdc, GraphLeft + 58, GraphBottom - 34
564:   strText = m_pXLabels.Element(0)
565:   GetTextExtentPoint32 hdc, strText, Len(strText), textSize
566:   rc.Left = GraphLeft + 58 - (textSize.cx / 2)
567:   rc.Top = GraphBottom - 32
568:   rc.Right = GraphLeft + 58 + (textSize.cx / 2)
569:   rc.Bottom = GraphBottom - 32 + textSize.cy
570:   DrawText hdc, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER
  ' Measure the text, and return it into the textSize SIZE structure

  ' MAKE DASHED HORIZONTAL LINES
  Dim lngCounter As Long
```

```
575:   lngCounter = 1

  Dim dblInc As Double
578:   dblInc = m_dblIncrement
  Dim dblVertVal As Double
580:   dblVertVal = ConvertValToGraphY(dblInc)
581:   Do Until dblInc > m_dblMaxGraphY

    ' BLACK TICS
584:     blackPen = CreatePen(PS_SOLID, 1, vbBlack)
585:     DeleteObject SelectObject(hdc, blackPen)
586:     MoveToEx hdc, GraphLeft + 55, dblVertVal, lpPoint
587:     LineTo hdc, GraphLeft + 59, dblVertVal

    ' GRAY DASHES
590:     blackDashPen = CreatePen(PS_DOT, 1, RGB(195, 195, 195))
591:     DeleteObject SelectObject(hdc, blackDashPen)
592:     LineTo hdc, GraphRight - 15, dblVertVal

    ' TEXT
595:     strText = m_pYLabels.Element(lngCounter)
596:     GetTextExtentPoint32 hdc, strText, Len(strText), textSize
597:     rc.Left = GraphLeft + 55 - textSize.cx - 2
598:     rc.Top = dblVertVal - (textSize.cy / 2)
599:     rc.Right = GraphLeft + 53
600:     rc.Bottom = dblVertVal + (textSize.cy / 2)
601:     DrawText hdc, strText, Len(strText), rc, DT_NOCLIP + DT_RIGHT

603:     dblInc = dblInc + m_dblIncrement
604:     dblVertVal = ConvertValToGraphY(dblInc)
605:     lngCounter = lngCounter + 1
606:   Loop

  ' MAKE DASHED VERTICAL LINES
609:   dblInc = m_dblXIncrement
  Dim dblHorizVal As Double
611:   dblHorizVal = ConvertValToGraphX(dblInc)
612:   lngCounter = 1

614:   Do Until dblInc > m_dblMaxX

    ' BLACK TICS
617:     blackPen = CreatePen(PS_SOLID, 1, vbBlack)
618:     DeleteObject SelectObject(hdc, blackPen)
619:     MoveToEx hdc, dblHorizVal, GraphBottom - 34, lpPoint
620:     LineTo hdc, dblHorizVal, GraphBottom - 38
```

```
    ' GRAY DASHES
623:     blackDashPen = CreatePen(PS_DOT, 1, RGB(195, 195, 195))
624:     DeleteObject SelectObject(hdc, blackDashPen)
625:     LineTo hdc, dblHorizVal, GraphTop + 17

    ' TEXT
628:     strText = m_pXLabels.Element(lngCounter)
629:     GetTextExtentPoint32 hdc, strText, Len(strText), textSize
630:     rc.Left = dblHorizVal - (textSize.cx / 2)
631:     rc.Top = GraphBottom - 32
632:     rc.Right = dblHorizVal + (textSize.cx / 2)
633:     rc.Bottom = GraphBottom - 32 + textSize.cy
634:     DrawText hdc, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

636:     dblInc = dblInc + m_dblXIncrement
637:     dblHorizVal = ConvertValToGraphX(dblInc)
638:     lngCounter = lngCounter + 1
639:   Loop

  ' FONT FOR Y-AXIS LABEL
642:   DeleteObject nFont
643:   lngFontHeight = -((11 * GetDeviceCaps(hdc, LOGPIXELSY)) / 72)
644:   nFont = CreateFont(lngFontHeight, 0, 900, 900, 700, 0, 0, 0, 1, 7, 0, 0, 0, ByVal "Arial")
645:   oFont = SelectObject(hdc, nFont)
646:   strText = m_strYName
647:   GetTextExtentPoint32 hdc, strText, Len(strText), textSize
648:   rc.Left = GraphLeft + (textSize.cx / 2)
649:   rc.Top = ((GraphBottom - GraphTop) / 2) + GraphTop + (textSize.cx / 2)
650:   rc.Right = rc.Left + textSize.cy
651:   rc.Bottom = rc.Top + textSize.cx ' (GraphBottom - GraphTop) / 2 + (textSize.cx / 2)
652:   DrawText hdc, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

'  Debug.Print "rc.Left = " & CStr(rc.Left)
'  Debug.Print "rc.Top = " & CStr(rc.Top)
'  Debug.Print "rc.Right = " & CStr(rc.Right)
'  Debug.Print "rc.Bottom = " & CStr(rc.Bottom)

  ' FONT FOR X-AXIS LABEL
660:   DeleteObject nFont
661:   lngFontHeight = -((11 * GetDeviceCaps(hdc, LOGPIXELSY)) / 72)
662:   nFont = CreateFont(lngFontHeight, 0, 0, 0, 700, 0, 0, 0, 1, 7, 0, 0, 0, ByVal "Arial")
663:   oFont = SelectObject(hdc, nFont)
664:   strText = m_strXName
665:   GetTextExtentPoint32 hdc, strText, Len(strText), textSize
666:   rc.Left = ((GraphRight - GraphLeft) / 2) + GraphLeft - (textSize.cx / 2)
667:   rc.Top = (GraphBottom - textSize.cy) - 1
668:   rc.Right = ((GraphRight - GraphLeft) / 2) + GraphLeft + (textSize.cx / 2)
```

```
669:   rc.Bottom = GraphBottom - 1
670:   DrawText hdc, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

672:   DeleteObject nFont

  ' DRAW THRESHOLD LINE
675:   magentaPen = CreatePen(PS_SOLID, 1, RGB(255, 0, 255))
676:   DeleteObject SelectObject(hdc, magentaPen)
677:   MoveToEx hdc, GraphLeft + 59, dblGraphThreshold, lpPoint
678:   LineTo hdc, GraphRight - 15, dblGraphThreshold

  ' DRAW GRAPH LINE
  Dim dblTempX As Double
  Dim dblTempY As Double
  Dim dblPrevX As Double
  Dim dblPrevY As Double
  Dim lngIndex As Long
  Dim booCurrentThresh As Boolean
  Dim booPreviousThresh As Boolean ' false for below threshold, true for above threshold

689:   dblTempY = m_dblGraphVals(0, 1)
  ' Y-AXIS REVERSED IN GRAPH SPACE!!!
691:   booCurrentThresh = (dblTempY <= dblGraphThreshold)
692:   dblPrevX = m_dblGraphVals(0, 0)
693:   dblPrevY = dblTempY
694:   dblTruePrevX = m_dblVals(0, 0)
  Dim dblProportion As Double
  Dim dblShortX As Double

698:   BeginPath hdc
699:   MoveToEx hdc, m_dblGraphVals(0, 0), m_dblGraphVals(0, 1), lpPoint

  ' FOR MAP SEGMENTS
702:   If m_booCreateShapes Then
    Dim pValueArray As esriSystem.IDoubleArray
    Dim dblMapX As Double
    Dim dblMapY As Double
    Dim dblMapShortX As Double
    Dim dblMapShortY As Double
    Dim dblMapPrevX As Double
    Dim dblMapPrevY As Double
    Dim pAboveSegment As IPointCollection
    Dim pBelowSegment As IPointCollection
    Dim pPoint As IPoint
    Dim pPrevPoint As IPoint
    Dim pShortPoint As IPoint
    Dim pClone As IClone
```

```
    Dim pSegAbovePair As esriSystem.IVariantArray
    Dim pTopoOp As ITopologicalOperator

719:     Set m_SegArray = New esriSystem.Array

721:     Set pValueArray = m_pDataArray.Element(0)
722:     dblMapX = pValueArray.Element(0)
723:     dblMapY = pValueArray.Element(1)
724:     Set pPoint = New Point
725:     pPoint.PutCoords dblMapX, dblMapY
726:     Set m_SegArray = New esriSystem.Array
727:     dblMapPrevX = dblMapX
728:     dblMapPrevY = dblMapY
729:     If booCurrentThresh Then
730:       Set pAboveSegment = New Polyline
'       pAboveSegment.AddPoint pPoint
732:     Else
733:       Set pBelowSegment = New Polyline
'       pBelowSegment.AddPoint pPoint
735:     End If
736:   End If

738:   If booCurrentThresh Then
739:     bluePen = CreatePen(PS_SOLID, 1, vbBlue)
740:     DeleteObject SelectObject(hdc, bluePen)
741:     booPreviousThresh = True
742:   Else
743:     redPen = CreatePen(PS_SOLID, 1, vbRed)
744:     DeleteObject SelectObject(hdc, redPen)
745:     booPreviousThresh = False
746:   End If

  ' FOR DEBUGGING
'  Static lngStaticCounter As Long
'  Static lngStaticCounter2 As Long
'  If m_booCreateShapes Then
'    lngStaticCounter = lngStaticCounter + 1
'    Debug.Print CStr(lngStaticCounter) & "] Creating shapes..."
'  End If
'  If Not m_booIsScrolling Then
'    lngStaticCounter2 = lngStaticCounter2 + 1
'    Debug.Print CStr(lngStaticCounter2) & "] Calculating Stats..."
'  End If

760:   For lngIndex = 0 To UBound(m_dblGraphVals, 1)
761:     dblTempX = m_dblGraphVals(lngIndex, 0)
762:     dblTempY = m_dblGraphVals(lngIndex, 1)
```

```
764:     booCurrentThresh = (dblTempY <= dblGraphThreshold)
765:     If booCurrentThresh = booPreviousThresh Then    ' THEN HAS NOT CROSSED THRESHOLD
766:       LineTo hdc, dblTempX, dblTempY

      ' IF CREATING SHAPES
769:       If m_booCreateShapes Then
770:         Set pValueArray = m_pDataArray.Element(lngIndex)
771:         dblMapX = pValueArray.Element(0)
772:         dblMapY = pValueArray.Element(1)
773:         dblTrueCurrentX = m_dblVals(lngIndex, 0)
774:         dblLength = (dblTrueCurrentX - dblTruePrevX)
775:         Set pPoint = New Point
776:         pPoint.PutCoords dblMapX, dblMapY

778:         If booCurrentThresh Then
779:           pAboveSegment.AddPoint pPoint
          ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
781:           If lngIndex = UBound(m_dblGraphVals, 1) Then
782:             Set pSegAbovePair = New esriSystem.VarArray
783:             Set pTopoOp = pAboveSegment
784:             pTopoOp.Simplify
785:             pSegAbovePair.Add pAboveSegment
786:             pSegAbovePair.Add True
787:             m_SegArray.Add pSegAbovePair
788:           End If
789:         Else
790:           pBelowSegment.AddPoint pPoint
          ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
792:           If lngIndex = UBound(m_dblGraphVals, 1) Then
793:             Set pSegAbovePair = New esriSystem.VarArray
794:             Set pTopoOp = pBelowSegment
795:             pTopoOp.Simplify
796:             pSegAbovePair.Add pBelowSegment
797:             pSegAbovePair.Add False
798:             m_SegArray.Add pSegAbovePair
799:           End If
800:         End If

802:         Set pClone = pPoint
803:         Set pPrevPoint = pClone.Clone
804:         dblMapPrevX = dblMapX
805:         dblMapPrevY = dblMapY

807:       End If

      ' IF FILLING OUT STATS BOXES, TOO...
```

```
      If Not m_booIsScrolling Then
        dblTrueCurrentX = m_dblVals(lngIndex, 0)
        dblLength = (dblTrueCurrentX - dblTruePrevX)
        If booCurrentThresh Then
          m_dblLengthAbove = m_dblLengthAbove + dblLength
          dblRunningLengthAbove = dblRunningLengthAbove + dblLength
      ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
          If lngIndex = UBound(m_dblGraphVals, 1) Then
            m_pAboveThreshold.Add dblRunningLengthAbove
          End If
        Else
          m_dblLengthBelow = m_dblLengthBelow + dblLength
          dblRunningLengthBelow = dblRunningLengthBelow + dblLength
      ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
          If lngIndex = UBound(m_dblGraphVals, 1) Then
            m_pBelowThreshold.Add dblRunningLengthBelow
          End If
        End If
        dblTruePrevX = dblTrueCurrentX
      End If

    Else    ' HAS CROSSED THRESHOLD; NEED TO MAKE TWO LINES
      dblProportion = (dblGraphThreshold - dblPrevY) / (dblTempY - dblPrevY)
      dblShortX = dblPrevX + ((dblTempX - dblPrevX) * dblProportion)
      LineTo hdc, dblShortX, dblGraphThreshold

    ' IF CREATING SHAPES
      If m_booCreateShapes Then
        Set pValueArray = m_pDataArray.Element(lngIndex)
        dblMapX = pValueArray.Element(0)
        dblMapY = pValueArray.Element(1)
        dblMapShortX = dblMapPrevX + ((dblMapX - dblMapPrevX) * dblProportion)
        dblMapShortY = dblMapPrevY + ((dblMapY - dblMapPrevY) * dblProportion)

        dblTrueCurrentX = m_dblVals(lngIndex, 0)
        dblLength = (dblTrueCurrentX - dblTruePrevX)
        Set pShortPoint = New Point
        pShortPoint.PutCoords dblMapShortX, dblMapShortY
        Set pPoint = New Point
        pPoint.PutCoords dblMapX, dblMapY

        Set pClone = pPoint
        Set pPrevPoint = pClone.Clone
        dblMapPrevX = dblMapX
        dblMapPrevY = dblMapY

      End If
```

```
      ' ADD TO CUMULATIVE DISTANCE TOTALS BASED ON WHAT PREVIOUS "OVER THRESHOLD" CONDITION WAS
859:        If Not m_booIsScrolling Then
860:          dblTrueCurrentX = m_dblVals(lngIndex, 0)
861:          dblTrueShortX = dblTruePrevX + ((dblTrueCurrentX - dblTruePrevX) * dblProportion)
862:          dblLength = (dblTrueShortX - dblTruePrevX)
863:          If booPreviousThresh Then
864:            m_dblLengthAbove = m_dblLengthAbove + dblLength
865:            dblRunningLengthAbove = dblRunningLengthAbove + dblLength
866:            m_pAboveThreshold.Add dblRunningLengthAbove
867:            dblRunningLengthAbove = 0
868:            dblRunningLengthBelow = 0
869:          Else
870:            m_dblLengthBelow = m_dblLengthBelow + dblLength
871:            dblRunningLengthBelow = dblRunningLengthBelow + dblLength
872:            m_pBelowThreshold.Add dblRunningLengthBelow
873:            dblRunningLengthAbove = 0
874:            dblRunningLengthBelow = 0
875:          End If
876:        End If

      ' END CURRENT PATH AND START ANOTHER
879:        EndPath hdc
880:        StrokePath hdc
881:        BeginPath hdc

883:        If Not m_booIsScrolling Then dblLength = (dblTrueCurrentX - dblTrueShortX)

885:        If booCurrentThresh Then ' THEN PREVIOUSLY WAS BELOW THRESHOLD, AND USING RED PEN.  NOW IS ABOVE THRESHOLD.
886:          bluePen = CreatePen(PS_SOLID, 1, vbBlue)
887:          DeleteObject SelectObject(hdc, bluePen)

        ' IF CREATING SHAPES
890:          If m_booCreateShapes Then

          ' ADD SEGMENT BEFORE CROSSING ABOVE THRESHOLD
893:            pBelowSegment.AddPoint pShortPoint
          'ADD THIS SEGMENT TO LIST
895:            Set pSegAbovePair = New esriSystem.VarArray
896:            Set pTopoOp = pBelowSegment
897:            pTopoOp.Simplify
898:            pSegAbovePair.Add pBelowSegment
899:            pSegAbovePair.Add False
900:            m_SegArray.Add pSegAbovePair

          ' ADD SEGMENT AFTER CROSSING ABOVE THRESHOLD
903:            Set pAboveSegment = New Polyline
```

```
904:           pAboveSegment.AddPoint pShortPoint
905:           pAboveSegment.AddPoint pPoint
         ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
907:           If lngIndex = UBound(m_dblGraphVals, 1) Then
908:             Set pSegAbovePair = New esriSystem.VarArray
909:             Set pTopoOp = pAboveSegment
910:             pTopoOp.Simplify
911:             pSegAbovePair.Add pAboveSegment
912:             pSegAbovePair.Add True
913:             m_SegArray.Add pSegAbovePair
914:           End If
915:         End If

       ' IF CALCULATING STATS
918:         If Not m_booIsScrolling Then
919:           m_dblLengthAbove = m_dblLengthAbove + dblLength
920:           dblRunningLengthAbove = dblRunningLengthAbove + dblLength
         ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
922:           If lngIndex = UBound(m_dblGraphVals, 1) Then
923:             m_pAboveThreshold.Add dblRunningLengthAbove
924:           End If
925:         End If
926:        Else                     ' THEN PREVIOUSLY WAS ABOVE THRESHOLD, AND USING BLUE PEN.  NOW IS BELOW THRESHOLD.
927:         redPen = CreatePen(PS_SOLID, 1, vbRed)
928:         DeleteObject SelectObject(hdc, redPen)

       ' IF CREATING SHAPES
931:         If m_booCreateShapes Then

         ' ADD SEGMENT BEFORE CROSSING BELOW THRESHOLD
934:           pAboveSegment.AddPoint pShortPoint
         'ADD THIS SEGMENT TO LIST
936:           Set pSegAbovePair = New esriSystem.VarArray
937:           Set pTopoOp = pAboveSegment
938:           pTopoOp.Simplify
939:           pSegAbovePair.Add pAboveSegment
940:           pSegAbovePair.Add True
941:           m_SegArray.Add pSegAbovePair

         ' ADD SEGMENT AFTER CROSSING ABOVE THRESHOLD
944:           Set pBelowSegment = New Polyline
945:           pBelowSegment.AddPoint pShortPoint
946:           pBelowSegment.AddPoint pPoint
         ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
948:           If lngIndex = UBound(m_dblGraphVals, 1) Then
949:             Set pSegAbovePair = New esriSystem.VarArray
950:             Set pTopoOp = pBelowSegment
```

```
951:             pTopoOp.Simplify
952:             pSegAbovePair.Add pBelowSegment
953:             pSegAbovePair.Add False
954:             m_SegArray.Add pSegAbovePair
955:           End If
956:         End If

       ' IF CALCULATING STATS
959:         If Not m_booIsScrolling Then
960:           m_dblLengthBelow = m_dblLengthBelow + dblLength
961:           dblRunningLengthBelow = dblRunningLengthBelow + dblLength
         ' ONLY ADD THIS SEGMENT TO LIST IF WE ARE ON LAST POINT
963:           If lngIndex = UBound(m_dblGraphVals, 1) Then
964:             m_pBelowThreshold.Add dblRunningLengthBelow
965:           End If
966:         End If
967:       End If

     ' IF CREATING SHAPES
970:       If m_booCreateShapes Then

972:       End If
973:       If Not m_booIsScrolling Then dblTruePrevX = dblTrueCurrentX

975:       LineTo hdc, dblTempX, dblTempY
976:     End If

978:     dblPrevY = dblTempY
979:     dblPrevX = dblTempX
980:     booPreviousThresh = booCurrentThresh

'    Debug.Print CStr(lngIndex) & "]  Original:  X = " & CStr(m_dblVals(lngIndex, 0)) & ",   Y = " & CStr(m_dblVals(lngIndex, 1))
'    Debug.Print "        Converted:  X = " & CStr(m_dblGraphVals(lngIndex, 0)) & ",   Y = " & CStr(m_dblGraphVals(lngIndex, 1))
984:   Next lngIndex

986:   EndPath hdc
987:   StrokePath hdc

'  Debug.Print "Filling Textboxes = " & CStr(Not m_booIsScrolling) & ",    Threshold = " & CStr(m_dblThreshold)

  ' IF CREATING SHAPES
992:   If m_booCreateShapes Then

994:   End If

996:   If Not m_booIsScrolling Then
    Dim lngIndex2 As Long
```

```
    Dim strAboveReport As String
    Dim strBelowReport As String

1001:     If m_pAboveThreshold.Count > 0 Then
      Dim dblAboveForSort() As Double
      ReDim dblAboveForSort(m_pAboveThreshold.Count - 1)
1004:       For lngIndex = 0 To m_pAboveThreshold.Count - 1
1005:         dblAboveForSort(lngIndex) = m_pAboveThreshold.Element(lngIndex)
1006:       Next lngIndex
1007:       Call QuickSort.DoubleDescending(dblAboveForSort, 0, UBound(dblAboveForSort))
1008:       strAboveReport = _
          "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
          "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fs16 Above Threshold Statistics:\b0\par" & vbCrLf & _
          "- Total Length = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMaxX, "0.00"))) & " " & m_UnitName & "\par" &
vbCrLf & _
          "- Total > Threshold = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblLengthAbove, "0.00"))) & "\par" & vbCrLf & _
          "- Proportion = " & Linkages.aml_func_mod.InsertCommas(CStr(Format((m_dblLengthAbove / m_dblMaxX) * 100, "0.00"))) & "%\par"
& vbCrLf & _
          "- Number of Segments = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_pAboveThreshold.Count))) & "\par" & vbCrLf & _
          "\par" & vbCrLf & "\b Segment Lengths:\b0\par" & vbCrLf
1016:       For lngIndex2 = 0 To UBound(dblAboveForSort)
1017:         strAboveReport = strAboveReport & "    " & _
          CStr(lngIndex2 + 1) & "]  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(dblAboveForSort(lngIndex2), "0.00"))) & "\par"
& vbCrLf
1019:       Next lngIndex2
1020:       strAboveReport = strAboveReport + "}"
1021:     Else
1022:       strAboveReport = _
          "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
          "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fs16 Above Threshold Statistics:\b0\par" & vbCrLf & _
          "- No portion of the route is above the threshold...\par" & vbCrLf & "}"
1026:     End If
1027:     rtbAboveStats.TextRTF = strAboveReport

1029:     If m_pBelowThreshold.Count > 0 Then
      Dim dblBelowForSort() As Double
      ReDim dblBelowForSort(m_pBelowThreshold.Count - 1)
1032:       For lngIndex = 0 To m_pBelowThreshold.Count - 1
1033:         dblBelowForSort(lngIndex) = m_pBelowThreshold.Element(lngIndex)
1034:       Next lngIndex
1035:       Call QuickSort.DoubleDescending(dblBelowForSort, 0, UBound(dblBelowForSort))
1036:       strBelowReport = _
          "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
          "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fs16 Below Threshold Statistics:\b0\par" & vbCrLf & _
          "- Total Length = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblMaxX, "0.00"))) & " " & m_UnitName & "\par" &
vbCrLf & _
          "- Total <= Threshold = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_dblLengthBelow, "0.00"))) & "\par" & vbCrLf & _
```

```
          "- Proportion = " & Linkages.aml_func_mod.InsertCommas(CStr(Format((m_dblLengthBelow / m_dblMaxX) * 100, "0.00"))) & "%\par"
& vbCrLf & _
          "- Number of Segments = " & Linkages.aml_func_mod.InsertCommas(CStr(Format(m_pBelowThreshold.Count))) & "\par" & vbCrLf & _
          "\par" & vbCrLf & "\b Segment Lengths:\b0\par" & vbCrLf
1044:       For lngIndex2 = 0 To UBound(dblBelowForSort)
1045:         strBelowReport = strBelowReport & "    " & _
          CStr(lngIndex2 + 1) & "]  " & Linkages.aml_func_mod.InsertCommas(CStr(Format(dblBelowForSort(lngIndex2), "0.00"))) & "\par"
& vbCrLf
1047:       Next lngIndex2
1048:       strBelowReport = strBelowReport + "}"
1049:     Else
1050:       strBelowReport = _
          "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
          "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fs16 Above Threshold Statistics:\b0\par" & vbCrLf & _
          "- No portion of the route is below the threshold...\par" & vbCrLf & "}"
1054:     End If
1055:     rtbBelowStats.TextRTF = strBelowReport
1056:   End If

1058:   Refresh

  ' DRAW GRAPHICS ON SCREEN
1061:   If m_booCreateShapes Then

    Dim pActiveView As esriCarto.IActiveView
1064:     Set pActiveView = m_pMxDoc.ActiveView
    Dim pGContainer As IGraphicsContainer
1066:     Set pGContainer = m_pMxDoc.FocusMap
    Dim pElement As IElement
    Dim pPolyline As IPolyline
    Dim booIsAbove As Boolean
    Dim strName As String
    Dim pLineElement As ILineElement
    Dim pSym As ISimpleLineSymbol
    Dim pElementProperties As IElementProperties
    Dim pEnvelope As IEnvelope
1075:     Set pEnvelope = New Envelope
'    MsgBox m_SegArray.Count & " segments..."
    Dim pRed As IColor
1078:     Set pRed = Linkages.MyGeneralOperations.MakeColorRGB(255, 0, 0)
    Dim pBlue As IColor
1080:     Set pBlue = Linkages.MyGeneralOperations.MakeColorRGB(0, 0, 255)

    ' DELETE EXISTING THRESHOLD ELEMENTS
1083:     pGContainer.Reset
1084:     Set pElement = pGContainer.Next
1085:     While Not pElement Is Nothing
```

```
1086:       Set pElementProperties = pElement
1087:       strName = Left(pElementProperties.Name, 17)
1088:       If (strName = "Above Threshold (") Or (strName = "Below Threshold (") Then
1089:         pGContainer.DeleteElement pElement
1090:         If (pEnvelope Is Nothing) Then
1091:           Set pEnvelope = pElement.Geometry.Envelope
1092:         Else
1093:           pEnvelope.Union pElement.Geometry.Envelope
1094:         End If
1095:       End If
1096:       Set pElement = pGContainer.Next
1097:     Wend

1099:     For lngIndex = 0 To m_SegArray.Count - 1
1100:       Set pSegAbovePair = m_SegArray.Element(lngIndex)
1101:       Set pPolyline = pSegAbovePair.Element(0)
1102:       If pPolyline.SpatialReference Is Nothing Then Set pPolyline.SpatialReference = m_SpRef
1103:       If (pEnvelope Is Nothing) Then
1104:         Set pEnvelope = pPolyline.Envelope
1105:       Else
1106:         pEnvelope.Union pPolyline.Envelope
1107:       End If
1108:       booIsAbove = pSegAbovePair.Element(1)
1109:       If booIsAbove Then
1110:         strName = "Above Threshold (>" & CStr(m_dblThreshold) & ")"
1111:       Else
1112:         strName = "Below Threshold (<=" & CStr(m_dblThreshold) & ")"
1113:       End If

1115:       Set pElement = Linkages.MyGeneralOperations.Graphic_ReturnElementFromGeometry(m_pMxDoc, pPolyline, strName, False)

1117:       Set pLineElement = pElement
1118:       Set pSym = New SimpleLineSymbol
1119:       If booIsAbove Then
1120:         pSym.Color = pBlue
1121:       Else
1122:         pSym.Color = pRed
1123:       End If
1124:       pSym.Width = 2
1125:       pSym.Style = esriSLSSolid
1126:       pLineElement.Symbol = pSym

1128:       pGContainer.AddElement pLineElement, 0
1129:     Next lngIndex

1131:     If (Not pEnvelope Is Nothing) Then
1132:       pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, pEnvelope
```

```
1133:     End If
1134:   End If

  Exit Sub
ErrorHandler:
  HandleError True, "FillGraph " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub cmdClose_Click()
  On Error GoTo ErrorHandler

1144:   Me.Frame.Visible = False
1145:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdClose_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdLayout_Click()
  On Error GoTo ErrorHandler

1155:   Call MakeEMF

  Exit Sub
ErrorHandler:
  HandleError True, "cmdLayout_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdMinimize_Click()
  On Error GoTo ErrorHandler


  ' MAKE DIALOG DISAPPEAR
  Dim pControl As Control
  Dim pWindowPos As IWindowPosition
1169:   Set pWindowPos = m_Frame

1171:   If m_booMinimizeToggle = False Then  ' THEN IT WAS MAXIMIZED, WILL CHANGE TO MINIMIZED
1172:     m_booMinimizeToggle = True
1173:     cmdMinimize.Caption = "Maximize >>"
1174:     pWindowPos.Width = (cmdMinimize.Width + 210) / 15
1175:     pWindowPos.Height = (cmdMinimize.Height + 510) / 15
1176:     cmdMinimize.Left = (240)
1177:     cmdMinimize.Top = (60)
```

```
1178:     For Each pControl In Me.Controls
1179:      pControl.Visible = False
1180:     Next pControl
1181:     pctBackColor.Visible = True
1182:     cmdMinimize.Visible = True

1184:   Else                                ' THEN IT WAS MINIMIZED, WILL CHANGE TO MAXIMIZED
1185:     m_booMinimizeToggle = False
1186:     cmdMinimize.Caption = "<< Minimize"
1187:     pWindowPos.Width = (9000) / 15
1188:     pWindowPos.Height = (6105) / 15
1189:     cmdMinimize.Left = (7785)
1190:     cmdMinimize.Top = (5355)
1191:     For Each pControl In Me.Controls
1192:      pControl.Visible = True
1193:     Next pControl
1194:     pctBackColor.Visible = False
1195:     FillGraph
1196:   End If

1198:   Refresh

  Exit Sub
ErrorHandler:
  HandleError True, "cmdMinimize_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdPoints_Click()
  On Error GoTo ErrorHandler


  Dim strReport As String

  Dim strFName As String

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String

  Dim newUid As New uID
1221:   newUid.Value = "Linkages.Extension"
  Dim ext As Linkages.Extension
1223:   Set ext = m_pApp.FindExtensionByCLSID(newUid)
```

```
  ' FOR USE WHEN WRAPPED INTO FULL EXTENSION
1226:   strDirPath = ext.ClipDirectoryPath
1227:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1228:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
1229:   End If
1230:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1231:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_pApp))
1232:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
1233:   End If

  ' USE BELOW UNTIL WRAPPED INTO FULL EXTENSION
'  strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.GetMxDocPath(m_pApp))
'  strDirPath = aml_func_mod.ReturnDir(strDirPath)
  '-----------------------------------------------------

1240:   If Right(strDirPath, 1) <> "/" And Right(strDirPath, 1) <> "\" Then strDirPath = strDirPath & "\"

  Dim lngIndex As Long

  Dim pMap As IMap
1245:   Set pMap = m_pMxDoc.FocusMap
  Dim pActiveView As IActiveView
1247:   Set pActiveView = m_pMxDoc.ActiveView

1249:   strFName = strDirPath & "bottleneck_points.shp"
1250:   strFName = Linkages.aml_func_mod.MakeUniqueFilename(strFName)
  Dim strJustFilename As String
1252:   strJustFilename = Linkages.aml_func_mod.ReturnFilename(strFName)

  ' MAKE SHAPEFILE
  Dim pPointsFClass As IFeatureClass
1256:   Set pPointsFClass = Linkages.aml_func_mod.CreateShapefile(strDirPath, strJustFilename, m_SpRef, "Point")

    ' MAKE FIELDS
    ' MAKE X_Coord FIELD
  Dim pPointsXField As IField
  Dim pPointsXFieldEdit As IFieldEdit
1262:   Set pPointsXField = New Field
1263:   Set pPointsXFieldEdit = pPointsXField
1264:   With pPointsXFieldEdit
1265:     .Name = "X_Coord"
1266:     .Type = esriFieldTypeDouble
1267:     .Precision = 16
1268:     .Scale = 8
1269:   End With
1270:   pPointsFClass.AddField pPointsXField
```

```
    ' MAKE Y_Coord FIELD
  Dim pPointsYField As IField
  Dim pPointsYFieldEdit As IFieldEdit
1275:   Set pPointsYField = New Field
1276:   Set pPointsYFieldEdit = pPointsYField
1277:   With pPointsYFieldEdit
1278:     .Name = "Y_Coord"
1279:     .Type = esriFieldTypeDouble
1280:     .Precision = 16
1281:     .Scale = 8
1282:   End With
1283:   pPointsFClass.AddField pPointsYField

    ' MAKE WIDTH FIELD
  Dim pPointsWidthField As IField
  Dim pPointsWidthFieldEdit As IFieldEdit
1288:   Set pPointsWidthField = New Field
1289:   Set pPointsWidthFieldEdit = pPointsWidthField
1290:   With pPointsWidthFieldEdit
1291:     .Name = "Width"
1292:     .Type = esriFieldTypeDouble
1293:     .Precision = 16
1294:     .Scale = 8
1295:   End With
1296:   pPointsFClass.AddField pPointsWidthField

  ' FIND VARIABLE INDEX VALUES
  Dim lngPointsID As Long
  Dim lngPointsX As Long
  Dim lngPointsY As Long
  Dim lngPointsWidth As Long
1303:   lngPointsID = pPointsFClass.FindField("Unique_ID")
1304:   lngPointsX = pPointsFClass.FindField("X_Coord")
1305:   lngPointsY = pPointsFClass.FindField("Y_Coord")
1306:   lngPointsWidth = pPointsFClass.FindField("Width")

  ' MAKE INSERT CURSOR
  Dim pPointsCur As IFeatureCursor
1310:   Set pPointsCur = pPointsFClass.Insert(True)
  Dim pPointsBuf As IFeatureBuffer
1312:   Set pPointsBuf = pPointsFClass.CreateFeatureBuffer

  Dim pPoint As IPoint
  Dim pValArray  As esriSystem.IDoubleArray
  Dim dblX As Double
  Dim dblY As Double
```

```
  Dim dblWidth As Double

  ' ADD DATA TO SHAPEFILE
1321:   For lngIndex = 0 To m_pDataArray.Count - 1
1322:     Set pValArray = m_pDataArray.Element(lngIndex)
1323:     dblX = pValArray.Element(0)
1324:     dblY = pValArray.Element(1)
1325:     dblWidth = pValArray.Element(2)

1327:     Set pPoint = New Point
1328:     pPoint.PutCoords dblX, dblY
1329:     Set pPoint.SpatialReference = m_SpRef
1330:     With pPointsBuf
1331:       Set .Shape = pPoint
1332:       .Value(lngPointsID) = lngIndex + 1
1333:       .Value(lngPointsX) = dblX
1334:       .Value(lngPointsY) = dblY
1335:       .Value(lngPointsWidth) = dblWidth
1336:     End With
1337:     pPointsCur.InsertFeature pPointsBuf
1338:   Next lngIndex

1340:   pPointsCur.Flush
  Dim pFeatureLayer As IFeatureLayer
1342:   Set pFeatureLayer = New FeatureLayer
1343:   Set pFeatureLayer.FeatureClass = pPointsFClass
1344:   pFeatureLayer.Name = Linkages.aml_func_mod.ClipExtension(strJustFilename)
1345:   pMap.AddLayer pFeatureLayer

1347:   pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, Nothing

  ' FOR REPORT
'  strReport = _
'       "The shapefile " & strJustFilename & "' has been generated and added to your view.  Individual points " & _
'       "include X-coordinates, Y-coordinates and Corridor Width values at each location." & vbCrLf & vbCrLf & _
'       "  \i Shapefile Saved to " & Replace(strFName, "\", "\\") & "\i0\par" & vbCrLf

  Dim pDocDirty As IDocumentDirty
1356:   Set pDocDirty = m_pMxDoc
1357:   pDocDirty.SetDirty

  ' FOR REPORT
1360:   strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\tx90\tx360\tx450\tx720\f0\fs16 " & _
      "The shapefile \b " & strJustFilename & "\b0  has been generated and added to your view.  Individual points " & _
      "include X-coordinates, Y-coordinates and Corridor Width values at each location.\par" & vbCrLf & "\par" & vbCrLf & _
```

```
      "\b Shapefile saved to:\b0 \i  " & Replace(strFName, "\", "\\") & "\i0\par" & vbCrLf & vbCrLf & "}"

    ' SHOW REPORT
  Dim frmReportForm As New Linkages.frmReport_modal
1369:   frmReportForm.txtReport.TextRTF = strReport
1370:   frmReportForm.Caption = "Operation Successful:"
1371:   frmReportForm.Show vbModal

  Exit Sub
ErrorHandler:
  HandleError True, "cmdPoints_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdTables_Click()
  On Error GoTo ErrorHandler

  Dim strReport As String

'  strReport = "The following tables have been created and added as Standalone Tables to your Map document.  " & _
      "All standalone tables are available by clicking the 'Source' tab at the bottom of the Map Table of Contents." & _
      vbCrLf & "-------------------------------------------------------" & vbCrLf

1387:   strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\tx90\tx360\tx450\tx720\f0\fs16 " & _
      "The following tables have been created and added as Standalone Tables to your Map document.  " & _
      "All standalone tables are available by clicking the 'Source' tab at the bottom of the Map Table of Contents.\par" & _
      vbCrLf & "\par" & vbCrLf & "-------------------------------------------------------\par" & vbCrLf

  Dim strFName As String

  Dim newUid As New uID
1397:   newUid.Value = "Linkages.Extension"
  Dim ext As Linkages.Extension
1399:   Set ext = m_pApp.FindExtensionByCLSID(newUid)

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String

  ' FOR USE WHEN WRAPPED INTO FULL EXTENSION
1409:   strDirPath = ext.ClipDirectoryPath
1410:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
```

```
1411:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
1412:   End If
1413:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1414:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_pApp))
1415:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
1416:   End If

  ' USE BELOW UNTIL WRAPPED INTO FULL EXTENSION
'  strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.GetMxDocPath(m_pApp))
'  strDirPath = aml_func_mod.ReturnDir(strDirPath)
  '-------------------------------------------------

1423:   If Right(strDirPath, 1) <> "/" And Right(strDirPath, 1) <> "\" Then strDirPath = strDirPath & "\"

  Dim lngCounter As Long
1426:   lngCounter = 0
  Dim lngIndex As Long

  Dim pStandaloneTableCollection As IStandaloneTableCollection
1430:   Set pStandaloneTableCollection = m_pMxDoc.FocusMap

  ' SEGMENT LENGTHS TABLE ------------------------------------------------------------------------------
1433:   strFName = strDirPath & "threshold_lengths.dbf"
1434:   strFName = aml_func_mod.MakeUniqueFilename(strFName)
    ' MAKE FIELDS
  Dim pSegsFields As IFields
  Dim pSegsFieldsEdit As IFieldsEdit
1438:   Set pSegsFields = New Fields
1439:   Set pSegsFieldsEdit = pSegsFields
1440:   pSegsFieldsEdit.FieldCount = 3

    ' MAKE UNIQUE ID FIELD
  Dim pSegsUniqueIDField As IField
  Dim pSegsUniqueIDFieldEdit As IFieldEdit
1445:   Set pSegsUniqueIDField = New Field
1446:   Set pSegsUniqueIDFieldEdit = pSegsUniqueIDField
1447:   With pSegsUniqueIDFieldEdit
1448:     .Name = "Unique_ID"
1449:     .Type = esriFieldTypeInteger
1450:     .Precision = 8
1451:   End With

    ' MAKE NAME FIELD
  Dim pSegsNameField As IField
  Dim pSegsNameFieldEdit As IFieldEdit
1456:   Set pSegsNameField = New Field
1457:   Set pSegsNameFieldEdit = pSegsNameField
```

```
1458:   With pSegsNameFieldEdit
1459:     .Name = "Name"
1460:     .Type = esriFieldTypeString
1461:     .Precision = 25
1462:   End With

    ' MAKE LENGTH FIELD
  Dim pSegsLengthField As IField
  Dim pSegsLengthFieldEdit As IFieldEdit
1467:   Set pSegsLengthField = New Field
1468:   Set pSegsLengthFieldEdit = pSegsLengthField
1469:   With pSegsLengthFieldEdit
1470:     .Name = "Length"
1471:     .Type = esriFieldTypeDouble
1472:     .Precision = 16
1473:     .Scale = 8
1474:   End With

1476:   Set pSegsFieldsEdit.Field(0) = pSegsUniqueIDField
1477:   Set pSegsFieldsEdit.Field(1) = pSegsNameField
1478:   Set pSegsFieldsEdit.Field(2) = pSegsLengthField

  ' MAKE dBASE TABLE
  Dim pSegsTable As ITable
1482:   Set pSegsTable = aml_func_mod.CreatedBASETable(strFName, pSegsFields)

  ' MAKE INSERT CURSOR
  Dim pSegsCur As ICursor
1486:   Set pSegsCur = pSegsTable.Insert(True)
  Dim pSegsBuf As IRowBuffer
1488:   Set pSegsBuf = pSegsTable.CreateRowBuffer

  ' FIND VARIABLE INDEX VALUES
  Dim lngSegsID As Long
  Dim lngSegsName As Long
  Dim lngSegsLength As Long
1494:   lngSegsID = pSegsTable.FindField("Unique_ID")
1495:   lngSegsName = pSegsTable.FindField("Name")
1496:   lngSegsLength = pSegsTable.FindField("Length")

  ' ADD DATA TO TABLE
1499:   lngCounter = lngCounter + 1
1500:   pSegsBuf.Value(lngSegsID) = lngCounter
1501:   pSegsBuf.Value(lngSegsName) = "Total Length"
1502:   pSegsBuf.Value(lngSegsLength) = m_dblMaxX
1503:   pSegsCur.InsertRow pSegsBuf
```

```
1505:   lngCounter = lngCounter + 1
1506:   pSegsBuf.Value(lngSegsID) = lngCounter
1507:   pSegsBuf.Value(lngSegsName) = "Threshold Value"
1508:   pSegsBuf.Value(lngSegsLength) = m_dblThreshold
1509:   pSegsCur.InsertRow pSegsBuf

1511:   If m_pAboveThreshold.Count > 0 Then
    Dim dblAboveForSort() As Double
    ReDim dblAboveForSort(m_pAboveThreshold.Count - 1)
1514:     For lngIndex = 0 To m_pAboveThreshold.Count - 1
1515:       dblAboveForSort(lngIndex) = m_pAboveThreshold.Element(lngIndex)
1516:     Next lngIndex
1517:     Call QuickSort.DoubleDescending(dblAboveForSort, 0, UBound(dblAboveForSort))

1519:     For lngIndex = 0 To UBound(dblAboveForSort)
1520:       lngCounter = lngCounter + 1
1521:       pSegsBuf.Value(lngSegsID) = lngCounter
1522:       pSegsBuf.Value(lngSegsName) = "Above Threshold #" & CStr(lngIndex + 1)
1523:       pSegsBuf.Value(lngSegsLength) = dblAboveForSort(lngIndex)
1524:       pSegsCur.InsertRow pSegsBuf
1525:     Next lngIndex
1526:   End If

1528:   If m_pBelowThreshold.Count > 0 Then
    Dim dblBelowForSort() As Double
    ReDim dblBelowForSort(m_pBelowThreshold.Count - 1)
1531:     For lngIndex = 0 To m_pBelowThreshold.Count - 1
1532:       dblBelowForSort(lngIndex) = m_pBelowThreshold.Element(lngIndex)
1533:     Next lngIndex
1534:     Call QuickSort.DoubleDescending(dblBelowForSort, 0, UBound(dblBelowForSort))

1536:     For lngIndex = 0 To UBound(dblBelowForSort)
1537:       lngCounter = lngCounter + 1
1538:       pSegsBuf.Value(lngSegsID) = lngCounter
1539:       pSegsBuf.Value(lngSegsName) = "Below Threshold #" & CStr(lngIndex + 1)
1540:       pSegsBuf.Value(lngSegsLength) = dblBelowForSort(lngIndex)
1541:       pSegsCur.InsertRow pSegsBuf
1542:     Next lngIndex
1543:   End If

1545:   pSegsCur.Flush

  ' MAKE STANDALONE TABLE
  Dim pSegsStandaloneTable As IStandaloneTable
  Dim pSegsTableWindow As ITableWindow2

1551:   Set pSegsStandaloneTable = New StandaloneTable
```

```
1552:   Set pSegsStandaloneTable.Table = pSegsTable
1553:   Set pSegsTableWindow = New TableWindow
1554:   With pSegsTableWindow
1555:     Set .StandaloneTable = pSegsStandaloneTable
1556:     Set .Application = m_pApp
1557:     .TableSelectionAction = esriSelectFeatures
1558:     .ShowAliasNamesInColumnHeadings = True
1559:     .ShowSelected = False
1560:     .Show True
1561:   End With
1562:   pStandaloneTableCollection.AddStandaloneTable pSegsStandaloneTable

  ' FOR REPORT
1565:   strReport = strReport & _
  "\b  1] Table of Segment Lengths:\b0\par" & vbCrLf & _
  "     \i Saved to " & Replace(strFName, "\", "\\") & "\i0\par" & vbCrLf



  ' GENERAL STATISTICS TABLE ------------------------------------------------------------------------------
1572:   strFName = strDirPath & "bottleneck_stats.dbf"
1573:   strFName = aml_func_mod.MakeUniqueFilename(strFName)
    ' MAKE FIELDS
  Dim pStatsFields As IFields
  Dim pStatsFieldsEdit As IFieldsEdit
1577:   Set pStatsFields = New Fields
1578:   Set pStatsFieldsEdit = pStatsFields
1579:   pStatsFieldsEdit.FieldCount = 3

    ' MAKE UNIQUE ID FIELD
  Dim pStatsUniqueIDField As IField
  Dim pStatsUniqueIDFieldEdit As IFieldEdit
1584:   Set pStatsUniqueIDField = New Field
1585:   Set pStatsUniqueIDFieldEdit = pStatsUniqueIDField
1586:   With pStatsUniqueIDFieldEdit
1587:     .Name = "Unique_ID"
1588:     .Type = esriFieldTypeInteger
1589:     .Precision = 8
1590:   End With

    ' MAKE Statistics FIELD
  Dim pStatsStatisticsField As IField
  Dim pStatsStatisticsFieldEdit As IFieldEdit
1595:   Set pStatsStatisticsField = New Field
1596:   Set pStatsStatisticsFieldEdit = pStatsStatisticsField
1597:   With pStatsStatisticsFieldEdit
1598:     .Name = "Statistic"
```

```
1599:     .Type = esriFieldTypeString
1600:     .Precision = 30
1601:   End With

    ' MAKE Value FIELD
  Dim pStatsValueField As IField
  Dim pStatsValueFieldEdit As IFieldEdit
1606:   Set pStatsValueField = New Field
1607:   Set pStatsValueFieldEdit = pStatsValueField
1608:   With pStatsValueFieldEdit
1609:     .Name = "Value"
1610:     .Type = esriFieldTypeDouble
1611:     .Precision = 16
1612:     .Scale = 8
1613:   End With

1615:   Set pStatsFieldsEdit.Field(0) = pStatsUniqueIDField
1616:   Set pStatsFieldsEdit.Field(1) = pStatsStatisticsField
1617:   Set pStatsFieldsEdit.Field(2) = pStatsValueField

  ' MAKE dBASE TABLE
  Dim pStatsTable As ITable
1621:   Set pStatsTable = aml_func_mod.CreatedBASETable(strFName, pStatsFields)

  ' MAKE INSERT CURSOR
  Dim pStatsCur As ICursor
1625:   Set pStatsCur = pStatsTable.Insert(True)
  Dim pStatsBuf As IRowBuffer
1627:   Set pStatsBuf = pStatsTable.CreateRowBuffer

  ' FIND VARIABLE INDEX VALUES
  Dim lngStatsID As Long
  Dim lngStatsStatistics As Long
  Dim lngStatsValue As Long
1633:   lngStatsID = pStatsTable.FindField("Unique_ID")
1634:   lngStatsStatistics = pStatsTable.FindField("Statistic")
1635:   lngStatsValue = pStatsTable.FindField("Value")

  ' ADD DATA TO TABLE
  ' MINIMUM
1639:   lngCounter = lngCounter + 1
1640:   pStatsBuf.Value(lngStatsID) = lngCounter
1641:   pStatsBuf.Value(lngStatsStatistics) = "Minimum Width"
1642:   pStatsBuf.Value(lngStatsValue) = m_dblMinY
1643:   pStatsCur.InsertRow pStatsBuf

  ' MAXIMUM
```

```
1646:   lngCounter = lngCounter + 1
1647:   pStatsBuf.Value(lngStatsID) = lngCounter
1648:   pStatsBuf.Value(lngStatsStatistics) = "Maximum Width"
1649:   pStatsBuf.Value(lngStatsValue) = m_dblMaxY
1650:   pStatsCur.InsertRow pStatsBuf

  ' RANGE
1653:   lngCounter = lngCounter + 1
1654:   pStatsBuf.Value(lngStatsID) = lngCounter
1655:   pStatsBuf.Value(lngStatsStatistics) = "Range of Width Values"
1656:   pStatsBuf.Value(lngStatsValue) = m_dblRangeY
1657:   pStatsCur.InsertRow pStatsBuf

  ' MEAN
1660:   lngCounter = lngCounter + 1
1661:   pStatsBuf.Value(lngStatsID) = lngCounter
1662:   pStatsBuf.Value(lngStatsStatistics) = "Mean Width"
1663:   pStatsBuf.Value(lngStatsValue) = m_dblMeanY
1664:   pStatsCur.InsertRow pStatsBuf

  ' MEDIAN
1667:   lngCounter = lngCounter + 1
1668:   pStatsBuf.Value(lngStatsID) = lngCounter
1669:   pStatsBuf.Value(lngStatsStatistics) = "Median Width"
1670:   pStatsBuf.Value(lngStatsValue) = m_dblMedY
1671:   pStatsCur.InsertRow pStatsBuf

  ' STANDARD DEVIATION
1674:   lngCounter = lngCounter + 1
1675:   pStatsBuf.Value(lngStatsID) = lngCounter
1676:   pStatsBuf.Value(lngStatsStatistics) = "Standard Deviation of Width"
1677:   pStatsBuf.Value(lngStatsValue) = m_dblSDY
1678:   pStatsCur.InsertRow pStatsBuf

  ' CENTERLINE LENGTH
1681:   lngCounter = lngCounter + 1
1682:   pStatsBuf.Value(lngStatsID) = lngCounter
1683:   pStatsBuf.Value(lngStatsStatistics) = "Centerline Length"
1684:   pStatsBuf.Value(lngStatsValue) = m_dblMaxX
1685:   pStatsCur.InsertRow pStatsBuf

  ' THRESHOLD VALUE
1688:   lngCounter = lngCounter + 1
1689:   pStatsBuf.Value(lngStatsID) = lngCounter
1690:   pStatsBuf.Value(lngStatsStatistics) = "Width Threshold Value"
1691:   pStatsBuf.Value(lngStatsValue) = m_dblThreshold
1692:   pStatsCur.InsertRow pStatsBuf
```

```
  ' LENGTH BELOW THRESHOLD
1695:   lngCounter = lngCounter + 1
1696:   pStatsBuf.Value(lngStatsID) = lngCounter
1697:   pStatsBuf.Value(lngStatsStatistics) = "Length Below Threshold"
1698:   pStatsBuf.Value(lngStatsValue) = m_dblLengthBelow
1699:   pStatsCur.InsertRow pStatsBuf

  ' PROPORTION BELOW THRESHOLD
1702:   lngCounter = lngCounter + 1
1703:   pStatsBuf.Value(lngStatsID) = lngCounter
1704:   pStatsBuf.Value(lngStatsStatistics) = "Proportion Below Threshold"
1705:   pStatsBuf.Value(lngStatsValue) = (m_dblLengthBelow / m_dblMaxX)
1706:   pStatsCur.InsertRow pStatsBuf

  ' LENGTH ABOVE THRESHOLD
1709:   lngCounter = lngCounter + 1
1710:   pStatsBuf.Value(lngStatsID) = lngCounter
1711:   pStatsBuf.Value(lngStatsStatistics) = "Length Above Threshold"
1712:   pStatsBuf.Value(lngStatsValue) = m_dblLengthAbove
1713:   pStatsCur.InsertRow pStatsBuf

  ' PROPORTION ABOVE THRESHOLD
1716:   lngCounter = lngCounter + 1
1717:   pStatsBuf.Value(lngStatsID) = lngCounter
1718:   pStatsBuf.Value(lngStatsStatistics) = "Proportion Above Threshold"
1719:   pStatsBuf.Value(lngStatsValue) = (m_dblLengthAbove / m_dblMaxX)
1720:   pStatsCur.InsertRow pStatsBuf

1722:   pStatsCur.Flush

  ' MAKE STANDALONE TABLE
  Dim pStatsStandaloneTable As IStandaloneTable
  Dim pStatsTableWindow As ITableWindow2

1728:   Set pStatsStandaloneTable = New StandaloneTable
1729:   Set pStatsStandaloneTable.Table = pStatsTable
1730:   Set pStatsTableWindow = New TableWindow
1731:   With pStatsTableWindow
1732:     Set .StandaloneTable = pStatsStandaloneTable
1733:     Set .Application = m_pApp
1734:     .TableSelectionAction = esriSelectFeatures
1735:     .ShowAliasNamesInColumnHeadings = True
1736:     .ShowSelected = False
1737:     .Show True
1738:   End With
1739:   pStandaloneTableCollection.AddStandaloneTable pStatsStandaloneTable
```

```
  Dim pDocDirty As IDocumentDirty
1742:   Set pDocDirty = m_pMxDoc
1743:   pDocDirty.SetDirty

  ' FOR REPORT
1746:   strReport = strReport & _
  "\b  2] Table of Bottleneck Statistics:\b0\par" & vbCrLf & _
  "     \i Saved to " & Replace(strFName, "\", "\\") & "\i0\par" & vbCrLf & "}"

    ' SHOW REPORT
  Dim frmReportForm As New Linkages.frmReport_modal
1752:   frmReportForm.txtReport.TextRTF = strReport
1753:   frmReportForm.Caption = "Operation Successful:"
1754:   frmReportForm.Show vbModal

  Exit Sub
ErrorHandler:
  HandleError True, "cmdTables_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Private Sub Command1_Click()
  On Error GoTo ErrorHandler

  Dim strReport As String

  Dim strFName As String


  Dim newUid As New uID
1771:   newUid.Value = "Linkages.Extension"
  Dim ext As Linkages.Extension
1773:   Set ext = m_pApp.FindExtensionByCLSID(newUid)

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String

  ' FOR USE WHEN WRAPPED INTO FULL EXTENSION
1783:   strDirPath = ext.ClipDirectoryPath
1784:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1785:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
```

```
1786:   End If
1787:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1788:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_pApp))
1789:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
1790:   End If

  ' USE BELOW UNTIL WRAPPED INTO FULL EXTENSION
'  strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.GetMxDocPath(m_pApp))
'  strDirPath = aml_func_mod.ReturnDir(strDirPath)
  '-----------------------------------------------------

1797:   If Right(strDirPath, 1) <> "/" And Right(strDirPath, 1) <> "\" Then strDirPath = strDirPath & "\"

  Dim lngIndex As Long

  Dim pMap As IMap
1802:   Set pMap = m_pMxDoc.FocusMap
  Dim pActiveView As IActiveView
1804:   Set pActiveView = m_pMxDoc.ActiveView

1806:   strFName = strDirPath & "bottleneck_segments.shp"
1807:   strFName = aml_func_mod.MakeUniqueFilename(strFName)
  Dim strJustFilename As String
1809:   strJustFilename = aml_func_mod.ReturnFilename(strFName)

  ' MAKE SHAPEFILE
  Dim pSegsFClass As IFeatureClass
1813:   Set pSegsFClass = aml_func_mod.CreateShapefile(strDirPath, strJustFilename, m_SpRef, "Polyline")

    ' MAKE FIELDS
    ' MAKE NAME FIELD
  Dim pSegsNameField As IField
  Dim pSegsNameFieldEdit As IFieldEdit
1819:   Set pSegsNameField = New Field
1820:   Set pSegsNameFieldEdit = pSegsNameField
1821:   With pSegsNameFieldEdit
1822:     .Name = "Name"
1823:     .Type = esriFieldTypeString
1824:     .Precision = 25
1825:   End With
1826:   pSegsFClass.AddField pSegsNameField

    ' MAKE LENGTH FIELD
  Dim pSegsLengthField As IField
  Dim pSegsLengthFieldEdit As IFieldEdit
1831:   Set pSegsLengthField = New Field
1832:   Set pSegsLengthFieldEdit = pSegsLengthField
```

```
1833:   With pSegsLengthFieldEdit
1834:     .Name = "Length"
1835:     .Type = esriFieldTypeDouble
1836:     .Precision = 16
1837:     .Scale = 8
1838:   End With
1839:   pSegsFClass.AddField pSegsLengthField

  Dim pEnvelope As IEnvelope
1842:   Set pEnvelope = New Envelope

  ' FIND VARIABLE INDEX VALUES
  Dim lngSegsID As Long
  Dim lngSegsName As Long
  Dim lngSegsLength As Long
1848:   lngSegsID = pSegsFClass.FindField("Unique_ID")
1849:   lngSegsName = pSegsFClass.FindField("Name")
1850:   lngSegsLength = pSegsFClass.FindField("Length")

  ' MAKE INSERT CURSOR
  Dim pSegsCur As IFeatureCursor
1854:   Set pSegsCur = pSegsFClass.Insert(True)
  Dim pSegsBuf As IFeatureBuffer
1856:   Set pSegsBuf = pSegsFClass.CreateFeatureBuffer

  Dim pPolyline As IPolyline
  Dim pSegAbovePair As esriSystem.IVariantArray
  Dim booIsAbove As Boolean
  Dim strName As String

  ' ADD DATA TO SHAPEFILE
1864:   For lngIndex = 0 To m_SegArray.Count - 1
1865:     Set pSegAbovePair = m_SegArray.Element(lngIndex)
1866:     Set pPolyline = pSegAbovePair.Element(0)
1867:     If pPolyline.SpatialReference Is Nothing Then Set pPolyline.SpatialReference = m_SpRef
1868:     If (pEnvelope Is Nothing) Then
1869:       Set pEnvelope = pPolyline.Envelope
1870:     Else
1871:       pEnvelope.Union pPolyline.Envelope
1872:     End If
1873:     booIsAbove = pSegAbovePair.Element(1)
1874:     If booIsAbove Then
1875:       strName = "Above Threshold (>" & CStr(m_dblThreshold) & ")"
1876:     Else
1877:       strName = "Below Threshold (<=" & CStr(m_dblThreshold) & ")"
1878:     End If
```

```
1880:     With pSegsBuf
1881:       Set .Shape = pPolyline
1882:       .Value(lngSegsID) = lngIndex + 1
1883:       .Value(lngSegsName) = strName
1884:       .Value(lngSegsLength) = pPolyline.length
1885:     End With
1886:     pSegsCur.InsertFeature pSegsBuf
1887:   Next lngIndex

1889:   pSegsCur.Flush
  Dim pFeatureLayer As IFeatureLayer
1891:   Set pFeatureLayer = New FeatureLayer
1892:   Set pFeatureLayer.FeatureClass = pSegsFClass
1893:   pFeatureLayer.Name = aml_func_mod.ClipExtension(strJustFilename)
1894:   pMap.AddLayer pFeatureLayer

1896:   If (Not pEnvelope Is Nothing) Then
1897:     pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, pEnvelope
1898:   End If

  Dim pDocDirty As IDocumentDirty
1901:   Set pDocDirty = m_pMxDoc
1902:   pDocDirty.SetDirty

  ' FOR REPORT
1905:   strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\tx90\tx360\tx450\tx720\f0\fs16 " & _
      "The shapefile \b " & strJustFilename & "\b0  has been generated and added to your view.  Individual segments " & _
      "are attributed according to whether they are above or below the threshold value of " & CStr(m_dblThreshold) & _
      " " & m_UnitName & ".\par" & vbCrLf & "\par" & vbCrLf & _
      "\b Shapefile saved to:\b0 \i  " & Replace(strFName, "\", "\\") & "\i0\par" & vbCrLf & vbCrLf & "}"

    ' SHOW REPORT
  Dim frmReportForm As New Linkages.frmReport_modal
1915:   frmReportForm.txtReport.TextRTF = strReport
1916:   frmReportForm.Caption = "Operation Successful:"
1917:   frmReportForm.Show vbModal
  ' MsgBox strReport

  Exit Sub
ErrorHandler:
  HandleError True, "Command1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
  On Error GoTo ErrorHandler

1928:   X = X / 15
1929:   Y = Y / 15
'  Debug.Print X & ", " & Y

1932:   If (X >= (GraphLeft + 58)) And (X <= (GraphRight - 15)) And _
    (Y >= (GraphTop + 17)) And (Y <= (GraphBottom - 38)) Then

    Dim dblTrueY As Double
1936:     dblTrueY = (GraphBottom - 38) - Y   ' = # PIXELS ABOVE BOTTOM OF GRAPH
1937:     dblTrueY = dblTrueY / (GraphBottom - 38 - (GraphTop + 17))   ' = PROPORTION UP
1938:     dblTrueY = dblTrueY * m_dblMaxGraphY
1939:     m_booIsScrolling = False
1940:     m_booCreateShapes = True
1941:     If m_dblMaxGraphY < 100 Then
1942:       txtDistance.Text = Round(dblTrueY, 5)
1943:     Else
1944:       txtDistance.Text = Int(dblTrueY)
1945:     End If
'    Debug.Print Int(dblTrueY)
   ' ConvertValToGraphY = (GraphBottom - 38) - ((dblDistance / m_dblMaxGraphY) * m_dblYRange)     ' USE m_dblMaxGraphY

1949:   End If


  Exit Sub
ErrorHandler:
  HandleError True, "Form_MouseDown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler

1960:   Set m_pApp = Nothing
1961:   Set m_pAboveThreshold = Nothing
1962:   Set m_pBelowThreshold = Nothing
1963:   Set m_SpRef = Nothing
1964:   Set m_Frame = Nothing
1965:   Set m_pXLabels = Nothing
1966:   Set m_pYLabels = Nothing
1967:   Set m_pDataArray = Nothing
1968:   Set m_pMxDoc = Nothing
1969:   Set m_SegArray = Nothing

  Exit Sub
```

```
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub slidThreshold_Click()
  On Error GoTo ErrorHandler

'  Debug.Print "Clicking..."
1980:   If m_dblMaxGraphY < 100 Then
1981:     m_dblThreshold = ((-slidThreshold.Value + 1) / 1000) * m_dblMaxGraphY
1982:   Else
1983:     m_dblThreshold = -slidThreshold.Value + 1
1984:   End If
'  m_dblThreshold = -(slidThreshold.Value - 1) * m_dblSliderAdjustment
1986:   txtDistance.Text = CStr(m_dblThreshold)
1987:   m_booIsScrolling = False
1988:   m_booCreateShapes = True
1989:   FillGraph
'  Debug.Print "Click:  " & -slidThreshold.Value + 1

  Exit Sub
ErrorHandler:
  HandleError True, "slidThreshold_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub slidThreshold_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  On Error GoTo ErrorHandler


'  m_dblThreshold = -(slidThreshold.Value - 1) * m_dblSliderAdjustment

2003:   If m_dblMaxGraphY < 100 Then
2004:     m_dblThreshold = ((-slidThreshold.Value + 1) / 1000) * m_dblMaxGraphY
2005:   Else
2006:     m_dblThreshold = -slidThreshold.Value + 1
2007:   End If
2008:   txtDistance.Text = CStr(m_dblThreshold)
2009:   m_booIsScrolling = False
2010:   m_booCreateShapes = True
2011:   FillGraph
'  Debug.Print "MouseUp:  " & -slidThreshold.Value + 1


  Exit Sub
ErrorHandler:
```

```
  HandleError True, "slidThreshold_MouseUp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub slidThreshold_Scroll()
  On Error GoTo ErrorHandler

'  Debug.Print "Scrolling..."
2024:   m_booIsScrolling = False
2025:   m_booCreateShapes = False
'  m_dblThreshold = -(slidThreshold.Value - 1) * m_dblSliderAdjustment

2028:   If m_dblMaxGraphY < 100 Then
2029:     m_dblThreshold = ((-slidThreshold.Value + 1) / 1000) * m_dblMaxGraphY
2030:   Else
2031:     m_dblThreshold = -slidThreshold.Value + 1
2032:   End If
2033:   txtDistance.Text = CStr(m_dblThreshold)


  Exit Sub
ErrorHandler:
  HandleError True, "slidThreshold_Scroll " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub txtDistance_Change()
  On Error GoTo ErrorHandler

'  Debug.Print "Changing Textbox...    m_booIsScrolling = " & CStr(m_booIsScrolling) & ", " & _
      "m_booCreateShapes = " & CStr(m_booCreateShapes)

'  Dim dblThreshDist As Double
  Dim dblDist As Double
2049:   If Not IsNumeric(txtDistance.Text) Then
2050:     dblDist = 0
2051:   Else
2052:     dblDist = CDbl(txtDistance.Text)
2053:   End If
'  If dblDist > m_dblMaxY Then      ' USE m_dblMaxGraphY
2055:   If dblDist > m_dblMaxGraphY Then      ' USE m_dblMaxGraphY
2056:     dblDist = m_dblMaxGraphY
2057:   ElseIf dblDist < 0 Then
2058:     dblDist = 0
2059:   End If

'  dblThreshDist = -(dblDist + 1) / m_dblSliderAdjustment
```

```
'  dblThreshDist = -(dblDist + 1)
2063:   m_dblThreshold = dblDist

2065:   If m_dblMaxGraphY < 100 Then
2066:     slidThreshold.Value = -Int((m_dblThreshold / m_dblMaxGraphY) * 1000) + 1
2067:   Else
2068:     slidThreshold.Value = -Int(m_dblThreshold) + 1
2069:   End If

'  slidThreshold.Value = dblThreshDist
2072:   FillGraph
'  Debug.Print slidThreshold.Value & ", " & dblThreshDist
  Exit Sub
ErrorHandler:
  HandleError True, "txtDistance_Change " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub txtDistance_KeyPress(KeyAscii As Integer)
  On Error GoTo ErrorHandler

2082:   Call Linkages.MyGeneralOperations.CheckNumericRealPositive(KeyAscii, txtDistance)
2083:   m_booIsScrolling = False
2084:   m_booCreateShapes = True
'  Debug.Print "calling KeyPress..."

  Exit Sub
ErrorHandler:
  HandleError True, "txtDistance_KeyPress " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub MakeEMF()
  On Error GoTo ErrorHandler

  Dim strFile As String
  Dim hdcEMF As Long
  Dim hemf As Long
  Dim rBounds As RECT
  Dim pWidth As Long
  Dim pHeight As Long

  Dim dblMultiplier As Double
2103:   dblMultiplier = 2

  Dim strDirPath As String
2106:   strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.GetMxDocPath(m_pApp))
```

```
2107:   strDirPath = aml_func_mod.ReturnDir(strDirPath)

  Dim boolWorkspaceExists As Boolean
2110:   boolWorkspaceExists = Not Dir$(strDirPath) = ""

2112:   If Not boolWorkspaceExists Then
2113:     strDirPath = aml_func_mod.GetFullFileString(aml_func_mod.TempPathLocation)
2114:   End If

2116:   If Right(strDirPath, 1) = "\" Then
2117:     strFile = strDirPath & "bottleneck_graph.emf"
2118:   Else
2119:     strFile = strDirPath & "\bottleneck_graph.emf"
2120:   End If
2121:   strFile = aml_func_mod.MakeUniqueFilename(strFile)

  Dim lngFontWeight As Long
  Dim lngFontHeight As Long
  Dim nFont As Long, oFont As Long
  Dim strText As String
  Dim textSize As SIZE
  Dim lngCounter As Long

2130:   rBounds.Left = 0
2131:   rBounds.Top = 0
2132:   rBounds.Right = (GraphRight - GraphLeft) * 27 * dblMultiplier
2133:   rBounds.Bottom = (GraphBottom - GraphTop) * 27 * dblMultiplier

  Dim emfGraphRight As Double
  Dim emfGraphLeft As Double
  Dim emfGraphTop As Double
  Dim emvGraphBottom As Double

'  strFile = App.Path & "\temp.emf"

2142:   hdcEMF = CreateEnhMetaFile(Me.hdc, strFile, rBounds, "")
2143:   SetMapMode hdcEMF, 1 ' MM_TEXT
2144:   SetBkMode hdcEMF, 1 ' transparent

  ' DRAW
  ' DIMENSION PENS AND BRUSHES
  Dim bluePen As Long
  Dim redPen As Long
  Dim nullPen As Long
  Dim blackPen As Long
  Dim blackDashPen As Long
  Dim magentaPen As Long
```

```
  Dim whiteBrush As Long

  ' PREPARE THRESHOLD STATS VARIABLES
2157:   m_pAboveThreshold.RemoveAll
2158:   m_pBelowThreshold.RemoveAll
2159:   m_dblLengthAbove = 0
2160:   m_dblLengthBelow = 0
  Dim dblLength As Double
  Dim dblTruePrevX As Double
  Dim dblTrueCurrentX As Double
  Dim dblTrueShortX As Double
  Dim dblRunningLengthAbove As Double
  Dim dblRunningLengthBelow As Double
2167:   dblRunningLengthAbove = 0
2168:   dblRunningLengthBelow = 0

  ' DRAW WHITE BOX
2171:   nullPen = CreatePen(PS_NULL, 0, vbRed)
2172:   DeleteObject SelectObject(hdc, nullPen)

2174:   whiteBrush = CreateSolidBrush(vbWhite)
2175:   DeleteObject SelectObject(hdc, whiteBrush)

'  RectangleX hdcEMF, GraphLeft, GraphTop, GraphRight, GraphBottom

  ' GET THRESHOLD VALUE IN GRAPH UNITS
  Dim dblGraphThreshold As Double
2181:   dblGraphThreshold = ConvertValToGraphY(m_dblThreshold)

  ' DRAW INSET EDGE
  Dim di As Long
  Dim rc As RECT
'    di = GetClientRect(hwnd, rc)

  ' DRAW X- AND Y-AXIS
2189:   blackPen = CreatePen(PS_SOLID, 1, vbBlack)
2190:   DeleteObject SelectObject(hdcEMF, blackPen)
  Dim lpPoint As POINTAPI
'  ' X-AXIS
'  MoveToEx hdcEMF, GraphLeft + 45, GraphBottom - 38, lpPoint
'  LineTo hdcEMF, GraphRight - 15, GraphBottom - 38
'  ' Y-AXIS
'  MoveToEx hdcEMF, GraphLeft + 58, GraphTop + 17, lpPoint
'  LineTo hdcEMF, GraphLeft + 58, GraphBottom - 30
'  Dim strReport As String

  ' FONT FOR NUMBERS
```

```
2201:   lngFontHeight = -((8 * dblMultiplier * GetDeviceCaps(hdcEMF, LOGPIXELSY)) / 72)
2202:   nFont = CreateFont(lngFontHeight, 0, 0, 0, 450, -1, 0, 0, 1, 7, 0, 0, 0, ByVal "Arial")
2203:   oFont = SelectObject(hdcEMF, nFont)

  ' X-AXIS
2206:   MoveToEx hdcEMF, (GraphLeft + 55) * dblMultiplier, (GraphBottom - 38) * dblMultiplier, lpPoint
2207:   LineTo hdcEMF, (GraphRight - 15) * dblMultiplier, (GraphBottom - 38) * dblMultiplier
2208:   strText = m_pYLabels.Element(0)
2209:   GetTextExtentPoint32 hdcEMF, strText, Len(strText), textSize
2210:   rc.Left = ((GraphLeft + 55 - 2) * dblMultiplier) - textSize.cx
2211:   rc.Top = (GraphBottom - 38) * dblMultiplier - (textSize.cy / 2)
2212:   rc.Right = rc.Left + textSize.cx
2213:   rc.Bottom = rc.Top + textSize.cy
2214:   DrawText hdcEMF, strText, Len(strText), rc, DT_NOCLIP + DT_RIGHT

  ' Y-AXIS
2217:   MoveToEx hdcEMF, (GraphLeft + 58) * dblMultiplier, (GraphTop + 17) * dblMultiplier, lpPoint
2218:   LineTo hdcEMF, (GraphLeft + 58) * dblMultiplier, (GraphBottom - 34) * dblMultiplier
2219:   strText = m_pXLabels.Element(0)
2220:   GetTextExtentPoint32 hdcEMF, strText, Len(strText), textSize
2221:   rc.Left = ((GraphLeft + 58) * dblMultiplier) - (textSize.cx / 2)
2222:   rc.Top = (GraphBottom - 32) * dblMultiplier
2223:   rc.Right = rc.Left + textSize.cx
2224:   rc.Bottom = rc.Top + textSize.cy
2225:   DrawText hdcEMF, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

'  strReport = strReport & _
    "X-Axis:  Y = " & CStr((GraphBottom - 38) * dblMultiplier) & ", X goes from " & _
        CStr((GraphLeft + 55) * dblMultiplier) & " to " & CStr((GraphRight - 15) * dblMultiplier) & "..." & vbCrLf & _
    "Y-Axis:  X = " & CStr((GraphLeft + 58) * dblMultiplier) & ", Y goes from " & _
        CStr((GraphTop + 17) * dblMultiplier) & " to " & CStr((GraphBottom - 30) * dblMultiplier) & "..." & vbCrLf

  ' MAKE DASHED HORIZONTAL LINES
2234:   lngCounter = 1
  Dim dblInc As Double
2236:   dblInc = m_dblIncrement
  Dim dblVertVal As Double
2238:   dblVertVal = ConvertValToGraphY(dblInc)
2239:   Do Until dblInc > m_dblMaxGraphY

    ' BLACK TICS
2242:     blackPen = CreatePen(PS_SOLID, 1, vbBlack)
2243:     DeleteObject SelectObject(hdcEMF, blackPen)
2244:     MoveToEx hdcEMF, (GraphLeft + 55) * dblMultiplier, Int(dblVertVal * dblMultiplier), lpPoint
2245:     LineTo hdcEMF, (GraphLeft + 59) * dblMultiplier, Int(dblVertVal * dblMultiplier)

    ' GRAY DASHES
```

```
2248:     blackDashPen = CreatePen(PS_DOT, 1, RGB(195, 195, 195))
2249:     DeleteObject SelectObject(hdcEMF, blackDashPen)
'    MoveToEx hdcEMF, (GraphLeft + 59) * dblMultiplier, dblVertVal * dblMultiplier, lpPoint
2251:     LineTo hdcEMF, (GraphRight - 15) * dblMultiplier, Int(dblVertVal * dblMultiplier)

    ' TEXT
2254:     strText = m_pYLabels.Element(lngCounter)
2255:     GetTextExtentPoint32 hdcEMF, strText, Len(strText), textSize
2256:     rc.Left = ((GraphLeft + 55 - 2) * dblMultiplier) - textSize.cx
2257:     rc.Top = (dblVertVal * dblMultiplier) - (textSize.cy / 2)
2258:     rc.Right = rc.Left + textSize.cx
2259:     rc.Bottom = rc.Top + textSize.cy
2260:     DrawText hdcEMF, strText, Len(strText), rc, DT_NOCLIP + DT_RIGHT

2262:     dblInc = dblInc + m_dblIncrement
2263:     dblVertVal = ConvertValToGraphY(dblInc)
2264:     lngCounter = lngCounter + 1

'    strReport = strReport & _
      "Horizontal Lines:  Y = " & CStr(Int(dblVertVal * dblMultiplier)) & ", X goes from " & _
          CStr((GraphLeft + 55) * dblMultiplier) & " to " & CStr((GraphRight - 15) * dblMultiplier) & "..." & vbCrLf
2269:   Loop

  ' MAKE DASHED VERTICAL LINES
2272:   lngCounter = 1
2273:   dblInc = m_dblXIncrement
  Dim dblHorizVal As Double
2275:   dblHorizVal = ConvertValToGraphX(dblInc)
2276:   Do Until dblInc > m_dblMaxX

    ' BLACK TICS
2279:     blackPen = CreatePen(PS_SOLID, 1, vbBlack)
2280:     DeleteObject SelectObject(hdcEMF, blackPen)
2281:     MoveToEx hdcEMF, Int((dblHorizVal) * dblMultiplier), (GraphBottom - 34) * dblMultiplier, lpPoint
2282:     LineTo hdcEMF, Int((dblHorizVal) * dblMultiplier), (GraphBottom - 38) * dblMultiplier

    ' GRAY DASHES
2285:     blackDashPen = CreatePen(PS_DOT, 1, RGB(195, 195, 195))
2286:     DeleteObject SelectObject(hdcEMF, blackDashPen)
'    MoveToEx hdcEMF, (dblHorizVal) * dblMultiplier, (GraphBottom - 38) * dblMultiplier, lpPoint
2288:     LineTo hdcEMF, Int((dblHorizVal) * dblMultiplier), (GraphTop + 17) * dblMultiplier

    ' TEXT
2291:     strText = m_pXLabels.Element(lngCounter)
2292:     GetTextExtentPoint32 hdcEMF, strText, Len(strText), textSize
2293:     rc.Left = (dblHorizVal * dblMultiplier) - (textSize.cx / 2)
2294:     rc.Top = (GraphBottom - 32) * dblMultiplier
```

```
2295:     rc.Right = rc.Left + textSize.cx
2296:     rc.Bottom = rc.Top + textSize.cy
2297:     DrawText hdcEMF, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

2299:     dblInc = dblInc + m_dblXIncrement
2300:     dblHorizVal = ConvertValToGraphX(dblInc)
2301:     lngCounter = lngCounter + 1

'    strReport = strReport & _
      "Vertical Lines:  X = " & CStr(Int((dblHorizVal) * dblMultiplier)) & ", Y goes from " & _
          CStr((GraphBottom - 34) * dblMultiplier) & " to " & CStr((GraphTop + 17) * dblMultiplier) & "..." & vbCrLf
2306:   Loop

  ' FONT FOR Y-AXIS LABEL
2309:   DeleteObject nFont
2310:   lngFontHeight = -((11 * dblMultiplier * GetDeviceCaps(hdcEMF, LOGPIXELSY)) / 72)
2311:   nFont = CreateFont(lngFontHeight, 0, 900, 900, 700, 0, 0, 0, 1, 7, 0, 0, 0, ByVal "Arial")
2312:   oFont = SelectObject(hdcEMF, nFont)
2313:   strText = m_strYName
2314:   GetTextExtentPoint32 hdcEMF, strText, Len(strText), textSize
'  rc.Left = ((GraphLeft + 3) * dblMultiplier) + textSize.cy
2316:   rc.Left = GraphLeft * dblMultiplier + (textSize.cx / 2)
2317:   rc.Top = (((GraphBottom - GraphTop) / 2) + GraphTop) * dblMultiplier + (textSize.cx / 2)
2318:   rc.Right = rc.Left + textSize.cy
2319:   rc.Bottom = rc.Top + textSize.cx ' (GraphBottom - GraphTop) / 2 + (textSize.cx / 2)
2320:   DrawText hdcEMF, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

  ' FONT FOR X-AXIS LABEL
2323:   DeleteObject nFont
2324:   lngFontHeight = -((11 * dblMultiplier * GetDeviceCaps(hdcEMF, LOGPIXELSY)) / 72)
2325:   nFont = CreateFont(lngFontHeight, 0, 0, 0, 700, 0, 0, 0, 1, 7, 0, 0, 0, ByVal "Arial")
2326:   oFont = SelectObject(hdcEMF, nFont)
2327:   strText = m_strXName
2328:   GetTextExtentPoint32 hdcEMF, strText, Len(strText), textSize
2329:   rc.Left = ((((GraphRight - GraphLeft) / 2) + GraphLeft) * dblMultiplier) - (textSize.cx / 2)
2330:   rc.Top = ((GraphBottom - 1) * dblMultiplier) - textSize.cy
2331:   rc.Right = rc.Left + textSize.cx ' ((GraphRight - GraphLeft) / 2) + GraphLeft + (textSize.cx / 2)
2332:   rc.Bottom = rc.Top + textSize.cy '  GraphBottom - 1
2333:   DrawText hdcEMF, strText, Len(strText), rc, DT_NOCLIP + DT_CENTER

2335:   DeleteObject nFont

  ' DRAW THRESHOLD LINE
2338:   magentaPen = CreatePen(PS_SOLID, 1, RGB(255, 0, 255))
2339:   DeleteObject SelectObject(hdcEMF, magentaPen)
'  MoveToEx hdcEMF, GraphLeft + 59, dblGraphThreshold, lpPoint
'  LineTo hdcEMF, GraphRight - 15, dblGraphThreshold
```

```
2342:   MoveToEx hdcEMF, (GraphLeft + 59) * dblMultiplier, (dblGraphThreshold) * dblMultiplier, lpPoint
2343:   LineTo hdcEMF, (GraphRight - 15) * dblMultiplier, (dblGraphThreshold) * dblMultiplier

  ' DRAW GRAPH LINE
  Dim dblTempX As Double
  Dim dblTempY As Double
  Dim dblPrevX As Double
  Dim dblPrevY As Double
  Dim lngIndex As Long
  Dim booCurrentThresh As Boolean
  Dim booPreviousThresh As Boolean ' false for below threshold, true for above threshold

2354:   dblTempY = m_dblGraphVals(0, 1)
  ' Y-AXIS REVERSED IN GRAPH SPACE!!!
2356:   booCurrentThresh = (dblTempY <= dblGraphThreshold)
2357:   dblPrevX = m_dblGraphVals(0, 0)
2358:   dblPrevY = dblTempY
2359:   dblTruePrevX = m_dblVals(0, 0)
  Dim dblProportion As Double
  Dim dblShortX As Double

2363:   BeginPath hdcEMF
'  MoveToEx hdcEMF, m_dblGraphVals(0, 0), m_dblGraphVals(0, 1), lpPoint
2365:   MoveToEx hdcEMF, (m_dblGraphVals(0, 0)) * dblMultiplier, (m_dblGraphVals(0, 1)) * dblMultiplier, lpPoint

2367:   If booCurrentThresh Then
2368:     bluePen = CreatePen(PS_SOLID, 1, vbBlue)
2369:     DeleteObject SelectObject(hdcEMF, bluePen)
2370:     booPreviousThresh = True
2371:   Else
2372:     redPen = CreatePen(PS_SOLID, 1, vbRed)
2373:     DeleteObject SelectObject(hdcEMF, redPen)
2374:     booPreviousThresh = False
2375:   End If

2377:   For lngIndex = 0 To UBound(m_dblGraphVals, 1)
2378:     dblTempX = m_dblGraphVals(lngIndex, 0)
2379:     dblTempY = m_dblGraphVals(lngIndex, 1)

2381:     booCurrentThresh = (dblTempY <= dblGraphThreshold)
2382:     If booCurrentThresh = booPreviousThresh Then    ' THEN HAS NOT CROSSED THRESHOLD
'      LineTo hdcEMF, dblTempX, dblTempY
2384:       LineTo hdcEMF, dblTempX * dblMultiplier, dblTempY * dblMultiplier

2386:     Else    ' HAS CROSSED THRESHOLD; NEED TO MAKE TWO LINES
2387:       dblProportion = (dblGraphThreshold - dblPrevY) / (dblTempY - dblPrevY)
2388:       dblShortX = dblPrevX + ((dblTempX - dblPrevX) * dblProportion)
```

```
'      LineTo hdcEMF, dblShortX, dblGraphThreshold
2390:       LineTo hdcEMF, dblShortX * dblMultiplier, dblGraphThreshold * dblMultiplier

      ' END CURRENT PATH AND START ANOTHER
2393:       EndPath hdcEMF
2394:       StrokePath hdcEMF
2395:       BeginPath hdcEMF

2397:       If booCurrentThresh Then ' THEN PREVIOUSLY WAS BELOW THRESHOLD, AND USING RED PEN.  NOW IS ABOVE THRESHOLD.
2398:         bluePen = CreatePen(PS_SOLID, 1, vbBlue)
2399:         DeleteObject SelectObject(hdcEMF, bluePen)
2400:       Else                      ' THEN PREVIOUSLY WAS ABOVE THRESHOLD, AND USING BLUE PEN.  NOW IS BELOW THRESHOLD.
2401:         redPen = CreatePen(PS_SOLID, 1, vbRed)
2402:         DeleteObject SelectObject(hdcEMF, redPen)
2403:       End If

'      LineTo hdcEMF, dblTempX, dblTempY
2406:       LineTo hdcEMF, dblTempX * dblMultiplier, dblTempY * dblMultiplier
2407:     End If

2409:     dblPrevY = dblTempY
2410:     dblPrevX = dblTempX
2411:     booPreviousThresh = booCurrentThresh

'    Debug.Print CStr(lngIndex) & "]  Original:  X = " & CStr(m_dblVals(lngIndex, 0)) & ",   Y = " & CStr(m_dblVals(lngIndex, 1))
'    Debug.Print "        Converted:  X = " & CStr(m_dblGraphVals(lngIndex, 0)) & ",   Y = " & CStr(m_dblGraphVals(lngIndex, 1))
2415:   Next lngIndex

2417:   EndPath hdcEMF
2418:   StrokePath hdcEMF

  'close metafile
2421:   hemf = CloseEnhMetaFile(hdcEMF)
2422:   DeleteDC hdcEMF
2423:   DeleteEnhMetaFile hemf
2424:   DeleteObject nFont
  'load it to printer
'  Printer.PaintPicture LoadPicture(strFile), 0, 0
'  Printer.EndDoc


'  Exit Sub

  Dim pMxDoc As IMxDocument
2433:   Set pMxDoc = m_pApp.Document
  Dim pGraphicContainer As IGraphicsContainer
2435:   Set pGraphicContainer = pMxDoc.PageLayout
```

```
  Dim pSelGraphContainer As IGraphicsContainerSelect
2437:   Set pSelGraphContainer = pGraphicContainer
2438:   pSelGraphContainer.UnselectAllElements

  Dim pPictureElement As IPictureElement
2441:   Set pPictureElement = New EmfPictureElement
  'open the emf file
2443:   pPictureElement.ImportPictureFromFile strFile
  Dim pElement As IElement
2445:   Set pElement = pPictureElement
  Dim pEnvelope As IEnvelope
2447:   Set pEnvelope = New Envelope
  'Set the position of the element (page coordinates)
2449:   pEnvelope.PutCoords 0, 0, 5.781, 1.854
2450:   pElement.Geometry = pEnvelope
2451:   pGraphicContainer.AddElement pElement, 0
'  pPictureElement.SavePictureInDocument = True
2453:   pSelGraphContainer.SelectElement pElement

  Dim strReport As String
2456:   strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\tx90\tx360\tx450\tx720\f0\fs16 " & _
      "The Bottleneck Chart has been converted to an Enhanced Windows Metafile (*.emf) and added to your " & _
      "layout view.  This graphic file is a standard Windows format and can be inserted into many " & _
      "other document types, such as Microsoft Word documents.\par" & vbCrLf & "\par" & vbCrLf & _
      "File located at " & Replace(strFile, "\", "\\") & "\par" & vbCrLf & "\par" & vbCrLf & _
      "\b Note:\b0   This graphic is not automatically saved into the Map document.  " & _
      "ArcMap only stores a reference to the graphic file on the hard drive, not the " & _
      "graphic itself.  If you delete the graphic file from the hard drive, then " & _
      "the graphic will be missing from your layout the next time you open your Map document.  " & _
      "If you wish to save the actual graphic in the map document file itself, then right-click on the graphic, " & _
      "select 'Properties', select the 'Picture' tab, and check the box for 'Save Picture as Part of Document'.\par" & _
      "}"

  Dim pDocDirty As IDocumentDirty
2472:   Set pDocDirty = m_pMxDoc
2473:   pDocDirty.SetDirty

  Dim pAV As IActiveView
2476:   Set pAV = pMxDoc.PageLayout
2477:   Set pMxDoc.ActiveView = pAV
2478:   pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing

  ' FOR REPORT
'  strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
```

```
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\tx90\tx360\tx450\tx720\f0\fs16 " & _
      "The shapefile \b " & strJustFilename & "\b0  has been generated and added to your view.  Individual points " & _
      "include X-coordinates, Y-coordinates and Corridor Width values at each location.\par" & vbCrLf & "\par" & vbCrLf & _
      "\b Shapefile saved to:\b0 \i  " & Replace(strFName, "\", "\\") & "\i0\par" & vbCrLf & vbCrLf & "}"

    ' SHOW REPORT
  Dim frmReportForm As New Linkages.frmReport_modal
2490:   frmReportForm.txtReport.TextRTF = strReport
2491:   frmReportForm.Caption = "Operation Successful:"
2492:   frmReportForm.Show vbModal

'  MsgBox strReport, , "Graph Export Successful:"

  Exit Sub
ErrorHandler:
  HandleError False, "MakeEMF " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub
```

Form 8: frmGraphicsShapefile.frm

```
VERSION 5.00
Begin VB.Form frmGraphicsShapefile
   BorderStyle     =   1  'Fixed Single
   Caption         =   " Create Shapefile:"
   ClientHeight    =   3705
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   8085
   Icon            =   "frmGraphicsShapefile.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   3705
   ScaleWidth      =   8085
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton cmdHelp
      Caption         =   "Help"
      Height          =   375
      Left            =   7140
      TabIndex        =   11
      Top             =   2085
      Width           =   885
   End
```

```
Begin VB.Frame frmNew
   BorderStyle     =   0  'None
   Height          =   1215
   Left            =   345
   TabIndex        =   15
   Top             =   645
   Width           =   2025
   Begin VB.OptionButton optPolygon
      Caption         =   "Polygon Shapefile"
      Height          =   240
      Left            =   180
      TabIndex        =   4
      Top             =   945
      Width           =   1935
   End
   Begin VB.OptionButton optPolyline
      Caption         =   "Polyline Shapefile"
      Height          =   240
      Left            =   180
      TabIndex        =   3
      Top             =   510
      Width           =   1620
   End
   Begin VB.OptionButton optPoint
      Caption         =   "Point Shapefile"
      Height          =   240
      Left            =   180
      TabIndex        =   2
      Top             =   75
      Width           =   2040
   End
End
Begin VB.ListBox lbxGraphics
   Height          =   1230
   Left            =   3240
   TabIndex        =   5
   Top             =   540
   Width           =   3660
End
Begin VB.CommandButton cmdCancel
   Caption         =   "Cancel"
   Height          =   375
   Left            =   7140
   TabIndex        =   10
   Top             =   1665
   Width           =   885
End
```

```
Begin VB.CommandButton cmdOK
   Caption         =   "OK"
   Height          =   375
   Left            =   7140
   TabIndex        =   12
   Top             =   2505
   Width           =   885
End
Begin VB.CommandButton cmdSpRef
   Caption         =   "Set Spatial Reference"
   Height          =   525
   Left            =   180
   TabIndex        =   7
   Top             =   2325
   Width           =   1350
End
Begin VB.TextBox txtOutput
   Height          =   330
   Left            =   165
   TabIndex        =   8
   Text            =   "Text1"
   Top             =   3270
   Width           =   6135
End
Begin VB.CommandButton cmdGetWorkspace
   Height          =   360
   Left            =   6390
   Picture         =   "frmGraphicsShapefile.frx":038A
   Style           =   1  'Graphical
   TabIndex        =   9
   ToolTipText     =   "Browse for Output Folder..."
   Top             =   3255
   Width           =   555
End
Begin VB.CheckBox chkSelected
   Caption         =   "Only convert selected graphics"
   Height          =   255
   Left            =   3705
   TabIndex        =   6
   Top             =   1830
   Width           =   2760
End
Begin VB.OptionButton optConvert
   Caption         =   "Convert graphics to shapefile"
   Height          =   255
   Left            =   3030
   TabIndex        =   1
```

```
   Top             =   135
   Width           =   3015
End
Begin VB.OptionButton optNew
   Caption         =   "Create new shapefile"
   Height          =   255
   Left            =   225
   TabIndex        =   0
   Top             =   135
   Width           =   1935
End
Begin VB.Image Image3
   Height          =   390
   Left            =   60
   Picture         =   "frmGraphicsShapefile.frx":0400
   Top             =   60
   Width           =   6975
End
Begin VB.Image Image2
   BorderStyle     =   1  'Fixed Single
   Height          =   720
   Left            =   30
   Top             =   2970
   Width           =   7020
End
Begin VB.Image Image1
   BorderStyle     =   1  'Fixed Single
   Height          =   765
   Left            =   30
   Top             =   2190
   Width           =   7020
End
Begin VB.Image imgBack1
   BorderStyle     =   1  'Fixed Single
   Height          =   2145
   Left            =   30
   Top             =   30
   Width           =   7020
End
Begin VB.Label lblOutputName
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Output Shapefile:"
   Height          =   195
   Left            =   150
   TabIndex        =   14
   Top             =   3060
```

```
      Width           =   1230
   End
   Begin VB.Image imgCorrIcon
      Height          =   855
      Left            =   7110
      Picture         =   "frmGraphicsShapefile.frx":920C
      Top             =   30
      Width           =   945
   End
   Begin VB.Label lblSpRef
      AutoSize        =   -1  'True
      Caption         =   "Shapefile Spatial Reference ="
      Height          =   195
      Left            =   1650
      TabIndex        =   13
      Top             =   2490
      Width           =   2115
   End
End
Attribute VB_Name = "frmGraphicsShapefile"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private m_MxDoc As esriArcMapUI.IMxDocument
Private m_pApp As IApplication
Private m_SpRef As ISpatialReference
Private m_ExtensionConfig As IExtensionConfig

Private m_AllPointGraphics As esriSystem.IVariantArray
Private m_SelPointGraphics As esriSystem.IVariantArray
Private m_AllPolylineGraphics As esriSystem.IVariantArray
Private m_SelPolylineGraphics As esriSystem.IVariantArray
Private m_AllPolygonGraphics As esriSystem.IVariantArray
Private m_SelPolygonGraphics As esriSystem.IVariantArray
Private m_AllGraphicsList As esriSystem.IStringArray
Private m_SelGraphicslist As esriSystem.IStringArray

Private m_AllPointGraphicsNames As esriSystem.IStringArray
Private m_SelPointGraphicsNames As esriSystem.IStringArray
Private m_AllPolylineGraphicsNames As esriSystem.IStringArray
Private m_SelPolylineGraphicsNames As esriSystem.IStringArray
Private m_AllPolygonGraphicsNames As esriSystem.IStringArray
Private m_SelPolygonGraphicsNames As esriSystem.IStringArray
```

```
Private m_booAllPointHasNames As Boolean
Private m_booSelPointHasNames As Boolean
Private m_booAllPolylineHasNames As Boolean
Private m_booSelPolylineHasNames As Boolean
Private m_booAllPolygonHasNames As Boolean
Private m_booSelPolygonHasNames As Boolean

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmGraphicsShapefile.frm"


Public Property Set ArcApplication(ByVal theApplication As IApplication)
  On Error GoTo ErrorHandler

37:   Set m_pApp = theApplication
38:   Set m_MxDoc = m_pApp.Document
39:   Call FillGraphicsArrays

  Exit Property
ErrorHandler:
  HandleError True, "ArcApplication " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property


Private Sub chkSelected_Click()
  On Error GoTo ErrorHandler

50:   Call FillGraphicsListBox

  Exit Sub
ErrorHandler:
  HandleError True, "chkSelected_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdGetWorkspace_Click()
  On Error GoTo ErrorHandler

  Dim pGxDialog As IGxDialog
61:   Set pGxDialog = New GxDialog

  Dim pGxDialogFilter As IGxObjectFilter
64:   Set pGxDialogFilter = New GxFilterShapefiles

  Dim pGxObject As IGxObject
  Dim pGxSelection As IEnumGxObject
```

```
  ' SET DEFAULT NAME
  Dim strDefault As String
  'strDefault = Linkages.aml_func_mod.MakeUniqueFilename(strDefault & "NewShape.shp")
72:   strDefault = Linkages.aml_func_mod.MakeUniqueFilename(txtOutput.Text)
'  MsgBox "strDefault = " & vbCrLf & strDefault
74:   With pGxDialog
75:     .AllowMultiSelect = False
76:     .StartingLocation = strDefault
77:     .Title = "Please enter the name of your new shapefile:"
78:     Set .ObjectFilter = pGxDialogFilter
79:     .Name = Linkages.aml_func_mod.ReturnFilename(strDefault)
80:   End With

82:   If (pGxDialog.DoModalSave(Me.hWnd) = True) Then
83:     txtOutput.Text = pGxDialog.FinalLocation.FullName & "\" & pGxDialog.Name
84:   End If

86:   Call CheckEnable

  Exit Sub
ErrorHandler:
  HandleError True, "cmdGetWorkspace_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4

End Sub

Private Sub cmdHelp_Click()

  Dim strPath As String
97:   strPath = App.Path & "\help"

99:   Call Linkages.MyGeneralOperations.OpenDoc("Shapefile_Subdocument.pdf", strPath)

End Sub

Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler

  Dim strFullName As String
107:   strFullName = txtOutput.Text
  Dim strDir As String
109:   strDir = Linkages.aml_func_mod.ReturnDir(strFullName)
  Dim strfilename As String
111:   strfilename = Linkages.aml_func_mod.ReturnFilename(strFullName)

  Dim pSpRef As ISpatialReference
114:   Set pSpRef = m_SpRef
```

```
  Dim pOrigSpRef As ISpatialReference
  Dim pGeoSpRef As IGeographicCoordinateSystem
  Dim pPrjSpRef As IProjectedCoordinateSystem

  Dim pFeatureClass As IFeatureClass
  Dim pFeatureBuf As IFeatureBuffer
  Dim pFeatureCursor As IFeatureCursor

  ' POLYGON SHAPEFILES WILL GET AREA FIELD
  ' POLYLINE SHAPEFILES WILL GET LENGTH FIELD
  ' POINT SHAPEFILES WILL GET X AND Y FIELD

  Dim pAreaField As IField
  Dim pAreaFieldEdit As IFieldEdit
130:   Set pAreaField = New Field
131:   Set pAreaFieldEdit = pAreaField
132:   With pAreaFieldEdit
133:     .Name = "Area"
134:     .Type = esriFieldTypeDouble
135:     .Precision = 16
136:     .Scale = 8
137:   End With

  Dim pLengthField As IField
  Dim pLengthFieldEdit As IFieldEdit
141:   Set pLengthField = New Field
142:   Set pLengthFieldEdit = pLengthField
143:   With pLengthFieldEdit
144:     .Name = "Length"
145:     .Type = esriFieldTypeDouble
146:     .Precision = 16
147:     .Scale = 8
148:   End With

  Dim pXField As IField
  Dim pXFieldEdit As IFieldEdit
152:   Set pXField = New Field
153:   Set pXFieldEdit = pXField
154:   With pXFieldEdit
155:     .Name = "X_Coord"
156:     .Type = esriFieldTypeDouble
157:     .Precision = 16
158:     .Scale = 8
159:   End With

  Dim pYField As IField
```

```
  Dim pYFieldEdit As IFieldEdit
163:   Set pYField = New Field
164:   Set pYFieldEdit = pYField
165:   With pYFieldEdit
166:     .Name = "Y_Coord"
167:     .Type = esriFieldTypeDouble
168:     .Precision = 16
169:     .Scale = 8
170:   End With


  Dim pNameField As IField
  Dim pNameFieldEdit As IFieldEdit
175:   Set pNameField = New Field
176:   Set pNameFieldEdit = pNameField
177:   With pNameFieldEdit
178:     .Name = "Name"
179:     .Type = esriFieldTypeString
180:     .Precision = 50
181:   End With

  Dim pGeometry As IGeometry
  Dim pPoint As IPoint
  Dim pPolyline As IPolyline
  Dim pPolygon As IPolygon
  Dim pSegCol As ISegmentCollection
  Dim pArea As IArea

  Dim lngIndex As Long
  Dim lngCounter As Long
192:   lngCounter = 0

  Dim strName As String

196:   If optNew.Value = True Then
197:     If optPoint.Value = True Then
198:       Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Point")
199:       If pFeatureClass Is Nothing Then
200:         MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
        Exit Sub
202:       End If
203:       pFeatureClass.AddField pXField
204:       pFeatureClass.AddField pYField
205:     ElseIf optPolyline.Value = True Then
206:       Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polyline")
207:       If pFeatureClass Is Nothing Then
208:         MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
```

```
       Exit Sub
210:      End If
211:      pFeatureClass.AddField pLengthField
212:    ElseIf optPolygon.Value = True Then
213:      Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polygon")
214:      If pFeatureClass Is Nothing Then
215:        MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
       Exit Sub
217:      End If
218:      pFeatureClass.AddField pAreaField
219:    End If
220:  ElseIf optConvert.Value = True Then
   Dim lngResponse As Long
222:    If chkSelected.Value = 0 Then                ' ALL GRAPHICS
     Select Case lbxGraphics.ListIndex
       Case 0                                     ' CONVERT POINTS TO POINT SHAPEFILE
225:         If m_AllPointGraphics.Count = 0 Then
226:           lngResponse = MsgBox("No graphic points in map!  No new shapefile will be made." & vbCrLf & _
                 "Do you wish to go back and " & _
                 "make a different selection?", vbOKCancel, "Unable to Make Point Shapefile:")
229:           If lngResponse <> 1 Then
230:             Unload Me
231:           End If
           Exit Sub
233:         End If
234:         Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Point")
235:         If pFeatureClass Is Nothing Then
236:           MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
           Exit Sub
238:         End If
239:         pFeatureClass.AddField pXField
240:         pFeatureClass.AddField pYField
241:         If m_booAllPointHasNames Then pFeatureClass.AddField pNameField

243:         Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
244:         Set pFeatureCursor = pFeatureClass.Insert(True)
245:         For lngIndex = 0 To m_AllPointGraphics.Count - 1
246:           lngCounter = lngCounter + 1
247:           Set pPoint = m_AllPointGraphics.Element(lngIndex)
248:           strName = m_AllPointGraphicsNames.Element(lngIndex)

         ' SET SPATIAL REFERENCE
251:           Set pOrigSpRef = pPoint.SpatialReference
252:           If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
254:             If TypeOf pSpRef Is IGeographicCoordinateSystem Then
255:               Set pPoint.SpatialReference = pSpRef
```

```
256:               Else
257:                 Set pPrjSpRef = pSpRef
258:                 Set pPoint.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
259:               End If
260:             ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
262:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
263:                 Set pPoint.SpatialReference = pSpRef
264:               Else
265:                 Set pPrjSpRef = pSpRef
266:                 Set pPoint.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
267:               End If
268:             End If
269:             If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPoint.SpatialReference) Then
270:               pPoint.Project pSpRef
271:             End If
           ' ADD DATA TO FEATURE BUFFER
273:             Set pFeatureBuf.Shape = pPoint
274:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
275:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("X_Coord")) = pPoint.X
276:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Y_Coord")) = pPoint.Y
277:             If m_booAllPointHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
278:             pFeatureCursor.InsertFeature pFeatureBuf
279:           Next lngIndex
280:           pFeatureCursor.Flush
       Case 1                                   ' CONVERT POLYLINE GRAPHICS TO POLYLINE SHAPEFILE
282:           If m_AllPolylineGraphics.Count = 0 Then
283:             lngResponse = MsgBox("No graphic polylines in map!  No new shapefile will be made." & vbCrLf & _
                   "Do you wish to go back and " & _
                   "make a different selection?", vbOKCancel, "Unable to Make Polyline Shapefile:")
286:             If lngResponse <> 1 Then
287:               Unload Me
288:             End If
           Exit Sub
290:           End If
291:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polyline")
292:           If pFeatureClass Is Nothing Then
293:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
           Exit Sub
295:           End If
296:           pFeatureClass.AddField pLengthField
297:           If m_booAllPolylineHasNames Then pFeatureClass.AddField pNameField

299:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
300:           Set pFeatureCursor = pFeatureClass.Insert(True)
301:           For lngIndex = 0 To m_AllPolylineGraphics.Count - 1
```

```
302:             lngCounter = lngCounter + 1
303:             Set pGeometry = m_AllPolylineGraphics.Element(lngIndex)
304:             strName = m_AllPolylineGraphicsNames.Element(lngIndex)

'            MsgBox "pGeometry.SpatialReference is Nothing? = " & CStr(pGeometry.SpatialReference Is Nothing)
307:             Set pSegCol = pGeometry
'            MsgBox "pSegCol.SpatialReference is Nothing? = " & CStr(pSegCol.SpatialReference Is Nothing)
309:             Set pPolyline = Linkages.MyGeometricOperations.CurveToPolyline(pGeometry, 200)
'            MsgBox "Length = " & pPolyline.length
'            MsgBox "pPolyline.SpatialReference is Nothing? = " & CStr(pPolyline.SpatialReference Is Nothing)
           ' SET SPATIAL REFERENCE
313:             Set pOrigSpRef = pPolyline.SpatialReference
'            MsgBox "Orig Sp Ref is nothing? " & CStr(pOrigSpRef Is Nothing)
315:             If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
'              MsgBox "pSpRef () = " & pSpRef.Name
318:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
319:                 Set pPolyline.SpatialReference = pSpRef
320:               Else
321:                 Set pPrjSpRef = pSpRef
322:                 Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
323:               End If
324:             ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
326:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
327:                 Set pPolyline.SpatialReference = pSpRef
328:               Else
329:                 Set pPrjSpRef = pSpRef
330:                 Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
331:               End If
332:             End If
333:             If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolyline.SpatialReference) Then
334:               pPolyline.Project pSpRef
335:             End If
           ' ADD DATA TO FEATURE BUFFER
'            MsgBox "Length Before = " & pPolyline.length
338:             Set pFeatureBuf.Shape = pPolyline
'            MsgBox "Length After = " & pPolyline.length
340:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
341:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Length")) = pPolyline.length
342:             If m_booAllPolylineHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
'            MsgBox "Adding Length = " & pPolyline.length & " to Field #" & pFeatureBuf.Fields.FindField("Length")
344:             pFeatureCursor.InsertFeature pFeatureBuf
345:           Next lngIndex
346:           pFeatureCursor.Flush
       Case 2                                   ' CONVERT POLYLINE + POLYGON GRAPHICS TO POLYLINE SHAPEFILE
```

```
348:           If m_AllPolylineGraphics.Count = 0 And m_AllPolygonGraphics.Count = 0 Then
349:             lngResponse = MsgBox("No graphic polylines or polygons in map!  No new shapefile will be made." & vbCrLf & _
                   "Do you wish to go back and " & _
                   "make a different selection?", vbOKCancel, "Unable to Make Polyline Shapefile:")
352:             If lngResponse <> 1 Then
353:               Unload Me
354:             End If
          Exit Sub
356:           End If
357:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polyline")
358:           If pFeatureClass Is Nothing Then
359:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
          Exit Sub
361:           End If
362:           pFeatureClass.AddField pLengthField
363:           If m_booAllPolylineHasNames Or m_booAllPolygonHasNames Then pFeatureClass.AddField pNameField

365:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
366:           Set pFeatureCursor = pFeatureClass.Insert(True)
367:           If m_AllPolylineGraphics.Count > 0 Then
368:             For lngIndex = 0 To m_AllPolylineGraphics.Count - 1
369:               lngCounter = lngCounter + 1
370:               Set pGeometry = m_AllPolylineGraphics.Element(lngIndex)
371:               strName = m_AllPolylineGraphicsNames.Element(lngIndex)

373:               Set pSegCol = pGeometry
374:               Set pPolyline = Linkages.MyGeometricOperations.CurveToPolyline(pGeometry, 200)
            ' SET SPATIAL REFERENCE
376:               Set pOrigSpRef = pPolyline.SpatialReference
377:               If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
379:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
380:                   Set pPolyline.SpatialReference = pSpRef
381:                 Else
382:                   Set pPrjSpRef = pSpRef
383:                   Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
384:                 End If
385:               ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
387:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
388:                   Set pPolyline.SpatialReference = pSpRef
389:                 Else
390:                   Set pPrjSpRef = pSpRef
391:                   Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
392:                 End If
393:               End If
```

```
394:                If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolyline.SpatialReference) Then
395:                  pPolyline.Project pSpRef
396:                End If
             ' ADD DATA TO FEATURE BUFFER
398:                Set pFeatureBuf.Shape = pPolyline
399:                pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
400:                pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Length")) = pPolyline.length
401:                If m_booAllPolylineHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
402:                pFeatureCursor.InsertFeature pFeatureBuf
403:              Next lngIndex
404:            End If
405:            If m_AllPolygonGraphics.Count > 0 Then
406:              For lngIndex = 0 To m_AllPolygonGraphics.Count - 1
407:                lngCounter = lngCounter + 1
408:                Set pGeometry = m_AllPolygonGraphics.Element(lngIndex)
409:                strName = m_AllPolygonGraphicsNames.Element(lngIndex)

411:                Set pSegCol = pGeometry
412:                Set pPolyline = Linkages.MyGeometricOperations.CurveToPolyline(pGeometry, 200)
             ' SET SPATIAL REFERENCE
414:                Set pOrigSpRef = pPolyline.SpatialReference
415:                If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                   ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
417:                  If TypeOf pSpRef Is IGeographicCoordinateSystem Then
418:                    Set pPolyline.SpatialReference = pSpRef
419:                  Else
420:                    Set pPrjSpRef = pSpRef
421:                    Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
422:                  End If
423:                ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                 ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
425:                  If TypeOf pSpRef Is IGeographicCoordinateSystem Then
426:                    Set pPolyline.SpatialReference = pSpRef
427:                  Else
428:                    Set pPrjSpRef = pSpRef
429:                    Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
430:                  End If
431:                End If
432:                If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolyline.SpatialReference) Then
433:                  pPolyline.Project pSpRef
434:                End If
             ' ADD DATA TO FEATURE BUFFER
436:                Set pFeatureBuf.Shape = pPolyline
437:                pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
438:                pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Length")) = pPolyline.length
439:                If m_booAllPolygonHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
```

```
440:               pFeatureCursor.InsertFeature pFeatureBuf
441:             Next lngIndex
442:           End If
443:           pFeatureCursor.Flush
       Case 3                                    ' CONVERT POLYGON GRAPHICS TO POLYGON SHAPEFILE
445:           If m_AllPolygonGraphics.Count = 0 Then
446:             lngResponse = MsgBox("No graphic polygons in map!  No new shapefile will be made." & vbCrLf & _
                 "Do you wish to go back and " & _
                 "make a different selection?", vbOKCancel, "Unable to Make Polygon Shapefile:")
449:             If lngResponse <> 1 Then
450:               Unload Me
451:             End If
           Exit Sub
453:           End If
454:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polygon")
455:           If pFeatureClass Is Nothing Then
456:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
           Exit Sub
458:           End If
459:           pFeatureClass.AddField pAreaField
460:           If m_booAllPolygonHasNames Then pFeatureClass.AddField pNameField

462:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
463:           Set pFeatureCursor = pFeatureClass.Insert(True)
464:           For lngIndex = 0 To m_AllPolygonGraphics.Count - 1
465:             lngCounter = lngCounter + 1
466:             Set pGeometry = m_AllPolygonGraphics.Element(lngIndex)
467:             strName = m_AllPolygonGraphicsNames.Element(lngIndex)

469:             Set pSegCol = pGeometry
470:             Set pPolygon = Linkages.MyGeometricOperations.CurveToPolygon(pGeometry, 200)
         ' SET SPATIAL REFERENCE
472:             Set pOrigSpRef = pPolygon.SpatialReference
473:             If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                   ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
475:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
476:                 Set pPolygon.SpatialReference = pSpRef
477:               Else
478:                 Set pPrjSpRef = pSpRef
479:                 Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
480:               End If
481:             ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                   ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
483:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
484:                 Set pPolygon.SpatialReference = pSpRef
485:               Else
```

```
                Set pPrjSpRef = pSpRef
                Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
              End If
            End If
            If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolygon.SpatialReference) Then
              pPolygon.Project pSpRef
            End If
        ' ADD DATA TO FEATURE BUFFER
            Set pFeatureBuf.Shape = pPolygon
            Set pArea = pPolygon
            pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
            pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Area")) = pArea.Area
            If m_booAllPolygonHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
            pFeatureCursor.InsertFeature pFeatureBuf
          Next lngIndex
          pFeatureCursor.Flush
      Case 4                                    ' CONVERT POLYLINE + POLYGON GRAPHICS TO POLYGON SHAPEFILE
          If m_AllPolylineGraphics.Count = 0 And m_AllPolygonGraphics.Count = 0 Then
            lngResponse = MsgBox("No graphic polylines or polygons in map!  No new shapefile will be made." & vbCrLf & _
              "Do you wish to go back and " & _
              "make a different selection?", vbOKCancel, "Unable to Make Polygon Shapefile:")
            If lngResponse <> 1 Then
              Unload Me
            End If
        Exit Sub
          End If
          Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polygon")
          If pFeatureClass Is Nothing Then
            MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
        Exit Sub
          End If
          pFeatureClass.AddField pAreaField
          If m_booAllPolylineHasNames Or m_booAllPolygonHasNames Then pFeatureClass.AddField pNameField

          Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
          Set pFeatureCursor = pFeatureClass.Insert(True)
          If m_AllPolylineGraphics.Count > 0 Then
            For lngIndex = 0 To m_AllPolylineGraphics.Count - 1
              lngCounter = lngCounter + 1
              Set pGeometry = m_AllPolylineGraphics.Element(lngIndex)
              strName = m_AllPolylineGraphicsNames.Element(lngIndex)

              Set pSegCol = pGeometry
              Set pPolygon = Linkages.MyGeometricOperations.CurveToPolygon(pGeometry, 200)
          ' SET SPATIAL REFERENCE
              Set pOrigSpRef = pPolygon.SpatialReference
              If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
```

```
                                                ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
534:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
535:                   Set pPolygon.SpatialReference = pSpRef
536:                 Else
537:                   Set pPrjSpRef = pSpRef
538:                   Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
539:                 End If
540:               ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
542:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
543:                   Set pPolygon.SpatialReference = pSpRef
544:                 Else
545:                   Set pPrjSpRef = pSpRef
546:                   Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
547:                 End If
548:               End If
549:               If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolygon.SpatialReference) Then
550:                 pPolygon.Project pSpRef
551:               End If
             ' ADD DATA TO FEATURE BUFFER
553:               Set pFeatureBuf.Shape = pPolygon
554:               Set pArea = pPolygon
555:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
556:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Area")) = pArea.Area
557:               If m_booAllPolylineHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
558:               pFeatureCursor.InsertFeature pFeatureBuf
559:             Next lngIndex
560:           End If
561:           If m_AllPolygonGraphics.Count > 0 Then
562:             For lngIndex = 0 To m_AllPolygonGraphics.Count - 1
563:               lngCounter = lngCounter + 1
564:               Set pGeometry = m_AllPolygonGraphics.Element(lngIndex)
565:               strName = m_AllPolygonGraphicsNames.Element(lngIndex)

567:               Set pSegCol = pGeometry
568:               Set pPolygon = Linkages.MyGeometricOperations.CurveToPolygon(pGeometry, 200)
             ' SET SPATIAL REFERENCE
570:               Set pOrigSpRef = pPolygon.SpatialReference
571:               If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
573:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
574:                   Set pPolygon.SpatialReference = pSpRef
575:                 Else
576:                   Set pPrjSpRef = pSpRef
577:                   Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
578:                 End If
```

```
579:               ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                 ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
581:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
582:                   Set pPolygon.SpatialReference = pSpRef
583:                 Else
584:                   Set pPrjSpRef = pSpRef
585:                   Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
586:                 End If
587:               End If
588:               If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolygon.SpatialReference) Then
589:                 pPolygon.Project pSpRef
590:               End If
             ' ADD DATA TO FEATURE BUFFER
592:               Set pFeatureBuf.Shape = pPolygon
593:               Set pArea = pPolygon
594:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
595:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Area")) = pArea.Area
596:               If m_booAllPolygonHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
597:               pFeatureCursor.InsertFeature pFeatureBuf
598:             Next lngIndex
599:           End If
600:           pFeatureCursor.Flush
601:       End Select
602:     Else                                          ' ONLY SELECTED GRAPHICS
      Select Case lbxGraphics.ListIndex
        Case 0                                     ' CONVERT POINTS TO POINT SHAPEFILE
605:           If m_SelPointGraphics.Count = 0 Then
606:             lngResponse = MsgBox("No selected graphic points in map!  No new shapefile will be made." & vbCrLf & _
                   "Do you wish to go back and " & _
                   "make a different selection?", vbOKCancel, "Unable to Make Point Shapefile:")
609:             If lngResponse <> 1 Then
610:               Unload Me
611:             End If
           Exit Sub
613:           End If
614:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Point")
615:           If pFeatureClass Is Nothing Then
616:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
           Exit Sub
618:           End If
619:           pFeatureClass.AddField pXField
620:           pFeatureClass.AddField pYField
621:           If m_booSelPointHasNames Then pFeatureClass.AddField pNameField

623:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
624:           Set pFeatureCursor = pFeatureClass.Insert(True)
```

```
625:           For lngIndex = 0 To m_SelPointGraphics.Count - 1
626:             lngCounter = lngCounter + 1
627:             Set pPoint = m_SelPointGraphics.Element(lngIndex)
628:             strName = m_SelPointGraphicsNames.Element(lngIndex)

           ' SET SPATIAL REFERENCE
631:             Set pOrigSpRef = pPoint.SpatialReference
632:             If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                  ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
634:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
635:                 Set pPoint.SpatialReference = pSpRef
636:               Else
637:                 Set pPrjSpRef = pSpRef
638:                 Set pPoint.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
639:               End If
640:             ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                  ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
642:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
643:                 Set pPoint.SpatialReference = pSpRef
644:               Else
645:                 Set pPrjSpRef = pSpRef
646:                 Set pPoint.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
647:               End If
648:             End If
649:             If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPoint.SpatialReference) Then
650:               pPoint.Project pSpRef
651:             End If
           ' ADD DATA TO FEATURE BUFFER
653:             Set pFeatureBuf.Shape = pPoint
654:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
655:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("X_Coord")) = pPoint.X
656:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Y_Coord")) = pPoint.Y
657:             If m_booSelPointHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
658:             pFeatureCursor.InsertFeature pFeatureBuf
659:           Next lngIndex
660:           pFeatureCursor.Flush
       Case 1                                     ' CONVERT POLYLINE GRAPHICS TO POLYLINE SHAPEFILE
662:           If m_SelPolylineGraphics.Count = 0 Then
663:             lngResponse = MsgBox("No selected graphic polylines in map!  No new shapefile will be made." & vbCrLf & _
                   "Do you wish to go back and " & _
                   "make a different selection?", vbOKCancel, "Unable to Make Polyline Shapefile:")
666:             If lngResponse <> 1 Then
667:               Unload Me
668:             End If
           Exit Sub
670:           End If
```

```
671:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polyline")
672:           If pFeatureClass Is Nothing Then
673:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
          Exit Sub
675:           End If
676:           pFeatureClass.AddField pLengthField
677:           If m_booSelPolylineHasNames Then pFeatureClass.AddField pNameField

679:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
680:           Set pFeatureCursor = pFeatureClass.Insert(True)
681:           For lngIndex = 0 To m_SelPolylineGraphics.Count - 1
682:             lngCounter = lngCounter + 1
683:             Set pGeometry = m_SelPolylineGraphics.Element(lngIndex)
684:             strName = m_SelPolylineGraphicsNames.Element(lngIndex)

686:             Set pSegCol = pGeometry
687:             Set pPolyline = Linkages.MyGeometricOperations.CurveToPolyline(pGeometry, 200)
          ' SET SPATIAL REFERENCE
689:             Set pOrigSpRef = pPolyline.SpatialReference
690:             If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                 ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
692:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
693:                 Set pPolyline.SpatialReference = pSpRef
694:               Else
695:                 Set pPrjSpRef = pSpRef
696:                 Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
697:               End If
698:             ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                 ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
700:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
701:                 Set pPolyline.SpatialReference = pSpRef
702:               Else
703:                 Set pPrjSpRef = pSpRef
704:                 Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
705:               End If
706:             End If
707:             If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolyline.SpatialReference) Then
708:               pPolyline.Project pSpRef
709:             End If
          ' ADD DATA TO FEATURE BUFFER
711:             Set pFeatureBuf.Shape = pPolyline
712:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
713:             pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Length")) = pPolyline.length
714:             If m_booSelPolylineHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
715:             pFeatureCursor.InsertFeature pFeatureBuf
716:           Next lngIndex
```

```
717:           pFeatureCursor.Flush
       Case 2                                       ' CONVERT POLYLINE + POLYGON GRAPHICS TO POLYLINE SHAPEFILE
719:           If m_SelPolylineGraphics.Count = 0 And m_SelPolygonGraphics.Count = 0 Then
720:             lngResponse = MsgBox("No selected graphic polylines or polygons in map!  No new shapefile will be made." & vbCrLf & _
                   "Do you wish to go back and " & _
                   "make a different selection?", vbOKCancel, "Unable to Make Polyline Shapefile:")
723:             If lngResponse <> 1 Then
724:               Unload Me
725:             End If
           Exit Sub
727:           End If
728:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polyline")
729:           If pFeatureClass Is Nothing Then
730:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
           Exit Sub
732:           End If
733:           pFeatureClass.AddField pLengthField
734:           If m_booSelPolylineHasNames Or m_booSelPolygonHasNames Then pFeatureClass.AddField pNameField

736:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
737:           Set pFeatureCursor = pFeatureClass.Insert(True)
738:           If m_SelPolylineGraphics.Count > 0 Then
739:             For lngIndex = 0 To m_SelPolylineGraphics.Count - 1
740:               lngCounter = lngCounter + 1
741:               Set pGeometry = m_SelPolylineGraphics.Element(lngIndex)
742:               strName = m_SelPolylineGraphicsNames.Element(lngIndex)

744:               Set pSegCol = pGeometry
745:               Set pPolyline = Linkages.MyGeometricOperations.CurveToPolyline(pGeometry, 200)
         ' SET SPATIAL REFERENCE
747:               Set pOrigSpRef = pPolyline.SpatialReference
748:               If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
750:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
751:                   Set pPolyline.SpatialReference = pSpRef
752:                 Else
753:                   Set pPrjSpRef = pSpRef
754:                   Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
755:                 End If
756:               ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
758:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
759:                   Set pPolyline.SpatialReference = pSpRef
760:                 Else
761:                   Set pPrjSpRef = pSpRef
762:                   Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
```

```
763:                 End If
764:               End If
765:               If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolyline.SpatialReference) Then
766:                 pPolyline.Project pSpRef
767:               End If
             ' ADD DATA TO FEATURE BUFFER
769:               Set pFeatureBuf.Shape = pPolyline
770:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
771:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Length")) = pPolyline.length
772:               If m_booSelPolylineHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
773:               pFeatureCursor.InsertFeature pFeatureBuf
774:             Next lngIndex
775:           End If
776:           If m_SelPolygonGraphics.Count > 0 Then
777:             For lngIndex = 0 To m_SelPolygonGraphics.Count - 1
778:               lngCounter = lngCounter + 1
779:               Set pGeometry = m_SelPolygonGraphics.Element(lngIndex)
780:               strName = m_SelPolygonGraphicsNames.Element(lngIndex)

782:               Set pSegCol = pGeometry
783:               Set pPolyline = Linkages.MyGeometricOperations.CurveToPolyline(pGeometry, 200)
             ' SET SPATIAL REFERENCE
785:               Set pOrigSpRef = pPolyline.SpatialReference
786:               If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
788:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
789:                   Set pPolyline.SpatialReference = pSpRef
790:                 Else
791:                   Set pPrjSpRef = pSpRef
792:                   Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
793:                 End If
794:               ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                    ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
796:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
797:                   Set pPolyline.SpatialReference = pSpRef
798:                 Else
799:                   Set pPrjSpRef = pSpRef
800:                   Set pPolyline.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
801:                 End If
802:               End If
803:               If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolyline.SpatialReference) Then
804:                 pPolyline.Project pSpRef
805:               End If
             ' ADD DATA TO FEATURE BUFFER
807:               Set pFeatureBuf.Shape = pPolyline
808:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
```

```
809:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Length")) = pPolyline.length
810:               If m_booSelPolygonHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
811:               pFeatureCursor.InsertFeature pFeatureBuf
812:             Next lngIndex
813:           End If
814:           pFeatureCursor.Flush
       Case 3                                    ' CONVERT POLYGON GRAPHICS TO POLYGON SHAPEFILE
816:           If m_SelPolygonGraphics.Count = 0 Then
817:             lngResponse = MsgBox("No selected graphic polygons in map!  No new shapefile will be made." & vbCrLf & _
                   "Do you wish to go back and " & _
                   "make a different selection?", vbOKCancel, "Unable to Make Polygon Shapefile:")
820:             If lngResponse <> 1 Then
821:               Unload Me
822:             End If
           Exit Sub
824:           End If
825:           Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polygon")
826:           If pFeatureClass Is Nothing Then
827:             MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
           Exit Sub
829:           End If
830:           pFeatureClass.AddField pAreaField
831:           If m_booSelPolygonHasNames Then pFeatureClass.AddField pNameField

833:           Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
834:           Set pFeatureCursor = pFeatureClass.Insert(True)
835:           For lngIndex = 0 To m_SelPolygonGraphics.Count - 1
836:             lngCounter = lngCounter + 1
837:             Set pGeometry = m_SelPolygonGraphics.Element(lngIndex)
838:             strName = m_SelPolygonGraphicsNames.Element(lngIndex)

840:             Set pSegCol = pGeometry
841:             Set pPolygon = Linkages.MyGeometricOperations.CurveToPolygon(pGeometry, 200)
         ' SET SPATIAL REFERENCE
843:             Set pOrigSpRef = pPolygon.SpatialReference
844:             If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                   ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
846:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
847:                 Set pPolygon.SpatialReference = pSpRef
848:               Else
849:                 Set pPrjSpRef = pSpRef
850:                 Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
851:               End If
852:             ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                   ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
854:               If TypeOf pSpRef Is IGeographicCoordinateSystem Then
```

```
855:                Set pPolygon.SpatialReference = pSpRef
856:              Else
857:                Set pPrjSpRef = pSpRef
858:                Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
859:              End If
860:            End If
861:            If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolygon.SpatialReference) Then
862:              pPolygon.Project pSpRef
863:            End If
          ' ADD DATA TO FEATURE BUFFER
865:            Set pFeatureBuf.Shape = pPolygon
866:            Set pArea = pPolygon
867:            pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
868:            pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Area")) = pArea.Area
869:            If m_booSelPolygonHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
870:            pFeatureCursor.InsertFeature pFeatureBuf
871:          Next lngIndex
872:          pFeatureCursor.Flush
      Case 4                                    ' CONVERT POLYLINE + POLYGON GRAPHICS TO POLYGON SHAPEFILE
874:          If m_SelPolylineGraphics.Count = 0 And m_SelPolygonGraphics.Count = 0 Then
875:            lngResponse = MsgBox("No selected graphic polylines or polygons in map!  No new shapefile will be made." & vbCrLf & _
                  "Do you wish to go back and " & _
                  "make a different selection?", vbOKCancel, "Unable to Make Polygon Shapefile:")
878:            If lngResponse <> 1 Then
879:              Unload Me
880:            End If
          Exit Sub
882:          End If
883:          Set pFeatureClass = Linkages.aml_func_mod.CreateShapefile(strDir, strfilename, pSpRef, "Polygon")
884:          If pFeatureClass Is Nothing Then
885:            MsgBox "Failed to Create Shapefile:" & vbCrLf & "Bailing out of 'cmdOK_Click' function...", , "Function Failed:"
          Exit Sub
887:          End If
888:          pFeatureClass.AddField pAreaField
889:          If m_booSelPolylineHasNames Or m_booSelPolygonHasNames Then pFeatureClass.AddField pNameField

891:          Set pFeatureBuf = pFeatureClass.CreateFeatureBuffer
892:          Set pFeatureCursor = pFeatureClass.Insert(True)
893:          If m_SelPolylineGraphics.Count > 0 Then
894:            For lngIndex = 0 To m_SelPolylineGraphics.Count - 1
895:              lngCounter = lngCounter + 1
896:              Set pGeometry = m_SelPolylineGraphics.Element(lngIndex)
897:              strName = m_SelPolylineGraphicsNames.Element(lngIndex)

899:              Set pSegCol = pGeometry
900:              Set pPolygon = Linkages.MyGeometricOperations.CurveToPolygon(pGeometry, 200)
            ' SET SPATIAL REFERENCE
```

```
902:               Set pOrigSpRef = pPolygon.SpatialReference
903:               If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                      ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
905:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
906:                   Set pPolygon.SpatialReference = pSpRef
907:                 Else
908:                   Set pPrjSpRef = pSpRef
909:                   Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
910:                 End If
911:               ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                      ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
913:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
914:                   Set pPolygon.SpatialReference = pSpRef
915:                 Else
916:                   Set pPrjSpRef = pSpRef
917:                   Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
918:                 End If
919:               End If
920:               If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolygon.SpatialReference) Then
921:                 pPolygon.Project pSpRef
922:               End If
            ' ADD DATA TO FEATURE BUFFER
924:               Set pFeatureBuf.Shape = pPolygon
925:               Set pArea = pPolygon
926:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
927:               pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Area")) = pArea.Area
928:               If m_booSelPolylineHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
929:               pFeatureCursor.InsertFeature pFeatureBuf
930:             Next lngIndex
931:           End If
932:           If m_SelPolygonGraphics.Count > 0 Then
933:             For lngIndex = 0 To m_SelPolygonGraphics.Count - 1
934:               lngCounter = lngCounter + 1
935:               Set pGeometry = m_SelPolygonGraphics.Element(lngIndex)
936:               strName = m_SelPolygonGraphicsNames.Element(lngIndex)

938:               Set pSegCol = pGeometry
939:               Set pPolygon = Linkages.MyGeometricOperations.CurveToPolygon(pGeometry, 200)
            ' SET SPATIAL REFERENCE
941:               Set pOrigSpRef = pPolygon.SpatialReference
942:               If pOrigSpRef Is Nothing Then      ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL REFERENCE YET
                                                      ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
944:                 If TypeOf pSpRef Is IGeographicCoordinateSystem Then
945:                   Set pPolygon.SpatialReference = pSpRef
946:                 Else
947:                   Set pPrjSpRef = pSpRef
```

```
948:                    Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
949:                  End If
950:                ElseIf TypeOf pOrigSpRef Is IUnknownCoordinateSystem Then ' PRESUMABLY CAN ONLY HAPPEN IF DATA FRAME HAS NO SPATIAL
REFERENCE YET
                                                  ' THEREFORE TREAT LIKE GEOGRAPHIC COORDINATES
952:                  If TypeOf pSpRef Is IGeographicCoordinateSystem Then
953:                    Set pPolygon.SpatialReference = pSpRef
954:                  Else
955:                    Set pPrjSpRef = pSpRef
956:                    Set pPolygon.SpatialReference = pPrjSpRef.GeographicCoordinateSystem
957:                  End If
958:                End If
959:                If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRef, pPolygon.SpatialReference) Then
960:                  pPolygon.Project pSpRef
961:                End If
              ' ADD DATA TO FEATURE BUFFER
963:                Set pFeatureBuf.Shape = pPolygon
964:                Set pArea = pPolygon
965:                pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Unique_ID")) = lngCounter
966:                pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Area")) = pArea.Area
967:                If m_booSelPolygonHasNames Then pFeatureBuf.Value(pFeatureBuf.Fields.FindField("Name")) = strName
968:                pFeatureCursor.InsertFeature pFeatureBuf
969:              Next lngIndex
970:            End If
971:            pFeatureCursor.Flush
972:        End Select
973:      End If
974:    End If

  Dim pFeatureLayer As IFeatureLayer
977:    Set pFeatureLayer = New FeatureLayer
978:    Set pFeatureLayer.FeatureClass = pFeatureClass
979:    pFeatureLayer.Name = Linkages.aml_func_mod.ClipExtension(strfilename)
980:    m_MxDoc.FocusMap.AddLayer pFeatureLayer

982:    Call Linkages.MyGeneralOperations.EnableSelectTool(m_pApp)

984:    Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub cmdSpRef_Click()
  On Error GoTo ErrorHandler
```

```
  Dim pSpRef As ISpatialReference
  Dim pSpRefDlg As ISpatialReferenceDialog2

997:   Set pSpRefDlg = New SpatialReferenceDialog
998:   Set pSpRef = pSpRefDlg.DoModalCreate(False, False, False, Me.hWnd)

1000:   If Not pSpRef Is Nothing Then
1001:     lblSpRef.Caption = "Spatial Reference = " & pSpRef.Name
1002:     Set m_SpRef = pSpRef
1003:   End If

1005:   Call CheckEnable

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSpRef_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub CheckEnable()
  On Error GoTo ErrorHandler

  Dim OptionsOK As Boolean
1016:   OptionsOK = ((optNew.Value = True) And ((optPoint.Value = True) Or (optPolyline.Value = True) Or (optPolygon.Value = True)))
Or _
      ((optConvert.Value = True) And (lbxGraphics.ListIndex > -1))

  Dim booSpRefOK As Boolean
1020:   booSpRefOK = (Not m_SpRef Is Nothing)

1022:   If booSpRefOK Then
1023:     booSpRefOK = (Not TypeOf m_SpRef Is IUnknownCoordinateSystem)
'    MsgBox "In CheckEnable: " & vbCrLf & _
          "Spatial Reference is Unknown? " & CStr(TypeOf m_SpRef Is IUnknownCoordinateSystem) & vbCrLf & _
          "Spatial Reference Name = " & m_SpRef.Name
1027:   End If

1029:   If optNew.Value = True Then
1030:     optPolyline.Enabled = True
1031:     optPoint.Enabled = True
1032:     optPolygon.Enabled = True
1033:     frmNew.Enabled = True
1034:     lbxGraphics.Enabled = False
1035:     chkSelected.Enabled = False
1036:   ElseIf optConvert.Value = True Then
1037:     optPolyline.Enabled = False
```

```
1038:     optPoint.Enabled = False
1039:     optPolygon.Enabled = False
1040:     frmNew.Enabled = False
1041:     lbxGraphics.Enabled = True
1042:     chkSelected.Enabled = True
1043:   End If

1045:   cmdOK.Enabled = booSpRefOK And OptionsOK

  Exit Sub
ErrorHandler:
  HandleError False, "CheckEnable " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler

1055:   Unload Me


  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler

1066:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
1067:   Me.Left = (2 * Screen.Width / 3) - (2 * Me.Width / 3)
1068:   Me.Top = (2 * Screen.Height / 3) - (Me.Height / 3)

  Dim newUid As New uID
1071:   newUid.Value = "Linkages.Extension"
1072:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
1074:   Set ext = m_ExtensionConfig

  ' PRESET OPTIONS
1077:   optNew.Value = True
1078:   optConvert.Value = False
1079:   optPolygon.Value = True
1080:   optPolyline.Value = False
1081:   optPoint.Value = False
1082:   lbxGraphics.Clear
```

```
1084:   If (m_SelPointGraphics.Count > 0) Or (m_SelPolylineGraphics.Count > 0) Or (m_SelPolygonGraphics.Count > 0) Then
1085:     chkSelected.Value = 1
1086:     optConvert.Value = True
1087:     optNew.Value = False
1088:   Else
1089:     optConvert.Value = False
1090:     optNew.Value = True
1091:     chkSelected.Value = 0
1092:   End If

1094:   Call FillGraphicsListBox

  ' PRESET SPATIAL REFERENCE
  Dim pSpRef As ISpatialReference
1098:   Set pSpRef = m_MxDoc.FocusMap.SpatialReference
1099:   If Not pSpRef Is Nothing Then
1100:     If TypeOf pSpRef Is IUnknownCoordinateSystem Then
1101:       Set pSpRef = Nothing
1102:       Set m_SpRef = Nothing
1103:     Else
1104:       Set m_SpRef = pSpRef
1105:     End If
1106:   Else
1107:     Set m_SpRef = Nothing
1108:   End If

1110:   If pSpRef Is Nothing Then
1111:     lblSpRef.Caption = "Spatial Reference = <-- No Spatial Reference Set -->"
1112:   Else
1113:     lblSpRef.Caption = "Spatial Reference = " & pSpRef.Name
1114:   End If

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String
  Dim strUserName As String

1124:   strDirPath = ext.ClipDirectoryPath
1125:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1126:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
1127:   End If
1128:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
1129:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_pApp))
```

```
1130:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
1131:   End If

1133:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then
1134:     strDirPath = strDirPath & "\"
1135:   End If

1137:   strDirPath = Linkages.aml_func_mod.MakeUniqueFilename(strDirPath & "NewShape.shp")
1138:   txtOutput.Text = strDirPath
1139:   optNew.BackColor = RGB(194, 194, 194)
1140:   optConvert.BackColor = RGB(194, 194, 194)

1142:   Call CheckEnable

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub FillGraphicsListBox()
  On Error GoTo ErrorHandler


  Dim lngIndex As Long
1154:   lbxGraphics.Clear

1156:   If chkSelected.Value = 1 Then
1157:     For lngIndex = 0 To m_SelGraphicslist.Count - 1
1158:       lbxGraphics.AddItem (m_SelGraphicslist.Element(lngIndex))
1159:     Next lngIndex
1160:   Else
1161:     For lngIndex = 0 To m_AllGraphicsList.Count - 1
1162:       lbxGraphics.AddItem (m_AllGraphicsList.Element(lngIndex))
1163:     Next lngIndex
1164:   End If


  Exit Sub
ErrorHandler:
  HandleError False, "FillGraphicsListBox " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub FillGraphicsArrays()
  On Error GoTo ErrorHandler
```

```
  Dim strPointWord As String
  Dim strPolylineWord As String
  Dim strPolygonWord As String
  Dim strSelPointWord As String
  Dim strSelPolylineWord As String
  Dim strSelPolygonWord As String

  Dim pGraphicsContainer As IGraphicsContainer
  Dim pGraphicsContainerSel As IGraphicsContainerSelect
1185:   Set pGraphicsContainer = m_MxDoc.FocusMap
1186:   Set pGraphicsContainerSel = m_MxDoc.FocusMap

1188:   pGraphicsContainer.Reset

1190:   Set m_AllPointGraphics = New esriSystem.VarArray
1191:   Set m_SelPointGraphics = New esriSystem.VarArray
1192:   Set m_AllPolylineGraphics = New esriSystem.VarArray
1193:   Set m_SelPolylineGraphics = New esriSystem.VarArray
1194:   Set m_AllPolygonGraphics = New esriSystem.VarArray
1195:   Set m_SelPolygonGraphics = New esriSystem.VarArray

1197:   Set m_AllPointGraphicsNames = New esriSystem.strArray
1198:   Set m_SelPointGraphicsNames = New esriSystem.strArray
1199:   Set m_AllPolylineGraphicsNames = New esriSystem.strArray
1200:   Set m_SelPolylineGraphicsNames = New esriSystem.strArray
1201:   Set m_AllPolygonGraphicsNames = New esriSystem.strArray
1202:   Set m_SelPolygonGraphicsNames = New esriSystem.strArray

1204:   m_booAllPointHasNames = False
1205:   m_booSelPointHasNames = False
1206:   m_booAllPolylineHasNames = False
1207:   m_booSelPolylineHasNames = False
1208:   m_booAllPolygonHasNames = False
1209:   m_booSelPolygonHasNames = False

  Dim pElement As IElement
  Dim pElementProperties As IElementProperties
1213:   Set pElement = pGraphicsContainer.Next
  Dim pGeometryType As esriGeometryType
'  Dim pElProps3 As IElementProperties3

  Dim pClone As IClone
  Dim pGeometry As IGeometry

  Dim strName As String
  Dim pElementProps As IElementProperties
```

```
1223:   Do Until pElement Is Nothing

1225:     Set pElementProps = pElement
1226:     Set pClone = pElement.Geometry
1227:     Set pGeometry = pClone.Clone

'    MsgBox "Fill Graphics Array: Spatial Reference = " & CStr(pElement.Geometry.SpatialReference.Name)
'    MsgBox "Fill Graphics Array: pGeometry Spatial Reference = " & CStr(pGeometry.SpatialReference.Name)
'    Set pElProps3 = pElement
'    MsgBox "pElProps3.SpRef = " & pElProps3.SpatialReference.Name

1234:     pGeometryType = pGeometry.GeometryType
1235:     strName = pElementProps.Name

1237:     If pGeometryType = esriGeometryPoint Then
1238:       m_AllPointGraphics.Add pGeometry
1239:       m_AllPointGraphicsNames.Add strName
1240:       If strName <> "" Then m_booAllPointHasNames = True
1241:     ElseIf pGeometryType = esriGeometryPolyline Then
1242:       m_AllPolylineGraphics.Add pGeometry
1243:       m_AllPolylineGraphicsNames.Add strName
1244:       If strName <> "" Then m_booAllPolylineHasNames = True
1245:     ElseIf pGeometryType = esriGeometryPolygon Then
1246:       m_AllPolygonGraphics.Add pGeometry
1247:       m_AllPolygonGraphicsNames.Add strName
1248:       If strName <> "" Then m_booAllPolygonHasNames = True
1249:     End If
1250:     Set pElement = pGraphicsContainer.Next
1251:   Loop

'  MsgBox pGraphicsContainerSel.ElementSelectionCount & " elements selected..."

1255:   If pGraphicsContainerSel.ElementSelectionCount > 0 Then
    Dim pEnum As IEnumElement
1257:     Set pEnum = pGraphicsContainerSel.SelectedElements
1258:     pEnum.Reset
1259:     Set pElement = pEnum.Next
1260:     Do Until pElement Is Nothing
'      MsgBox "Looping..."
1262:       Set pElementProps = pElement
1263:       strName = pElementProps.Name
1264:       Set pClone = pElement.Geometry
1265:       Set pGeometry = pClone.Clone
1266:       pGeometryType = pGeometry.GeometryType
1267:       If pGeometryType = esriGeometryPoint Then
1268:         m_SelPointGraphics.Add pGeometry
1269:         m_SelPointGraphicsNames.Add strName
```

```
        If strName <> "" Then m_booSelPointHasNames = True
      ElseIf pGeometryType = esriGeometryPolyline Then
        m_SelPolylineGraphics.Add pGeometry
        m_SelPolylineGraphicsNames.Add strName
        If strName <> "" Then m_booSelPolylineHasNames = True
      ElseIf pGeometryType = esriGeometryPolygon Then
        m_SelPolygonGraphics.Add pGeometry
        m_SelPolygonGraphicsNames.Add strName
        If strName <> "" Then m_booSelPolygonHasNames = True
      End If
      Set pElement = pEnum.Next
    Loop
  End If

  If (m_AllPointGraphics.Count = 1) Then
    strPointWord = " Point"
  Else
    strPointWord = " Points"
  End If
  If (m_AllPolylineGraphics.Count = 1) Then
    strPolylineWord = " Polyline"
  Else
    strPolylineWord = " Polylines"
  End If
  If (m_AllPolygonGraphics.Count = 1) Then
    strPolygonWord = " Polygon"
  Else
    strPolygonWord = " Polygons"
  End If
  If (m_SelPointGraphics.Count = 1) Then
    strSelPointWord = " Point"
  Else
    strSelPointWord = " Points"
  End If
  If (m_SelPolylineGraphics.Count = 1) Then
    strSelPolylineWord = " Polyline"
  Else
    strSelPolylineWord = " Polylines"
  End If
  If (m_SelPolygonGraphics.Count = 1) Then
    strSelPolygonWord = " Polygon"
  Else
    strSelPolygonWord = " Polygons"
  End If

  Set m_AllGraphicsList = New esriSystem.strArray
  Set m_SelGraphicslist = New esriSystem.strArray
```

```
1318:   m_AllGraphicsList.Add "1) Point Shapefile: " & CStr(m_AllPointGraphics.Count) & strPointWord
1319:   m_AllGraphicsList.Add "2) Polyline Shapefile: " & CStr(m_AllPolylineGraphics.Count) & strPolylineWord
1320:   m_AllGraphicsList.Add "3) Polyline Shapefile: " & CStr(m_AllPolylineGraphics.Count) & strPolylineWord & " + " & _
                                                CStr(m_AllPolygonGraphics.Count) & strPolygonWord
1322:   m_AllGraphicsList.Add "4) Polygon Shapefile: " & CStr(m_AllPolygonGraphics.Count) & strPolygonWord
1323:   m_AllGraphicsList.Add "5) Polygon Shapefile: " & CStr(m_AllPolylineGraphics.Count) & strPolylineWord & " + " & _
                                                CStr(m_AllPolygonGraphics.Count) & strPolygonWord

1326:   m_SelGraphicslist.Add "1) Point Shapefile: " & CStr(m_SelPointGraphics.Count) & strSelPointWord
1327:   m_SelGraphicslist.Add "2) Polyline Shapefile: " & CStr(m_SelPolylineGraphics.Count) & strSelPolylineWord
1328:   m_SelGraphicslist.Add "3) Polyline Shapefile: " & CStr(m_SelPolylineGraphics.Count) & strSelPolylineWord & " + " & _
                                                CStr(m_SelPolygonGraphics.Count) & strSelPolygonWord
1330:   m_SelGraphicslist.Add "4) Polygon Shapefile: " & CStr(m_SelPolygonGraphics.Count) & strSelPolygonWord
1331:   m_SelGraphicslist.Add "5) Polygon Shapefile: " & CStr(m_SelPolylineGraphics.Count) & strSelPolylineWord & " + " & _
                                                CStr(m_SelPolygonGraphics.Count) & strSelPolygonWord



  Exit Sub
ErrorHandler:
  HandleError False, "FillGraphicsArrays " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler

1344:   Set m_pApp = Nothing
1345:   Set m_MxDoc = Nothing
1346:   Set m_SpRef = Nothing
1347:   Set m_ExtensionConfig = Nothing
1348:   Set m_AllPointGraphics = Nothing
1349:   Set m_SelPointGraphics = Nothing
1350:   Set m_AllPolylineGraphics = Nothing
1351:   Set m_SelPolylineGraphics = Nothing
1352:   Set m_AllPolygonGraphics = Nothing
1353:   Set m_SelPolygonGraphics = Nothing
1354:   Set m_AllGraphicsList = Nothing
1355:   Set m_SelGraphicslist = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub
```

```
Private Sub lbxGraphics_Click()
1363:   Call CheckEnable
End Sub

Private Sub optConvert_Click()
  On Error GoTo ErrorHandler

1369:   Call FillGraphicsListBox
1370:   Call CheckEnable

  Exit Sub
ErrorHandler:
  HandleError True, "optConvert_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optNew_Click()
  On Error GoTo ErrorHandler

1380:   Call FillGraphicsListBox
1381:   Call CheckEnable


  Exit Sub
ErrorHandler:
  HandleError True, "optNew_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optPoint_Click()
  On Error GoTo ErrorHandler

1392:   Call CheckEnable


  Exit Sub
ErrorHandler:
  HandleError True, "optPoint_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optPolygon_Click()
  On Error GoTo ErrorHandler

1403:   Call CheckEnable
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "optPolygon_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optPolyline_Click()
  On Error GoTo ErrorHandler

1414:   Call CheckEnable


  Exit Sub
ErrorHandler:
  HandleError True, "optPolyline_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 9: frmHabSuitStats.frm

```
VERSION 5.00
Begin VB.Form frmHabSuitStats
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Habitat Suitability Model Statistics:"
   ClientHeight    =   3465
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   5040
   Icon            =   "frmHabSuitStats.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   3465
   ScaleWidth      =   5040
   ShowInTaskbar   =   0   'False
   StartUpPosition =   1  'CenterOwner
   Begin VB.CommandButton cmdGetFile
      Height          =   360
      Left            =   4358
      Picture         =   "frmHabSuitStats.frx":038A
      Style           =   1  'Graphical
      TabIndex        =   2
      ToolTipText     =   "Browse for Output Filename..."
      Top             =   2505
      Width           =   555
   End
```

```
Begin VB.TextBox txtOutput
   Height          =   330
   Left            =   165
   Locked          =   -1  'True
   TabIndex        =   1
   Top             =   2535
   Width           =   4110
End
Begin VB.TextBox txtNumBins
   Height          =   345
   Left            =   2970
   TabIndex        =   3
   Top             =   2955
   Width           =   1035
End
Begin VB.CommandButton cmdOK
   Caption         =   "OK"
   Height          =   345
   Left            =   3937
   TabIndex        =   6
   Top             =   1755
   Width           =   1080
End
Begin VB.CommandButton cmdHelp
   Caption         =   "Help"
   Height          =   345
   Left            =   3937
   TabIndex        =   5
   Top             =   1350
   Width           =   1080
End
Begin VB.CommandButton cmdCancel
   Caption         =   "Cancel"
   Height          =   345
   Left            =   3937
   TabIndex        =   4
   Top             =   945
   Width           =   1080
End
Begin VB.ListBox lbxLayers
   Height          =   1815
   Left            =   15
   TabIndex        =   0
   Top             =   300
   Width           =   3840
End
Begin VB.Image Image1
```

```
      BorderStyle     =   1  'Fixed Single
      Height          =   1215
      Left            =   45
      Top             =   2220
      Width           =   4965
   End
   Begin VB.Label lbl3
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Specify folder for output tables:"
      Height          =   195
      Left            =   165
      TabIndex        =   9
      Top             =   2295
      Width           =   2190
   End
   Begin VB.Label lblHistogramBins
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Number of Histogram Bins:"
      Height          =   195
      Left            =   1028
      TabIndex        =   8
      Top             =   3015
      Width           =   1875
   End
   Begin VB.Image imgCorrIcon
      Height          =   855
      Left            =   4005
      Picture         =   "frmHabSuitStats.frx":0400
      Top             =   30
      Width           =   945
   End
   Begin VB.Label lblSelLayer
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Select Habitat Suitability Grid"
      Height          =   195
      Left            =   915
      TabIndex        =   7
      Top             =   45
      Width           =   2040
   End
End
Attribute VB_Name = "frmHabSuitStats"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
```

```
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private m_App As IApplication
Private m_ExtensionConfig As IExtensionConfig
Private m_CurrentSelected As IRasterLayer
Private m_RasterLayers As esriSystem.IVariantArray
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmHabSuitStats.frm"



Public Property Set ArcApp(ByVal vNewValue As IApplication)
  On Error GoTo ErrorHandler


15:   Set m_App = vNewValue


  Exit Property
ErrorHandler:
  HandleError True, "ArcApp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Set theRasterLayers(ByVal theLayers As esriSystem.IVariantArray)
  On Error GoTo ErrorHandler


27:   Set m_RasterLayers = theLayers


  Exit Property
ErrorHandler:
  HandleError True, "theRasterLayers " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set theCurrentSelected(ByVal aPossibleLayer As IRasterLayer)
  On Error GoTo ErrorHandler


39:   Set m_CurrentSelected = aPossibleLayer


  Exit Property
ErrorHandler:
  HandleError True, "theCurrentSelected " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Property

Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler

50:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdGetFile_Click()
  On Error GoTo ErrorHandler


61:   SetWindowPos Me.hWnd, -2, 0, 0, 0, 0, &H1 Or &H2

  Dim strDirPath As String
  Dim strUserName As String

66:   strDirPath = txtOutput.Text
67:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then strDirPath = strDirPath & "\"
68:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
69:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_App))
70:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
71:   End If

  Dim boolWorkspaceExists As Boolean
74:   boolWorkspaceExists = Not Dir$(strDirPath) = ""

76:   If Not boolWorkspaceExists Then
77:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.TempPathLocation)
78:   End If

  Dim pGxDialog As IGxDialog
81:   Set pGxDialog = New GxDialog

  Dim pGxDialogFilter As IGxObjectFilter
'  Set pGxDialogFilter = New GxFilterWorkspaces
85:   Set pGxDialogFilter = New GxFilterBasicTypes
'  pGxDialogFilter.Name = "Folders"
'  Set pGxDialogFilter = New GxFilterContainers      ' INCLUDED GRIDS AND COVERAGES

  Dim pGxObject As IGxObject
```

```
  Dim pGxSelection As IEnumGxObject

92:   With pGxDialog
93:     .AllowMultiSelect = False
94:     .StartingLocation = strDirPath
95:     .Title = "Please select (don't open!) folder to contain your dBASE Tables:"
96:     Set .ObjectFilter = pGxDialogFilter
97:   End With

  Dim theFinalString As String
'  If Not pGxDialog.DoModalOpen(0, pEnumGx) Then
'Exit Sub 'Exit if user press Cancel
'End If
'MsgBox pEnumGx.Next.FullName
104:   If (pGxDialog.DoModalOpen(Me.hWnd, pGxSelection) = True) Then
    'Set pGxObject = pGxDialog.FinalLocation
106:     Set pGxObject = pGxSelection.Next

    Dim pGxFile As IGxFile
109:     Set pGxFile = pGxObject

111:     theFinalString = pGxObject.FullName
112:     If (Right(theFinalString, 1) <> "\") And (Right(theFinalString, 1) <> "/") Then theFinalString = theFinalString & "\"

'    If aml_func_mod.ExistFileDir(theFinalString) Then
'      theFinalString = aml_func_mod.MakeUniqueFilename(theFinalString)
'
'      MsgBox "Unable to overwrite the file '" & pGxDialog.Name & "'.  The new file will be saved to " & _
'        theFinalString & "..."
'    End If

121:     txtOutput.Text = theFinalString

123:   End If


126:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2




  Exit Sub
ErrorHandler:
  HandleError True, "cmdGetFile_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub cmdHelp_Click()
  On Error GoTo ErrorHandler


  Dim strPath As String
142:   strPath = App.Path & "\help"

144:   Call Linkages.MyGeneralOperations.OpenDoc("Hab_Statistics_Subdocument.pdf", strPath)


  Exit Sub
ErrorHandler:
  HandleError True, "cmdHelp_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler

  ' UPDATE SAVED NUMBER OF BINS VALUE
  Dim newUid As New uID
157:   newUid.Value = "Linkages.Extension"
158:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
160:   Set ext = m_ExtensionConfig
161:   ext.HistogramBinCount = CLng(txtNumBins.Text)

  Dim theTimeBegan As Date
164:   theTimeBegan = Now

  ' UPDATE CURRENT SAVE FOLDER
  Dim strWorkFolder As String
168:   strWorkFolder = txtOutput.Text
169:   If Right(strWorkFolder, 1) <> "\" And Right(strWorkFolder, 1) <> "/" Then
170:     strWorkFolder = strWorkFolder & "\"
171:   End If
172:   ext.ClipDirectoryPath = strWorkFolder

  ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
176:   Set psbar = m_App.StatusBar
  Dim pPro As IStepProgressor
178:   Set pPro = psbar.ProgressBar
179:   Screen.MousePointer = vbHourglass

  ' CALCULATE STATS
```

```
  Dim pRasterLayer As IRasterLayer
  Dim pRaster As IRaster
  Dim pRasterBandCollection As IRasterBandCollection
  Dim pRasterBand As IRasterBand
  Dim pRasterStatistics As IRasterStatistics
  Dim pRasterDataset As IRasterDataset
  Dim dblCellSize As Double
  Dim pLogicalOp As ILogicalOp
  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit
  Dim pCountRasterBand As IRasterBand
  Dim pCountRasterBandCollection As IRasterBandCollection
  Dim pCountGeoDataset As IGeoDataset
  Dim pCountCursor As ICursor
  Dim pCountRow As IRow
  Dim lngCountValField As Long
  Dim lngCountCountField As Long
  Dim lngCountValValue As Long
  Dim lngCountCountValue As Long
  Dim lngNumBins As Long
  Dim strReport As String
  Dim strHistReport As String
  Dim pTable As ITable
  Dim strfilename As String

209:   Set pRasterLayer = m_RasterLayers.Element(lbxLayers.ListIndex)
210:   Set pRaster = pRasterLayer.Raster
211:   Set pRasterBandCollection = pRaster
212:   Set pRasterBand = pRasterBandCollection.Item(0)
213:   Set pRasterStatistics = pRasterBand.Statistics
214:   Set pRasterDataset = pRasterBand.RasterDataset

216:   lngNumBins = CLng(txtNumBins.Text)

  ' MAKE EMPTY FIELD INFO ARRAY; HISTOGRAM FUNCTION EXPECTS IT
  Dim pFieldInfoArray As esriSystem.IStringArray
220:   Set pFieldInfoArray = New esriSystem.strArray

  ' GET INFORMATION ON FIELD AND ADD TO FIELD INFO ARRAY
223:   pFieldInfoArray.Add pRasterLayer.Name

  ' GET CELL SIZE
226:   dblCellSize = (Linkages.GridFunctions.ReturnCellSize(pRaster)) ^ 2

  ' GET GENERAL COUNT OF NON-NULL CELL VALUES
```

```
229:   Set pLogicalOp = New RasterMathOps
230:   Set pCountGeoDataset = pLogicalOp.IsNull(pRaster)
231:   Set pCountRasterBandCollection = pCountGeoDataset
232:   Set pCountRasterBand = pCountRasterBandCollection.Item(0)
233:   Set pTable = pCountRasterBand.AttributeTable
234:   lngCountValField = pTable.FindField("Value")
235:   lngCountCountField = pTable.FindField("Count")
236:   lngCountCountValue = -9999
237:   Set pCountCursor = pTable.Search(Nothing, True)
238:   Set pCountRow = pCountCursor.NextRow
239:   Do Until pCountRow Is Nothing
240:     lngCountValValue = pCountRow.Value(lngCountValField)
241:     If lngCountValValue = 0 Then
242:       lngCountCountValue = pCountRow.Value(lngCountCountField)
243:     End If
244:     Set pCountRow = pCountCursor.NextRow
245:   Loop

  Dim lngHistArray() As Long
248:   lngHistArray = Linkages.CorridorAnalysisFunctions.GridHistogram(pRasterLayer, 0, _
      100, lngNumBins, m_App)

'  strReport = "Statistics Report on Grid '" & pRasterLayer.Name & "':" & vbCrLf & _
      "-------------------------------------------------" & vbCrLf & _
      "Non-Null Grid Cell Count = " & Linkages.aml_func_mod.InsertCommas(lngCountCountValue) & _
      " grid cells" & vbCrLf & _
      "-------------------------------------------------" & vbCrLf & _
      "  --> Continuous Grid Value Statistics" & vbCrLf & _
      "  --> Data saved to zzzFilename" & vbCrLf & _
      "  --> Statistics: " & vbCrLf & _
      "       a] Minimum = " & CStr(pRasterStatistics.Minimum) & vbCrLf & _
      "       b] Maximum = " & CStr(pRasterStatistics.Maximum) & vbCrLf & _
      "       c] Mean = " & CStr(pRasterStatistics.Mean) & vbCrLf & _
      "       d] Standard Deviation = " & CStr(pRasterStatistics.StandardDeviation) & vbCrLf & _
      "       e] Histogram:" & vbCrLf

265:   strReport = _
      "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fprq2\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\tx90\tx360\tx450\tx720\b\f0\fs16 Statistics Report on Grid '" & _
      pRasterLayer.Name & "'\b0\par" & vbCrLf & _
      "\pard ------------------------------------------------------------------------------\par" & vbCrLf & _
      "\b Non-Null Grid Cell Count: \b0 " & Linkages.aml_func_mod.InsertCommas(lngCountCountValue) & _
        " grid cells\par" & vbCrLf & _
      "------------------------------------------------------------------------------\par" & vbCrLf & _
      "\b Continuous Grid Value Statistics\b0\par" & vbCrLf & _
      "\b Data saved to \b0 zzzFilename\par" & vbCrLf & _
      "\b Statistics: \b0\par" & vbCrLf & _
```

```
      "\b     a] Minimum: \b0 " & CStr(pRasterStatistics.Minimum) & "\par" & vbCrLf & _
      "\b     b] Maximum: \b0 " & CStr(pRasterStatistics.Maximum) & "\par" & vbCrLf & _
      "\b     c] Mean: \b0 " & CStr(pRasterStatistics.Mean) & "\par" & vbCrLf & _
      "\b     d] Standard Deviation: \b0 " & CStr(pRasterStatistics.StandardDeviation) & "\par" & vbCrLf & _
      "\b     e] Histogram: \b0\par" & vbCrLf

282:   If UBound(lngHistArray) = 0 Then
283:     strReport = strReport & "        !!! Single Value:  No Histogram created...\par" & vbCrLf
284:   Else
    ' REMEMBER THAT HABITAT SUITABILITY ANALYSIS ALWAYS GOES FROM 0 TO 100
286:     strHistReport = Linkages.CorridorAnalysisFunctions.MakeHistogramData(pFieldInfoArray, lngNumBins, 0, _
       100, lngHistArray, m_App, txtOutput.Text, 1)
288:     strReport = strReport & strHistReport
289:   End If

  ' MAKE NEW TABLE
292:   Set pNewFields = New Fields
293:   Set pNewFieldsEdit = pNewFields
294:   pNewFieldsEdit.FieldCount = 5

  ' MAKE UNIQUE ID FIELD
297:   Set pNewField = New Field
298:   Set pNewFieldEdit = pNewField
299:   pNewFieldEdit.Name = "Unique_ID"
300:   pNewFieldEdit.Type = esriFieldTypeInteger
301:   pNewFieldEdit.Precision = 8
302:   Set pNewFieldsEdit.Field(0) = pNewField

  ' MAKE STAT FIELDS
305:   Set pNewField = New Field
306:   Set pNewFieldEdit = pNewField
307:   With pNewFieldEdit
308:     .Type = esriFieldTypeDouble
309:     .Name = "Minimum"
310:     .Precision = 14
311:     .Scale = 8
312:   End With
313:   Set pNewFieldsEdit.Field(1) = pNewField

315:   Set pNewField = New Field
316:   Set pNewFieldEdit = pNewField
317:   With pNewFieldEdit
318:     .Type = esriFieldTypeDouble
319:     .Name = "Maximum"
320:     .Precision = 14
321:     .Scale = 8
322:   End With
```

```
323:   Set pNewFieldsEdit.Field(2) = pNewField

325:   Set pNewField = New Field
326:   Set pNewFieldEdit = pNewField
327:   With pNewFieldEdit
328:     .Type = esriFieldTypeDouble
329:     .Name = "Mean"
330:     .Precision = 14
331:     .Scale = 8
332:   End With
333:   Set pNewFieldsEdit.Field(3) = pNewField

335:   Set pNewField = New Field
336:   Set pNewFieldEdit = pNewField
337:   With pNewFieldEdit
338:     .Type = esriFieldTypeDouble
339:     .Name = "St_Dev"
340:     .Precision = 14
341:     .Scale = 8
342:   End With
343:   Set pNewFieldsEdit.Field(4) = pNewField

345:   strfilename = txtOutput.Text
346:   If Right(strfilename, 1) <> "/" And Right(strfilename, 1) <> "\" Then strfilename = strfilename & "\"
347:   strfilename = strfilename & pRasterLayer.Name & "_stats.dbf"
348:   strfilename = Linkages.aml_func_mod.MakeUniqueFilename(strfilename)
349:   Set pTable = Linkages.aml_func_mod.CreatedBASETable(strfilename, pNewFields)

  ' ADD DATA
  Dim pRow As IRow
353:   Set pRow = pTable.CreateRow
354:   pRow.Value(pTable.FindField("Unique_ID")) = 1
355:   pRow.Value(pTable.FindField("Minimum")) = pRasterStatistics.Minimum
356:   pRow.Value(pTable.FindField("Maximum")) = pRasterStatistics.Maximum
357:   pRow.Value(pTable.FindField("Mean")) = pRasterStatistics.Mean
358:   pRow.Value(pTable.FindField("St_Dev")) = pRasterStatistics.StandardDeviation
359:   pRow.Store

  ' MAKE TABLE AND ADD TO MAP DOCUMENT
  Dim pNewStandaloneTable As IStandaloneTable
  Dim pTableWindow2 As ITableWindow2
  Dim pStandaloneTableCollection As IStandaloneTableCollection
  Dim pMxDoc As IMxDocument

367:   Set pNewStandaloneTable = New StandaloneTable
368:   Set pNewStandaloneTable.Table = pTable
```

```
370:   Set pTableWindow2 = New TableWindow

372:   With pTableWindow2
373:     Set .StandaloneTable = pNewStandaloneTable
374:     Set .Application = m_App
375:     .TableSelectionAction = esriSelectFeatures
376:     .ShowAliasNamesInColumnHeadings = True
377:     .ShowSelected = False
378:     .Show True
379:   End With
380:   Set pMxDoc = m_App.Document
381:   Set pStandaloneTableCollection = pMxDoc.FocusMap
382:   pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable

384:   pMxDoc.UpdateContents

386:   strReport = Linkages.aml_func_mod.SubstituteString(strReport, "zzzFilename", strfilename)

388:   psbar.HideProgressBar
389:   Screen.MousePointer = vbDefault

391:   strReport = strReport & _
          "==========================================================\par" & vbCrLf & _
          Linkages.MyGeneralOperations.ReturnTimeElapsedRTF(theTimeBegan, Now, 7) & "}"

  ' SHOW REPORT
  Dim frmReportForm As New Linkages.frmReport_modal
397:   frmReportForm.txtReport.TextRTF = strReport
398:   frmReportForm.Show vbModal

  ' UNLOAD DIALOG
401:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub CheckOK()
  On Error GoTo ErrorHandler


  Dim booCheckOK As Boolean
413:   booCheckOK = (txtNumBins.Text <> "") And IsNumeric(txtNumBins.Text) And (lbxLayers.ListIndex > -1)
414:   cmdOK.Enabled = booCheckOK
```

```
  Exit Sub
ErrorHandler:
  HandleError False, "CheckOK " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler

425:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2

  Dim newUid As New uID
428:   newUid.Value = "Linkages.Extension"
429:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
431:   Set ext = m_ExtensionConfig

433:   If ext.HistogramBinCount > 0 Then
434:     txtNumBins.Text = CStr(ext.HistogramBinCount)
435:   Else
436:     txtNumBins.Text = "9"
437:     ext.HistogramBinCount = 9
438:   End If

440:   lbxLayers.Clear
  Dim pRasterLayer As IRasterLayer
  Dim lngIndex As Long
  Dim lngSelIndex As Long
444:   lngSelIndex = -9999
445:   For lngIndex = 0 To m_RasterLayers.Count - 1
446:     Set pRasterLayer = m_RasterLayers.Element(lngIndex)
447:     If pRasterLayer Is m_CurrentSelected Then lngSelIndex = lngIndex
448:     lbxLayers.AddItem pRasterLayer.Name
449:   Next lngIndex

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String
  Dim strUserName As String

459:   strDirPath = ext.ClipDirectoryPath
460:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
461:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
462:   End If
```

```
463:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
464:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(m_App))
465:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
466:   End If

468:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then
469:     strDirPath = strDirPath & "\"
470:   End If

472:   txtOutput.Text = strDirPath

474:   Call CheckOK

476:   If lngSelIndex > -1 Then
477:     lbxLayers.ListIndex = lngSelIndex
478:     Call lbxLayers_Click
479:   End If

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub Form_Terminate()
  On Error GoTo ErrorHandler

489:   Set m_CurrentSelected = Nothing
490:   Set m_RasterLayers = Nothing
491:   Set m_App = Nothing
492:   Set m_ExtensionConfig = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
```

```
End Sub

Private Sub lbxLayers_Click()
  On Error GoTo ErrorHandler

513:   Call CheckOK

  Exit Sub
ErrorHandler:
  HandleError True, "lbxLayers_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub txtNumBins_Change()
  On Error GoTo ErrorHandler

523:   Call CheckOK

  Exit Sub
ErrorHandler:
  HandleError True, "txtNumBins_Change " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub txtNumBins_KeyPress(KeyAscii As Integer)
  On Error GoTo ErrorHandler

533:   Call Linkages.MyGeneralOperations.CheckNumericRealPositive(KeyAscii, txtNumBins)

  Exit Sub
ErrorHandler:
  HandleError True, "txtNumBins_KeyPress " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 10: frmReport_modal.frm

```
VERSION 5.00
Object = "{3B7C8863-D78F-101B-B9B5-04021C009402}#1.2#0"; "RICHTX32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Begin VB.Form frmReport_modal
   Caption         =   "Report"
   ClientHeight    =   3180
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   6540
   Icon            =   "frmReport_modal.frx":0000
```

```
LinkTopic       =   "Form1"
LockControls    =   -1  'True
ScaleHeight     =   3180
ScaleWidth      =   6540
StartUpPosition =   1  'CenterOwner
Begin MSComDlg.CommonDialog CommonDialog1
   Left            =   225
   Top             =   2790
   _ExtentX        =   847
   _ExtentY        =   847
   _Version        =   393216
End
Begin RichTextLib.RichTextBox txtReport
   Height          =   2460
   Left            =   180
   TabIndex        =   3
   Top             =   165
   Width           =   6315
   _ExtentX        =   11139
   _ExtentY        =   4339
   _Version        =   393217
   BorderStyle     =   0
   ReadOnly        =   -1  'True
   ScrollBars      =   2
   Appearance      =   0
   TextRTF         =   $"frmReport_modal.frx":038A
End
Begin VB.CommandButton cmdOK
   Caption         =   "&Exit"
   Default         =   -1  'True
   Height          =   405
   Left            =   4080
   TabIndex        =   0
   Top             =   2745
   Width           =   1500
End
Begin VB.CommandButton cmdPrint
   Caption         =   "&Print"
   Height          =   405
   Left            =   960
   TabIndex        =   1
   Top             =   2745
   Width           =   1500
End
Begin VB.CommandButton cmdCopy
   Caption         =   "&Copy to Clipboard"
   Height          =   405
```

```
      Left            =   2520
      TabIndex        =   2
      Top             =   2745
      Width           =   1500
   End
   Begin VB.TextBox txtBackground
      Height          =   2670
      Left            =   30
      Locked          =   -1  'True
      TabIndex        =   4
      TabStop         =   0   'False
      Top             =   15
      Width           =   6510
   End
   Begin VB.Image imgCornerBars
      Height          =   225
      Left            =   6315
      Picture         =   "frmReport_modal.frx":040C
      Top             =   2955
      Width           =   225
   End
End
Attribute VB_Name = "frmReport_modal"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Anchors As AnchorObjectList  ' Main anchor control object
Private Const AnInch As Long = 1440   '1440 twips per inch
Private Const QuarterInch As Long = 360

Private Type RECT
   Left As Long
   Top As Long
   Right As Long
   Bottom As Long
End Type

Private Type CharRange
  cpMin As Long     ' First character of range (0 for start of doc)
  cpMax As Long     ' Last character of range (-1 for end of doc)
End Type

Private Type FormatRange
  hdc As Long       ' Actual DC to draw on
  hdcTarget As Long ' Target DC for determining text formatting
```

```
  rc As RECT        ' Region of the DC to draw to (in twips)
  rcPage As RECT    ' Region of the entire DC (page size) (in twips)
  chrg As CharRange ' Range of text to draw (see above declaration)
End Type

Private Const WM_USER As Long = &H400
Private Const EM_FORMATRANGE As Long = WM_USER + 57
Private Const EM_SETTARGETDEVICE As Long = WM_USER + 72
Private Const PHYSICALOFFSETX As Long = 112
Private Const PHYSICALOFFSETY As Long = 113

Private Declare Function GetDeviceCaps Lib "gdi32" ( _
   ByVal hdc As Long, ByVal nIndex As Long) As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
   (ByVal hWnd As Long, ByVal msg As Long, ByVal wp As Long, _
   lp As Any) As Long
Private Declare Function CreateDC Lib "gdi32" Alias "CreateDCA" _
   (ByVal lpDriverName As String, ByVal lpDeviceName As String, _
   ByVal lpOutput As Long, ByVal lpInitData As Long) As Long

Option Explicit
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmReport_modal.frm"

Private Sub cmdCopy_Click()
  On Error GoTo ErrorHandler

'    Clipboard.Clear
'    Clipboard.SetText (txtReport.Text)
'    cmdOK.SetFocus

51:     Clipboard.Clear
52:     Clipboard.SetText txtReport.TextRTF, vbCFRTF
53:     cmdOK.SetFocus

  Exit Sub
ErrorHandler:
  HandleError True, "cmdCopy_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler


64:   Unload Me
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub


Private Sub cmdPrint_Click()
  On Error GoTo ErrHand

'   Dim sinOutMemo As Single           ' Single:  used to break up strComments
'   Dim sinLenOutMemo As Single        ' Single:  used to mark length of strComments
'   Dim strPrintOutMemo As String      ' Printer input memo string
'
''~~~~~~~~~~This section formats the definitions for the printer, by breaking the definition
''          up into printer-sized strings and sticking "vbNewLines" between them.  It also
''          searches for spaces to make the breaks at.
'
'
'    strPrintOutMemo = txtReport.Text
'    sinLenOutMemo = Len(strPrintOutMemo)
'
'    Dim sinTestNum As Long
'    sinTestNum = InStr(1, strPrintOutMemo, vbNewLine)
'    If (sinTestNum < 70) And (sinTestNum <> 0) Then
'      sinOutMemo = sinTestNum
'    Else
'      sinOutMemo = 70
'    End If
'
'    Dim theDecreaseInc As Long
'
'    Do While sinLenOutMemo > sinOutMemo
'        sinTestNum = InStr(sinOutMemo, strPrintOutMemo, vbNewLine)
'        If sinTestNum < (sinOutMemo + 70) And sinTestNum <> 0 Then
'          sinOutMemo = InStr(sinOutMemo, strPrintOutMemo, vbNewLine) + 1
'        Else
'          sinOutMemo = sinOutMemo + 70
'          theDecreaseInc = 70
'          Do While InStr(sinOutMemo, strPrintOutMemo, " ") <> sinOutMemo And InStr(sinOutMemo, strPrintOutMemo, "-") <> sinOutMemo
'              sinOutMemo = sinOutMemo - 1
'              theDecreaseInc = theDecreaseInc - 1
'              If theDecreaseInc = 0 Then
'                sinOutMemo = sinOutMemo + 70
'                Exit Do
'              End If
'          Loop
```

```
'          strPrintOutMemo = Left(strPrintOutMemo, sinOutMemo) & vbNewLine & Mid(strPrintOutMemo, sinOutMemo + 1)
'        End If
'    Loop
'
'
'    Printer.FontSize = 12
'    Printer.Print strPrintOutMemo
'    Printer.Print
'    Printer.EndDoc
'
'cmdOK.SetFocus
124:   CommonDialog1.PrinterDefault = True
125:   CommonDialog1.CancelError = True
126:   CommonDialog1.ShowPrinter
  On Error Resume Next
128:   If Err Then
      ' This code runs if the dialog was cancelled
'      MsgBox "Dialog Cancelled"
      Exit Sub
132:   End If

134: '  MsgBox Printer.DeviceName

136:   PrintRTF txtReport, AnInch, AnInch, AnInch, AnInch
137:   cmdOK.SetFocus

  Exit Sub
ErrHand:

End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler

147: SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
148:   Set Anchors = New AnchorObjectList  ' Create new instance
149:    With Anchors
150:       With .Item(txtBackground)
151:          .SetAnchors enumStartEnd, enumStartEnd
152:       End With
153:       With .Item(txtReport)
154:          .SetAnchors enumStartEnd, enumStartEnd
155:       End With
156:       With .Item(cmdOK)
157:          .SetAnchors enumNone, enumSizeEnd
158:       End With
159:       With .Item(cmdCopy)
```

```
160:           .SetAnchors enumNone, enumSizeEnd
161:       End With
162:       With .Item(cmdPrint)
163:           .SetAnchors enumNone, enumSizeEnd
164:       End With
165:       With .Item(imgCornerBars)
166:           .SetAnchors enumSizeEnd, enumSizeEnd
167:       End With
168:       .Form = Me  ' Set form reference (suggested to be last step)
169:    End With

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Private Sub Form_Resize()
  On Error GoTo ErrorHandler

'Move objects according to new object size.

'Dim theFrmHeight As Long
'Dim theFrmWidth As Long
'theFrmHeight = Me.Height
'theFrmWidth = Me.Width
'
'
''Debug.Print "Height = " & theFrmHeight & ",   Width = " & theFrmWidth
'
'
''If pWindowPosition.Top < 3000 And theFrmHeight > 2600 Then
'
'If theFrmWidth > 2000 And theFrmHeight > 2600 Then
'    txtBackground.Height = theFrmHeight - 915
'    txtBackground.Width = theFrmWidth - 195
'    txtReport.Height = theFrmHeight - 1125
'    txtReport.Width = theFrmWidth - 390
'    cmdOK.Top = theFrmHeight - 840
'    cmdCopy.Top = theFrmHeight - 840
'    cmdPrint.Top = theFrmHeight - 840
'    imgCornerBars.Top = theFrmHeight - 630
'    imgCornerBars.Left = theFrmWidth - 345
'End If

'If frmOutDefinition.Top < 45000 And frmOutDefinition.Height > 2600 Then
'    txtBackground.Height = frmOutDefinition.Height - 915
'    txtBackground.Width = frmOutDefinition.Width - 195
```

```
'    txtDefinition.Height = frmOutDefinition.Height - 1125
'    txtDefinition.Width = frmOutDefinition.Width - 390
'    cmdOK.Top = frmOutDefinition.Height - 840
'    cmdOK.Left = frmOutDefinition.Width - 1905
'    cmdCopy.Top = frmOutDefinition.Height - 840
'    cmdCopy.Left = frmOutDefinition.Width - 3578
'    cmdPrint.Top = frmOutDefinition.Height - 840
'    cmdPrint.Left = frmOutDefinition.Width - 5265
'End If


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Resize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' WYSIWYG_RTF - Sets an RTF control to display itself the same as it
'               would print on the default printer
'
' RTF - A RichTextBox control to set for WYSIWYG display.
'
' LeftMarginWidth - Width of desired left margin in twips
'
' RightMarginWidth - Width of desired right margin in twips
'
' Returns - The length of a line on the printer in twips
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub WYSIWYG_RTF(RTF As RichTextBox, LeftMarginWidth As Long, RightMarginWidth As Long, TopMarginWidth As Long,
BottomMarginWidth As Long, PrintableWidth As Long, PrintableHeight As Long)
  On Error GoTo ErrorHandler

   Dim LeftOffset As Long
   Dim LeftMargin As Long
   Dim RightMargin As Long
   Dim TopOffset As Long
   Dim TopMargin As Long
   Dim BottomMargin As Long
   Dim PrinterhDC As Long
   Dim r As Long

248:    Printer.ScaleMode = vbTwips

   ' Get the left offset to the printable area on the page in twips
251:    LeftOffset = GetDeviceCaps(Printer.hdc, PHYSICALOFFSETX)
```

```
252:    LeftOffset = Printer.ScaleX(LeftOffset, vbPixels, vbTwips)

   ' Calculate the Left, and Right margins
255:    LeftMargin = LeftMarginWidth - LeftOffset
256:    RightMargin = (Printer.Width - RightMarginWidth) - LeftOffset

   ' Calculate the line width
259:    PrintableWidth = RightMargin - LeftMargin

   ' Get the top offset to the printable area on the page in twips
262:    TopOffset = GetDeviceCaps(Printer.hdc, PHYSICALOFFSETY)
263:    TopOffset = Printer.ScaleX(TopOffset, vbPixels, vbTwips)

   ' Calculate the Left, and Right margins
266:    TopMargin = TopMarginWidth - TopOffset
267:    BottomMargin = (Printer.Height - BottomMarginWidth) - TopOffset

   ' Calculate the line width
270:    PrintableHeight = BottomMargin - TopMargin


   ' Create an hDC on the Printer pointed to by the Printer object
   ' This DC needs to remain for the RTF to keep up the WYSIWYG display
275:    PrinterhDC = CreateDC(Printer.DriverName, Printer.DeviceName, 0, 0)

   ' Tell the RTF to base it's display off of the printer
   '    at the desired line width
279:    r = SendMessage(RTF.hWnd, EM_SETTARGETDEVICE, PrinterhDC, _
      ByVal PrintableWidth)

  Exit Sub
ErrorHandler:
  HandleError True, "WYSIWYG_RTF " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' PrintRTF - Prints the contents of a RichTextBox control using the
'            provided margins
'
' RTF - A RichTextBox control to print
'
' LeftMarginWidth - Width of desired left margin in twips
'
' TopMarginHeight - Height of desired top margin in twips
'
```

```
' RightMarginWidth - Width of desired right margin in twips
'
' BottomMarginHeight - Height of desired bottom margin in twips
'
' Notes - If you are also using WYSIWYG_RTF() on the provided RTF
'         parameter you should specify the same LeftMarginWidth and
'         RightMarginWidth that you used to call WYSIWYG_RTF()
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub PrintRTF(RTF As RichTextBox, LeftMarginWidth As Long, _
   TopMarginHeight, RightMarginWidth, BottomMarginHeight)
  On Error GoTo ErrorHandler

   Dim LeftOffset As Long, TopOffset As Long
   Dim LeftMargin As Long, TopMargin As Long
   Dim RightMargin As Long, BottomMargin As Long
   Dim fr As FormatRange
   Dim rcDrawTo As RECT
   Dim rcPage As RECT
   Dim TextLength As Long
   Dim NextCharPosition As Long
   Dim r As Long

   ' Start a print job to get a valid Printer.hDC
321:    Printer.Print Space(1)
322:    Printer.ScaleMode = vbTwips

   ' Get the offsett to the printable area on the page in twips
325:    LeftOffset = Printer.ScaleX(GetDeviceCaps(Printer.hdc, _
      PHYSICALOFFSETX), vbPixels, vbTwips)
327:    TopOffset = Printer.ScaleY(GetDeviceCaps(Printer.hdc, _
      PHYSICALOFFSETY), vbPixels, vbTwips)

   ' Calculate the Left, Top, Right, and Bottom margins
331:    LeftMargin = LeftMarginWidth - LeftOffset
332:    TopMargin = TopMarginHeight - TopOffset
333:    RightMargin = (Printer.Width - RightMarginWidth) - LeftOffset
334:    BottomMargin = (Printer.Height - BottomMarginHeight) - TopOffset

   ' Set printable area rect
337:    rcPage.Left = 0
338:    rcPage.Top = 0
339:    rcPage.Right = Printer.ScaleWidth
340:    rcPage.Bottom = Printer.ScaleHeight

   ' Set rect in which to print (relative to printable area)
343:    rcDrawTo.Left = LeftMargin
344:    rcDrawTo.Top = TopMargin
```

```
345:    rcDrawTo.Right = RightMargin
346:    rcDrawTo.Bottom = BottomMargin

   ' Set up the print instructions
349:    fr.hdc = Printer.hdc   ' Use the same DC for measuring and rendering
350:    fr.hdcTarget = Printer.hdc  ' Point at printer hDC
351:    fr.rc = rcDrawTo            ' Indicate the area on page to draw to
352:    fr.rcPage = rcPage          ' Indicate entire size of page
353:    fr.chrg.cpMin = 0           ' Indicate start of text through
354:    fr.chrg.cpMax = -1          ' end of the text

   ' Get length of text in RTF
357:    TextLength = Len(RTF.Text)

   ' Loop printing each page until done
360:    Do
     ' Print the page by sending EM_FORMATRANGE message
362:       NextCharPosition = SendMessage(RTF.hWnd, EM_FORMATRANGE, True, fr)
363:       If NextCharPosition >= TextLength Then Exit Do  'If done then exit
364:       fr.chrg.cpMin = NextCharPosition ' Starting position for next page
365:       Printer.NewPage                  ' Move on to next page
366:       Printer.Print Space(1) ' Re-initialize hDC
367:       fr.hdc = Printer.hdc
368:       fr.hdcTarget = Printer.hdc
369:    Loop

   ' Commit the print job
372:    Printer.EndDoc

   ' Allow the RTF to free up memory
375:    r = SendMessage(RTF.hWnd, EM_FORMATRANGE, False, ByVal CLng(0))

  Exit Sub
ErrorHandler:
  HandleError True, "PrintRTF " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub
```

Form 11: frmSelScreen.frm

```
VERSION 5.00
Begin VB.Form frmSelScreen
   Caption         =   "Form1"
   ClientHeight    =   4410
   ClientLeft      =   60
   ClientTop       =   345
```

```
ClientWidth     =   5715
BeginProperty Font
   Name            =   "Times New Roman"
   Size            =   8.25
   Charset         =   0
   Weight          =   400
   Underline       =   0   'False
   Italic          =   0   'False
   Strikethrough   =   0   'False
EndProperty
Icon            =   "frmSelScreen.frx":0000
LinkTopic       =   "Form1"
LockControls    =   -1  'True
ScaleHeight     =   4410
ScaleWidth      =   5715
StartUpPosition =   1  'CenterOwner
Begin VB.CommandButton cmdCancel
   Caption         =   "Cancel"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   345
   Left            =   4695
   TabIndex        =   5
   Top             =   955
   Width           =   945
End
Begin VB.CommandButton cmdHelp
   Caption         =   "Help"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   345
   Left            =   4695
   TabIndex        =   4
```

```
   Top             =   1340
   Width           =   945
End
Begin VB.CommandButton cmdAccept
   Caption         =   "Accept"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   345
   Left            =   4695
   TabIndex        =   2
   Top             =   1725
   Width           =   945
End
Begin VB.ListBox lbxThemes
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   2010
   Left            =   75
   TabIndex        =   1
   Top             =   60
   Width           =   4485
End
Begin VB.TextBox txtCoords
   BackColor       =   &H8000000F&
   BeginProperty Font
      Name            =   "Microsoft Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
```

```
   Height          =   1755
   Left            =   195
   MultiLine       =   -1  'True
   TabIndex        =   0
   Text            =   "frmSelScreen.frx":038A
   Top             =   2250
   Width           =   4800
End
Begin VB.Image imgDrawDisable
   Height          =   360
   Left            =   5145
   Picture         =   "frmSelScreen.frx":039A
   ToolTipText     =   "Draw Corridor Polygon..."
   Top             =   3255
   Width           =   360
End
Begin VB.Image imgDrawEnable
   Height          =   360
   Left            =   5145
   Picture         =   "frmSelScreen.frx":0A9E
   ToolTipText     =   "Draw Corridor Polygon..."
   Top             =   3255
   Width           =   360
End
Begin VB.Image imgDrawClick
   Height          =   360
   Left            =   5145
   Picture         =   "frmSelScreen.frx":11A2
   ToolTipText     =   "Draw Corridor Polygon..."
   Top             =   3255
   Width           =   360
End
Begin VB.Label lblLink
   AutoSize        =   -1  'True
   Caption         =   "http://www.corridordesign.org"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00FF0000&
   Height          =   195
   Left            =   1785
```

```
      MousePointer    =   99  'Custom
      TabIndex        =   3
      Top             =   4185
      Width           =   2130
   End
   Begin VB.Image imgIcon
      Height          =   855
      Left            =   4695
      Picture         =   "frmSelScreen.frx":18A6
      Top             =   60
      Width           =   945
   End
   Begin VB.Image imgToolDisable
      Height          =   360
      Left            =   5145
      Picture         =   "frmSelScreen.frx":43AA
      ToolTipText     =   "Click to Select Polygon..."
      Top             =   2820
      Width           =   360
   End
   Begin VB.Image imgToolOut
      Height          =   360
      Left            =   5145
      Picture         =   "frmSelScreen.frx":4AAE
      ToolTipText     =   "Click to Select Polygon..."
      Top             =   2820
      Width           =   360
   End
   Begin VB.Image imgToolIn
      Height          =   360
      Left            =   5145
      Picture         =   "frmSelScreen.frx":51B2
      Top             =   2820
      Width           =   360
   End
   Begin VB.Image imgBoundaryBox
      BorderStyle     =   1  'Fixed Single
      Height          =   2010
      Left            =   60
      Top             =   2145
      Width           =   5610
   End
End
Attribute VB_Name = "frmSelScreen"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
```

```
Attribute VB_Exposed = False
Option Explicit

Private m_pMxDoc As esriArcMapUI.IMxDocument
'Private m_Command As esriSystemUI.ICommand
Private m_pApp As IApplication
Private m_Frame As IModelessFrame
'Private m_WindowPos As IWindowPosition
Private m_ExtensionConfig As IExtensionConfig
Private m_SearchMessage As String
Private m_ToolEnable As Boolean
Private m_DrawToolEnable As Boolean
Private m_strNameArray() As String
Private m_intNameCount As Integer
Private m_colPolygons As Collection
Private m_PolygonPurpose As String
Private m_pPolygon As IPolygon

Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private Declare Function ShellExecute Lib "shell32.dll" Alias _
           "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As String, _
           ByVal lpFile As String, ByVal lpParameters As String, _
           ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Private Anchors As AnchorObjectList  ' Main anchor control object

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\frmSelScreen.frm"

Public Property Set thePolygon(pPolygon As IPolygon)
  On Error GoTo ErrorHandler

35:   Set m_pPolygon = pPolygon

  Exit Property
ErrorHandler:
  HandleError True, "thePolygon " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Public Property Let PolygonPurpose(strPurpose As String)
  On Error GoTo ErrorHandler

45:   m_PolygonPurpose = strPurpose
```

```
  Exit Property
ErrorHandler:
  HandleError True, "PolygonPurpose " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let NameList(strNameArray() As String)
  On Error GoTo ErrorHandler

54:   m_strNameArray = strNameArray

  Exit Property
ErrorHandler:
  HandleError True, "NameList " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property
Public Property Set NameCollection(colNames As Collection)
  On Error GoTo ErrorHandler

63:   Set m_colPolygons = colNames

  Exit Property
ErrorHandler:
  HandleError True, "NameCollection " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get GetNameCollection() As Collection
  On Error GoTo ErrorHandler

72:   Set GetNameCollection = m_colPolygons

  Exit Property
ErrorHandler:
  HandleError True, "GetNameCollection " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let NameCount(intNameCount As Long)
  On Error GoTo ErrorHandler

81:   m_intNameCount = intNameCount

  Exit Property
ErrorHandler:
  HandleError True, "NameCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Let EnableTool(booEnable As Boolean)
  On Error GoTo ErrorHandler
```

```
91:   m_ToolEnable = booEnable

  Exit Property
ErrorHandler:
  HandleError True, "EnableTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property


Public Property Let SearchMessage(strMessage As String)
  On Error GoTo ErrorHandler


103:   m_SearchMessage = strMessage


  Exit Property
ErrorHandler:
  HandleError True, "SearchMessage " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set ArcApplication(ByVal theApplication As IApplication)
  On Error GoTo ErrorHandler

114:   Set m_pApp = theApplication

  Exit Property
ErrorHandler:
  HandleError True, "ArcApplication " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Function Frame() As IModelessFrame
  On Error GoTo ErrorHandler

123:   If m_Frame Is Nothing Then
124:     Set m_Frame = New ModelessFrame
125:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
128:   End If

130:   Set Frame = m_Frame

  Exit Function
ErrorHandler:
```

```
  HandleError True, "Frame " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
Public Property Set Doc(pDoc As esriArcMapUI.IMxDocument)
  On Error GoTo ErrorHandler

139:   Set m_pMxDoc = pDoc

  Exit Property
ErrorHandler:
  HandleError True, "Doc " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Private Sub cmdAccept_Click()
  On Error GoTo ErrorHandler


150:   If m_pPolygon Is Nothing Then
151:     MsgBox "Error identifying selected polygon!  Please click on desired polygon again and try this button one more time...", _
       vbOKOnly, "Unexpected Problem:"
    Exit Sub
154:   End If

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
158:   pUID.Value = "Linkages.toolReturnCoords"

  Dim ext As Linkages.Extension
161:   Set ext = m_ExtensionConfig
162:   ext.EnableSelTool = False
163:   ext.EnableDrawTool = False
164:   ext.CorrPolygonIsDrawn = lbxThemes.ListIndex = 0

  Select Case m_PolygonPurpose
    Case "Wildland1", "Wildland2", "Corridor"
      Dim pStep1Form As Linkages.Jennessent_CompareParameters
169:       Set pStep1Form = ext.frmStep1
    Case "Bottleneck_Wildland1", "Bottleneck_Wildland2", "Bottleneck_Corridor"
      Dim pBottleneckForm As Linkages.frmBottleneck
172:       Set pBottleneckForm = ext.BottleneckForm
    Case "Clip"
      Dim pClipForm As Linkages.frmClip
175:       Set pClipForm = ext.frmClipForm
176:   End Select

  Dim pTopoOp2 As ITopologicalOperator2
179:   Set pTopoOp2 = m_pPolygon
180:   If Not pTopoOp2.IsSimple Then pTopoOp2.Simplify
```

```
' MsgBox "Spatial Reference Name = " & (m_pPolygon.SpatialReference.Name)
  Select Case m_PolygonPurpose
    Case "Wildland1"
184:       Set ext.PolyWildland1 = m_pPolygon
185:       pStep1Form.m_lngWB1Count = 1
    Case "Wildland2"
187:       Set ext.PolyWildland2 = m_pPolygon
188:       pStep1Form.m_lngWB2Count = 1
    Case "Corridor"
190:       Set ext.PolyCorridor = m_pPolygon
191:       pStep1Form.m_lngCorrCount = 1
    Case "Bottleneck_Wildland1"
193:       Set ext.PolyWildland1 = m_pPolygon
194:       pBottleneckForm.m_lngWB1Count = 1
    Case "Bottleneck_Wildland2"
196:       Set ext.PolyWildland2 = m_pPolygon
197:       pBottleneckForm.m_lngWB2Count = 1
    Case "Bottleneck_Corridor"
199:       Set ext.PolyCorridor = m_pPolygon
200:       pBottleneckForm.m_lngCorrCount = 1
    Case "Clip"
202:       Set ext.PolyCorridor = m_pPolygon
203:       pClipForm.m_lngCorrCount = 1
204:   End Select

206:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
207:   Set m_pApp.CurrentTool = Nothing

  Dim pCommandItem As ICommandItem
210:   Set pCommandItem = pCommand

212:   pCommandItem.Refresh

214:   Set ext.aSelForm = Nothing

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
217:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")

  ' UPDATE CALLING FORM
'  If m_PolygonPurpose = "Clip" Then
'    Call pClipForm.UpdateCheckmarks
'    Call pClipForm.EnableOKButton
'    pClipForm.Frame.Visible = True
'  ElseIf Not pStep1Form Is Nothing Then
'    Call pStep1Form.UpdateCheckmarks
'    Call pStep1Form.EnableOKButton
'  End If
```

```
  Select Case m_PolygonPurpose
    Case "Wildland1", "Wildland2", "Corridor"
231:       Call pStep1Form.UpdateCheckmarks
232:       Call pStep1Form.EnableOKButton
233:       pStep1Form.Frame.Visible = True
    Case "Bottleneck_Wildland1", "Bottleneck_Wildland2", "Bottleneck_Corridor"
235:       Call pBottleneckForm.UpdateCheckmarks
236:       Call pBottleneckForm.EnableOKButton
237:       pBottleneckForm.Frame.Visible = True
    Case "Clip"
239:       Call pClipForm.UpdateCheckmarks
240:       Call pClipForm.EnableOKButton
241:       pClipForm.Frame.Visible = True
242:   End Select

244:   Unload Me



  Exit Sub
ErrorHandler:
  HandleError True, "cmdAccept_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler

  Dim newUid As New uID
258:   newUid.Value = "Linkages.Extension"
259:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)

  Dim ext As Linkages.Extension
262:   Set ext = m_ExtensionConfig

  Select Case m_PolygonPurpose
    Case "Wildland1", "Wildland2", "Corridor"
      Dim pStep1Form As Linkages.Jennessent_CompareParameters
267:       Set pStep1Form = ext.frmStep1
268:       Call pStep1Form.UpdateCheckmarks
269:       Call pStep1Form.EnableOKButton
270:       pStep1Form.Frame.Visible = True
    Case "Bottleneck_Wildland1", "Bottleneck_Wildland2", "Bottleneck_Corridor"
      Dim pBottleneckForm As Linkages.frmBottleneck
273:       Set pBottleneckForm = ext.BottleneckForm
```

```
274:       Call pBottleneckForm.UpdateCheckmarks
275:       Call pBottleneckForm.EnableOKButton
276:       pBottleneckForm.Frame.Visible = True
    Case "Clip"
      Dim pClipForm As Linkages.frmClip
279:       Set pClipForm = ext.frmClipForm
280:       Call pClipForm.UpdateCheckmarks
281:       Call pClipForm.EnableOKButton
282:       pClipForm.Frame.Visible = True
283:   End Select

285:   Unload Me

  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdHelp_Click()
  On Error GoTo ErrorHandler

  Dim strPath As String
296:   strPath = App.Path & "\help"

298:   Call Linkages.MyGeneralOperations.OpenDoc("Select_Tool_Subdocument.pdf", strPath)


  '  MsgBox App.Path & vbCrLf & App.EXEName & vbCrLf & App.FileDescription


  Exit Sub
ErrorHandler:
  HandleError True, "cmdHelp_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Initialize()
  On Error GoTo ErrorHandler

'  Set m_Command = New Linkages.toolReturnCoords


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Sub

Private Sub Text1_Change()
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
  HandleError False, "Text1_Change " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler

333:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2

335:   Me.Left = (Screen.Width / 3) - (Me.Width / 2)
336:   Me.Top = (Screen.Height / 2) - (Me.Height / 2)

338:   If m_Frame Is Nothing Then
339:     Set m_Frame = New ModelessFrame
340:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
343:   End If

'235:    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height& Vbcrlf & _
'        cmdhelp.Container & "   x   " & cmdhelp.Container.height
  ' ANCHORS FOR RESIZE FUNCTIONS
348:   Set Anchors = New AnchorObjectList  ' Create new instance
349:    With Anchors
350:       With .Item(cmdAccept)
351:          .SetAnchors enumSizeEnd, enumStartSize
'         Set .WindowPos = m_WindowPos
353:       End With
354:       With .Item(imgBoundaryBox)
355:          .SetAnchors enumStartEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
357:       End With
358:       With .Item(cmdHelp)
359:          .SetAnchors enumSizeEnd, enumStartSize
'         Set .WindowPos = m_WindowPos
361:       End With
362:       With .Item(cmdCancel)
363:          .SetAnchors enumSizeEnd, enumStartSize
```

```
'         Set .WindowPos = m_WindowPos
365:       End With
366:       With .Item(imgIcon)
367:          .SetAnchors enumSizeEnd, enumStartSize
'         Set .WindowPos = m_WindowPos
369:       End With
370:       With .Item(imgToolDisable)
371:          .SetAnchors enumSizeEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
373:       End With
374:       With .Item(imgToolIn)
375:          .SetAnchors enumSizeEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
377:       End With
378:       With .Item(imgToolOut)
379:          .SetAnchors enumSizeEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
381:       End With
382:       With .Item(imgDrawClick)
383:          .SetAnchors enumSizeEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
385:       End With
386:       With .Item(imgDrawDisable)
387:          .SetAnchors enumSizeEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
389:       End With
390:       With .Item(imgDrawEnable)
391:          .SetAnchors enumSizeEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
393:       End With
394:       With .Item(lbxThemes)
395:          .SetAnchors enumStartEnd, enumStartEnd
'         Set .WindowPos = m_WindowPos
397:       End With
398:       With .Item(lblLink)
399:          .SetAnchors enumSize, enumSizeEnd
'         Set .WindowPos = m_WindowPos
401:       End With
402:       With .Item(txtCoords)
403:          .SetAnchors enumStartEnd, enumSizeEnd
'         Set .WindowPos = m_WindowPos
405:       End With
406:       .Form = Me  ' Set form reference (suggested to be last step)
407:    End With

  Dim newUid As New uID
410:   newUid.Value = "Linkages.Extension"
```

```
411:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)

  Dim ext As Linkages.Extension
414:   Set ext = m_ExtensionConfig
415:   Set ext.aSelForm = Me

417:   m_ToolEnable = False

  Dim anIndex As Integer
420:   For anIndex = 0 To m_intNameCount
421:     lbxThemes.AddItem m_strNameArray(anIndex)
422:   Next anIndex

  Dim strInstructions As String
425:   strInstructions = _
    "INSTRUCTIONS:" & vbCrLf & _
    "1) Select the appropriate layer from the list above." & vbCrLf & _
    "2) Enable the Polygon Selection tool by clicking the crosshair button ---->" & vbCrLf & _
    "3) Click on the map to select your " & m_SearchMessage & " polygon." & vbCrLf & _
    "4) Click 'Accept'."

432:   txtCoords.Text = strInstructions
433:   txtCoords.Locked = True

435:   lblLink.MouseIcon = LoadResPicture(102, vbResCursor)

437:   cmdAccept.Enabled = False

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
440:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")


  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub


Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler

'  Set m_Command = Nothing

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
455:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")

  Dim pCommand As esriSystemUI.ICommand
```

```
  Dim pUID As New uID
459:   pUID.Value = "Linkages.toolReturnCoords"

  Dim ext As Linkages.Extension
462:   Set ext = m_ExtensionConfig
463:   ext.EnableSelTool = False
464:   ext.EnableDrawTool = False

'  ' SET ORIGINAL FORM BACK TO VISIBLE
'  If Not ext.frmStep1 Is Nothing Then
'    Dim pForm As Linkages.Jennessent_CompareParameters
'    Set pForm = ext.frmStep1
'    pForm.Frame.Visible = True
'  End If

473:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
474:   Set m_pApp.CurrentTool = Nothing

  Dim pCommandItem As ICommandItem
477:   Set pCommandItem = pCommand

479:   pCommandItem.Refresh

481:   Set ext.aSelForm = Nothing

483:   Set pCommand = Nothing
484:   Set pCommandItem = Nothing
485:   Set ext = Nothing
486:   Set m_pApp = Nothing
487:   Set m_pMxDoc = Nothing
488:   Set m_Frame = Nothing
489:   Set m_ExtensionConfig = Nothing
490:   Set m_colPolygons = Nothing
491:   Set m_pPolygon = Nothing
'  Set m_WindowPos = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub
Private Sub imgDrawClick_Click()
  On Error GoTo ErrorHandler

501:   m_DrawToolEnable = False
502:   m_ToolEnable = False
503:   Call SetSelToolEnabled
```

```
  Dim ext As Linkages.Extension
506:   Set ext = m_ExtensionConfig
507:   ext.EnableSelTool = False
508:   ext.EnableDrawTool = False

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
512:   pUID.Value = "Linkages.toolDrawPoly"

514:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
515:   Set m_pApp.CurrentTool = Nothing

  Dim pCommandItem As ICommandItem
518:   Set pCommandItem = pCommand

520:   pCommandItem.Refresh


  Exit Sub
ErrorHandler:
  HandleError True, "imgDrawClick_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub imgDrawEnable_Click()
  On Error GoTo ErrorHandler

531:   m_DrawToolEnable = True
532:   m_ToolEnable = False
533:   Call SetToolEnabledDraw
534:   Call ClickOnScreenDraw

  Exit Sub
ErrorHandler:
  HandleError True, "imgDrawEnable_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Sub ClickOnScreenDraw()
On Error GoTo ErrorHandler

543:   If (m_ExtensionConfig Is Nothing) Then
544:     MsgBox "Unexpected error!  'm_ExtensionConfig' set to nothing..."
    Exit Sub
546:   End If

  Dim ext As Linkages.Extension
```

```
549:   Set ext = m_ExtensionConfig

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
553:   pUID.Value = "Linkages.toolDrawPoly"

555:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
556:   ext.EnableSelTool = False
557:   ext.EnableDrawTool = True

  Dim pCommandItem As ICommandItem
560:   Set pCommandItem = pCommand

562:   pCommandItem.Refresh

564:   Set m_pApp.CurrentTool = pCommandItem

  Exit Sub
ErrorHandler:
  HandleError False, "ClickOnScreenDraw " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Sub SetToolEnabledDraw()
  On Error GoTo ErrorHandler


'  MsgBox m_ToolEnable & vbCrLf & "Starting SetToolEnabled Function" & vbCrLf & lbxThemes.ListIndex

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
577:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")
578:   cmdAccept.Enabled = False

580:   If (lbxThemes.ListIndex = -1) Then  ' SET DISABLED IMAGE FOR BOTH TOOLS IF NOTHING SELECTED IN LIST
581:     imgToolOut.Visible = False
582:     imgToolIn.Visible = False
583:     imgToolDisable.Visible = True
584:     imgDrawEnable.Visible = False
585:     imgDrawClick.Visible = False
586:     imgDrawDisable.Visible = True
587:   Else
588:     imgToolOut.Visible = True              ' SELECT TOOL SHOULD BE ENABLED BUT NOT CLICKED
589:     imgToolIn.Visible = False
590:     imgToolDisable.Visible = False
591:     If (lbxThemes.ListIndex = 0) Then
592:       If (m_DrawToolEnable) Then             ' IF DRAWING TOOL CAN BE ENABLED AND CLICKED, SET IT TO ENABLED
593:         imgDrawClick.Visible = True
594:         imgDrawEnable.Visible = False
```

```
595:         imgDrawDisable.Visible = False
596:       Else
597:         imgDrawClick.Visible = False
598:         imgDrawEnable.Visible = True
599:         imgDrawDisable.Visible = False
600:       End If
601:     Else
602:       imgDrawEnable.Visible = False
603:       imgDrawClick.Visible = False
604:       imgDrawDisable.Visible = True
605:     End If
606:   End If

608:   Me.Refresh


  Exit Sub
ErrorHandler:
  HandleError True, "SetToolEnabledDraw " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Sub SetSelToolEnabled()
  On Error GoTo ErrorHandler


'  MsgBox m_ToolEnable & vbCrLf & "Starting SetSelToolEnabled Function" & vbCrLf & lbxThemes.ListIndex

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
622:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")
623:   cmdAccept.Enabled = False

625:   If (lbxThemes.ListIndex = -1) Then
626:     imgToolOut.Visible = False
627:     imgToolIn.Visible = False
628:     imgToolDisable.Visible = True
629:     imgDrawEnable.Visible = False
630:     imgDrawClick.Visible = False
631:     imgDrawDisable.Visible = True
632:   Else
633:     If (lbxThemes.ListIndex = 0) Then
634:       imgDrawClick.Visible = False            ' DRAW TOOL SHOULD BE ENABLED BUT NOT CLICKED, BUT ONLY IF
                                                ' FIRST ITEM IS SELECTED IN LISTBOX
636:       imgDrawEnable.Visible = True
637:       imgDrawDisable.Visible = False
638:     Else
639:       imgDrawClick.Visible = False            ' DRAW TOOL SHOULD BE ENABLED BUT NOT CLICKED, BUT ONLY IF
                                                ' FIRST ITEM IS SELECTED IN LISTBOX
```

```
641:       imgDrawEnable.Visible = False
642:       imgDrawDisable.Visible = True

644:     End If
645:     If (m_ToolEnable) Then
646:       imgToolOut.Visible = False
647:       imgToolIn.Visible = True
648:       imgToolDisable.Visible = False
649:     Else
650:       imgToolOut.Visible = True
651:       imgToolIn.Visible = False
652:       imgToolDisable.Visible = False
653:     End If
654:   End If

656:   Me.Refresh


  Exit Sub
ErrorHandler:
  HandleError True, "SetSelToolEnabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ClickOnScreen()
On Error GoTo ErrorHandler

667:   If (m_ExtensionConfig Is Nothing) Then
668:     MsgBox "Unexpected error!  'm_ExtensionConfig' set to nothing..."
    Exit Sub
670:   End If


  Dim ext As Linkages.Extension
674:   Set ext = m_ExtensionConfig

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
678:   pUID.Value = "Linkages.toolReturnCoords"

680:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
681:   ext.EnableSelTool = True
682:   ext.EnableDrawTool = False

  Dim pCommandItem As ICommandItem
685:   Set pCommandItem = pCommand
```

```
687:   pCommandItem.Refresh

689:   Set m_pApp.CurrentTool = pCommandItem

  Exit Sub
ErrorHandler:
  HandleError False, "ClickOnScreen " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Sub imgToolIn_Click()
  On Error GoTo ErrorHandler

698:   m_ToolEnable = False
699:   m_DrawToolEnable = False
700:   Call SetSelToolEnabled

  Dim ext As Linkages.Extension
703:   Set ext = m_ExtensionConfig
704:   ext.EnableSelTool = False
705:   ext.EnableDrawTool = False

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
709:   pUID.Value = "Linkages.toolReturnCoords"

711:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
712:   Set m_pApp.CurrentTool = Nothing

  Dim pCommandItem As ICommandItem
715:   Set pCommandItem = pCommand

717:   pCommandItem.Refresh


  Exit Sub
ErrorHandler:
  HandleError True, "imgToolIn_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub imgToolOut_Click()
  On Error GoTo ErrorHandler

728:   m_ToolEnable = True
729:   m_DrawToolEnable = False
730:   Call SetSelToolEnabled
731:   Call ClickOnScreen
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "imgToolOut_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Private Sub lblLink_Click()
  On Error GoTo ErrorHandler

742:   Call ShellExecute(0, vbNullString, "http://www.corridordesign.org/", vbNullString, "", 1)

  Exit Sub
ErrorHandler:
  HandleError True, "lblLink_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  On Error GoTo ErrorHandler


753:   lblLink.ForeColor = vbBlue


  Exit Sub
ErrorHandler:
  HandleError True, "Form_MouseMove " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lblLink_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  On Error GoTo ErrorHandler

764:   lblLink.ForeColor = vbRed

  Exit Sub
ErrorHandler:
  HandleError True, "lblLink_MouseMove " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbxThemes_Click()
  On Error GoTo ErrorHandler

774:   m_ToolEnable = False
```

```
775:   m_DrawToolEnable = False

  Dim strInstructions As String

779:   If lbxThemes.ListIndex > 0 Then
780:     strInstructions = _
      "INSTRUCTIONS:" & vbCrLf & _
      "1) Select the appropriate layer from the list above." & vbCrLf & _
      "2) Enable the Polygon Selection tool by clicking the crosshair button ---->" & vbCrLf & _
      "3) Click on the map to select your " & m_SearchMessage & " polygon." & vbCrLf & _
      "4) Click 'Accept'."
786:   Else
787:       strInstructions = _
      "INSTRUCTIONS:" & vbCrLf & _
      "1) Select the appropriate layer from the list above." & vbCrLf & _
      "2) To SELECT an existing graphic, enable the Polygon Selection tool by clicking the crosshair button " & _
      "on the upper right.  Then click on the map to select your " & m_SearchMessage & " polygon." & vbCrLf & _
      "3) To DRAW a new graphic, click the Polygon Draw tool on the lower right and use it to draw your " & _
       m_SearchMessage & " polygon. " & vbCrLf & _
      "4) Click 'Accept'."
795:   End If

797:   txtCoords.Text = strInstructions
798:   txtCoords.Locked = True

800:   Call SetSelToolEnabled

  Exit Sub
ErrorHandler:
  HandleError True, "lbxThemes_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 12: frmStep2.frm

```
VERSION 5.00
Begin VB.Form frmStep2
   Caption         =   "Form1"
   ClientHeight    =   3195
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   4680
   LinkTopic       =   "Form1"
   ScaleHeight     =   3195
   ScaleWidth      =   4680
```

```
   StartUpPosition =   3  'Windows Default
End
Attribute VB_Name = "frmStep2"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

Form 13: jennessent_compareparameters.frm

```
VERSION 5.00
Object = "{3B7C8863-D78F-101B-B9B5-04021C009402}#1.2#0"; "RICHTX32.OCX"
Begin VB.Form Jennessent_CompareParameters
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Describe Patch-to-Patch Distances:"
   ClientHeight    =   4470
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   9945
   Icon            =   "jennessent_compareparameters.frx":0000
   LinkTopic       =   "Jennessent_CompareParameters"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   4470
   ScaleWidth      =   9945
   StartUpPosition =   2  'CenterScreen
   Begin VB.CommandButton cmdManual
      Caption         =   "Open Manual"
      Height          =   345
      Left            =   495
      TabIndex        =   26
      Top             =   4125
      Width           =   1335
   End
   Begin RichTextLib.RichTextBox rtbHelp
      Height          =   2190
      Left            =   5760
      TabIndex        =   22
      Top             =   45
      Width           =   4155
      _ExtentX        =   7329
      _ExtentY        =   3863
      _Version        =   393217
      Enabled         =   -1  'True
```

```
   ScrollBars      =   2
   TextRTF         =   $"jennessent_compareparameters.frx":038A
End
Begin VB.CheckBox chkUsePatches
   Height          =   240
   Left            =   195
   TabIndex        =   6
   Top             =   1740
   Width           =   255
End
Begin VB.ComboBox cbxPatchLayer
   Height          =   315
   Left            =   2415
   TabIndex        =   7
   Text            =   "Patch Polygon Layer:"
   Top             =   2160
   Width           =   2790
End
Begin VB.OptionButton optSubSet
   Height          =   270
   Left            =   720
   TabIndex        =   9
   Top             =   2820
   Width           =   270
End
Begin VB.OptionButton optUseAll
   Height          =   270
   Left            =   720
   TabIndex        =   8
   Top             =   2535
   Width           =   270
End
Begin VB.TextBox txtValue
   Height          =   300
   Left            =   3360
   TabIndex        =   12
   Text            =   "txtValue"
   Top             =   3555
   Width           =   1845
End
Begin VB.ComboBox cbxValue
   Height          =   315
   Left            =   2490
   TabIndex        =   11
   Text            =   "Attribute Value "
   Top             =   3540
   Width           =   750
```

```
End
Begin VB.ComboBox cbxPatchField
   Height          =   315
   Left            =   2910
   TabIndex        =   10
   Text            =   "Patch Attribute Field:"
   Top             =   3165
   Width           =   2295
End
Begin VB.CommandButton cmdSelLink
   Caption         =   "Select"
   Height          =   255
   Left            =   4830
   TabIndex        =   5
   Top             =   1185
   Width           =   750
End
Begin VB.ComboBox cbxCorridor
   Height          =   315
   Left            =   2325
   Style           =   2  'Dropdown List
   TabIndex        =   4
   Top             =   1155
   Width           =   2400
End
Begin VB.CommandButton cmdSel2
   Caption         =   "Select"
   Height          =   255
   Left            =   4830
   TabIndex        =   3
   Top             =   720
   Width           =   750
End
Begin VB.ComboBox cbxHab2
   Height          =   315
   Left            =   1875
   Style           =   2  'Dropdown List
   TabIndex        =   2
   Top             =   690
   Width           =   2850
End
Begin VB.CommandButton cmdSel1
   Caption         =   "Select"
   Height          =   255
   Left            =   4830
   TabIndex        =   1
   Top             =   255
```

```
   Width           =   750
End
Begin VB.ComboBox cbxHab1
   Height          =   315
   Left            =   1875
   Style           =   2  'Dropdown List
   TabIndex        =   0
   Top             =   225
   Width           =   2850
End
Begin VB.CommandButton cmdHelp
   Caption         =   "Show Help >>"
   Height          =   345
   Left            =   3945
   TabIndex        =   15
   Top             =   4125
   Width           =   1275
End
Begin VB.CommandButton cmdOK
   Caption         =   "OK"
   Height          =   345
   Left            =   2835
   TabIndex        =   14
   Top             =   4125
   Width           =   1095
End
Begin VB.CommandButton cmdCancel
   Caption         =   "Cancel"
   Height          =   345
   Left            =   1845
   TabIndex        =   13
   Top             =   4125
   Width           =   975
End
Begin VB.Image imgHelpPatches2
   Height          =   2190
   Left            =   5760
   Picture         =   "jennessent_compareparameters.frx":0415
   Top             =   2250
   Width           =   4155
End
Begin VB.Image imgUnCheckSpCorr
   Height          =   375
   Left            =   90
   Picture         =   "jennessent_compareparameters.frx":1DED9
   Top             =   1110
   Width           =   375
```

```
End
Begin VB.Image imgUnCheckWB2
   Height          =   375
   Left            =   90
   Picture         =   "jennessent_compareparameters.frx":1DF4D
   Top             =   645
   Width           =   375
End
Begin VB.Image imgUnCheckWB1
   Height          =   375
   Left            =   90
   Picture         =   "jennessent_compareparameters.frx":1DFC1
   Top             =   180
   Width           =   375
End
Begin VB.Label Label4
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Include Patch Polygons"
   Height          =   195
   Left            =   495
   TabIndex        =   25
   Top             =   1755
   Width           =   1680
End
Begin VB.Label Label3
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Use Subset of Patch Polygons, where:"
   Height          =   195
   Left            =   1020
   TabIndex        =   24
   Top             =   2850
   Width           =   2730
End
Begin VB.Label Label2
   BackStyle       =   0  'Transparent
   Caption         =   "Use All Patch Polygons"
   Height          =   225
   Left            =   1020
   TabIndex        =   23
   Top             =   2565
   Width           =   1980
End
Begin VB.Image imgHelpPatches
   Height          =   2190
   Left            =   5760
```

```
   Picture         =   "jennessent_compareparameters.frx":1E035
   Top             =   2250
   Width           =   4155
End
Begin VB.Image imgHelpPolygons
   Height          =   2190
   Left            =   5760
   Picture         =   "jennessent_compareparameters.frx":3BAF9
   Top             =   2250
   Width           =   4155
End
Begin VB.Image imgCheckSpCorr
   Height          =   375
   Left            =   90
   Picture         =   "jennessent_compareparameters.frx":595BD
   Top             =   1110
   Width           =   375
End
Begin VB.Image imgCheckWB2
   Height          =   375
   Left            =   90
   Picture         =   "jennessent_compareparameters.frx":596D4
   Top             =   645
   Width           =   375
End
Begin VB.Image imgCheckWB1
   Height          =   375
   Left            =   90
   Picture         =   "jennessent_compareparameters.frx":597EB
   Top             =   180
   Width           =   375
End
Begin VB.Shape boxBottom
   Height          =   2490
   Left            =   45
   Top             =   1605
   Width           =   5670
End
Begin VB.Shape boxTop
   Height          =   1560
   Left            =   45
   Top             =   60
   Width           =   5670
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
```

```
   Caption         =   "Wildland Block #1:"
   Height          =   195
   Left            =   480
   TabIndex        =   21
   Top             =   255
   Width           =   1365
End
Begin VB.Label lbl_cbxValue
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Attribute Value "
   Height          =   195
   Left            =   1335
   TabIndex        =   20
   Top             =   3585
   Width           =   1080
End
Begin VB.Label lbl_cbxPatchField
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Patch Attribute Field:"
   Height          =   195
   Left            =   1320
   TabIndex        =   19
   Top             =   3210
   Width           =   1470
End
Begin VB.Label lbl_cbxPatchLayer
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Patch Polygon Layer:"
   Height          =   195
   Left            =   735
   TabIndex        =   18
   Top             =   2205
   Width           =   1515
End
Begin VB.Label lbl_cbxCorridor
   AutoSize        =   -1  'True
   BackStyle       =   0  'Transparent
   Caption         =   "Species Corridor Polygon:"
   Height          =   195
   Left            =   480
   TabIndex        =   17
   Top             =   1170
   Width           =   1830
End
```

```
   Begin VB.Label lbl_cbxHab2
      AutoSize        =   -1  'True
      BackStyle       =   0  'Transparent
      Caption         =   "Wildland Block #2:"
      Height          =   195
      Left            =   480
      TabIndex        =   16
      Top             =   720
      Width           =   1365
   End
   Begin VB.Image imgFrameBack
      Height          =   2100
      Left            =   150
      Picture         =   "jennessent_compareparameters.frx":59902
      Top             =   1935
      Width           =   5445
   End
End
Attribute VB_Name = "Jennessent_CompareParameters"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
' PUT IN GENERAL DECLARATIONS SECTION
Private Anchors As AnchorObjectList  ' Main anchor control object
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private m_MxDoc As esriArcMapUI.IMxDocument
Private m_pApp As IApplication
Private m_Frame As IModelessFrame
'Private m_WindowPos As IWindowPosition
Private m_ExtensionConfig As IExtensionConfig
Private m_strNameArray() As String
Private m_intNameCount As Integer
Private m_colPolygons As Collection
Private m_booHelpToggle As Boolean
Private m_IntHelpCategory As Integer
Private m_intValueIndex As Integer
Private m_booUsePatches As Integer
Private m_colFieldNames As Collection
Public m_lngWB1Count As Long
Public m_lngWB2Count As Long
Public m_lngCorrCount As Long
```

```
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\jennessent_compareparameters.frm"


Public Property Set ArcApplication(ByVal theApplication As IApplication)
  On Error GoTo ErrorHandler


33:   Set m_pApp = theApplication


  Exit Property
ErrorHandler:
  HandleError True, "ArcApplication " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Function Frame() As IModelessFrame
  On Error GoTo ErrorHandler

44:   If m_Frame Is Nothing Then
45:     Set m_Frame = New ModelessFrame
46:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
49:   End If

51:   Set Frame = m_Frame

  Exit Function
ErrorHandler:
  HandleError True, "Frame " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Public Property Set Doc(pDoc As esriArcMapUI.IMxDocument)
  On Error GoTo ErrorHandler

61:   Set m_MxDoc = pDoc

  Exit Property
ErrorHandler:
  HandleError True, "Doc " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Private Sub cbxCorridor_Click()
  On Error GoTo ErrorHandler
```

```
  Dim ext As Linkages.Extension
72:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
75:   lngIndex = cbxCorridor.ListIndex

77:   Set ext.PolyCorridor = Nothing

79:   If lngIndex >= 2 Then
    Dim strLayerName As String
81:     strLayerName = cbxCorridor.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
84:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pFeatureCursor As IFeatureCursor
87:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
90:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
93:     Set pGeometry = pFeature.ShapeCopy

95:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
98:       Set pPolygon = pGeometry

100:       Set ext.PolyCorridor = pPolygon
101:       m_lngCorrCount = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
102:     End If
103:   End If
104:   cbxCorridor.BackColor = vbWhite
105:   Call UpdateCheckmarks
106:   Call UpdateSelectButtons
107:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxCorridor_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxCorridor_GotFocus()
  On Error GoTo ErrorHandler
```

```
118:   m_IntHelpCategory = 1
119:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxCorridor_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab1_Click()
  On Error GoTo ErrorHandler
128:   m_IntHelpCategory = 1
129:   Call UpdateHelpScreen

  Dim ext As Linkages.Extension
132:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
135:   lngIndex = cbxHab1.ListIndex

137:   Set ext.PolyWildland1 = Nothing

139:   If lngIndex >= 2 Then
    Dim strLayerName As String
141:     strLayerName = cbxHab1.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
144:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pFeatureCursor As IFeatureCursor
147:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
150:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
153:     Set pGeometry = pFeature.ShapeCopy

155:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
158:       Set pPolygon = pGeometry

160:       Set ext.PolyWildland1 = pPolygon
161:       m_lngWB1Count = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
162:     End If
```

```
164:   End If
165:   cbxHab1.BackColor = vbWhite
166:   Call UpdateCheckmarks
167:   Call UpdateSelectButtons
168:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab1_GotFocus()
  On Error GoTo ErrorHandler

178:   m_IntHelpCategory = 1
179:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab1_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab2_Click()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
190:   Set ext = m_ExtensionConfig

  Dim lngIndex As Long
193:   lngIndex = cbxHab2.ListIndex

195:   Set ext.PolyWildland2 = Nothing

197:   If lngIndex >= 2 Then
    Dim strLayerName As String
199:     strLayerName = cbxHab2.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
202:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pFeatureCursor As IFeatureCursor
205:     Set pFeatureCursor = pFeatureLayer.Search(Nothing, True)

    Dim pFeature As IFeature
```

```
208:     Set pFeature = pFeatureCursor.NextFeature

    Dim pGeometry As IGeometry
211:     Set pGeometry = pFeature.ShapeCopy

213:     If TypeOf pGeometry Is IPolygon Then

      Dim pPolygon As IPolygon
216:       Set pPolygon = pGeometry

218:       Set ext.PolyWildland2 = pPolygon
219:       m_lngWB2Count = pFeatureLayer.FeatureClass.FeatureCount(Nothing)
220:     End If
221:   End If
222:   cbxHab2.BackColor = vbWhite
223:   Call UpdateCheckmarks
224:   Call UpdateSelectButtons
225:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxHab2_GotFocus()
  On Error GoTo ErrorHandler


236:   m_IntHelpCategory = 1
237:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxHab2_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxPatchField_Click()
  On Error GoTo ErrorHandler


248:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxPatchField_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Sub

Private Sub cbxPatchField_GotFocus()
  On Error GoTo ErrorHandler

258:   m_IntHelpCategory = 3
259:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxPatchField_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxPatchLayer_Click()
  On Error GoTo ErrorHandler

  Dim lngIndex As Long
270:   lngIndex = cbxPatchLayer.ListIndex

272:   cbxPatchField.Clear
273:   cbxPatchField.AddItem ("Select Attribute Field...")

'  MsgBox cbxPatchLayer.ListIndex

  ' GET POLYGON LAYER IF SELECTION INDEX GREATER THAN 1
278:   If lngIndex >= 1 Then

    Dim colFieldNames As Collection
281:     Set colFieldNames = New Collection

    Dim strLayerName As String
284:     strLayerName = cbxPatchLayer.List(lngIndex)

    Dim pFeatureLayer As IFeatureLayer
287:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)

    Dim pLayerFields As ILayerFields
290:     Set pLayerFields = pFeatureLayer

    Dim pFieldInfo As IFieldInfo
    Dim pField As IField
    Dim lngIndex2 As Long
    Dim strAlias As String
'    MsgBox pLayerFields.FieldCount & " fields..."
```

```
298:     For lngIndex2 = 0 To (pLayerFields.FieldCount - 1)
299:       Set pFieldInfo = pLayerFields.FieldInfo(lngIndex2)
300:       strAlias = pFieldInfo.Alias
301:       Set pField = pLayerFields.Field(lngIndex2)
'      MsgBox CStr(lngIndex2) & ":  Type = " & pField.Type & ",   Name = " & strAlias
303:       If pField.Type < 7 Then
304:         cbxPatchField.AddItem strAlias
305:         colFieldNames.Add lngIndex2, strAlias
306:       End If
307:     Next lngIndex2
308:     Set m_colFieldNames = colFieldNames
309:   End If

311:   cbxPatchField.ListIndex = 0
312:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "cbxPatchLayer_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxPatchLayer_GotFocus()
  On Error GoTo ErrorHandler

322:   m_IntHelpCategory = 2
323:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxPatchLayer_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cbxValue_GotFocus()
  On Error GoTo ErrorHandler

333:   m_IntHelpCategory = 3
334:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "cbxValue_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub chkUsePatches_Click()
```

```
  On Error GoTo ErrorHandler


345:   m_IntHelpCategory = 2
346:   Call UpdateHelpScreen
347:   Call EnablePatchOptions
348:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "chkUsePatches_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub EnablePatchOptions()
  On Error GoTo ErrorHandler

  Dim booUsePatches As Boolean
359:       booUsePatches = chkUsePatches.Value
  Dim booUseSelection As Boolean
361:       booUseSelection = optSubSet.Value
362:   lbl_cbxPatchLayer.Enabled = booUsePatches
363:   cbxPatchLayer.Enabled = booUsePatches
364:   optUseAll.Enabled = booUsePatches
365:   optSubSet.Enabled = booUsePatches
366:   Label2.Enabled = booUsePatches
367:   Label3.Enabled = booUsePatches
368:   lbl_cbxPatchField.Enabled = booUsePatches And booUseSelection
369:   cbxPatchField.Enabled = booUsePatches And booUseSelection
370:   lbl_cbxValue.Enabled = booUsePatches And booUseSelection
371:   cbxValue.Enabled = booUsePatches And booUseSelection
372:   txtValue.Enabled = booUsePatches And booUseSelection


  Exit Sub
ErrorHandler:
  HandleError False, "EnablePatchOptions " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub chkUsePatches_GotFocus()
  On Error GoTo ErrorHandler

383:   m_IntHelpCategory = 2
384:   Call UpdateHelpScreen

  Exit Sub
```

```
ErrorHandler:
  HandleError True, "chkUsePatches_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdCancel_Click()
  On Error GoTo ErrorHandler

  '  ORIGINAL AVENUE CODE
  '  ' Jennessent.CompareParametersCancel
  '
  '  self.GetDialog.SetModalResult(nil)
  '  self.GetDialog.Close
399:   Unload Me

  '

  Exit Sub
ErrorHandler:
  HandleError True, "cmdCancel_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdHelp_Click()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
412:   Set ext = m_ExtensionConfig
  Dim booHelpToggle As Boolean
414:   booHelpToggle = ext.HelpToggle1
  Dim pWindowPosition As IWindowPosition
416:   Set pWindowPosition = m_Frame

418:   If booHelpToggle = False Then
419:     booHelpToggle = True
420:     ext.HelpToggle1 = True
421:     cmdHelp.Caption = "<< Hide Help"
422:     pWindowPosition.Width = 669
423:   Else
424:     booHelpToggle = False
425:     ext.HelpToggle1 = False
426:     cmdHelp.Caption = "Show Help >>"
427:     pWindowPosition.Width = 390
428:   End If

430:   Call UpdateHelpScreen
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "cmdHelp_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub SetBackColorsWhite()
  On Error GoTo ErrorHandler

440:   cbxHab1.BackColor = vbWhite
441:   cbxHab2.BackColor = vbWhite
442:   cbxCorridor.BackColor = vbWhite

  Exit Sub
ErrorHandler:
  HandleError False, "SetBackColorsWhite " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdManual_Click()

  Dim strPath As String
452:   strPath = App.Path & "\help"

454:   Call Linkages.MyGeneralOperations.OpenDoc("Patches_Subdocument.pdf", strPath)
End Sub

Private Sub cmdOK_Click()
  On Error GoTo ErrorHandler

460:   Call SetBackColorsWhite

  Dim lngIndex As Long
  Dim strLayerName As String
  Dim strLayerName2 As String
  Dim pFeatureLayer As IFeatureLayer
  Dim pFeatureLayer2 As IFeatureLayer
  Dim pPatchLayer As IFeatureLayer
  Dim intHab1Index As Integer
  Dim intHab2Index As Integer
  Dim intCorrIndex As Integer
  Dim strHab1Name As String
  Dim strHab2Name As String
  Dim strCorrName As String

475:   intHab1Index = cbxHab1.ListIndex
476:   intHab2Index = cbxHab2.ListIndex
```

```
477:   intCorrIndex = cbxCorridor.ListIndex

  Dim pRelOp As IRelationalOperator
  Dim ext As Linkages.Extension
481:   Set ext = m_ExtensionConfig

  Dim pHabPoly1 As IPolygon
484:       Set pHabPoly1 = ext.PolyWildland1
  Dim pHabPoly2 As IPolygon
486:       Set pHabPoly2 = ext.PolyWildland2
  Dim pCorrPoly As IPolygon
488:       Set pCorrPoly = ext.PolyCorridor

  Dim boo1And2 As Boolean           '  SHOULD BE FALSE
  Dim boo1AndCorr As Boolean        '  SHOULD BE TRUE
  Dim boo2AndCorr As Boolean        '  SHOULD BE TRUE

  ' FIRST PUT ALL 3 POLYGONS INTO SAME SPATIAL REFERENCE AS pCorrPoly
  Dim pSpRefHab1 As ISpatialReference
496:   Set pSpRefHab1 = pHabPoly1.SpatialReference
  Dim pSpRefHab2 As ISpatialReference
498:   Set pSpRefHab2 = pHabPoly2.SpatialReference
  Dim pSpRefCorr As ISpatialReference
500:   Set pSpRefCorr = pCorrPoly.SpatialReference

502:   If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRefCorr, pSpRefHab1) Then
503:     pHabPoly1.Project pSpRefCorr
504:     Set ext.PolyWildland1 = pHabPoly1
505:   End If

507:   If Not Linkages.MyGeneralOperations.CompareSpatialReferences(pSpRefCorr, pSpRefHab2) Then
508:     pHabPoly2.Project pSpRefCorr
509:     Set ext.PolyWildland2 = pHabPoly2
510:   End If

512:   Set pRelOp = pHabPoly1
513:   boo1And2 = Not pRelOp.Disjoint(pHabPoly2) ' (pRelOp.Touches(pHabPoly2)) Or (pRelOp.Overlaps(pHabPoly2))
514:   boo1AndCorr = Not pRelOp.Disjoint(pCorrPoly) ' (pRelOp.Touches(pCorrPoly)) Or (pRelOp.Overlaps(pCorrPoly))

516:   Set pRelOp = pHabPoly2
517:   boo2AndCorr = Not pRelOp.Disjoint(pCorrPoly) ' (pRelOp.Touches(pCorrPoly)) Or (pRelOp.Overlaps(pCorrPoly))

'  MsgBox boo1And2 & vbCrLf & boo1AndCorr & vbCrLf & boo2AndCorr

  ' ERROR CHECKING CODE ----------------------------------------------------------------
  ' CHECK FOR THEMES WITH MULTIPLE POLYGONS
523:   If m_lngWB1Count = 0 Then
```

```
524:     cbxHab1.BackColor = vbYellow
525:     MsgBox "Unable to find a polygon for Wildland Block #1!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
      , vbOKOnly, "Error Found in Input Data:"
528:     cbxHab1.SetFocus
    Exit Sub
530:   ElseIf m_lngWB1Count > 1 Then
531:     cbxHab1.BackColor = vbYellow
532:     lngIndex = cbxHab1.ListIndex
533:     strLayerName = cbxHab1.List(lngIndex)
534:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
535:     MsgBox "Multiple polygons (" & CStr(m_lngWB1Count) & ") found in Wildland Block #1 Layer '" & _
       pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
      "Please use the ""Select"" button to select a single " & _
      "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
539:     cbxHab1.SetFocus
    Exit Sub
541:   ElseIf m_lngWB2Count = 0 Then
542:     cbxHab2.BackColor = vbYellow
543:     MsgBox "Unable to find a polygon for Wildland Block #2!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
      , vbOKOnly, "Error Found in Input Data:"
546:     cbxHab2.SetFocus
    Exit Sub
548:   ElseIf m_lngWB2Count > 1 Then
549:     cbxHab2.BackColor = vbYellow
550:     lngIndex = cbxHab2.ListIndex
551:     strLayerName = cbxHab2.List(lngIndex)
552:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
553:     MsgBox "Multiple polygons (" & CStr(m_lngWB2Count) & ") found in Wildland Block #2 Layer '" & _
       pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
       "Please use the ""Select"" button to select a single " & _
       "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
557:     cbxHab2.SetFocus
    Exit Sub
559:   ElseIf m_lngCorrCount = 0 Then
560:     cbxCorridor.BackColor = vbYellow
561:     MsgBox "Unable to find a polygon for the Species Corridor Polygon!" & vbCrLf & vbCrLf & _
      "Please re-select your polygon..." _
      , vbOKOnly, "Error Found in Input Data:"
564:     cbxCorridor.SetFocus
    Exit Sub
566:   ElseIf m_lngCorrCount > 1 Then
567:     cbxCorridor.BackColor = vbYellow
568:     lngIndex = cbxCorridor.ListIndex
569:     strLayerName = cbxCorridor.List(lngIndex)
570:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
```

```
571:     MsgBox "Multiple polygons (" & CStr(m_lngCorrCount) & ") found in Species Corridor Polygon Layer '" & _
        pFeatureLayer.Name & "'!" & vbCrLf & vbCrLf & _
        "Please use the ""Select"" button to select a single " & _
        "polygon from this layer...", vbOKOnly, "Error Found in Input Data:"
575:     cbxCorridor.SetFocus
    Exit Sub

  ' CHECK FOR SAME LAYER SELECTED
579:   ElseIf (intHab1Index > 1) And (intHab2Index > 1) And (intHab1Index = intHab2Index) Then
580:     cbxHab1.BackColor = vbYellow
581:     cbxHab2.BackColor = vbYellow
582:     lngIndex = cbxHab1.ListIndex
583:     strLayerName = cbxHab1.List(lngIndex)
584:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
585:     MsgBox "Both selected Wildland Habitat Block Layers point to the same layer! ('" & _
        pFeatureLayer.Name & "')!" & vbCrLf & vbCrLf & _
        "Please select different layers for each wildland habitat block...", _
        vbOKOnly, "Error Found in Input Data:"
589:     cbxHab1.SetFocus
    Exit Sub
591:   ElseIf (intHab1Index > 1) And (intCorrIndex > 1) And (intHab1Index = intCorrIndex) Then
592:     cbxHab1.BackColor = vbYellow
593:     cbxCorridor.BackColor = vbYellow
594:     lngIndex = cbxHab1.ListIndex
595:     strLayerName = cbxHab1.List(lngIndex)
596:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
597:     MsgBox "Both the Wildland Habitat Block #1 layer and the Species Corridor layer " & _
        "point to the same layer! ('" & _
        pFeatureLayer.Name & "')!" & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
602:     cbxHab1.SetFocus
    Exit Sub
604:   ElseIf (intHab2Index > 1) And (intCorrIndex > 1) And (intHab2Index = intCorrIndex) Then
605:     cbxHab2.BackColor = vbYellow
606:     cbxCorridor.BackColor = vbYellow
607:     lngIndex = cbxHab2.ListIndex
608:     strLayerName = cbxHab2.List(lngIndex)
609:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
610:     MsgBox "Both the Wildland Habitat Block #2 layer and the Species Corridor layer " & _
        "point to the same layer! ('" & _
        pFeatureLayer.Name & "')!" & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
615:     cbxHab2.SetFocus
    Exit Sub
```

```
  ' CHECK FOR INTERSECTING POLYGONS
619:   ElseIf boo1And2 Then
620:     cbxHab1.BackColor = vbYellow
621:     cbxHab2.BackColor = vbYellow
622:     lngIndex = cbxHab1.ListIndex
623:     If lngIndex = 1 Then
624:       strHab1Name = "Selected Graphic"
625:     Else
626:       strLayerName = cbxHab1.List(lngIndex)
627:       Set pFeatureLayer = m_colPolygons.Item(strLayerName)
628:       strHab1Name = pFeatureLayer.Name
629:     End If
630:     lngIndex = cbxHab2.ListIndex
631:     If lngIndex = 1 Then
632:       strHab1Name = "Selected Graphic"
633:     Else
634:       strLayerName2 = cbxHab2.List(lngIndex)
635:       Set pFeatureLayer2 = m_colPolygons.Item(strLayerName2)
636:       strHab2Name = pFeatureLayer2.Name
637:     End If

639:     MsgBox "Your Wildland Habitat Block #1 layer ('" & strHab1Name & "') intersects " & _
       "your Wildland Habitat Block #2 layer ('" & strHab2Name & "')!  If this is true, " & _
       "there is no need for a Species Corridor..." & vbCrLf & vbCrLf & _
       "Please select different layers for these two polygon sources...", _
       vbOKOnly, "Error Found in Input Data:"
644:     cbxHab1.SetFocus
    Exit Sub
646:   ElseIf Not boo1AndCorr Then
647:     cbxHab1.BackColor = vbYellow
648:     cbxCorridor.BackColor = vbYellow
649:     lngIndex = cbxHab1.ListIndex
650:     If lngIndex = 1 Then
651:       strHab1Name = "Selected Graphic"
652:     Else
653:       strLayerName = cbxHab1.List(lngIndex)
654:       Set pFeatureLayer = m_colPolygons.Item(strLayerName)
655:       strHab1Name = pFeatureLayer.Name
656:     End If

658:     lngIndex = cbxCorridor.ListIndex
659:     If lngIndex = 1 Then
660:       strCorrName = "Selected Graphic"
661:     Else
662:       strLayerName2 = cbxCorridor.List(lngIndex)
663:       Set pFeatureLayer2 = m_colPolygons.Item(strLayerName2)
664:       strCorrName = pFeatureLayer2.Name
```

```
665:     End If

667:     MsgBox "Your Wildland Habitat Block #1 layer ('" & strHab1Name & "') does not intersect " & _
        "your Species Corridor layer ('" & strCorrName & "')!  If this is true, " & _
        "then the species corridor polygon cannot connect your two wildland habitat blocks..." & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
672:     cbxHab1.SetFocus
    Exit Sub

675:   ElseIf Not boo2AndCorr Then
676:     cbxHab2.BackColor = vbYellow
677:     cbxCorridor.BackColor = vbYellow
678:     lngIndex = cbxHab2.ListIndex
679:     If lngIndex = 1 Then
680:       strHab2Name = "Selected Graphic"
681:     Else
682:       strLayerName = cbxHab2.List(lngIndex)
683:       Set pFeatureLayer = m_colPolygons.Item(strLayerName)
684:       strHab2Name = pFeatureLayer.Name
685:     End If

687:     lngIndex = cbxCorridor.ListIndex
688:     If lngIndex = 1 Then
689:       strCorrName = "Selected Graphic"
690:     Else
691:       strLayerName2 = cbxCorridor.List(lngIndex)
692:       Set pFeatureLayer2 = m_colPolygons.Item(strLayerName2)
693:       strCorrName = pFeatureLayer2.Name
694:     End If
695:     MsgBox "Your Wildland Habitat Block #2 layer ('" & strHab2Name & "') does not intersect " & _
        "your Species Corridor layer ('" & strCorrName & "')!  If this is true, " & _
        "then the species corridor polygon cannot connect your two wildland habitat blocks..." & vbCrLf & vbCrLf & _
        "Please select different layers for these two polygon sources...", _
        vbOKOnly, "Error Found in Input Data:"
700:     cbxHab2.SetFocus
    Exit Sub
702:   End If

  ' IF PASSED ERROR CHECKS, THEN SAVE PATCH INFO TO EXTENSION PROPERTIES
705:   ext.PatchUse = chkUsePatches.Value
706:   If chkUsePatches.Value Then
707:     lngIndex = cbxPatchLayer.ListIndex
708:     strLayerName = cbxPatchLayer.List(lngIndex)
709:     Set pPatchLayer = m_colPolygons.Item(strLayerName)

711:     Set ext.PatchLayer = pPatchLayer
```

```
    Dim pPatchArray As esriSystem.IArray
714:     Set pPatchArray = New esriSystem.Array

    Dim pFeatureClass As IFeatureClass
717:     Set pFeatureClass = pPatchLayer.FeatureClass

    Dim pFeatureCursor As esrigeodatabase.IFeatureCursor
    Dim pSpatialFilter As ISpatialFilter
721:     Set pSpatialFilter = New SpatialFilter

723:     If optSubSet.Value Then

      Dim intFieldIndex As Integer
      Dim strFieldName As String
727:       strFieldName = cbxPatchField.List(cbxPatchField.ListIndex)
728:       intFieldIndex = m_colFieldNames.Item(strFieldName)

      Dim pLayerFields As ILayerFields
731:       Set pLayerFields = pPatchLayer
      Dim pField As IField
733:       Set pField = pLayerFields.Field(intFieldIndex)

      Dim strValue As String
736:       strValue = txtValue.Text
737:       If pField.Type = esriFieldTypeString Then
738:         If (Not Left(strValue, 1) = Chr(34)) And (Not Right(strValue, 1) = Chr(34)) Then
739:           strValue = Linkages.aml_func_mod.QuoteString(strValue)
740:         End If
741:       End If

743:       Set ext.PatchField = pField
744:       ext.PatchAttFieldIndex = intFieldIndex
745:       ext.PatchOperatorIndex = cbxValue.ListIndex
746:       ext.PatchAttValue = strValue

      Dim theDataSourceType As String
      Dim theSQLStyleKey As String
      Dim theFieldName As String
751:       theFieldName = pField.Name
752:       theDataSourceType = pPatchLayer.DataSourceType
      Dim strOperator As String
754:       strOperator = cbxValue.List(cbxValue.ListIndex)

      ' IF NECESSARY TO QUERY DATA, NEED TO KNOW DATA SOURCE TYPE
      ' DOCUMENTATION IS UNFORTUNATELY VAGUE ABOUT THIS.
758:       If theDataSourceType = "Personal Geodatabase Feature Class" Then
```

```
759:         theSQLStyleKey = "brackets"
760:       ElseIf theDataSourceType = "SDE Feature Class" Then
761:         theSQLStyleKey = "Not Enclosed"
762:       ElseIf InStr(1, theDataSourceType, "IMS", vbBinaryCompare) > 0 Then
763:         theSQLStyleKey = "Not Enclosed"
764:       Else
765:         theSQLStyleKey = "Quotations"
766:       End If

      Dim theQueryString As String

770:       If theSQLStyleKey = "brackets" Then
771:         theQueryString = "[" & theFieldName & "] " & strOperator & strValue

773:       ElseIf theSQLStyleKey = "Quotations" Then
774:         theQueryString = """" & theFieldName & """" & strOperator & strValue

776:       ElseIf theSQLStyleKey = "Not Enclosed" Then
777:         theQueryString = theQueryString & theFieldName & strOperator & strValue

779:       End If

      'Debug.Print theQueryString

783:       pSpatialFilter.WhereClause = theQueryString
784:       pSpatialFilter.SubFields = (theFieldName)
785:     End If

787:     With pSpatialFilter
788:       Set .Geometry = pCorrPoly
789:       .GeometryField = "SHAPE"
790:       .SpatialRel = esriSpatialRelIntersects
791:       .AddField "SHAPE"
792:     End With

794:     Set pFeatureCursor = pFeatureClass.Search(pSpatialFilter, True)

'    MsgBox "Feature Cursor is nothing? " & CStr(pFeatureCursor Is Nothing) & vbCrLf & _
        "Number Patches = " & pFeatureClass.FeatureCount(pSpatialFilter)

    Dim pFeature As IFeature
800:     Set pFeature = pFeatureCursor.NextFeature

    Dim pPatchPoly As IGeometry
    Dim lngCounter As Long
804:     Do While Not pFeature Is Nothing
805:       lngCounter = lngCounter + 1
```

```
    '  MsgBox lngCounter
      'If pFeature.Shape.IsEmpty Then
      '  MsgBox "Empty???"
      'End If
810:       Set pPatchPoly = pFeature.ShapeCopy
811:       pPatchArray.Add pPatchPoly
812:       Set pFeature = pFeatureCursor.NextFeature
813:     Loop

815:     Set ext.PatchArray = pPatchArray

817:   Else
818:     Set ext.PatchLayer = Nothing
819:     Set ext.PatchField = Nothing
820:     ext.PatchUseAll = False
821:     ext.PatchOperatorIndex = 0
822:     ext.PatchAttValue = ""
823:     ext.PatchAttFieldIndex = 0
824:     Set ext.PatchArray = Nothing
825:   End If

827:   Call CheckSavedValues
  Exit Sub
ErrorHandler:
  HandleError True, "cmdOK_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub CheckSavedValues()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
837:   Set ext = m_ExtensionConfig

  Dim pArea1 As IArea
  Dim pPoly1 As IPolygon
  Dim pArea2 As IArea
  Dim pPoly2 As IPolygon
  Dim pArea3 As IArea
  Dim pPoly3 As IPolygon

  Dim pPatchLayer As IFeatureLayer
  Dim pField As IField
  Dim intFieldIndex As Integer
  Dim booUsePatchas As Boolean
  Dim booUseSubset As Boolean
  Dim intPatchOperator As Integer
```

```
  Dim txtPatchValue As String

854:   Set pPoly1 = ext.PolyWildland1
855:   Set pArea1 = pPoly1

857:   Set pPoly2 = ext.PolyWildland2
858:   Set pArea2 = pPoly2

860:   Set pPoly3 = ext.PolyCorridor
861:   Set pArea3 = pPoly3
  Dim pPatchArray As esriSystem.IArray

864:   If ext.PatchArray Is Nothing Then
865:     Set pPatchArray = New esriSystem.Array
866:   Else
867:     Set pPatchArray = ext.PatchArray
868:   End If

'  MsgBox "PatchArray nothing? " & CStr(pPatchArray Is Nothing)

'  Dim strReport As String
'  strReport = "Wildland Block #1 Area = " & CStr(pArea1.Area) & vbCrLf & _
'              "Wildland Block #2 Area = " & CStr(pArea2.Area) & vbCrLf & _
'              "Corridor Area = " & CStr(pArea3.Area) & vbCrLf & _
'              "Use Patch Polygons = " & CStr(ext.PatchUse) & vbCrLf & _
'              "Patch Layer = " & ext.PatchLayer.Name & vbCrLf & _
'              "Use All Patches = " & ext.PatchUseAll & vbCrLf & _
'              "Patch Attribute Field = " & ext.PatchField.Name & vbCrLf & _
'              "Patch Attribute Field Index = " & CStr(ext.PatchAttFieldIndex) & vbCrLf & _
'              "Patch Attribute Value = " & CStr(ext.PatchAttValue) & vbCrLf & _
'              pPatchArray.Count & " selected patches..."
'  MsgBox strReport


    ' GET INTERNAL NODES - FROM ORIGINAL TEST CODE
  Dim pStartArray As IArray
888:   Set pStartArray = pPatchArray

  ' GET START AND END NODES - FROM ORIGINAL TEST CODE
  Dim pStartPolygon As IPolygon
  Dim pEndPolygon As IPolygon
893:   Set pStartPolygon = ext.PolyWildland1
894:   Set pEndPolygon = ext.PolyWildland2

896:   pStartArray.Insert 0, pStartPolygon
897:   pStartArray.Add pEndPolygon
```

```
  Dim pClone As IClone

  ' GET CORRIDOR - FROM ORIGINAL TEST CODE
  Dim pCorPolygon As IPolygon
903:   Set pCorPolygon = ext.PolyCorridor









'  ' TURNED OFF REMAINING CODE FOR ESRI CONFERENCE
'  Dim pFrm1 As New Linkages.frmEsriSample
'  pFrm1.Show vbModal


'  Set pFrm1 = Nothing






  ' PROGRESS METER STUFF --------------------------------------------------------------------
  Dim frmProgress As New frmJenProgressPercent
  Dim theTimeBegan As Date
  Dim theDetailedDescription As String

'  frmProgress.SetExpanded = ext.ProgressDialogSetExpanded
931:   frmProgress.SetExpanded = True
932:   frmProgress.SetAutoClose = ext.ProgressDialogAutoClose

934:   theTimeBegan = Now
935:   frmProgress.ShouldContinue = True
936:   frmProgress.ProgBeginTime = Now
937:   frmProgress.ProgRecCount = 0
938:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
939:   frmProgress.lblBeginTime.Caption = "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd")

941:   theDetailedDescription = "Analyzing Patch Connectors..." & vbCrLf & _
      "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd") & vbCrLf & _
      "-----------------------------------------" & vbCrLf
944:   frmProgress.txtDetails.Text = theDetailedDescription
```

```
946:   frmProgress.Frame.Caption = "Current Status:"
947:   frmProgress.Frame.Visible = True
948:   frmProgress.cmdDetails.Visible = False
949:   frmProgress.cmdStop.Visible = False

951:   Me.Frame.Visible = False

'   theProgressTimeCheck = CDate(50000)
  Dim theDescription As String
955:   theDescription = "Analyzing Patch Connectors..."
  ' PROGRESS METER STUFF ------------------------------------------------------------------

  Dim pParamDetails As esriSystem.IVariantArray
959:   Set pParamDetails = New esriSystem.VarArray

  Dim strLayerName As String
  Dim pFeatureLayer As IFeatureLayer
  Dim lngIndex As Long

  ' WILDLAND BLOCK 1
966:   If cbxHab1.ListIndex < 2 Then
967:     pParamDetails.Add "Selected Specific Polygon from Map..."
968:   Else
969:     strLayerName = cbxHab1.List(cbxHab1.ListIndex)
970:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
971:     pParamDetails.Add pFeatureLayer.Name
972:   End If

  ' WILDLAND BLOCK 2
975:   If cbxHab2.ListIndex < 2 Then
976:     pParamDetails.Add "Selected Specific Polygon from Map..."
977:   Else
978:     strLayerName = cbxHab2.List(cbxHab2.ListIndex)
979:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
980:     pParamDetails.Add pFeatureLayer.Name
981:   End If

  ' CORRIDOR POLYGON
984:   If cbxCorridor.ListIndex < 2 Then
985:     pParamDetails.Add "Selected Specific Polygon from Map..."
986:   Else
987:     strLayerName = cbxCorridor.List(cbxCorridor.ListIndex)
988:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
989:     pParamDetails.Add pFeatureLayer.Name
990:   End If

992:   pParamDetails.Add CBool(chkUsePatches.Value = 1)
```

```
  ' PATCH LAYER
995:   If cbxPatchLayer.ListIndex = 0 Then
996:     pParamDetails.Add cbxPatchLayer.Text
997:   Else
998:     strLayerName = cbxPatchLayer.List(cbxPatchLayer.ListIndex)
999:     Set pFeatureLayer = m_colPolygons.Item(strLayerName)
1000:     pParamDetails.Add pFeatureLayer.Name
1001:   End If

1003:   pParamDetails.Add optUseAll.Value
1004:   pParamDetails.Add cbxPatchField.Text
1005:   pParamDetails.Add cbxValue.Text
1006:   pParamDetails.Add txtValue.Text

'   ' FOR DEBUGGING
'   Dim strTimeElapsedRTF As String
'   strTimeElapsedRTF = MyGeneralOperations.ReturnTimeElapsedRTF(Now - 1000, Now, 8)
'   Dim dblFinalDistances() As Double
'   ReDim dblFinalDistances(5)
'   dblFinalDistances(0) = 23
'   dblFinalDistances(1) = 3245
'   dblFinalDistances(2) = 34
'   dblFinalDistances(3) = 222
'   dblFinalDistances(4) = 11
'   dblFinalDistances(5) = 111
'   ' MAKE OUTPUT REPORT
'   Call CorridorAnalysisFunctions.MakePatchReport(dblFinalDistances, m_MxDoc, strTimeElapsedRTF, pParamDetails)
'   Exit Sub
'
'   ' END DEBUGGING

1025:   CorridorAnalysisFunctions.ImplementKruskall m_MxDoc, pStartPolygon, pEndPolygon, _
      pStartArray, pCorPolygon, frmProgress, m_ExtensionConfig, pParamDetails

1028:   ext.ProgressDialogAutoClose = (frmProgress.chkClose.Value = 1)
1029:   ext.ProgressDialogSetExpanded = (frmProgress.SetExpanded)

1031:   If frmProgress.chkClose.Value = 1 Then
1032:     Unload frmProgress
1033:     Set frmProgress = Nothing
1034:   End If

1036:   Me.Frame.Visible = True

1038:   Unload Me
```

```
  Exit Sub
ErrorHandler:
  HandleError False, "CheckSavedValues " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdSel1_Click()
  On Error GoTo ErrorHandler

  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
1056:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
1059:   Set theFormObject = ext.aSelForm

1061:   If (theFormObject Is Nothing) Then
1062:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
    Dim pMxDoc As IMxDocument
1066:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType

1082:     intKey = -1

1084:     Set theMap = pMxDoc.FocusMap
```

```
    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
1088:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
1091:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
    Dim pEnumElement As IEnumElement

1094:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)

    Dim booHasPolygon As Boolean
'    booHasPolygon = False
1099:     booHasPolygon = True
'    If (Not pEnumElement Is Nothing) Then
'      pEnumElement.Reset
'
'      Dim pElement As IElement
'      Set pElement = pEnumElement.Next
'
'      Dim pGeometry As IGeometry
'
'      Do Until pElement Is Nothing
'        Set pGeometry = pElement.Geometry
'        If TypeOf pGeometry Is IPolygon Then
'          booHasPolygon = True
'          Exit Do
'        End If
'        Set pElement = pEnumElement.Next
'      Loop
'    End If

'    MsgBox booHasPolygon

    ReDim strNameArray(theMap.LayerCount)

1122:     If (booHasPolygon) Then
1123:       intKey = intKey + 1
1124:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
1125:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1126:       strNameArray(intKey) = strPolyName
1127:     End If

    Dim pFeatureLayerForValid As IFeatureLayer

1131:     If (theMap.LayerCount > 0) Then
1132:       Set pEnumLayer = theMap.Layers(, True)
```

```
1133:       pEnumLayer.Reset

1135:       Set pLayer = pEnumLayer.Next
1136:       Do Until pLayer Is Nothing
1137:         If TypeOf pLayer Is IFeatureLayer Then
1138:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
1140:           If pFeatureLayerForValid.Valid Then
           ' CHECK IF POLYGON LAYER
1142:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
1143:             pGeometryType = pFeatureClass.ShapeType
1144:             If (pGeometryType = esriGeometryPolygon) Then
1145:               intKey = intKey + 1
1146:               Set pFeatureLayer = pLayer
1147:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
1148:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1149:               strNameArray(intKey) = strPolyName
1150:             End If
1151:           End If
1152:         End If
1153:         Set pLayer = pEnumLayer.Next
1154:       Loop
1155:     End If

'    Dim theReport As String
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport

1163:     Set frmSelect.ArcApplication = m_pApp
1164:     Set frmSelect.Doc = m_MxDoc
1165:     frmSelect.NameList = strNameArray
1166:     frmSelect.NameCount = intKey
1167:     Set frmSelect.NameCollection = colPolygonLayers
1168:     frmSelect.PolygonPurpose = "Wildland1"
1169:     frmSelect.SearchMessage = "Wildland Block #1"

1171:     frmSelect.EnableTool = False
1172:     frmSelect.SetSelToolEnabled
1173:     Load frmSelect

1175:     frmSelect.Frame.Caption = "Select Wildland Block #1..."
1176:     frmSelect.Frame.Visible = True
1177:   Else
1178:     Set frmSelect = theFormObject
1179:   End If
```

```
1181:   cbxHab1.BackColor = vbWhite

'  frmSelect.Show vbModeless
1184:   frmSelect.Frame.Visible = True
1185:   Me.Frame.Visible = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSel1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdSel2_Click()
  On Error GoTo ErrorHandler
1194:   m_IntHelpCategory = 1
1195:   Call UpdateHelpScreen

  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
1200:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
1203:   Set theFormObject = ext.aSelForm

1205:   If (theFormObject Is Nothing) Then
1206:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
    Dim pMxDoc As IMxDocument
1210:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType
```

```
1226:     intKey = -1

1228:     Set theMap = pMxDoc.FocusMap

    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
1232:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
1235:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
    Dim pEnumElement As IEnumElement

1238:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)

    Dim booHasPolygon As Boolean
'    booHasPolygon = False
1243:     booHasPolygon = True
'
'    If (Not pEnumElement Is Nothing) Then
'      pEnumElement.Reset
'
'      Dim pElement As IElement
'      Set pElement = pEnumElement.Next
'
'      Dim pGeometry As IGeometry
'
'      Do Until pElement Is Nothing
'        Set pGeometry = pElement.Geometry
'        If TypeOf pGeometry Is IPolygon Then
'          booHasPolygon = True
'          Exit Do
'        End If
'        Set pElement = pEnumElement.Next
'      Loop
'    End If

'    MsgBox booHasPolygon

    ReDim strNameArray(theMap.LayerCount)

1267:     If (booHasPolygon) Then
1268:       intKey = intKey + 1
1269:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
1270:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1271:       strNameArray(intKey) = strPolyName
1272:     End If
```

```
    Dim pFeatureLayerForValid As IFeatureLayer

1276:     If (theMap.LayerCount > 0) Then
1277:       Set pEnumLayer = theMap.Layers(, True)
1278:       pEnumLayer.Reset

1280:       Set pLayer = pEnumLayer.Next
1281:       Do Until pLayer Is Nothing
1282:         If TypeOf pLayer Is IFeatureLayer Then
1283:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
1285:           If pFeatureLayerForValid.Valid Then
            ' CHECK IF POLYGON LAYER
1287:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
1288:             pGeometryType = pFeatureClass.ShapeType
1289:             If (pGeometryType = esriGeometryPolygon) Then
1290:               intKey = intKey + 1
1291:               Set pFeatureLayer = pLayer
1292:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
1293:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1294:               strNameArray(intKey) = strPolyName
1295:             End If
1296:           End If
1297:         End If
1298:         Set pLayer = pEnumLayer.Next
1299:       Loop
1300:     End If

'    Dim theReport As String
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport

1308:     Set frmSelect.ArcApplication = m_pApp
1309:     Set frmSelect.Doc = m_MxDoc
1310:     frmSelect.NameList = strNameArray
1311:     frmSelect.NameCount = intKey
1312:     Set frmSelect.NameCollection = colPolygonLayers
1313:     frmSelect.PolygonPurpose = "Wildland2"
1314:     frmSelect.SearchMessage = "Wildland Block #2"

1316:     frmSelect.EnableTool = False
1317:     frmSelect.SetSelToolEnabled
1318:     Load frmSelect
```

```
1320:     frmSelect.Frame.Caption = "Select Wildland Block #2..."
1321:     frmSelect.Frame.Visible = True
1322:   Else
1323:     Set frmSelect = theFormObject
1324:   End If

1326:   cbxHab2.BackColor = vbWhite

'  frmSelect.Show vbModeless
1329:   frmSelect.Frame.Visible = True
1330:   Me.Frame.Visible = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSel2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cmdSelLink_Click()
  On Error GoTo ErrorHandler

1340:   m_IntHelpCategory = 1
1341:   Call UpdateHelpScreen
  Dim frmSelect As Linkages.frmSelScreen

  Dim ext As Linkages.Extension
1345:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
1348:   Set theFormObject = ext.aSelForm

1350:   If (theFormObject Is Nothing) Then
1351:     Set frmSelect = New Linkages.frmSelScreen

    ' IDENTIFY POLYGON THEMES AND GRAPHICS
    Dim pMxDoc As IMxDocument
1355:     Set pMxDoc = m_MxDoc

    Dim colPolygonLayers As New Collection
    Dim strNameArray() As String

    Dim theMap As IMap
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pLayer As IUnknown
    Dim anIndex As Long
    Dim strPolyName As String
```

```
    Dim intKey As Integer

    Dim pFeatureClass As IFeatureClass
    Dim pGeometryType As esriGeometryType

1371:     intKey = -1

1373:     Set theMap = pMxDoc.FocusMap

    ' CHECK IF GRAPHICS LAYER IS AVAILABLE
    Dim pGraphicsContainer As IGraphicsContainer
1377:     Set pGraphicsContainer = theMap

    Dim pEnvelope As IEnvelope
1380:     Set pEnvelope = pMxDoc.ActiveView.FullExtent
    Dim pEnumElement As IEnumElement

1383:     Set pEnumElement = pGraphicsContainer.LocateElementsByEnvelope(pEnvelope)
'    MsgBox (pEnumElement Is Nothing)

    Dim booHasPolygon As Boolean
1387:     booHasPolygon = True
'    booHasPolygon = False
'
'    If (Not pEnumElement Is Nothing) Then
'      pEnumElement.Reset
'
'      Dim pElement As IElement
'      Set pElement = pEnumElement.Next
'
'      Dim pGeometry As IGeometry
'
'      Do Until pElement Is Nothing
'        Set pGeometry = pElement.Geometry
'        If TypeOf pGeometry Is IPolygon Then
'          booHasPolygon = True
'          Exit Do
'        End If
'        Set pElement = pEnumElement.Next
'      Loop
'    End If

'    MsgBox booHasPolygon

    ReDim strNameArray(theMap.LayerCount)

1412:     If (booHasPolygon) Then
```

```
1413:       intKey = intKey + 1
1414:       strPolyName = "1] <-- Select or Draw Graphic Polygon -->"
1415:       colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1416:       strNameArray(intKey) = strPolyName
1417:     End If

    Dim pFeatureLayerForValid As IFeatureLayer

1421:     If (theMap.LayerCount > 0) Then
1422:       Set pEnumLayer = theMap.Layers(, True)
1423:       pEnumLayer.Reset

1425:       Set pLayer = pEnumLayer.Next
1426:       Do Until pLayer Is Nothing
1427:         If TypeOf pLayer Is IFeatureLayer Then
1428:           Set pFeatureLayerForValid = pLayer
          ' CHECK IF FEATURE LAYER IS VALID
1430:           If pFeatureLayerForValid.Valid Then
          ' CHECK IF POLYGON LAYER
1432:             Set pFeatureClass = pFeatureLayerForValid.FeatureClass
1433:             pGeometryType = pFeatureClass.ShapeType
1434:             If (pGeometryType = esriGeometryPolygon) Then
1435:               intKey = intKey + 1
1436:               Set pFeatureLayer = pLayer
1437:               strPolyName = CStr(intKey + 1) & "] " & pFeatureLayer.Name
1438:               colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1439:               strNameArray(intKey) = strPolyName
1440:             End If
1441:           End If
1442:         End If
1443:         Set pLayer = pEnumLayer.Next
1444:       Loop
1445:     End If

'    Dim theReport As String
'    For anIndex = 0 To intKey
'      theReport = theReport & strNameArray(anIndex) & vbCrLf
'    Next anIndex
'    MsgBox theReport

1453:     Set frmSelect.ArcApplication = m_pApp
1454:     Set frmSelect.Doc = m_MxDoc
1455:     frmSelect.NameList = strNameArray
1456:     frmSelect.NameCount = intKey
1457:     Set frmSelect.NameCollection = colPolygonLayers
1458:     frmSelect.PolygonPurpose = "Corridor"
1459:     frmSelect.SearchMessage = "Species Corridor"
```

```
1461:     frmSelect.EnableTool = False
1462:     frmSelect.SetSelToolEnabled
1463:     Load frmSelect

1465:     frmSelect.Frame.Caption = "Select Species Corridor Polygon..."
1466:     frmSelect.Frame.Visible = True
1467:   Else
1468:     Set frmSelect = theFormObject
1469:   End If

1471:   cbxCorridor.BackColor = vbWhite

'  frmSelect.Show vbModeless
1474:   frmSelect.Frame.Visible = True
1475:   Me.Frame.Visible = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdSelLink_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Sub UpdateCheckmarks()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
1486:   Set ext = m_ExtensionConfig

  Dim pHab1Polygon As IPolygon
  Dim pHab2Polygon As IPolygon
  Dim pCorPolygon As IPolygon

1492:   Set pHab1Polygon = ext.PolyWildland1
1493:   Set pHab2Polygon = ext.PolyWildland2
1494:   Set pCorPolygon = ext.PolyCorridor

1496:   imgCheckWB1.Visible = (Not ext.PolyWildland1 Is Nothing)
1497:   imgCheckWB2.Visible = (Not ext.PolyWildland2 Is Nothing)
1498:   imgCheckSpCorr.Visible = (Not ext.PolyCorridor Is Nothing)
1499:   imgUnCheckWB1.Visible = Not imgCheckWB1.Visible
1500:   imgUnCheckWB2.Visible = Not imgCheckWB2.Visible
1501:   imgUnCheckSpCorr.Visible = Not imgCheckSpCorr.Visible

  Exit Sub
ErrorHandler:
  HandleError True, "UpdateCheckmarks " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler
1510:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
  '  ORIGINAL AVENUE CODE
  '  ' Jennessent.CompareParametersOpen
  '
  '  AVUpperLeft = av.ReturnOrigin
  '  AVCenter = avUpperLeft + (av.ReturnExtent / (2@2))
  '  halfDialogWidthHeight = Self.ReturnExtent.ReturnSize / (2@2)
  '  MovePoint = AVCenter - halfDialogWidthHeight
  '  Self.MoveTo(MovePoint.GetX, MovePoint.GetY)
  '
  '  theDialog = self
  '  cmdOK = theDialog.FindByName("cmdOK")
  '  cmdCancel = theDialog.FindByName("cmdCancel")
  '
'  NOTES ON CONTROLS:
'  Remember to reset label autosize to true, if desired.
'   --> Contains Control Panels:  Make sure to cut and paste nested controls
'          if you want them to be properly nested in the Form Frame.
'   --> Contains Control Panels:  Make sure to cut and paste nested controls
'          if you want them to be properly nested in the Form Frame.
'   --> Contains Control Panels:  Make sure to cut and paste nested controls
'          if you want them to be properly nested in the Form Frame.
'   --> icon_CornerBars: Original Image = jennessent_bars.gif
1533:   SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
1534:   Me.Left = (Screen.Width / 3) - (Me.Width / 2)
1535:   Me.Top = (Screen.Height / 2) - (Me.Height / 2)

1537:   If m_Frame Is Nothing Then
1538:     Set m_Frame = New ModelessFrame
1539:     m_Frame.Create Me
'    Set m_WindowPos = m_Frame
'    MsgBox m_WindowPos.Width & "    x    " & m_WindowPos.Height
1542:   End If

  ' CLEAR ANY SAVED EXTENSION POLYGONS
  Dim newUid As New uID
1546:   newUid.Value = "Linkages.Extension"
1547:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
1549:   Set ext = m_ExtensionConfig

  ' FILL VALUE BOX
```

```
1552:   cbxValue.Clear
1553:   cbxValue.AddItem ("<")
1554:   cbxValue.AddItem ("<=")
1555:   cbxValue.AddItem ("=")
1556:   cbxValue.AddItem (">=")
1557:   cbxValue.AddItem (">")
1558:   cbxValue.AddItem ("<>")
1559:   m_intValueIndex = ext.PatchOperatorIndex
1560:   If m_intValueIndex = 0 Then
1561:     m_intValueIndex = 3
1562:     ext.PatchOperatorIndex = 3
1563:   End If

1565:   cbxValue.ListIndex = m_intValueIndex - 1

1567:   Set ext.PolyWildland1 = Nothing
1568:   Set ext.PolyWildland2 = Nothing
1569:   Set ext.PolyCorridor = Nothing
1570:   Set ext.frmStep1 = Me

  ' CHECK HELP WINDOW STATUS
1573:   m_booHelpToggle = ext.HelpToggle1

    ' IDENTIFY POLYGON THEMES
  Dim pMxDoc As IMxDocument
1577:   Set pMxDoc = m_MxDoc

  Dim colPolygonLayers As New Collection
  Dim strNameArray() As String

  Dim theMap As IMap
  Dim pEnumLayer As IEnumLayer
  Dim pFeatureLayer As IFeatureLayer
  Dim pLayer As IUnknown
  Dim anIndex As Long
  Dim strPolyName As String
  Dim intKey As Integer

  Dim pFeatureClass As IFeatureClass
  Dim pGeometryType As esriGeometryType

1593:   intKey = 1

1595:   Set theMap = pMxDoc.FocusMap

  ReDim strNameArray(theMap.LayerCount + 2)
```

```
1599:   colPolygonLayers.Add "placeholder_FirstLine", "FirstLine"
1600:   strNameArray(0) = "FirstLine"

1602:   colPolygonLayers.Add "placeholder_SelFromView", "SelFromView"
1603:   strNameArray(1) = "SelFromView"

  Dim pFeatureLayerForValid As IFeatureLayer

1607:   If (theMap.LayerCount > 0) Then
1608:     Set pEnumLayer = theMap.Layers(, True)
1609:     pEnumLayer.Reset

1611:     Set pLayer = pEnumLayer.Next
1612:     Do Until pLayer Is Nothing
1613:       If TypeOf pLayer Is IFeatureLayer Then
1614:         Set pFeatureLayerForValid = pLayer
        ' CHECK IF FEATURE LAYER IS VALID
1616:         If pFeatureLayerForValid.Valid Then
          ' CHECK IF POLYGON LAYER
1618:           Set pFeatureClass = pFeatureLayerForValid.FeatureClass
1619:           pGeometryType = pFeatureClass.ShapeType
1620:           If (pGeometryType = esriGeometryPolygon) Then
1621:             intKey = intKey + 1
1622:             Set pFeatureLayer = pLayer
1623:             strPolyName = CStr(intKey - 1) & "] " & pFeatureLayer.Name
1624:             colPolygonLayers.Add pFeatureLayer, CStr(strPolyName)
1625:             strNameArray(intKey) = strPolyName
1626:           End If
1627:         End If
1628:       End If
1629:       Set pLayer = pEnumLayer.Next
1630:     Loop
1631:   End If

1633:   m_strNameArray = strNameArray
1634:   Set m_colPolygons = colPolygonLayers

  ' FILL LISTBOXES WITH POLYGON SOURCE OPTIONS

  Dim strWB1Array() As String
  ReDim strWB1Array(intKey)
1640:   strWB1Array(0) = "Wildland Block #1 Source..."
1641:   strWB1Array(1) = "<-- Select by clicking on map -->"

  Dim strWB2Array() As String
  ReDim strWB2Array(intKey)
1645:   strWB2Array(0) = "Wildland Block #2 Source..."
```

```
1646:   strWB2Array(1) = "<-- Select by clicking on map -->"

  Dim strCorArray() As String
  ReDim strCorArray(intKey)
1650:   strCorArray(0) = "Species Corridor Source..."
1651:   strCorArray(1) = "<-- Select by clicking on map -->"

  Dim strPatchArray() As String
  ReDim strPatchArray(intKey - 1)
1655:   strPatchArray(0) = "Patch Polygon Layer..."


'  Dim theReport As String
'  theReport = "Upper bound of 'm_strNameArray' = " & CStr(UBound(m_strNameArray)) & vbCrLf & _
'      "Value = " & m_strNameArray(UBound(m_strNameArray))
'  For anIndex = LBound(m_strNameArray) To UBound(m_strNameArray)
'    theReport = theReport & "Index " & CStr(anIndex) & " --> " & m_strNameArray(anIndex) _
'        & vbCrLf
'  Next anIndex
'
'  MsgBox theReport

1668:   For anIndex = 2 To UBound(m_strNameArray)
1669:     If Not m_strNameArray(anIndex) = "" Then
1670:       strWB1Array(anIndex) = m_strNameArray(anIndex)
1671:       strWB2Array(anIndex) = m_strNameArray(anIndex)
1672:       strCorArray(anIndex) = m_strNameArray(anIndex)
1673:       strPatchArray(anIndex - 1) = m_strNameArray(anIndex)
1674:     End If
1675:   Next anIndex

1677:   cbxHab1.Clear
1678:   cbxHab2.Clear
1679:   cbxCorridor.Clear
1680:   cbxPatchLayer.Clear

1682:   For anIndex = 0 To UBound(strWB1Array)
1683:     cbxHab1.AddItem (strWB1Array(anIndex))
1684:     cbxHab2.AddItem (strWB2Array(anIndex))
1685:     cbxCorridor.AddItem (strCorArray(anIndex))
1686:   Next anIndex

1688:   For anIndex = 0 To UBound(strPatchArray)
1689:     cbxPatchLayer.AddItem (strPatchArray(anIndex))
1690:   Next anIndex

1692:   cbxHab1.ListIndex = 0
```

```
1693:   cbxHab2.ListIndex = 0
1694:   cbxCorridor.ListIndex = 0
1695:   cbxPatchLayer.ListIndex = 0

  ' SET PATCH CHOICE OPTIONS
  Dim booPatchUseAll As Boolean
1699:   booPatchUseAll = ext.PatchUseAll
1700:   optUseAll.Value = booPatchUseAll
1701:   optSubSet.Value = Not booPatchUseAll
1702:   Call EnablePatchOptions

1704:   cbxPatchField.Clear
1705:   cbxPatchField.AddItem ("Select Attribute Field...")
1706:   cbxPatchField.ListIndex = 0

  Dim strAttributeValue As String
1709:   strAttributeValue = ext.PatchAttValue
1710:   txtValue.Text = strAttributeValue

  ' SET CHECKMARKS AND UPDATE BUTTONS
1713:   Call UpdateCheckmarks
1714:   Call UpdateSelectButtons

  ' UPDATE HELP SCREEN POSITION
  Dim booHelpToggle As Boolean
1718:   booHelpToggle = ext.HelpToggle1
  Dim pWindowPosition As IWindowPosition
1720:   Set pWindowPosition = m_Frame

1722:   If booHelpToggle = False Then
1723:     cmdHelp.Caption = "Show Help >>"
1724:     pWindowPosition.Width = 390
1725:   Else
1726:     cmdHelp.Caption = "<< Hide Help"
1727:     pWindowPosition.Width = 669
1728:   End If
1729:   m_IntHelpCategory = 1
1730:   Call UpdateHelpScreen

  ' MAKE SURE BACK COLORS ARE WHITE
1733:   Call SetBackColorsWhite

1735:   Me.Refresh

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
```

```
End Sub

Private Sub UpdateHelpScreen()
  On Error GoTo ErrorHandler

  Dim ext As Linkages.Extension
1746:   Set ext = m_ExtensionConfig
  Dim booHelpToggle As Boolean
1748:   booHelpToggle = ext.HelpToggle1

  Dim booPatchesVisible As Boolean
1751:   booPatchesVisible = booHelpToggle And (m_IntHelpCategory = 2)
  Dim booPatches2Visible As Boolean
1753:   booPatches2Visible = booHelpToggle And (m_IntHelpCategory = 3)
  Dim booPolygonsVisible As Boolean
1755:   booPolygonsVisible = booHelpToggle And (m_IntHelpCategory = 1)

1757:   imgHelpPolygons.Visible = booPolygonsVisible
1758:   imgHelpPatches.Visible = booPatchesVisible
1759:   imgHelpPatches2.Visible = booPatches2Visible
1760:   rtbHelp.Visible = booHelpToggle

  Dim strHelpText As String
  Select Case m_IntHelpCategory
    Case 1  ' POLYGON SELECTION, UPPER PART OF FORM
1765:       strHelpText = Linkages.modHelpStrings.Step1Polygons
1766:       rtbHelp.TextRTF = strHelpText
    Case 2  ' PATCH SELECTION, GENERAL
1768:       strHelpText = Linkages.modHelpStrings.Step2PatchesGeneral
1769:       rtbHelp.TextRTF = strHelpText
    Case 3  ' PATCH SELECTION, BY CRITERIA
1771:       strHelpText = Linkages.modHelpStrings.Step2PatchesCriteria
1772:       rtbHelp.TextRTF = strHelpText
1773:   End Select

1775:   rtbHelp.SelStart = 0
1776:   rtbHelp.Locked = True

  Exit Sub
ErrorHandler:
  HandleError False, "UpdateHelpScreen " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub UpdateSelectButtons()
  On Error GoTo ErrorHandler
```

```
1787:   cmdSel1.Enabled = (cbxHab1.ListIndex = 1)
1788:   cmdSel2.Enabled = (cbxHab2.ListIndex = 1)
1789:   cmdSelLink.Enabled = (cbxCorridor.ListIndex = 1)


  Exit Sub
ErrorHandler:
  HandleError False, "UpdateSelectButtons " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler

  '  ORIGINAL AVENUE CODE
  '  ' Jennessent.CompareParametersClose
  '
  '  self.SetObjectTag(nil)
  '  self.FindByName("cmdOK").SetObjectTag(nil)
  '  self.FindByName("cmdCancel").SetObjectTag(nil)
  '
    ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
1808:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors")

  Dim pCommand As esriSystemUI.ICommand
  Dim pUID As New uID
1812:   pUID.Value = "Linkages.toolReturnCoords"

  Dim ext As Linkages.Extension
1815:   Set ext = m_ExtensionConfig
1816:   ext.EnableSelTool = False
1817:   ext.EnableDrawTool = False
1818:   Set ext.PolyCorridor = Nothing
1819:   Set ext.PolyWildland1 = Nothing
1820:   Set ext.PolyWildland2 = Nothing
1821:   Set ext.frmStep1 = Nothing
1822:   Set ext.PatchField = Nothing
1823:   Set ext.PatchLayer = Nothing

  ' UNLOAD SELECTION FORM IF IT IS OPEN
1826:   If (Not ext.aSelForm Is Nothing) Then
    Dim pSelForm As Linkages.frmSelScreen
1828:     Set pSelForm = ext.aSelForm
1829:     Unload pSelForm
1830:     Set ext.aSelForm = Nothing
1831:   End If
```

```
1833:   Set pCommand = m_pApp.Document.CommandBars.Find(pUID)
1834:   If pCommand.Checked Then
1835:     Set m_pApp.CurrentTool = Nothing

    Dim pCommandItem As ICommandItem
1838:     Set pCommandItem = pCommand

1840:     pCommandItem.Refresh
1841:   End If

1843:   Set pCommand = Nothing
1844:   Set pCommandItem = Nothing
1845:   Set ext = Nothing
1846:   Set m_pApp = Nothing
1847:   Set m_MxDoc = Nothing
1848:   Set m_Frame = Nothing
1849:   Set m_ExtensionConfig = Nothing
1850:   Set m_colPolygons = Nothing
'  Set m_WindowPos = Nothing



  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub imgFrameBack_Click()

1862:   m_IntHelpCategory = 2
1863:   Call UpdateHelpScreen

End Sub

Private Sub Label1_Click()
  On Error GoTo ErrorHandler

1870:   m_IntHelpCategory = 1
1871:   Call UpdateHelpScreen


  Exit Sub
ErrorHandler:
  HandleError True, "Label1_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Sub

Private Sub Label2_Click()
  On Error GoTo ErrorHandler

1882:   optUseAll.Value = True
1883:   m_IntHelpCategory = 3
1884:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "Label2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Label3_Click()
  On Error GoTo ErrorHandler

1894:   optSubSet.Value = True
1895:   m_IntHelpCategory = 3
1896:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "Label3_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Label4_Click()
  On Error GoTo ErrorHandler


1907:   If chkUsePatches.Value = 0 Then
1908:     chkUsePatches.Value = 1
1909:   Else
1910:     chkUsePatches.Value = 0
1911:   End If
1912:   m_IntHelpCategory = 2
1913:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "Label4_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbl_cbxCorridor_Click()
```

```
  On Error GoTo ErrorHandler


1924:   m_IntHelpCategory = 1
1925:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxCorridor_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbl_cbxHab2_Click()
  On Error GoTo ErrorHandler

1935:   m_IntHelpCategory = 1
1936:   Call UpdateHelpScreen


  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxHab2_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbl_cbxPatchField_Click()
  On Error GoTo ErrorHandler

1947:   m_IntHelpCategory = 3
1948:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxPatchField_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub lbl_cbxPatchLayer_Click()
  On Error GoTo ErrorHandler

1958:   m_IntHelpCategory = 2
1959:   Call UpdateHelpScreen
1960:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxPatchLayer_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Sub

Private Sub lbl_cbxValue_Click()
  On Error GoTo ErrorHandler

1970:   m_IntHelpCategory = 3
1971:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError False, "lbl_cbxValue_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optSubSet_Click()
  On Error GoTo ErrorHandler

1981:   Call EnablePatchOptions
1982:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "optSubSet_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optSubSet_GotFocus()
  On Error GoTo ErrorHandler

1992:   m_IntHelpCategory = 3
1993:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "optSubSet_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optUseAll_Click()
  On Error GoTo ErrorHandler

2003:   Call EnablePatchOptions
2004:   Call EnableOKButton

  Exit Sub
ErrorHandler:
```

```
  HandleError True, "optUseAll_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub optUseAll_GotFocus()
  On Error GoTo ErrorHandler

2014:   m_IntHelpCategory = 2
2015:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "optUseAll_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub txtValue_Change()
  On Error GoTo ErrorHandler


2026:   Call EnableOKButton

  Exit Sub
ErrorHandler:
  HandleError True, "txtValue_Change " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub txtValue_GotFocus()
  On Error GoTo ErrorHandler

2036:   m_IntHelpCategory = 3
2037:   Call UpdateHelpScreen

  Exit Sub
ErrorHandler:
  HandleError True, "txtValue_GotFocus " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Sub EnableOKButton()
  On Error GoTo ErrorHandler


  Dim booPolysOK As Boolean
2049:   booPolysOK = imgCheckWB1.Visible And imgCheckWB2.Visible And imgCheckSpCorr.Visible
```

```
  Dim booPatchOK As Boolean
2052:   booPatchOK = (chkUsePatches.Value = False) Or _
              ((cbxPatchLayer.ListIndex >= 1) And _
                          ((optUseAll.Value) Or _
                          ((cbxPatchField.ListIndex >= 1) And (txtValue.Text <> ""))))

2057:   cmdOK.Enabled = booPolysOK And booPatchOK

'  MsgBox "optUseAll.Value = " & optUseAll.Value & vbCrLf & _
'         "(cbxPatchField.ListIndex > 1) = " & (cbxPatchField.ListIndex > 1) & vbCrLf & _
'         "(txtValue.Text <> "") = " & (txtValue.Text <> "") & vbCrLf & _
'         "((optUseAll.Value) Or ((cbxPatchField.ListIndex > 1) And (txtValue.Text <> ..))) = " & _
'         ((optUseAll.Value) Or ((cbxPatchField.ListIndex > 1) And (txtValue.Text <> ""))) & vbCrLf



  Exit Sub
ErrorHandler:
  HandleError True, "EnableOKButton " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Form 14: progress_single.frm

```
VERSION 5.00
Begin VB.Form frmJenProgressPercent
   AutoRedraw      =   -1  'True
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Current Status..."
   ClientHeight    =   4785
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   5145
   Icon            =   "progress_single.frx":0000
   LinkTopic       =   "zzzzzzz"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   4785
   ScaleWidth      =   5145
   ShowInTaskbar   =   0   'False
   StartUpPosition =   1  'CenterOwner
   Begin VB.CheckBox chkClose
      Caption         =   "Close When Finished..."
      Height          =   285
      Left            =   75
      TabIndex        =   13
```

```
   Top             =   1965
   Width           =   2340
End
Begin VB.TextBox txtDetails
   BeginProperty Font
      Name            =   "Arial"
      Size            =   8.25
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   2430
   Left            =   45
   MultiLine       =   -1  'True
   ScrollBars      =   2  'Vertical
   TabIndex        =   8
   Text            =   "progress_single.frx":038A
   Top             =   2310
   Width           =   5055
End
Begin VB.CommandButton cmdDetails
   Caption         =   "Show Details"
   Height          =   315
   Left            =   2685
   TabIndex        =   7
   Top             =   1935
   Width           =   1320
End
Begin VB.PictureBox Picture1
   Height          =   390
   Left            =   3885
   ScaleHeight     =   330
   ScaleWidth      =   1200
   TabIndex        =   4
   Top             =   30
   Width           =   1260
   Begin VB.Label lblCurrentTime
      Alignment       =   2  'Center
      Caption         =   "12:21:15"
      Height          =   225
      Left            =   75
      TabIndex        =   5
      Top             =   75
      Width           =   1020
   End
```

```
End
Begin VB.PictureBox pct_barBorder
   AutoRedraw      =   -1  'True
   BackColor       =   &H80000000&
   Height          =   300
   Left            =   15
   Picture         =   "progress_single.frx":0390
   ScaleHeight     =   240
   ScaleWidth      =   4155
   TabIndex        =   3
   Top             =   1560
   Width           =   4215
   Begin VB.PictureBox icnProgressLine
      Appearance      =   0  'Flat
      AutoRedraw      =   -1  'True
      BackColor       =   &H80000005&
      BorderStyle     =   0  'None
      ForeColor       =   &H80000008&
      Height          =   105
      Left            =   15
      Picture         =   "progress_single.frx":3B58
      ScaleHeight     =   105
      ScaleMode       =   0  'User
      ScaleWidth      =   4125
      TabIndex        =   6
      Top             =   60
      Width           =   4125
   End
End
Begin VB.CommandButton cmdStop
   Caption         =   "Stop"
   Height          =   315
   Left            =   4050
   TabIndex        =   2
   Top             =   1935
   Width           =   1080
End
Begin VB.Label lblTimeLeft
   AutoSize        =   -1  'True
   Caption         =   "estimated time left"
   Height          =   195
   Left            =   90
   TabIndex        =   12
   Top             =   980
   Width           =   1260
End
Begin VB.Label lblIndex
```

```
      AutoSize        =   -1  'True
      Caption         =   "index"
      Height          =   195
      Left            =   90
      TabIndex        =   11
      Top             =   715
      Width           =   375
   End
   Begin VB.Label lblRecordNumber
      AutoSize        =   -1  'True
      Caption         =   "record number"
      Height          =   195
      Left            =   90
      TabIndex        =   10
      Top             =   450
      Width           =   1020
   End
   Begin VB.Label lblEstCompletionTime
      AutoSize        =   -1  'True
      Caption         =   "Estimated Completion Time:"
      Height          =   195
      Left            =   90
      TabIndex        =   9
      Top             =   1245
      Width           =   1950
   End
   Begin VB.Label lblPercentDone
      Caption         =   "( 0%)"
      Height          =   300
      Left            =   4395
      TabIndex        =   1
      Top             =   1560
      Width           =   705
   End
   Begin VB.Label lblBeginTime
      Caption         =   "12:21:15"
      Height          =   360
      Left            =   90
      TabIndex        =   0
      Top             =   150
      Width           =   3645
   End
End
Attribute VB_Name = "frmJenProgressPercent"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
```

```
Attribute VB_Exposed = False
Option Explicit
' PUT IN GENERAL DECLARATIONS SECTION
'Private Anchors As AnchorObjectList  ' Main anchor control object
Private mShouldContinue As Boolean
Private mBeginTime As Date
Private mRecordCount As Long
Private mRecordCountString As String
Private mCheckSecond As Date
Private mCheckProgress As Single
Private mExpanded As Boolean
Private m_AutoClose As Boolean
Private m_CapHeight As Long
Private m_Frame As IModelessFrame
' Variables used by the Error handler function - DO NOT REMOVE
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\progress_single.frm"


Public Function Frame() As IModelessFrame
  On Error GoTo ErrorHandler


22:   If m_Frame Is Nothing Then
23:     Set m_Frame = New ModelessFrame
24:     m_Frame.Create Me
25:     m_Frame.Caption = "Current Status:"
26:   End If

28:   Set Frame = m_Frame


  Exit Function
ErrorHandler:
  HandleError True, "Frame " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Private Sub cmdDetails_Click()
  On Error GoTo ErrorHandler

39:   mExpanded = Not mExpanded
40:   Call ExpandProgress

  Exit Sub
ErrorHandler:
  HandleError True, "cmdDetails_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub cmdStop_Click()
  On Error GoTo ErrorHandler

50:   mShouldContinue = False

  Exit Sub
ErrorHandler:
  HandleError True, "cmdStop_Click " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub cpCurrentTime_DragDrop(Source As Control, X As Single, Y As Single)
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
  HandleError False, "cpCurrentTime_DragDrop " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Form_Load()
  On Error GoTo ErrorHandler



' SAMPLE CODE
'  Dim theTotalCount As Long
'  theTotalCount = 160000
'
'  Dim theDescription As String
'  theDescription = "Counting to " & aml_func_mod.InsertCommas(theTotalCount) & "..."
'
'  Dim theBeginTime As Date
'  theBeginTime = Now
'
'  Dim ProgressMeter As frmProgress1
'  Set ProgressMeter = New frmProgress1
'
'  ProgressMeter.ShouldContinue = True
'  ProgressMeter.ProgBeginTime = theBeginTime
'  ProgressMeter.ProgRecCount = theTotalCount
'  ProgressMeter.lblCurrentTime.Caption = Format(Now, "ttttt")
'  ProgressMeter.lblBeginTime.Caption = "Began Job: " & Format(theBeginTime, "ttttt,  ddddd")
'
```

```
'  Dim theDetailedDescription As String
'  theDetailedDescription = "Testing Progress Meter..." & vbCrLf & _
'      "Began Job: " & Format(theBeginTime, "ttttt,  ddddd") & vbCrLf & _
'      "----------------------------------------" & vbCrLf
'  ProgressMeter.txtDetails.Text = theDetailedDescription
'
'  ProgressMeter.Show
'
'  Dim aNumber As Long
'
'  For aNumber = 1 To theTotalCount
'
'    If ProgressMeter.ShouldContinue = False Then Exit For
'
'    If aNumber Mod 1000 = 0 Then
'      theDetailedDescription = theDetailedDescription & _
'        " --> Working on Number " & aml_func_mod.InsertCommas(aNumber) & vbCrLf & _
'        "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf
'
'      ProgressMeter.txtDetails.Text = theDetailedDescription
'      ProgressMeter.txtDetails.SelStart = Len(theDetailedDescription)
'    End If
'
'    ProgressMeter.Est_Time_Left aNumber, theDescription
'    ProgressMeter.lblCurrentTime.Caption = Format(Now, "ttttt")
'
'  Next aNumber
'
'  MsgBox ProgressMeter.ShouldContinue
'
'  Unload ProgressMeter
'  Set ProgressMeter = Nothing



127:   m_CapHeight = Me.Height - Me.ScaleHeight

129:   mShouldContinue = True
130:   mCheckSecond = Now
131:   mCheckProgress = 0
'   MsgBox "Should set auto-close? " & CStr(m_AutoClose)
133:   If m_AutoClose Then
134:     chkClose.Value = 1
135:   Else
136:     chkClose.Value = 0
137:   End If
```

```
'  mExpanded = False
139:   Call Frame
140:   Call ExpandProgress

'  Dim pWindowPosition As IWindowPosition
'  Set pWindowPosition = m_Frame
'
'  pWindowPosition.Left = (Screen.Width / 2) - (pWindowPosition.Width / 3)
'  pWindowPosition.Top = (Screen.Height / 3) - (pWindowPosition.Height / 2)

148:   lblBeginTime.Caption = "Began Job: " & Format(Now, "ddddd ttttt")


  '  lblCurrentTime.SetLabel(date.now.setFormat("h:m:s AMPM").AsString)
152:   lblTimeLeft.Caption = ("Estimated time remaining: ---:---:---")
153:   lblEstCompletionTime.Caption = "Estimated completion time: -----------"
154:   lblPercentDone.Caption = "(00.0%)"
155:   lblRecordNumber.Caption = "-----"
156:   lblIndex.Caption = "-----"

158:   icnProgressLine.Width = 0              ' START GREEN PROGRESS BAR AT 0 PIXELS WIDE



  Exit Sub
ErrorHandler:
  HandleError True, "Form_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub


Public Sub Est_Time_Left(theRecordNumber As Long, theCurrentDescription As String, strRecNumHeader As String)
  On Error GoTo ErrorHandler

' zzzzzzz_EstTimeLeft

' ESTIMATED TIME LEFT CODE


  Dim thePercentDone As Single
177:   If mRecordCount <= 0 Then
178:     thePercentDone = 0
179:   Else
180:     thePercentDone = (theRecordNumber / mRecordCount)
181:   End If

  Dim thePercentForCalcs As Single
184:   thePercentForCalcs = Round(thePercentDone * 100, 2)
```

```
186:   If thePercentForCalcs >= mCheckProgress + 0.1 Then

188:     DoEvents

190:     mCheckProgress = mCheckProgress + 0.1

192:     lblIndex.Caption = theCurrentDescription
193:     If mRecordCount = 0 Then
194:       lblRecordNumber.Caption = strRecNumHeader
195:     Else
196:       lblRecordNumber.Caption = strRecNumHeader & aml_func_mod.InsertCommas(theRecordNumber) & _
             " of " + mRecordCountString & "..."
198:     End If

200:     icnProgressLine.Width = (icnProgressLine.Container.Width - 90) * thePercentDone
201:     icnProgressLine.Refresh

203:     lblPercentDone.Caption = "(" & thePercentForCalcs & "%)"
'    lblPercentDone.Refresh

    Dim theDuration As Long
207:     theDuration = DateDiff("s", mBeginTime, Now)

    Dim PredictedDuration As Long
210:     PredictedDuration = (theDuration * mRecordCount) / (theRecordNumber)

    Dim EstTimeLeft As Long
213:     EstTimeLeft = PredictedDuration - theDuration

    Dim EstFinish As String

217:     If DateDiff("s", mCheckSecond, Now) >= 1 Then
218:       lblCurrentTime.Caption = Format(Now, "ttttt")

220:       mCheckSecond = Now

222:       If PredictedDuration > 86400 Then
223:         EstFinish = Format(DateAdd("s", PredictedDuration, mBeginTime), "ttttt, ddddd")
224:       Else
225:         EstFinish = Format(DateAdd("s", PredictedDuration, mBeginTime), "ttttt")
226:       End If

228:       lblEstCompletionTime.Caption = "Estimated completion time: " & EstFinish

      Dim EstHoursLeft As Long
      Dim EstMinutesLeft As Long
```

```
      Dim EstSecondsLeft As Long

234:       EstHoursLeft = Fix(EstTimeLeft / 3600)
235:       EstMinutesLeft = Fix((EstTimeLeft - (EstHoursLeft * 3600)) / 60)
236:       EstSecondsLeft = Fix(EstTimeLeft - (EstHoursLeft * 3600) - (EstMinutesLeft * 60))

    '  If (EstMinutesLeft >= 10) Then
    '    EstMinutesStr = EstMinutesLeft.AsString
    '  Else
    '    EstMinutesStr = "0" + EstMinutesLeft.AsString
    '  End
    '  If (EstSecondsLeft >= 10) Then
    '    EstSecondsStr = EstSecondsLeft.AsString
    '  Else
    '    EstSecondsStr = "0" + EstSecondsLeft.AsString
    '  End

      Dim EstTimeLeftStr As String
250:       EstTimeLeftStr = EstHoursLeft & ":" & Format(EstMinutesLeft, "00") & ":" & Format(EstSecondsLeft, "00")
251:       lblTimeLeft.Caption = "Estimated time remaining:  " & EstTimeLeftStr
252:     End If
253:   Else
254:     lblIndex.Caption = theCurrentDescription
255:     lblRecordNumber.Caption = strRecNumHeader
256:   End If
257:   lblRecordNumber.Refresh
258:   pct_barBorder.Refresh
259:   Me.Refresh

  Exit Sub
ErrorHandler:
  HandleError True, "Est_Time_Left " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Property Get ShouldContinue() As Boolean
  On Error GoTo ErrorHandler

269:   ShouldContinue = mShouldContinue

  Exit Property
ErrorHandler:
  HandleError True, "ShouldContinue " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Let ShouldContinue(ByVal booContinue As Boolean)
```

```
  On Error GoTo ErrorHandler

279:   mShouldContinue = booContinue

  Exit Property
ErrorHandler:
  HandleError True, "ShouldContinue " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Let ProgBeginTime(ByVal dateBeginTime As Date)
  On Error GoTo ErrorHandler

289:   mBeginTime = dateBeginTime

  Exit Property
ErrorHandler:
  HandleError True, "ProgBeginTime " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Get ProgBeginTime() As Date
  On Error GoTo ErrorHandler

299:   ProgBeginTime = mBeginTime

  Exit Property
ErrorHandler:
  HandleError True, "ProgBeginTime " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let ProgRecCount(ByVal lngRecCount As Long)
  On Error GoTo ErrorHandler

308:   mRecordCount = lngRecCount
309:   mRecordCountString = aml_func_mod.InsertCommas(lngRecCount)

  Exit Property
ErrorHandler:
  HandleError True, "ProgRecCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Let SetExpanded(MakeExpanded As Boolean)

318:   mExpanded = MakeExpanded
```

```
End Property

Public Property Get SetExpanded() As Boolean

324:   SetExpanded = mExpanded

End Property

Public Property Let SetAutoClose(MakeAutoClose As Boolean)

330:   m_AutoClose = MakeAutoClose

End Property

Public Property Get SetAutoClose() As Boolean

336:   SetAutoClose = m_AutoClose

End Property
Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo ErrorHandler

342:   mRecordCountString = ""

  Exit Sub
ErrorHandler:
  HandleError True, "Form_Unload " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Public Sub ExpandProgress()
  On Error GoTo ErrorHandler

  Dim pWindowPosition As IWindowPosition
353:   Set pWindowPosition = m_Frame

355:   If mExpanded Then
356:     txtDetails.Visible = True
357:     pWindowPosition.Height = (4785 / 15) + (m_CapHeight / 15)
358:     cmdDetails.Caption = "Minimize"
359:   Else
360:     txtDetails.Visible = False
361:     pWindowPosition.Height = (2295 / 15) + (m_CapHeight / 15)
362:     cmdDetails.Caption = "Show Details"
363:   End If

  Exit Sub
```

```
ErrorHandler:
  HandleError True, "ExpandProgress " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 1: Anchor

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "Anchor"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
' Anchors
' Class: Anchor
' By neophile (n_e_o_p_h_i_l_e@yahoo.com)
' 5/28/2002
' ------------------------------------------------
' MODIFIED JANUARY 2006 TO ALLOW FOR DIFFERENT ANCHOR TYPES
' JEFF JENNESS (jeffj@jennessent.com)
'

Option Explicit

Public Enum AnchorTypes
   enumNone        ' Avenue Fastener ( - , - , - )
   enumStart       ' Avenue Fastener (Left/Top, - , - )
   enumStartSize   ' Avenue Fastener (Left/Top, Width/Height, - )
   enumStartEnd    ' Avenue Fastener (Left/Top, - , Right/Bottom )
   enumSize        ' Avenue Fastener ( - , Width/Height, - )
   enumSizeEnd     ' Avenue Fastener ( - , Width/Height, Right/Bottom)
   enumEnd         ' Avenue Fastener ( - , - , Right/Bottom)

   'atPosition  ' Anchor position (Left/Top)
   'atSize      ' Anchor size (Width/Height)
End Enum


Public AnchorType As AnchorTypes  ' Anchor type
```

```
Public MinValue As Long  ' Minimum value

Public MaxValue As Long  ' Maximum value

Public Value As Single  ' Relative distance


Private Sub Class_Initialize()
36:    MinValue = -&H7FFFFFFF  ' Set to max lower limit
37:    MaxValue = &H7FFFFFFF  ' Set to max upper limit
End Sub
```

Class 2: AnchorObject

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "AnchorObject"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
' Anchors
' Class: AnchorObject
' By neophile (n_e_o_p_h_i_l_e@yahoo.com)
' 5/28/2002
' ------------------------------------------------
' MODIFIED JANUARY 2006 TO ALLOW FOR DIFFERENT ANCHOR TYPES
' JEFF JENNESS (jeffj@jennessent.com)
'

Option Explicit

Private mCtl As Control    ' Control reference
Private mX As Anchor       ' X anchor
Private mY As Anchor       ' Y anchor
Private mX2 As Anchor      ' X2 anchor
Private mY2 As Anchor      ' Y2 anchor
Private mWidth As Anchor   ' Width anchor
Private mHeight As Anchor  ' Height anchor
```

```
'Private mWindowPos As IWindowPosition  ' FOR ESRI MODELESS FRAMES

'Public Property Set WindowPos(pWindowPos As IWindowPosition)
'  Set mWindowPos = pWindowPos
'End Property


Public Property Let Control(vData As Object)
27:    Set mCtl = vData  ' Set reference
End Property
Public Property Get Control() As Object
30:    Set Control = mCtl  ' Return reference
End Property

Public Property Get X() As Anchor
34:    Set X = mX  ' Return X anchor
End Property

Public Property Get Y() As Anchor
38:    Set Y = mY  ' Return Y anchor
End Property

Public Property Get X2() As Anchor
42:    Set X2 = mX2  ' Return X anchor
End Property

Public Property Get Y2() As Anchor
46:    Set Y2 = mY2  ' Return Y anchor
End Property

Public Sub SetAnchors(Optional ByVal XType As AnchorTypes, Optional ByVal YType As AnchorTypes)
  ' MODIFIED FOR IMODELESS FRAMES
'  MsgBox mWindowPos.Width & "    x    " & mWindowPos.Height


'   X.AnchorType = XType  ' Set X anchor type
'   Select Case XType  ' X anchor
'      Case enumNone        ' Avenue Fastener ( - , - , - )
'        mX.Value = mCtl.Left / mWindowPos.Width
'        mWidth.Value = mCtl.Width / mWindowPos.Width
'      Case enumStart       ' Avenue Fastener (Left, - , - )   DON'T ADJUST X AT ALL!
'        mX.Value = mCtl.Left
'        mWidth.Value = (mCtl.Width / mWindowPos.Width)
''        Debug.Print mCtl.Width
''        Debug.Print mCtl.Container.Width
''        Debug.Print (mCtl.Width / mCtl.Container.Width)
''        Debug.Print mWidth.Value
```

```
'      Case enumStartSize   ' Avenue Fastener (Left, Width/, - )
'        mX.Value = mCtl.Left
'        mWidth.Value = mCtl.Width
'      Case enumStartEnd    ' Avenue Fastener (Left, - , Right )
'        mX.Value = mCtl.Left
'        mWidth.Value = mWindowPos.Width - mCtl.Width
'      Case enumSize        ' Avenue Fastener ( - , Width, - )
'        mX.Value = ((mCtl.Left + (mCtl.Width / 2)) / mWindowPos.Width)
'        mX2.Value = mCtl.Width / 2
'        mWidth.Value = mCtl.Width
'      Case enumSizeEnd     ' Avenue Fastener ( - , Width, Right)
'        mX.Value = mWindowPos.Width - mCtl.Left
'        mWidth.Value = mCtl.Width
'      Case enumEnd         ' Avenue Fastener ( - , - , Right)
'        mX.Value = mCtl.Width / mWindowPos.Width
'        mX2.Value = mWindowPos.Width - mCtl.Width - mCtl.Left
'        mWidth.Value = mCtl.Width / mWindowPos.Width
'   End Select
'   Y.AnchorType = YType  ' Set Y anchor type
'   Select Case YType  ' Y anchor
'      Case enumNone        ' Avenue Fastener ( - , - , - )
'        mY.Value = mCtl.Top / mWindowPos.Height
'        mHeight.Value = mCtl.Height / mWindowPos.Height
'      Case enumStart       ' Avenue Fastener (Top, - , - )   DON'T ADJUST Y AT ALL!
'        mY.Value = mCtl.Top
'        mHeight.Value = mCtl.Height / mWindowPos.Height
'      Case enumStartSize   ' Avenue Fastener (Top, Height, - )
'        mY.Value = mCtl.Top
'      Case enumStartEnd    ' Avenue Fastener (Top, - , Bottom )
'        mY.Value = mCtl.Top
'        mHeight.Value = mWindowPos.Height - mCtl.Height
'      Case enumSize        ' Avenue Fastener ( - , Height, - )
'        mY.Value = ((mCtl.Top + (mCtl.Height / 2)) / mWindowPos.Height)
'        mY2.Value = mCtl.Height / 2
'      Case enumSizeEnd     ' Avenue Fastener ( - , Height, Bottom)
'        mY.Value = mWindowPos.Height - mCtl.Top
'      Case enumEnd         ' Avenue Fastener ( - , - , Bottom)
'        mY.Value = mCtl.Height / mWindowPos.Height
'        mY2.Value = mWindowPos.Height - mCtl.Height - mCtl.Top
'        mHeight.Value = mCtl.Height / mWindowPos.Height
'   End Select


'

111:    X.AnchorType = XType  ' Set X anchor type
   Select Case XType  ' X anchor
```

```
      Case enumNone        ' Avenue Fastener ( - , - , - )
114:         mX.Value = mCtl.Left / mCtl.Container.Width
115:         mWidth.Value = mCtl.Width / mCtl.Container.Width
      Case enumStart       ' Avenue Fastener (Left, - , - )   DON'T ADJUST X AT ALL!
117:         mX.Value = mCtl.Left
118:         mWidth.Value = (mCtl.Width / mCtl.Container.Width)
'        Debug.Print mCtl.Width
'        Debug.Print mCtl.Container.Width
'        Debug.Print (mCtl.Width / mCtl.Container.Width)
'        Debug.Print mWidth.Value
      Case enumStartSize   ' Avenue Fastener (Left, Width/, - )
124:         mX.Value = mCtl.Left
125:         mWidth.Value = mCtl.Width
      Case enumStartEnd    ' Avenue Fastener (Left, - , Right )
127:         mX.Value = mCtl.Left
128:         mWidth.Value = mCtl.Container.Width - mCtl.Width
      Case enumSize        ' Avenue Fastener ( - , Width, - )
130:         mX.Value = ((mCtl.Left + (mCtl.Width / 2)) / mCtl.Container.Width)
131:         mX2.Value = mCtl.Width / 2
132:         mWidth.Value = mCtl.Width
      Case enumSizeEnd     ' Avenue Fastener ( - , Width, Right)
134:         mX.Value = mCtl.Container.Width - mCtl.Left
135:         mWidth.Value = mCtl.Width
      Case enumEnd         ' Avenue Fastener ( - , - , Right)
137:         mX.Value = mCtl.Width / mCtl.Container.Width
138:         mX2.Value = mCtl.Container.Width - mCtl.Width - mCtl.Left
139:         mWidth.Value = mCtl.Width / mCtl.Container.Width
140:    End Select
141:    Y.AnchorType = YType  ' Set Y anchor type
   Select Case YType  ' Y anchor
      Case enumNone        ' Avenue Fastener ( - , - , - )
144:         mY.Value = mCtl.Top / mCtl.Container.Height
145:         mHeight.Value = mCtl.Height / mCtl.Container.Height
      Case enumStart       ' Avenue Fastener (Top, - , - )   DON'T ADJUST Y AT ALL!
147:         mY.Value = mCtl.Top
148:         mHeight.Value = mCtl.Height / mCtl.Container.Height
      Case enumStartSize   ' Avenue Fastener (Top, Height, - )
150:         mY.Value = mCtl.Top
      Case enumStartEnd    ' Avenue Fastener (Top, - , Bottom )
152:         mY.Value = mCtl.Top
153:         mHeight.Value = mCtl.Container.Height - mCtl.Height
      Case enumSize        ' Avenue Fastener ( - , Height, - )
155:         mY.Value = ((mCtl.Top + (mCtl.Height / 2)) / mCtl.Container.Height)
156:         mY2.Value = mCtl.Height / 2
      Case enumSizeEnd     ' Avenue Fastener ( - , Height, Bottom)
158:         mY.Value = mCtl.Container.Height - mCtl.Top
      Case enumEnd         ' Avenue Fastener ( - , - , Bottom)
```

```
160:         mY.Value = mCtl.Height / mCtl.Container.Height
161:         mY2.Value = mCtl.Container.Height - mCtl.Height - mCtl.Top
162:         mHeight.Value = mCtl.Height / mCtl.Container.Height
163:    End Select



'   Select Case YType  ' Y anchor
'      Case atPosition  ' Get position
'         mY.Value = mCtl.Container.Height - mCtl.Top  ' Control's top relative to form's bottom
'      Case atSize  ' Get size
'         mY.Value = mCtl.Container.Height - mCtl.Height  ' Control's bottom relative to form's bottom
'   End Select
End Sub

Public Sub DoAnchors()
   On Error Resume Next  ' Ignore errors


'
'   Select Case mX.AnchorType  ' X anchor
'      Case enumNone        ' Avenue Fastener ( - , - , - )
'   ' MODIFIED FOR ESRI MODELESS FRAMES
'        mCtl.Left = mWindowPos.Width * mX.Value
'        mCtl.Width = mWindowPos.Width * mWidth.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'      Case enumStart       ' Avenue Fastener (Left, - , - )   DON'T ADJUST Y AT ALL!
''        mCtl.Left = mX.Value
'        mCtl.Width = mWindowPos.Width * mWidth.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'      Case enumStartSize   ' Avenue Fastener (Left, Width, - )
''        mCtl.Left = mX.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'      Case enumStartEnd    ' Avenue Fastener (Left, - , Right )
''        mCtl.Left = mX.Value
'        mCtl.Width = mWindowPos.Width - mWidth.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'      Case enumSize        ' Avenue Fastener ( - , Width, - )
'        mCtl.Left = (mWindowPos.Width * mX.Value) - mX2.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'      Case enumSizeEnd     ' Avenue Fastener ( - , Width, Right)
```

```
'        mCtl.Left = mWindowPos.Width - mX.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'      Case enumEnd         ' Avenue Fastener ( - , - , Right)
'        mCtl.Left = mWindowPos.Width - (mWindowPos.Width * mX.Value) - mX2.Value
'        mCtl.Width = mWindowPos.Width * mWidth.Value
''         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
''         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
'   End Select
'   Select Case mY.AnchorType  ' Y anchor
'      Case enumNone        ' Avenue Fastener ( - , - , - )
'        mCtl.Top = mWindowPos.Height * mY.Value
'        mCtl.Height = mWindowPos.Height * mHeight.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'      Case enumStart       ' Avenue Fastener (Top, - , - )   DON'T ADJUST Y AT ALL!
''        mCtl.Top = mY.Value
'        mCtl.Height = mWindowPos.Height * mHeight.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'      Case enumStartSize   ' Avenue Fastener (Top, Height, - )
''        mCtl.Top = mY.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'      Case enumStartEnd    ' Avenue Fastener (Top, - , Bottom )
''        mCtl.Top = mY.Value
'        mCtl.Height = mWindowPos.Height - mHeight.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'      Case enumSize        ' Avenue Fastener ( - , Height, - )
'        mCtl.Top = (mWindowPos.Height * mY.Value) - mY2.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'      Case enumSizeEnd     ' Avenue Fastener ( - , Height, Bottom)
'        mCtl.Top = mWindowPos.Height - mY.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'      Case enumEnd         ' Avenue Fastener ( - , - , Bottom)
'        mCtl.Top = mWindowPos.Height - (mWindowPos.Height * mY.Value) - mY2.Value
'        mCtl.Height = mWindowPos.Height * mHeight.Value
''         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
''         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
'   End Select


   Select Case mX.AnchorType  ' X anchor
      Case enumNone        ' Avenue Fastener ( - , - , - )
```

```
254:         mCtl.Left = mCtl.Container.Width * mX.Value
255:         mCtl.Width = mCtl.Container.Width * mWidth.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
      Case enumStart       ' Avenue Fastener (Left, - , - )   DON'T ADJUST Y AT ALL!
'        mCtl.Left = mX.Value
260:         mCtl.Width = mCtl.Container.Width * mWidth.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
      Case enumStartSize   ' Avenue Fastener (Left, Width, - )
'        mCtl.Left = mX.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
      Case enumStartEnd    ' Avenue Fastener (Left, - , Right )
'        mCtl.Left = mX.Value
269:         mCtl.Width = mCtl.Container.Width - mWidth.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
      Case enumSize        ' Avenue Fastener ( - , Width, - )
273:         mCtl.Left = (mCtl.Container.Width * mX.Value) - mX2.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
      Case enumSizeEnd     ' Avenue Fastener ( - , Width, Right)
277:         mCtl.Left = mCtl.Container.Width - mX.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
      Case enumEnd         ' Avenue Fastener ( - , - , Right)
281:         mCtl.Left = mCtl.Container.Width - (mCtl.Container.Width * mX.Value) - mX2.Value
282:         mCtl.Width = mCtl.Container.Width * mWidth.Value
'         If mCtl.Left < mX.MinValue Then mCtl.Left = mX.MinValue  ' Lower limit
'         If mCtl.Left > mX.MaxValue Then mCtl.Left = mX.MaxValue  ' Upper limit
285:    End Select
   Select Case mY.AnchorType  ' Y anchor
      Case enumNone        ' Avenue Fastener ( - , - , - )
288:         mCtl.Top = mCtl.Container.Height * mY.Value
289:         mCtl.Height = mCtl.Container.Height * mHeight.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
      Case enumStart       ' Avenue Fastener (Top, - , - )   DON'T ADJUST Y AT ALL!
'        mCtl.Top = mY.Value
294:         mCtl.Height = mCtl.Container.Height * mHeight.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
      Case enumStartSize   ' Avenue Fastener (Top, Height, - )
'        mCtl.Top = mY.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
```

```
      Case enumStartEnd    ' Avenue Fastener (Top, - , Bottom )
'        mCtl.Top = mY.Value
303:         mCtl.Height = mCtl.Container.Height - mHeight.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
      Case enumSize        ' Avenue Fastener ( - , Height, - )
307:         mCtl.Top = (mCtl.Container.Height * mY.Value) - mY2.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
      Case enumSizeEnd     ' Avenue Fastener ( - , Height, Bottom)
311:         mCtl.Top = mCtl.Container.Height - mY.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
      Case enumEnd         ' Avenue Fastener ( - , - , Bottom)
315:         mCtl.Top = mCtl.Container.Height - (mCtl.Container.Height * mY.Value) - mY2.Value
316:         mCtl.Height = mCtl.Container.Height * mHeight.Value
'         If mCtl.Top < mY.MinValue Then mCtl.Top = mY.MinValue  ' Lower limit
'         If mCtl.Top > mY.MaxValue Then mCtl.Top = mY.MaxValue  ' Upper limit
319:    End Select


   On Error GoTo 0  ' Stop ignoring errors
End Sub


Private Sub Class_Initialize()
327:    Set mX = New Anchor       ' Create new anchor instance
328:    Set mY = New Anchor       ' Create new anchor instance
329:    Set mX2 = New Anchor      ' Create new anchor instance
330:    Set mY2 = New Anchor      ' Create new anchor instance
331:    Set mWidth = New Anchor   ' Create new anchor instance
332:    Set mHeight = New Anchor  ' Create new anchor instance
'   Set mWindowPos = New ModelessFrame
End Sub

Private Sub Class_Terminate()
337:    Set mX = Nothing       ' Discard anchor instance
338:    Set mY = Nothing       ' Discard anchor instance
339:    Set mX2 = Nothing      ' Discard anchor instance
340:    Set mY2 = Nothing      ' Discard anchor instance
341:    Set mWidth = Nothing   ' Discard anchor instance
342:    Set mHeight = Nothing  ' Discard anchor instance
'   Set mWindowPos = Nothing
End Sub
```

Class 3: AnchorObjectList

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "AnchorObjectList"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
' Anchors
' Class: AnchorObjectList
' By neophile (n_e_o_p_h_i_l_e@yahoo.com)
' 5/28/2002
' ------------------------------------------------
' MODIFIED JANUARY 2006 TO ALLOW FOR DIFFERENT ANCHOR TYPES
' JEFF JENNESS (jeffj@jennessent.com)
'

Option Explicit

Private Declare Function LockWindowUpdate Lib "user32" (ByVal hwndLock As Long) As Long

Private WithEvents mForm As Form  ' Form reference
Attribute mForm.VB_VarHelpID = -1
Private mCol As Collection  ' Item collection


Public Property Let Form(vData As Object)
19:    Set mForm = vData  ' Set reference
End Property
Public Property Get Form() As Object
22:    Set Form = mForm   ' Return reference
End Property

Public Function Count() As Long
26:    Count = mCol.Count  ' Return anchor collection count
End Function

Public Function Item(Control As Object) As AnchorObject
   Dim lIdx As Long
31:    lIdx = IndexOf(Control)  ' Get position in item collection
32:    If lIdx = 0 Then  ' If no item was found...
```

```
33:       Set Item = New AnchorObject  ' ...create a new item
34:       Item.Control = Control  ' Set reference
35:       mCol.Add Item  ' Add item to collection
36:    Else
37:       Set Item = mCol(IndexOf(Control))  ' Return item from collection
38:    End If
End Function

Public Function IndexOf(Control As Object) As Long
   Dim l As Long
43:    If mCol.Count > 0 Then  ' If there are any items...
44:       For l = 1 To mCol.Count  ' ...loop through them
45:          If mCol(l) Is Control Then  ' If the refrences match...
46:             IndexOf = l  ' ...return its position
47:             Exit For  ' Stop looping
48:          End If
49:       Next
50:    End If
End Function

Public Sub Remove(Control As Object)
54:    mCol.Remove IndexOf(Control)  ' Remove item from collection
End Sub

Public Sub SetAnchors()
   Dim oAO As AnchorObject
59:    For Each oAO In mCol  ' Loop through items
60:       oAO.SetAnchors  ' Set both anchors
61:    Next
End Sub

Public Sub DoAnchors()
   Dim oAO As AnchorObject
'   If Not (mForm Is Nothing) Then Call LockWindowUpdate(mForm.hWnd)  ' Lock repainting
67:    For Each oAO In mCol  ' Loop through items
68:       oAO.DoAnchors  ' Do both anchors
69:    Next
'   Call LockWindowUpdate(0)  ' Unlock repainting
End Sub


Private Sub mForm_Resize()
75:    Me.DoAnchors  ' Do all anchors
End Sub


Private Sub Class_Initialize()
```

```
80:    Set mCol = New Collection  ' Create new collection
End Sub

Private Sub Class_Terminate()
84:    Set mCol = Nothing  ' Discard collection
End Sub
```

Class 4: clsBottleneckDemo

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsBottleneckDemo"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Const c_sModuleFileName As String = ""
Implements ICommand
Private Sub Class_Initialize()
  On Error GoTo ErrorHandler


14:   Set m_bitmap = LoadResPicture(117, vbResBitmap)



  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub Class_Terminate()
  On Error GoTo ErrorHandler


27:   Set m_bitmap = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
37:   If (Not m_ExtensionConfig Is Nothing) Then
38:     If (m_ExtensionConfig.State = esriESEnabled) Then
39:       ICommand_Enabled = True
40:     End If
41:   Else
42:     ICommand_Enabled = False
43:   End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
54:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
65:   ICommand_Name = "CorridorDesigner_BottleneckStatsDemo"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
76:   ICommand_Caption = "Calculate Bottleneck Statistics - Demo"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
87:   ICommand_Tooltip = "Bottleneck Statistics - Demo"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
98:   ICommand_Message = "Calculate Bottleneck Statistics - Demo"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
109:   ICommand_HelpFile = ""
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
120:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
131:   ICommand_Bitmap = m_bitmap.Handle

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
142:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
```

```
154:   Set m_App = hook
155:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
158:   newUid.Value = "Linkages.Extension"
159:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)

' 153:   SetWindowLong m_frm1.hwnd, GWL_HWNDPARENT, m_App.hwnd
'  m_frm1.Frame

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

    ' ------------------------------------------------------------------
'  MsgBox "<-- Code in development -->" & vbCrLf & vbCrLf & _
        "The following dialogs behave as they will when the code is finished, but will not actually " & _
        "perform the bottleneck analysis.  The 'Bottleneck Results' dialog has full functionality, " & _
        "but will display a sample dataset." & vbCrLf & vbCrLf & _
        "Please check http://www.corridordesign.org for updates...", , "Function Not Implemented Yet:"

  Dim frmBottleneck As Linkages.frmBottleneck
  Dim frmSelect As Linkages.frmSelScreen

  ' CLOSE EXISTING OPEN DIALOGS
  Dim ext As Linkages.Extension
186:   Set ext = m_ExtensionConfig
187:   If Not ext.BottleneckForm Is Nothing Then
188:     Set frmBottleneck = ext.BottleneckForm
189:     Unload frmBottleneck
190:     Set ext.BottleneckForm = Nothing
191:   End If
192:   If Not ext.aSelForm Is Nothing Then
193:     Set frmSelect = ext.aSelForm
194:     Unload frmSelect
195:     Set ext.aSelForm = Nothing
196:   End If

  ' MAKE NEW STEP 1 FORM
```

```
199:   Set frmBottleneck = New Linkages.frmBottleneck
200:   Set ext.BottleneckForm = frmBottleneck
201:   Set frmBottleneck.ArcApplication = m_App
202:   Set frmBottleneck.Doc = m_MxDoc
203:   Load frmBottleneck
204:   frmBottleneck.Frame.Caption = "Describe Bottlenecks:"
205:   frmBottleneck.Frame.Visible = True
206:   frmBottleneck.IsDemo = True

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 5: clsToolBar

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsToolBar"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

' Variables used by the Error handler function - DO NOT REMOVE

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\clsToolBar"       ' Constant reflect file module
name
Implements IToolBarDef

Private Property Get IToolBarDef_ItemCount() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
12:    IToolBarDef_ItemCount = 12
```

```
  Exit Property
ErrorHandler:
  HandleError True, "IToolBarDef_ItemCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub IToolBarDef_GetItemInfo(ByVal pos As Long, ByVal itemDef As esriSystemUI.IItemDef)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
    Select Case pos
    Case 0
25:       itemDef.ID = "Linkages.cmdCompare"
    Case 1
27:       itemDef.ID = "Linkages.cmdBottleneck"
    Case 2
29:       itemDef.ID = "Linkages.cmdSumMod"
    Case 3
31:       itemDef.ID = "linkages.cmdHabSuitStats"
    Case 4
33:       itemDef.ID = "Linkages.cmdClip"
34:       itemDef.Group = True
    Case 5
36:       itemDef.ID = "Linkages.cmdHistogramStats"
    Case 6
38:       itemDef.ID = "Linkages.cmdDeleteCorGraphics"
    Case 7
40:       itemDef.ID = "Linkages.cmdNewShapefile"
    Case 8
42:       itemDef.ID = "Linkages.cmdOpenTable"
    Case 9
44:       itemDef.ID = "Linkages.toolReturnCoords"
45:       itemDef.Group = True
    Case 10
47:       itemDef.ID = "Linkages.toolDrawPoly"
    Case 11
49:       itemDef.ID = "Linkages.cmdAbout"
50:       itemDef.Group = True

52:   End Select

  Exit Sub
ErrorHandler:
  HandleError True, "IToolBarDef_GetItemInfo " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Property Get IToolBarDef_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
63:   IToolBarDef_Name = "CorridorDesigner_CDToolBar"

  Exit Property
ErrorHandler:
  HandleError True, "IToolBarDef_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get IToolBarDef_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
74:   IToolBarDef_Caption = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "IToolBarDef_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

Class 6: clsToolBarForConference

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsToolBarForConference"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

' Variables used by the Error handler function - DO NOT REMOVE

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\clsToolBarForConference"       ' Constant
reflect file module name
Implements IToolBarDef
```

```
Private Property Get IToolBarDef_ItemCount() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
12:    IToolBarDef_ItemCount = 4

  Exit Property
ErrorHandler:
  HandleError True, "IToolBarDef_ItemCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub IToolBarDef_GetItemInfo(ByVal pos As Long, ByVal itemDef As esriSystemUI.IItemDef)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
    Select Case pos
    Case 0
25:       itemDef.ID = "Linkages.cmdCompare"
'    Case 1
'      itemDef.ID = "Linkages.cmdSumMod"
'    Case 2
'      itemDef.ID = "Linkages.cmdHistogramStats"
'    Case 3
'      itemDef.ID = "Linkages.cmdClip"
    Case 1
33:       itemDef.ID = "Linkages.toolReturnCoords"
34:       itemDef.Group = True
    Case 2
36:       itemDef.ID = "Linkages.toolDrawPoly"
    Case 3
38:       itemDef.ID = "Linkages.cmdAbout"
39:       itemDef.Group = True
'    Case 7
'      itemDef.ID = "Linkages.cmdTestCode"
'      itemDef.Group = True
'    Case 4
'      itemDef.ID = "SaguaroOpenTable.cmdQuickLoad"
'    Case 5
'      itemDef.ID = "SaguaroOpenTable.cmdSelByAttSaguaro"
'      itemDef.Group = True
'    Case 6
'      itemDef.ID = "SaguaroOpenTable.cmdSelByLocation"
'    Case 7
'      itemDef.ID = "SaguaroOpenTable.cmdSelByGraphic"
'    Case 8
'      itemDef.ID = "SaguaroOpenTable.cmdSelectNone"
```

```
'      itemDef.Group = True
'    Case 9
'      itemDef.ID = "SaguaroOpenTable.cmdSelectSwitch"
'    Case 10
'      itemDef.ID = "SaguaroOpenTable.cmdDeleteSelLayers"
'      itemDef.Group = True
'    Case 11
'      itemDef.ID = "SaguaroOpenTable.cmdAbout"
'      itemDef.Group = True
63:   End Select

  Exit Sub
ErrorHandler:
  HandleError True, "IToolBarDef_GetItemInfo " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get IToolBarDef_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
74:   IToolBarDef_Name = "CorridorDesigner_ConferenceToolBar"

  Exit Property
ErrorHandler:
  HandleError True, "IToolBarDef_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get IToolBarDef_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
85:   IToolBarDef_Caption = "Corridor Designer - ESRI Conference 2007"

  Exit Property
ErrorHandler:
  HandleError True, "IToolBarDef_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

Class 7: cmdAbout

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
```

```
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdAbout"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

' Variables used by the Error handler function - DO NOT REMOVE

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Const c_sModuleFileName As String = ""       ' Constant reflect file module name
Implements ICommand

Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

16:   Set m_bitmap = LoadResPicture(106, vbResBitmap)

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

26:   Set m_bitmap = Nothing
27:   Set m_MxDoc = Nothing
28:   Set m_App = Nothing
29:   Set m_ExtensionConfig = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Property Get ICommand_Enabled() As Boolean
```

```
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
39:   If (Not m_ExtensionConfig Is Nothing) Then
40:     If (m_ExtensionConfig.State = esriESEnabled) Then
41:       ICommand_Enabled = True
42:     End If
43:   Else
44:     ICommand_Enabled = False
45:   End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
56:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
67:   ICommand_Name = "CorridorDesigner_About"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
78:   ICommand_Caption = "About Corridor Designer and Manual"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
89:   ICommand_Tooltip = "About Corridor Designer and Manual"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
100:   ICommand_Message = "About Corridor Designer"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
111:   ICommand_HelpFile = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
122:   ICommand_HelpContextID = 0
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
133:   ICommand_Bitmap = m_bitmap.Handle

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
144:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

156:   Set m_App = hook
157:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
160:   newUid.Value = "Linkages.Extension"
161:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
172:   Linkages.frmAbout.Show vbModal

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 8: cmdBottleneck

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdBottleneck"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdBottleneck.cls"
Implements ICommand
Private Sub Class_Initialize()
  On Error GoTo ErrorHandler


14:   Set m_bitmap = LoadResPicture(113, vbResBitmap)



  Exit Sub
```

```
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler


27:   Set m_bitmap = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
37:   If (Not m_ExtensionConfig Is Nothing) Then
38:     If (m_ExtensionConfig.State = esriESEnabled) Then
39:       ICommand_Enabled = True
40:     End If
41:   Else
42:     ICommand_Enabled = False
43:   End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
54:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
```

```
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
65:   ICommand_Name = "CorridorDesigner_BottleneckStats"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
76:   ICommand_Caption = "Calculate Bottleneck Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
87:   ICommand_Tooltip = "Bottleneck Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
98:   ICommand_Message = "Calculate Bottleneck Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

```
Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
109:   ICommand_HelpFile = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
120:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
131:   ICommand_Bitmap = m_bitmap.Handle

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
142:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

```
Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

154:   Set m_App = hook
155:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
158:   newUid.Value = "Linkages.Extension"
159:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)

' 153:   SetWindowLong m_frm1.hwnd, GWL_HWNDPARENT, m_App.hwnd
'  m_frm1.Frame

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

    ' ------------------------------------------------------------------
'  MsgBox "<-- Code in development -->" & vbCrLf & vbCrLf & _
        "The following dialogs behave as they will when the code is finished, but will not actually " & _
        "perform the bottleneck analysis.  The 'Bottleneck Results' dialog has full functionality, " & _
        "but will display a sample dataset." & vbCrLf & vbCrLf & _
        "Please check http://www.corridordesign.org for updates...", , "Function Not Implemented Yet:"

  Dim frmBottleneck As Linkages.frmBottleneck
  Dim frmSelect As Linkages.frmSelScreen

  ' CLOSE EXISTING OPEN DIALOGS
  Dim ext As Linkages.Extension
186:   Set ext = m_ExtensionConfig
187:   If Not ext.BottleneckForm Is Nothing Then
188:     Set frmBottleneck = ext.BottleneckForm
189:     Unload frmBottleneck
190:     Set ext.BottleneckForm = Nothing
191:   End If
192:   If Not ext.aSelForm Is Nothing Then
193:     Set frmSelect = ext.aSelForm
```

```
194:     Unload frmSelect
195:     Set ext.aSelForm = Nothing
196:   End If

  ' MAKE NEW STEP 1 FORM
199:   Set frmBottleneck = New Linkages.frmBottleneck
200:   Set ext.BottleneckForm = frmBottleneck
201:   Set frmBottleneck.ArcApplication = m_App
202:   Set frmBottleneck.Doc = m_MxDoc
203:   Load frmBottleneck
204:   frmBottleneck.Frame.Caption = "Describe Bottlenecks:"
205:   frmBottleneck.Frame.Visible = True
206:   frmBottleneck.IsDemo = False

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 9: cmdClip

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdClip"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ClipForm As Linkages.frmClip
Private m_ExtensionConfig As IExtensionConfig

Implements ICommand
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdClip.cls"
```

```
Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

16:   Set m_bitmap = LoadResPicture(110, vbResBitmap)


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

27:   Set m_ClipForm = Nothing
28:   Set m_bitmap = Nothing
29:   Set m_ExtensionConfig = Nothing
30:   Set m_App = Nothing
31:   Set m_MxDoc = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

42:   If (Not m_ExtensionConfig Is Nothing) Then
43:     If (m_ExtensionConfig.State = esriESEnabled) Then
44:       ICommand_Enabled = True
45:     End If
46:   Else
47:     ICommand_Enabled = False
48:   End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
```

```
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
60:    ICommand_Checked = False


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
73:   ICommand_Name = "CorridorDesigner_ClipLayer"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
86:   ICommand_Caption = "Clip Layer"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
99:   ICommand_Tooltip = "Clip Layer"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
112:   ICommand_Message = "Clip layer to polygon..."


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
125:   ICommand_HelpFile = ""


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
138:   ICommand_HelpContextID = 0


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
151:   ICommand_Bitmap = m_bitmap.Handle


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
164:   ICommand_Category = "Corridor Designer Tools"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler


176:   Set m_App = hook
177:   Set m_MxDoc = m_App.Document
  Dim newUid As New uID
179:   newUid.Value = "Linkages.Extension"
180:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub ICommand_OnClick()

'  MsgBox "First Line of Code; Click Event about to start (Step 1)..."  ' ************************************************************

  On Error GoTo ErrorHandler

  Dim newUid As New uID
195:   newUid.Value = "Linkages.Extension"
196:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
198:   Set ext = m_ExtensionConfig
199:   Set m_ClipForm = New Linkages.frmClip
200:   Set m_ClipForm.ArcApplication = m_App
201:   Set m_ClipForm.Doc = m_MxDoc
202:   Set ext.PolyCorridor = Nothing
203:   Load m_ClipForm

205:   Set ext.frmClipForm = m_ClipForm

207:   m_ClipForm.Frame.Caption = "Clip Data to Corridor:"
208:   m_ClipForm.Frame.Visible = True

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 10: cmdCompare

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdCompare"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
```

```
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdCompare.cls"
Implements ICommand
Private Sub Class_Initialize()
  On Error GoTo ErrorHandler


14:   Set m_bitmap = LoadResPicture(114, vbResBitmap)



  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler


27:   Set m_bitmap = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
37:   If (Not m_ExtensionConfig Is Nothing) Then
38:     If (m_ExtensionConfig.State = esriESEnabled) Then
39:       ICommand_Enabled = True
40:     End If
41:   Else
42:     ICommand_Enabled = False
43:   End If
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
54:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
65:   ICommand_Name = "CorridorDesigner_PatchStats"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
76:   ICommand_Caption = "Calculate Patch Distance Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
```

```
87:   ICommand_Tooltip = "Patch Distance Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
98:   ICommand_Message = "Calculate Patch Distance Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
109:   ICommand_HelpFile = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
120:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
131:   ICommand_Bitmap = m_bitmap.Handle

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
142:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

154:   Set m_App = hook
155:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
158:   newUid.Value = "Linkages.Extension"
159:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)

' 153:   SetWindowLong m_frm1.hwnd, GWL_HWNDPARENT, m_App.hwnd
'  m_frm1.Frame

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
```

```
'  MsgBox "<-- Code in development -->" & vbCrLf & _
'         "Please check http://www.corridordesign.org for updates...", , "Function Not Implemented Yet:"
'  Exit Sub

  Dim frmStep1 As Jennessent_CompareParameters
  Dim frmSelect As Linkages.frmSelScreen

  ' CLOSE EXISTING OPEN DIALOGS
  Dim ext As Linkages.Extension
183:   Set ext = m_ExtensionConfig
184:   If Not ext.frmStep1 Is Nothing Then
185:     Set frmStep1 = ext.frmStep1
186:     Unload frmStep1
187:     Set ext.frmStep1 = Nothing
188:   End If
189:   If Not ext.aSelForm Is Nothing Then
190:     Set frmSelect = ext.aSelForm
191:     Unload frmSelect
192:     Set ext.aSelForm = Nothing
193:   End If

  ' MAKE NEW STEP 1 FORM
196:   Set frmStep1 = New Jennessent_CompareParameters
197:   Set ext.frmStep1 = frmStep1
198:   Set frmStep1.ArcApplication = m_App
199:   Set frmStep1.Doc = m_MxDoc
200:   Load frmStep1
201:   frmStep1.Frame.Caption = "Describe Patch-to-Patch Distances:"
202:   frmStep1.Frame.Visible = True

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 11: cmdDeleteCorGraphics

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdDeleteCorGraphics"
```

```
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Implements ICommand
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdDeleteCorGraphics.cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

15:   Set m_bitmap = LoadResPicture(111, vbResBitmap)

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

25:   Set m_bitmap = Nothing
26:   Set m_MxDoc = Nothing
27:   Set m_App = Nothing
28:   Set m_ExtensionConfig = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  Dim booHasExtension As Boolean
39:   If (Not m_ExtensionConfig Is Nothing) Then
40:     If (m_ExtensionConfig.State = esriESEnabled) Then
41:       booHasExtension = True
```

```
42:     End If
43:   Else
44:     booHasExtension = False
45:   End If


48:   ICommand_Enabled = booHasExtension


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
61:    ICommand_Checked = False


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
74:   ICommand_Name = "CorridorDesigner_DeleteCorridorGraphics"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
87:   ICommand_Caption = "Delete CD Graphics"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
100:   ICommand_Tooltip = "Delete Corridor Designer Graphics"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
113:   ICommand_Message = "Delete all graphics created by Corridor Designer functions..."


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
126:   ICommand_HelpFile = ""


  Exit Property
```

```
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
139:   ICommand_HelpContextID = 0


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
152:   ICommand_Bitmap = m_bitmap.Handle


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
165:   ICommand_Category = "Corridor Designer Tools"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

```
Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler


177:   Set m_App = hook
178:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
181:   newUid.Value = "Linkages.Extension"
182:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

'  Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors")
'  Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "delete_corridors_orig")
'  Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_MxDoc, "Route_Graphics")

  Dim pActiveView As IActiveView
198:   Set pActiveView = m_MxDoc.ActiveView

  Dim pGContainer As IGraphicsContainer
201:   Set pGContainer = m_MxDoc.FocusMap
  Dim pElement As IElement
  Dim strName As String
  Dim pElementProperties As IElementProperties
  Dim pEnvelope As IEnvelope

  ' DELETE EXISTING THRESHOLD ELEMENTS
208:   pGContainer.Reset
209:   Set pElement = pGContainer.Next
210:   While Not pElement Is Nothing
211:     Set pElementProperties = pElement
212:     strName = Left(pElementProperties.Name, 17)
213:     If (strName = "Above Threshold (") Or (strName = "Below Threshold (") _
          Or (strName = "delete_corridors") Or (strName = "delete_corridors_") _
          Or (strName = "Route_Graphics") Then
216:       pGContainer.DeleteElement pElement
217:       If (pEnvelope Is Nothing) Then
218:         Set pEnvelope = pElement.Geometry.Envelope
```

```
219:       Else
220:         pEnvelope.Union pElement.Geometry.Envelope
221:       End If
222:     End If
223:     Set pElement = pGContainer.Next
224:   Wend
225:   If (Not pEnvelope Is Nothing) Then
226:     If pEnvelope.IsEmpty Then
227:       pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, Nothing
228:     Else
229:       pEnvelope.Expand 1.1, 1.1, True
230:       pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, pEnvelope
231:     End If
232:   Else
233:     pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, Nothing
234:   End If

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 12: cmdHabSuitStats

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdHabSuitStats"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdHabSuitStats.cls"
```

```
Implements ICommand
Private Sub Class_Initialize()
  On Error GoTo ErrorHandler


14:   Set m_bitmap = LoadResPicture(115, vbResBitmap)



  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler


27:   Set m_bitmap = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
37:   If (Not m_ExtensionConfig Is Nothing) Then
38:     If (m_ExtensionConfig.State = esriESEnabled) Then
39:       ICommand_Enabled = True
40:     End If
41:   Else
42:     ICommand_Enabled = False
43:   End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
54:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
65:   ICommand_Name = "CorridorDesigner_HabitatSuitabilityStats"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
76:   ICommand_Caption = "Habitat Suitability Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
87:   ICommand_Tooltip = "Habitat Suitability Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
98:   ICommand_Message = "Calculate Habitat Suitability Statistics"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
109:   ICommand_HelpFile = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
120:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
131:   ICommand_Bitmap = m_bitmap.Handle

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
```

```
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
142:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

154:   Set m_App = hook
155:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
158:   newUid.Value = "Linkages.Extension"
159:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)

' 153:   SetWindowLong m_frm1.hwnd, GWL_HWNDPARENT, m_App.hwnd
'  m_frm1.Frame

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

'  MsgBox "First Line of Code; Click Event about to start (Step 1)..."  ' ************************************************************

  On Error GoTo ErrorHandler

  Dim pRaster As IRaster
  Dim pRasterLayer As IRasterLayer
  Dim pRasterBand As IRasterBand
  Dim pRasterBandCollection As IRasterBandCollection
  Dim booHasTable As Boolean

  Dim pSelItem As IUnknown
  Dim pSelRaster As IRasterLayer
```

```
184:   Set pSelItem = m_MxDoc.SelectedItem
185:   If Not pSelItem Is Nothing Then
186:     If TypeOf pSelItem Is IRasterLayer Then
187:       Set pSelRaster = pSelItem
188:     Else
189:       Set pSelItem = Nothing
190:     End If
191:   End If
  Dim pLayer As ILayer

  ' MAKE COLLECTION OF VALID FEATURE/RASTER LAYERS AND STANDALONE TABLES
  Dim pLayerArray As esriSystem.IVariantArray
196:   Set pLayerArray = New esriSystem.VarArray

  Dim pEnumLayer As IEnumLayer

'  MsgBox "Finished dimensioning Variables (Step 2)..."  ' ************************************************************

202:   If (m_MxDoc.FocusMap.LayerCount = 0) Then
203:     MsgBox "No continuous grid layers available in map!  This function requires a continuous grid with " & _
      "values ranging between 0 and 100.  Bailing out...", , "Missing Data:"
    Exit Sub
206:   End If

208:   Set pEnumLayer = m_MxDoc.FocusMap.Layers(, True)
209:   pEnumLayer.Reset
210:   Set pLayer = pEnumLayer.Next

'  MsgBox "Step 3"  ' ************************************************************

  Dim anIndex As Long
  Dim intIndex As Long
216:   intIndex = -1
  Dim booFoundRaster As Boolean
218:   booFoundRaster = False

  Dim pRasterProps As IRasterProps

  ' MAKE COLLECTION OF VALID FEATURE AND RASTER LAYERS

224:   Do While Not pLayer Is Nothing
225:     If pLayer.Valid = True Then
226:       If TypeOf pLayer Is IRasterLayer Then
227:         Set pRasterLayer = pLayer
228:         Set pRaster = pRasterLayer.Raster
229:         Set pRasterProps = pRaster
230:         Set pRasterBandCollection = pRaster
```

```
231:         Set pRasterBand = pRasterBandCollection.Item(0)
'        pRasterBand.HasTable booHasTable
      '  MsgBox "Raster is " & pRasterLayer.Name & vbCrLf & "Has Table? " & booHasTable
'        If Not booHasTable Then
235:         If Not pRasterProps.IsInteger Then
236:           booFoundRaster = True
237:           pLayerArray.Add pRasterLayer
238:         End If
239:       End If
240:     End If
241:     Set pLayer = pEnumLayer.Next
242:   Loop

244:   If (Not booFoundRaster) Then
245:     MsgBox "No continuous grid layers available in map!  This function requires a continuous grid with " & _
      "values ranging between 0 and 100.  Bailing out...", , "Missing Data:"
    Exit Sub
248:   End If

'  MsgBox "Step 5"  ' ************************************************************

'  MsgBox "Step 6"  ' ************************************************************

'  MsgBox "Step 7"  ' ************************************************************

  Dim pSumForm As frmHabSuitStats
257:   Set pSumForm = New frmHabSuitStats

'  MsgBox "Step 8"  ' ************************************************************

261:   Set pSumForm.theRasterLayers = pLayerArray

'  MsgBox "Step 9"  ' ************************************************************

'  MsgBox "Step 10"  ' ***********************************************************

267:   Set pSumForm.theCurrentSelected = pSelItem
 ' MsgBox "Step 11"  ' ***********************************************************

270:   Set pSumForm.ArcApp = m_App
'  MsgBox "Step 12"  ' ***********************************************************
'  MsgBox "Before Show"

274:   pSumForm.Show vbModal

'  MsgBox "After Show"
277:   Set pSumForm = Nothing
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 13: cmdHistogramStats

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdHistogramStats"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Implements ICommand
' Variables used by the Error handler function - DO NOT REMOVE
Const c_sModuleFileName As String = "D:\arcGIS_stuff\saguaro\cmdHistogramStats.cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

16:   Set m_bitmap = LoadResPicture(102, vbResBitmap)

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
```

```
  On Error GoTo ErrorHandler

26:   Set m_bitmap = Nothing
27:   Set m_MxDoc = Nothing
28:   Set m_App = Nothing
29:   Set m_ExtensionConfig = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  Dim booHasExtension As Boolean
40:   If (Not m_ExtensionConfig Is Nothing) Then
41:     If (m_ExtensionConfig.State = esriESEnabled) Then
42:       booHasExtension = True
43:     End If
44:   Else
45:     booHasExtension = False
46:   End If

  Dim pMxDoc As IMxDocument
49:   Set pMxDoc = m_MxDoc

  Dim theMaps As IMaps
52:   Set theMaps = pMxDoc.Maps

  Dim boolShouldEnable As Boolean
55:   boolShouldEnable = False

  Dim theMap As IMap
  Dim theSTCol As IStandaloneTableCollection
  Dim pstandalonetable As IStandaloneTable

  Dim pEnumLayer As IEnumLayer
  Dim pLayer As Variant
  Dim anIndex As Integer

  Dim pTableSelection As ITableSelection
  Dim pSelectionSet As ISelectionSet
  Dim pFeatureLayer As IFeatureLayer
  Dim anIndex2 As Long
```

```
70:   For anIndex = 0 To theMaps.Count - 1
71:     Set theMap = theMaps.Item(anIndex)
72:     Set theSTCol = theMap

74:     If (theSTCol.StandaloneTableCount > 0) Then
75:       For anIndex2 = 0 To theSTCol.StandaloneTableCount - 1
76:         Set pstandalonetable = theSTCol.StandaloneTable(anIndex2)
77:         Set pTableSelection = pstandalonetable
78:         Set pSelectionSet = pTableSelection.SelectionSet
79:         If Not pSelectionSet Is Nothing Then
80:           If pSelectionSet.Count > 0 Then
81:             boolShouldEnable = True
82:             Exit For
83:           End If
84:         End If
85:       Next anIndex2
86:       If boolShouldEnable Then
87:         Exit For
88:       End If
89:     End If

91:     If (theMap.LayerCount > 0) Then
92:       Set pEnumLayer = theMap.Layers
93:       pEnumLayer.Reset

95:       Set pLayer = pEnumLayer.Next
96:       If TypeOf pLayer Is IFeatureLayer Then
97:         Set pFeatureLayer = pLayer
98:         Set pTableSelection = pFeatureLayer
99:         Set pSelectionSet = pTableSelection.SelectionSet
100:         If Not pSelectionSet Is Nothing Then
101:           If pSelectionSet.Count > 0 Then
102:             boolShouldEnable = True
103:             Exit For
104:           End If
105:         End If
106:       End If

108:       Do Until pLayer Is Nothing Or boolShouldEnable
  '      Debug.Print "  --> " & pLayer.Name
110:         If TypeOf pLayer Is IFeatureLayer Then
111:           Set pFeatureLayer = pLayer
112:           Set pTableSelection = pFeatureLayer
113:           Set pSelectionSet = pTableSelection.SelectionSet
114:           If Not pSelectionSet Is Nothing Then
115:             If pSelectionSet.Count > 0 Then
116:               boolShouldEnable = True
```

```
117:               Exit For
118:             End If
119:           End If
120:         End If
121:         Set pLayer = pEnumLayer.Next
122:       Loop

124:       If boolShouldEnable Then
125:         Exit For
126:       End If
127:     End If
128:   Next anIndex

130:   ICommand_Enabled = boolShouldEnable And booHasExtension


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
143:    ICommand_Checked = False


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
156:   ICommand_Name = "CorridorDesigner_StatisticsHistogramCommand"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
169:   ICommand_Caption = "ArcGIS Statistics"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
182:   ICommand_Tooltip = "Statistics on Selected Features"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
195:   ICommand_Message = "Open the standard ArcGIS statistics window, including simple statistics on selected features..."


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
208:   ICommand_HelpFile = ""


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
221:   ICommand_HelpContextID = 0


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
234:   ICommand_Bitmap = m_bitmap.Handle


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
247:   ICommand_Category = "Corridor Designer Tools"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler


259:   Set m_App = hook
260:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
263:   newUid.Value = "Linkages.Extension"
264:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler


  Dim u As New uID
277:   u.Value = "{3E58D6D0-DF7A-11D1-ADD9-080009EC732A}"

  Dim pCmdItem As ICommandItem
280:   Set pCmdItem = m_App.Document.CommandBars.Find(u)
281:   pCmdItem.Execute


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 14: cmdNewShapefile

```
VERSION 1.0 CLASS
```

```
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdNewShapefile"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Implements ICommand
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdNewShapefile.cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

15:   Set m_bitmap = LoadResPicture(116, vbResBitmap)

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

25:   Set m_bitmap = Nothing
26:   Set m_MxDoc = Nothing
27:   Set m_App = Nothing
28:   Set m_ExtensionConfig = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  Dim booHasExtension As Boolean
39:   If (Not m_ExtensionConfig Is Nothing) Then
40:     If (m_ExtensionConfig.State = esriESEnabled) Then
41:       booHasExtension = True
42:     End If
43:   Else
44:     booHasExtension = False
45:   End If


48:   ICommand_Enabled = booHasExtension


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
61:    ICommand_Checked = False


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
74:   ICommand_Name = "CorridorDesigner_MakeShapefile"


  Exit Property
ErrorHandler:
```

```
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
87:   ICommand_Caption = "Create New Shapefile"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
100:   ICommand_Tooltip = "Create New Shapefile"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
113:   ICommand_Message = "Create new shapefile, either empty or by converting graphics..."


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
```

```
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
126:   ICommand_HelpFile = ""


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
139:   ICommand_HelpContextID = 0


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
152:   ICommand_Bitmap = m_bitmap.Handle


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
165:   ICommand_Category = "Corridor Designer Tools"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler


177:   Set m_App = hook
178:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
181:   newUid.Value = "Linkages.Extension"
182:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  Dim pForm As Linkages.frmGraphicsShapefile
194:   Set pForm = New Linkages.frmGraphicsShapefile
195:   Set pForm.ArcApplication = m_App

197:   pForm.Show vbModal

199:   Set pForm = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 15: cmdOpenTable

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdOpenTable"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_currentUser As String
Private m_element As IElement
Private m_bitmap As IPictureDisp

Implements ICommand
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdOpenTable.cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

16:   Set m_bitmap = LoadResPicture(112, vbResBitmap)


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

27:   Set m_bitmap = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Sub

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  Dim pDoc As IMxDocument
  Dim bolEnabled As Boolean
40:   Set pDoc = m_MxDoc
41:   bolEnabled = True       ' DEFAULT TO TRUE

  Dim pCV As IContentsView
44:   Set pCV = pDoc.CurrentContentsView

  Dim pSelItem As IUnknown
47:   Set pSelItem = pDoc.SelectedItem

  ' Disable if the selected item is nothing or if
  ' it is not a layer or table
51:   If pSelItem Is Nothing Then
52:     bolEnabled = False
53:   ElseIf Not ((pCV.Name = "Display") Or (pCV.Name = "Source")) Then
54:     bolEnabled = False
55:   ElseIf Not (TypeOf pSelItem Is IFeatureLayer Or TypeOf pSelItem Is IStandaloneTable Or TypeOf pSelItem Is ISet Or _
      TypeOf pSelItem Is IRasterLayer) Then
57:     bolEnabled = False
58:   Else

60:     If TypeOf pSelItem Is IRasterLayer Then
      Dim pRasterLayer As IRasterLayer
62:       Set pRasterLayer = pSelItem

64:       If Not pRasterLayer.Valid Then
65:         bolEnabled = False
66:       Else
        Dim pRaster As IRaster
        Dim pRasterBand As IRasterBand
        Dim pRasterBandCollection As IRasterBandCollection

71:         Set pRaster = pRasterLayer.Raster
72:         Set pRasterBandCollection = pRaster
73:         Set pRasterBand = pRasterBandCollection.Item(0)
74:         pRasterBand.HasTable bolEnabled
75:       End If
76:     End If
77:   End If

79:   ICommand_Enabled = bolEnabled
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
91:   ICommand_Checked = False


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
104:   ICommand_Name = "CorridorDesigner_OpenTableTool"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
117:   ICommand_Caption = "Open Table"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
130:   ICommand_Tooltip = "Open Attribute Table"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
143:   ICommand_Message = "Open attribute tables for all currently selected themes..."


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
156:   ICommand_HelpFile = ""


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
169:   ICommand_HelpContextID = 0


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
182:   ICommand_Bitmap = m_bitmap.Handle


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
195:   ICommand_Category = "Corridor Designer Tools"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
208:   Set m_App = hook
209:   Set m_MxDoc = m_App.Document
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' ADAPTED FROM OpenFeatureLayerTable.bas

  Dim pMxDoc As IMxDocument
224:   Set pMxDoc = m_MxDoc
  Dim pMap As IMap
226:   Set pMap = pMxDoc.FocusMap

  Dim pCV As IContentsView
229:   Set pCV = pMxDoc.CurrentContentsView

 ' Debug.Print pCV.Name & ": Visible = " & pCV.Visible & vbCrLf & pCatView.Name & ": Visible = " & pCatView.Visible

  Dim theSelectedLayers As Variant

235:   Set theSelectedLayers = pMxDoc.SelectedItem

'  If ((TypeOf pCV Is TOCCatalogView) Or (TypeOf pCV Is TOCDisplayView)) And ((TypeOf pCV.SelectedItem Is ISet) Or
238:   If ((pCV.Name = "Source") Or (pCV.Name = "Display")) And ((TypeOf pCV.SelectedItem Is ISet) Or _
      (TypeOf pCV.SelectedItem Is IFeatureLayer) Or (TypeOf pCV.SelectedItem Is IStandaloneTable) Or _
      (TypeOf pCV.SelectedItem Is IRasterLayer)) Then
241:     Set theSelectedLayers = pCV.SelectedItem
242:   Else
    Exit Sub
244:   End If

  Dim pLayer As IFeatureLayer
  Dim pFeatureLayer As IFeatureLayer           ' JUST TO CHECK VALIDITY
  Dim pTable As ITableWindow
  Dim pTable2 As ITableWindow
  Dim pTableWindow As ITableWindow
  Dim pRasterTable As ITable

  Dim pRasterLayer As IRasterLayer
  Dim pRaster As IRaster
  Dim pRasterBand As IRasterBand
  Dim pRasterBandCollection As IRasterBandCollection
```

```
  Dim pRasterDataset As IRasterDataset
  Dim booHasTable As Boolean

  Dim pTableWindow2 As ITableWindow2           ' NEED #2 TO FIND OPEN STANDALONE TABLES
  Dim pstandalonetable As IStandaloneTable
  Dim pExistingTableWindow As ITableWindow     ' FindViaStandaloneTable RETURNS A ITableWindow

264:   If TypeOf theSelectedLayers Is IFeatureLayer Then
265:     Set pFeatureLayer = theSelectedLayers
266:     If pFeatureLayer.Valid = False Then
267:       MsgBox "There is something wrong with " & pFeatureLayer.Name & "!  It is not a valid " & _
         "layer.  Unable to open table...", vbInformation, "Problem with Data:"
      Exit Sub
270:     End If

    'Debug.Print theSelectedLayers.Type
273:     Set pLayer = theSelectedLayers

275:     Set pTable = New TableWindow

    ' CHECK IF TABLE IS ALREADY OPEN
278:     Set pTable = pTable.FindViaFeatureLayer(pLayer, False)

280:     If pTable Is Nothing Then
      'Associate the table and a feature layer
282:       Set pTable2 = New TableWindow
283:       With pTable2
284:         Set .FeatureLayer = pLayer
285:         Set .Application = m_App
286:         .TableSelectionAction = esriSelectFeatures
287:         .ShowAliasNamesInColumnHeadings = True
288:         .Show True
289:       End With
290:     Else
291:       pTable.Show True
292:     End If

    'Debug.Print pLayer.Name

296:   ElseIf TypeOf theSelectedLayers Is IStandaloneTable Then
297:     Set pstandalonetable = theSelectedLayers
298:     If pstandalonetable.Valid = False Then
299:       MsgBox "There is something wrong with " & pstandalonetable.Name & "!  It is not a valid " & _
         "table.  Unable to open table...", vbInformation, "Problem with Data:"
      Exit Sub
302:     End If
```

```
304:     Set pTableWindow2 = New TableWindow
305:     Set pExistingTableWindow = pTableWindow2.FindViaStandaloneTable(pstandalonetable)

    ' Check if a table already exists; if not, create one
308:     If pExistingTableWindow Is Nothing Then

310:       With pTableWindow2
311:         Set .StandaloneTable = pstandalonetable
312:         Set .Application = m_App
313:         .TableSelectionAction = esriSelectFeatures
314:         .ShowAliasNamesInColumnHeadings = True
315:         .ShowSelected = False
316:         .Show True

318:       End With

320:     Else
321:       pExistingTableWindow.Show True
322:     End If

324:   ElseIf TypeOf theSelectedLayers Is IRasterLayer Then        ' IF RASTER LAYER

326:     Set pRasterLayer = theSelectedLayers

'    MsgBox "Is Raster layer..."

330:     If Not pRasterLayer.Valid Then
331:       MsgBox "There is something wrong with " & pRasterLayer.Name & "!  It is not a valid " & _
         "dataset.  Unable to open table...", vbInformation, "Problem with Data:"
      Exit Sub
334:     End If
335:     Set pRaster = pRasterLayer.Raster
336:     Set pRasterBandCollection = pRaster
337:     Set pRasterBand = pRasterBandCollection.Item(0)
338:     Set pRasterDataset = pRasterBand.RasterDataset
339:     pRasterBand.HasTable booHasTable

341:     If booHasTable Then                    ' USE VAT TABLE FIELDS

'      MsgBox "Has Table..."
344:       Set pRasterTable = pRasterBand.AttributeTable
345:       Set pTableWindow = New TableWindow

      ' CHECK IF TABLE IS ALREADY OPEN
348:       Set pTableWindow = pTableWindow.FindViaTable(pRasterTable, False)

350:       If pTableWindow Is Nothing Then
```

```
        'Associate the table and the raster layer
352:          Set pTableWindow = New TableWindow
353:          With pTableWindow
354:            Set .Table = pRasterTable
355:            Set .Application = m_App
356:            .TableSelectionAction = esriSelectFeatures
357:            .ShowAliasNamesInColumnHeadings = True
358:            .Show True
359:          End With
360:        Else
361:          pTableWindow.Show True
362:        End If
363:      End If


366:    ElseIf TypeOf theSelectedLayers Is ISet Then

    Dim pSelSet As ISet
369:      Set pSelSet = theSelectedLayers

371:      If (pSelSet.Count = 0) Then
       Exit Sub
373:      End If

    ' ROTATE THROUGH SELECTED TABLES AND FEATURE LAYERS, IN RANDOM ORDER PROVIDED BY ISet INTERFACE
    ' WITH TESTING, IT APPEARS THAT SELECTING MULTIPLE TABLES CAUSES A "NOTHING" OBJECT RATHER THAN A SET, DESPITE
    ' WHAT THE DOCUMENTATION CLAIMS WILL HAPPEN.  THEREFORE THERE WILL NEVER BE MULTIPLE TABLES SELECTED.

379:      pSelSet.Reset
    Dim pUnknownLayer As Variant
381:      Set pUnknownLayer = pSelSet.Next

383:      Do Until pUnknownLayer Is Nothing

385:        If TypeOf pUnknownLayer Is IFeatureLayer Then
386:          Set pFeatureLayer = pUnknownLayer
387:          If pFeatureLayer.Valid = False Then
388:            MsgBox "There is something wrong with " & pFeatureLayer.Name & "!  It is not a valid " & _
              "layer.  Unable to open table...", vbInformation, "Problem with Data:"
           Exit Sub
391:          End If

        'Debug.Print theSelectedLayers.Type

395:          Set pLayer = pUnknownLayer

    '    Dim pTable As ITableWindow
```

```
398:         Set pTable = New TableWindow

        ' CHECK IF TABLE IS ALREADY OPEN
401:         Set pTable = pTable.FindViaFeatureLayer(pLayer, False)

403:         If pTable Is Nothing Then
          'Associate the table and a feature layer
405:           Set pTable2 = New TableWindow
406:           With pTable2
407:             Set .FeatureLayer = pLayer
408:             Set .Application = m_App
409:             .TableSelectionAction = esriSelectFeatures
410:             .ShowAliasNamesInColumnHeadings = True
411:             .Show True
412:           End With
413:         Else
414:           pTable.Show True
415:         End If

        'Debug.Print pLayer.Name

419:       ElseIf TypeOf pUnknownLayer Is IStandaloneTable Then
420:         Set pstandalonetable = pUnknownLayer
421:         If pstandalonetable.Valid = False Then
422:           MsgBox "There is something wrong with " & pstandalonetable.Name & "!  It is not a valid " & _
             "table.  Unable to open table...", vbInformation, "Problem with Data:"
          Exit Sub
425:         End If

        ' A standalone table
428:         Set pTableWindow2 = New TableWindow
429:         Set pExistingTableWindow = pTableWindow2.FindViaStandaloneTable(pstandalonetable)

        ' Check if a table already exists; if not, create one
432:         If pExistingTableWindow Is Nothing Then

434:           With pTableWindow2
435:             Set .StandaloneTable = pstandalonetable
436:             Set .Application = m_App
437:             .TableSelectionAction = esriSelectFeatures
438:             .ShowAliasNamesInColumnHeadings = True
439:             .ShowSelected = False
440:             .Show True

442:           End With

444:         Else
```

```
445:           pExistingTableWindow.Show True
446:         End If

448:       ElseIf TypeOf pUnknownLayer Is IRasterLayer Then        ' IF RASTER LAYER

450:         Set pRasterLayer = pUnknownLayer

452:         If Not pRasterLayer.Valid Then
453:           MsgBox "There is something wrong with " & pRasterLayer.Name & "!  It is not a valid " & _
              "dataset.  Unable to open table...", vbInformation, "Problem with Data:"
          Exit Sub
456:         End If
457:         Set pRaster = pRasterLayer.Raster
458:         Set pRasterBandCollection = pRaster
459:         Set pRasterBand = pRasterBandCollection.Item(0)
460:         Set pRasterDataset = pRasterBand.RasterDataset
461:         pRasterBand.HasTable booHasTable

463:         If booHasTable Then                    ' USE VAT TABLE FIELDS

465:           Set pRasterTable = pRasterBand.AttributeTable
466:           Set pTableWindow = New TableWindow

          ' CHECK IF TABLE IS ALREADY OPEN
469:           Set pTableWindow = pTableWindow.FindViaTable(pRasterTable, False)

471:           If pTableWindow Is Nothing Then
            'Associate the table and the raster layer
473:             Set pTableWindow = New TableWindow
474:             With pTableWindow
475:               Set .Table = pRasterTable
476:               Set .Application = m_App
477:               .TableSelectionAction = esriSelectFeatures
478:               .ShowAliasNamesInColumnHeadings = True
479:               .Show True
480:             End With
481:           Else
482:             pTableWindow.Show True
483:           End If
484:         End If

486:       End If

488:       Set pUnknownLayer = pSelSet.Next

490:     Loop
```

```
492:   End If

  'Debug.Print ""  ' new line

  Exit Sub

ErrorHandler:
499:   MsgBox "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl) & vbCrLf & _
    Err.Number & vbCrLf & Err.Source & vbCrLf & Err.Description
End Sub
```

Class 16: cmdSumMod

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdSumMod"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Implements ICommand
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdSumMod.cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler


16:   Set m_bitmap = LoadResPicture(107, vbResBitmap)
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

27:   Set m_App = Nothing
28:   Set m_bitmap = Nothing
29:   Set m_MxDoc = Nothing
30:   Set m_ExtensionConfig = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  Dim booShouldEnable As Boolean

42:   If (Not m_ExtensionConfig Is Nothing) Then
43:     If (m_ExtensionConfig.State = esriESEnabled) Then
44:       booShouldEnable = True
45:     End If
46:   Else
47:     booShouldEnable = False
48:   End If

50:   If booShouldEnable Then

  ' USE THE CODE BELOW TO ALWAYS ENABLE BUTTON
    Dim pEnumLayer As IEnumLayer
    Dim pFeatureLayer As IFeatureLayer
    Dim pRasterLayer As IRasterLayer

57:     booShouldEnable = False
    Dim pUID As New uID
59:     pUID.Value = "{E156D7E5-22AF-11D3-9F99-00C04F6BC78E}"   ' ALL DATA LAYERS

    On Error Resume Next
62:     Set pEnumLayer = m_MxDoc.FocusMap.Layers(Nothing, True)
    On Error GoTo ErrorHandler
```

```
65:     If Not pEnumLayer Is Nothing Then
      Dim pLayer As ILayer

     ' CHECK FOR FEATURE AND RASTER LAYERS
69:       Set pLayer = pEnumLayer.Next
70:       Do While Not pLayer Is Nothing
71:         pEnumLayer.Reset
72:         If pLayer.Valid = True Then
  '        MsgBox "Examining " & pLayer.Name
74:           If TypeOf pLayer Is IFeatureLayer Or TypeOf pLayer Is IRasterLayer Then
75:             booShouldEnable = True
  '          MsgBox pLayer.Name & " --> Should enable = " & booShouldEnable
77:             Exit Do
78:           End If
79:         End If
80:       Loop
81:     End If

    ' CHECK FOR STANDALONE TABLES
84:     If Not booShouldEnable Then
      Dim pStTabCol As IStandaloneTableCollection
86:       Set pStTabCol = m_MxDoc.FocusMap

88:       If pStTabCol.StandaloneTableCount > 0 Then
        Dim pstandalonetable As IStandaloneTable

        Dim anIndex As Long
92:         For anIndex = 0 To (pStTabCol.StandaloneTableCount - 1)
93:           Set pstandalonetable = pStTabCol.StandaloneTable(anIndex)
94:           If pstandalonetable.Valid Then
95:             booShouldEnable = True
96:             Exit For
97:           End If
98:         Next anIndex
99:       End If
100:    End If
101:  End If
102:  ICommand_Enabled = booShouldEnable

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
```

```
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
114:    ICommand_Checked = False


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
127:   ICommand_Name = "CorridorDesigner_CorridorSummarize"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
140:   ICommand_Caption = "Corridor Summarize"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
153:   ICommand_Tooltip = "Summary Statistics"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
166:   ICommand_Message = "Generate general summary statistics on tables, feature layers and grids..."


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
179:   ICommand_HelpFile = ""


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
192:   ICommand_HelpContextID = 0


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
205:   ICommand_Bitmap = m_bitmap.Handle


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler


  ' TODO: Add your implementation here
218:   ICommand_Category = "Corridor Designer Tools"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler


230:   Set m_App = hook
231:   Set m_MxDoc = m_App.Document
  Dim newUid As New uID
233:   newUid.Value = "Linkages.Extension"
234:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub ICommand_OnClick()

'  MsgBox "First Line of Code; Click Event about to start (Step 1)..."  ' ************************************************************

  On Error GoTo ErrorHandler

  Dim pSelItem As IUnknown
249:   Set pSelItem = m_MxDoc.SelectedItem
  Dim pFeatureLayer As IFeatureLayer
  Dim pLayer As ILayer
  Dim pstandalonetable As IStandaloneTable
'  Dim pTableFields As ITableFields
  Dim pTableFields As IUnknown
  Dim pRasterLayer As IRasterLayer

  ' MAKE COLLECTION OF VALID FEATURE/RASTER LAYERS AND STANDALONE TABLES
  Dim colLayerCol As Collection
  Dim pEnumLayer As IEnumLayer
  Dim pStTabColl As IStandaloneTableCollection
261:   Set pStTabColl = m_MxDoc.FocusMap

'  MsgBox "Finished dimensioning Variables (Step 2)..."  ' ************************************************************

  If (m_MxDoc.FocusMap.LayerCount = 0) And (pStTabColl.StandaloneTableCount = 0) Then Exit Sub

'  Dim pUID As New uID
'  pUID.Value = "{E156D7E5-22AF-11D3-9F99-00C04F6BC78E}"
'
'  On Error Resume Next
'  Set pEnumLayer = pMxDoc.FocusMap.Layers(pUID, True)

273:   Set pEnumLayer = m_MxDoc.FocusMap.Layers(, True)
274:   pEnumLayer.Reset
275:   Set pLayer = pEnumLayer.Next

'  MsgBox "Step 3"  ' ************************************************************

  Dim anIndex As Long
  Dim intIndex As Long
281:   intIndex = -1

  ' MAKE COLLECTION OF VALID FEATURE AND RASTER LAYERS
284:   Set colLayerCol = New Collection
285:   Do While Not pLayer Is Nothing
286:     If pLayer.Valid = True Then
287:       If TypeOf pLayer Is IFeatureLayer Or TypeOf pLayer Is IRasterLayer Then
```

```
288:         intIndex = intIndex + 1
289:         colLayerCol.Add pLayer, CStr(intIndex)
290:       End If
291:     End If
292:     Set pLayer = pEnumLayer.Next
293:   Loop

'  MsgBox "Step 5"  ' ************************************************************

297:   For anIndex = 0 To pStTabColl.StandaloneTableCount - 1
298:     Set pstandalonetable = pStTabColl.StandaloneTable(anIndex)
299:     If pstandalonetable.Valid = True Then
300:       intIndex = intIndex + 1
301:       colLayerCol.Add pstandalonetable, CStr(intIndex)
302:     End If
303:   Next anIndex

'  MsgBox "Step 6"  ' ************************************************************

  ' PRESELECT ACTIVE LAYER OR TABLE IF ANY SINGLE LAYER/TABLE IS ACTIVE
308:   If pSelItem Is Nothing Then
309:     Set pTableFields = Nothing
310:   ElseIf TypeOf pSelItem Is IFeatureLayer Then
311:     Set pFeatureLayer = pSelItem
312:     If pFeatureLayer.Valid = True Then
313:       Set pTableFields = pFeatureLayer
314:     End If

316:   ElseIf TypeOf pSelItem Is IStandaloneTable Then
317:     Set pstandalonetable = pSelItem
318:     If pstandalonetable.Valid = True Then
319:       Set pTableFields = pstandalonetable
320:     End If
321:   ElseIf TypeOf pSelItem Is IRasterLayer Then
322:     Set pRasterLayer = pSelItem
323:     If pRasterLayer.Valid Then
324:       Set pTableFields = pRasterLayer
325:     End If

327:   Else
328:     Set pTableFields = Nothing
329:     Set pSelItem = Nothing
330:   End If

'  MsgBox "Step 7"  ' ************************************************************

'  Dim strReport As String
```

```
'  For anIndex = 1 To colLayerCol.Count
'    strReport = "Layer #" & CStr(anIndex) & " is Nothing = " & CStr(colLayerCol.Item(anIndex) Is Nothing)
'  Next anIndex
'  MsgBox strReport

  Dim pSumForm As frm_Summarize
341:   Set pSumForm = New frm_Summarize

'  MsgBox "Step 8"  ' ************************************************************

345:   Set pSumForm.theTableLayers = colLayerCol

'  MsgBox "Step 9"  ' ************************************************************

349:   Set pSumForm.Field_Array = pTableFields
'  MsgBox "Step 10"  ' ************************************************************

352:   Set pSumForm.theCurrentSelected = pSelItem
 ' MsgBox "Step 11"  ' ************************************************************

355:   Set pSumForm.ArcApp = m_App
'  MsgBox "Step 12"  ' ************************************************************
'  MsgBox "Before Show"

359:   pSumForm.Show vbModal

'  MsgBox "After Show"
362:   Set pSumForm = Nothing

'  MsgBox "After Nothing"
'  Dim pSumUI As ISummarizeUI
'  Set pSumUI = New SummarizeUI
'  Set pSumUI.Application = m_App
'  Set pSumUI.SummarizeTable = pSelItem
'  Set pSumUI.SummarizeField = pField
'  pSumUI.SummarizeOnSelectedOnly = True
'  pSumUI.DoModal m_App.hWnd


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 17: cmdTestCode

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cmdTestCode"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit


Private m_App As IApplication
Private m_MxDoc As IMxDocument
Private m_bitmap As IPictureDisp
Private m_ExtensionConfig As IExtensionConfig

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\cmdTestCode.cls"
Implements ICommand

Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

15:   Set m_bitmap = LoadResPicture(103, vbResBitmap)

  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

25:   Set m_bitmap = Nothing
26:   Set m_MxDoc = Nothing
27:   Set m_App = Nothing
28:   Set m_ExtensionConfig = Nothing
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
38:   ICommand_Enabled = True


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
50:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
61:   ICommand_Name = "CorridorDesigner_TestTools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
72:   ICommand_Caption = "Test Tools"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
83:   ICommand_Tooltip = "Test Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
94:   ICommand_Message = "Test Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
105:   ICommand_HelpFile = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
```

```
116:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
127:   ICommand_Bitmap = m_bitmap.Handle

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
138:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

150:   Set m_App = hook
151:   Set m_MxDoc = m_App.Document

  Dim newUid As New uID
154:   newUid.Value = "Linkages.Extension"
155:   Set m_ExtensionConfig = m_App.FindExtensionByCLSID(newUid)

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
```

```
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  Dim pExt As Linkages.Extension
166:   Set pExt = m_ExtensionConfig

  ' TODO: Add your implementation here
169:   MyGeneralOperations.DeleteGraphicsByName m_MxDoc, "test_matrix"
170:   MyGeneralOperations.DeleteGraphicsByName m_MxDoc, "DeleteMatrix"
171:   MyGeneralOperations.DeleteGraphicsByName m_MxDoc, "Route_Graphics"
172:   MyGeneralOperations.DeleteGraphicsByName m_MxDoc, "test_order"

  ' GET INTERNAL NODES - FROM ORIGINAL TEST CODE
  Dim pStartArray As IArray
176:   Set pStartArray = New esriSystem.Array
177:   Set pStartArray = MyGeneralOperations.ReturnGraphicsByName(m_MxDoc, "Internal", False)

  ' GET START AND END NODES - FROM ORIGINAL TEST CODE
  Dim pStartPolygon As IPolygon
  Dim pEndPolygon As IPolygon
182:   Set pStartPolygon = MyGeneralOperations.ReturnGraphicsByName(m_MxDoc, "Start", False).Element(0)
183:   Set pEndPolygon = MyGeneralOperations.ReturnGraphicsByName(m_MxDoc, "End", False).Element(0)

185:   pStartArray.Insert 0, pStartPolygon
186:   pStartArray.Add pEndPolygon

  Dim pClone As IClone

  ' GET CORRIDOR - FROM ORIGINAL TEST CODE
  Dim pCorridorArray As IArray
192:   Set pCorridorArray = MyGeneralOperations.ReturnGraphicsByName(m_MxDoc, "Corridor", False)
  Dim pCorPolygon As IPolygon
194:   Set pCorPolygon = pCorridorArray.Element(0)

  ' PROGRESS METER STUFF ------------------------------------------------------------------
  Dim frmProgress As New frmJenProgressPercent
  Dim theTimeBegan As Date
  Dim theDetailedDescription As String

201:   frmProgress.SetExpanded = pExt.ProgressDialogSetExpanded
202:   frmProgress.SetAutoClose = pExt.ProgressDialogAutoClose

204:   theTimeBegan = Now
205:   frmProgress.ShouldContinue = True
```

```
206:   frmProgress.ProgBeginTime = Now
207:   frmProgress.ProgRecCount = 0
208:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
209:   frmProgress.lblBeginTime.Caption = "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd")

211:   theDetailedDescription = "Analyzing Patch Connectors..." & vbCrLf & _
      "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd") & vbCrLf & _
      "-----------------------------------------" & vbCrLf
214:   frmProgress.txtDetails.Text = theDetailedDescription

216:   frmProgress.Frame.Caption = "Current Status:"
217:   frmProgress.Frame.Visible = True

'   theProgressTimeCheck = CDate(50000)
  Dim theDescription As String
221:   theDescription = "Analyzing Patch Connectors..."
  ' PROGRESS METER STUFF ------------------------------------------------------------------

224:   CorridorAnalysisFunctions.ImplementKruskall m_MxDoc, pStartPolygon, pEndPolygon, _
      pStartArray, pCorPolygon, frmProgress, m_ExtensionConfig, Nothing

227:   pExt.ProgressDialogAutoClose = (frmProgress.chkClose.Value = 1)
228:   pExt.ProgressDialogSetExpanded = (frmProgress.SetExpanded)

230:   If frmProgress.chkClose.Value = 1 Then
231:     Unload frmProgress
232:     Set frmProgress = Nothing
233:   End If


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 18: CollectionMod

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
```

```
Attribute VB_Name = "CollectionMod"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Dim m_collection As Collection
Dim m_Keys() As String

'Public Property Set Collection(colSet As Collection)
'  Set m_collection = colSet
'End Property

'Public Property Get Collection() As Collection
'  Set IntCollection = m_collection
'End Property


Public Property Get Count() As Long
16:   Count = m_collection.Count
End Property

Public Property Get ReturnKeys() As String()

  Dim strReturnKeys() As String
22:   strReturnKeys = m_Keys
23:   ReturnKeys = strReturnKeys

End Property

'Public Function HasKeyUsingError(strkey As String) As Boolean
'On Error GoTo ErrorHandler
'
'  HasKeyUsingError = True
'  Dim varTest As Variant
'  varTest = m_collection.Item(strkey)
'
'  Exit Function
'ErrorHandler
'  HasKeyUsingError = False
'End Function

Public Function HasKey(strKey As String) As Boolean

41:   HasKey = False
  Dim lngCounter As Long
```

```
43:   For lngCounter = 0 To UBound(m_Keys)
44:     If m_Keys(lngCounter) = strKey Then
45:       HasKey = True
46:       Exit For
47:     End If
48:   Next lngCounter

End Function

Public Function AddObject(aValue As IUnknown, strKey As String, ShouldReplace As Boolean) As Boolean
  Dim HasVal As Variant
54:   AddObject = False
55:   HasVal = Not GetObject(strKey) Is Nothing
56:   If HasVal Then           ' ALREADY AN ITEM AVAILABLE
57:     If ShouldReplace Then
58:       Me.SetOrReplace aValue, strKey
59:       AddObject = True
60:     End If
61:   Else
62:     m_collection.Add aValue, strKey
63:     AddObject = True
64:     AddKey strKey
65:   End If

End Function
Public Function AddVariable(aValue As Variant, strKey As String, ShouldReplace As Boolean) As Boolean
  Dim HasVal As Variant
70:   AddVariable = False
71:   HasVal = GetVariable(strKey)
72:   If Not IsNull(HasVal) Then          ' ALREADY AN ITEM AVAILABLE
73:     If ShouldReplace Then
74:       Me.SetOrReplace aValue, strKey
75:       AddVariable = True
76:     End If
77:   Else
78:     m_collection.Add aValue, strKey
79:     AddVariable = True
80:     AddKey strKey
81:   End If

End Function

Private Sub AddKey(strKey As String)
  ReDim Preserve m_Keys(m_collection.Count - 1)
87:   m_Keys(m_collection.Count - 1) = strKey
End Sub
```

```
Private Sub RemoveKey(strKey As String)

  Dim strTestKey As String
  Dim strTemp() As String
  Dim FoundKey As Boolean
95:   FoundKey = False
  ReDim strTemp(UBound(m_Keys))

  Dim lngCounter As Long
99:   lngCounter = -1
  Dim lngIndex As Long

102:   For lngIndex = 0 To UBound(m_Keys)
103:     strTestKey = m_Keys(lngIndex)
104:     If Not strTestKey = strKey Then
105:       lngCounter = lngCounter + 1
106:       strTemp(lngCounter) = strTestKey
107:     Else
108:       FoundKey = True
109:     End If
110:   Next lngIndex

112:   If FoundKey Then
    ReDim Preserve strTemp(UBound(m_Keys) - 1)
114:     m_Keys = strTemp
115:   End If

End Sub

Public Sub RemoveWithoutError(strKey As String)

  On Error Resume Next
122:   m_collection.Remove strKey
123:   RemoveKey strKey

End Sub
Public Sub SetOrReplace(aValue As Variant, strKey As String)

  On Error Resume Next
129:   m_collection.Remove strKey
130:   RemoveKey strKey
131:   m_collection.Add aValue, strKey
132:   AddKey strKey

End Sub

Public Function GetVariable(strKey As String) As Variant
```

```
  Dim IsNil As Boolean
138:   IsNil = True
139:   Err.Clear

  Dim varResponseVal As Variant
  On Error GoTo FoundError

'  If VarType(m_collection.Item(strKey) = 13) Then
'    Set varResponseVal = m_collection.Item(strKey)
'  Else
147:     varResponseVal = m_collection.Item(strKey)
'  End If

  ' SHOULD SKIP NEXT LINE IF INDEX DID NOT EXIST (TRIGGERING AN ERROR)
151:   IsNil = False
152:   GetVariable = varResponseVal
  Exit Function

FoundError:
'  Debug.Print Err.Description & ",   " & Err.Number

158:   If IsNil Then varResponseVal = Null
159:   GetVariable = varResponseVal

End Function
Public Function GetObject(strKey As String) As IUnknown
  Dim IsNil As Boolean
164:   IsNil = True
165:   Err.Clear

  Dim pResponseVal As IUnknown
  On Error GoTo FoundError

'  If VarType(m_collection.Item(strKey) = 13) Then
'    Set varResponseVal = m_collection.Item(strKey)
'  Else
173:     Set pResponseVal = m_collection.Item(strKey)
'  End If

  ' SHOULD SKIP NEXT LINE IF INDEX DID NOT EXIST (TRIGGERING AN ERROR)
177:   IsNil = False
178:   Set GetObject = pResponseVal
  Exit Function

FoundError:
'  Debug.Print Err.Description & ",   " & Err.Number
```

```
184:   If IsNil Then Set GetObject = Nothing

End Function
Private Sub Class_Initialize()
'  Dim m_collection As Collection
189:   Set m_collection = New Collection
  ReDim m_Keys(0)

End Sub

Private Sub Class_Terminate()

196:   Set m_collection = Nothing
End Sub
```

Class 19: Extension

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "Extension"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\Extension.cls"
Const c_requiredLicenseProductCode = esriLicenseProductCodeArcView
Private m_App As IApplication
Private m_AoInitialize As IAoInitialize
Private m_CorrectLicense As Boolean
Private m_SpatialAnalyst As Boolean
Private m_LicenseDescription As String
Private m_ExtensionState As esriExtensionState
Private m_ExtensionLicenseStatus As esriLicenseStatus
Private m_SelForm As Object
Private m_FormStep1 As Object
Private m_FormStep2 As Object
Private m_frmBottleneck As Object
Private m_ClipForm As Object
```

```
Private m_EnableSelTool As Boolean
Private m_EnableDrawTool As Boolean
Private m_WildlandPolygon1 As IPolygon
Private m_WildlandPolygon2 As IPolygon
Private m_Corridor As IPolygon
Private m_booHelpToggle1 As Boolean
Private m_booUsePatchesStep1 As Boolean
Private m_pPatchLayerStep1 As IFeatureLayer
Private m_booDoAllPatchesStep1 As Boolean
Private m_pPatchAttFieldStep1 As IField
Private m_intPatchOperatorStep1 As Integer
Private m_strPatchValueStep1 As String
Private m_intPatchFieldIndexStep1 As Integer
Private m_ProgressDialogExpanded As Boolean
Private m_ProgressDialogCloseOnComplete As Boolean
Private m_PatchArray As esriSystem.IArray
Private m_ClipDirectoryPath As String
Private m_CorrPolygonIsDrawn As Boolean
Private m_lngHistogramBinCount As Long

Private m_arcToolboxExtension As IArcToolboxExtension

Const c_toolboxName As String = "CorridorDesigner Arizona.tbx"
Private WithEvents DocumentEvents As MxDocument
Attribute DocumentEvents.VB_VarHelpID = -1

Implements IExtension
Implements IExtensionConfig
Implements IPersistVariant

Public Property Let HistogramBinCount(lngBinCount As Long)
  On Error GoTo ErrorHandler

49:   Let m_lngHistogramBinCount = lngBinCount

  Exit Property
ErrorHandler:
  HandleError True, "HistogramBinCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get HistogramBinCount() As Long
  On Error GoTo ErrorHandler

58:   HistogramBinCount = m_lngHistogramBinCount

  Exit Property
ErrorHandler:
```

```
  HandleError True, "HistogramBinCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let CorrPolygonIsDrawn(booIsDrawn As Boolean)            ' FOR frmClip; FLAG TO INDICATE IF POLYGON WAS HAND-DRAWN
  On Error GoTo ErrorHandler

67:   Let m_CorrPolygonIsDrawn = booIsDrawn

  Exit Property
ErrorHandler:
  HandleError True, "CorrPolygonIsDrawn " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get CorrPolygonIsDrawn() As Boolean                      ' FOR frmClip; FLAG TO INDICATE IF POLYGON WAS HAND-DRAWN
  On Error GoTo ErrorHandler

76:   CorrPolygonIsDrawn = m_CorrPolygonIsDrawn

  Exit Property
ErrorHandler:
  HandleError True, "CorrPolygonIsDrawn " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let ClipDirectoryPath(strPathName As String)            ' FOR frmClip Default Folder Path
  On Error GoTo ErrorHandler

85:   Let m_ClipDirectoryPath = strPathName

  Exit Property
ErrorHandler:
  HandleError True, "ClipDirectoryPath " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get ClipDirectoryPath() As String                        ' FOR frmClip Default Folder Path
  On Error GoTo ErrorHandler

94:   ClipDirectoryPath = m_ClipDirectoryPath

  Exit Property
ErrorHandler:
  HandleError True, "ClipDirectoryPath " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let ProgressDialogAutoClose(SetAutoClose As Boolean)    ' FOR frmJenProgressPercent
  On Error GoTo ErrorHandler

103:   Let m_ProgressDialogCloseOnComplete = SetAutoClose
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ProgressDialogAutoClose " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get ProgressDialogAutoClose() As Boolean           ' FOR frmJenProgressPercent
  On Error GoTo ErrorHandler

112:   ProgressDialogAutoClose = m_ProgressDialogCloseOnComplete

  Exit Property
ErrorHandler:
  HandleError True, "ProgressDialogAutoClose " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Let ProgressDialogSetExpanded(SetExpanded As Boolean)    ' FOR frmJenProgressPercent
  On Error GoTo ErrorHandler

122:   Let m_ProgressDialogExpanded = SetExpanded

  Exit Property
ErrorHandler:
  HandleError True, "ProgressDialogSetExpanded " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get ProgressDialogSetExpanded() As Boolean           ' FOR frmJenProgressPercent
  On Error GoTo ErrorHandler

131:   ProgressDialogSetExpanded = m_ProgressDialogExpanded

  Exit Property
ErrorHandler:
  HandleError True, "ProgressDialogSetExpanded " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Let PatchAttFieldIndex(intIndex As Integer)   ' FOR Jennessent_CompareParameters.cbxPatchField
  On Error GoTo ErrorHandler

140:   Let m_intPatchFieldIndexStep1 = intIndex

  Exit Property
ErrorHandler:
  HandleError True, "PatchAttFieldIndex " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

```
Public Property Get PatchAttFieldIndex() As Integer           ' FOR Jennessent_CompareParameters.cbxPatchField
  On Error GoTo ErrorHandler

149:   PatchAttFieldIndex = m_intPatchFieldIndexStep1

  Exit Property
ErrorHandler:
  HandleError True, "PatchAttFieldIndex " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Let PatchAttValue(strValue As String)         ' FOR Jennessent_CompareParameters.txtValue
  On Error GoTo ErrorHandler

159:   Let m_strPatchValueStep1 = strValue

  Exit Property
ErrorHandler:
  HandleError True, "PatchAttValue " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get PatchAttValue() As String                 ' FOR Jennessent_CompareParameters.txtValue
  On Error GoTo ErrorHandler

168:   PatchAttValue = m_strPatchValueStep1

  Exit Property
ErrorHandler:
  HandleError True, "PatchAttValue " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Let PatchOperatorIndex(intIndex As Integer)   ' FOR Jennessent_CompareParameters.cbxValue
  On Error GoTo ErrorHandler

178:   Let m_intPatchOperatorStep1 = intIndex

  Exit Property
ErrorHandler:
  HandleError True, "PatchOperatorIndex " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get PatchOperatorIndex() As Integer           ' FOR Jennessent_CompareParameters.cbxValue
  On Error GoTo ErrorHandler

187:   PatchOperatorIndex = m_intPatchOperatorStep1
```

```
  Exit Property
ErrorHandler:
  HandleError True, "PatchOperatorIndex " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set PatchField(pField As IField)              ' FOR Jennessent_CompareParameters.cbxPatchField
  On Error GoTo ErrorHandler

197:   Set m_pPatchAttFieldStep1 = pField

  Exit Property
ErrorHandler:
  HandleError True, "PatchField " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
Public Property Get PatchField() As IField                    ' FOR Jennessent_CompareParameters.cbxPatchField
  On Error GoTo ErrorHandler

206:   Set PatchField = m_pPatchAttFieldStep1

  Exit Property
ErrorHandler:
  HandleError True, "PatchField " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Public Property Let PatchUseAll(booPatchUseAll As Boolean)    ' FOR Jennessent_CompareParameters.optUseAll, optSubSet
  On Error GoTo ErrorHandler

216:   Let m_booDoAllPatchesStep1 = booPatchUseAll
  Exit Property
ErrorHandler:
  HandleError True, "PatchUseAll " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4


End Property
Public Property Get PatchUseAll() As Boolean                  ' FOR Jennessent_CompareParameters.optUseAll, optSubSet
  On Error GoTo ErrorHandler

226:   PatchUseAll = m_booDoAllPatchesStep1
  Exit Property
ErrorHandler:
  HandleError True, "PatchUseAll " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Public Property Set PatchLayer(pFeatureLayer As IFeatureLayer) ' FOR Jennessent_CompareParameters.cbxPatchLayer
  On Error GoTo ErrorHandler

234:   Set m_pPatchLayerStep1 = pFeatureLayer

  Exit Property
ErrorHandler:
  HandleError True, "PatchLayer " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
Public Property Get PatchLayer() As IFeatureLayer             ' FOR Jennessent_CompareParameters.cbxPatchLayer
  On Error GoTo ErrorHandler

243:   Set PatchLayer = m_pPatchLayerStep1

  Exit Property
ErrorHandler:
  HandleError True, "PatchLayer " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property


Public Property Set PatchArray(pPatchArray As esriSystem.IArray) ' FOR Jennessent_CompareParameters.cbxPatchLayer
  On Error GoTo ErrorHandler

254:   Set m_PatchArray = pPatchArray

  Exit Property
ErrorHandler:
  HandleError True, "PatchArray " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
Public Property Get PatchArray() As esriSystem.IArray             ' FOR Jennessent_CompareParameters.cbxPatchLayer
  On Error GoTo ErrorHandler

263:   Set PatchArray = m_PatchArray

  Exit Property
ErrorHandler:
  HandleError True, "PatchArray " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property


Public Property Let PatchUse(booPatchUse As Boolean)          ' FOR Jennessent_CompareParameters.chkUsePatches
  On Error GoTo ErrorHandler
```

```
274:   Let m_booUsePatchesStep1 = booPatchUse

  Exit Property
ErrorHandler:
  HandleError True, "PatchUse " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property
Public Property Get PatchUse() As Boolean                     ' FOR Jennessent_CompareParameters.chkUsePatches
  On Error GoTo ErrorHandler

283:   PatchUse = m_booUsePatchesStep1

  Exit Property
ErrorHandler:
  HandleError True, "PatchUse " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Let HelpToggle1(booHelp As Boolean)           ' FOR Jennessent_CompareParameters.cmdHelp
  On Error GoTo ErrorHandler

293:   Let m_booHelpToggle1 = booHelp

  Exit Property
ErrorHandler:
  HandleError True, "HelpToggle1 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
Public Property Get HelpToggle1() As Boolean                  ' FOR Jennessent_CompareParameters.cmdHelp
  On Error GoTo ErrorHandler

302:   HelpToggle1 = m_booHelpToggle1

  Exit Property
ErrorHandler:
  HandleError True, "HelpToggle1 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Public Property Get PolyWildland1() As IPolygon               ' FOR Jennessent_CompareParameters.imgCheckWB1, cbxHab1
  On Error GoTo ErrorHandler

312:   Set PolyWildland1 = m_WildlandPolygon1

  Exit Property
ErrorHandler:
  HandleError True, "PolyWildland1 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
```

```
Public Property Set PolyWildland1(ByRef pPolygon As IPolygon) ' FOR Jennessent_CompareParameters.imgCheckWB1, cbxHab1
  On Error GoTo ErrorHandler

322:   Set m_WildlandPolygon1 = pPolygon

  Exit Property
ErrorHandler:
  HandleError True, "PolyWildland1 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Get PolyWildland2() As IPolygon               ' FOR Jennessent_CompareParameters.imgCheckWB2, cbxHab2
  On Error GoTo ErrorHandler

332:   Set PolyWildland2 = m_WildlandPolygon2

  Exit Property
ErrorHandler:
  HandleError True, "PolyWildland2 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set PolyWildland2(ByRef pPolygon As IPolygon) ' FOR Jennessent_CompareParameters.imgCheckWB2, cbxHab2
  On Error GoTo ErrorHandler

342:   Set m_WildlandPolygon2 = pPolygon

  Exit Property
ErrorHandler:
  HandleError True, "PolyWildland2 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Get PolyCorridor() As IPolygon                ' FOR Jennessent_CompareParameters.imgCheckSpCorr, cbxCorridor
  On Error GoTo ErrorHandler

352:   Set PolyCorridor = m_Corridor

  Exit Property
ErrorHandler:
  HandleError True, "PolyCorridor " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set PolyCorridor(ByRef pPolygon As IPolygon)  ' FOR Jennessent_CompareParameters.imgCheckSpCorr, cbxCorridor
  On Error GoTo ErrorHandler
```

```
362:   Set m_Corridor = pPolygon

  Exit Property
ErrorHandler:
  HandleError True, "PolyCorridor " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property


Public Property Get aSelForm() As Object                        ' FOR frmSelScreen
  On Error GoTo ErrorHandler

373:   Set aSelForm = m_SelForm

  Exit Property
ErrorHandler:
  HandleError True, "aSelForm " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Set aSelForm(ByRef vNewValue As Object)         ' FOR frmSelScreen
  On Error GoTo ErrorHandler

383:   Set m_SelForm = vNewValue

  Exit Property
ErrorHandler:
  HandleError True, "aSelForm " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Get frmStep1() As Object                        ' FOR Jennessent_CompareParameters
  On Error GoTo ErrorHandler

393:   Set frmStep1 = m_FormStep1

  Exit Property
ErrorHandler:
  HandleError True, "frmStep1 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Set frmStep1(ByRef frm1 As Object)              ' FOR Jennessent_CompareParameters
  On Error GoTo ErrorHandler

403:   Set m_FormStep1 = frm1

  Exit Property
ErrorHandler:
```

```
  HandleError True, "frmStep1 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Get BottleneckForm() As Object                      ' FOR frmBottleneck
  On Error GoTo ErrorHandler

413:   Set BottleneckForm = m_frmBottleneck

  Exit Property
ErrorHandler:
  HandleError True, "BottleneckForm " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Set BottleneckForm(ByRef frm1 As Object)            ' FOR frmBottleneck
  On Error GoTo ErrorHandler

423:   Set m_frmBottleneck = frm1

  Exit Property
ErrorHandler:
  HandleError True, "BottleneckForm " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
Public Property Get frmClipForm() As Object                       ' FOR frmClip
  On Error GoTo ErrorHandler

432:   Set frmClipForm = m_ClipForm

  Exit Property
ErrorHandler:
  HandleError True, "frmClipForm " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Public Property Set frmClipForm(ByRef frm1 As Object)            ' FOR frmClip
  On Error GoTo ErrorHandler

442:   Set m_ClipForm = frm1

  Exit Property
ErrorHandler:
  HandleError True, "frmClipForm " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Public Property Get frmStep2() As Object                        ' FOR frmStep1
  On Error GoTo ErrorHandler

453:   Set frmStep2 = m_FormStep2

  Exit Property
ErrorHandler:
  HandleError True, "frmStep2 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Set frmStep2(ByRef frm2 As Object)              ' FOR frmStep1
  On Error GoTo ErrorHandler

463:   Set m_FormStep2 = frm2

  Exit Property
ErrorHandler:
  HandleError True, "frmStep2 " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Public Property Let EnableSelTool(ShouldEnable As Boolean)      ' FOR frmSelScreen.lbxThemes and imgToolDisable, In, Out
  On Error GoTo ErrorHandler

473:   m_EnableSelTool = ShouldEnable


  Exit Property
ErrorHandler:
  HandleError True, "EnableSelTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Get EnableSelTool() As Boolean                  ' FOR frmSelScreen.lbxThemes and imgToolDisable, In, Out
  On Error GoTo ErrorHandler

484:   EnableSelTool = m_EnableSelTool


  Exit Property
ErrorHandler:
  HandleError True, "EnableSelTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property


Public Property Let EnableDrawTool(ShouldEnable As Boolean)     ' FOR frmSelScreen.lbxThemes and imgToolDisable, In, Out
  On Error GoTo ErrorHandler
```

```
496:   m_EnableDrawTool = ShouldEnable

  Exit Property
ErrorHandler:
  HandleError True, "EnableDrawTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Get EnableDrawTool() As Boolean                ' FOR frmSelScreen.lbxThemes and imgToolDisable, In, Out
  On Error GoTo ErrorHandler

506:   EnableDrawTool = m_EnableDrawTool


  Exit Property
ErrorHandler:
  HandleError True, "EnableDrawTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Public Property Get ArcApp() As IMxApplication
  On Error GoTo ErrorHandler


518:   Set ArcApp = m_App


  Exit Property
ErrorHandler:
  HandleError True, "ArcApp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Property

Private Property Get IExtension_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
530:   IExtension_Name = "Linkages_Extension"

  Exit Property
ErrorHandler:
  HandleError True, "IExtension_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub IExtension_Startup(ByRef initializationData As Variant)
  On Error GoTo ErrorHandler
```

```
  If (TypeOf initializationData Is IMxApplication) Then
    Set m_App = initializationData
    Set DocumentEvents = m_App.Document

  ' Get the current user so the commands don't have to do this...
  Dim documentInfo As IDocumentInfo
    Set documentInfo = m_App.Document

    Set m_AoInitialize = New AoInitialize

  Dim licenseStatus As esriLicenseStatus
  'First try copy protection EngineGeoDB
    licenseStatus = CheckOutLicenses(esriLicenseProductCodeArcInfo)
    If (licenseStatus = esriLicenseNotLicensed) Then
    'Next try Desktop ArcEditor
      licenseStatus = CheckOutLicenses(esriLicenseProductCodeArcEditor)
    'For Desktop licenses we also need to consider them being unavailable
      If ((licenseStatus = esriLicenseNotLicensed) Or (licenseStatus = esriLicenseUnavailable)) Then
      'Last try Desktop ArcView
        licenseStatus = CheckOutLicenses(esriLicenseProductCodeArcView)
      End If
    End If

  'Take a look at the licenseStatus to see if it failed
  'Not licensed
    If (licenseStatus = esriLicenseNotLicensed) Then
      m_SpatialAnalyst = False
      m_LicenseDescription = "You are not licensed to run this product.  You are probably missing a license for Spatial Analyst."
  'The licenses needed are currently in use
    ElseIf (licenseStatus = esriLicenseUnavailable) Then
      m_SpatialAnalyst = False
      m_LicenseDescription = "There are insufficient licenses to run..."
  'The licenses unexpected license failure
    ElseIf (licenseStatus = esriLicenseFailure) Then
      m_SpatialAnalyst = False
      m_LicenseDescription = "Unexpected license failure!  Please contact your administrator..."
  'Already initialized (Initialization can only occur once)
    ElseIf (licenseStatus = esriLicenseAlreadyInitialized) Then
      m_SpatialAnalyst = True
      m_LicenseDescription = "Licenses checked out successfully..."
  'Everything was checkedout successfully
    ElseIf (licenseStatus = esriLicenseCheckedOut) Then
      m_SpatialAnalyst = True
      m_LicenseDescription = "Licenses checked out successfully..."
    End If
```

```
586:     m_EnableSelTool = False
587:     m_EnableDrawTool = False

589:   End If

  ' Find the toolboxextension
  Dim uID As New uID
593:   uID.Value = "{BD6262BC-D9D4-4B93-87E0-E442702D93E6}"
594:   Set m_arcToolboxExtension = m_App.FindExtensionByCLSID(uID)

  Exit Sub
ErrorHandler:
  HandleError True, "IExtension_Startup " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
 Private Function CheckOutLicenses(productCode As esriLicenseProductCode) As esriLicenseStatus
  On Error GoTo ErrorHandler


  Dim licenseStatus As esriLicenseStatus
605:   Set m_AoInitialize = New AoInitialize
606:   CheckOutLicenses = esriLicenseUnavailable

  'Check the productCode
609:   licenseStatus = m_AoInitialize.IsProductCodeAvailable(productCode)
610:   If (licenseStatus = esriLicenseAvailable) Then
    'Check the extensionCode
612:     licenseStatus = m_AoInitialize.IsExtensionCodeAvailable(productCode, esriLicenseExtensionCodeSpatialAnalyst)
613:     If (licenseStatus = esriLicenseAvailable) Then
      'Initialize the license
615:       licenseStatus = m_AoInitialize.Initialize(productCode)
616:       If (licenseStatus = esriLicenseCheckedOut) Then
          'Checkout the extension
618:         licenseStatus = m_AoInitialize.CheckOutExtension(esriLicenseExtensionCodeSpatialAnalyst)
619:       End If
620:     End If
621:   End If

623:   CheckOutLicenses = licenseStatus


  Exit Function
ErrorHandler:
  HandleError False, "CheckOutLicenses " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Private Sub IExtension_Shutdown()
```

```
  On Error GoTo ErrorHandler

633:   Set m_App = Nothing
634:   Set DocumentEvents = Nothing
635:   m_AoInitialize.CheckInExtension (esriLicenseExtensionCodeSpatialAnalyst)
636:   Set m_WildlandPolygon1 = Nothing
637:   Set m_WildlandPolygon2 = Nothing
638:   Set m_Corridor = Nothing
639:   Set m_SelForm = Nothing
640:   Set m_FormStep1 = Nothing
641:   Set m_FormStep2 = Nothing
642:   Set m_pPatchLayerStep1 = Nothing
643:   Set m_pPatchAttFieldStep1 = Nothing
644:   Set m_arcToolboxExtension = Nothing
645:   Set m_AoInitialize = Nothing
646:   Set m_PatchArray = Nothing

'MsgBox "Shutting down..."
  Exit Sub
ErrorHandler:
  HandleError True, "IExtension_Shutdown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get IExtensionConfig_ProductName() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
658:   IExtensionConfig_ProductName = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "IExtensionConfig_ProductName " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get IExtensionConfig_Description() As String
  On Error GoTo ErrorHandler

  If (m_App Is Nothing) Then Exit Property

670:   IExtensionConfig_Description = "Corridor Designer Tools   " & _
    "©2006-2008" & vbCrLf & _
    "ArcGIS Tools and Information for Designing Wildlife Corridors" & vbCrLf & _
    "Visit us at http://www.corridordesign.org/" & vbCrLf & vbCrLf & _
    "Requires Spatial Analyst  [License is " & IIf(m_SpatialAnalyst, "available]", _
    "not available]") & vbCrLf & m_LicenseDescription & vbCrLf
```

```
  Exit Property
ErrorHandler:
  HandleError True, "IExtensionConfig_Description " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get IExtensionConfig_State() As esriSystem.esriExtensionState
  On Error GoTo ErrorHandler

686:   IExtensionConfig_State = m_ExtensionState
687:   If (Not m_SpatialAnalyst) Then
688:     IExtensionConfig_State = esriESUnavailable
689:   End If

  Exit Property
ErrorHandler:
  HandleError True, "IExtensionConfig_State " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Function CheckIfToolboxLoaded(ToolboxName As String) As Boolean
  On Error GoTo ErrorHandler
  Dim arcToolbox As IArcToolbox
699:   Set arcToolbox = m_arcToolboxExtension.arcToolbox

  Dim booToolboxLoaded As Boolean
  Dim pGPTool As IGPTool
703:   Set pGPTool = arcToolbox.GetToolbyNameString(ToolboxName)
704:   booToolboxLoaded = (Not pGPTool Is Nothing)

706:   CheckIfToolboxLoaded = booToolboxLoaded
  Exit Function
ErrorHandler:
709:   CheckIfToolboxLoaded = False

End Function

Private Property Let IExtensionConfig_State(ByVal ExtensionState As esriSystem.esriExtensionState)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
717:   m_ExtensionState = ExtensionState

'  Dim arcToolbox As IArcToolbox
'  Set arcToolbox = m_arcToolboxExtension.arcToolbox
```

```
''  MsgBox App.Path & vbCrLf & "ArcToolbox is nothing? " & CStr(arcToolbox Is Nothing)
'
' Dim ToolboxIsLoaded As Boolean
' ToolboxIsLoaded = CheckIfToolboxLoaded("CorridorModelAZ")
'
'' MsgBox "Toolbox loaded? " & CStr(ToolboxIsLoaded)
'
'
'  Dim DLLPath As String
'  DLLPath = App.Path
'
'  Dim gpToolbox As IGPToolbox
'
''    MsgBox "About to test code..."
''    Dim strName As String
''    strName = c_toolboxName
''    Dim pTestTool As IGPToolbox
''    MsgBox "About to query toolbox"
''    Set pTestTool = arcToolbox.GetToolbyNameString(strName)
''    MsgBox pTestTool.Alias
''    MsgBox "Done testing code..."
''    MsgBox "About to test code..." & vbCrLf & "Extension Enabled = " & CStr(m_ExtensionState = esriESEnabled)
'
'  If (m_ExtensionState = esriESEnabled) Then   ' LOAD TOOLBOX IF ENABLED
'       Set gpToolbox = FindToolBox(DLLPath, c_toolboxName)
'
''       MsgBox "Hello" & vbCrLf & c_toolboxName
''       MsgBox "Is Corridor Designer toolbox nothing?  " & CStr(gpToolbox Is Nothing)
'
'       If (gpToolbox Is Nothing) Then
''          Set gpToolbox = CreateToolBox(DLLPath, c_toolboxName, c_toolboxName)
'         MsgBox "Was not able to find the Corridor Designer toolbox (CorridorDesigner Arizona.tbx) in the " & _
'            "following folder:" & vbCrLf & vbCrLf & "  -->  " & DLLPath & vbCrLf & vbCrLf & "The toolbox " & _
'            "must be located in this folder in order to be automatically loaded..." & vbCrLf & vbCrLf & _
'            "If your version is lost or corrupted, please visit http://www.corridordesign.org to " & _
'            "download the latest version.", vbInformation, _
'            "Caution - CorridorDesigner Toolbox is not available:"
'
'       Else
'         arcToolbox.AddToolbox gpToolbox
'         arcToolbox.Refresh
'       End If
'
'   Else  ' UNLOAD TOOLBOX IF EXTENSION UNLOADED
'       Set gpToolbox = FindToolBox(DLLPath, c_toolboxName)
'  '       MsgBox "Hello; in Unload Toolbox routine..." & vbCrLf & c_toolboxName
'
```

```
'       If (Not gpToolbox Is Nothing) Then
'       ' THIS SEEMS TO CRASH IF GPTOOLBOX IS NOT LOADED
'       ' THERE DOES NOT SEEM TO BE A WAY TO EASILY CHECK IF A TOOLBOX IS LOADED
'       ' HOWEVER, THERE IT DOES NOT SEEM TO LOAD IT TWICE, SO WE CAN FIRST LOAD IT THEN UNLOAD IT...
'       '
'       ' BACK OFF; SEEMS THAT THIS FUNCTION CRASHES ArcGIS IF THE TOOLBOX IS NOT MANUALLY OPENED BEFORE TURNING OFF EXTENSION
''         MsgBox "About to add duplicate version of toolbox..."
''         arcToolbox.AddToolbox gpToolbox
''         arcToolbox.Refresh
'         If ToolboxIsLoaded Then
''           MsgBox "Refreshing the toolbox..."
''           arcToolbox.Refresh
''           MsgBox "Try to open dockable window"
''           Dim pDocWin As IDockableWindow
''           Set pDocWin = arcToolbox.hWnd
''           MsgBox "About to remove toolbox..."
''           arcToolbox.RemoveToolbox gpToolbox
' '           arcToolbox.Refresh
'         End If
'       End If
'   End If

  '  Dim strToolName As String
  '  strToolName = "CorridorModelAZ"



  'MsgBox "FINISHED: --> GPToolboxExtension_IExtensionConfig_State"
  Exit Property
ErrorHandler:
  HandleError True, "IExtensionConfig_State " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property
 Public Function FindToolBox(sTBLocation As String, sTBName As String) As IGPToolbox
  On Error GoTo ErrorHandler


    'MsgBox "GPToolboxExtension_FindToolBox"
'  MsgBox sTBLocation & vbCrLf & sTBName

  ' CHECK IF ANY TOOLBOX FILES EXIST IN FOLDER
  Dim intCount As Integer
  Dim fileCollection As Collection
811:   Set fileCollection = Linkages.aml_func_mod.ReturnFiles(sTBLocation, "*.tbx", True)
812:   intCount = fileCollection.Item(1)
```

```
'  MsgBox "Looking for " & sTBName & " in " & sTBLocation & ":" & vbCrLf & _
    "  --> Found " & CStr(intCount) & " toolbox files..."

  ' FOR DEBUGGING
'  Dim theReport As String
'  theReport = CStr(intCount) & " toolbox files (*.tbx)" & vbCrLf
'  Dim strArray() As String
'  strArray = fileCollection.Item(2)
'    Dim anIndex As Integer
'  For anIndex = LBound(strArray) To UBound(strArray)
'    theReport = theReport & "  --> " & strArray(anIndex)
'  Next anIndex
'  MsgBox theReport

828:   If intCount = 0 Then
829:     Set FindToolBox = Nothing
    Exit Function
831:   End If

  'Create a toolbox workspace factory
  Dim ToolboxWorkspaceFactory As IWorkspaceFactory
835:   Set ToolboxWorkspaceFactory = New esriGeoprocessing.ToolboxWorkspaceFactory

  'Open a toolbox workspace
  Dim ToolboxWorkspace As IToolboxWorkspace
839:   Set ToolboxWorkspace = ToolboxWorkspaceFactory.OpenFromFile(sTBLocation, 0)

  Dim enumTB As IEnumGPToolbox
842:   Set enumTB = ToolboxWorkspace.Toolboxes
843:   enumTB.Reset

  Dim gpOutToolbox As IGPToolbox
846:   Set gpOutToolbox = Nothing

  Dim gpToolbox As IGPToolbox
849:   Set gpToolbox = enumTB.Next

851:   Do Until gpToolbox Is Nothing
'    MsgBox gpToolbox.Alias & vbCrLf & gpToolbox.PathName & vbCrLf & InStr(1, gpToolbox.PathName, sTBName, vbTextCompare)
853:     If InStr(1, gpToolbox.PathName, sTBName, vbTextCompare) > 0 Then
'    If (gpToolbox.Alias = sTBName) Then
855:             Set gpOutToolbox = gpToolbox
856:     End If
857:     Set gpToolbox = enumTB.Next
858:   Loop

860:   Set FindToolBox = gpOutToolbox
```

```
'  MsgBox "FINISHED: --> GPToolboxExtension_FindToolBox" & vbCrLf & "Found Toolbox = " & CStr(Not gpOutToolbox Is Nothing) & _
      vbCrLf & "Is IGPToolbox?  " & CStr(TypeOf gpOutToolbox Is IGPToolbox)


  Exit Function
ErrorHandler:
  HandleError True, "FindToolBox " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Function FindTool(gpToolbox As IGPToolbox, sFunctionName As String) As IGPTool
  On Error GoTo ErrorHandler

    'MsgBox "GPToolboxExtension_FindTool"
  Dim gpTool As IGPTool
  Dim gpOutTool As IGPTool

  Dim enumGPTool As IEnumGPTool
879:   Set enumGPTool = gpToolbox.Tools

881:   enumGPTool.Reset
882:   Set gpTool = enumGPTool.Next

884:   Do While Not gpTool Is Nothing
885:     If (gpTool.Name = sFunctionName) Then
886:       Set gpOutTool = gpTool
887:    End If
888:         Set gpTool = enumGPTool.Next
889:    Loop

891:   Set FindTool = gpOutTool
    'MsgBox "FINISHED: --> GPToolboxExtension_FindTool"

  Exit Function
ErrorHandler:
  HandleError False, "FindTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4

End Function

Private Property Get IPersistVariant_ID() As esriSystem.IUID
  On Error GoTo ErrorHandler

  Dim newUid As New uID
904:   newUid.Value = "Linkages.Extension"

906:   Set IPersistVariant_ID = newUid
```

```
  Exit Property
ErrorHandler:
  HandleError True, "IPersistVariant_ID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub IPersistVariant_Load(ByVal Stream As esriSystem.IVariantStream)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "IPersistVariant_Load " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub IPersistVariant_Save(ByVal Stream As esriSystem.IVariantStream)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "IPersistVariant_Save " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Class 20: toolDrawPoly

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "toolDrawPoly"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit
```

```
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\toolDrawPoly.cls"
Implements ICommand
Implements ITool

Private m_pApp As esriFramework.IApplication
Private m_pMxDoc As esriArcMapUI.IMxDocument
Private m_ExtensionConfig As IExtensionConfig

'Private m_FrmSelScreen As Linkages.frmSelScreen

Private m_pBitmap As IPictureDisp       'Bitmap for the tool
Private m_pCursor As IPictureDisp       'Cursor for the tool

'Public Property Set HomeForm(ByVal pForm As Linkages.frmSelScreen)
''  Set m_FrmSelScreen = pForm
'End Property
Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

23:   Set m_pBitmap = LoadResPicture(109, vbResBitmap)
24:   Set m_pCursor = LoadResPicture(101, vbResCursor)


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

36:   Set m_pBitmap = Nothing
37:   Set m_pCursor = Nothing
'  Set m_FrmSelScreen = Nothing
39:   Set m_pMxDoc = Nothing
40:   Set m_pApp = Nothing
41:   Set m_ExtensionConfig = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
  Dim ext As Linkages.Extension
54:   Set ext = m_ExtensionConfig
 ' MsgBox "ext is nothing? " & CStr(ext Is Nothing) & vbCrLf & "m_ExtensionConfig is nothing? " & _
      CStr(m_ExtensionConfig Is Nothing)

58:   If (Not m_ExtensionConfig Is Nothing) Then
59:     If (m_ExtensionConfig.State = esriESEnabled) Then
60:       ICommand_Enabled = ext.EnableDrawTool
61:     End If
62:   Else
63:     ICommand_Enabled = False
64:   End If
'  If (ICommand_Enabled) Then
'    Set m_pBitmap = LoadResPicture(103, vbResBitmap)
'  Else
'    Set m_pBitmap = LoadResPicture(104, vbResBitmap)
'  End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
80:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
91:   ICommand_Name = "CorridorDesigner_DrawPolygon"

  Exit Property
```

```
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
102:   ICommand_Caption = "Draw Polygon"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
113:   ICommand_Tooltip = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
124:   ICommand_Message = "Used in conjuction with Corridor Statistics..."

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
135:   ICommand_HelpFile = ""
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
146:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
157:   ICommand_Bitmap = m_pBitmap

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
168:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
179:   Set m_pApp = hook
```

```
180:   Set m_pMxDoc = m_pApp.Document
  Dim newUid As New uID
182:   newUid.Value = "Linkages.Extension"
183:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get ITool_Cursor() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
204:   ITool_Cursor = m_pCursor

  Exit Property
ErrorHandler:
  HandleError True, "ITool_Cursor " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
'  Linkages.frmSelScreen.txtCoords.Text = "X = " & x & ", Y = " & y
  Dim pMxDocument As esriArcMapUI.IMxDocument
  Dim pActiveView As esriCarto.IActiveView

219:   Set pMxDocument = m_pApp.Document
220:   Set pActiveView = pMxDocument.FocusMap
  Dim ext As Linkages.Extension
222:   Set ext = m_ExtensionConfig
```

```
  Dim theFormObject As Object
  Dim theForm As Linkages.frmSelScreen

227:   Set theFormObject = ext.aSelForm
228:   Set theForm = theFormObject

230:   theForm.cmdAccept.Enabled = False

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
233:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")
234:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors_orig")

  Dim pPolygon As IPolygon
  Dim pGraCont As IGraphicsContainer
  Dim pGraContSel As IGraphicsContainerSelect
  Dim pRubberPoly As IRubberBand
  Dim pElem As IElement

  ' QI for the IGraphicsContainerSelect interface on the document's activeview
243:   Set pGraCont = pMxDocument.ActiveView

  ' Create a new RubberPolygon
246:   Set pRubberPoly = New RubberPolygon

  'Check which mouse button was pressed...

250:   If Button = 1 Then ' If button 1 (left) then create a new polygon (TrackNew)

    ' Return a new Polygon from the tracker object using TrackNew
253:     Set pPolygon = pRubberPoly.TrackNew(pMxDocument.ActiveView.ScreenDisplay, Nothing)
254:     If Not pPolygon Is Nothing Then
      ' Create a new PolygonElement and set its Geometry
256:       Set pElem = New PolygonElement
257:       pElem.Geometry = pPolygon
      Dim pOrigElemProps As IElementProperties
259:       Set pOrigElemProps = pElem
260:       pOrigElemProps.Name = "delete_corridors_orig"

      'Add the new element at Z order zero
263:       pGraCont.AddElement pElem, 0
264:     Else
      Exit Sub
266:     End If
267:   Else ' If button 2 (right) then move an existing polygon (TrackExisting)

    ' QI for IGraphicsContainerSelect
270:     Set pGraContSel = pGraCont
```

```
    ' Check that we have some selected elements
273:     If pGraContSel.ElementSelectionCount > 0 Then
      ' If there is only one selected element then get it
275:       If pGraContSel.ElementSelectionCount = 1 Then
276:         Set pElem = pGraContSel.SelectedElement(0)

      ' If there is more than one selected element then get the dominant one
279:       ElseIf pGraContSel.ElementSelectionCount > 1 Then
280:         Set pElem = pGraContSel.DominantElement
281:       End If

      ' Check that the selected element is a PolygonElement
284:       If TypeOf pElem Is IPolygonElement Then
        ' Create a new RubberPolygon
286:         Set pRubberPoly = New RubberPolygon
        ' Retrieve the current geometry of our element
288:         Set pPolygon = pElem.Geometry
        ' Use track existing, passing in the Polygon's geometry by reference
        ' NB all User input is now handled by the RubberBand until the Mouse up occurs)
291:         pRubberPoly.TrackExisting pMxDocument.ActiveView.ScreenDisplay, Nothing, pPolygon
        ' Set the Element's geometry (pPoly has been altered by TrackExisting)
293:         pElem.Geometry = pPolygon
        ' Update the element
295:         pGraCont.UpdateElement pElem
296:       End If
297:     End If
298:   End If

'  MsgBox "Polygon is empty? " & pPolygon.IsEmpty

'  ' Refresh the active view
'  pMxDocument.ActiveView.Refresh



'
'
'
'
'  Dim pGContainer As IGraphicsContainer
'  Set pGContainer = m_pMxDoc.FocusMap
'


  Dim pNewPoly As IPolygon
  Dim pClone As IClone
```

```
318:   Set pClone = pPolygon
319:   Set pNewPoly = pClone.Clone

321:   Set theForm.thePolygon = pNewPoly
322:   theForm.cmdAccept.Enabled = True

  Dim pArea As IArea
325:   Set pArea = pPolygon

  Dim pElement As IElement
  Dim pPolygonElement As IPolygonElement
  Dim pSpatialReference As ISpatialReference
  Dim pGraphicElement As IGraphicElement
  Dim pElementProperties As IElementProperties

  'MsgBox "Spatial Ref is nothing = " & CStr(pPolygon.SpatialReference Is Nothing)

  'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
'  Set pElement = New PolygonElement
337:   Set pNewPoly.SpatialReference = m_pMxDoc.ActiveView.FocusMap.SpatialReference
'  pElement.Geometry = pNewPoly
'  Set pGraphicElement = pElement
'  Set pNewPoly.SpatialReference = m_pMxDoc.ActiveView.FocusMap.SpatialReference
'  pElement.Geometry = pNewPoly
'  Set pGraphicElement = pElement
'  Set pSpatialReference = pPolygon.SpatialReference
'  Set pGraphicElement.SpatialReference = pSpatialReference
'  Set pElementProperties = pElement
'  pElementProperties.Name = "delete_corridors"

348:   Set pElement = pElem
349:   Set pGraphicElement = pElement
350:   Set pElementProperties = pElement
351:   pElementProperties.Name = "delete_corridors_orig"

  ' ADD SYMBOLOGY TO GRAPHIC ELEMENT

  Dim pFillShapeElement As IFillShapeElement
356:   Set pFillShapeElement = pElement

  Dim pColor As IColor
359:   Set pColor = Linkages.MyGeneralOperations.MakeColorRGB(0, 200, 100)

  Dim pCartoLine As ICartographicLineSymbol
362:   Set pCartoLine = New CartographicLineSymbol
363:   With pCartoLine
364:     .Cap = esriLCSButt
```

```
365:      .Join = esriLJSBevel
366:      .Color = pColor
367:      .Width = 1
368:    End With

  Dim pLineFill As ILineFillSymbol
371:    Set pLineFill = New LineFillSymbol
372:    With pLineFill
373:      .Angle = -30
374:      .Separation = 3
375:      .Offset = 5
376:    End With
377:    Set pLineFill.LineSymbol = pCartoLine

  Dim pLineSymbol As ISimpleLineSymbol
380:    Set pLineSymbol = New SimpleLineSymbol

  Dim pLineColor As IColor
383:    Set pLineColor = Linkages.MyGeneralOperations.MakeColorRGB(0, 250, 100)
384:    pLineSymbol.Color = pLineColor
385:    pLineSymbol.Width = 2
386:    pLineSymbol.Style = esriSLSSolid
387:    pLineFill.Outline = pLineSymbol
388:    pFillShapeElement.Symbol = pLineFill

  ' ADD GRAPHIC TO GRAPHICS CONTAINER
'  pGraCont.AddElement pFillShapeElement, 0
392:    pGraCont.UpdateElement pElement
'
'  'Draw
395:    m_pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, pPolygon.Envelope

'    MsgBox "Selecting from " + pFeatureLayer.Name & vbCrLf & _
'        "Area = " & pArea.Area




'  theForm.txtCoords.Text = "X: " & pPoint.x & "  Y: " & pPoint.y
'  theForm.cmdAccept.Enabled = True

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseDown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseMove " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseUp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnDblClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnDblClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyDown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyUp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long) As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Function
ErrorHandler:
  HandleError True, "ITool_OnContextMenu " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Private Sub ITool_Refresh(ByVal hdc As esriSystem.OLE_HANDLE)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_Refresh " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Function ITool_Deactivate() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
485:   ITool_Deactivate = True

  Exit Function
ErrorHandler:
  HandleError True, "ITool_Deactivate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
```

Class 21: toolReturnCoords

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "toolReturnCoords"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit


Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\toolReturnCoords.cls"
Implements ICommand
Implements ITool

Private m_pApp As esriFramework.IApplication
Private m_pMxDoc As esriArcMapUI.IMxDocument
Private m_ExtensionConfig As IExtensionConfig

'Private m_FrmSelScreen As Linkages.frmSelScreen

Private m_pBitmap As IPictureDisp       'Bitmap for the tool
Private m_pCursor As IPictureDisp       'Cursor for the tool

'Public Property Set HomeForm(ByVal pForm As Linkages.frmSelScreen)
''  Set m_FrmSelScreen = pForm
'End Property
Private Sub Class_Initialize()
21:   Set m_pBitmap = LoadResPicture(108, vbResBitmap)
22:   Set m_pCursor = LoadResPicture(101, vbResCursor)

End Sub


Private Sub Class_Terminate()
28:   Set m_pBitmap = Nothing
29:   Set m_pCursor = Nothing
'  Set m_FrmSelScreen = Nothing
31:   Set m_pMxDoc = Nothing
32:   Set m_pApp = Nothing
33:   Set m_ExtensionConfig = Nothing
```

```
End Sub

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
  Dim ext As Linkages.Extension
42:   Set ext = m_ExtensionConfig
 ' MsgBox "ext is nothing? " & CStr(ext Is Nothing) & vbCrLf & "m_ExtensionConfig is nothing? " & _
      CStr(m_ExtensionConfig Is Nothing)

46:   If (Not m_ExtensionConfig Is Nothing) Then
47:     If (m_ExtensionConfig.State = esriESEnabled) Then
48:       ICommand_Enabled = ext.EnableSelTool
49:     End If
50:   Else
51:     ICommand_Enabled = False
52:   End If
'  If (ICommand_Enabled) Then
'    Set m_pBitmap = LoadResPicture(103, vbResBitmap)
'  Else
'    Set m_pBitmap = LoadResPicture(104, vbResBitmap)
'  End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
68:   ICommand_Checked = False

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
79:   ICommand_Name = "CorridorDesigner_SelObjects"
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
90:   ICommand_Caption = "Select Features"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
101:   ICommand_Tooltip = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
112:   ICommand_Message = "Used in conjuction with Corridor Statistics..."

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
```

```
123:   ICommand_HelpFile = ""

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
134:   ICommand_HelpContextID = 0

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Bitmap() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
145:   ICommand_Bitmap = m_pBitmap

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
156:   ICommand_Category = "Corridor Designer Tools"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here
167:   Set m_pApp = hook
168:   Set m_pMxDoc = m_pApp.Document
  Dim newUid As New uID
170:   newUid.Value = "Linkages.Extension"
171:   Set m_ExtensionConfig = m_pApp.FindExtensionByCLSID(newUid)

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Property Get ITool_Cursor() As esriSystem.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
192:   ITool_Cursor = m_pCursor

  Exit Property
ErrorHandler:
  HandleError True, "ITool_Cursor " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Property

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
'  Linkages.frmSelScreen.txtCoords.Text = "X = " & x & ", Y = " & y
  Dim pMxDocument As esriArcMapUI.IMxDocument
  Dim pActiveView As esriCarto.IActiveView

  Dim pPoint As esriGeometry.IPoint

  Dim pGContainer As IGraphicsContainer
```

```
210:   Set pGContainer = m_pMxDoc.FocusMap

  Dim ext As Linkages.Extension
213:   Set ext = m_ExtensionConfig

  Dim theFormObject As Object
  Dim theForm As Linkages.frmSelScreen

218:   Set theFormObject = ext.aSelForm
219:   Set theForm = theFormObject

  ' GET SELECTED LAYER
  Dim theItemIndex As Integer
223:   theItemIndex = theForm.lbxThemes.ListIndex
224:   If theItemIndex = -1 Then
225:     MsgBox "Please select a polygon layer from the list before clicking on the screen...", vbOKOnly, _
          "Unable To Select Polygon:"
    Exit Sub
228:   End If

  ' Get the Active View
231:   Set pMxDocument = m_pApp.Document
232:   Set pActiveView = pMxDocument.FocusMap

  'Get the point location of the mouse click event
235:   Set pPoint = pActiveView.ScreenDisplay.DisplayTransformation.ToMapPoint(X, Y)

  ' FIGURE OUT WHICH LAYER IS BEING SELECTED FROM
  Dim theItemName As String
239:   theItemName = theForm.lbxThemes.List(theItemIndex)

  ' DELETE CURRENT GRAPHICS NAMED "DELETE_CORRIDORS"
242:   Call Linkages.MyGeneralOperations.DeleteGraphicsByName(m_pMxDoc, "delete_corridors")

  Dim pPolygon As IPolygon

246:   If (theItemName = "1] <-- Select or Draw Graphic Polygon -->") Then

    Dim pExistingElement As IElement
    Dim pExistingEnumElement As IEnumElement
    Dim pExistingGeometry As IGeometry
    Dim booFoundPolygon As Boolean
252:         booFoundPolygon = False
    Dim intPolyCount As Integer
254:         intPolyCount = 0

256:     Set pExistingEnumElement = pGContainer.LocateElements(pPoint, 0)
```

```
257:     If (Not pExistingEnumElement Is Nothing) Then
258:       pExistingEnumElement.Reset
259:       Set pExistingElement = pExistingEnumElement.Next
260:       Do Until pExistingElement Is Nothing
261:         Set pExistingGeometry = pExistingElement.Geometry
262:         If TypeOf pExistingGeometry Is IPolygon Then
263:           booFoundPolygon = True
264:           intPolyCount = intPolyCount + 1
          Dim pClone As esriSystem.IClone
266:           Set pClone = pExistingElement.Geometry
267:           Set pPolygon = pClone.Clone
268:         End If
269:         Set pExistingElement = pExistingEnumElement.Next
270:       Loop
271:     End If

273:     If Not booFoundPolygon Then
274:       Beep
275:       Set theForm.thePolygon = Nothing
276:       theForm.cmdAccept.Enabled = False
      Exit Sub
278:     ElseIf intPolyCount > 1 Then
279:       MsgBox "Clicked on " & CStr(intPolyCount) & " graphic polygons!  Please click on only a " & _
          "single polygon..."
281:       Set pPolygon = Nothing
      Exit Sub
283:     End If

285:     Set theForm.thePolygon = pPolygon
286:     theForm.cmdAccept.Enabled = True

288:   Else

    Dim pFeatureLayer As IFeatureLayer
    Dim pFeatureClass As IFeatureClass

    ' SELECT FROM THAT LAYER
294:     Set pFeatureLayer = theForm.GetNameCollection.Item(theItemName)
295:     Set pFeatureClass = pFeatureLayer.FeatureClass

    ' GET AREA FIELD
    Dim pFlds As IFields
    Dim pFld As IField
    Dim lAIndex As Long

302:     Set pFlds = pFeatureClass.Fields
303:     lAIndex = pFlds.FindField("shape")
```

```
304:     Set pFld = pFlds.Field(lAIndex)

    Dim pFilter As ISpatialFilter
307:     Set pFilter = New SpatialFilter

309:     With pFilter
310:       Set .Geometry = pPoint
311:       .GeometryField = "SHAPE"
312:       .SpatialRel = esriSpatialRelIntersects
313:     End With

    Dim pFeatureCursor As IFeatureCursor
316:     Set pFeatureCursor = pFeatureClass.Search(pFilter, False)

    Dim lngCount As Long
319:     lngCount = pFeatureClass.FeatureCount(pFilter)

321:     If lngCount = 0 Then
322:       Beep
323:       Set theForm.thePolygon = Nothing
324:       theForm.cmdAccept.Enabled = False
      Exit Sub
326:     End If

    Dim pFeature As IFeature
329:     Set pFeature = pFeatureCursor.NextFeature

331:     Set pPolygon = pFeature.ShapeCopy
332:     Set theForm.thePolygon = pPolygon
333:     theForm.cmdAccept.Enabled = True

335:   End If

  Dim pArea As IArea
338:   Set pArea = pPolygon

  Dim pElement As IElement
  Dim pPolygonElement As IPolygonElement
  Dim pSpatialReference As ISpatialReference
  Dim pGraphicElement As IGraphicElement
  Dim pElementProperties As IElementProperties

  'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
347:   Set pElement = New PolygonElement
348:   pElement.Geometry = pPolygon
349:   Set pGraphicElement = pElement
350:   Set pSpatialReference = pPolygon.SpatialReference
```

```
351:   Set pGraphicElement.SpatialReference = pSpatialReference
352:   Set pElementProperties = pElement
353:   pElementProperties.Name = "delete_corridors"

  ' ADD SYMBOLOGY TO GRAPHIC ELEMENT

  Dim pFillShapeElement As IFillShapeElement
358:   Set pFillShapeElement = pElement

  Dim pColor As IColor
361:   Set pColor = Linkages.MyGeneralOperations.MakeColorRGB(0, 200, 100)

  Dim pCartoLine As ICartographicLineSymbol
364:   Set pCartoLine = New CartographicLineSymbol
365:   With pCartoLine
366:     .Cap = esriLCSButt
367:     .Join = esriLJSBevel
368:     .Color = pColor
369:     .Width = 1
370:   End With

'  Dim pLineProperties As ILineProperties
'  Set pLineProperties = pCartoLine
'  pLineProperties.Offset = 0
'  Dim hpe(6) As Double
'  hpe(0) = 0
'  hpe(1) = 7
'  hpe(2) = 1
'  hpe(3) = 1
'  hpe(4) = 1
'  hpe(5) = 0
'  Dim eLineTemplate As ITemplate
'  Set eLineTemplate = New Template
'  eLineTemplate.Interval = 10
'  Dim ix As Integer, jx As Integer
'  jx = 0
'  For ix = 1 To 3
'    eLineTemplate.AddPatternElement hpe(jx), hpe(jx + 1)
'    jx = jx + 2
'  Next ix
'  Set pLineProperties.Template = eLineTemplate

  Dim pLineFill As ILineFillSymbol
394:   Set pLineFill = New LineFillSymbol
395:   With pLineFill
396:     .Angle = -30
397:     .Separation = 3
```

```
398:     .Offset = 5
399:   End With
400:   Set pLineFill.LineSymbol = pCartoLine

  Dim pLineSymbol As ISimpleLineSymbol
403:   Set pLineSymbol = New SimpleLineSymbol

  Dim pLineColor As IColor
406:   Set pLineColor = Linkages.MyGeneralOperations.MakeColorRGB(0, 250, 100)
407:   pLineSymbol.Color = pLineColor
408:   pLineSymbol.Width = 2
409:   pLineSymbol.Style = esriSLSSolid
410:   pLineFill.Outline = pLineSymbol
411:   pFillShapeElement.Symbol = pLineFill

  ' ADD GRAPHIC TO GRAPHICS CONTAINER
414:   pGContainer.AddElement pFillShapeElement, 0

  'Draw
417:   m_pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, pPolygon.Envelope

'    MsgBox "Selecting from " + pFeatureLayer.Name & vbCrLf & _
'        "Area = " & pArea.Area




'  theForm.txtCoords.Text = "X: " & pPoint.x & "  Y: " & pPoint.y
'  theForm.cmdAccept.Enabled = True

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseDown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseMove " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseUp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnDblClick()
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnDblClick " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyDown " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyUp " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long) As Boolean
  On Error GoTo ErrorHandler
```

```
  ' TODO: Add your implementation here

  Exit Function
ErrorHandler:
  HandleError True, "ITool_OnContextMenu " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Private Sub ITool_Refresh(ByVal hdc As esriSystem.OLE_HANDLE)
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Sub
ErrorHandler:
  HandleError True, "ITool_Refresh " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Private Function ITool_Deactivate() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here
507:   ITool_Deactivate = True

  Exit Function
ErrorHandler:
  HandleError True, "ITool_Deactivate " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
```

Module 1: aml_func_mod

```
Attribute VB_Name = "aml_func_mod"
'             Environmental Systems Research Institute, Inc.
'Module Name: aml_func.bas
'Description: Used to perform AML-like functions for string manipipulation.
'             Also parses out tokens in a string to elements in an array.
'
'   Requires: ParseString and ParseStringR require that you Dim the named array arg in
'             the calling propgram: Must be a zero dimensioned string array:
'             Dim yourarray() as string

'    Methods: After   - Returns the substring to the right of the leftmost occurrence of
'                       searchStr
'             Before  - Returns the substring to the left of the leftmost occurrence of
```

```
'                       searchStr.
'        ExistFileDir - Returns True/False if file or Directory exists.
'             Extract - Returns an element from a list of elements
'             Index   - Returns the position of the leftmost occurrence of a specified
'                       string in a target string.
'             Keyword - Returns the position of a keyword within a list of keywords.
'             Search  - Returns the position of the first character of a search string
'                       in a target string.
'             Sort    - Returns a sorted a list of elements.
'             Subst   - Returns a string that has had one string subsituted for another
'             Substr  - Returns a substring that starts at a specified character position.
'             Token   - Allows tokens in a list to be manipulated.
'
'                       Count - the number of tokens in a list
'                       Find  - the position of a token in a list
'                       Move  - tokens in a list
'                       Insert- a new token into a list
'                       Delete- a token in a list
'                       Replace-one token in a list for another
'                       Switch -one token in a list for another
'
'        ParseString  - Populates string array with tokens in a string.
'        ParseStringR - Same as ParseString except blanks and commas are
'                       treated as delimiters.
'
'     History: DMA     - 03/04/97 - Original coding
'        Glenn Meister - 12/19/97 - Added ExistFileDir function
'
'    MODIFICATIONS:  JEFF JENNESS
'    GetFullFileString - ' CONVERTS 8.3 FILESTRING TO FULL TEXT
'            ReturnDir - GIVEN A STRING, RETURNS TEXT PRECEDING LAST "\" CHARACTER
'       ReturnFilename - GIVEN A STRING, RETURNS TEXT FOLLOWING LAST "\" CHARACTER
'          ReturnFiles - GIVEN A STRING PATH, RETURNS COLLECTION CONTAINING THE NUMBER OF FILES THAT MET CRITERIA
'                        AND ARRAY OF STRING FILENAMES.  OPTIONS FOR INCLUDE READ-ONLY, SYSTEM, HIDDEN FILES.
'        ReturnFolders - GIVENA STRING PATH, RETURNS COLLECTION CONTAINING THE NUMBER OF FOLDERS THAT MET CRITERIA
'                        AND ARRAY OF FOLDER FILENAMES.
'        ClipExtension - GIVEN A STRING, RETURNS TEXT PRECEDING LAST "." CHARACTER
'         SetExtension - GIVEN A STRING AND EXTENSION, RETURNS TEXT PRECEDING LAST "." CHARACTER, PLUS "." AND NEW EXTENSION
'     GetExtensionText - GIVEN A STRING, RETURNS TEXT FOLLOWING LAST "." CHARACTER
'       FieldIsNumeric - GIVEN A FIELD, RETURNS BOOLEAN
'        FieldIsString - GIVEN A FIELD, RETURNS BOOLEAN
'          FieldIsDate - GIVEN A FIELD, RETURNS BOOLEAN
'         FieldIsShape - GIVEN A FIELD, RETURNS BOOLEAN
'         InsertCommas - GIVEN A NUMBER OR STRING, INSERTS COMMAS AND RETURNS STRING
'      BasicTrimAvenue - GIVEN A STRING, RETURNS A STRING WITH VALUES TRIMMED OFF EACH SIDE
'   MakeUniqueFilename - GIVEN A STRING REPRESENTING A FILEPATH, RETURNS A UNIQUE FILENAME WHICH MAY HAVE NUMBERS APPENDED TO IT
'       GetTheUserName - JUST RETURNS THE USER NAME; CAN BE USED TO FIND THE "MY DOCUMENTS" FOLDER
```

```
'    TempPathLocation - RETURNS PATHNAME TO TEMP DIRECTORY
'     CreateShapefile - GIVEN PATHNAME AND SHAPE TYPE, RETURNS A pFeatureClass.  ESRI CODE, MODIFIED BY JENNESS
'    CreatedBASETable - GIVEN PATHNAME AND OPTIONAL FIELDS, RETURNS ITABLE
'        GetMxDocPath - GIVEN APPLICATION OBJECT, RETURNS PATHNAME OF MAP DOCUMENT
'         QuoteString - GIVEN A STRING, RETURNS A QUOTED VERSION OF THAT STRING
'    SubstituteString - GIVEN A STRING, A SEARCH STRING AND A SUBSTITUTE STRING, REPLACES ALL INSTANCES OF SEARCH TEXT WITH
'                       SUBSTITUTE TEXT IN THE ORIGINAL STRING
'ReturnArcGISInstallDir - RETURNS A STRING CONTAINING THE ARCGIS INSTALL LOCATION
'ReturnArcGISGeneralDir - RETURNS DIRECTORY STRING FOR INSTALL, LAST LOCATION, LAST BROWSED, LAST EXPORT AND LAST SAVE TO
' ReturnArcGISVersion - RETURNS ARCGIS VERSION NUMBER
'          FileExists - GIVEN A STRING PATH, RETURNS TRUE OR FALSE
' ReturnShapeTypeName - GIVEN A SHAPE TYPE ENUMERATION, RETURNS SHAPE NAME
' QueryForNewFilename - GIVEN A SAMPLE FILENAME, OPENS DIALOG AND QUERIES USER TO SPECIFY NEW FILENAME

Option Explicit

' FILENAME 8.3 FORMAT CONVERSION BELOW COPIED FROM http://forums.esri.com/Thread.asp?c=93&f=993&t=123512&mc=1#msgid400360
'   POSTED BY  Brett N.Meroney
'              GIS Programmer / Analyst
'              Integrated Laboratory Systems, Inc.
'              contractor to US-EPA Region VIII
'              Golden , CO
' NECESSARY TO CONVERT 8.3 FILENAMES TO USEABLE FILENAMES

Private Const strMAXPATH = 260
Private Const m_Quotation = """"

'
' Win32 Registry functions
'
Private Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" _
    (ByVal hKey As Long, _
     ByVal lpSubKey As String, _
     ByVal ulOptions As Long, _
     ByVal samDesired As Long, _
     phkResult As Long) _
     As Long
Private Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias "RegCreateKeyExA" _
    (ByVal hKey As Long, _
     ByVal lpSubKey As String, _
     ByVal Reserved As Long, _
     ByVal lpClass As String, _
     ByVal dwOptions As Long, _
     ByVal samDesired As Long, _
     lpSecurityAttributes As Any, _
     phkResult As Long, _
     lpdwDisposition As Long) _
```

```
     As Long
Private Declare Function RegQueryValueEx Lib "advapi32.dll" Alias "RegQueryValueExA" _
     (ByVal hKey As Long, _
     ByVal lpValueName As String, _
     ByVal lpReserved As Long, _
     lpType As Long, _
     lpData As Any, _
     lpcbData As Long) _
     As Long         ' Note that if you declare the lpData parameter as String, you must pass it By Value.
Private Declare Function RegQueryInfoKey Lib "advapi32.dll" Alias "RegQueryInfoKeyA" _
     (ByVal hKey As Long, _
     ByVal lpClass As String, _
     lpcbClass As Long, _
     lpReserved As Long, _
     lpcSubKeys As Long, _
     lpcbMaxSubKeyLen As Long, _
     lpcbMaxClassLen As Long, _
     lpcValues As Long, _
     lpcbMaxValueNameLen As Long, _
     lpcbMaxValueLen As Long, _
     lpcbSecurityDescriptor As Long, _
     lpftLastWriteTime As Any) _
     As Long
Private Declare Function RegEnumKeyEx Lib "advapi32.dll" Alias "RegEnumKeyExA" _
     (ByVal hKey As Long, _
     ByVal dwIndex As Long, _
     ByVal lpName As String, _
     lpcbName As Long, _
     lpReserved As Long, _
     ByVal lpClass As String, _
     lpcbClass As Long, _
     lpftLastWriteTime As Any) _
     As Long
Private Declare Function RegEnumValue Lib "advapi32.dll" Alias "RegEnumValueA" _
     (ByVal hKey As Long, _
      ByVal dwIndex As Long, _
     ByVal lpValueName As String, _
     lpcbValueName As Long, _
     lpReserved As Long, _
     lpType As Long, _
     lpData As Any, _
     lpcbData As Long) _
     As Long
Private Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" _
     (ByVal hKey As Long, _
     ByVal lpValueName As String, _
     ByVal Reserved As Long, _
```

```
     ByVal dwType As Long, _
     lpData As Any, _
     ByVal cbData As Long) _
     As Long         ' Note that if you declare the lpData parameter as String, you must pass it By Value.
Private Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" _
     (ByVal hKey As Long, _
      ByVal lpSubKey As String) _
     As Long
Private Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValueA" _
     (ByVal hKey As Long, _
     ByVal lpValueName As String) _
     As Long
Private Declare Function RegCloseKey Lib "advapi32.dll" _
     (ByVal hKey As Long) _
     As Long
'
' Constants for Windows 32-bit Registry API
'
Private Const HKEY_CLASSES_ROOT = &H80000000
Private Const HKEY_CURRENT_USER = &H80000001
Private Const HKEY_LOCAL_MACHINE = &H80000002
Private Const HKEY_USERS = &H80000003
Private Const HKEY_PERFORMANCE_DATA = &H80000004
Private Const HKEY_CURRENT_CONFIG = &H80000005
Private Const HKEY_DYN_DATA = &H80000006
'
' Reg result codes
'
Private Const REG_CREATED_NEW_KEY = &H1                      ' New Registry Key created
Private Const REG_OPENED_EXISTING_KEY = &H2                     ' Existing Key opened
'
' Reg Create Type Values...
'
Private Const REG_OPTION_RESERVED = 0           ' Parameter is reserved
Private Const REG_OPTION_NON_VOLATILE = 0       ' Key is preserved when system is rebooted
Private Const REG_OPTION_VOLATILE = 1           ' Key is not preserved when system is rebooted
Private Const REG_OPTION_CREATE_LINK = 2        ' Created key is a symbolic link
Private Const REG_OPTION_BACKUP_RESTORE = 4     ' open for backup or restore
'
' Reg Key Security Options
'
Private Const DELETE = &H10000
Private Const READ_CONTROL = &H20000
Private Const WRITE_DAC = &H40000
Private Const WRITE_OWNER = &H80000
Private Const SYNCHRONIZE = &H100000
Private Const STANDARD_RIGHTS_READ = (READ_CONTROL)
```

```
Private Const STANDARD_RIGHTS_WRITE = (READ_CONTROL)
Private Const STANDARD_RIGHTS_EXECUTE = (READ_CONTROL)
Private Const STANDARD_RIGHTS_REQUIRED = &HF0000
Private Const STANDARD_RIGHTS_ALL = &H1F0000
Private Const SPECIFIC_RIGHTS_ALL = &HFFFF
Private Const KEY_QUERY_VALUE = &H1
Private Const KEY_SET_VALUE = &H2
Private Const KEY_CREATE_SUB_KEY = &H4
Private Const KEY_ENUMERATE_SUB_KEYS = &H8
Private Const KEY_NOTIFY = &H10
Private Const KEY_CREATE_LINK = &H20
Private Const KEY_READ = ((STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY) And (Not SYNCHRONIZE))
Private Const KEY_WRITE = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY) And (Not SYNCHRONIZE))
Private Const KEY_ALL_ACCESS = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY Or
KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY Or KEY_CREATE_LINK) And (Not SYNCHRONIZE))
Private Const KEY_EXECUTE = ((KEY_READ) And (Not SYNCHRONIZE))

Private Const ERROR_SUCCESS = 0&
Private Const ERROR_MORE_DATA = 234
Private Const ERROR_NO_MORE_ITEMS = 259

Private Const REG_SZ = 1                         ' Unicode nul terminated string
'

Private Declare Function GetLongPathName Lib "Kernel32" Alias _
    "GetLongPathNameA" (ByVal lpszShortPath As String, _
    ByVal lpszLongPath As String, ByVal cchBuffer As Long) _
    As Long

Public Enum enumArcGISFolderTypes
   enumLastBrowsedLocation
   enumLastExportToLocation
   enumLastLocation
   enumLastSaveToLocation
   enumArcGISInstallLocation
End Enum

Private Declare Function GetUserName Lib "advapi32.dll" Alias "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) As Long
Private Declare Function GetTempPath Lib "Kernel32" Alias "GetTempPathA" (ByVal nBufferLength As Long, ByVal lpBuffer As String) As
Long
Const Quotation = """"

Public Function QueryForNewFilename(strSampleName As String) As String



End Function
```

```
Public Function ReturnShapeName(pEnum As esriGeometryType) As String

  Select Case pEnum
  Case 0
252:     ReturnShapeName = "Unknown Geometry"
  Case 1
254:     ReturnShapeName = "Point"
  Case 2
256:     ReturnShapeName = "Multipoint"
  Case 3
258:     ReturnShapeName = "Polyline"
  Case 4
260:     ReturnShapeName = "Polygon"
  Case 5
262:     ReturnShapeName = "Envelope"
  Case 6
264:     ReturnShapeName = "Path"
  Case 7
266:     ReturnShapeName = "Unknown Geometry"
  Case 9
268:     ReturnShapeName = "Multipatch"
  Case 11
270:     ReturnShapeName = "Ring"
  Case 13
272:     ReturnShapeName = "Line"
  Case 14
274:     ReturnShapeName = "Circular Arc"
  Case 15
276:     ReturnShapeName = "Bezier Curve"
  Case 16
278:     ReturnShapeName = "Elliptic Arc"
  Case 17
280:     ReturnShapeName = "Geometry Bag"
  Case 18
282:     ReturnShapeName = "Triangle Strip"
  Case 19
284:     ReturnShapeName = "Triangle Fan"
  Case 20
286:     ReturnShapeName = "Ray"
  Case 21
288:     ReturnShapeName = "Sphere"
  Case 22
290:     ReturnShapeName = "Triangles"
  Case Else
292:     ReturnShapeName = "Unknown Geometry"
```

```
293:   End Select

End Function


Public Function FileExists(strfilename As String) As Boolean

300:   FileExists = Dir(strfilename) <> ""

End Function

Public Function ReturnArcGISVersion() As String
' ORIGINAL FUNCTION:
' Public Function GetSetting(ByVal Section As String, ByVal Key As String, Optional ByVal Default As String = "") As String
   ' Section   Required. String expression containing the name of the section where the key setting is found.
   '           If omitted, key setting is assumed to be in default subkey.
   ' Key       Required. String expression containing the name of the key setting to return.
   ' Default   Optional. Expression containing the value to return if no value is set in the key setting.
   '           If omitted, default is assumed to be a zero-length string ("").
   ' ADAPTED BY JENNESS FROM Ask the VB Pro column, [Getting Started with VB", Spring 1998.]
   ' SET DEFAULT VALUE
314:    ReturnArcGISVersion = "Unable to determine ArcGIS Version!"

   Dim nRet As Long
   Dim hKey As Long
   Dim nType As Long
   Dim nBytes As Long
   Dim Buffer As String

   Dim strSection As String
323:    strSection = ""
   Dim strKey As String
325:    strKey = "RealVersion"
   Dim strDefault As String
327:    strDefault = ""
   Dim strDir As String

   ' Assume failure and set return to Default
'   GetSetting = Default

   ' Open key
334:    nRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey("ESRI", "ArcGIS", strSection), 0&, KEY_READ, hKey)
335:    If nRet = ERROR_SUCCESS Then
      ' Set appropriate value for default query
337:       If strKey = "*" Then strKey = vbNullString

      ' Determine how large the buffer needs to be
```

```
340:       nRet = RegQueryValueEx(hKey, strKey, 0&, nType, ByVal Buffer, nBytes)
341:       If nRet = ERROR_SUCCESS Then
         ' Build buffer and get data
343:          If nBytes > 0 Then
344:             Buffer = Space(nBytes)
345:             nRet = RegQueryValueEx(hKey, strKey, 0&, nType, ByVal Buffer, Len(Buffer))
346:             If nRet = ERROR_SUCCESS Then
              ' Trim NULL and return successful query!
348:                strDir = Left(Buffer, nBytes - 1)
349:                ReturnArcGISVersion = strDir
350:             End If
351:          End If
352:       End If
353:       Call RegCloseKey(hKey)
354:    End If


End Function

Public Function ReturnArcGISInstallDir() As String

' ORIGINAL FUNCTION:
' Public Function GetSetting(ByVal Section As String, ByVal Key As String, Optional ByVal Default As String = "") As String
   ' Section   Required. String expression containing the name of the section where the key setting is found.
   '           If omitted, key setting is assumed to be in default subkey.
   ' Key       Required. String expression containing the name of the key setting to return.
   ' Default   Optional. Expression containing the value to return if no value is set in the key setting.
   '           If omitted, default is assumed to be a zero-length string ("").
   ' ADAPTED BY JENNESS FROM Ask the VB Pro column, [Getting Started with VB", Spring 1998.]
   ' SET DEFAULT VALUE
370:    ReturnArcGISInstallDir = "Unable to determine ArcGIS Install location!"

   Dim nRet As Long
   Dim hKey As Long
   Dim nType As Long
   Dim nBytes As Long
   Dim Buffer As String

   Dim strSection As String
379:    strSection = ""
   Dim strKey As String
381:    strKey = "InstallDir"
   Dim strDefault As String
383:    strDefault = ""
   Dim strDir As String
```

```
   ' Assume failure and set return to Default
'   GetSetting = Default

   ' Open key
391:    nRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey("ESRI", "ArcGIS", strSection), 0&, KEY_READ, hKey)
392:    If nRet = ERROR_SUCCESS Then
      ' Set appropriate value for default query
394:       If strKey = "*" Then strKey = vbNullString

      ' Determine how large the buffer needs to be
397:       nRet = RegQueryValueEx(hKey, strKey, 0&, nType, ByVal Buffer, nBytes)
398:       If nRet = ERROR_SUCCESS Then
         ' Build buffer and get data
400:          If nBytes > 0 Then
401:             Buffer = Space(nBytes)
402:             nRet = RegQueryValueEx(hKey, strKey, 0&, nType, ByVal Buffer, Len(Buffer))
403:             If nRet = ERROR_SUCCESS Then
               ' Trim NULL and return successful query!
405:                strDir = Left(Buffer, nBytes - 1)
406:                If Right(strDir, 1) = "\" Then strDir = Left(strDir, Len(strDir) - 1)
407:                ReturnArcGISInstallDir = strDir
408:             End If
409:          End If
410:       End If
411:       Call RegCloseKey(hKey)
412:    End If

End Function

Public Function ReturnArcGISGeneralDir(pArcGISFolderType As enumArcGISFolderTypes) As String

' ORIGINAL FUNCTION:
' Public Function GetSetting(ByVal Section As String, ByVal Key As String, Optional ByVal Default As String = "") As String
   ' Section   Required. String expression containing the name of the section where the key setting is found.
   '           If omitted, key setting is assumed to be in default subkey.
   ' Key       Required. String expression containing the name of the key setting to return.
   ' Default   Optional. Expression containing the value to return if no value is set in the key setting.
   '           If omitted, default is assumed to be a zero-length string ("").
   ' ADAPTED BY JENNESS FROM Ask the VB Pro column, [Getting Started with VB", Spring 1998.]
   ' SET DEFAULT VALUE


  Dim nRet As Long
  Dim hKey As Long
  Dim nType As Long
  Dim nBytes As Long
  Dim Buffer As String
```

```
  Dim strKey As String
  Dim strDir As String

  Select Case pArcGISFolderType
    Case enumArcGISInstallLocation
439:       ReturnArcGISGeneralDir = "Unable to determine ArcGIS Install location!"
440:       nRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, "Software\ESRI\ArcGIS", 0&, KEY_READ, hKey)
441:       strKey = "InstallDir"
    Case enumLastSaveToLocation
443:       ReturnArcGISGeneralDir = "Unable to determine ArcGIS Last Saved To location!"
444:       nRet = RegOpenKeyEx(HKEY_CURRENT_USER, "Software\ESRI\ArcCatalog\Settings", 0&, KEY_READ, hKey)
445:       strKey = "LastSaveToLocation"
    Case enumLastBrowsedLocation
447:       ReturnArcGISGeneralDir = "Unable to determine ArcGIS Last Browsed To location!"
448:       nRet = RegOpenKeyEx(HKEY_CURRENT_USER, "Software\ESRI\ArcCatalog\Settings", 0&, KEY_READ, hKey)
449:       strKey = "LastBrowseLocation"
    Case enumLastExportToLocation
451:       ReturnArcGISGeneralDir = "Unable to determine ArcGIS Last Exported To location!"
452:       nRet = RegOpenKeyEx(HKEY_CURRENT_USER, "Software\ESRI\ArcCatalog\Settings", 0&, KEY_READ, hKey)
453:       strKey = "LastExportToLocation"
    Case enumLastLocation
455:       ReturnArcGISGeneralDir = "Unable to determine ArcGIS Last location!"
456:       nRet = RegOpenKeyEx(HKEY_CURRENT_USER, "Software\ESRI\ArcCatalog\Settings", 0&, KEY_READ, hKey)
457:       strKey = "LastLocation"
458:   End Select

  ' Assume failure and set return to Default
  '   GetSetting = Default

   ' Open key
464:    If nRet = ERROR_SUCCESS Then
      ' Set appropriate value for default query
466:       If strKey = "*" Then strKey = vbNullString

      ' Determine how large the buffer needs to be
469:       nRet = RegQueryValueEx(hKey, strKey, 0&, nType, ByVal Buffer, nBytes)
470:       If nRet = ERROR_SUCCESS Then
         ' Build buffer and get data
472:          If nBytes > 0 Then
473:             Buffer = Space(nBytes)
474:             nRet = RegQueryValueEx(hKey, strKey, 0&, nType, ByVal Buffer, Len(Buffer))
475:             If nRet = ERROR_SUCCESS Then
              ' Trim NULL and return successful query!
477:                strDir = Left(Buffer, nBytes - 1)
478:                If Right(strDir, 1) = "\" Then strDir = Left(strDir, Len(strDir) - 1)
479:                ReturnArcGISGeneralDir = strDir
480:             End If
```

```
481:          End If
482:       End If
483:       Call RegCloseKey(hKey)
484:    End If

End Function

Private Function SubKey(ByVal strCompany As String, ByVal strAppName As String, Optional ByVal Section As String = "") As String

   ' Build SubKey from known values
491:    SubKey = "Software\" & strCompany & "\" & strAppName
492:    If Len(Section) Then
493:       SubKey = SubKey & "\" & Section
494:    End If

End Function
Public Function ReturnFolders(ByVal DirPath As String, Optional ByVal FolderName As String) As Collection

  ' RETURNS A COLLECTION CONTAINING:
  '    A) Count of files that met criteria
  '    B) String Array of full folder filenames

  Dim returnCollection As Collection
504:   Set returnCollection = New Collection

  Dim strFolders() As String

  Dim strSearchString As String
509:   If FolderName = "" Then
510:     strSearchString = "*"
511:   Else
512:     strSearchString = FolderName
513:   End If

  Dim strSearchPath As String

517:   DirPath = Trim(DirPath)
518:   If Not Right(DirPath, 1) = "\" Then DirPath = DirPath & "\"
519:   strSearchPath = DirPath & strSearchString

  Dim strFoundFolder As String
  Dim anIndex As Integer
523:   anIndex = -1

525:   strFoundFolder = Dir(strSearchPath, 16)

527:   Do While Not strFoundFolder = ""
```

```
528:     If (Not strFoundFolder = ".") And (Not strFoundFolder = "..") Then
529:       If GetAttr(DirPath & strFoundFolder) = vbDirectory Then
530:         anIndex = anIndex + 1
        ReDim Preserve strFolders(anIndex)
532:         strFolders(anIndex) = DirPath & strFoundFolder
533:       End If
534:     End If
535:     strFoundFolder = Dir
536:   Loop

538:   returnCollection.Add (UBound(strFolders) - LBound(strFolders) + 1)
539:   returnCollection.Add (strFolders)

541:   Set ReturnFolders = returnCollection

End Function

Public Function ReturnFiles(ByVal DirPath As String, Optional ByVal FileName As String, Optional ByVal IncludeReadOnlyFiles As
Boolean, _
                            Optional ByVal IncludeHiddenFiles As Boolean, Optional ByVal IncludeSystemFiles As Boolean) As Collection

  ' RETURNS A COLLECTION CONTAINING:
  '    A) Count of files that met criteria
  '    B) String Array of full filenames

  Dim returnCollection As Collection
553:   Set returnCollection = New Collection

  Dim strFiles() As String
  Dim intOption As Integer
557:   intOption = 0
558:   If IncludeReadOnlyFiles Then intOption = intOption + 1
559:   If IncludeHiddenFiles Then intOption = intOption + 2
560:   If IncludeSystemFiles Then intOption = intOption + 4

  Dim strSearchString As String
563:   If FileName = "" Then
564:     strSearchString = "*"
565:   Else
566:     strSearchString = FileName
567:   End If

  Dim strSearchPath As String

571:   DirPath = Trim(DirPath)
572:   If Not Right(DirPath, 1) = "\" Then DirPath = DirPath & "\"
573:   strSearchPath = DirPath & strSearchString
```

```
  Dim strFoundFile As String
  Dim anIndex As Integer

578:   strFoundFile = Dir(strSearchPath, intOption)

580:   Do While Not strFoundFile = ""
    ReDim Preserve strFiles(anIndex)
582:     strFiles(anIndex) = DirPath & strFoundFile
583:     strFoundFile = Dir
584:     anIndex = anIndex + 1
585:   Loop

587:   returnCollection.Add anIndex
'  If anIndex = 0 Then
'    returnCollection.Add 0
'  Else
'    returnCollection.Add (UBound(strFiles) - LBound(strFiles) + 1)
'  End If
593:   returnCollection.Add (strFiles)

595:   Set ReturnFiles = returnCollection

End Function


Public Function QuoteString(strInput As String) As String

  Dim strQuoted As String
603:   strQuoted = m_Quotation & SubstituteString(strInput, Chr(34), m_Quotation & m_Quotation) & m_Quotation

605:   QuoteString = strQuoted

End Function

Public Function ContainsString(strInText As String, strSearchText As String) As Boolean

611:   If strInText = "" Or strSearchText = "" Then
612:     ContainsString = False
613:   Else
614:     ContainsString = InStr(1, strInText, strSearchText, vbTextCompare) > 0
615:   End If

End Function

Public Function SubstituteString(strFullText As String, strSearchText As String, strSubstituteText As String)
```

```
  Dim lngIndex As Long
  Dim lngStartPos As Long
  Dim lngSearchLength As Long
  Dim lngFullLength As Long

626:   lngStartPos = 1
627:   lngSearchLength = Len(strSearchText)
628:   lngFullLength = Len(strFullText)

630:   lngIndex = InStr(lngStartPos, strFullText, strSearchText, vbTextCompare)

  Dim strNewString As String
633:   If lngIndex = 0 Then
634:     strNewString = strFullText
635:   Else
636:     strNewString = Left(strFullText, lngIndex - 1) & strSubstituteText
637:     lngStartPos = lngIndex + lngSearchLength
638:   End If

640:   Do While lngIndex <> 0
641:     lngIndex = InStr(lngStartPos, strFullText, strSearchText, vbTextCompare)
642:     If lngIndex = 0 Then
643:       strNewString = strNewString & Right(strFullText, lngFullLength - lngStartPos + 1)
644:     Else
645:       strNewString = strNewString & Mid(strFullText, lngStartPos, lngIndex - lngStartPos) & strSubstituteText
646:       lngStartPos = lngIndex + lngSearchLength
647:     End If

649:   Loop
650:   SubstituteString = strNewString

End Function

Public Function GetMxDocPath(pApp As IApplication) As String

  Dim pTemplates As ITemplates
  Dim lTempCount As Long

659:   Set pTemplates = pApp.Templates
660:   lTempCount = pTemplates.Count

  ' The document is always the last item
663:   GetMxDocPath = pTemplates.Item(lTempCount - 1)

End Function

Public Function CreatedBASETable(strFullName As String, Optional pFields As IFields) As ITable
```

```
' createDBF: simple function to create a DBASE file.
' note: the name of the DBASE file should not contain the .dbf extension
' ESRI Sample; modified by Jenness August 20 2007

  Dim strName As String
  Dim strFolder As String

676:   strFolder = aml_func_mod.ReturnDir(strFullName)
677:   strName = aml_func_mod.ReturnFilename(strFullName)
678:   If Right(strName, 4) = ".dbf" Then strName = Left(strName, Len(strName) - 4)

  ' Open the Workspace
  Dim pFWS As IFeatureWorkspace
  Dim pWorkspaceFactory As IWorkspaceFactory
  Dim fs As Object
  Dim pFieldsEdit As IFieldsEdit
  Dim pFieldEdit As IFieldEdit
  Dim pField As IField

688:   Set pWorkspaceFactory = New ShapefileWorkspaceFactory
689:   If Not aml_func_mod.ExistFileDir(strFolder) Then
690:     MsgBox "Folder does not exist: " & vbCr & strFolder
    Exit Function
692:   End If

694:   Set pFWS = pWorkspaceFactory.OpenFromFile(strFolder, 0)

  ' if a fields collection is not passed in then create one
697:   If pFields Is Nothing Then
    ' create the fields used by our object
699:     Set pFields = New Fields
700:     Set pFieldsEdit = pFields
701:     pFieldsEdit.FieldCount = 1

    'Create text Field
704:     Set pField = New Field
705:     Set pFieldEdit = pField
706:     With pFieldEdit
707:         .Precision = 8
708:         .Name = "Unique_ID"
709:         .Type = esriFieldTypeInteger
710:     End With
711:     Set pFieldsEdit.Field(0) = pField
712:   End If

  Dim strString As String
```

```
  Dim lngIndex As Long
  Dim pFieldInfo As IFieldInfo

718:   Set pField = pFields.Field(0)


'  MsgBox pField.Name
'  MsgBox pField.Scale
'  MsgBox pField.Precision
'  MsgBox pField.Type

'  For lngIndex = 0 To pFields.FieldCount - 1
'    strString = strString + "-----------------" & "  Field Name = " & pFields.Field(lngIndex).Name & vbCrLf & _
'        "  Field Scale = " & pFields.Field(lngIndex).Scale & vbCrLf & "  Precision = " & pFields.Field(lngIndex).Precision & _
'        vbCrLf & "  Field Type = " & CStr(pFields.Field(lngIndex).Type) & vbCrLf
'  Next lngIndex
'  MsgBox "Problem with workspace? " & CStr(pFWS Is Nothing) & vbCrLf & "Filename = " & strFullName & vbCrLf & _
'        "Folder = " & strFolder & vbCrLf & "strName = " & strName & vbCrLf & pFields.FieldCount & " fields..." & vbCrLf & _
'        "File already exists? " & CBool(FileExists(strFullName)) & vbCrLf & strString

735:   Set CreatedBASETable = pFWS.CreateTable(strName, pFields, Nothing, Nothing, "")

End Function


Public Function CreateShapefile(sPath As String, sName As String, pSpatialReference As ISpatialReference, strShapeType As String) As
IFeatureClass   ' Don't include filename!

742:   If Right(sPath, 4) = ".shp" Then sPath = ReturnDir(sPath)
743:   If Right(sName, 4) = ".shp" Then sName = Left(sName, Len(sName) - 4)

  ' SET GEOMETRY TYPE, AND EXIT IF NOT ONE OF STANDARD OPTIONS
  Dim pGeomDef As IGeometryDef
  Dim pGeomDefEdit As IGeometryDefEdit
748:   Set pGeomDef = New GeometryDef
749:   Set pGeomDefEdit = pGeomDef
750:   With pGeomDefEdit
    Select Case strShapeType
      Case "Polygon", "polygon"
753:         .GeometryType = esriGeometryPolygon
      Case "Polyline", "polyline"
755:         .GeometryType = esriGeometryPolyline
      Case "Point", "point"
757:         .GeometryType = esriGeometryPoint
      Case "Multipoint", "multipoint" Or "MultiPoint"
759:         .GeometryType = esriGeometryMultipoint
      Case "Multipatch", "multipatch" Or "MultiPatch"
```

```
761:          .GeometryType = esriGeometryMultiPatch
      Case Else
763:          MsgBox "Invalid Shape Type [" & strShapeType & "]!  This function is only written to generate " & _
             "Point, Polyline, Polygon, Multipoint or Multipatch shapefiles...", vbCritical, "Invalid Shape Type:"
765:      End Select
'     Set .SpatialReference = New UnknownCoordinateSystem
767:      Set .SpatialReference = pSpatialReference
768:    End With

  ' Open the folder to contain the shapefile as a workspace
  Dim pFWS As IFeatureWorkspace
  Dim pWorkspaceFactory As IWorkspaceFactory
773:    Set pWorkspaceFactory = New ShapefileWorkspaceFactory

775:    If Not pWorkspaceFactory.IsWorkspace(sPath) Then
776:      MsgBox "Unable to create Feature Class:" & vbCrLf & _
            sPath & " is not a valid workspace...", , "Failed to Create Feature Class:"
778:      Set CreateShapefile = Nothing
    Exit Function
780:    End If

782:    Set pFWS = pWorkspaceFactory.OpenFromFile(sPath, 0)

  ' Set up a simple fields collection
  Dim pFields As IFields
  Dim pFieldsEdit As IFieldsEdit
787:    Set pFields = New Fields
788:    Set pFieldsEdit = pFields

  Dim pField As IField
  Dim pFieldEdit As IFieldEdit

  ' Make the shape field
  ' it will need a geometry definition, with a spatial reference
795:    Set pField = New Field
796:    Set pFieldEdit = pField
797:    pFieldEdit.Name = "Shape"
798:    pFieldEdit.Type = esriFieldTypeGeometry

800:    Set pFieldEdit.GeometryDef = pGeomDef
801:    pFieldsEdit.AddField pField

  ' Add an ID field
804:    Set pField = New Field
805:    Set pFieldEdit = pField
806:    With pFieldEdit
807:        .length = 8
```

```
808:       .Name = "Unique_ID"
809:       .Type = esriFieldTypeInteger
810:       .Precision = 0
811:   End With
812:   pFieldsEdit.AddField pField

  ' Create the shapefile
  ' (some parameters apply to geodatabase options and can be defaulted as Nothing)
  Dim booFileExists As Boolean
  Dim strCheckString As String
818:   If Right(sPath, 1) = "\" Then
819:     strCheckString = sPath & sName & ".shp"
'    MsgBox sPath & sName & ".shp" & vbCrLf & "File Exists? " & CStr(Dir(sPath & sName & ".shp") <> "")
821:   Else
822:     strCheckString = sPath & "\" & sName & ".shp"
'    MsgBox sPath & "\" & sName & ".shp" & vbCrLf & "File Exists? " & CStr(Dir(sPath & "\" & sName & ".shp") <> "")
824:   End If

826:   booFileExists = (Dir(strCheckString) <> "")

828:   If booFileExists Then
829:     MsgBox "The following file already exists:" & vbCrLf & vbCrLf & strCheckString & vbCrLf & vbCrLf & _
           "Please select a new filename...", , "Bailing out of 'CreateShapefile':"
831:     Set CreateShapefile = Nothing
    Exit Function
833:   End If

  Dim pFeatClass As IFeatureClass
836:   Set pFeatClass = pFWS.CreateFeatureClass(sName, pFields, Nothing, _
                                        Nothing, esriFTSimple, "Shape", "")

839:   Set CreateShapefile = pFeatClass

End Function




Public Function TempPathLocation() As String

  Dim sBuffer As String
849:   sBuffer = Space(strMAXPATH)
850:   If GetTempPath(strMAXPATH, sBuffer) <> 0 Then
851:     TempPathLocation = Left$(sBuffer, _
      InStr(sBuffer, vbNullChar) - 1)
853:   Else
854:     TempPathLocation = ""
```

```
855:   End If

End Function


Public Function GetTheUserName() As String

  Dim sBuffer As String
  Dim sUName As String
  Dim lSize As Long
865:   sBuffer = Space$(255)
866:   lSize = Len(sBuffer)
867:   Call GetUserName(sBuffer, lSize)
868:   If lSize > 0 Then
869:     sUName = Left$(sBuffer, lSize)
870:   Else
871:     sUName = vbNullString
872:   End If
873:   GetTheUserName = BasicTrimAvenue(sUName, "", Chr(0))     ' NEED TO PEEL OFF THAT LAST ODD CHARACTER

End Function

Public Function BasicTrimAvenue(aString As String, aTrimLeft As String, aTrimRight As String) As String

879:   Do While (aString <> "") And (InStr(1, aTrimRight, Right(aString, 1), vbTextCompare) > 0)
880:     aString = Left(aString, Len(aString) - 1)
881:   Loop
882:   Do While (aString <> "") And (InStr(1, aTrimLeft, Left(aString, 1), vbTextCompare) > 0)
883:     aString = Right(aString, Len(aString) - 1)
884:   Loop

886:   BasicTrimAvenue = aString

End Function

Public Function InsertCommas(InputValue As Variant) As String

  Dim theString As String
893:   theString = CStr(InputValue)

  Dim theDecLocation As Long
896:   theDecLocation = InStr(theString, ".")

  Dim HasDecimal As Boolean
899:   HasDecimal = theDecLocation > 0

  Dim theLength As Long
```

```
902:   theLength = Len(theString)

  Dim theBaseNumber As String
  Dim theRemainder As String

907:   If HasDecimal Then
908:     theRemainder = Right(theString, theLength - theDecLocation)
909:     theBaseNumber = Left(theString, theDecLocation - 1)
910:   Else
911:     theRemainder = ""
912:     theBaseNumber = theString
913:   End If

  Dim theCount As Long
916:   theCount = Len(theBaseNumber)

  Dim theCommaString As String

920:   If theCount > 3 Then
    Dim anIndex As Long
922:     For anIndex = (theCount - 2) To 1 Step -3
923:       theCommaString = Mid(theBaseNumber, anIndex, 3) & "," & theCommaString
924:       If anIndex < 4 Then
925:         theCommaString = Left(theBaseNumber, anIndex - 1) & "," & theCommaString
926:       End If
927:     Next anIndex

929:     Do While Right(theCommaString, 1) = ","
930:       theCommaString = Left(theCommaString, Len(theCommaString) - 1)
931:     Loop
932:     Do While Left(theCommaString, 1) = ","
933:       theCommaString = Right(theCommaString, Len(theCommaString) - 1)
934:     Loop
935:   Else
936:     theCommaString = theBaseNumber
937:   End If

939:   If HasDecimal Then
940:     theCommaString = theCommaString & "." & theRemainder
941:   End If

943:   InsertCommas = theCommaString

End Function

Public Function ClipExtension(strPathName As String) As String
```

```
  Dim strDirPath As String
  Dim strDirTokens() As String

952:   aml_func_mod.ParseString strPathName, strDirTokens, "."
953:   strDirPath = strDirTokens(0)

955:   If (UBound(strDirTokens) = 0) Then
956:     ClipExtension = strDirPath
957:   Else
    Dim anIndex As Long
959:     For anIndex = 1 To (UBound(strDirTokens) - 1)
960:       strDirPath = strDirPath & "." & strDirTokens(anIndex)
961:     Next anIndex
962:     ClipExtension = strDirPath
963:   End If

End Function

Public Function FieldIsNumeric(pTheField As IField) As Boolean

  Dim theFieldType As esriFieldType
970:   theFieldType = pTheField.Type

972:   FieldIsNumeric = _
    (theFieldType = esriFieldTypeSmallInteger) Or (theFieldType = esriFieldTypeDouble) Or (theFieldType = esriFieldTypeInteger) Or _
          (theFieldType = esriFieldTypeSingle)

End Function
Public Function FieldIsString(pTheField As IField) As Boolean

  Dim theFieldType As esriFieldType
980:   theFieldType = pTheField.Type

982:   FieldIsString = (theFieldType = esriFieldTypeString)

End Function
Public Function FieldIsDate(pTheField As IField) As Boolean

  Dim theFieldType As esriFieldType
988:   theFieldType = pTheField.Type

990:   FieldIsDate = (theFieldType = esriFieldTypeDate)

End Function
Public Function FieldIsShape(pTheField As IField) As Boolean

  Dim theFieldType As esriFieldType
```

```
996:   theFieldType = pTheField.Type

998:   FieldIsShape = (theFieldType = esriFieldTypeGeometry)

End Function

Public Function SetExtension(strPathName As String, strExtension As String) As String

  Dim theClippedPath As String
1005:   SetExtension = ClipExtension(strPathName) & "." & strExtension

End Function

Public Function GetExtensionText(strPathName As String) As String

  Dim strDirPath As String
  Dim strDirTokens() As String

1014:   aml_func_mod.ParseString strPathName, strDirTokens, "."
1015:   If UBound(strDirTokens) = 0 Then
1016:     GetExtensionText = ""
1017:   Else
1018:     GetExtensionText = strDirTokens(UBound(strDirTokens))
1019:   End If

End Function

Public Function GetFullFileString(str83Type As String) As String

  ' ADAPTED FROM BRETT MERONEY'S POST ABOVE

  Dim lLen As Long
  Dim sBuffer As String

1030:   sBuffer = String$(strMAXPATH, 0)
1031:   lLen = GetLongPathName(str83Type, sBuffer, Len(sBuffer))
1032:   If lLen > 0 And Err.Number = 0 Then
1033:     GetFullFileString = Left$(sBuffer, lLen)
1034:   Else
1035:     GetFullFileString = str83Type
1036:   End If

End Function

Public Function ReturnDir(strPathName As String) As String

  Dim strDirPath As String
```

```
  Dim strDirTokens() As String

1045:   aml_func_mod.ParseString strPathName, strDirTokens, "\"
1046:   strDirPath = strDirTokens(0)

1048:   If (UBound(strDirTokens) = 0) Then
1049:     ReturnDir = strDirPath
1050:   Else
    Dim anIndex As Long
1052:     For anIndex = 1 To (UBound(strDirTokens) - 1)
1053:       strDirPath = strDirPath & "\" & strDirTokens(anIndex)
1054:     Next anIndex
1055:     ReturnDir = strDirPath
1056:   End If

1058:   ReturnDir = ReturnDir & "\"

End Function


Public Function ReturnFilename(strPathName As String) As String

  Dim strDirPath As String
  Dim strDirTokens() As String

1068:   aml_func_mod.ParseString strPathName, strDirTokens, "\"
1069:   ReturnFilename = strDirTokens(UBound(strDirTokens))

End Function
Public Sub ParseString(Str As String, strArray() As String, Delim As String)

' Populates a named string array with elements in a string. Each array element
' contains one word. Multiple words ' within single quotes ' are treated as
' one word. NOTE: Use parseStringR if you want both commas and blanks to be
' treated as delimiting characters.
'
' Before calling this Sub you must declare your array in the calling program
' As String, with no bounds.
'
' Dim myarray() As String
' mystring = "ARC YES, POLY NO, TICS YES"
' parseString (mystring), myarray, ","
' Returns:
' array(0) = ARC YES
' array(1) = POLY NO
' array(2) = TICS YES
```

```
' parseString(mystring),myarray," "
' Returns:
' array(0) = ARC
' array(1) = YES,
' array(2) = POLY
' array(3) = NO,
' array(4) = TICS
' array(5) = YES,

' mystring = "'Universe,Medium','Helvetica,Bold','Times,Medium'"
' parsestring(mystring),myarray,","
' Returns:
' array(0) = Universe,Medium
' array(1) = Helvetica,Bold
' array(2) = Times,Medium

' Dim counters

Dim i As Long
Dim tokenlen As Long
Dim tmpstr As String
Dim position As Long
Dim length As Long

'Dim variables to keep track of embedded quotes

Dim switch As Long
Dim position1 As Long
Dim position2 As Long
Dim pair As Long

On Error Resume Next

' If string contains no elements raise error
1124:   If Trim(Subst(Str, Delim)) = "" Then
1125:     Err.Raise vbObjectError + 1, "aml_func.ParseString", _
    "StringPassed"
    Exit Sub
1128:   End If

' intialize array. Warning: This will overwrite any data elements currently
' stored in this named array

  ReDim strArray(0)

'intializer counters and tracking variables
```

```
1137:   pair = False
1138:   switch = 0
1139:   length = Len(Str)
1140:   position = 1
1141:   i = 0
1142:   tmpstr = Str

'check each character in the array. If it is a quote, store if it is first or last
' 0 = havent read one yet
' 1 = read first single quote
' 2 = read second single quote

1149:   Do While position < length
1150:     If Mid(tmpstr, position, 1) = "'" Then
1151:       If Not (switch = 1) Then
1152:         switch = 1
1153:         position1 = position
1154:         pair = False
1155:       Else
1156:         switch = 2
1157:         position2 = position
1158:         pair = True
1159:       End If
1160:     End If

' if last char read was last in a pair of quotes, store contents between first and last
' in current array element and reset tracking variables

1165:     If pair = True Then
1166:       Mid(tmpstr, position1, 1) = " "
1167:       Mid(tmpstr, position2, 1) = " "
1168:       strArray(i) = Mid(tmpstr, position1, position2 - position1)
1169:       strArray(i) = Trim(strArray(i))
1170:       pair = False
1171:       switch = 0

' check to see if we are reading till the next single quote. If switch = 0, we are not
' if not check if the next character is a delimiter. if it is store everything to the left
' replace everything to the left of original str with blanks so we can safely use LEFT
' function then trim the blanks

1178:     Else
1179:       If switch = 0 Then
1180:         If Mid(tmpstr, position, 1) = Delim Then
1181:           strArray(i) = Left(tmpstr, position)
1182:           tokenlen = Len(strArray(i))
1183:           strArray(i) = Trim(strArray(i))
```

```
1184:           If Not (Len(strArray(i)) = 0) Then
1185:             Mid(tmpstr, 1, tokenlen) = String(tokenlen, " ")
          ReDim Preserve strArray(LBound(strArray) To UBound(strArray) + 1)
1187:             i = i + 1
1188:           End If
1189:         End If
1190:       End If
1191:     End If
1192:     position = position + 1
1193:   Loop
1194:   strArray(i) = Trim(tmpstr)

' we have populated our array, now remove the delimiters from each element

1198:   position = 1
1199:   For i = 0 To UBound(strArray)
1200:     position = Len(strArray(i))
1201:     If Mid(strArray(i), position, 1) = Delim Then
1202:       Mid(strArray(i), position, 1) = " "
1203:       strArray(i) = Trim(strArray(i))
1204:     End If
1205:   Next i

End Sub

Public Function After(Str As String, SearchStr As String) As String

' Returns the substring of Str to the right of the leftmost
' occurrence of the searchStr.
Dim position As Long
Dim length As Long

1216:   position = InStr(Str, SearchStr)
1217:   length = Len(SearchStr)
1218:   If Not (position = 0) Then
1219:    After = Mid(Str, position + length)
1220:   End If

End Function

Public Function Before(Str As String, SearchStr As String) As String

' Returns the substring of Str to the left of the leftmost
' occurrence of the searchStr.
Dim position As Long
Dim length As Long
```

```
1231:   position = InStr(Str, SearchStr)
1232:   length = Len(SearchStr)
1233:   If Not (position = 0) Then
1234:    Before = Mid(Str, 1, position - 1)
1235:   End If

End Function

Function ExistFileDir(sTest As String) As Boolean

'Checks for the existance of a File or Directory
  Dim af As Long
1243:   af = -1
  On Error Resume Next
1245:   af = GetAttr(sTest)
1246:   ExistFileDir = (af <> -1)

End Function



Public Function MakeUniqueFilename(strfilename As String) As String

1254:   If Not ExistFileDir(strfilename) Then
1255:     MakeUniqueFilename = strfilename
    Exit Function
1257:   Else

    Dim theCounter As Long
1260:     theCounter = 1

    Dim theFilename As String
    Dim theBaseName As String
    Dim thePointPos As Long
    Dim theExtension As String

1267:     If InStr(1, Right(strfilename, 5), ".", vbTextCompare) > 0 Then
1268:       thePointPos = InStrRev(strfilename, ".", -1, vbTextCompare)
1269:       theExtension = Right(strfilename, Len(strfilename) - (thePointPos - 1))
1270:       theFilename = Left(strfilename, thePointPos - 1)
1271:     Else
1272:       theExtension = ""
1273:       theFilename = strfilename
1274:     End If

1276:     theBaseName = theFilename
```

```
1278:     Do While ExistFileDir(theFilename & theExtension)
1279:       theCounter = theCounter + 1
1280:       theFilename = theBaseName & "_" & CStr(theCounter)
1281:     Loop

1283:     MakeUniqueFilename = theFilename & theExtension

1285:   End If

End Function

Public Function Extract(ElemNum As Long, ElemList As String) As String

' extracts an element from a list of elements

Dim strArray() As String

1295:   ParseStringR (ElemList), strArray
1296:   If ElemNum > UBound(strArray) + 1 Then
   Exit Function
1298:   End If

1300:   If ElemNum = 0 Then
   Exit Function
1302:   End If
1303:   Extract = strArray(ElemNum - 1)

End Function

Public Function Index(Str As String, SearchStr As String) As Long

' Returns the position of the leftmost occurrence of searcStr in str.

1311:   Index = InStr(Str, SearchStr)

End Function

Public Sub ParseStringR(Str As String, strArray() As String, Optional ReturnQuoted)

' Populates a named string arrary with elements in a string. Each array element
' contains one word. Multiple words ' within single quotes ' are treated as
' one word. Treats both blanks and commas as delimters NOTE: Use parseString
' to specify a specific delimiting character.

' ReturnQuoted - indicates if elements are to be returned quoted.
' FALSE - DEFAULT return elements unquoted
' TRUE - return elements quoted
```

```
' Before calling this function you must declare your array in the calling program
' As String, with no bounds.
'
' Dim myarray() As String
' mystring = "ARC YES, POLY NO, TICS YES"
' parseStringR(mystring),myarray

' Returns:
' array(0) = ARC
' array(1) = YES
' array(2) = POLY
' array(3) = NO
' array(4) = TICS
' array(5) = YES

' mystring = "'Universe,Medium','Helvetica,Bold','Times,Medium'"

' parsestringR(mystring),myarray
' Returns:
' array(0) = Universe,Medium
' array(1) = Helvetica,Bold
' array(2) = Times,Medium

' parsestring(mystring,myarray,TRUE)
' Returns:
' array(0) = 'Universe,Medium'
' array(1) = 'Helvetica,Bold'
' array(2) = 'Times,Medium'

' Dim counters

Dim i As Long
Dim tokenlen As Long
Dim switch As Long
Dim position1 As Long
Dim position2 As Long
Dim pair As Long
Dim parseAgain As Boolean
Dim tmpstr As String
Dim position As Long
Dim length As Long

On Error Resume Next

' If string contains no elements raise error
1371:   If Trim(Subst(Str, ",")) = "" Then
```

```
1372:     Err.Raise vbObjectError + 1, "aml_func.ParseStringR", _
    "StringPassed"
    Exit Sub
1375:   End If

' intialize counters and tracking variables
  ReDim strArray(0)
1379:   pair = False
1380:   switch = 0
1381:   length = Len(Str)
1382:   position = 1
1383:   i = 0
1384:   tmpstr = Str

1386:   If IsMissing(ReturnQuoted) Then
1387:     ReturnQuoted = False
1388:   End If
1389:   If Not (ReturnQuoted = False) Then
1390:     ReturnQuoted = True
1391:   End If

' check each character in the array. If it is a quote, store if it is first or last
' 0 = havent read one yet
' 1 = read first single quote
' 2 = just read second single quote

1398:   Do While position < length
1399:     If Mid(tmpstr, position, 1) = "'" Then
1400:       If Not (switch = 1) Then
1401:       switch = 1
1402:       position1 = position
1403:       pair = False
1404:     Else
1405:       switch = 2
1406:       position2 = position
1407:       pair = True
1408:       End If
1409:     End If

' if last char read was last in a pair of single quotes, store contents between first
' and last in current array element and reset tracking variables

1414:     If pair = True Then
1415:       Mid(tmpstr, position1, 1) = " "
1416:       Mid(tmpstr, position2, 1) = " "
1417:       strArray(i) = Mid(tmpstr, position1, position2 - position1)
1418:       strArray(i) = Trim(strArray(i))
```

```
1419:       pair = False
1420:       switch = 0

' check to see if we are reading till the next single quote. If switch = 0, we are not
' if not check if the next character is a delimiter. if it is store everything to the left
' replace everything to the left of original str with blanks so we can safely use LEFT
' function then trim the blanks

1427:     Else
1428:       If switch = 0 Then
1429:         If Mid(tmpstr, position, 1) = "," Or Mid(tmpstr, position, 1) = " " Then
1430:           strArray(i) = Left(tmpstr, position)
1431:           tokenlen = Len(strArray(i))
1432:           strArray(i) = Trim(strArray(i))
1433:           If Not (Len(strArray(i)) = 0) Then
1434:             Mid(tmpstr, 1, tokenlen) = String(tokenlen, " ")
            ReDim Preserve strArray(LBound(strArray) To UBound(strArray) + 1)
1436:             i = i + 1
1437:           End If
1438:         End If
1439:       End If
1440:     End If
1441:     position = position + 1
1442:   Loop
1443:     strArray(i) = Trim(tmpstr)

' we have populated our array, now remove the delimiters from each element
' set parseAgain flag if there are any blank elements

1448:   parseAgain = False
1449:   position = 1

1451:   For i = 0 To UBound(strArray)
1452:     position = Len(strArray(i))
1453:     If Mid(strArray(i), position, 1) = "," Then
1454:       Mid(strArray(i), position, 1) = " "
1455:       strArray(i) = Trim(strArray(i))
1456:       If Len(strArray(i)) = 0 Then
1457:         parseAgain = True
1458:       End If
1459:     End If
1460:   Next i

' now remove any blank elements

1464:   If parseAgain = True Then
1465:     tmpstr = ""
```

```
1466:     For i = 0 To UBound(strArray)
1467:       If Not (Len(strArray(i))) = 0 Then
1468:         tmpstr = tmpstr & "'" & strArray(i) & "'" & ","
1469:       End If
1470:     Next i
1471:     ParseString (tmpstr), strArray, ","
1472:   End If

1474:   If ReturnQuoted = True Then
1475:     For i = 0 To UBound(strArray)
1476:      strArray(i) = "'" & strArray(i) & "'"
1477:     Next i
1478:   End If

End Sub

Public Function Keyword(Str As String, SearchStr As String) As Long

' Returns the position of a string within a list of keywords.
' converts Str and searchStr to upper case before comparing
'  0  if keyword not found
' -1  if string is ambiguous - mutiple occurances of same keyword
'  n  position of keyword in string

Dim strArray() As String
Dim i As Long
Dim keywordCnt As Long

1494:   ParseStringR (Str), strArray
1495:   keywordCnt = 0
1496:     For i = 0 To UBound(strArray)
1497:     If UCase(SearchStr) = UCase(strArray(i)) Then
1498:       Keyword = i + 1
1499:       keywordCnt = keywordCnt + 1
1500:     End If
1501:     Next i

1503:   If keywordCnt > 1 Then
1504:     Keyword = -1
1505:   End If

End Function

Public Function Search(Str, SearchStr) As Long

' Returns the position of the first character in Str
' which occurs in searchStr.
```

```
Dim strArray() As String
Dim i As Long
Dim Index As Long
Dim firstchar
Dim InString As Boolean

1520:   Index = 1
  ReDim strArray(Len(Str))

1523:   For i = 0 To Len(SearchStr) - 1
1524:     firstchar = Mid(SearchStr, i + 1, 1)
1525:     Index = InStr(Str, firstchar)
1526:     strArray(Index) = i + 1
1527:   Next i

1529:   InString = False

1531:   For i = 1 To UBound(strArray)
1532:     If Not (strArray(i)) = "" Then
1533:     InString = True
1534:     Exit For
1535:     Else
1536:     End If
1537:   Next i

1539:   If InString = False Then
1540:     Search = 0
1541:   Else
1542:     Search = i
1543:   End If

End Function

Public Function Sort(Str As String, Optional SortOption, Optional SortType) As String

' Returns a string of sorted elements

1551:   If IsMissing(SortOption) Then
1552:     SortOption = "-ASCEND"
1553:   ElseIf Not (UCase(SortOption) = "-DESCEND") Then
1554:     SortOption = "-ASCEND"
1555:   End If

1557:   If IsMissing(SortType) Then
1558:     SortType = "-CHARACTER"
1559:   ElseIf Not (UCase(SortType) = "-NUMERIC") Then
```

```
1560:     SortType = "-CHARACTER"
1561:   End If

1563:   If (UCase(SortType)) = "-NUMERIC" Then
1564:     Call Sort_Num(Str, SortOption)
1565:   Else
1566:     Call Sort_Char(Str, SortOption, True)
1567:   End If
1568:     Sort = Str

End Function

Private Function Sort_Num(Str As String, SortOption) As String

' Sort function - performs a numerical sort
' Ref: Selectionsort Chapter8 of VB Algorithms; Rod Stephens

Dim i As Long
Dim j As Long
Dim min As Long
Dim max As Long
Dim best_value As String
Dim best_j As Long
Dim sortArray() As String
Dim sorted As String

1586:   ParseStringR (Str), sortArray

1588:   min = LBound(sortArray)
1589:   max = UBound(sortArray)

1591:   For i = min To max - 1
1592:     best_value = sortArray(i)
1593:     best_j = i

1595:     For j = i + 1 To max
1596:       If Val(sortArray(j)) < Val(best_value) Then
1597:         best_value = sortArray(j)
1598:         best_j = j
1599:       End If
1600:     Next j

1602:     sortArray(best_j) = sortArray(i)
1603:     sortArray(i) = best_value
1604:   Next i

1606:   If UCase(SortOption) = "-DESCEND" Then
```

```
1607:     For i = max To min Step -1
1608:       sorted = sorted & sortArray(i) & ","
1609:     Next i
1610:   Else
1611:     For i = min To max
1612:       sorted = sorted & sortArray(i) & ","
1613:     Next i
1614:   End If

1616:   Mid(sorted, Len(sorted), 1) = " "
1617:   Str = sorted
1618:   Sort_Num = sorted

End Function

Private Function Sort_Char(Str As String, SortOption, Optional ReturnQuoted) As String

' Sort function - performs a character sort
' Ref: Selectionsort Chapter8 of VB Algorithms; Rod Stephens

Dim i As Long
Dim j As Long
Dim min As Long
Dim max As Long
Dim best_value As String
Dim best_j As Long
Dim sortArray() As String
Dim sorted As String

1636: If IsMissing(ReturnQuoted) Then
1637:   ReturnQuoted = False
1638: End If
1639: If Not (ReturnQuoted = False) Then
1640:   ReturnQuoted = True
1641: End If

1643: ParseStringR (Str), sortArray, ReturnQuoted

1645: min = LBound(sortArray)
1646: max = UBound(sortArray)

1648:   For i = min To max - 1
1649:     best_value = sortArray(i)
1650:     best_j = i

1652:     For j = i + 1 To max
1653:       If sortArray(j) < best_value Then
```

```
1654:       best_value = sortArray(j)
1655:       best_j = j
1656:       End If
1657:     Next j

1659:     sortArray(best_j) = sortArray(i)
1660:     sortArray(i) = best_value
1661:   Next i

1663:   If UCase(SortOption) = "-DESCEND" Then
1664:     For i = max To min Step -1
1665:       sorted = sorted & sortArray(i) & " "
1666:     Next i
1667:   Else
1668:     For i = min To max
1669:       sorted = sorted & sortArray(i) & " "
1670:     Next i
1671:   End If
1672:   Mid(sorted, Len(sorted), 1) = " "

1674:   Str = sorted
1675:   Sort_Char = sorted

End Function

Public Function Subst(Str As String, SearchChar As String, Optional ReplaceChar) As String

' Replaces all occurances of specified char in string.

Dim complete As Boolean
Dim i As Long
Dim first As String
Dim last As String
Dim tmpstr As String
Dim position As Long

1690: tmpstr = Str

1692: position = 1
1693: complete = False

1695: If IsMissing(ReplaceChar) Then
1696:   Do Until complete = True
1697:     position = InStr(position, tmpstr, SearchChar)
1698:     If position = 0 Or position > Len(tmpstr) Then
1699:       complete = True
1700:     Else
```

```
1701:       first = Before(tmpstr, SearchChar)
1702:       last = After(tmpstr, SearchChar)
1703:       tmpstr = first & last
1704:     End If
1705:   Loop
1706: End If

1708: Do Until complete = True
1709:   position = InStr(position, tmpstr, SearchChar)
1710:   If position = 0 Or position > Len(tmpstr) Then
1711:     complete = True
1712:   Else
1713:     Mid(tmpstr, position, Len(ReplaceChar)) = ReplaceChar
1714:     position = position + Len(ReplaceChar)
1715:   End If
1716: Loop

Subst = tmpstr

End Function

Public Function Substr(Str As String, position As Long, Optional NumChars) As String

'extracts a substring starting at a specified character position.

1726: If IsMissing(NumChars) Then
1727:   If position = 0 Or position > Len(Str) Then
1728:     Substr = ""
1729:   Else
1730:     Substr = Mid(Str, position)
1731:   End If
1732: Else
1733:   If position = 0 Or position > Len(Str) Then
1734:     Substr = ""
1735:   Else
1736:    Substr = Mid(Str, position, NumChars)
1737:   End If
1738: End If

End Function

Public Function Token(ElemList As String, Arg As String, ParamArray OtherArgs()) As Variant

' Performs various functions for string manipulation

Dim strArray() As String
Dim i As Long
```

```
Dim temp As String
Dim from_elem  As Long
Dim to_elem As Long
Dim start_elem As Long
Dim insertStr As String
Dim DELETE As Long
Dim SearchStr As String

' Parse ElemList out to strarray
' Select TOKEN argument and perform function

1759:   ParseStringR (ElemList), strArray
1760:   Arg = Subst(Arg, "-")

  Select Case UCase(Arg)

  ' Count - returns the number of tokens in a list
    Case "COUNT"
1766:       Token = UBound(strArray) + 1

  ' Find <token> - returns the position of a token in a list
    Case "FIND"
1770:       SearchStr = OtherArgs(0)
1771:       Token = 0
1772:       For i = 0 To UBound(strArray)
1773:         If UCase(SearchStr) = UCase(strArray(i)) Then
1774:           Token = i + 1
1775:         End If
1776:       Next i

  ' Move <from_position> <to_position> - moves a token in the list
    Case "MOVE"
1780:       from_elem = OtherArgs(0) - 1
1781:       to_elem = OtherArgs(1)
1782:       temp = strArray(to_elem)
1783:       strArray(to_elem) = strArray(from_elem)
1784:       For i = from_elem To to_elem - 1
1785:         strArray(i) = strArray(i + 1)
1786:         Next i
1787:          strArray(i) = temp
1788:       For i = 0 To UBound(strArray) - 1
1789:         Token = Token & strArray(i) & ","
1790:       Next i

  ' Insert <position> - inserts a new token at <position> in the list
    Case "INSERT"
      ReDim Preserve strArray(UBound(strArray) + 1)
```

```
1795:       start_elem = OtherArgs(0) - 1
1796:       insertStr = OtherArgs(1)
1797:       For i = UBound(strArray) To start_elem Step -1
1798:         strArray(i) = strArray(i - 1)
1799:         Next i
1800:         strArray(start_elem) = insertStr
1801:         For i = 0 To UBound(strArray) - 1
1802:         Token = Token & strArray(i) & ","
1803:       Next i

  ' Delete <position> - removes the token at <position> from the list.
    Case "DELETE"
1807:       DELETE = OtherArgs(0) - 1
1808:       For i = DELETE To UBound(strArray) - 1
1809:         strArray(i) = strArray(i + 1)
1810:       Next i
1811:       For i = 0 To UBound(strArray) - 1
1812:         Token = Token & strArray(i) & ","
1813:       Next i

  ' Replace <position> <new_string> - replaces the token at <position> with the
  ' <new_string>.
    Case "REPLACE"
1818:       strArray(OtherArgs(1) - 1) = OtherArgs(0)
1819:       For i = 0 To UBound(strArray)
1820:         Token = Token & strArray(i) & ","
1821:       Next i

  ' Switch <position_1> <position_2> - moves token at <position_1> to <position_2> and moves
  ' token at <position_2> to <position_1>.
    Case "SWITCH"
1826:       from_elem = OtherArgs(0) - 1
1827:       to_elem = OtherArgs(1) - 1
1828:       temp = strArray(to_elem)
1829:       strArray(to_elem) = strArray(from_elem)
1830:       strArray(from_elem) = temp
1831:       For i = 0 To UBound(strArray)
1832:         Token = Token & strArray(i) & ","
1833:       Next i

    Case Else
1836:   End Select

End Function
```

Module 2: CorridorAnalysisFunctions

```
Attribute VB_Name = "CorridorAnalysisFunctions"
Option Explicit
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\CorridorAnalyses.bas"


' ApplyCorridorSymbol - GIVEN POLYGON ELEMENT, RETURNS AN IFillShapeElement
' StatsForStrings - GIVEN IStringArray AND OPTIONAL IDblArray, RETURNS ARRAY OF COUNTS, PROPORTIONS AND SIZES OF EACH UNIQUE ID
' CheckCollectionForKey - GIVEN pCollection and STRING, RETURNS BOOLEAN INDICATING WHETHER COLLECTION HAS THAT KEY OR NOT
' StatsForDates - GIVEN IVariantArray, RETURNS IVariantArray CONTAINING EARLIEST AND LATEST DATE
' GridHistogram - GIVEN RASTER, LOW VAL, HIGH VAL AND NUMBER BINS, RETURN
' BottleneckAnalysis - RUN BOTTLENECK ANALYSIS AND OPEN DIALOG

Public Sub BottleneckAnalysisDemo(pApp As IApplication)
  On Error GoTo ErrorHandler

  Dim pSpatialReferenceFactory2 As ISpatialReferenceFactory2
16:   Set pSpatialReferenceFactory2 = New SpatialReferenceEnvironment ' The file test.prj contains a spatial reference in its WKT aka
string representation.
  Dim pSpRef As ISpatialReference
18:   Set pSpRef = pSpatialReferenceFactory2.CreateSpatialReference(esriSRProjCS_NAD1983UTM_12N)

  ' PERFORM ANALYSIS
  Dim pDataArray As esriSystem.IVariantArray
22:   Set pDataArray = Linkages.CorridorSampleData.MakeData
  Dim frmGraph As Linkages.frmWidthGraph
24:   Set frmGraph = New Linkages.frmWidthGraph

26:   Set frmGraph.ArcApplication = pApp
27:   Set frmGraph.SpatReference = pSpRef
28:   Set frmGraph.GraphNumbers = pDataArray

30:   frmGraph.Frame.Visible = True

  Exit Sub
ErrorHandler:
  HandleError True, "BottleneckAnalysisDemo " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Sub BottleneckAnalysis(pParamDetails As esriSystem.IVariantArray, pMxDoc As IMxDocument, pApp As IApplication, _
  pCorridor As IPolygon, pStartBlock As IPolygon, pEndBlock As IPolygon, ProgressForm As Linkages.frmJenProgressPercent)
  On Error GoTo ErrorHandler
```

```
  Dim anIndex As Long

  Dim pSpRef As ISpatialReference
45:   Set pSpRef = pCorridor.SpatialReference

  ' PERFORM ANALYSIS
  Dim pDataArray As esriSystem.IVariantArray
'  Set pDataArray = Linkages.CorridorSampleData.MakeData
  Dim pStartPoly As IPolygon4
  Dim pEndPoly As IPolygon4
  Dim pCorPoly As IPolygon4
53:   Set pStartPoly = pStartBlock
54:   Set pEndPoly = pEndBlock
55:   Set pCorPoly = pCorridor

57:   Set pDataArray = RunBottleneck(pStartPoly, pEndPoly, pCorPoly, pMxDoc, pApp, ProgressForm)
'  MsgBox "Data Array is Nothing? " & CStr(pDataArray Is Nothing)
'  MsgBox pDataArray.Count & " items in array..."
  Dim frmGraph As Linkages.frmWidthGraph
61:   Set frmGraph = New Linkages.frmWidthGraph
'  MsgBox "Spatial Reference Name = " & pSpRef.Name
63:   Set frmGraph.ArcApplication = pApp
64:   Set frmGraph.SpatReference = pSpRef
'  frmGraph.GraphNumbers = dblVal
66:   Set frmGraph.GraphNumbers = pDataArray

68:   If ProgressForm.chkClose.Value = 1 Then
69:     Unload ProgressForm
70:     Set ProgressForm = Nothing
71:   End If

73:   frmGraph.Frame.Visible = True


  Exit Sub
ErrorHandler:
  HandleError True, "BottleneckAnalysis " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Function RunBottleneck(pStartPoly As IPolygon4, pEndPoly As IPolygon4, pCorPolygon As IPolygon4, _
    pMxDoc As IMxDocument, pApp As IApplication, frmProgress As Linkages.frmJenProgressPercent) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler

  Dim pSpRef As ISpatialReference
86:   Set pSpRef = pCorPolygon.SpatialReference
```

```
  Dim pTopoOp As ITopologicalOperator
89:   Set pTopoOp = pCorPolygon
  Dim pRelOp As IRelationalOperator
91:   Set pRelOp = pCorPolygon

  Dim strMessage As String
  Dim lngCounter As Long
  Dim lngCount As Long
96:   lngCounter = 0
97:   lngCount = 18

  Dim theDetailedDescription As String
100:   theDetailedDescription = frmProgress.txtDetails.Text

  ' #1
103:   lngCounter = lngCounter + 1
104:   strMessage = "Preparing habitat and corridor polygons (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt," ddddd") & "]" & vbCrLf & vbCrLf
107:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
110:   frmProgress.ProgRecCount = lngCount
111:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
112:   theDetailedDescription = theDetailedDescription & strMessage
113:   frmProgress.txtDetails.Text = theDetailedDescription
114:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
115:   frmProgress.Est_Time_Left lngCounter, "Preparing Data...", "Step "
116:   frmProgress.Refresh
117:   frmProgress.icnProgressLine.Refresh
118:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

  ' CREATE FINAL START POLYGON REPRESENTING ONLY OUTERMOST RINGS OF ALL SUB-PARTS OF START POLYGON THAT
  ' INTERSECT CORRIDOR POLYGON
  ' GET OUTERMOST COMPONENTS OF START POLYGON THAT INTERSECT CORRIDOR
  Dim pPolyOutStart() As IPolygon 'Declare an array of polygon
  Dim lngNumParts As Long
  Dim lngIndex As Long
127:   lngNumParts = pStartPoly.ExteriorRingCount
  ReDim pPolyOutStart(lngNumParts - 1) 'Redimension the array of polygon with number of exterior ring
129:   pStartPoly.GetConnectedComponents lngNumParts, pPolyOutStart(0) 'Pass the first element of the array

  ' GET OUTERMOST RING OF EACH POLYGON IN ARRAY
  Dim pClone As IClone
  Dim pTempStartA As IPolygon4
  Dim pRing() As IRing
```

```
  Dim pOuterRing As IRing
  Dim pPath As ISegmentCollection
  Dim pTopoForSimplify As ITopologicalOperator
  Dim pFinalStartPolygon As IPolygon4
139:   Set pFinalStartPolygon = New Polygon
140:   Set pFinalStartPolygon.SpatialReference = pSpRef
  Dim pOutputPolygon As IGeometryCollection
142:   Set pOutputPolygon = pFinalStartPolygon
  Dim pFinalStartLine As IPolyline
144:   Set pFinalStartLine = New Polyline
145:   Set pFinalStartLine.SpatialReference = pSpRef
  Dim pOutputPolyline As IGeometryCollection
147:   Set pOutputPolyline = pFinalStartLine
148:   For lngIndex = 0 To lngNumParts - 1
149:     Set pTempStartA = pPolyOutStart(lngIndex)
150:     Set pTopoForSimplify = pTempStartA
151:     pTopoForSimplify.Simplify
152:     If Not pRelOp.Disjoint(pTempStartA) Then  ' ONLY PICK SUB-POLYGONS THAT INTERSECT CORRIDOR
      ReDim pRing(0)
154:       pTempStartA.QueryExteriorRingsEx 1, pRing(0)   ' ONLY GET OUTERMOST RING OF POLYGON
155:       Set pOuterRing = pRing(0)
156:       Set pClone = pOuterRing
157:       pOutputPolygon.AddGeometry pOuterRing
158:       Set pPath = New Path
159:       pPath.AddSegmentCollection pClone.Clone
160:       pOutputPolyline.AddGeometry pPath
161:     End If
162:   Next
163:   Set pTopoForSimplify = pFinalStartPolygon
164:   pTopoForSimplify.Simplify
165:   Set pTopoForSimplify = pOutputPolyline
166:   pTopoForSimplify.Simplify

  Dim pClipStart As IPolygon
169:   Set pClipStart = pTopoOp.Intersect(pFinalStartPolygon, esriGeometry2Dimension)
  'Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pFinalStartPolygon, "delete_corridors"

  ' CREATE FINAL END POLYGON REPRESENTING ONLY OUTERMOST RINGS OF ALL SUB-PARTS OF END POLYGON THAT
  ' INTERSECT CORRIDOR POLYGON
  Dim pPolyOutEnd() As IPolygon 'Declare an array of polygon
175:   lngNumParts = pEndPoly.ExteriorRingCount
  ReDim pPolyOutEnd(lngNumParts - 1) 'Redimension the array of polygon with number of exterior ring
177:   pEndPoly.GetConnectedComponents lngNumParts, pPolyOutEnd(0) 'Pass the first element of the array

  ' GET OUTERMOST RING OF EACH POLYGON IN ARRAY
  Dim pTempEndA As IPolygon4
  Dim pFinalEndPolygon As IPolygon4
```

```
182:   Set pFinalEndPolygon = New Polygon
  Dim pOutputEndPolygon As IGeometryCollection
184:   Set pOutputEndPolygon = pFinalEndPolygon
  Dim pFinalEndPolyline As IPolyline
186:   Set pFinalEndPolyline = New Polyline
  Dim pOutputEndPolyline As IGeometryCollection
188:   Set pOutputEndPolyline = pFinalEndPolyline
189:   For lngIndex = 0 To lngNumParts - 1
190:     Set pTempEndA = pPolyOutEnd(lngIndex)
191:     Set pTopoForSimplify = pTempEndA
192:     pTopoForSimplify.Simplify
193:     If Not pRelOp.Disjoint(pTempEndA) Then  ' ONLY PICK SUB-POLYGONS THAT INTERSECT CORRIDOR
      ReDim pRing(0)
195:       pTempEndA.QueryExteriorRingsEx 1, pRing(0)   ' ONLY GET OUTERMOST RING OF POLYGON
196:       Set pOuterRing = pRing(0)
197:       Set pClone = pOuterRing
198:       pOutputEndPolygon.AddGeometry pOuterRing
199:       Set pPath = New Path
200:       pPath.AddSegmentCollection pClone.Clone
201:       pOutputEndPolyline.AddGeometry pPath
202:     End If
203:   Next
204:   Set pTopoForSimplify = pFinalEndPolygon
205:   pTopoForSimplify.Simplify
206:   Set pTopoForSimplify = pOutputEndPolyline
207:   pTopoForSimplify.Simplify

  Dim pClipEnd As IPolygon
210:   Set pClipEnd = pTopoOp.Intersect(pEndPoly, esriGeometry2Dimension)


  ' GET SET OF POLYLINES, REPRESENTING PORTION OF FINAL START/END POLYGON EDGES THAT LAY WITHIN CORRIDOR
  Dim pStartPolyline As IPolyline
215:   Set pStartPolyline = pTopoOp.Intersect(pOutputPolyline, esriGeometry1Dimension)
  Dim pEndPolyline As IPolyline
217:   Set pEndPolyline = pTopoOp.Intersect(pOutputEndPolyline, esriGeometry1Dimension)


'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pClipStart, "delete_corridors"
'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pClipEnd, "delete_corridors"
'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pStartPolyline, "delete_corridors"
'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pEndPolyline, "delete_corridors"

'  ' MAKE DISTANCE GRID
'  ' FIRST MAKE POLYON, BASED ON BOUNDARY EXTENT WITH BOUNDARY CLIPPED OUT

  ' COMBINE CORRIDOR WITH START AND END HABITAT BLOCK SHELLS (NO INTERIOR HOLES, ONLY PORTIONS THAT INTERSECT CORRIDOR
```

```
  Dim pUnionPoly As IPolygon4
230:   Set pTopoOp = pCorPolygon
231:   Set pUnionPoly = pTopoOp.Union(pFinalEndPolygon)
232:   Set pTopoOp = pUnionPoly
  Dim pUnionPoly2 As IPolygon4
234:   Set pUnionPoly2 = pTopoOp.Union(pFinalStartPolygon)
235:   Set pTopoOp = pUnionPoly2

  ' CREATE AOI POLYGON
  Dim pAOIPoly As IPolygon4
239:   Set pAOIPoly = New Polygon
240:   Set pAOIPoly.SpatialReference = pSpRef
  Dim pIntermediatePoly As IPointCollection
242:   Set pIntermediatePoly = pAOIPoly

  ' CREATE AOI ENVELOPE
  Dim pAOIEnv As IEnvelope
246:   Set pAOIEnv = pCorPolygon.Envelope
247:   pAOIEnv.Expand 1.1, 1.1, True

  ' CONVERT AOI ENVELOPE TO AOI POLYGON
  Dim dXmin As Double
  Dim dYmin As Double
  Dim dXmax As Double
  Dim dYmax As Double
254:   pAOIEnv.QueryCoords dXmin, dYmin, dXmax, dYmax
  Dim pPoint1 As IPoint
  Dim pPoint2 As IPoint
  Dim pPoint3 As IPoint
  Dim pPoint4 As IPoint
259:   Set pPoint1 = New Point
260:   Set pPoint2 = New Point
261:   Set pPoint3 = New Point
262:   Set pPoint4 = New Point
263:   pPoint1.PutCoords dXmin, dYmin
264:   pPoint2.PutCoords dXmin, dYmax
265:   pPoint3.PutCoords dXmax, dYmax
266:   pPoint4.PutCoords dXmax, dYmin
267:   pIntermediatePoly.AddPoint pPoint1
268:   pIntermediatePoly.AddPoint pPoint2
269:   pIntermediatePoly.AddPoint pPoint3
270:   pIntermediatePoly.AddPoint pPoint4
271:   pIntermediatePoly.AddPoint pPoint1
272:   Set pTopoForSimplify = pAOIPoly
273:   pTopoForSimplify.Simplify

  ' CLIP OUT CORRIDOR UNION FROM AOI POLYGON
```

```
276:   Set pTopoOp = pAOIPoly
  Dim pNonHabitatPolygon As IPolygon4
278:   Set pNonHabitatPolygon = pTopoOp.Difference(pUnionPoly2)
279:   Set pTopoForSimplify = pNonHabitatPolygon
280:   pTopoForSimplify.Simplify
  Dim pHabitatPolygon As IPolygon4
282:   Set pHabitatPolygon = pTopoOp.Intersect(pUnionPoly2, esriGeometry2Dimension)
283:   Set pTopoForSimplify = pHabitatPolygon
284:   pTopoForSimplify.Simplify

'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pHabitatPolygon, "delete_corridors"

  Dim dblMaxDimension As Double
  Dim dblCellSize As Double

291:   dblMaxDimension = pAOIEnv.Width
292:   If pAOIEnv.Height > pAOIEnv.Width Then dblMaxDimension = pAOIEnv.Height

294:   dblCellSize = dblMaxDimension / 1000
  ' FOR DEBUGGING
'  dblCellSize = dblMaxDimension / 70

  ' #2
299:   lngCounter = lngCounter + 1
300:   strMessage = "Preparing Raster Analysis Environment (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
303:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
306:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
307:   theDetailedDescription = theDetailedDescription & strMessage
308:   frmProgress.txtDetails.Text = theDetailedDescription
309:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
310:   frmProgress.Est_Time_Left lngCounter, "Preparing Analysis Environment...", "Step "
311:   frmProgress.Refresh
312:   frmProgress.icnProgressLine.Refresh
313:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------


  ' MAKE CORRIDOR RASTER
  Dim pCorRaster As IRaster
  Dim pEnv As IRasterAnalysisEnvironment  'Create a RasterMakerOp operator
  Dim pRasMakerOp As IRasterMakerOp
321:   Set pRasMakerOp = New RasterMakerOp
```

```
  'Create an analysis environment object
'  Dim pEnv As IRasterAnalysisEnvironment
325:   Set pEnv = pRasMakerOp

  'Set cellsize
328:   pEnv.SetCellSize esriRasterEnvValue, dblCellSize

  'Set output extent
331:   pEnv.SetExtent esriRasterEnvValue, pAOIEnv

  'Set output spatial reference
334:   Set pEnv.OutSpatialReference = pSpRef

  'Create a constant raster
  Dim pBaseRaster As IRaster
338:   Set pBaseRaster = pRasMakerOp.MakeConstant(1, True)

'  GridFunctions.ReturnCellSize pBaseRaster

  ' CLIP TO CORRIDOR POLYGON
343:   Set pCorRaster = Linkages.GridFunctions.ClipRasterToPolygon(pBaseRaster, pHabitatPolygon, True, pAOIEnv, dblCellSize)

  ' FOR DEBUGGING
'  Dim pRasterLayer As IRasterLayer
'  Set pRasterLayer = New RasterLayer
'  pRasterLayer.CreateFromRaster pCorRaster
'  pDoc.FocusMap.AddLayer pRasterLayer

  ' MAKE RASTER OPERATIONS OBJECTS
  Dim pLogOp As ILogicalOp         ' FOR CHECKING IF NULL
  Dim pDistanceOp As IDistanceOp   ' FOR CALCULATING EUCLIDIAN DISTANCE
  Dim pMathOp As IMathOp           ' FOR MULTIPLYING
  Dim pCondOp As IConditionalOp    ' FOR SETTING TO NULL

357:   Set pLogOp = New RasterMathOps
358:   Set pDistanceOp = New RasterDistanceOp
359:   Set pMathOp = New RasterMathOps
360:   Set pCondOp = New RasterConditionalOp

  ' #3
363:   lngCounter = lngCounter + 1
364:   strMessage = "Generating grid of corridor region (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt, ddddd") & "]" & vbCrLf & vbCrLf
367:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
```

```
370:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
371:   theDetailedDescription = theDetailedDescription & strMessage
372:   frmProgress.txtDetails.Text = theDetailedDescription
373:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
374:   frmProgress.Est_Time_Left lngCounter, "Generating Corridor Grid...", "Step "
375:   frmProgress.Refresh
376:   frmProgress.icnProgressLine.Refresh
377:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

  ' MAKE RASTER OF NULL PARTS OF pCorRaster
  Dim pOutsideRasterA As IGeoDataset
382:   Set pOutsideRasterA = pLogOp.IsNull(pCorRaster)
  Dim pOutsideRasterB As IGeoDataset
384:   Set pOutsideRasterB = pCondOp.SetNull(pLogOp.NotEqual(pOutsideRasterA, pBaseRaster), pOutsideRasterA)

  ' FOR DEBUGGING
'  Dim pOutsideRasterLayer As IRasterLayer
'  Set pOutsideRasterLayer = New RasterLayer
'  pOutsideRasterLayer.CreateFromRaster pOutsideRasterB
'  pDoc.FocusMap.AddLayer pOutsideRasterLayer

  ' #4
393:   lngCounter = lngCounter + 1
394:   strMessage = "Calculating distance to corridor edge (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf

398:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
401:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
402:   theDetailedDescription = theDetailedDescription & strMessage
403:   frmProgress.txtDetails.Text = theDetailedDescription
404:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
405:   frmProgress.Est_Time_Left lngCounter, "Generating Distance Grid...", "Step "
406:   frmProgress.Refresh
407:   frmProgress.icnProgressLine.Refresh
408:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

  Dim pOutputRaster As IGeoDataset
412:   Set pOutputRaster = pDistanceOp.EucDistance(pOutsideRasterB)

  ' FOR DEBUGGING
'  Dim pDistRasterLayerA As IRasterLayer
'  Set pDistRasterLayerA = New RasterLayer
```

```
'  pDistRasterLayerA.CreateFromRaster pOutputRaster
'  pDoc.FocusMap.AddLayer pDistRasterLayerA

  ' #5
421:   lngCounter = lngCounter + 1
422:   strMessage = "Combining distance and corridor grids (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
425:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
428:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
429:   theDetailedDescription = theDetailedDescription & strMessage
430:   frmProgress.txtDetails.Text = theDetailedDescription
431:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
432:   frmProgress.Est_Time_Left lngCounter, "Combining Grids...", "Step "
433:   frmProgress.Refresh
434:   frmProgress.icnProgressLine.Refresh
435:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

  Dim pDistanceRaster As IGeoDataset
439:   Set pDistanceRaster = pMathOp.Times(pCorRaster, pOutputRaster)

  ' FOR DEBUGGING
'  Dim pDistRasterLayer As IRasterLayer
'  Set pDistRasterLayer = New RasterLayer
'  pDistRasterLayer.CreateFromRaster pDistanceRaster
'  pDoc.FocusMap.AddLayer pDistRasterLayer

  ' MAKE DIRECTION AND FLOW ACCUMULATION GRID FROM DISTANCE GRID
  Dim pDirGrid As IGeoDataset
  Dim pFlowAccGrid As IGeoDataset
  Dim pHydOp As IHydrologyOp
451:   Set pHydOp = New RasterHydrologyOp

  ' #6
454:   lngCounter = lngCounter + 1
455:   strMessage = "Generating Flow Direction grid (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
458:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
461:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
462:   theDetailedDescription = theDetailedDescription & strMessage
463:   frmProgress.txtDetails.Text = theDetailedDescription
```

```
464:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
465:   frmProgress.Est_Time_Left lngCounter, "Generating Flow Direction Grid...", "Step "
466:   frmProgress.Refresh
467:   frmProgress.icnProgressLine.Refresh
468:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

471:   Set pDirGrid = pHydOp.FlowDirection(pDistanceRaster, False, False)

  ' #7
474:   lngCounter = lngCounter + 1
475:   strMessage = "Generating Flow Accumulation grid (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
478:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
481:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
482:   theDetailedDescription = theDetailedDescription & strMessage
483:   frmProgress.txtDetails.Text = theDetailedDescription
484:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
485:   frmProgress.Est_Time_Left lngCounter, "Generating Flow Accumulation Grid...", "Step "
486:   frmProgress.Refresh
487:   frmProgress.icnProgressLine.Refresh
488:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

491:   Set pFlowAccGrid = pHydOp.FlowAccumulation(pDirGrid)

  ' GET CELLS WITH 0 FLOW ACCUMULATION
  ' #8
495:   lngCounter = lngCounter + 1
496:   strMessage = "Identifying centerline cells in corridor (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
499:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
502:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
503:   theDetailedDescription = theDetailedDescription & strMessage
504:   frmProgress.txtDetails.Text = theDetailedDescription
505:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
506:   frmProgress.Est_Time_Left lngCounter, "Identifying Central Cells...", "Step "
507:   frmProgress.Refresh
508:   frmProgress.icnProgressLine.Refresh
509:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------
```

```
  Dim pZeroGrid As IGeoDataset
513:   Set pZeroGrid = pLogOp.EqualTo(pFlowAccGrid, pRasMakerOp.MakeConstant(0, True))

  ' #9
516:   lngCounter = lngCounter + 1
517:   strMessage = "Clipping centerline cells (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
520:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
523:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
524:   theDetailedDescription = theDetailedDescription & strMessage
525:   frmProgress.txtDetails.Text = theDetailedDescription
526:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
527:   frmProgress.Est_Time_Left lngCounter, "Clipping Central Cells...", "Step "
528:   frmProgress.Refresh
529:   frmProgress.icnProgressLine.Refresh
530:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

  Dim pPossibleGrid As IGeoDataset
534:   Set pPossibleGrid = pCondOp.SetNull(pLogOp.NotEqual(pZeroGrid, pBaseRaster), pZeroGrid)

  ' FOR DEBUGGING
'  Dim pPossibleGridLayer As IRasterLayer
'  Set pPossibleGridLayer = New RasterLayer
'  pPossibleGridLayer.CreateFromRaster pPossibleGrid
'  pDoc.FocusMap.AddLayer pPossibleGridLayer

  ' MAKE START AND END GRIDS
  Dim pStartGrid As IGeoDataset
  Dim pEndGrid As IGeoDataset
  Dim pExtractionOp As IExtractionOp2

  ' #10
548:   lngCounter = lngCounter + 1
549:   strMessage = "Clipping Habitat Block #1 to analysis area (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
552:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
555:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
556:   theDetailedDescription = theDetailedDescription & strMessage
557:   frmProgress.txtDetails.Text = theDetailedDescription
```

```
558:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
559:   frmProgress.Est_Time_Left lngCounter, "Clipping Habitat Block 1...", "Step "
560:   frmProgress.Refresh
561:   frmProgress.icnProgressLine.Refresh
562:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

565:   Set pStartGrid = Linkages.GridFunctions.ClipRasterToPolygon(pBaseRaster, pFinalStartPolygon, True, pAOIEnv, dblCellSize)

  ' #11
568:   lngCounter = lngCounter + 1
569:   strMessage = "Clipping Habitat Block #2 to analysis area (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
572:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
575:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
576:   theDetailedDescription = theDetailedDescription & strMessage
577:   frmProgress.txtDetails.Text = theDetailedDescription
578:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
579:   frmProgress.Est_Time_Left lngCounter, "Clipping Habitat Block 2...", "Step "
580:   frmProgress.Refresh
581:   frmProgress.icnProgressLine.Refresh
582:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

585:   Set pEndGrid = Linkages.GridFunctions.ClipRasterToPolygon(pBaseRaster, pFinalEndPolygon, True, pAOIEnv, dblCellSize)

'  Set pStartGrid = pExtractionOp.Polygon(pBaseRaster, pFinalStartPolygon, True)
'  Set pEndGrid = pExtractionOp.Polygon(pBaseRaster, pFinalEndPolygon, True)

  ' FOR DEBUGGING
'  Dim pStartGridLayer As IRasterLayer
'  Set pStartGridLayer = New RasterLayer
'  pStartGridLayer.CreateFromRaster pStartGrid
'  pDoc.FocusMap.AddLayer pStartGridLayer

  ' FOR DEBUGGING
'  Dim pEndGridLayer As IRasterLayer
'  Set pEndGridLayer = New RasterLayer
'  pEndGridLayer.CreateFromRaster pEndGrid
'  pDoc.FocusMap.AddLayer pEndGridLayer

  Dim pRasterStats As IRasterStatistics
  Dim pRasterBandCol As IRasterBandCollection
604:   Set pRasterBandCol = pDistanceRaster
```

```
  Dim pRasterBand As IRasterBand
606:   Set pRasterBand = pRasterBandCol.Item(0)
607:   Set pRasterStats = pRasterBand.Statistics
  Dim dblMaxDist As Double
609:   dblMaxDist = pRasterStats.Maximum

'  Debug.Print "dblMaxDist = " & CStr(dblMaxDist)

  ' #12
614:   lngCounter = lngCounter + 1
615:   strMessage = "Generating Burn Grid of central cells (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt, ddddd") & "]" & vbCrLf & vbCrLf
618:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
621:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
622:   theDetailedDescription = theDetailedDescription & strMessage
623:   frmProgress.txtDetails.Text = theDetailedDescription
624:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
625:   frmProgress.Est_Time_Left lngCounter, "Generating Burn Grid...", "Step "
626:   frmProgress.Refresh
627:   frmProgress.icnProgressLine.Refresh
628:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------


  Dim pBurnGrid As IGeoDataset
633:   Set pBurnGrid = pCondOp.Con(pLogOp.EqualTo(pZeroGrid, pBaseRaster), _
        pZeroGrid, pRasMakerOp.MakeConstant(1000000000.001, False))

  ' #13
637:   lngCounter = lngCounter + 1
638:   strMessage = "Generating Resistance Grid (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt, ddddd") & "]" & vbCrLf & vbCrLf
641:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
644:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
645:   theDetailedDescription = theDetailedDescription & strMessage
646:   frmProgress.txtDetails.Text = theDetailedDescription
647:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
648:   frmProgress.Est_Time_Left lngCounter, "Generating Resistance Grid..", "Step "
649:   frmProgress.Refresh
650:   frmProgress.icnProgressLine.Refresh
651:   DoEvents
```

```
  ' ---- END PROGRESS METER STUFF ---------------

  Dim pResistanceGrid As IGeoDataset
  Dim pMathSupOp As IMathSupportOp
656:   Set pMathSupOp = New RasterMathSupportOp
  Dim pMaxDistGrid As IGeoDataset
658:   Set pMaxDistGrid = pRasMakerOp.MakeConstant(dblMaxDist, False)
659:   Set pResistanceGrid = pMathSupOp.Plus(pMathSupOp.Times( _
        pMathSupOp.Divide(pMathSupOp.Minus(pMaxDistGrid, pDistanceRaster), pMaxDistGrid), _
        pRasMakerOp.MakeConstant(100000, False)), pBurnGrid)

  ' FOR DEBUGGING
'  Dim pResistanceLayer As IRasterLayer
'  Set pResistanceLayer = New RasterLayer
'  pResistanceLayer.CreateFromRaster pResistanceGrid
'  pDoc.FocusMap.AddLayer pResistanceLayer


'  theLengthFactor = theOriginPoly.Distance(theEndPoly)
'
'  ' MAXIMUM GRID VALUE RANGE = 1.175494351e-38 to 3.402823466e+38
'  theMaxDist = theDistGrid.GetStatistics.Get(1)
'  theBurnGrid = (theZeroGrid=1).Con(theZeroGrid, (1000000000.001).AsGrid)
'  theResistanceGrid = ((((theMaxDist.AsGrid) - theDistGrid) / (theMaxDist.AsGrid)) * 1000) + theBurnGrid
'  theDirFN = FileName.GetCWD.MakeTmp("dir", "")
'  'theAllFN = Filename.GetCWD.MakeTmp("all", "")
'  theCostGrid = theStartGrid.CostDistance(theResistanceGrid, theDirFN, nil, nil)
'  theDirGrid = Grid.Make(Grid.MakeSrcName(theDirFN.GetFullName))
'  'theAllGrid = Grid.Make(Grid.MakeSrcName(theAllFN.GetFullName))
'
  ' MAKE FORWARD AND REVERSE COST AND DIRECTION GRIDS
  Dim pForwardBandCol As IRasterBandCollection
  Dim pRevBandCol As IRasterBandCollection
  Dim pFullForward As IGeoDataset
  Dim pFullBack As IGeoDataset

  ' #14
689:   lngCounter = lngCounter + 1
690:   strMessage = "Calc Cost Distance from Habitat Block #1 (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
693:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
696:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
697:   theDetailedDescription = theDetailedDescription & strMessage
698:   frmProgress.txtDetails.Text = theDetailedDescription
```

```
699:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
700:   frmProgress.Est_Time_Left lngCounter, "Generating Cost Distance Grid #1...", "Step "
701:   frmProgress.Refresh
702:   frmProgress.icnProgressLine.Refresh
703:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

706:   Set pFullForward = pDistanceOp.CostDistanceFull(pStartGrid, pResistanceGrid, True, True, False)

  ' #15
709:   lngCounter = lngCounter + 1
710:   strMessage = "Calc Cost Distance from Habitat Block #2 (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
713:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
716:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
717:   theDetailedDescription = theDetailedDescription & strMessage
718:   frmProgress.txtDetails.Text = theDetailedDescription
719:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
720:   frmProgress.Est_Time_Left lngCounter, "Generating Cost Distance Grid #2...", "Step "
721:   frmProgress.Refresh
722:   frmProgress.icnProgressLine.Refresh
723:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

726:   Set pFullBack = pDistanceOp.CostDistanceFull(pEndGrid, pResistanceGrid, True, True, False)

728:   Set pForwardBandCol = pFullForward
729:   Set pRevBandCol = pFullBack

  Dim pForCostBand As IRasterBand
  Dim pForDirBand As IRasterBand
  Dim pBackCostBand As IRasterBand
  Dim pBackDirBand As IRasterBand

  Dim pForCostGrid As IRasterBandCollection
  Dim pForDirGrid As IRasterBandCollection
  Dim pBackCostGrid As IRasterBandCollection
  Dim pBackDirGrid As IRasterBandCollection

741:   Set pForCostBand = pForwardBandCol.Item(0)
742:   Set pForDirBand = pForwardBandCol.Item(1)
743:   Set pBackCostBand = pRevBandCol.Item(0)
744:   Set pBackDirBand = pRevBandCol.Item(1)
```

```
746:   Set pForCostGrid = New Raster
747:   pForCostGrid.Add pForCostBand, 0
748:   Set pForDirGrid = New Raster
749:   pForDirGrid.Add pForDirBand, 0
750:   Set pBackCostGrid = New Raster
751:   pBackCostGrid.Add pBackCostBand, 0
752:   Set pBackDirGrid = New Raster
753:   pBackDirGrid.Add pBackDirBand, 0

  ' FOR DEBUGGING
'  Dim pForCostLayer As IRasterLayer
'  Set pForCostLayer = New RasterLayer
'  pForCostLayer.CreateFromRaster pForCostGrid
'  pForCostLayer.Name = "pForCostGrid"
'  pDoc.FocusMap.AddLayer pForCostLayer
  ' FOR DEBUGGING
'  Dim pForDirLayer As IRasterLayer
'  Set pForDirLayer = New RasterLayer
'  pForDirLayer.CreateFromRaster pForDirGrid
'  pForDirLayer.Name = "pForDirGrid"
'  pDoc.FocusMap.AddLayer pForDirLayer
  ' FOR DEBUGGING
'  Dim pBackCostLayer As IRasterLayer
'  Set pBackCostLayer = New RasterLayer
'  pBackCostLayer.CreateFromRaster pBackCostGrid
'  pBackCostLayer.Name = "pBackCostGrid"
'  pDoc.FocusMap.AddLayer pBackCostLayer
  ' FOR DEBUGGING
'  Dim pBackDirLayer As IRasterLayer
'  Set pBackDirLayer = New RasterLayer
'  pBackDirLayer.CreateFromRaster pBackDirGrid
'  pBackDirLayer.Name = "pBackDirGrid"
'  pDoc.FocusMap.AddLayer pBackDirLayer

  ' #16
781:   lngCounter = lngCounter + 1
782:   strMessage = "Examining potential routes from HB #1 (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf
785:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
788:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
789:   theDetailedDescription = theDetailedDescription & strMessage
790:   frmProgress.txtDetails.Text = theDetailedDescription
791:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
792:   frmProgress.Est_Time_Left lngCounter, "Examining Potential Routes: #1...", "Step "
```

```
793:   frmProgress.Refresh
794:   frmProgress.icnProgressLine.Refresh
795:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------


  Dim pMultPt As IPointCollection
800:   Set pMultPt = New Multipoint
  Dim dblIndex As Double
  Dim pStepPoint As IPoint
803:   Set pStepPoint = New Point
804:   For dblIndex = 0 To pStartPolyline.length Step dblCellSize
805:     pStartPolyline.QueryPoint esriNoExtension, dblIndex, False, pStepPoint
806:     Set pClone = pStepPoint
807:     pMultPt.AddPoint pClone.Clone
808:   Next dblIndex
809:   Set pClone = pStartPolyline.ToPoint
810:   pMultPt.AddPoint pClone.Clone

  Dim pStartPtCollection As IPointCollection
813:   Set pStartPtCollection = New Multipoint

  ' MAKE ARRAY OF CELL VALUES AT POINTS
  Dim pValArray As esriSystem.IVariantArray
  Dim vVal As Variant

819:   Set pValArray = Linkages.GridFunctions.CellValues(pMultPt, pZeroGrid)
820:   For lngIndex = 0 To pValArray.Count - 1
821:     vVal = pValArray.Element(lngIndex)
822:     If Not IsNull(vVal) Then
823:       If vVal = 1 Then
'        Set pTestPoint = pMultPt.Point(lngIndex)
825:         pStartPtCollection.AddPoint pMultPt.Point(lngIndex)
'        Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pMultPt.Point(lngIndex), "delete_corridors"
827:       End If
828:     End If
829:   Next lngIndex

831:   If pStartPtCollection.PointCount = 0 Then Set pStartPtCollection = pMultPt
  Dim pStartPolylines As IGeometryCollection
833:   Set pStartPolylines = pDistanceOp.CostPathAsPolyline(pStartPtCollection, pBackCostGrid, pBackDirGrid)

  ' START EXAMINING ACTUAL LINES
  ' LOOKING FOR BEST LINE BASED ON 4 CRITERIA:
  ' 1) MOST IMPORTANT:  SELECT LINE WITH HIGHEST MINIMUM DISTANCE VALUE
  ' 2) IF > 1, SUB-SELECT LINE WITH LOWEST CUMULATIVE COST
  ' 3) IF STILL > 1, SUB-SELECT LINE WITH HIGHEST MEAN DISTANCE VALUE
```

```
  ' 4) IF STILL > 1, SUB-SELECT SHORTEST LINE
  ' 5) IF STILL > 1, THEN ALMOST CERTAINLY SAME LINE.  JUST TAKE THE FIRST ONE.

  Dim pFinalLineArray As esriSystem.IArray
844:   Set pFinalLineArray = New esriSystem.Array
  Dim pLngKeys As esriSystem.ILongArray
846:   Set pLngKeys = New esriSystem.LongArray

  Dim pLineDataArray As esriSystem.IVariantArray
  Dim pTestPolyline As IPolyline
  Dim pTestMultipoint As IPointCollection
  Dim dblLength As Double
  Dim pTestStatsArray As esriSystem.IVariantArray
  Dim lngIndex2 As Long
  Dim vValue As Variant
  Dim dblValue As Double
  Dim dblMin As Double
  Dim dblMean As Double
  Dim vCost As Variant
  Dim dblCost As Double
  Dim booFoundValue As Boolean
  Dim pFinalDataArray As esriSystem.IVariantArray
  Dim pPointDblArray As esriSystem.IDoubleArray
  Dim lngMeanCounter As Long

865:   If pStartPolylines.GeometryCount > 0 Then
866:     For lngIndex = 0 To pStartPolylines.GeometryCount - 1

868:       strMessage = "  --> From HB #1:  Calculating Stats for Route #" & CStr(lngIndex + 1) _
       & " of " & CStr(pStartPolylines.GeometryCount) & "..." & vbCrLf
870:       Debug.Print strMessage

      ' ---- PROGRESS METER STUFF
873:       frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
874:       theDetailedDescription = theDetailedDescription & strMessage
875:       frmProgress.txtDetails.Text = theDetailedDescription
876:       frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
877:       frmProgress.Est_Time_Left lngCounter, "Examining Potential Routes: #1...", "Step "
878:       frmProgress.Refresh
879:       frmProgress.icnProgressLine.Refresh
880:       DoEvents
      ' ---- END PROGRESS METER STUFF ---------------

883:       Set pTestPolyline = pStartPolylines.Geometry(lngIndex)
884:       Set pTopoOp = pTestPolyline
885:       pTopoOp.Simplify
```

```
887:       dblLength = pTestPolyline.length

889:       If dblLength > 0 Then
890:         Set pTestMultipoint = New Multipoint
891:         Set pFinalDataArray = New esriSystem.VarArray
892:         lngMeanCounter = 0

       ' GENERATE POINTS ALONG ROUTE
895:         For dblIndex = 0 To dblLength Step dblCellSize
896:           pTestPolyline.QueryPoint esriNoExtension, dblIndex, False, pStepPoint
897:           Set pClone = pStepPoint
898:           pTestMultipoint.AddPoint pClone.Clone
899:         Next dblIndex
900:         Set pStepPoint = pTestPolyline.ToPoint
901:         Set pClone = pStepPoint
902:         pTestMultipoint.AddPoint pClone.Clone

       ' CALCULATE MAXIMUM COST OF ROUTE
905:         vCost = Linkages.GridFunctions.CellValue(pStepPoint, pForCostGrid)
906:         If IsNull(vCost) Then
907:           dblCost = 2 ^ 30
908:         Else
909:           dblCost = CDbl(vCost)
910:         End If

       ' GET CELL VALUES OF ROUTE
913:         Set pTestStatsArray = Linkages.GridFunctions.CellValues(pTestMultipoint, pDistanceRaster)
914:         booFoundValue = False

       ' CALCULATE STATISTICS OF CELL VALUES
917:         dblMean = 0
918:         For lngIndex2 = 0 To pTestStatsArray.Count - 1
919:           vValue = pTestStatsArray.Element(lngIndex2)
920:           If Not IsNull(vValue) Then
921:             dblValue = CDbl(vValue) * 2    ' (Width = distance to edge x 2)
922:             If Not booFoundValue Then
923:               dblMin = dblValue
924:             Else
925:               If dblValue < dblMin Then dblMin = dblValue
926:             End If
927:             dblMean = dblMean + dblValue
928:             booFoundValue = True
929:             lngMeanCounter = lngMeanCounter + 1
930:           Else
931:             dblValue = -99
932:           End If
```

```
          ' MAKE FINAL OUTPUT ARRAY, IN CASE THIS IS THE BEST LINE
935:            Set pPointDblArray = New esriSystem.DoubleArray
936:            Set pStepPoint = pTestMultipoint.Point(lngIndex2)
937:            pPointDblArray.Add pStepPoint.X
938:            pPointDblArray.Add pStepPoint.Y
939:            pPointDblArray.Add dblValue
940:            If lngIndex2 = pTestStatsArray.Count - 1 Then
941:              pPointDblArray.Add pTestPolyline.length
942:            Else
943:              pPointDblArray.Add lngIndex2 * dblCellSize
944:            End If
945:            pFinalDataArray.Add pPointDblArray
946:         Next lngIndex2

948:         If booFoundValue Then
          ' ADD DATA TO OVERALL ARRAY, TO ANALYZE LATER
950:            Set pLineDataArray = New esriSystem.VarArray

952:            dblMean = dblMean / lngMeanCounter
953:            pLineDataArray.Add pTestPolyline
954:            pLineDataArray.Add dblMin
955:            pLineDataArray.Add dblMean
956:            pLineDataArray.Add dblCost
957:            pLineDataArray.Add pFinalDataArray
958:            pLngKeys.Add lngIndex
959:            pFinalLineArray.Add pLineDataArray
960:         End If
961:       End If
962:     Next lngIndex
963:   End If

'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pStartPolylines.Geometry(3), "delete_corridors"

  ' #17
968:   lngCounter = lngCounter + 1
969:   strMessage = vbCrLf & "Examining potential routes from HB #2 (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf
972:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
975:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
976:   theDetailedDescription = theDetailedDescription & strMessage
977:   frmProgress.txtDetails.Text = theDetailedDescription
978:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
979:   frmProgress.Est_Time_Left lngCounter, "Examining Potential Routes: #2...", "Step "
980:   frmProgress.Refresh
```

```
981:   frmProgress.icnProgressLine.Refresh
982:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------

  Dim pMultPt2 As IPointCollection
986:   Set pMultPt2 = New Multipoint
  Dim pEndPtColl As IPointCollection
988:   Set pEndPtColl = New Multipoint
989:   Set pStepPoint = New Point
990:   For dblIndex = 0 To pEndPolyline.length Step dblCellSize
991:     pEndPolyline.QueryPoint esriNoExtension, dblIndex, False, pStepPoint
992:     Set pClone = pStepPoint
993:     pMultPt2.AddPoint pClone.Clone
994:   Next dblIndex
995:   Set pClone = pEndPolyline.ToPoint
996:   pMultPt2.AddPoint pClone.Clone

998:   Set pValArray = Linkages.GridFunctions.CellValues(pMultPt2, pZeroGrid)
999:   For lngIndex = 0 To pValArray.Count - 1
1000:     vVal = pValArray.Element(lngIndex)
1001:     If Not IsNull(vVal) Then
1002:       If vVal = 1 Then
1003:         pEndPtColl.AddPoint pMultPt2.Point(lngIndex)
'        Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pMultPt2.Point(lngIndex), "delete_corridors"
1005:       End If
1006:     End If
1007:   Next lngIndex

1009:   If pEndPtColl.PointCount = 0 Then Set pEndPtColl = pMultPt2
  Dim pEndPolylines As IGeometryCollection
1011:   Set pEndPolylines = pDistanceOp.CostPathAsPolyline(pEndPtColl, pForCostGrid, pForDirGrid)

  ' REMEMBER TO FLIP THESE LINES!

1015:   If pEndPolylines.GeometryCount > 0 Then
1016:     For lngIndex = 0 To pEndPolylines.GeometryCount - 1

1018:       strMessage = "  --> From HB #2:  Calculating Stats for Route #" & CStr(lngIndex + 1) _
       & " of " & CStr(pEndPolylines.GeometryCount) & "..." & vbCrLf
1020:       Debug.Print strMessage

      ' ---- PROGRESS METER STUFF
1023:       frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
1024:       theDetailedDescription = theDetailedDescription & strMessage
1025:       frmProgress.txtDetails.Text = theDetailedDescription
1026:       frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
1027:       frmProgress.Est_Time_Left lngCounter, "Examining Potential Routes: #2...", "Step "
```

```
1028:       frmProgress.Refresh
1029:       frmProgress.icnProgressLine.Refresh
1030:       DoEvents
      ' ---- END PROGRESS METER STUFF ---------------

1033:       Set pTestPolyline = pEndPolylines.Geometry(lngIndex)
1034:       Set pTopoOp = pTestPolyline
1035:       pTopoOp.Simplify
1036:       pTestPolyline.ReverseOrientation

1038:       dblLength = pTestPolyline.length

1040:       If dblLength > 0 Then
1041:         Set pTestMultipoint = New Multipoint
1042:         Set pFinalDataArray = New esriSystem.VarArray
1043:         lngMeanCounter = 0

        ' GENERATE POINTS ALONG ROUTE
1046:         For dblIndex = 0 To dblLength Step dblCellSize
1047:           pTestPolyline.QueryPoint esriNoExtension, dblIndex, False, pStepPoint
1048:           Set pClone = pStepPoint
1049:           pTestMultipoint.AddPoint pClone.Clone
1050:         Next dblIndex
1051:         Set pStepPoint = pTestPolyline.ToPoint
1052:         Set pClone = pStepPoint
1053:         pTestMultipoint.AddPoint pClone.Clone

        ' CALCULATE MAXIMUM COST OF ROUTE
1056:         vCost = Linkages.GridFunctions.CellValue(pTestPolyline.FromPoint, pBackCostGrid)
1057:         If IsNull(vCost) Then
1058:           dblCost = 2 ^ 30
1059:         Else
1060:           dblCost = CDbl(vCost)
1061:         End If

        ' GET CELL VALUES OF ROUTE
1064:         Set pTestStatsArray = Linkages.GridFunctions.CellValues(pTestMultipoint, pDistanceRaster)
1065:         booFoundValue = False

        ' CALCULATE STATISTICS OF CELL VALUES
1068:         dblMean = 0
1069:         For lngIndex2 = 0 To pTestStatsArray.Count - 1
1070:           vValue = pTestStatsArray.Element(lngIndex2)
1071:           If Not IsNull(vValue) Then
1072:             dblValue = CDbl(vValue) * 2    ' (Width = distance to edge x 2)
1073:             If Not booFoundValue Then
1074:               dblMin = dblValue
```

```
1075:             Else
1076:               If dblValue < dblMin Then dblMin = dblValue
1077:             End If
1078:             dblMean = dblMean + dblValue
1079:             booFoundValue = True
1080:             lngMeanCounter = lngMeanCounter + 1
1081:           Else
1082:             dblValue = -99
1083:           End If

         ' MAKE FINAL OUTPUT ARRAY, IN CASE THIS IS THE BEST LINE
1086:           Set pPointDblArray = New esriSystem.DoubleArray
1087:           Set pStepPoint = pTestMultipoint.Point(lngIndex2)
1088:           pPointDblArray.Add pStepPoint.X
1089:           pPointDblArray.Add pStepPoint.Y
1090:           pPointDblArray.Add dblValue
1091:           If lngIndex2 = pTestStatsArray.Count - 1 Then
1092:             pPointDblArray.Add pTestPolyline.length
1093:           Else
1094:             pPointDblArray.Add lngIndex2 * dblCellSize
1095:           End If
1096:           pFinalDataArray.Add pPointDblArray
1097:         Next lngIndex2

1099:         If booFoundValue Then
         ' ADD DATA TO OVERALL ARRAY, TO ANALYZE LATER
1101:           Set pLineDataArray = New esriSystem.VarArray

1103:           dblMean = dblMean / lngMeanCounter
1104:           pLineDataArray.Add pTestPolyline
1105:           pLineDataArray.Add dblMin
1106:           pLineDataArray.Add dblMean
1107:           pLineDataArray.Add dblCost
1108:           pLineDataArray.Add pFinalDataArray
1109:           pLngKeys.Add lngIndex
1110:           pFinalLineArray.Add pLineDataArray
1111:         End If
1112:       End If
1113:     Next lngIndex
1114:   End If

  ' BAIL OUT IF DIDN'T FIND ANY POSSIBLE LINES
1117:   If pFinalLineArray.Count = 0 Then
1118:     MsgBox "Unable to calculate connector line!  Reasons unknown." & vbCrLf & "Bailing out...", , "Operation Failed:"
    Exit Function
1120:   End If
```

```
  ' FIND BEST LINE
  Dim pEligibleIndices As esriSystem.ILongArray
1124:   Set pEligibleIndices = New esriSystem.LongArray
  Dim pTempIndices As esriSystem.ILongArray

  ' FIND LINES WITH LARGEST MINIMUM DISTANCES
  Dim dblMinDistValue As Double
1129:   For lngIndex = 0 To pFinalLineArray.Count - 1
1130:     Set pLineDataArray = pFinalLineArray.Element(lngIndex)
1131:     Set pTestPolyline = pLineDataArray.Element(0)
1132:     dblMin = pLineDataArray.Element(1)
1133:     If dblMin > dblMinDistValue Then
1134:       dblMinDistValue = dblMin
1135:       pEligibleIndices.RemoveAll
1136:       pEligibleIndices.Add lngIndex
1137:     ElseIf dblMinDistValue = dblMin Then
1138:       pEligibleIndices.Add lngIndex
1139:     End If
1140:   Next lngIndex

  ' IF > 1, SUB-SELECT LINES WITH LOWEST TOTAL COST
  Dim lngFinalIndex As Long
1144:   lngFinalIndex = -999
  Dim dblLowestCost As Double
1146:   If pEligibleIndices.Count > 1 Then
1147:     Set pTempIndices = New esriSystem.LongArray
1148:     For lngIndex = 0 To pEligibleIndices.Count - 1
1149:       lngIndex2 = pEligibleIndices.Element(lngIndex)
1150:       Set pLineDataArray = pFinalLineArray.Element(lngIndex2)
1151:       Set pTestPolyline = pLineDataArray.Element(0)
1152:       dblCost = pLineDataArray.Element(3)
1153:       If lngIndex = 0 Then ' IF THIS IS FIRST VALUE
1154:         dblLowestCost = dblCost
1155:         pTempIndices.RemoveAll
1156:         pTempIndices.Add lngIndex2
1157:       Else
1158:         If dblCost < dblLowestCost Then
1159:           dblLowestCost = dblCost
1160:           pTempIndices.RemoveAll
1161:           pTempIndices.Add lngIndex2
1162:         ElseIf dblCost = dblLowestCost Then
1163:           pTempIndices.Add lngIndex2
1164:         End If
1165:       End If
1166:     Next lngIndex

1168:     Set pEligibleIndices = pTempIndices
```

```
1169:   Else
1170:     lngFinalIndex = pEligibleIndices.Element(0)
1171:   End If

  ' IF > 1, SUB-SELECT LINES WITH HIGHEST MEAN DISTANCE
'  Dim dblHighestMeanDist As Double
'  If pEligibleIndices.Count > 1 Then
'    Set pTempIndices = New esriSystem.LongArray
'    For lngIndex = 0 To pEligibleIndices.Count - 1
'      lngIndex2 = pEligibleIndices.Element(lngIndex)
'      Set pLineDataArray = pFinalLineArray.Element(lngIndex2)
'      Set pTestPolyline = pLineDataArray.Element(0)
'      dblMean = pLineDataArray.Element(2)
'      If lngIndex = 0 Then ' IF THIS IS FIRST VALUE
'        dblHighestMeanDist = dblMean
'        pTempIndices.RemoveAll
'        pTempIndices.Add lngIndex2
'      Else
'        If dblMean > dblHighestMeanDist Then
'          dblHighestMeanDist = dblMean
'          pTempIndices.RemoveAll
'          pTempIndices.Add lngIndex2
'        ElseIf dblCost = dblLowestCost Then
'          pTempIndices.Add lngIndex2
'        End If
'      End If
'    Next lngIndex
'
'    Set pEligibleIndices = pTempIndices
'  Else
'    lngFinalIndex = pEligibleIndices.Element(0)
'  End If

  ' IF > 1, SUB-SELECT SHORTEST LINE
  Dim dblShortestLength As Double
1204:   If pEligibleIndices.Count > 1 Then
1205:     Set pTempIndices = New esriSystem.LongArray
1206:     For lngIndex = 0 To pEligibleIndices.Count - 1
1207:       lngIndex2 = pEligibleIndices.Element(lngIndex)
1208:       Set pLineDataArray = pFinalLineArray.Element(lngIndex2)
1209:       Set pTestPolyline = pLineDataArray.Element(0)
1210:       dblLength = pTestPolyline.length
1211:       If lngIndex = 0 Then ' IF THIS IS FIRST VALUE
1212:         dblShortestLength = dblLength
1213:         pTempIndices.RemoveAll
1214:         pTempIndices.Add lngIndex2
1215:       Else
```

```
1216:         If dblLength < dblShortestLength Then
1217:           dblShortestLength = dblLength
1218:           pTempIndices.RemoveAll
1219:           pTempIndices.Add lngIndex2
1220:         ElseIf dblCost = dblLowestCost Then
1221:           pTempIndices.Add lngIndex2
1222:         End If
1223:       End If
1224:     Next lngIndex

1226:     Set pEligibleIndices = pTempIndices
1227:   Else
1228:     lngFinalIndex = pEligibleIndices.Element(0)
1229:   End If

  ' IF > 1, JUST TAKE THE FIRST ONE
1232:   If lngFinalIndex < 0 Then lngFinalIndex = pEligibleIndices.Element(0)

  ' GET OUTPUT ARRAY
  Dim pFinalStatsArray As esriSystem.IVariantArray
1236:   Set pLineDataArray = pFinalLineArray.Element(lngFinalIndex)
1237:   Set pFinalStatsArray = pLineDataArray.Element(4)

'  Linkages.mygeneraloperations.Graphic_MakeFromGeometry Document, pLineDataArray.Element(0), "delete_corridors"

  ' #18
1242:   lngCounter = lngCounter + 1
1243:   strMessage = vbCrLf & "Clearing memory and temporary datasets (Step " & CStr(lngCounter) _
      & " of " & CStr(lngCount) & ")..." & vbCrLf & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
1246:   Debug.Print strMessage

  ' ---- PROGRESS METER STUFF
1249:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
1250:   theDetailedDescription = theDetailedDescription & strMessage
1251:   frmProgress.txtDetails.Text = theDetailedDescription
1252:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
1253:   frmProgress.Est_Time_Left lngCounter, "Clearing temporary datasets...", "Step "
1254:   frmProgress.Refresh
1255:   frmProgress.icnProgressLine.Refresh
1256:   DoEvents
  ' ---- END PROGRESS METER STUFF ---------------


1260:   Debug.Print "Finished"

1262:   Set RunBottleneck = pFinalStatsArray
```

```
  Exit Function
ErrorHandler:
  HandleError True, "RunBottleneck " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function MakeHistogramData(pFieldInfoArray As esriSystem.IStringArray, lngNumBins As Long, _
    dblMinimum As Double, dblMaximum As Double, lngHistArray() As Long, pApp As IApplication, strFolder As String, _
    Optional lngNumDecimalPlaces As Long = -999) As String
  On Error GoTo ErrorHandler

  Dim strDecFormatString As String
1276:   strDecFormatString = "0"
1277:   If lngNumDecimalPlaces > 0 Then strDecFormatString = strDecFormatString + "." + String(lngNumDecimalPlaces, "0")

  Dim strFieldName As String
  Dim strfilename As String
1281:   strFieldName = pFieldInfoArray.Element(0)
1282:   If Right(strFolder, 1) <> "/" And Right(strFolder, 1) <> "\" Then strFolder = strFolder & "\"
1283:   strfilename = strFolder & strFieldName & "_histogram.dbf"
1284:   strfilename = Linkages.aml_func_mod.MakeUniqueFilename(strfilename)

  Dim strReport As String
  Dim dblRunningBinVal As Double
  Dim dblBinInterval As Double
  Dim lngMaxBinCount As Long
  Dim dblBinRatio As Double
  Dim lngBinCount As Long

1293:   dblRunningBinVal = dblMinimum
1294:   dblBinInterval = (dblMaximum - dblMinimum) / lngNumBins
  ' FIRST GET MAXIMUM BIN COUNT, THEN SCALE IT TO 40 CHARACTERS
1296:   lngMaxBinCount = 0
1297:   dblBinRatio = 1
  Dim anIndex As Long
1299:   For anIndex = 1 To lngNumBins
1300:     lngBinCount = lngHistArray(anIndex - 1)
1301:     If lngBinCount > lngMaxBinCount Then lngMaxBinCount = lngBinCount
1302:   Next anIndex
1303:   If lngMaxBinCount > 110 Then
1304:     dblBinRatio = 110 / lngMaxBinCount
1305:   End If
  Dim strSub1 As String
  Dim strSub2 As String
```

```
  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit

1314:   Set pNewFields = New Fields
1315:   Set pNewFieldsEdit = pNewFields
  ' 1) Unique_ID
  ' 2) Bin_Start
  ' 3) Bin_End
  ' 4) Bin_Count
1320:   pNewFieldsEdit.FieldCount = 4

1322:   Set pNewField = New Field
1323:   Set pNewFieldEdit = pNewField
1324:   With pNewFieldEdit
1325:     .Name = "Bin_ID"
1326:     .Precision = 8
1327:     .Type = esriFieldTypeInteger
1328:   End With
1329:   Set pNewFieldsEdit.Field(0) = pNewField

1331:   Set pNewField = New Field
1332:   Set pNewFieldEdit = pNewField
1333:   With pNewFieldEdit
1334:     .Name = "Bin_Start"
1335:     .Precision = 16
1336:     .Scale = 8
1337:     .Type = esriFieldTypeDouble
1338:   End With
1339:   Set pNewFieldsEdit.Field(1) = pNewField

1341:   Set pNewField = New Field
1342:   Set pNewFieldEdit = pNewField
1343:   With pNewFieldEdit
1344:     .Name = "Bin_End"
1345:     .Precision = 16
1346:     .Scale = 8
1347:     .Type = esriFieldTypeDouble
1348:   End With
1349:   Set pNewFieldsEdit.Field(2) = pNewField

1351:   Set pNewField = New Field
1352:   Set pNewFieldEdit = pNewField
1353:   With pNewFieldEdit
1354:     .Name = "Bin_Count"
1355:     .Precision = 16
```

```
1356:     .Type = esriFieldTypeInteger
1357:   End With
1358:   Set pNewFieldsEdit.Field(3) = pNewField

  Dim pTable As ITable
1361:   Set pTable = Linkages.aml_func_mod.CreatedBASETable(strfilename, pNewFields)

  Dim pRow As IRow

  ' MAKE REPORT AND FILL TABLE
1366:   strReport = "\b        --- " & strFieldName & " Histogram ---------------------------\b0\par" & vbCrLf
1367:   For anIndex = 1 To lngNumBins

1369:     lngBinCount = lngHistArray(anIndex - 1)
1370:     If lngNumDecimalPlaces < 0 Then
1371:       strSub1 = strSub1 & "  Bin " & CStr(anIndex) & "]  Range = " & CStr(dblRunningBinVal) _
          & " to " & CStr(dblRunningBinVal + dblBinInterval) & "     Bin Size = " & _
          Linkages.aml_func_mod.InsertCommas(lngBinCount) & " cases...\par" & vbCrLf
1374:     Else
1375:       strSub1 = strSub1 & "  Bin " & CStr(anIndex) & "]  Range = " & CStr(Format(dblRunningBinVal, strDecFormatString)) _
          & " to " & CStr(Format((dblRunningBinVal + dblBinInterval), strDecFormatString)) & "     Bin Size = " & _
          Linkages.aml_func_mod.InsertCommas(lngBinCount) & " cases...\par" & vbCrLf
1378:     End If
1379:     strSub2 = strSub2 & "  " & CStr(anIndex) & "]\tab " & String(Round(lngBinCount * dblBinRatio), _
        "|") & "\par" & vbCrLf
1381:     If lngNumDecimalPlaces < 0 Then
1382:       strReport = strReport & "   Bin " & CStr(anIndex) & "]  Range = " & CStr(dblRunningBinVal) _
          & " to " & CStr(dblRunningBinVal + dblBinInterval) & "     Bin Size = " & _
          Linkages.aml_func_mod.InsertCommas(lngBinCount) & " cases...\par" & vbCrLf & "        " & _
          String(Round(lngBinCount * dblBinRatio), "|") & "\par" & vbCrLf
1386:     Else
1387:       strReport = strReport & "   Bin " & CStr(anIndex) & "]  Range = " & CStr(Format(dblRunningBinVal, strDecFormatString)) _
          & " to " & CStr(Format((dblRunningBinVal + dblBinInterval), strDecFormatString)) & "     Bin Size = " & _
          Linkages.aml_func_mod.InsertCommas(lngBinCount) & " cases...\par" & vbCrLf & "        " & _
          String(Round(lngBinCount * dblBinRatio), "|") & "\par" & vbCrLf
1391:     End If
1392:     Set pRow = pTable.CreateRow
1393:     pRow.Value(pTable.FindField("Bin_ID")) = anIndex
1394:     pRow.Value(pTable.FindField("Bin_Start")) = dblRunningBinVal
1395:     pRow.Value(pTable.FindField("Bin_End")) = dblRunningBinVal + dblBinInterval
1396:     pRow.Value(pTable.FindField("Bin_Count")) = lngBinCount
1397:     pRow.Store
1398:     dblRunningBinVal = dblRunningBinVal + dblBinInterval
1399:   Next anIndex

  ' MAKE TABLE WINDOW AND ADD IT TO DOCUMENT
  Dim pNewStandaloneTable As IStandaloneTable
```

```
1403:   Set pNewStandaloneTable = New StandaloneTable
1404:   Set pNewStandaloneTable.Table = pTable

  Dim pTableWindow2 As ITableWindow2
1407:   Set pTableWindow2 = New TableWindow

  Dim lngLeft As Long
  Dim lngTop As Long
  Dim lngRight As Long
  Dim lngBottom As Long

1414:   With pTableWindow2
1415:     Set .StandaloneTable = pNewStandaloneTable
1416:     Set .Application = pApp
1417:     .TableSelectionAction = esriSelectFeatures
1418:     .ShowAliasNamesInColumnHeadings = True
1419:     .ShowSelected = False
1420:     .Show True
1421:   End With

  Dim pMxDoc As IMxDocument
1424:   Set pMxDoc = pApp.Document
  Dim pStandaloneTableCollection As IStandaloneTableCollection
1426:   Set pStandaloneTableCollection = pMxDoc.FocusMap
1427:   pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable

1429:   pMxDoc.UpdateContents

'  MakeHistogramData = "        [Histogram data table saved to " & strfilename & "]" & vbCrLf & _
'      strReport & "        --- " & strFieldName & " Histogram Reorganized ----------" & vbCrLf & strSub1 & strSub2
1433:   MakeHistogramData = "        --- [Histogram data table saved to " & _
      Linkages.aml_func_mod.SubstituteString(strfilename, "\", "\\") & "]\par" & vbCrLf & _
      "\b        --- " & strFieldName & " Histogram ----------\b0\par" & vbCrLf & strSub1 & strSub2


  Exit Function
ErrorHandler:
  HandleError True, "MakeHistogramData " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function GridHistogram(pRasterLayer As IRasterLayer, dblLowVal As Double, dblHighVal As Double, _
    lngNumBins As Long, pApp As IApplication) As Long()
  On Error GoTo ErrorHandler


  Dim pRaster As IRaster
```

```
1449:   Set pRaster = pRasterLayer.Raster

  Dim lngHistCount() As Long
  ReDim lngHistCount(lngNumBins - 1)

  Dim dblInterval As Double
1455:   dblInterval = (dblHighVal - dblLowVal) / lngNumBins

  'Get rasters
  Dim pRasOut As IRaster

  Dim pMapAlgebraOp As IMapAlgebraOp
1461:   Set pMapAlgebraOp = New RasterMapAlgebraOp
1462:   pMapAlgebraOp.BindRaster pRaster, "pRaster"

  Dim lngIndex As Long
  Dim dblLow As Double
  Dim dblHigh As Double

1468:   dblLow = dblLowVal
1469:   dblHigh = dblLowVal + dblInterval

  Dim pRasterBandCollection As IRasterBandCollection
  Dim pRasterBand As IRasterBand
  Dim pTable As ITable
  Dim lngTotalCount As Long
  Dim lngValField As Long
  Dim lngCountField As Long
  Dim pCursor As ICursor
  Dim pRow As IRow
  Dim lngVal As Long
  Dim lngCount As Long

1482:   lngTotalCount = 0

    ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
1486:   Set psbar = pApp.StatusBar
  Dim pPro As IStepProgressor
1488:   Set pPro = psbar.ProgressBar
1489:   pPro.position = 1
1490:   Screen.MousePointer = vbHourglass
1491:   psbar.ShowProgressBar "Generating Histogram for [" & pRasterLayer.Name & "]", 1, _
          lngNumBins, 1, True

1494:   For lngIndex = 0 To lngNumBins - 1
1495:     If lngIndex = lngNumBins - 1 Then
```

```
1496:       Set pRasOut = pMapAlgebraOp.Execute("([pRaster] >= " & CStr(dblLow) & ") AND ([pRaster] <= " & CStr(dblHigh) & ")")
1497:     Else
1498:       Set pRasOut = pMapAlgebraOp.Execute("([pRaster] >= " & CStr(dblLow) & ") AND ([pRaster] < " & CStr(dblHigh) & ")")
1499:     End If

1501:     Set pRasterBandCollection = pRasOut
1502:     Set pRasterBand = pRasterBandCollection.Item(0)
1503:     Set pTable = pRasterBand.AttributeTable

1505:     lngValField = pTable.FindField("Value")
1506:     lngCountField = pTable.FindField("Count")

1508:     Set pCursor = pTable.Search(Nothing, True)
1509:     Set pRow = pCursor.NextRow
1510:     lngCount = 0

1512:     Do Until pRow Is Nothing
1513:       lngVal = pRow.Value(lngValField)
1514:       If lngVal = 1 Then                ' MEANING THE EXPRESSION ABOVE EVALUATES TO "TRUE"
1515:         lngCount = pRow.Value(lngCountField)
1516:       End If
1517:       Set pRow = pCursor.NextRow
1518:     Loop

1520:     lngHistCount(lngIndex) = lngCount
1521:     lngTotalCount = lngTotalCount + lngCount

1523:     dblLow = dblLow + dblInterval
1524:     dblHigh = dblLow + dblInterval
1525:     psbar.StepProgressBar
1526:   Next lngIndex

1528:   Screen.MousePointer = vbDefault
1529:   psbar.HideProgressBar
1530:   GridHistogram = lngHistCount


  Exit Function
ErrorHandler:
  HandleError True, "GridHistogram " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function StatsForDates(ByVal pDateArray As esriSystem.IVariantArray) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler
```

```
  Dim pReturn As esriSystem.IVariantArray
1543:   Set pReturn = New esriSystem.VarArray

  Dim dateArray() As Date
  ReDim dateArray(pDateArray.Count - 1)
  Dim anIndex As Long
1548:   For anIndex = 0 To pDateArray.Count - 1
1549:     dateArray(anIndex) = pDateArray.Element(anIndex)
1550:   Next anIndex

1552:   Call Linkages.QuickSort.DatesAscending(dateArray, 0, UBound(dateArray))

  Dim dateStart As Date
  Dim dateEnd As Date
1556:   dateStart = dateArray(0)
1557:   dateEnd = dateArray(UBound(dateArray))

1559:   pReturn.Add dateStart
1560:   pReturn.Add dateEnd

1562:   Set StatsForDates = pReturn


  Exit Function
ErrorHandler:
  HandleError True, "StatsForDates " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function StatsForStrings(ByVal pStringArray As esriSystem.IStringArray, _
      Optional ByVal pValArray As esriSystem.IDoubleArray) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler


  ' RETURN VARIANT ARRAY WILL BE FILLED WITH SUB VARIANT ARRAYS.
  ' EACH SUB-VARIANT ARRAY WILL REFLECT ONE UNIQUE STRING VALUE FROM INPUT pStringArray
  ' EACH SUB-VARIANT ARRAY WILL CONTAIN [STRING VALUE, COUNT, PROPORTION, SIZE]
  ' IMPORTANT:  IF OPTIONAL WEIGHTING ARRAY IS INCLUDED, THEN PROPORTION WILL BE BASED ON SIZE-WEIGHTED COUNTS.  OTHERWISE
  '             PROPORTION IS BASED ONLY ON COUNTS.
  ' IMPORTANT:  IF DRAWN FROM POLYLINE OR POLYGON FEATURE CLASS, THEN CHOOSING PROPORTION FORCES SIZE-WEIGHTED ANALYSIS.

  Dim lngIndex As Long
  Dim strVal As String
  Dim dblSize As Double
  Dim strUniqueStrings() As String
  ReDim strUniqueStrings(pStringArray.Count)
```

```
  Dim lngUniqueCounter As Long
1589:   lngUniqueCounter = -1
  Dim booHasKey As Boolean

  Dim booUseSize As Boolean
1593:   booUseSize = (Not pValArray Is Nothing)

  Dim lngCount As Long
  Dim dblProportion As Double
  Dim dblTotalSize As Double
1598:   dblTotalSize = 0
  Dim lngTotalCount As Long
1600:   lngTotalCount = 0
  Dim dblRunningSize As Double
1602:   dblRunningSize = 0
  Dim lngRunningCount As Long
1604:   lngRunningCount = 0
  Dim dblTotalUniqueSize As Double
1606:   dblTotalUniqueSize = 0
  Dim lngTotalUniqueCount As Long
1608:   lngTotalUniqueCount = 0

  Dim pReturnArray As esriSystem.IVariantArray
1611:   Set pReturnArray = New esriSystem.VarArray
  Dim pSubArray As esriSystem.IVariantArray

  Dim pCollection As New Collection
  Dim pCountCollection As New Collection

  ' IF pStringArray IS EMPTY, THEN RETURN EMPTY pReturnArray

1619:   If pStringArray.Count = 1 Then
1620:     Set pSubArray = New esriSystem.VarArray
1621:     pSubArray.Add pStringArray.Element(0)
1622:     lngCount = 1
1623:     pSubArray.Add lngCount
1624:     dblProportion = 1
1625:     pSubArray.Add dblProportion
1626:     If booUseSize Then
1627:       dblTotalSize = pValArray.Element(0)
1628:     Else
1629:       dblTotalSize = 1
1630:     End If
1631:     pSubArray.Add dblTotalSize
1632:     pReturnArray.Add pSubArray
1633:   ElseIf pStringArray.Count > 1 Then
1634:     For lngIndex = 0 To pStringArray.Count - 1
```

```
1635:       strVal = pStringArray.Element(lngIndex)
1636:       If booUseSize Then
1637:         dblSize = pValArray.Element(lngIndex)
1638:       Else
1639:         dblSize = 1
1640:       End If

1642:       booHasKey = CheckCollectionForKey(pCollection, strVal)
1643:       If booHasKey Then
        ' UPDATE SIZE COLLECTION
1645:         dblRunningSize = pCollection.Item(strVal)
1646:         dblRunningSize = dblRunningSize + dblSize
1647:         pCollection.Remove strVal
1648:         pCollection.Add dblRunningSize, strVal
        ' UPDATE COUNT COLLECTION
1650:         lngRunningCount = pCountCollection.Item(strVal)
1651:         lngRunningCount = lngRunningCount + 1
1652:         pCountCollection.Remove strVal
1653:         pCountCollection.Add lngRunningCount, strVal
1654:       Else
1655:         pCollection.Add dblSize, strVal
1656:         pCountCollection.Add 1, strVal
1657:         lngUniqueCounter = lngUniqueCounter + 1
1658:         strUniqueStrings(lngUniqueCounter) = strVal
1659:       End If
1660:     Next lngIndex
    ReDim Preserve strUniqueStrings(lngUniqueCounter)

1663:     dblTotalSize = 0
1664:     For lngIndex = 0 To lngUniqueCounter
1665:       strVal = strUniqueStrings(lngIndex)
1666:       dblTotalSize = dblTotalSize + pCollection(strVal)
1667:     Next lngIndex

1669:     Call Linkages.QuickSort.StringsAscending(strUniqueStrings, 0, lngUniqueCounter)
1670:     For lngIndex = 0 To lngUniqueCounter
1671:       Set pSubArray = New esriSystem.VarArray

1673:       strVal = strUniqueStrings(lngIndex)
1674:       dblTotalUniqueSize = pCollection.Item(strVal)
1675:       lngTotalUniqueCount = pCountCollection.Item(strVal)
1676:       dblProportion = dblTotalUniqueSize / dblTotalSize

1678:       pSubArray.Add strVal
1679:       pSubArray.Add lngTotalUniqueCount
1680:       pSubArray.Add dblProportion
1681:       pSubArray.Add dblTotalUniqueSize
```

```
1682:       pReturnArray.Add pSubArray
1683:     Next lngIndex
1684:   End If

1686:   Set StatsForStrings = pReturnArray


  Exit Function
ErrorHandler:
  HandleError True, "StatsForStrings " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function StatsPropsForNumbers(ByVal pNumberArray As esriSystem.IDoubleArray, _
      Optional ByVal pValArray As esriSystem.IDoubleArray) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler


  ' RETURN VARIANT ARRAY WILL BE FILLED WITH SUB VARIANT ARRAYS.
  ' EACH SUB-VARIANT ARRAY WILL REFLECT ONE UNIQUE VALUE FROM INPUT pNumberArray
  ' EACH SUB-VARIANT ARRAY WILL CONTAIN [VALUE, COUNT, PROPORTION, SIZE]
  ' IMPORTANT:  IF OPTIONAL WEIGHTING ARRAY IS INCLUDED, THEN PROPORTION WILL BE BASED ON SIZE-WEIGHTED COUNTS.  OTHERWISE
  '             PROPORTION IS BASED ONLY ON COUNTS.
  ' IMPORTANT:  IF DRAWN FROM POLYLINE OR POLYGON FEATURE CLASS, THEN CHOOSING PROPORTION FORCES SIZE-WEIGHTED ANALYSIS.

  Dim lngIndex As Long
  Dim dblVal As Double
  Dim dblSize As Double
  Dim dblUniqueNumbers() As Double
  ReDim dblUniqueNumbers(pNumberArray.Count)
  Dim lngUniqueCounter As Long
1712:   lngUniqueCounter = -1
  Dim booHasKey As Boolean

  Dim booUseSize As Boolean
1716:   booUseSize = (Not pValArray Is Nothing)

  Dim lngCount As Long
  Dim dblProportion As Double
  Dim dblTotalSize As Double
1721:   dblTotalSize = 0
  Dim lngTotalCount As Long
1723:   lngTotalCount = 0
  Dim dblRunningSize As Double
1725:   dblRunningSize = 0
  Dim lngRunningCount As Long
1727:   lngRunningCount = 0
```

```
  Dim dblTotalUniqueSize As Double
1729:   dblTotalUniqueSize = 0
  Dim lngTotalUniqueCount As Long
1731:   lngTotalUniqueCount = 0

  Dim strVal As String

  Dim pReturnArray As esriSystem.IVariantArray
1736:   Set pReturnArray = New esriSystem.VarArray
  Dim pSubArray As esriSystem.IVariantArray

  Dim pCollection As New Collection
  Dim pCountCollection As New Collection

 ' MsgBox "CorridorAnalysisFunctions.StatsPropsForNumbers" & vbCrLf & "booUseSize = " & CStr(booUseSize)

  ' IF pDoubleArray IS EMPTY, THEN RETURN EMPTY pReturnArray

1746:   If pNumberArray.Count = 1 Then
1747:     Set pSubArray = New esriSystem.VarArray
1748:     pSubArray.Add pNumberArray.Element(0)
1749:     lngCount = 1
1750:     pSubArray.Add lngCount
1751:     dblProportion = 1
1752:     pSubArray.Add dblProportion
1753:     If booUseSize Then
1754:       dblTotalSize = pValArray.Element(0)
1755:     Else
1756:       dblTotalSize = 1
1757:     End If
1758:     pSubArray.Add dblTotalSize
1759:     pReturnArray.Add pSubArray
1760:   ElseIf pNumberArray.Count > 1 Then
1761:     For lngIndex = 0 To pNumberArray.Count - 1
1762:       dblVal = pNumberArray.Element(lngIndex)
1763:       If booUseSize Then
1764:         dblSize = pValArray.Element(lngIndex)
1765:       Else
1766:         dblSize = 1
1767:       End If

1769:       strVal = CStr(dblVal)

1771:       booHasKey = CheckCollectionForKey(pCollection, strVal)
1772:       If booHasKey Then
        ' UPDATE SIZE COLLECTION
1774:         dblRunningSize = pCollection.Item(strVal)
```

```
1775:         dblRunningSize = dblRunningSize + dblVal
1776:         pCollection.Remove strVal
1777:         pCollection.Add dblRunningSize, strVal
        ' UPDATE COUNT COLLECTION
1779:         lngRunningCount = pCountCollection.Item(strVal)
1780:         lngRunningCount = lngRunningCount + dblSize
1781:         pCountCollection.Remove strVal
1782:         pCountCollection.Add lngRunningCount, strVal
1783:       Else
1784:         pCollection.Add dblSize, strVal
1785:         pCountCollection.Add dblSize, strVal
1786:         lngUniqueCounter = lngUniqueCounter + 1
1787:         dblUniqueNumbers(lngUniqueCounter) = dblVal
1788:       End If
1789:     Next lngIndex
    ReDim Preserve dblUniqueNumbers(lngUniqueCounter)

1792:     dblTotalSize = 0
1793:     For lngIndex = 0 To lngUniqueCounter
1794:       dblVal = dblUniqueNumbers(lngIndex)
1795:       dblTotalSize = dblTotalSize + pCollection(CStr(dblVal))
1796:     Next lngIndex

1798:     Call Linkages.QuickSort.DoubleAscending(dblUniqueNumbers, 0, lngUniqueCounter)
1799:     For lngIndex = 0 To lngUniqueCounter
1800:       Set pSubArray = New esriSystem.VarArray

1802:       dblVal = dblUniqueNumbers(lngIndex)
1803:       dblTotalUniqueSize = pCollection.Item(CStr(dblVal))
1804:       lngTotalUniqueCount = pCountCollection.Item(CStr(dblVal))
1805:       dblProportion = dblTotalUniqueSize / dblTotalSize

1807:       pSubArray.Add dblVal
1808:       pSubArray.Add lngTotalUniqueCount
1809:       pSubArray.Add dblProportion
1810:       pSubArray.Add dblTotalUniqueSize
1811:       pReturnArray.Add pSubArray
1812:     Next lngIndex
1813:   End If

1815:   Set StatsPropsForNumbers = pReturnArray


  Exit Function
ErrorHandler:
  HandleError True, "StatsPropsForNumbers " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Function
Public Function CheckCollectionForKey(colCollection As Collection, strKey As String) As Boolean
  On Error GoTo ErrorHandler

1825:   CheckCollectionForKey = True
  Dim varTest As Variant
1827:   varTest = colCollection.Item(strKey)

  Exit Function
ErrorHandler:
1831:   CheckCollectionForKey = False

End Function

Public Function ApplyCorridorSymbol(pPolygonElement As IElement) As IFillShapeElement
  On Error GoTo ErrorHandler


  Dim pFillShapeElement As IFillShapeElement
1840:   Set pFillShapeElement = pPolygonElement

  Dim pColor As IColor
1843:   Set pColor = Linkages.MyGeneralOperations.MakeColorRGB(0, 200, 100)

  Dim pCartoLine As ICartographicLineSymbol
1846:   Set pCartoLine = New CartographicLineSymbol
1847:   With pCartoLine
1848:     .Cap = esriLCSButt
1849:     .Join = esriLJSBevel
1850:     .Color = pColor
1851:     .Width = 1
1852:   End With

  Dim pLineFill As ILineFillSymbol
1855:   Set pLineFill = New LineFillSymbol
1856:   With pLineFill
1857:     .Angle = -30
1858:     .Separation = 3
1859:     .Offset = 5
1860:   End With
1861:   Set pLineFill.LineSymbol = pCartoLine

  Dim pLineSymbol As ISimpleLineSymbol
1864:   Set pLineSymbol = New SimpleLineSymbol

  Dim pLineColor As IColor
1867:   Set pLineColor = Linkages.MyGeneralOperations.MakeColorRGB(0, 250, 100)
```

```
1868:   pLineSymbol.Color = pLineColor
1869:   pLineSymbol.Width = 2
1870:   pLineSymbol.Style = esriSLSSolid
1871:   pLineFill.Outline = pLineSymbol
1872:   pFillShapeElement.Symbol = pLineFill

1874:   Set ApplyCorridorSymbol = pFillShapeElement

  Exit Function
ErrorHandler:
  HandleError True, "ApplyCorridorSymbol " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Sub ImplementKruskall(pMxDoc As IMxDocument, pStartPolygon As IPolygon, pEndPolygon As IPolygon, _
    pStartArray As esriSystem.IArray, pCorPolygon As IPolygon, ByRef frmProgressDialog As Object, _
    pExtensionConfig As IExtensionConfig, pParamDetails As esriSystem.IVariantArray)
  On Error GoTo ErrorHandler

  Dim pApp As IApplication
  Dim pDoc As IDocument
1889:   Set pDoc = pMxDoc
1890:   Set pApp = pDoc.Parent

'  Screen.MousePointer = vbHourglass

  ' pStartArray IS ARRAY OF ALL PATCHES THAT INTERSECT CORRIDOR; UNCLIPPED

  Dim frmProgress As Linkages.frmJenProgressPercent
1897:   Set frmProgress = frmProgressDialog

  ' ---- PROGRESS METER STUFF
  Dim theProgressID As String
  Dim theProgressSpecific As String
  Dim theDetailedDescription As String
1903:   theProgressSpecific = "Preliminary analysis..."
1904:   theDetailedDescription = frmProgress.txtDetails.Text
1905:   theDetailedDescription = theDetailedDescription & _
    " --> " & theProgressSpecific & vbCrLf & _
    "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
1908:   frmProgress.txtDetails.Text = theDetailedDescription
1909:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)

1911:   frmProgress.Est_Time_Left 0, "Clipping to Corridor...", "Clipping to Corridor..."
1912:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
1913:   frmProgress.Refresh
```

```
1914:   frmProgress.icnProgressLine.Refresh
1915:   DoEvents
1916:   If BailOutOfProgress(frmProgress, pExtensionConfig) Then
    Exit Sub
1918:   End If
  ' ---- END PROGRESS METER STUFF --------------------------------

  Dim varStartTime As Variant
1922:   varStartTime = Timer

  ' START AND END POLYGONS ARE INSERTED IN 'OK' FUNCTION OF FORM
'  pStartArray.Insert 0, pStartPolygon
'  pStartArray.Add pEndPolygon

  Dim pClone As IClone

  ' CLIP PATCH POLYGONS TO CORRIDOR; MIGHT END UP WITH MORE PATCH POLYGONS THAN BEFORE IF SOME PATCHES ARE SPLIT
  ' ALSO JOIN ADJACENT POLYGONS
  Dim pClipCollection As Collection
1933:   Set pClipCollection = ClipPolysToCorridor(pStartArray, pCorPolygon, pApp)
1934:   If Not pClipCollection.Item(1) Then          ' HERE IT RETURNS AN ERROR OF SOMETHING WAS NOT VALID
    Dim strResponse As String
1936:     strResponse = pClipCollection.Item(2)
1937:     Debug.Print strResponse
    Exit Sub
1939:   End If

  Dim pArray As IArray
1942:   Set pArray = pClipCollection.Item(5)

  Dim strStartNode As String
1945:       strStartNode = "0"
  Dim strEndNode As String
1947:       strEndNode = CStr(pArray.Count - 1)

  ' FOR DEBUGGING
'  Dim pCheckPoly As IPolygon
'  Dim pCheckIndex As Long
'  For pCheckIndex = 0 To pArray.Count - 1 '
'    Set pCheckPoly = pArray.Element(pCheckIndex)
'    MyGeneralOperations.Graphic_MakeFromGeometry pMxDoc, pCheckPoly, "test_order"
'  Next pCheckIndex

  Dim lngArrayMaxIndex As Long
1958:   lngArrayMaxIndex = pArray.Count - 1

  Dim pColArray As IVariantArray
```

```
1961:       Set pColArray = New VarArray
  Dim pPolyline As IPolyline
  Dim lngIndex As Long
  Dim lngIndex2 As Long
  Dim strID As String
  Dim dblDist As Double
  Dim dblBear As Double

  Dim pShape1 As IPolygon
  Dim pShape2 As IPolygon
  Dim lngCheckIndex As Long

  Dim booShouldReject As Boolean
1974:   DoEvents

    ' ---- PROGRESS METER STUFF
1977:   theProgressSpecific = "Calculating Distance Matrix..."
1978:   theDetailedDescription = theDetailedDescription & _
    " --> " & theProgressSpecific & vbCrLf & _
    "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
1981:   frmProgress.txtDetails.Text = theDetailedDescription
1982:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)

1984:   frmProgress.Est_Time_Left 0, "Preliminary Analysis...", "Calculating Distance Matrix..."
1985:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
1986:   frmProgress.Refresh
1987:   frmProgress.icnProgressLine.Refresh
1988:   If BailOutOfProgress(frmProgress, pExtensionConfig) Then
    Exit Sub
1990:   End If
  ' ---- END PROGRESS METER STUFF --------------------------------


  Dim pCollection As Collection
1995:       Set pCollection = MyGeometricOperations.CalcDistMatrix(pArray, True, False, pApp)
'  Debug.Print pArray.Count
'  Debug.Print "Collection Count = " & pCollection.Count

  ' FOR DEBUGGING BELOW
    ' REMEMBER THAT STARTING SHAPE CAN ONLY GO OUT, AND ENDING SHAPE CAN ONLY COME IN
'  For lngIndex = 0 To lngArrayMaxIndex          ' CAN'T GO BACKWARDS FROM ENDING POLYGON
'    For lngIndex2 = 0 To lngArrayMaxIndex         ' CAN'T GO BACKWARDS TO STARTING POLYGON
'      strID = lngIndex & "_" & lngIndex2
'      Set pColArray = pCollection.Item(strID)
'      Set pPolyline = pColArray.Element(3)
'      Debug.Print strID & ",    Distance = " & _
'        pColArray.Element(2) & ",     Polyline Distance = " & pPolyline.Length
```

```
'    Next lngIndex2
'  Next lngIndex

  Dim pFinalCollection As New Collection
  Dim pIDArray As IStringArray
2013:       Set pIDArray = New strArray
  Dim pDoneCollection As CollectionMod
2015:       Set pDoneCollection = New CollectionMod
  Dim pCounter As Long
2017:   pCounter = -1
  Dim pSubArray As IVariantArray

   ' FOR DEBUGGING
2021:   MyGeneralOperations.DeleteGraphicsByName pMxDoc, "test_matrix"

  ' REMEMBER THAT CORRIDOR MIGHT BE MULTI-STRAND.  THEREFORE ONLY CONSIDER PAIRS OF PATCH POLYGONS
  ' THAT ARE IN THE SAME STRAND.
  Dim pIntPolygon As IPolygon4
2026:       Set pIntPolygon = pCorPolygon
  Dim pRelOp As IRelationalOperator
  Dim pGeometryCollection As IGeometryCollection
2029:       Set pGeometryCollection = pIntPolygon.ConnectedComponentBag
  Dim aSubPolyIndex As Long
  Dim pCorStrandPolygon As IPolygon4
  Dim pSubStartPoly As IPolygon
  Dim pSubEndPoly As IPolygon
  Dim IsMultiStrand As Boolean
2035:       IsMultiStrand = pGeometryCollection.GeometryCount > 1
  Dim IsInSameStrand As Boolean

    ' ---- PROGRESS METER STUFF
2039:   theProgressSpecific = "Preliminary analysis of matrix..."
2040:   theDetailedDescription = theDetailedDescription & _
    " --> " & theProgressSpecific & vbCrLf & _
    "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
2043:   frmProgress.txtDetails.Text = theDetailedDescription
2044:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)

2046:   frmProgress.Est_Time_Left 0, "Preliminary Analysis...", "Preliminary analysis of matrix..."
2047:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2048:   If BailOutOfProgress(frmProgress, pExtensionConfig) Then
    Exit Sub
2050:   End If
2051:   frmProgress.Refresh
2052:   frmProgress.icnProgressLine.Refresh
2053:   DoEvents
  ' ---- END PROGRESS METER STUFF --------------------------------
```

```
  ' REMEMBER THAT STARTING SHAPE CAN ONLY GO OUT, AND ENDING SHAPE CAN ONLY COME IN
2057:   For lngIndex = 0 To (lngArrayMaxIndex - 1)      ' CAN'T GO BACKWARDS FROM ENDING POLYGON
2058:     For lngIndex2 = 1 To lngArrayMaxIndex         ' CAN'T GO BACKWARDS TO STARTING POLYGON
2059:       If lngIndex <> lngIndex2 Then               ' DON'T CONSIDER DISTANCE FROM A SHAPE TO ITSELF

        ' IF NECESSARY, CHECK TO MAKE SURE 2ND POLYGON INTERSECTS SAME CORRIDOR STRAND AS FIRST POLYGON
2062:         If IsMultiStrand Then
2063:           Set pSubStartPoly = pArray.Element(lngIndex)
2064:           Set pSubEndPoly = pArray.Element(lngIndex2)
2065:           For aSubPolyIndex = 0 To pGeometryCollection.GeometryCount - 1
2066:             Set pCorStrandPolygon = pGeometryCollection.Geometry(aSubPolyIndex)
2067:             Set pRelOp = pCorStrandPolygon
2068:             IsInSameStrand = (Not pRelOp.Disjoint(pSubStartPoly)) And (Not pRelOp.Disjoint(pSubEndPoly))
2069:             If IsInSameStrand Then Exit For
2070:           Next aSubPolyIndex
2071:         Else
2072:           IsInSameStrand = True
2073:         End If

2075:         If IsInSameStrand Then
2076:           strID = lngIndex & "_" & lngIndex2
2077:           Set pColArray = pCollection.Item(strID)
        '  pColArray.Add (lngIndex)
2079:           pFinalCollection.Add pColArray, strID
2080:           pIDArray.Add strID

2082:           pCounter = pCounter + 1
2083:           Set pSubArray = New VarArray
2084:           pSubArray.Add pCounter
2085:           pSubArray.Add False
2086:           pDoneCollection.AddObject pSubArray, strID, True
2087:         End If
2088:       End If
2089:     Next lngIndex2
2090:   Next lngIndex
  ' NOW HAVE A COLLECTION OF VARIANT ARRAYS, WHERE VARIANT ARRAY CONTAINS {POLYLINE, DISTANCE}
  ' ALSO HAVE AN ARRAY OF STRING ID VALUES FOR ALL OBJECTS IN COLLECTION

'  For lngIndex = 0 To pIDArray.Count - 1
'    Set pColArray = pFinalCollection.Item(pIDArray.Element(lngIndex))
'    Debug.Print pIDArray.Element(lngIndex) & ",    Distance = " & _
'        pColArray.Element(2)
'  Next lngIndex

  ' FIRST CHECK FOR LINES THAT CROSS EDGE; NEED TO GENERATE GRID PATH FOR THEM
  Dim lngFullIndex As Long
```

```
  Dim booHasBeenDone As Boolean
  Dim booCrossesEdge As Boolean
  Dim lngFinalIndex As Long
2105:   lngFinalIndex = -1
  Dim pGridLine As IPolyline
  Dim pGridPoly1 As IPolygon
  Dim pGridPoly2 As IPolygon
  Dim pCorRaster As IRaster
  Dim pEnv As IRasterAnalysisEnvironment
  Dim strRevID As String, pReverseLine As IPolyline, pNewColArray As IVariantArray
  Dim lngRevIndex As Long
  Dim booHasRevDir As Boolean

  Dim pRemoveKeyArray As IStringArray
2116:   Set pRemoveKeyArray = New strArray

'  Dim pArray2 As esriSystem.IArray2

    ' ---- PROGRESS METER STUFF
2121:   frmProgress.ProgRecCount = pIDArray.Count
2122:   theProgressSpecific = "Checking for Boundary Intersections..."
2123:   theDetailedDescription = theDetailedDescription & _
    " --> " & theProgressSpecific & vbCrLf & _
    "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
2126:   frmProgress.txtDetails.Text = theDetailedDescription
2127:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)

2129:   frmProgress.Est_Time_Left 0, "Boundary Intersections...", "Working on Analysis #"
2130:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2131:   frmProgress.Refresh
2132:   frmProgress.icnProgressLine.Refresh
2133:   DoEvents
2134:   If BailOutOfProgress(frmProgress, pExtensionConfig) Then
    Exit Sub
2136:   End If
  ' ---- END PROGRESS METER STUFF --------------------------------

  Dim psbar As IStatusBar
2140:   Set psbar = pApp.StatusBar
  Dim pPro As IStepProgressor
2142:   Set pPro = psbar.ProgressBar
  Dim lngCounter As Long
2144:   lngCounter = 0
  Dim lngTotalCount As Long
2146:   lngTotalCount = pIDArray.Count - 1
  Dim strTotalCount As String
2148:   strTotalCount = CStr(pIDArray.Count)
```

```
2149:   pPro.position = 1
2150:   psbar.ShowProgressBar "Refining Distance Matrix:  Step 1 of " & strTotalCount & "...", 1, _
          lngTotalCount + 1, 1, True

2153:   For lngFullIndex = 0 To pIDArray.Count - 1
2154:     strID = pIDArray.Element(lngFullIndex)
2155:     Set pSubArray = pDoneCollection.GetObject(strID)
2156:     booHasBeenDone = pSubArray.Element(1)
2157:     Set pColArray = pFinalCollection.Item(strID)
2158:     lngIndex = pColArray.Element(0)
2159:     lngIndex2 = pColArray.Element(1)

2161:     lngCounter = lngCounter + 1
2162:     pPro.Message = "Refining Distance Matrix:  Step " & CStr(lngCounter) & " of " & strTotalCount & " ( ID = " & strID & ")..."
2163:     psbar.StepProgressBar

    ' ---- PROGRESS METER STUFF
2166:     frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2167:     theProgressID = CStr(lngIndex + 1) & " and " & CStr(lngIndex2 + 1)
2168:     theProgressSpecific = "Checking connector between patch " & theProgressID & "..."
2169:     theDetailedDescription = theDetailedDescription & _
      " --> " & theProgressSpecific & vbCrLf
2171:     frmProgress.txtDetails.Text = theDetailedDescription
2172:     frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
2173:     frmProgress.Est_Time_Left lngFullIndex, "Boundary Intersections...", "Working on Analysis #"
2174:     frmProgress.Refresh
2175:     frmProgress.icnProgressLine.Refresh
2176:     DoEvents
    ' ---- END PROGRESS METER STUFF ---------------

2179:     If Not booHasBeenDone Then

 '     ThisDocument.Graphic_MakeFromGeometry pMxDoc, pArray.Element(0), "DeleteMatrix"
 '     ThisDocument.Graphic_MakeFromGeometry pMxDoc, pArray.Element(1), "DeleteMatrix"

2184:       dblDist = pColArray.Element(2)
2185:       Set pPolyline = pColArray.Element(3)
2186:       booCrossesEdge = CheckIntersectBoundary(pPolyline, pCorPolygon)

      ' CREATE NEW ARRAY FOR REVERSE DIRECTION, SO ONLY HAVE TO EXAMINE THIS ROUTE ONCE
      ' ONLY DO THIS IF THERE ACTUALLY IS A REVERSE DIRECTION UNDER CONSIDERATION

2191:       strRevID = CStr(lngIndex2) & "_" & CStr(lngIndex)
2192:       booHasRevDir = pDoneCollection.HasKey(strRevID)

2194:       If booHasRevDir Then
2195:         Set pNewColArray = New VarArray
```

```
2196:         pNewColArray.Add lngIndex2
2197:         pNewColArray.Add lngIndex
2198:         pNewColArray.Add dblDist
2199:         Set pSubArray = pDoneCollection.GetObject(strRevID)
2200:         lngRevIndex = pSubArray.Element(0)
2201:       End If

2203:       If booCrossesEdge Then

'        Debug.Print "LINE CROSSED!!!! BAD BAD BAD!!!  Line connecting polygon " & CStr(lngIndex) & _
'            " and " & CStr(lngIndex2) & "... uh oh!"
'        Debug.Print "Creating grid-based line..."
2208:         Set pGridPoly1 = pArray.Element(lngIndex)
2209:         Set pGridPoly2 = pArray.Element(lngIndex2)
  '      Set pGridPoly1 = pArray.Element(2)
  '      Set pGridPoly2 = pArray.Element(4)
2212:         If pCorRaster Is Nothing Then

          ' ---- PROGRESS METER STUFF
2215:           frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2216:           theDetailedDescription = theDetailedDescription & _
            "           Setting up raster analysis environment." & vbCrLf
2218:           frmProgress.txtDetails.Text = theDetailedDescription
2219:           frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
2220:           frmProgress.Est_Time_Left lngFullIndex, "Boundary Intersections...", "Setting Up Analysis Environment..."
2221:           frmProgress.Refresh
2222:           frmProgress.icnProgressLine.Refresh
2223:           DoEvents
          ' ---- END PROGRESS METER STUFF ---------------

2226:           Set pCorRaster = GenerateCorridorRaster(pCorPolygon, pEnv)

2228:         End If
        ' ---- PROGRESS METER STUFF
2230:         frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2231:         theDetailedDescription = theDetailedDescription & _
          "           Patch " & theProgressID & " Connector crosses outside corridor." & vbCrLf & _
          "           *** Creating grid-based connector..." & vbCrLf
2234:         frmProgress.txtDetails.Text = theDetailedDescription
2235:         frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
2236:         frmProgress.Est_Time_Left lngFullIndex + 1, "Boundary Intersections...", "Working on Analysis #"
2237:         frmProgress.Refresh
2238:         frmProgress.icnProgressLine.Refresh
2239:         DoEvents
        ' ---- END PROGRESS METER STUFF ---------------

  '      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pGridPoly1, "DeleteMatrix"
```

```
  '      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pGridPoly2, "DeleteMatrix"
  '      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pCorPolygon, "DeleteMatrix"
2245:         Set pGridLine = GridFunctions.CalcGridLine(pGridPoly1, pGridPoly2, pCorPolygon, pCorRaster, pEnv)
  '       Set pGridLine = pPolyline
  '      Dim zzzEnv As IEnvelope
  '      pEnv.GetExtent 3, zzzEnv
  '      ThisDocument.Graphic_MakeFromGeometry pMxDoc, zzzEnv, "DeleteMatrix"

        ' IT IS POSSIBLE THAT pGridLine WILL BE EMPTY IF THERE WAS NO WAY TO GENERATE A LEAST-COST PATH
        ' IF SO, REMOVE SEGMENT FROM CONSIDERATION
2253:         If pGridLine.IsEmpty Then
2254:           pRemoveKeyArray.Add strID
2255:           pRemoveKeyArray.Add strRevID
2256:         ElseIf pGridLine.length = 0 Then
2257:           pRemoveKeyArray.Add strID
2258:           pRemoveKeyArray.Add strRevID
2259:         Else
2260:           Set pGridLine = AttachPolylineToPatches(pGridLine, pGridPoly1, pGridPoly2)
2261:         End If

2263:         pColArray.Remove 3
2264:         pColArray.Insert 3, pGridLine

        'ADD TO ARRAY FOR REVERSE DIRECTION
2267:         If booHasRevDir Then
2268:           Set pClone = pGridLine
2269:           Set pReverseLine = pClone.Clone
2270:           pReverseLine.ReverseOrientation
2271:           pNewColArray.Add pReverseLine
2272:         End If
2273:       Else
  '       Debug.Print "Line connecting polygon " & CStr(lngIndex) & _
  '           " and " & CStr(lngIndex2) & " does not intersect boundary..."

        ' ---- PROGRESS METER STUFF
2278:         frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2279:         theDetailedDescription = theDetailedDescription & _
          "          Patch " & theProgressID & " Connector does not cross corridor boundary." & vbCrLf
2281:         frmProgress.txtDetails.Text = theDetailedDescription
2282:         frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
2283:         frmProgress.Est_Time_Left lngFullIndex, "Boundary Intersections...", "Working on Analysis #"
2284:         frmProgress.Refresh
2285:         frmProgress.icnProgressLine.Refresh
2286:         DoEvents
        ' ---- END PROGRESS METER STUFF ---------------

        'ADD TO ARRAY FOR REVERSE DIRECTION
```

```
2290:         If booHasRevDir Then
2291:           Set pClone = pPolyline
2292:           Set pReverseLine = pClone.Clone
2293:           pReverseLine.ReverseOrientation
2294:           pNewColArray.Add pReverseLine
2295:         End If
2296:       End If

2298:       If booHasRevDir Then
        ' REPLACE REVERSE LINE IN ORIGINAL FINAL COLLECTION
2300:         pFinalCollection.Remove strRevID
2301:         pFinalCollection.Add pNewColArray, strRevID

        ' MARK THIS REVERSE LINE AS HAVING BEEN DONE
2304:         Set pSubArray = pDoneCollection.GetObject(strRevID)
2305:         pSubArray.Remove 1
2306:         pSubArray.Add True
2307:         pDoneCollection.AddObject pSubArray, strRevID, True
2308:       End If


  '    ThisDocument.Graphic_MakeFromGeometry pMxDoc, pColArray.Element(3), "DeleteMatrix"
  '    ThisDocument.Graphic_MakeFromGeometry pMxDoc, pArray.Element(lngIndex), "DeleteMatrix"
  '    ThisDocument.Graphic_MakeFromGeometry pMxDoc, pArray.Element(lngIndex2), "DeleteMatrix"
  '    pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing
  '    ThisDocument.DeleteGraphicsByName pMxDoc, "DeleteMatrix"
2316:     Else

        ' ---- PROGRESS METER STUFF
2319:         frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2320:         theDetailedDescription = theDetailedDescription & _
          "          Patch " & theProgressID & " Connector has already been analyzed." & vbCrLf
2322:         frmProgress.txtDetails.Text = theDetailedDescription
2323:         frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
2324:         frmProgress.Est_Time_Left lngFullIndex, "Boundary Intersections...", "Working on Analysis #"
        ' ---- END PROGRESS METER STUFF ---------------

 '     Debug.Print "Line connecting polygon " & CStr(lngIndex) & _
 '           " and " & CStr(lngIndex2) & " has been analyzed previously..."
2329:     End If

    ' ---- PROGRESS METER STUFF
2332:     frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2333:     theDetailedDescription = theDetailedDescription & _
      "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
2335:     frmProgress.txtDetails.Text = theDetailedDescription
2336:     frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
```

```
2337:     frmProgress.Est_Time_Left lngFullIndex, "Boundary Intersections...", "Working on Analysis #"
2338:     frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2339:     If BailOutOfProgress(frmProgress, pExtensionConfig) Then
      Exit Sub
2341:     End If
    ' ---- END PROGRESS METER STUFF --------------------------------

2344:   Next lngFullIndex

2346:   pPro.position = 1
2347:   psbar.HideProgressBar

  ' FINALLY, ONLY CONSIDER LINES THAT DO NOT INTERSECT OTHER SHAPES
  ' ALSO REMOVE ANY SEGMENTS LISTED IN pRemoveKeyArray

  Dim pFinalIDArray As IStringArray
2353:       Set pFinalIDArray = New strArray
  Dim dblDistArray() As Double
      ReDim dblDistArray(pFinalCollection.Count)
  Dim pDistCollection As CollectionMod
2357:       Set pDistCollection = New CollectionMod
  Dim strFormatString As String
2359:       strFormatString = "0.000000000000"
  Dim strDistString As String
  Dim pFinalCollection3 As New CollectionMod
  Dim booIsInRemoveList As Boolean
  Dim lngRemoveKeyIndex As Long

2365:   For lngFullIndex = 0 To pIDArray.Count - 1
2366:     strID = pIDArray.Element(lngFullIndex)
2367:     Set pColArray = pFinalCollection.Item(strID)
2368:     lngIndex = pColArray.Element(0)
2369:     lngIndex2 = pColArray.Element(1)
2370:     dblDist = pColArray.Element(2)
2371:     booShouldReject = False
2372:     Set pPolyline = pColArray.Element(3)

    ' RECALCULATE DISTANCE HERE BECAUSE IT HAS LIKELY CHANGED
2375:     dblDist = pPolyline.length

    ' FOR DEBUGGING
'    Linkages.MyGeneralOperations.Graphic_MakeFromGeometry pMxDoc, pPolyline, "pre_delete_" & strID

    ' CHECK TO SEE IF KEY IS IN REMOVE LIST
2381:     booIsInRemoveList = False
2382:     For lngRemoveKeyIndex = 0 To pRemoveKeyArray.Count - 1
2383:       If pRemoveKeyArray.Element(lngRemoveKeyIndex) = strID Then
```

```
2384:         booIsInRemoveList = True
2385:         Exit For
2386:       End If
2387:     Next lngRemoveKeyIndex

2389:     If Not booIsInRemoveList Then
      ' CHECK IF CONNECTOR LINE INTERSECTS ANY SHAPES OTHER THAN THE TWO IN QUESTION
2391:       For lngCheckIndex = 0 To lngArrayMaxIndex
2392:         If (lngCheckIndex <> lngIndex) And (lngCheckIndex <> lngIndex2) Then
2393:           Set pShape1 = pArray.Element(lngCheckIndex)
2394:           Set pRelOp = pPolyline
2395:           If Not pRelOp.Disjoint(pShape1) Then    ' THIS CHECKS FOR INTERSECTION
2396:             booShouldReject = True
2397:             Exit For
2398:           End If
2399:         End If
2400:       Next lngCheckIndex

2402:       If Not booShouldReject Then
2403:         lngFinalIndex = lngFinalIndex + 1
2404:         strDistString = Format(dblDist, strFormatString)

2406:         Do While pDistCollection.HasKey(strDistString)
   '     Do While Not IsNull(pDistCollection.Item(strDistString))
2408:           dblDist = dblDist * 1.00000000001
2409:           strDistString = Format(dblDist, strFormatString)
2410:         Loop

2412:         dblDistArray(lngFinalIndex) = dblDist
2413:         pDistCollection.AddVariable strID, strDistString, False
2414:         pFinalIDArray.Add strID
  '      Debug.Print "Adding " & strID & " to pFinalCollection3..."
2416:         pFinalCollection3.AddObject pColArray, strID, False

  '      Debug.Print "Connecting " & CStr(lngIndex) & " to " & CStr(lngIndex2) & ":  Distance = " & _
                  CStr(dblDist) & ".     EMPTY POLYLINE = " & CStr(pPolyline.IsEmpty)

 '       MyGeneralOperations.Graphic_MakeFromGeometry pMxDoc, pPolyline, "test_matrix"
2422:       End If
2423:     End If
2424:   Next lngFullIndex
  ReDim Preserve dblDistArray(lngFinalIndex)
'  Debug.Print pFinalIDArray.Count

  Dim dblSortArray() As Double
  ReDim dblSortArray(lngFullIndex)
2430:   dblSortArray = dblDistArray
```

```
2432:   Call QuickSort.DoubleAscending(dblSortArray, 0, lngFinalIndex)


'   NET DICTIONARY -------------------------------
'   DICTIONARY STRUCTURED AS:  CollectionMod Object
'                   KEY = NODE ID STRING
'               ELEMENT = CollectionMod OF POSSIBLE MOVEMENT OPTIONS:  WATCH FOR 1-WAY SITUATIONS
'                         SUCH AS START NODE CAN ONLY GO OUT, END NODE ONLY HAS INCOMING
'                                 KEY = NODE ID STRING WHERE CONNECTOR GOES TO
'                             ELEMENT = IVarArray, {CONNECTOR LINE, CONNECTOR LENGTH}

    ' ---- PROGRESS METER STUFF
2444:   theProgressSpecific = "Identifying Best Path..."
2445:   theDetailedDescription = theDetailedDescription & _
    " --> " & theProgressSpecific & vbCrLf & _
    "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf & vbCrLf
2448:   frmProgress.txtDetails.Text = theDetailedDescription
2449:   frmProgress.txtDetails.SelStart = Len(theDetailedDescription)

2451:   frmProgress.Est_Time_Left lngFullIndex, "Final Analysis...", "Preliminary analysis of matrix..."
2452:   frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
2453:   If BailOutOfProgress(frmProgress, pExtensionConfig) Then
    Exit Sub
2455:   End If
2456:   frmProgress.Refresh
2457:   frmProgress.icnProgressLine.Refresh
  ' ---- END PROGRESS METER STUFF --------------------------------

  Dim theNetDictionary As New CollectionMod
  Dim theNodeDictionary As CollectionMod
  Dim theNodeElement As IVariantArray

  Dim lngFrom As Long
  Dim lngTo As Long
  Dim strFrom As String
  Dim strTo As String

  Dim pKeys() As String
2470:   pKeys = pFinalCollection3.ReturnKeys
  Dim pVariant As Variant

2473:   For lngFullIndex = 0 To lngFinalIndex
2474:     strID = pFinalIDArray.Element(lngFullIndex)
2475:     Set pColArray = pFinalCollection3.GetObject(strID)
2476:     lngFrom = pColArray.Element(0)
2477:     lngTo = pColArray.Element(1)
```

```
2478:     strFrom = CStr(lngFrom)
2479:     strTo = CStr(lngTo)
2480:     dblDist = pColArray.Element(2)
2481:     Set pPolyline = pColArray.Element(3)

    ' FOR DEBUGGING
'    Linkages.MyGeneralOperations.Graphic_MakeFromGeometry pMxDoc, pPolyline, "post_delete_" & strID & "_dist_" & CStr(Format(dblDist,
"0"))

2486:     If Not theNetDictionary.HasKey(strFrom) Then
2487:       Set theNodeDictionary = New CollectionMod
2488:       Set theNodeElement = New VarArray

2490:       theNodeElement.Add pPolyline
2491:       theNodeElement.Add dblDist

2493:       theNodeDictionary.SetOrReplace theNodeElement, strTo
2494:       theNetDictionary.SetOrReplace theNodeDictionary, strFrom
2495:     Else
2496:       Set theNodeDictionary = theNetDictionary.GetObject(strFrom)
2497:       If theNodeDictionary.HasKey(strTo) Then
2498:         Set theNodeElement = theNodeDictionary.GetObject(strTo)
2499:         theNodeElement.Add pPolyline
2500:         theNodeElement.Add dblDist
2501:       Else
2502:         Set theNodeElement = New VarArray
2503:         theNodeElement.Add pPolyline
2504:         theNodeElement.Add dblDist

2506:         theNodeDictionary.SetOrReplace theNodeElement, strTo
2507:       End If
2508:     End If

2510:   Next lngFullIndex
'theDictionaryOfConnectors = Dictionary.Make(theLineDistances.Count)
'

  Dim theDictionaryOfConnectors As New CollectionMod
  Dim anIndex As Long
'for each aLineDistance in theLineDistances
'  theData = theDictionaryOfDistances.Get(aLineDistance)
'  thePointA = theData.Get(0)
'  thePointB = theData.Get(1)
'  theLine = theData.Get(2)
'
'  theColor = Color.GetBlue
'
```

```
'  if ((theDictionaryOfConnectors.Get(thePointA) <> nil) AND
'      (theDictionaryOfConnectors.Get(thePointB) <> nil)) then
'      IsAcyclic = av.Run("Jennessent.CheckForAcyclicity", {thePointA, thePointB, theDictionaryOfConnectors})
'      if (IsAcyclic.Not) then
''        theColor = Color.GetGreen
''        theGraphic = GraphicShape.Make(theLine)
''        theGraphic.GetSymbol.SetColor(theColor)
''        theGraphic.SetName("Temp")
''        theGraphics.Add(theGraphic)
'        continue
'      end
'  end


  Dim lngHighIndex As Long
  Dim pIndexArray As IStringArray
  Dim pConnectionResult As IUnknown

  Dim IsAcyclic As Boolean
  Dim pRouteArray As esriSystem.IArray

  ' Dim strDebugReport As String

2547:   For anIndex = 0 To UBound(dblSortArray)
2548:     dblDist = dblSortArray(anIndex)
2549:     strID = pDistCollection.GetVariable(Format(dblDist, strFormatString))
2550:     Set pColArray = pFinalCollection3.GetObject(strID)

2552:     lngFrom = pColArray.Element(0)
2553:     lngTo = pColArray.Element(1)
2554:     strFrom = CStr(lngFrom)
2555:     strTo = CStr(lngTo)
2556:     Set pPolyline = pColArray.Element(3)

2558:     IsAcyclic = True
2559:     If theDictionaryOfConnectors.HasKey(strFrom) And theDictionaryOfConnectors.HasKey(strTo) Then
      ' CHECK FOR ACYCLICITY
2561:       IsAcyclic = CheckForAcyclicity(strFrom, strTo, theDictionaryOfConnectors)

2563:     End If

'    strDebugReport = strDebugReport & "Testing ID Val [" & strID & "]:  IsAcyclic = " & CStr(IsAcyclic) & vbCrLf & _
       "  --> Length = " & CStr(Format(pPolyline.length, "0")) & vbCrLf & _
       "  --> From = " & strFrom & vbCrLf & _
       "  --> To = " & strTo & vbCrLf & vbCrLf

'  theCurrentA = theDictionaryOfConnectors.Get(thePointA)
```

```
'  if (theCurrentA = nil) then
'    theDictionaryOfConnectors.Set(thePointA, {thePointB})
'  else
'    theCurrentA.Add(thePointB)
'  end
'  theCurrentB = theDictionaryOfConnectors.Get(thePointB)
'  if (theCurrentB = nil) then
'    theDictionaryOfConnectors.Set(thePointB, {thePointA})
'  else
'    theCurrentB.Add(thePointA)
'  end
'
2583:     If IsAcyclic Then

2585:       If Not theDictionaryOfConnectors.HasKey(strFrom) Then
2586:         Set pIndexArray = New strArray
2587:         pIndexArray.Add (strTo)
2588:         theDictionaryOfConnectors.AddObject pIndexArray, strFrom, False
2589:       Else
2590:         Set pIndexArray = theDictionaryOfConnectors.GetObject(strFrom)
2591:         pIndexArray.Add (strTo)
2592:       End If

2594:       If Not theDictionaryOfConnectors.HasKey(strTo) Then
2595:         Set pIndexArray = New strArray
2596:         pIndexArray.Add (strFrom)
2597:         theDictionaryOfConnectors.AddObject pIndexArray, strTo, False
2598:       Else
2599:         Set pIndexArray = theDictionaryOfConnectors.GetObject(strTo)
2600:         pIndexArray.Add (strFrom)
2601:       End If

      ' CHECK FOR CONNECTION
2604:       Set pRouteArray = CheckForConnection(strStartNode, strEndNode, theDictionaryOfConnectors, theNetDictionary)

'      strDebugReport = strDebugReport & "****  Connection Found = " & UCase(CStr(Not pRouteArray Is Nothing)) & vbCrLf & vbCrLf

2608:       If Not pRouteArray Is Nothing Then Exit For

'  theGraphic = GraphicShape.Make(theLine)
'  theGraphic.GetSymbol.SetColor(theColor)
'  theGraphic.SetName("Temp")
'  theGraphics.Add(theGraphic)
'
'  x = 1
'
'  theResult = av.Run("Jennessent.CheckForConnection", {theOriginPoint, theEndPoint, theDictionaryOfConnectors, theNetDictionary})
```

```
'  if (theResult <> nil) then
'    thePath = theResult
'    break
'  end
'
'end
2624:     End If

2626:   Next anIndex




  Dim pElement As ILineElement
  Dim pPolyElement As IFillShapeElement
  Dim pLineSymbol As ISimpleLineSymbol
2634:   Set pLineSymbol = New SimpleLineSymbol
  Dim pColor As IColor
2636:   Set pColor = MyGeneralOperations.MakeColorRGB(0, 100, 0)
  Dim pInternalColor As IColor
2638:   Set pInternalColor = MyGeneralOperations.MakeColorRGB(240, 255, 240)
2639:   pLineSymbol.Color = pColor
2640:   pLineSymbol.Width = 1.5

  Dim pCartoLine As ICartographicLineSymbol
2643:   Set pCartoLine = New CartographicLineSymbol
2644:   With pCartoLine
2645:     .Width = 1.5
2646:     .Color = pColor
2647:   End With

 'Create a Simple Fill
  Dim pSmplFill As ISimpleFillSymbol
2651:   Set pSmplFill = New SimpleFillSymbol

 'Create the Fill Symbol and set the properties
  Dim pFillSym As IFillSymbol
2655:   Set pFillSym = pSmplFill
2656:   With pFillSym
2657:     .Color = pInternalColor
2658:     .Outline = pCartoLine
2659:   End With


  Dim pGContainer As IGraphicsContainer
2663:   Set pGContainer = pMxDoc.FocusMap
  Dim pPolygon As IPolygon
```

```
  Dim anIndex2 As Long

  ' FOR GRADIENT FILL
  Dim pTempArray As IVariantArray
  Dim pGradStartPoint As IPoint
  Dim pGradEndPoint As IPoint
  Dim pPolyline2 As IPolyline
  Dim pStartPoint As IPoint
  Dim pEndPoint As IPoint
2674:   Set pStartPoint = New Point
2675:   Set pEndPoint = New Point
  Dim pStartPoint2 As IPoint
  Dim pEndPoint2 As IPoint
2678:   Set pStartPoint2 = New Point
2679:   Set pEndPoint2 = New Point
  Dim pProxOp As IProximityOperator
  Dim dblStartDist As Double
  Dim dblEndDist As Double
  Dim dblBearing As Double
  Dim pGradFillSym As IGradientFillSymbol
  Dim pAlgColRamp As IAlgorithmicColorRamp
2686:   Set pAlgColRamp = New AlgorithmicColorRamp
   '** Set the Start and End Colors
2688:   pAlgColRamp.ToColor = pColor
2689:   pAlgColRamp.FromColor = pInternalColor
2690:   pAlgColRamp.Algorithm = esriHSVAlgorithm
2691:   pAlgColRamp.SIZE = 30
2692:   pAlgColRamp.CreateRamp (True)

  Dim pLineArray As IVariantArray
  Dim booWasSuccessful As Boolean

2697:   If pRouteArray Is Nothing Then
2698:     MsgBox "Unable to determine route!  Bailing out..."
2699:   Else
    Dim dblFinalDistances() As Double
    ReDim dblFinalDistances(pRouteArray.Count - 1)

2703:     For anIndex = 0 To pRouteArray.Count - 1
2704:       Set pLineArray = pRouteArray.Element(anIndex)
'      Debug.Print "Moving from " & pLineArray.Element(2) & " to "; pLineArray.Element(3) _
         & ";    Distance = " & pLineArray.Element(1)

2708:       Set pPolyline = pLineArray.Element(0)
2709:       dblFinalDistances(anIndex) = pPolyline.length

2711:       Set pElement = MyGeneralOperations.Graphic_ReturnElementFromGeometry(pMxDoc, pPolyline, "Route_Graphics", False)
```

```
2712:       pElement.Symbol = pLineSymbol

      ' ADD GRAPHIC TO GRAPHICS CONTAINER
2715:       pGContainer.AddElement pElement, 0

2717:       Set pGradFillSym = New GradientFillSymbol

2719:       pPolyline.QueryFromPoint pStartPoint
2720:       pPolyline.QueryToPoint pEndPoint

      ' GET POLYGONS
2723:       If anIndex = 0 Then
2724:         Set pPolygon = pArray.Element(0)
2725:         Set pPolyElement = MyGeneralOperations.Graphic_ReturnElementFromGeometry(pMxDoc, pPolygon, "Route_Graphics", False)

        ' FOR GRADIENT FILL
2728:         Set pProxOp = pStartPoint
2729:         dblStartDist = pProxOp.ReturnDistance(pPolygon)
2730:         Set pProxOp = pEndPoint
2731:         dblEndDist = pProxOp.ReturnDistance(pPolygon)
2732:         If dblStartDist < dblEndDist Then
2733:           dblBearing = MyGeometricOperations.CalcBearing(pStartPoint, pEndPoint)
2734:         Else
2735:           dblBearing = MyGeometricOperations.CalcBearing(pEndPoint, pStartPoint)
2736:         End If
2737:         dblBearing = 90 - dblBearing
2738:         If dblBearing < -180 Then dblBearing = dblBearing + 360
2739:         pGradFillSym.GradientAngle = dblBearing
2740:         pGradFillSym.Outline = pCartoLine
2741:         pGradFillSym.ColorRamp = pAlgColRamp
2742:         pGradFillSym.IntervalCount = 30
2743:         pGradFillSym.Style = esriGFSLinear
2744:         pGradFillSym.GradientPercentage = 1

'        Debug.Print dblStartDist & ", " & dblEndDist & ", " & dblBearing

2748:         pPolyElement.Symbol = pGradFillSym
2749:         pGContainer.AddElement pPolyElement, 0
2750:       End If

      ' FOR GRADIENT FILL
2753:       Set pProxOp = pStartPoint
2754:       dblStartDist = pProxOp.ReturnDistance(pPolygon)
2755:       Set pProxOp = pEndPoint
2756:       dblEndDist = pProxOp.ReturnDistance(pPolygon)
2757:       If anIndex <> pRouteArray.Count - 1 Then    ' THEN NOT LAST POLYGON, AND CAN SHADE BETWEEN CONNECTORS
2758:         If dblStartDist < dblEndDist Then
```

```
2759:           Set pGradStartPoint = pStartPoint
2760:         Else
2761:           Set pGradStartPoint = pEndPoint
2762:         End If
2763:         Set pTempArray = pRouteArray.Element(anIndex + 1)
2764:         Set pPolyline2 = pTempArray.Element(0)
2765:         pPolyline2.QueryFromPoint pStartPoint2
2766:         pPolyline2.QueryToPoint pEndPoint2
2767:         Set pProxOp = pStartPoint
2768:         dblStartDist = pProxOp.ReturnDistance(pPolygon)
2769:         Set pProxOp = pEndPoint
2770:         dblEndDist = pProxOp.ReturnDistance(pPolygon)
2771:         If dblStartDist < dblEndDist Then
2772:           Set pGradEndPoint = pStartPoint2
2773:         Else
2774:           Set pGradEndPoint = pEndPoint2
2775:         End If
2776:         dblBearing = MyGeometricOperations.CalcBearing(pGradStartPoint, pGradEndPoint)
2777:       Else
2778:         If dblStartDist < dblEndDist Then
2779:           dblBearing = MyGeometricOperations.CalcBearing(pStartPoint, pEndPoint)
2780:         Else
2781:           dblBearing = MyGeometricOperations.CalcBearing(pEndPoint, pStartPoint)
2782:         End If
2783:       End If
2784:       dblBearing = 90 - dblBearing
2785:       If dblBearing < -180 Then dblBearing = dblBearing + 360
2786:       pGradFillSym.GradientAngle = dblBearing
2787:       pGradFillSym.Outline = pCartoLine
2788:       pGradFillSym.ColorRamp = pAlgColRamp
2789:       pGradFillSym.IntervalCount = 145
2790:       pGradFillSym.Style = esriGFSLinear
2791:       pGradFillSym.GradientPercentage = 1

2793:       anIndex2 = CLng(pLineArray.Element(3))
2794:       Set pPolygon = pArray.Element(anIndex2)
2795:       Set pPolyElement = MyGeneralOperations.Graphic_ReturnElementFromGeometry(pMxDoc, pPolygon, "Route_Graphics", False)
2796:       pPolyElement.Symbol = pGradFillSym
2797:       pGContainer.AddElement pPolyElement, 0
2798:     Next anIndex

2800:     booWasSuccessful = True
2801:   End If

2803:   pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing

  ' PROGRESS METER STUFF
```

```
  Dim theDateString As String
  Dim theElapsedTimeString As String

2810:   theDateString = Format(Now, "long date") & ";  " & Format(Now, "long time")
2811:   Screen.MousePointer = vbDefault
  Dim theTimeBegan As Date
  Dim theTimeEnd As Date
2814:   theTimeBegan = frmProgress.ProgBeginTime
2815:   theTimeEnd = Now

2817:   theElapsedTimeString = MyGeneralOperations.ReturnTimeElapsed(theTimeBegan, theTimeEnd)

  Dim theReport As String
2820:   theReport = "Operation completed at " & theDateString & vbCrLf & _
      "-----------------------------------------------------------" & vbCrLf & theElapsedTimeString

  ' FOR DEBUGGING
'  theReport = theReport & vbCrLf & "-----------------------------------------------" & vbCrLf & strDebugReport

  Dim strCurrentReport As String
2827:   strCurrentReport = frmProgress.txtDetails.Text
2828:   frmProgress.txtDetails.Text = strCurrentReport & vbCrLf & theReport

  Dim strTimeElapsedRTF As String
2831:   strTimeElapsedRTF = MyGeneralOperations.ReturnTimeElapsedRTF(theTimeBegan, theTimeEnd, 8)

2833:   If booWasSuccessful Then
    ' MAKE OUTPUT REPORT
2835:     Call MakePatchReport(dblFinalDistances, pMxDoc, strTimeElapsedRTF, pParamDetails)
2836:   End If

'  For lngFullIndex = 0 To lngFinalIndex
'    Debug.Print "Distance = " & pSortArray(lngFullIndex) & ",  " & "Node ID Value for " & _
'        Format(pSortArray(lngFullIndex), strFormatString) & " = " & _
'        pDistCollection.GetVariable(Format(pSortArray(lngFullIndex), strFormatString))
'  Next lngFullIndex

' Debug.Print "Calculation Time = " & (Timer - varStartTime) & " seconds..."


  Exit Sub
ErrorHandler:
  HandleError True, "ImplementKruskall " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

```
Public Sub MakePatchReport(dblDistances() As Double, pMxDoc As IMxDocument, strElapsedTime As String, _
    pParamDetails As esriSystem.IVariantArray)
  On Error GoTo ErrorHandler

  Dim pApp As IApplication
  Dim pDoc As IDocument
2858:   Set pDoc = pMxDoc
2859:   Set pApp = pDoc.Parent

2861:   Call QuickSort.DoubleDescending(dblDistances, 0, UBound(dblDistances))

  Dim strReport As String
  Dim strHab1 As String
  Dim strHab2 As String
  Dim strCor As String
  Dim booUsePatches As Boolean
  Dim strPatch As String
  Dim booUseAllPatches As Boolean
  Dim strPatchField As String
  Dim strOperator As String
  Dim strValue As String

2874:   strHab1 = pParamDetails.Element(0)
2875:   strHab2 = pParamDetails.Element(1)
2876:   strCor = pParamDetails.Element(2)
2877:   booUsePatches = pParamDetails.Element(3)
2878:   strPatch = pParamDetails.Element(4)
2879:   booUseAllPatches = pParamDetails.Element(5)
2880:   strPatchField = pParamDetails.Element(6)
2881:   strOperator = pParamDetails.Element(7)
2882:   strValue = pParamDetails.Element(8)

  ' MAKE NEW TABLE
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit
  Dim pNewFields As IFields
  Dim pNewFieldsEdit As IFieldsEdit
2889:   Set pNewFields = New Fields
2890:   Set pNewFieldsEdit = pNewFields
2891:   pNewFieldsEdit.FieldCount = 2

  ' MAKE UNIQUE ID FIELD
2894:   Set pNewField = New Field
2895:   Set pNewFieldEdit = pNewField
2896:   pNewFieldEdit.Name = "Unique_ID"
2897:   pNewFieldEdit.Type = esriFieldTypeInteger
2898:   pNewFieldEdit.Precision = 8
```

```
2899:   Set pNewFieldsEdit.Field(0) = pNewField

  ' MAKE LENGTH FIELD
2902:   Set pNewField = New Field
2903:   Set pNewFieldEdit = pNewField
2904:   With pNewFieldEdit
2905:     .Type = esriFieldTypeDouble
2906:     .Name = "Seg_Length"
2907:     .Precision = 14
2908:     .Scale = 8
2909:   End With
2910:   Set pNewFieldsEdit.Field(1) = pNewField

  ' WORKSPACE
  ' FIRST SEE IF IT HAS BEEN SAVED TO EXTENSION PROPERTIES.  THIS PROPERTY WILL BE EMPTY THE FIRST TIME THE DIALOG
  '      IS OPENED, BUT EACH TIME THEREAFTER IT WILL HAVE A VALUE.
  ' IF NOT IN EXTENSION PROPERTY, THEN CHECK ArcGIS LAST SAVE TO LOCATION
  ' IF THIS DOESN'T WORK, USE MxDoc PATH NAME.
  Dim strDirPath As String
  Dim strUserName As String
  Dim strFName As String

  Dim newUid As New uID
  Dim pExtConfig As IExtensionConfig
2923:   newUid.Value = "Linkages.Extension"
2924:   Set pExtConfig = pApp.FindExtensionByCLSID(newUid)
  Dim ext As Linkages.Extension
2926:   Set ext = pExtConfig

2928:   strDirPath = ext.ClipDirectoryPath
2929:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
2930:     strDirPath = Linkages.aml_func_mod.ReturnArcGISGeneralDir(enumLastSaveToLocation)
2931:   End If
2932:   If Not Linkages.aml_func_mod.ExistFileDir(strDirPath) Then
2933:     strDirPath = Linkages.aml_func_mod.GetFullFileString(Linkages.aml_func_mod.GetMxDocPath(pApp))
2934:     strDirPath = Linkages.aml_func_mod.ReturnDir(strDirPath)
2935:   End If

2937:   If Right(strDirPath, 1) <> "\" And Right(strDirPath, 1) <> "/" Then
2938:     strDirPath = strDirPath & "\"
2939:   End If
2940:   strFName = strDirPath & "segment_lengths.dbf"

2942:   strFName = Linkages.aml_func_mod.MakeUniqueFilename(strFName)
  Dim pTable As ITable
2944:   Set pTable = Linkages.aml_func_mod.CreatedBASETable(strFName, pNewFields)
  Dim pCursor As ICursor
```

```
  Dim pRowBuffer As IRowBuffer
2947:   Set pCursor = pTable.Insert(True)
2948:   Set pRowBuffer = pTable.CreateRowBuffer

2950:   strReport = "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
              "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\b\f0\fs16 Report of Patch Analysis:\b0\par" & vbCrLf & _
              "\b ----------------------------------------------------------\b0\par" & vbCrLf & _
              "\pard\b Habitat Block #1 \b0 = " & strHab1 & "\par" & vbCrLf & _
              "\b Habitat Block #2 \b0 = " & strHab2 & "\par" & vbCrLf & _
              "\b Corridor Polygon \b0 = " & strCor & "\par" & vbCrLf
2956:   If Not booUsePatches Then
2957:     strReport = strReport & "No patch layer used in analysis...\par" & vbCrLf
2958:   Else
2959:     strReport = strReport & "\b Patch Layer \b0 = " & strPatch & "\par" & vbCrLf
2960:     If booUseAllPatches Then
2961:       strReport = strReport & "  ** All patch polygons considered in analysis...\par" & vbCrLf
2962:     Else
2963:       strReport = strReport & "  ** Patch Query String:  " & strPatchField & " " & _
          Linkages.aml_func_mod.BasicTrimAvenue(strOperator, " ", " ") & " " & _
          Linkages.aml_func_mod.BasicTrimAvenue(strValue, " ", " ") & "\par" & vbCrLf
2966:     End If
2967:   End If
2968:   strReport = strReport & _
              "\pard\qc\b ----------------------------------------------------------\b0\par" & vbCrLf & _
              "\pard\b " & CStr(UBound(dblDistances) + 1) & " segments required to move from one " & _
              "habitat block to the other.\b0\par" & vbCrLf & _
              "Segment lengths listed in decreasing order:\par" & vbCrLf

  Dim lngIndex As Long
2975:   For lngIndex = 0 To UBound(dblDistances)
2976:     strReport = strReport & "     " & CStr(lngIndex + 1) & "] " & _
        CStr(Format(dblDistances(lngIndex), "0.0000000")) & "\par" & vbCrLf
2978:     pRowBuffer.Value(pRowBuffer.Fields.FindField("Unique_ID")) = (lngIndex + 1)
2979:     pRowBuffer.Value(pRowBuffer.Fields.FindField("Seg_Length")) = dblDistances(lngIndex)
2980:     pCursor.InsertRow pRowBuffer
2981:   Next lngIndex
2982:   strReport = strReport & "\par" & vbCrLf & _
              "\b Table of Segment Lengths saved to:\b0\par" & vbCrLf & _
              Linkages.aml_func_mod.SubstituteString(strFName, "\", "\\") & "\par" & vbCrLf & _
              "\par" & vbCrLf & _
              "\b Note: \b0  You can use the ""Create New Shapefile"" function to convert " & _
              "your graphic patch polygons and segment polylines to new polyline and polygon shapefiles.  " & _
              "Polygons and polylines produced by this analysis will be named ""Route_Graphics"" in the " & _
              "new shapefile attribute tables.\par" & vbCrLf & _
              "\pard\qc\b ----------------------------------------------------------\b0\par" & vbCrLf & strElapsedTime _
              & "\par" & vbCrLf & "}"
```

```
2993:   pCursor.Flush

  ' MAKE TABLE AND ADD TO MAP DOCUMENT
  Dim pNewStandaloneTable As IStandaloneTable
  Dim pTableWindow2 As ITableWindow2
  Dim pStandaloneTableCollection As IStandaloneTableCollection

3000:   Set pNewStandaloneTable = New StandaloneTable
3001:   Set pNewStandaloneTable.Table = pTable

3003:   Set pTableWindow2 = New TableWindow

3005:   With pTableWindow2
3006:     Set .StandaloneTable = pNewStandaloneTable
3007:     Set .Application = pApp
3008:     .TableSelectionAction = esriSelectFeatures
3009:     .ShowAliasNamesInColumnHeadings = True
3010:     .ShowSelected = False
3011:     .Show True
3012:   End With
3013:   Set pStandaloneTableCollection = pMxDoc.FocusMap
3014:   pStandaloneTableCollection.AddStandaloneTable pNewStandaloneTable


  Dim frmReportForm As New frmReport_modal
3018:   frmReportForm.txtReport.TextRTF = strReport
3019:   frmReportForm.Show vbModal

3021:   pMxDoc.UpdateContents



  Exit Sub
ErrorHandler:
  HandleError True, "MakePatchReport " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Function CheckForConnection(strStartNode As String, strEndNode As String, _
    theDictionaryOfConnectors As CollectionMod, theNetDictionary As CollectionMod) As esriSystem.IArray
  On Error GoTo ErrorHandler


'   NET DICTIONARY -------------------------------
'   DICTIONARY STRUCTURED AS:  CollectionMod Object
'                   KEY = NODE ID STRING
'               ELEMENT = CollectionMod OF POSSIBLE MOVEMENT OPTIONS:  WATCH FOR 1-WAY SITUATIONS
```

```
'                         SUCH AS START NODE CAN ONLY GO OUT, END NODE ONLY HAS INCOMING
'                                 KEY = NODE ID STRING WHERE CONNECTOR GOES TO
'                             ELEMENT = IVarArray, {CONNECTOR LINE, CONNECTOR LENGTH}

'   theDictionaryOfConnectors ----------------------------
'   DICTIONARY STRUCTURED AS:  CollectionMod Object
'                     KEY = NODE ID STRING
'                ELEMENT = IStringArray OF POSSIBLE CONNECTION NODE ID STRINGS

Dim theReturnDictionary As esriSystem.IArray

' IF START OR END NODES HAVE NEVER BEEN RECORDED IN theDictionaryOfConnectors THEN THERE CANNOT BE A PATH
3051: If (Not theDictionaryOfConnectors.HasKey(strStartNode)) Or (Not theDictionaryOfConnectors.HasKey(strEndNode)) Then
3052:   Set CheckForConnection = Nothing
  Exit Function
3054: End If

Dim theDoneDictionary As New CollectionMod
3057: theDoneDictionary.SetOrReplace True, strStartNode

Dim pPathArray As IStringArray
3060: Set pPathArray = New strArray
3061: pPathArray.Add strStartNode

Dim thePathDictionary As New CollectionMod
3064: thePathDictionary.AddObject pPathArray, strStartNode, False

Dim IsConnected As Boolean
3067: IsConnected = False

Dim theListOfPossibleConnectors As IStringArray
3070: Set theListOfPossibleConnectors = theDictionaryOfConnectors.GetObject(strStartNode)

Dim strConnector As String
Dim lngIndex As Long
3074: For lngIndex = 0 To theListOfPossibleConnectors.Count - 1
3075:   strConnector = theListOfPossibleConnectors.Element(lngIndex)
3076:   Set pPathArray = New strArray
3077:   pPathArray.Add strStartNode
3078:   pPathArray.Add strConnector
3079:   thePathDictionary.AddObject pPathArray, strConnector, True
3080: Next lngIndex

Dim pNewList As IStringArray
Dim theCheckPoint As String
Dim pFinalList As IStringArray
Dim pNewPoints As IStringArray
```

```
Dim lngNodeIndex As Long
Dim strNode As String
Dim theNodeDictionary As CollectionMod
Dim strNextNode As String
Dim pPathData As IVariantArray
Dim theCurrentList As IStringArray
Dim strNewPoint As String
Dim strSaveList As IStringArray
Dim pReturnPathData As IVariantArray
Dim anIndex As Long
Dim lngIndex2 As Long

3099: Set theReturnDictionary = Nothing

3101: Do While Not IsConnected And (theListOfPossibleConnectors.Count > 0)
3102:   Set pNewList = New strArray
3103:   For anIndex = 0 To theListOfPossibleConnectors.Count - 1
3104:     theCheckPoint = theListOfPossibleConnectors.Element(anIndex)
3105:     If theCheckPoint = strEndNode Then
3106:       IsConnected = True
3107:       Set pFinalList = thePathDictionary.GetObject(strEndNode)
3108:       Set theReturnDictionary = New esriSystem.Array

3110:       For lngNodeIndex = 0 To pFinalList.Count - 2
3111:         strNode = pFinalList.Element(lngNodeIndex)
3112:         Set theNodeDictionary = theNetDictionary.GetObject(strNode)
3113:         strNextNode = pFinalList.Element(lngNodeIndex + 1)

3115:         Set pPathData = theNodeDictionary.GetObject(strNextNode)
3116:         Set pReturnPathData = New VarArray
3117:         pReturnPathData.Add pPathData.Element(0)
3118:         pReturnPathData.Add pPathData.Element(1)
3119:         pReturnPathData.Add strNode
3120:         pReturnPathData.Add strNextNode

3122:         theReturnDictionary.Add pReturnPathData      ' CONTAINS (0) POLYLINE; (1) LENGTH

3124:       Next lngNodeIndex

3126:       Set CheckForConnection = theReturnDictionary
      Exit Function     ' <---------------------------------------- EXIT FUNCTION WITH A FULL ROUTE AT THIS POINT
3128:     ElseIf Not theDoneDictionary.HasKey(theCheckPoint) Then         ' DON'T WANT IT DOUBLING BACK
3129:       theDoneDictionary.SetOrReplace True, theCheckPoint
3130:       Set theCurrentList = thePathDictionary.GetObject(theCheckPoint)
3131:       If theDictionaryOfConnectors.HasKey(theCheckPoint) Then
3132:         Set pNewPoints = theDictionaryOfConnectors.GetObject(theCheckPoint)
```

```
3133:         For lngIndex = 0 To pNewPoints.Count - 1
3134:           strNewPoint = pNewPoints.Element(lngIndex)
3135:           Set strSaveList = New strArray
3136:           For lngIndex2 = 0 To theCurrentList.Count - 1
3137:             strSaveList.Add theCurrentList.Element(lngIndex2)
3138:           Next lngIndex2
3139:           strSaveList.Add strNewPoint
3140:           thePathDictionary.AddObject strSaveList, strNewPoint, True

3142:           pNewList.Add strNewPoint
3143:         Next lngIndex
3144:       End If
3145:     End If
3146:   Next anIndex

3148:   Set theListOfPossibleConnectors = pNewList
3149: Loop

3151: Set CheckForConnection = theReturnDictionary


  Exit Function
ErrorHandler:
  HandleError True, "CheckForConnection " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function CheckForAcyclicity(ByRef strNodeA As String, strNodeB As String, theDictionaryOfConnectors As CollectionMod) As
Boolean
  On Error GoTo ErrorHandler


' Jennessent.CheckForAcyclicity

' TREE IS ACYCLIC IF POINT A AND POINT B ARE ALREADY CONNECTED

Dim pListOfPossibleConnectors As IStringArray
3168: Set pListOfPossibleConnectors = theDictionaryOfConnectors.GetObject(strNodeA)

Dim IsAcyclic As Boolean
3171: IsAcyclic = True
Dim theCheckPoint As String

Dim theDoneDictionary As CollectionMod
3175: Set theDoneDictionary = New CollectionMod
3176: theDoneDictionary.AddVariable True, strNodeA, False
```

```
Dim pNewList As IStringArray
Dim lngIndex As Long
Dim lngCheckIndex As Long
Dim pNewPoints As IStringArray
Dim lngIndex2 As Long

3184: Do While IsAcyclic And (pListOfPossibleConnectors.Count > 0)
3185:   Set pNewList = New strArray
3186:   For lngIndex = 0 To (pListOfPossibleConnectors.Count - 1)
3187:     theCheckPoint = pListOfPossibleConnectors.Element(lngIndex)

3189:     If theCheckPoint = strNodeB Then
3190:       IsAcyclic = False
3191:       Exit For

3193:     ElseIf Not theDoneDictionary.HasKey(theCheckPoint) Then  ' DON'T WANT TO HAVE IT DOUBLING BACK
3194:       theDoneDictionary.SetOrReplace True, theCheckPoint
3195:       Set pNewPoints = theDictionaryOfConnectors.GetObject(theCheckPoint)
3196:       If Not pNewPoints Is Nothing Then
3197:         For lngIndex2 = 0 To pNewPoints.Count - 1
3198:           pNewList.Add pNewPoints.Element(lngIndex2)
3199:         Next lngIndex2
3200:       End If
3201:     End If
3202:   Next lngIndex

3204:   Set pListOfPossibleConnectors = pNewList
3205: Loop

3207: CheckForAcyclicity = IsAcyclic


  Exit Function
ErrorHandler:
  HandleError True, "CheckForAcyclicity " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Public Function ClipPolysToCorridor(ByVal pArray As IArray, pCorPolygon As IPolygon, pApp As IApplication) As Collection
  On Error GoTo ErrorHandler


'      Dim pMxDoc As IMxDocument
'      Set pMxDoc = ThisDocument


  Dim pResponseCollection As New Collection
  Dim StartIntersects As Boolean
```

```
  Dim EndIntersects As Boolean
  Dim pReturnArray As IArray
  Dim pWorked As Boolean
  Dim strMessage As String

  Dim pRelOp As IRelationalOperator
  Dim pTopOp As ITopologicalOperator
  Dim pTopOp2 As ITopologicalOperator
  Dim pTopOp4 As ITopologicalOperator2
  Dim pProxOp As IProximityOperator
  Dim pSpatialRef As ISpatialReference

  Dim pTopoOpSimp As ITopologicalOperator

  Dim anIndex As Long
  Dim pPolygon As IPolygon
  Dim pStartPolygon As IPolygon
  Dim pEndPolygon As IPolygon
  Dim pIntPolygon As IPolygon
  Dim pCheckCountPoly As IPolygon4
  Dim pGeometryBag As IGeometryCollection
  Dim pSubPoly As IPolygon4
  Dim anIndex2 As Long

3248:   Set pReturnArray = New esriSystem.Array
3249:   Set pRelOp = pCorPolygon
3250:   Set pTopOp = pCorPolygon

  Dim pCombinedArray As esriSystem.IArray
  Dim pCombinedArrayInfo As esriSystem.IVariantArray
  Dim booAllIntersects As Boolean

    ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
3258:   Set psbar = pApp.StatusBar
  Dim pPro As IStepProgressor
3260:   Set pPro = psbar.ProgressBar
  Dim lngCounter As Long
3262:   lngCounter = 0
  Dim lngTotalCount As Long
3264:   lngTotalCount = ((pArray.Count - 2) * 3)
  Dim strTotalCount As String
3266:   strTotalCount = "approximately " & CStr(lngTotalCount)
3267:   pPro.position = 1
3268:   psbar.ShowProgressBar "Clipping patch polygons to corridor:  Step 1 of " & strTotalCount & "...", 1, _
          lngTotalCount, 1, True
```

```
3271:   If pArray.Count = 1 Then
3272:     strMessage = strMessage + "Origin polygon not separate from Destination polygon!  No corridor necessary!..." + vbCrLf
3273:   Else

      ' EXTRACT START AND END POLYGONS
3276:     Set pStartPolygon = pArray.Element(0)
3277:     pArray.Remove (0)
3278:     Set pEndPolygon = pArray.Element(pArray.Count - 1)
3279:     pArray.Remove (pArray.Count - 1)

    ' FIRST SEE IF START AND END POLYGONS INTERSECT, OR IF THEY BOTH INTERSECT CORRIDOR

3283:     pPro.Message = "Confirming Habitat Block 1 intersects Corridor..."
'    Set pStartPolygon = pArray.Element(0)
3285:     StartIntersects = Not pRelOp.Disjoint(pStartPolygon)
3286:     If Not StartIntersects Then strMessage = strMessage + "Origin polygon does not intersect corridor..." + vbCrLf

3288:     pPro.Message = "Confirming Habitat Block 2 intersects Corridor..."
'    Set pEndPolygon = pArray.Element(pArray.Count - 1)
3290:     EndIntersects = Not pRelOp.Disjoint(pEndPolygon)
3291:     If Not EndIntersects Then strMessage = strMessage + "Destination polygon does not intersect corridor..." + vbCrLf

    Dim pRelOp3 As IRelationalOperator
3294:     Set pRelOp3 = pStartPolygon
3295:     If Not pRelOp3.Disjoint(pEndPolygon) Then
3296:         strMessage = strMessage + "Origin polygon intersects with Destination polygon!  No corridor necessary!..." + vbCrLf
3297:     End If
3298:   End If

3300:   If Not (StartIntersects And EndIntersects) Then
3301:     pWorked = False
3302:     Set pReturnArray = New esriSystem.Array
3303:     pReturnArray.Add pStartPolygon
3304:     pReturnArray.Add pEndPolygon

3306:   Else
    Dim pRelOp2 As IRelationalOperator

3309:     pWorked = False
3310:     strMessage = ""

    ' CLIP START AND END POLYGONS TO CORRIDOR
3313:     Set pStartPolygon = pTopOp.Intersect(pStartPolygon, esriGeometry2Dimension)
3314:     Set pEndPolygon = pTopOp.Intersect(pEndPolygon, esriGeometry2Dimension)
'    Module4.DeleteGraphicsByName Document, "test_order"
'    Module4.Graphic_MakeFromGeometry Document, pCorPolygon, "test_order"
'    Module4.Graphic_MakeFromGeometry Document, pStartPolygon, "test_order"
```

```
'    Module4.Graphic_MakeFromGeometry Document, pEndPolygon, "test_order"

3320:     If pArray.Count = 0 Then
'      strMessage = "Array contained no polygons!"
'      StartIntersects = False
'      EndIntersects = False
3324:       Set pReturnArray = New esriSystem.Array
3325:       pReturnArray.Add pStartPolygon
3326:       pReturnArray.Add pEndPolygon
3327:       pWorked = True
3328:     Else
      ' CONVERT ANY ODD SHAPES TO NORMAL POLYGONS BEFORE DOING ANYTHING ELSE
      ' FORCE OBJECT TO BE POLYGON IF THE FIRST SEGMENT IS A CURVE

      Dim booWorkingWithCurves As Boolean
      Dim pSegmentCollectionCurves As ISegmentCollection
      Dim pSegmentCurve As ISegment
      Dim pGeometryTypeA As esriGeometryType
      Dim pNewPoints As IPointCollection
      Dim pNewPolygon As IPointCollection
      Dim pRecreatedPolygon As IPolygon
      Dim pRevisedArray As esriSystem.IArray
3340:       Set pRevisedArray = New esriSystem.Array

   '   MsgBox "Before:  " & pArray.Count

3344:       For anIndex = 0 To pArray.Count - 1

3346:         lngCounter = lngCounter + 1
3347:         pPro.Message = "Clipping patch polygons to corridor:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
3348:         psbar.StepProgressBar

3350:         Set pPolygon = pArray.Element(anIndex)

3352:         Set pSegmentCollectionCurves = pPolygon
3353:         Set pSegmentCurve = pSegmentCollectionCurves.Segment(0)
3354:         pGeometryTypeA = pSegmentCurve.GeometryType

3356:         booWorkingWithCurves = (pGeometryTypeA = esriGeometryBezier3Curve) Or _
          (pGeometryTypeA = esriGeometryCircularArc) Or _
          (pGeometryTypeA = esriGeometryEllipticArc)

3360:         If booWorkingWithCurves Then
3361:           Set pNewPoints = GridFunctions.EllipticArcToPolygon2(pSegmentCollectionCurves, 100)
3362:           Set pNewPolygon = New Polygon
3363:           pNewPolygon.SetPointCollection pNewPoints
3364:           Set pRecreatedPolygon = pNewPolygon
```

```
3365:           Set pTopOp4 = pRecreatedPolygon
3366:           pTopOp4.IsKnownSimple = False
3367:           pTopOp4.Simplify
3368:           pRevisedArray.Add pRecreatedPolygon
  '       ThisDocument.Graphic_MakeFromGeometry pMxDoc, pNewPolygon, "test_order"
3370:         Else
3371:           pRevisedArray.Add pPolygon
  '       ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPolygon, "test_order"
3373:         End If
3374:       Next anIndex
3375:       Set pArray = pRevisedArray
      ' DONE CONVERTING CIRCLES, ELLIPTIC ARCS AND BEZIER CURVES TO POLYGONS

  '    ThisDocument.DeleteGraphicsByName pMxDoc, "test_order"
  '    For anIndex = 0 To pArray.Count - 1
  '      Set pPolygon = pArray.Element(anIndex)
  '      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPolygon, "test_order"
  '    Next anIndex

      ' INTERSECT POLYGONS WITH CORRIDOR, AND SEPARATE SUB-POLYGONS
3385:       For anIndex = 0 To pArray.Count - 1
3386:         lngCounter = lngCounter + 1
3387:         pPro.Message = "Checking for Multiple Strands:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
3388:         psbar.StepProgressBar

3390:         Set pPolygon = pArray.Element(anIndex)
3391:         Set pSpatialRef = pPolygon.SpatialReference
3392:         Set pIntPolygon = pTopOp.Intersect(pPolygon, esriGeometry2Dimension)
3393:         Set pTopoOpSimp = pIntPolygon

3395:         pTopoOpSimp.Simplify

3397:         Set pCheckCountPoly = pIntPolygon
        ' SPLIT INTO MULTIPLE POLYGONS IF NECESSARY; ONLY IF NOT FIRST OR LAST POLYGON

3400:         If anIndex = 0 Or anIndex = pArray.Count - 1 Then
3401:           pReturnArray.Add pIntPolygon
3402:         Else
3403:          Set pGeometryBag = pCheckCountPoly.ConnectedComponentBag

3405:          For anIndex2 = 0 To pGeometryBag.GeometryCount - 1
3406:            Set pSubPoly = pGeometryBag.Geometry(anIndex2)

3408:            If Not pSubPoly.IsEmpty Then
3409:              Set pSubPoly.SpatialReference = pSpatialRef
3410:              pReturnArray.Add pSubPoly
3411:            End If
```

```
3412:          Next anIndex2
3413:         End If
3414:       Next anIndex
3415:       pWorked = True

      ' COMBINE ADJACENT POLYGONS
'      MsgBox "Before Combine: " & pReturnArray.Count + 2
3419:       Set pCombinedArrayInfo = CombineAdjacentPolygons(pReturnArray, _
        pStartPolygon, pEndPolygon, pPro, psbar, lngCounter, strTotalCount)
3421:       booAllIntersects = pCombinedArrayInfo.Element(0)
3422:       Set pCombinedArray = pCombinedArrayInfo.Element(1)
'      MsgBox "After Combine: " & pCombinedArray.Count

3425:       Set pReturnArray = pCombinedArray

3427:       If booAllIntersects Then
3428:         strMessage = strMessage + "Continuous strand of patch connecting habitat blocks!  No corridor necessary!..." + vbCrLf
3429:       End If
3430:     End If   ' END CHECKING IF REMAINING ARRAY CONTAINS ANY POLYGONS
3431:   End If  ' END CHECKING IF BOTH START AND END POLYGONS INTERSECT WITH CORRIDOR

3433:   pResponseCollection.Add pWorked
3434:   pResponseCollection.Add strMessage
3435:   pResponseCollection.Add StartIntersects
3436:   pResponseCollection.Add EndIntersects
3437:   pResponseCollection.Add pReturnArray

3439:   pPro.position = 1
3440:   psbar.HideProgressBar

3442:   Set ClipPolysToCorridor = pResponseCollection


  Exit Function
ErrorHandler:
  HandleError True, "ClipPolysToCorridor " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function CombineAdjacentPolygons( _
    pArray As esriSystem.IArray, _
    pStartPolygon As IPolygon, _
    pEndPolygon As IPolygon, _
    pPro As IStepProgressor, _
    psbar As IStatusBar, _
    lngCounter As Long, _
    strTotalCount As String) As esriSystem.IVariantArray
```

```
  On Error GoTo ErrorHandler


  Dim pNewArray As esriSystem.IArray
  Dim lngIndex As Long
  Dim pPolygon As IPolygon
  Dim pRelOp As IRelationalOperator
  Dim pPolygon2 As IPolygon
  Dim pTopoOp As ITopologicalOperator
  Dim booFoundAdjacent As Boolean

3469:   Set pNewArray = New esriSystem.Array

3471:   Do Until pArray.Count = 0

3473:     lngCounter = lngCounter + 1
3474:     pPro.Message = "Checking for Multiple Strands:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
3475:     psbar.StepProgressBar

    ' PULL OUT FIRST POLYGON IN LIST
3478:     Set pPolygon = pArray.Element(0)
3479:     pArray.Remove (0)
3480:     Set pRelOp = pPolygon
3481:     Set pTopoOp = pPolygon
3482:     booFoundAdjacent = True

    ' COMPARE FIRST POLYGON WITH EACH OTHER POLYGON IN LIST.
    ' IF ONE IS FOUND TO BE ADJACENT, UNION THEM AND START OVER
3486:     Do Until booFoundAdjacent = False
3487:       booFoundAdjacent = False

      ' ONLY GO THROUGH REST OF LIST IF THERE ACTUALLY IS MORE TO THE LIST
3490:       If pArray.Count > 0 Then
3491:         For lngIndex = 0 To pArray.Count - 1
3492:           Set pPolygon2 = pArray.Element(lngIndex)
          ' IF SECOND POLYGON INTERSECTS, UNION IT AND START LIST OVER
3494:           If Not pRelOp.Disjoint(pPolygon2) Then
3495:             Set pPolygon = pTopoOp.Union(pPolygon2)
3496:             Set pRelOp = pPolygon
3497:             Set pTopoOp = pPolygon
3498:             booFoundAdjacent = True
3499:             pArray.Remove (lngIndex)
3500:             Exit For
3501:           End If
3502:         Next lngIndex
3503:       End If
3504:     Loop
```

```
3505:     pNewArray.Add pPolygon
3506:   Loop

  ' GET START POLYGON AND COMBINE ANY INTERSECTING POLYGONS
  Dim pUsedIndexArray As esriSystem.ILongArray
3510:   Set pUsedIndexArray = New esriSystem.LongArray
3511:   If pNewArray.Count > 0 Then
3512:     Set pRelOp = pStartPolygon
3513:     Set pTopoOp = pStartPolygon
3514:     For lngIndex = 0 To pNewArray.Count - 1
3515:       Set pPolygon2 = pNewArray.Element(lngIndex)
3516:       If Not pRelOp.Disjoint(pPolygon2) Then
3517:         pUsedIndexArray.Add (lngIndex)
3518:         Set pStartPolygon = pTopoOp.Union(pPolygon2)
3519:         Set pRelOp = pStartPolygon
3520:         Set pTopoOp = pStartPolygon
3521:       End If
3522:     Next lngIndex
3523:   End If

  ' GET END POLYGON AND COMBINE ANY INTERSECTING POLYGONS
3526:   If pNewArray.Count > 0 Then
3527:     Set pRelOp = pEndPolygon
3528:     Set pTopoOp = pEndPolygon
3529:     For lngIndex = 0 To pNewArray.Count - 1
3530:       Set pPolygon2 = pNewArray.Element(lngIndex)
3531:       If Not pRelOp.Disjoint(pPolygon2) Then
3532:         pUsedIndexArray.Add (lngIndex)
3533:         Set pEndPolygon = pTopoOp.Union(pPolygon2)
3534:         Set pRelOp = pEndPolygon
3535:         Set pTopoOp = pEndPolygon
3536:       End If
3537:     Next lngIndex
3538:   End If

  ' CHECK IF START AND END POLYGONS INTERSECT EACH OTHER NOW
  Dim booIntersects As Boolean
3542:   Set pRelOp = pStartPolygon
3543:   booIntersects = Not pRelOp.Disjoint(pEndPolygon)

  ' INSERT START AND END POLYGONS
  Dim pFinalArray As esriSystem.IArray
3547:   Set pFinalArray = New esriSystem.Array
3548:   pFinalArray.Add pStartPolygon
  Dim lngCheckIndex As Long
  Dim booSkipIndex As Boolean
3551:   For lngIndex = 0 To pNewArray.Count - 1
```

```
3552:     booSkipIndex = False
3553:     If pUsedIndexArray.Count > 0 Then
3554:       For lngCheckIndex = 0 To pUsedIndexArray.Count - 1
3555:         If pUsedIndexArray.Element(lngCheckIndex) = lngIndex Then
3556:           booSkipIndex = True
3557:           Exit For
3558:         End If
3559:       Next lngCheckIndex
3560:     End If
3561:     If Not booSkipIndex Then pFinalArray.Add pNewArray.Element(lngIndex)
3562:   Next lngIndex
3563:   pFinalArray.Add pEndPolygon

  Dim pReturnArray As esriSystem.IVariantArray
3566:   Set pReturnArray = New esriSystem.VarArray
3567:   pReturnArray.Add booIntersects
3568:   pReturnArray.Add pFinalArray

3570:   Set CombineAdjacentPolygons = pReturnArray


  Exit Function
ErrorHandler:
  HandleError True, "CombineAdjacentPolygons " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function



Public Function ClipPolysToCorridor_ORIG(ByVal pArray As IArray, pCorPolygon As IPolygon, pApp As IApplication) As Collection
  On Error GoTo ErrorHandler


'      Dim pMxDoc As IMxDocument
'      Set pMxDoc = ThisDocument


  Dim pResponseCollection As New Collection
  Dim StartIntersects As Boolean
  Dim EndIntersects As Boolean
  Dim pReturnArray As IArray
  Dim pWorked As Boolean
  Dim strMessage As String

  Dim pRelOp As IRelationalOperator
  Dim pTopOp As ITopologicalOperator
  Dim pTopOp2 As ITopologicalOperator
```

```
  Dim pTopOp4 As ITopologicalOperator2
  Dim pProxOp As IProximityOperator
  Dim pSpatialRef As ISpatialReference

  Dim anIndex As Long
  Dim pPolygon As IPolygon
  Dim pStartPolygon As IPolygon
  Dim pEndPolygon As IPolygon
  Dim pIntPolygon As IPolygon
  Dim pCheckCountPoly As IPolygon4
  Dim pGeometryBag As IGeometryCollection
  Dim pSubPoly As IPolygon4
  Dim anIndex2 As Long

3612:   Set pReturnArray = New esriSystem.Array
3613:   Set pRelOp = pCorPolygon
3614:   Set pTopOp = pCorPolygon

  Dim pRelOp2 As IRelationalOperator

3618:   pWorked = False
3619:   strMessage = ""

3621:   If pArray.Count = 0 Then
3622:     strMessage = "Array contained no polygons!"
3623:     StartIntersects = False
3624:     EndIntersects = False
3625:   Else
    ' CONVERT ANY ODD SHAPES TO NORMAL POLYGONS BEFORE DOING ANYTHING ELSE
    ' FORCE OBJECT TO BE POLYGON IF THE FIRST SEGMENT IS A CURVE

      ' PROGRESS BAR STUFF
    Dim psbar As IStatusBar
3631:     Set psbar = pApp.StatusBar
    Dim pPro As IStepProgressor
3633:     Set pPro = psbar.ProgressBar
    Dim lngCounter As Long
3635:     lngCounter = 0
    Dim lngTotalCount As Long
3637:     lngTotalCount = (pArray.Count) ^ 2 + (pArray.Count * 2)
    Dim strTotalCount As String
3639:     strTotalCount = CStr(lngTotalCount)
3640:     pPro.position = 1
3641:     psbar.ShowProgressBar "Clipping patch polygons to corridor:  Step 1 of " & strTotalCount & "...", 1, _
            lngTotalCount, 1, True

    Dim booWorkingWithCurves As Boolean
```

```
    Dim pSegmentCollectionCurves As ISegmentCollection
    Dim pSegmentCurve As ISegment
    Dim pGeometryTypeA As esriGeometryType
    Dim pNewPoints As IPointCollection
    Dim pNewPolygon As IPointCollection
    Dim pRecreatedPolygon As IPolygon
    Dim pRevisedArray As esriSystem.IArray
3652:     Set pRevisedArray = New esriSystem.Array

3654:     For anIndex = 0 To pArray.Count - 1

3656:       lngCounter = lngCounter + 1
3657:       pPro.Message = "Clipping patch polygons to corridor:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
3658:       psbar.StepProgressBar

3660:       Set pPolygon = pArray.Element(anIndex)

3662:       Set pSegmentCollectionCurves = pPolygon
3663:       Set pSegmentCurve = pSegmentCollectionCurves.Segment(0)
3664:       pGeometryTypeA = pSegmentCurve.GeometryType

3666:       booWorkingWithCurves = (pGeometryTypeA = esriGeometryBezier3Curve) Or _
        (pGeometryTypeA = esriGeometryCircularArc) Or _
        (pGeometryTypeA = esriGeometryEllipticArc)

3670:       If booWorkingWithCurves Then
3671:         Set pNewPoints = GridFunctions.EllipticArcToPolygon2(pSegmentCollectionCurves, 100)
3672:         Set pNewPolygon = New Polygon
3673:         pNewPolygon.SetPointCollection pNewPoints
3674:         Set pRecreatedPolygon = pNewPolygon
3675:         Set pTopOp4 = pRecreatedPolygon
3676:         pTopOp4.IsKnownSimple = False
3677:         pTopOp4.Simplify
3678:         pRevisedArray.Add pRecreatedPolygon
'         ThisDocument.Graphic_MakeFromGeometry pMxDoc, pNewPolygon, "test_order"
3680:       Else
3681:         pRevisedArray.Add pPolygon
'         ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPolygon, "test_order"
3683:       End If
3684:     Next anIndex
3685:     Set pArray = pRevisedArray
    ' DONE CONVERTING CIRCLES, ELLIPTIC ARCS AND BEZIER CURVES TO POLYGONS

    ' NEXT, COMBINE ADJACENT POLYGONS
    Dim booFoundAdjacent As Boolean
    Dim pCheckAdjacentArray As esriSystem.IArray
3691:         Set pCheckAdjacentArray = New esriSystem.Array
```

```
    Dim pCombineArray As esriSystem.ILongArray
    Dim pNewArray As esriSystem.IArray
    Dim pDoneIndicesArray As esriSystem.ILongArray
    Dim pCombinePolygon As IPolygon
    Dim booIncludesEnd As Boolean
    Dim lngCombineIndex As Long
    Dim booAlreadyBeenCombined As Boolean
    Dim lngCheckDoneIndex As Long
    Dim pEndCombinedPolygon As IPolygon
3701:     Set pEndCombinedPolygon = pArray.Element(pArray.Count - 1)

'    ThisDocument.Graphic_MakeFromGeometry pMxDoc, pEndCombinedPolygon, "test_order"

3705:     booFoundAdjacent = False


'    ThisDocument.DeleteGraphicsByName pMxDoc, "test_order"
'    For anIndex = 0 To pArray.Count - 1
'      Set pPolygon = pArray.Element(anIndex)
'      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPolygon, "test_order"
'    Next anIndex

3714:     For anIndex = 0 To pArray.Count - 1
3715:       Set pCombineArray = New LongArray
'      Set pProxOp = pArray.Element(anIndex)
3717:       Set pRelOp2 = pArray.Element(anIndex)
3718:       For anIndex2 = 0 To pArray.Count - 1

3720:         lngCounter = lngCounter + 1
3721:         pPro.Message = "Clipping patch polygons to corridor:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
3722:         psbar.StepProgressBar

3724:         Set pPolygon = pArray.Element(anIndex2)
3725:         If Not pRelOp2.Disjoint(pPolygon) Then
'        If pProxOp.ReturnDistance(pPolygon) = 0 Then
3727:           pCombineArray.Add (anIndex2)    ' BUILD LIST OF ALL POLYGON INDICES THAT INTERSECT CURRENT POLYGON
3728:           booFoundAdjacent = True
3729:         End If
3730:       Next anIndex2
3731:       pCheckAdjacentArray.Add pCombineArray
3732:     Next anIndex

'    ThisDocument.DeleteGraphicsByName pMxDoc, "test_order"
'    For anIndex = 0 To pArray.Count - 1
'      Set pPolygon = pArray.Element(anIndex)
'      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPolygon, "test_order"
'    Next anIndex
```

```
3740:     booFoundAdjacent = False
'    MyGeneralOperations.DeleteGraphicsByName pMxDoc, "test_order"
3742:     For anIndex = 0 To pArray.Count - 1
3743:       Set pCombineArray = pCheckAdjacentArray.Element(anIndex)
3744:       If pCombineArray.Count > 1 Then
3745:         booFoundAdjacent = True
3746:         Exit For
3747:       End If
3748:     Next anIndex

3750:     If booFoundAdjacent Then
3751:       Set pNewArray = New esriSystem.Array
3752:       Set pDoneIndicesArray = New LongArray
3753:       For anIndex = 0 To pCheckAdjacentArray.Count - 1

3755:         Set pCombineArray = pCheckAdjacentArray.Element(anIndex)

        ' CHECK IF THIS INDEX HAS ALREADY BEEN DONE
3758:         booAlreadyBeenCombined = False
3759:         If pDoneIndicesArray.Count > 0 Then
3760:           For lngCheckDoneIndex = 0 To pDoneIndicesArray.Count - 1
3761:             If pDoneIndicesArray.Element(lngCheckDoneIndex) = anIndex Then
3762:               booAlreadyBeenCombined = True
3763:               Exit For
3764:             End If
3765:           Next lngCheckDoneIndex
3766:         End If

3768:         If Not booAlreadyBeenCombined Then
3769:           lngCombineIndex = pCombineArray.Element(0)
3770:           Set pCombinePolygon = pArray.Element(lngCombineIndex)

3772:           booIncludesEnd = (anIndex = pCheckAdjacentArray.Count - 1)

 '         Set pCombineArray = pCheckAdjacentArray.Element(anIndex)
3775:           If pCombineArray.Count > 1 Then

3777:             booIncludesEnd = (pCombineArray.Element(pCombineArray.Count - 1) = (pArray.Count - 1))

3779:             Set pTopOp2 = pCombinePolygon
3780:             For anIndex2 = 1 To pCombineArray.Count - 1
3781:               lngCombineIndex = pCombineArray.Element(anIndex2)
3782:               Set pCombinePolygon = pTopOp2.Union(pArray.Element(lngCombineIndex))
3783:               pDoneIndicesArray.Add (lngCombineIndex)
3784:             Next anIndex2
'           pDoneIndicesArray.Add (anIndex)
```

```
3786:             If booIncludesEnd Then
3787:               Set pEndCombinedPolygon = pCombinePolygon
3788:             End If
3789:           End If
3790:           If Not booIncludesEnd Then

3792:             Set pTopOp2 = pCombinePolygon
3793:             pTopOp2.Simplify
3794:             pNewArray.Add pCombinePolygon
'            ThisDocument.Graphic_MakeFromGeometry pMxDoc, pCombinePolygon, "test_order"
3796:           End If
3797:           pDoneIndicesArray.Add (anIndex)
3798:         End If
3799:       Next anIndex

3801:       Set pTopOp2 = pEndCombinedPolygon
3802:       pTopOp2.Simplify

3804:       pNewArray.Add pEndCombinedPolygon

3806:       Set pArray = pNewArray
3807:     End If

'    ThisDocument.DeleteGraphicsByName pMxDoc, "test_order"
'    For anIndex = 0 To pArray.Count - 1
'      Set pPolygon = pArray.Element(anIndex)
'      ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPolygon, "test_order"
'    Next anIndex

    ' DONE COMBINING ADJACENT POLYGONS ------------------------------

3817:     pPro.Message = "Confirming Habitat Block 1 intersects Corridor..."
3818:     Set pStartPolygon = pArray.Element(0)
3819:     StartIntersects = Not pRelOp.Disjoint(pStartPolygon)
3820:     If Not StartIntersects Then strMessage = strMessage + "Origin polygon does not intersect corridor..." + vbCrLf

3822:     pPro.Message = "Confirming Habitat Block 2 intersects Corridor..."
3823:     Set pEndPolygon = pArray.Element(pArray.Count - 1)
3824:     EndIntersects = Not pRelOp.Disjoint(pEndPolygon)
3825:     If Not EndIntersects Then strMessage = strMessage + "Destination polygon does not intersect corridor..." + vbCrLf

3827:     If pArray.Count = 1 Then
3828:       strMessage = strMessage + "Origin polygon not separate from Destination polygon!  No corridor necessary!..." + vbCrLf
3829:     Else
      Dim pRelOp3 As IRelationalOperator
3831:       Set pRelOp3 = pStartPolygon
3832:       If Not pRelOp3.Disjoint(pEndPolygon) Then
```

```
3833:           strMessage = strMessage + "Origin polygon intersects with Destination polygon!  No corridor necessary!..." + vbCrLf
3834:       End If
3835:     End If

3837:     If StartIntersects And EndIntersects Then
3838:       For anIndex = 0 To pArray.Count - 1

3840:         lngCounter = lngCounter + 1
3841:         pPro.Message = "Checking for Multiple Strands:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
3842:         psbar.StepProgressBar

3844:         Set pPolygon = pArray.Element(anIndex)
3845:         Set pSpatialRef = pPolygon.SpatialReference
3846:         Set pIntPolygon = pTopOp.Intersect(pPolygon, esriGeometry2Dimension)

3848:         Set pCheckCountPoly = pIntPolygon
       ' SPLIT INTO MULTIPLE POLYGONS IF NECESSARY; ONLY IF NOT FIRST OR LAST POLYGON
 '       Set pTopOp2 = pPolygon

3852:         If anIndex = 0 Or anIndex = pArray.Count - 1 Then
3853:           pReturnArray.Add pIntPolygon
3854:         Else
3855:          Set pGeometryBag = pCheckCountPoly.ConnectedComponentBag
3856:          For anIndex2 = 0 To pGeometryBag.GeometryCount - 1
3857:            Set pSubPoly = pGeometryBag.Geometry(anIndex2)

3859:            If Not pSubPoly.IsEmpty Then
3860:              Set pSubPoly.SpatialReference = pSpatialRef
3861:              pReturnArray.Add pSubPoly
3862:            End If
3863:          Next anIndex2
3864:         End If
3865:       Next anIndex
3866:       pWorked = True
3867:     End If
3868:   End If

3870:   pResponseCollection.Add pWorked
3871:   pResponseCollection.Add strMessage
3872:   pResponseCollection.Add StartIntersects
3873:   pResponseCollection.Add EndIntersects
3874:   pResponseCollection.Add pReturnArray

3876:   pPro.position = 1
3877:   psbar.HideProgressBar

3879:   Set ClipPolysToCorridor_ORIG = pResponseCollection
```

```
  Exit Function
ErrorHandler:
  HandleError True, "ClipPolysToCorridor_ORIG " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function CheckIntersectBoundary(pPolyline As IPolyline, pCorPolygon As IPolygon) As Boolean
  On Error GoTo ErrorHandler


  Dim pBoundaryLine As IPolyline
  Dim pTopoOp As ITopologicalOperator
  Dim pRelOp As IRelationalOperator
3894:   Set pTopoOp = pCorPolygon

3896:   Set pBoundaryLine = pTopoOp.Boundary

3898:   Set pRelOp = pBoundaryLine
3899:   CheckIntersectBoundary = Not pRelOp.Disjoint(pPolyline)


  Exit Function
ErrorHandler:
  HandleError True, "CheckIntersectBoundary " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function GenerateCorridorRaster(pCorPolygon As IPolygon, ByRef pEnv As IRasterAnalysisEnvironment) As IRaster
  On Error GoTo erh

    Dim pEnvelope As IEnvelope
'  Set pEnvelope = pStartPolygon.Envelope
'  pEnvelope.Union pEndPolygon.Envelope

3914:   Set pEnvelope = pCorPolygon.Envelope
3915:   pEnvelope.Expand 1.05, 1.05, True

  ' CREATE A RASTER ENVIRONMENT OBJECT
  Dim dblCellSize As Double
  Dim dblWidth As Double
  Dim dblHeight As Double
3921:   dblWidth = pEnvelope.Width
3922:   dblHeight = pEnvelope.Height
3923:   If dblWidth > dblHeight Then
3924:     dblCellSize = dblWidth / 600
```

```
3925:   Else
3926:     dblCellSize = dblHeight / 600
3927:   End If

  'Create a RasterMakerOp operator
  Dim pRasMakerOp As IRasterMakerOp
3931:   Set pRasMakerOp = New RasterMakerOp

  'Create an analysis environment object
'  Dim pEnv As IRasterAnalysisEnvironment
3935:   Set pEnv = pRasMakerOp

  'Set cellsize
3938:   pEnv.SetCellSize esriRasterEnvValue, dblCellSize

  'Set output extent
3941:   pEnv.SetExtent esriRasterEnvValue, pEnvelope

  'Set output spatial reference
  Dim pSpRef As ISpatialReference
3945:   Set pSpRef = pCorPolygon.SpatialReference
3946:   Set pEnv.OutSpatialReference = pSpRef

  'Create a constant raster
  Dim pBaseRaster As IRaster
3950:   Set pBaseRaster = pRasMakerOp.MakeConstant(1, True)

3952:   DoEvents

3954:   GridFunctions.ReturnCellSize pBaseRaster

  ' CLIP TO CORRIDOR POLYGON
  Dim pCorRaster As IRaster
3958:   Set pCorRaster = GridFunctions.ClipRasterToPolygon(pBaseRaster, pCorPolygon, True, pEnvelope, dblCellSize)

3960:   Set GenerateCorridorRaster = pCorRaster
  Exit Function
erh:
3963:   MsgBox "Failed in GenerateCorridorRaster: " & Err.Description
End Function



Public Function AttachPolylineToPatches(pPolyline As IPolyline, pStartPatch As IPolygon, pEndPatch As IPolygon) As IPolyline
  On Error GoTo ErrorHandler
```

```
  Dim pTopoOp As ITopologicalOperator
3973:   Set pTopoOp = pPolyline

  Dim pNewPolyline As IPolyline
3976:   Set pNewPolyline = pTopoOp.Difference(pStartPatch)

3978:   Set pTopoOp = pNewPolyline
3979:   Set pNewPolyline = pTopoOp.Difference(pEndPatch)

  Dim pGeoCollection As IGeometryCollection
3982:   Set pGeoCollection = pNewPolyline

  ' IF TRIMMING OFF THE EDGES RESULTED IN A MULTIPART POLYLINE, THEN IT IS TOO DIFFICULT TO FIGURE OUT THE
  ' BEST WAY TO ATTACH THIS POLYLINE.  JUST RETURN THE ORIGINAL POLYLINE

3987:   If pGeoCollection.GeometryCount > 1 Or pNewPolyline.IsEmpty Then
3988:     Set AttachPolylineToPatches = pPolyline
    Exit Function
3990:   End If

  Dim pPointStart As IPoint
3993:       Set pPointStart = New Point
  Dim pPointEnd As IPoint
3995:       Set pPointEnd = New Point

3997:   pNewPolyline.QueryPoint esriNoExtension, 0, True, pPointStart
3998:   pNewPolyline.QueryPoint esriNoExtension, 100, True, pPointEnd

  ' FLIP IF NECESSARY
  Dim pProxOp As IProximityOperator
4002:   Set pProxOp = pStartPatch
  Dim booShouldFlip As Boolean
4004:   booShouldFlip = False
4005:   If pProxOp.ReturnDistance(pPointStart) > pProxOp.ReturnDistance(pPointEnd) Then
    Dim pTempPoint As IPoint
4007:     Set pTempPoint = pPointStart
4008:     Set pPointStart = pPointEnd
4009:     Set pPointEnd = pTempPoint
4010:     booShouldFlip = True
'    pNewPolyline.ReverseOrientation
4012:   End If

  ' FIND NEAREST POINT ON START PATCH TO START POINT
  Dim pArray As esriSystem.IArray
4016:   Set pProxOp = pNewPolyline
  Dim pEndPatchPoint As IPoint
  Dim pStartPatchPoint As IPoint
```

```
  Dim pPointCollection As IPointCollection
4020:   Set pPointCollection = pNewPolyline

4022:   If booShouldFlip Then
4023:     If pProxOp.ReturnDistance(pStartPatch) > 0 Then
4024:       Set pArray = MyGeometricOperations.CalcClosestPoints(pPointEnd, pStartPatch)
4025:       Set pStartPatchPoint = pArray.Element(2)
4026:       pPointCollection.AddPoint pStartPatchPoint, 0
4027:     End If
    ' FIND NEAREST POINT ON END PATCH TO END POINT
4029:     If pProxOp.ReturnDistance(pEndPatch) > 0 Then
4030:       Set pArray = MyGeometricOperations.CalcClosestPoints(pPointStart, pEndPatch)
4031:       Set pEndPatchPoint = pArray.Element(2)
4032:       pPointCollection.AddPoint pEndPatchPoint
4033:     End If

4035:   Else
4036:     If pProxOp.ReturnDistance(pStartPatch) > 0 Then
4037:       Set pArray = MyGeometricOperations.CalcClosestPoints(pPointStart, pStartPatch)
4038:       Set pStartPatchPoint = pArray.Element(2)
4039:       pPointCollection.AddPoint pStartPatchPoint, 0
4040:     End If
    ' FIND NEAREST POINT ON END PATCH TO END POINT
4042:     If pProxOp.ReturnDistance(pEndPatch) > 0 Then
4043:       Set pArray = MyGeometricOperations.CalcClosestPoints(pPointEnd, pEndPatch)
4044:       Set pEndPatchPoint = pArray.Element(2)
4045:       pPointCollection.AddPoint pEndPatchPoint
4046:     End If
4047:   End If

4049:   Set pNewPolyline.SpatialReference = pPolyline.SpatialReference

4051:   Set AttachPolylineToPatches = pNewPolyline


  Exit Function
ErrorHandler:
  HandleError True, "AttachPolylineToPatches " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function CleanPolyline(ByVal pPolyline As IPolyline, pBoundary As IPolyline) As IPolyline
  On Error GoTo ErrorHandler


  Dim pPointCol As IPointCollection
```

```
4065:   Set pPointCol = pPolyline

  Dim pNewPointCol As IPointCollection
4068:   Set pNewPointCol = New Polyline

  Dim pSegment As IPointCollection
  Dim pStartPoint As IPoint
  Dim pEndPoint As IPoint
  Dim pClone As IClone

  Dim pCount As Long
4076:   pCount = pPointCol.PointCount

4078:   Set pStartPoint = pPointCol.Point(0)
4079:   Set pClone = pStartPoint
4080:   pNewPointCol.AddPoint pClone.Clone

  Dim anIndex1 As Long
  Dim anIndex2 As Long
  Dim anIndexStep As Long

  Dim pRelOp As IRelationalOperator

4088:   anIndex1 = 1
4089:   anIndexStep = 0

4091:   Do Until anIndex1 = pPointCol.PointCount - 1
4092:     Set pEndPoint = pPointCol.Point(anIndex1)
4093:     Set pSegment = New Polyline
4094:     pSegment.AddPoint pStartPoint
4095:     pSegment.AddPoint pEndPoint
4096:     Set pRelOp = pSegment
4097:     If pRelOp.Disjoint(pBoundary) Then
4098:       anIndex1 = anIndex1 + 1
4099:     Else
4100:       anIndex2 = anIndex1 - 1
4101:       Do Until anIndex2 <= anIndexStep
4102:         Set pStartPoint = pPointCol.Point(anIndex2)
4103:         Set pSegment = New Polyline
4104:         pSegment.AddPoint pStartPoint
4105:         pSegment.AddPoint pEndPoint
4106:         Set pRelOp = pSegment
4107:         If pRelOp.Disjoint(pBoundary) Then
4108:           anIndex2 = anIndex2 - 1
4109:         Else
4110:           Set pClone = pStartPoint
4111:           pNewPointCol.AddPoint pClone.Clone
```

```
4112:           anIndexStep = anIndex2 + 1
4113:           Set pStartPoint = pPointCol.Point(anIndexStep)
4114:           Set pClone = pStartPoint
4115:           pNewPointCol.AddPoint pClone.Clone
4116:           anIndex1 = anIndexStep + 1
4117:           Exit Do
4118:         End If
4119:       Loop
4120:     End If
4121:   Loop
4122:   Set pClone = pPointCol.Point(pPointCol.PointCount - 1)
4123:   pNewPointCol.AddPoint pClone.Clone
4124:   Set CleanPolyline = pNewPointCol
4125:   Set CleanPolyline.SpatialReference = pPolyline.SpatialReference


  Exit Function
ErrorHandler:
  HandleError True, "CleanPolyline " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function BailOutOfProgress(frmProgressDialog As Object, pExtensionConfig As IExtensionConfig) As Boolean
  On Error GoTo ErrorHandler


  Dim pExt As Linkages.Extension
4139:   Set pExt = pExtensionConfig

  Dim frmProgress As Linkages.frmJenProgressPercent
4142:   Set frmProgress = frmProgressDialog

4144:   pExt.ProgressDialogAutoClose = (frmProgress.chkClose.Value = 1)
4145:   pExt.ProgressDialogSetExpanded = (frmProgress.SetExpanded)

'  MsgBox "Checking Autoclose Stuff: " & vbCrLf & _
      "Check-Box checked? " & CStr(frmProgress.chkClose.Value = 1) & vbCrLf & _
      "Extension Property Checked? " & CStr(pExt.ProgressDialogAutoClose)


  ' PROGRESS METER STUFF ------------------------------------------------------
4153:   BailOutOfProgress = Not frmProgress.ShouldContinue
4154:   If BailOutOfProgress Then
    Dim theDateString As String
    Dim theElapsedTimeString As String
```

```
4158:     theDateString = Format(Now, "long date") & ";  " & Format(Now, "long time")
4159:     Screen.MousePointer = vbDefault
    Dim theTimeBegan As Date
    Dim theTimeEnd As Date
4162:     theTimeBegan = frmProgress.ProgBeginTime
4163:     theTimeEnd = Now

4165:     theElapsedTimeString = MyGeneralOperations.ReturnTimeElapsed(theTimeBegan, theTimeEnd)

    Dim theReport As String
4168:     theReport = "Operation cancelled prematurely at " & theDateString & vbCrLf & _
        "-----------------------------------------------------" & vbCrLf & theElapsedTimeString

    Dim strCurrentReport As String
4172:     strCurrentReport = frmProgress.txtDetails.Text
4173:     frmProgress.txtDetails.Text = strCurrentReport & vbCrLf & theReport

4175:     If frmProgress.chkClose.Value = 1 Then
    '  Unload frmProgress
    '  Set frmProgress = Nothing
4178:       frmProgress.Frame.Visible = False
4179:     End If

    Exit Function
4182:   End If


  Exit Function
ErrorHandler:
  HandleError True, "BailOutOfProgress " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
'
'
'
'
'Screen.MousePointer = vbHourglass
'
'
'
'
'  ' PROGRESS METER STUFF ------------------------------------------------------------------
'  Dim frmProgress As New frmJenProgressPercent
'  frmProgress.ShouldContinue = True
'  frmProgress.ProgBeginTime = Now
'  frmProgress.ProgRecCount = theFinalList.Count - 2
'  frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
```

```
'  frmProgress.lblBeginTime.Caption = "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd")
'
'  theDetailedDescription = "Pass 2:  Generating Hexagons..." & vbCrLf & _
'      "Began Job: " & Format(theTimeBegan, "ttttt,  ddddd") & vbCrLf & _
'      "-----------------------------------------" & vbCrLf
'  frmProgress.txtDetails.Text = theDetailedDescription
'
'  frmProgress.Frame.Caption = "Current Status:"
'  frmProgress.Frame.Visible = True
'
'  theProgressTimeCheck = CDate(50000)
'  Dim theDescription As String
'  theDescription = "Pass 2:  Generating Hexagons..."
'  ' PROGRESS METER STUFF ------------------------------------------------------------------
'
'
'
'
'
'
'
'
'
'
'    Do While theHorizIndex < theMiddleCount
'
'      ' PROGRESS METER STUFF ------------------------------------------------------------
'      If frmProgress.ShouldContinue = False Then
'        theDateString = Format(Now, "long date") & ";  " & Format(Now, "long time")
'        Screen.MousePointer = vbDefault
'
'        theTimeEnd = Now
'
'        theElapsedTimeString = MyGeneralOperations.ReturnTimeElapsed(theTimeBegan, theTimeEnd)
'
'        theReport = theReport & "Operation cancelled prematurely at " & theDateString & vbCrLf & _
'            "-----------------------------------------------------------" & vbCrLf & theElapsedTimeString
'        Unload frmProgress
'        Set frmProgress = Nothing
'
'        CreateHexagonShapefile = theReport
'        Exit Function
'      End If
'
'      If Abs(DateDiff("s", theProgressTimeCheck, Now)) >= 1 Then    ' UPDATE STUFF AFTER EVERY SECOND
'        theProgressTimeCheck = Now
'        theProgressSpecific = "Working on Row #" & aml_func_mod.InsertCommas(theCounter)
```

```
'        theDetailedDescription = theDetailedDescription & _
'          " --> " & theProgressSpecific & vbCrLf & _
'          "          " & aml_func_mod.InsertCommas(theHexagonCount) & " preliminary hexagons created so far..." & vbCrLf & _
'          "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf
'        frmProgress.txtDetails.Text = theDetailedDescription
'        frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
'
'        frmProgress.Est_Time_Left theCounter, "Generating Hexagons...", "Working on Row #"
'        frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
'
'      End If
'      ' END PROGRESS METER STUFF ----------------------------------------------------
'
'
'
'
'
'
'
'
'
'  theProgressTimeCheck = Now
'  theProgressSpecific = "Saving Final Shapefile..."
'  theDetailedDescription = theDetailedDescription & _
'    " --> " & theProgressSpecific & vbCrLf & _
'    "          " & aml_func_mod.InsertCommas(theHexagonCount) & " preliminary hexagons created so far..." & vbCrLf & _
'    "          [time stamp " & Format(Now, "ttttt,  ddddd") & "]" & vbCrLf
'  frmProgress.txtDetails.Text = theDetailedDescription
'  frmProgress.txtDetails.SelStart = Len(theDetailedDescription)
'
'  frmProgress.Est_Time_Left theCounter, "Saving Shapefile...", "Working on Row #"
'  frmProgress.lblCurrentTime.Caption = Format(Now, "ttttt")
'
'  pFeatCur.Flush
'
'  Screen.MousePointer = vbDefault
'
'  Unload frmProgress
'  Set frmProgress = Nothing
'  theTimeEnd = Now
'
'  strTimeString = MyGeneralOperations.ReturnTimeElapsed(theTimeBegan, theTimeEnd)
'
```

Module 3: CorridorSampleData

```
Attribute VB_Name = "CorridorSampleData"
Option Explicit
```

```
Private Sub AdData(dblX As Double, dblY As Double, dblWidth As Double, dblDistance As Double, pArray As esriSystem.IVariantArray)
  Dim pDblArray As esriSystem.IDoubleArray
5:   Set pDblArray = New esriSystem.DoubleArray
6:   pDblArray.Add dblX
7:   pDblArray.Add dblY
8:   pDblArray.Add dblWidth
9:   pDblArray.Add dblDistance
10:   pArray.Add pDblArray
End Sub

Public Function MakeData() As esriSystem.IVariantArray
  Dim pDataArray As esriSystem.IVariantArray
15:   Set pDataArray = New esriSystem.VarArray
16:   MakeData2 pDataArray
17:   MakeData3 pDataArray
18:   MakeData4 pDataArray
19:   MakeData5 pDataArray
20:   MakeData6 pDataArray
21:   MakeData7 pDataArray

23:   Set MakeData = pDataArray

End Function

Private Sub MakeData2(pDataArray As esriSystem.IVariantArray)
28:   AdData 494226, 3489889, 641.783, 0, pDataArray
29:   AdData 494250, 3489889, 641.783, 24.5128, pDataArray
30:   AdData 494275, 3489889, 641.783, 49.0256, pDataArray
31:   AdData 494299, 3489889, 641.783, 73.5384, pDataArray
32:   AdData 494324, 3489889, 641.783, 98.0512, pDataArray
33:   AdData 494348, 3489889, 641.783, 122.564, pDataArray
34:   AdData 494373, 3489889, 641.783, 147.077, pDataArray
35:   AdData 494397, 3489889, 641.783, 171.59, pDataArray
36:   AdData 494422, 3489889, 641.783, 196.102, pDataArray
37:   AdData 494446, 3489889, 641.783, 220.615, pDataArray
38:   AdData 494471, 3489889, 641.783, 245.128, pDataArray
39:   AdData 494495, 3489889, 641.783, 269.641, pDataArray
40:   AdData 494520, 3489889, 641.783, 294.154, pDataArray
41:   AdData 494544, 3489889, 641.783, 318.667, pDataArray
42:   AdData 494569, 3489889, 641.783, 343.179, pDataArray
43:   AdData 494593, 3489889, 641.783, 367.692, pDataArray
44:   AdData 494618, 3489889, 641.783, 392.205, pDataArray
45:   AdData 494643, 3489889, 641.783, 416.718, pDataArray
46:   AdData 494667, 3489889, 641.783, 441.231, pDataArray
47:   AdData 494692, 3489889, 641.783, 465.743, pDataArray
48:   AdData 494716, 3489889, 641.783, 490.256, pDataArray
```

```
49:   AdData 494741, 3489889, 641.783, 514.769, pDataArray
50:   AdData 494765, 3489889, 641.783, 539.282, pDataArray
51:   AdData 494790, 3489889, 641.783, 563.795, pDataArray
52:   AdData 494814, 3489889, 641.783, 588.307, pDataArray
53:   AdData 494839, 3489889, 642.041, 612.82, pDataArray
54:   AdData 494863, 3489889, 642.762, 637.333, pDataArray
55:   AdData 494888, 3489889, 644.655, 661.846, pDataArray
56:   AdData 494912, 3489889, 647.201, 686.359, pDataArray
57:   AdData 494937, 3489889, 650.766, 710.872, pDataArray
58:   AdData 494961, 3489889, 655.347, 735.384, pDataArray
59:   AdData 494986, 3489889, 660.552, 759.897, pDataArray
60:   AdData 495010, 3489889, 666.822, 784.41, pDataArray
61:   AdData 495035, 3489889, 671.684, 808.923, pDataArray
62:   AdData 495059, 3489889, 674.1, 833.436, pDataArray
63:   AdData 495084, 3489889, 663.699, 857.948, pDataArray
64:   AdData 495108, 3489889, 653.979, 882.461, pDataArray
65:   AdData 495132, 3489890, 644.839, 906.974, pDataArray
66:   AdData 495150, 3489907, 640.714, 931.487, pDataArray
67:   AdData 495167, 3489924, 642.229, 956, pDataArray
68:   AdData 495191, 3489927, 644.744, 980.512, pDataArray
69:   AdData 495215, 3489927, 647.349, 1005.03, pDataArray
70:   AdData 495240, 3489927, 650.914, 1029.54, pDataArray
71:   AdData 495264, 3489927, 648.92, 1054.05, pDataArray
72:   AdData 495289, 3489927, 645.693, 1078.56, pDataArray
73:   AdData 495313, 3489927, 643.539, 1103.08, pDataArray
74:   AdData 495338, 3489927, 642.388, 1127.59, pDataArray
75:   AdData 495362, 3489927, 641.783, 1152.1, pDataArray
76:   AdData 495387, 3489927, 641.783, 1176.61, pDataArray
77:   AdData 495411, 3489927, 638.186, 1201.13, pDataArray
78:   AdData 495436, 3489927, 632.312, 1225.64, pDataArray
79:   AdData 495460, 3489927, 625.678, 1250.15, pDataArray
80:   AdData 495481, 3489936, 617.899, 1274.67, pDataArray
81:   AdData 495498, 3489953, 611.377, 1299.18, pDataArray
82:   AdData 495518, 3489964, 607.906, 1323.69, pDataArray
83:   AdData 495543, 3489964, 605.619, 1348.2, pDataArray
84:   AdData 495567, 3489964, 596.944, 1372.72, pDataArray
85:   AdData 495592, 3489964, 584.509, 1397.23, pDataArray
86:   AdData 495616, 3489964, 566.016, 1421.74, pDataArray
87:   AdData 495641, 3489964, 548.077, 1446.26, pDataArray
88:   AdData 495665, 3489964, 530.505, 1470.77, pDataArray
89:   AdData 495690, 3489964, 513.628, 1495.28, pDataArray
90:   AdData 495714, 3489964, 496.148, 1519.79, pDataArray
91:   AdData 495739, 3489964, 478.476, 1544.31, pDataArray
92:   AdData 495763, 3489964, 461.652, 1568.82, pDataArray
93:   AdData 495788, 3489964, 445.489, 1593.33, pDataArray
94:   AdData 495812, 3489964, 429.859, 1617.85, pDataArray
95:   AdData 495837, 3489964, 415.517, 1642.36, pDataArray
```

```
96:   AdData 495861, 3489964, 401.96, 1666.87, pDataArray
97:   AdData 495886, 3489964, 389.188, 1691.38, pDataArray
98:   AdData 495910, 3489964, 378.164, 1715.9, pDataArray
99:   AdData 495935, 3489964, 367.541, 1740.41, pDataArray
100:   AdData 495959, 3489964, 357.371, 1764.92, pDataArray
101:   AdData 495984, 3489964, 345.587, 1789.44, pDataArray
102:   AdData 496008, 3489964, 334.328, 1813.95, pDataArray
103:   AdData 496033, 3489964, 324.517, 1838.46, pDataArray
104:   AdData 496057, 3489964, 316.714, 1862.97, pDataArray
105:   AdData 496082, 3489964, 310.139, 1887.49, pDataArray
106:   AdData 496107, 3489964, 305.63, 1912, pDataArray
107:   AdData 496131, 3489964, 296.49, 1936.51, pDataArray
108:   AdData 496156, 3489964, 286.033, 1961.02, pDataArray
109:   AdData 496180, 3489964, 277.805, 1985.54, pDataArray
110:   AdData 496205, 3489964, 271.561, 2010.05, pDataArray
111:   AdData 496229, 3489964, 266.773, 2034.56, pDataArray
112:   AdData 496254, 3489964, 265.031, 2059.08, pDataArray
113:   AdData 496278, 3489964, 264.264, 2083.59, pDataArray
114:   AdData 496303, 3489964, 264.264, 2108.1, pDataArray
115:   AdData 496327, 3489964, 264.264, 2132.61, pDataArray
116:   AdData 496352, 3489964, 265.098, 2157.13, pDataArray
117:   AdData 496376, 3489964, 266.84, 2181.64, pDataArray
118:   AdData 496401, 3489964, 271.758, 2206.15, pDataArray
119:   AdData 496425, 3489964, 278.121, 2230.67, pDataArray
120:   AdData 496450, 3489964, 286.349, 2255.18, pDataArray
121:   AdData 496474, 3489964, 295.599, 2279.69, pDataArray
122:   AdData 496499, 3489964, 302.015, 2304.2, pDataArray
123:   AdData 496523, 3489964, 302.015, 2328.72, pDataArray
124:   AdData 496542, 3489951, 302.712, 2353.23, pDataArray
125:   AdData 496559, 3489934, 313.161, 2377.74, pDataArray
126:   AdData 496581, 3489927, 327.914, 2402.26, pDataArray
127:   AdData 496606, 3489927, 339.767, 2426.77, pDataArray
128:   AdData 496630, 3489927, 339.767, 2451.28, pDataArray
129:   AdData 496655, 3489927, 339.767, 2475.79, pDataArray
End Sub
Private Sub MakeData3(pDataArray As esriSystem.IVariantArray)
132:   AdData 496679, 3489927, 339.767, 2500.31, pDataArray
133:   AdData 496697, 3489910, 342.566, 2524.82, pDataArray
134:   AdData 496714, 3489892, 353.139, 2549.33, pDataArray
135:   AdData 496737, 3489889, 367.284, 2573.85, pDataArray
136:   AdData 496762, 3489889, 376.548, 2598.36, pDataArray
137:   AdData 496786, 3489889, 372.843, 2622.87, pDataArray
138:   AdData 496805, 3489876, 371.663, 2647.38, pDataArray
139:   AdData 496823, 3489859, 383.828, 2671.9, pDataArray
140:   AdData 496840, 3489842, 386.936, 2696.41, pDataArray
141:   AdData 496857, 3489824, 389.319, 2720.92, pDataArray
142:   AdData 496875, 3489807, 394.124, 2745.43, pDataArray
```

```
143:   AdData 496892, 3489790, 389.297, 2769.95, pDataArray
144:   AdData 496909, 3489772, 398.135, 2794.46, pDataArray
145:   AdData 496927, 3489755, 391.339, 2818.97, pDataArray
146:   AdData 496944, 3489738, 401.343, 2843.49, pDataArray
147:   AdData 496969, 3489738, 404.78, 2868, pDataArray
148:   AdData 496990, 3489730, 399.332, 2892.51, pDataArray
149:   AdData 497007, 3489712, 392.253, 2917.02, pDataArray
150:   AdData 497027, 3489700, 396.504, 2941.54, pDataArray
151:   AdData 497051, 3489700, 404.594, 2966.05, pDataArray
152:   AdData 497076, 3489700, 400.518, 2990.56, pDataArray
153:   AdData 497099, 3489697, 392.536, 3015.08, pDataArray
154:   AdData 497116, 3489679, 389.627, 3039.59, pDataArray
155:   AdData 497134, 3489662, 394.415, 3064.1, pDataArray
156:   AdData 497158, 3489662, 402.505, 3088.61, pDataArray
157:   AdData 497183, 3489662, 409.38, 3113.13, pDataArray
158:   AdData 497207, 3489662, 415.01, 3137.64, pDataArray
159:   AdData 497232, 3489662, 415.271, 3162.15, pDataArray
160:   AdData 497256, 3489662, 415.271, 3186.67, pDataArray
161:   AdData 497281, 3489662, 415.271, 3211.18, pDataArray
162:   AdData 497305, 3489662, 415.271, 3235.69, pDataArray
163:   AdData 497330, 3489662, 415.271, 3260.2, pDataArray
164:   AdData 497354, 3489662, 415.271, 3284.72, pDataArray
165:   AdData 497379, 3489662, 416.153, 3309.23, pDataArray
166:   AdData 497403, 3489662, 417.821, 3333.74, pDataArray
167:   AdData 497428, 3489662, 421.13, 3358.26, pDataArray
168:   AdData 497447, 3489650, 411.943, 3382.77, pDataArray
169:   AdData 497464, 3489633, 406.04, 3407.28, pDataArray
170:   AdData 497482, 3489615, 395.069, 3431.79, pDataArray
171:   AdData 497499, 3489598, 385.966, 3456.31, pDataArray
172:   AdData 497516, 3489581, 379.583, 3480.82, pDataArray
173:   AdData 497534, 3489563, 367.322, 3505.33, pDataArray
174:   AdData 497551, 3489546, 367.284, 3529.84, pDataArray
175:   AdData 497568, 3489529, 350.473, 3554.36, pDataArray
176:   AdData 497586, 3489511, 348.056, 3578.87, pDataArray
177:   AdData 497610, 3489511, 348.056, 3603.38, pDataArray
178:   AdData 497631, 3489503, 338.71, 3627.9, pDataArray
179:   AdData 497649, 3489486, 331.626, 3652.41, pDataArray
180:   AdData 497666, 3489469, 331.724, 3676.92, pDataArray
181:   AdData 497683, 3489451, 320.93, 3701.43, pDataArray
182:   AdData 497702, 3489436, 324.754, 3725.95, pDataArray
183:   AdData 497726, 3489436, 324.754, 3750.46, pDataArray
184:   AdData 497747, 3489426, 320.858, 3774.97, pDataArray
185:   AdData 497764, 3489409, 326.4, 3799.49, pDataArray
186:   AdData 497781, 3489391, 333.254, 3824, pDataArray
187:   AdData 497799, 3489374, 334.373, 3848.51, pDataArray
188:   AdData 497817, 3489360, 348.056, 3873.02, pDataArray
189:   AdData 497842, 3489360, 348.056, 3897.54, pDataArray
```

```
190:   AdData 497862, 3489349, 348.977, 3922.05, pDataArray
191:   AdData 497879, 3489331, 360.891, 3946.56, pDataArray
192:   AdData 497900, 3489323, 373.631, 3971.08, pDataArray
193:   AdData 497924, 3489323, 377.336, 3995.59, pDataArray
194:   AdData 497949, 3489323, 377.519, 4020.1, pDataArray
195:   AdData 497970, 3489316, 378.652, 4044.61, pDataArray
196:   AdData 497988, 3489298, 387.926, 4069.13, pDataArray
197:   AdData 498007, 3489285, 404.444, 4093.64, pDataArray
198:   AdData 498031, 3489285, 417.931, 4118.15, pDataArray
199:   AdData 498056, 3489285, 425.837, 4142.67, pDataArray
200:   AdData 498080, 3489285, 431.569, 4167.18, pDataArray
201:   AdData 498105, 3489285, 438.995, 4191.69, pDataArray
202:   AdData 498129, 3489285, 446.307, 4216.2, pDataArray
203:   AdData 498153, 3489284, 453.139, 4240.72, pDataArray
204:   AdData 498171, 3489266, 457.185, 4265.23, pDataArray
205:   AdData 498188, 3489249, 465.788, 4289.74, pDataArray
206:   AdData 498212, 3489247, 473.098, 4314.25, pDataArray
207:   AdData 498236, 3489247, 480.573, 4338.77, pDataArray
208:   AdData 498261, 3489247, 489.175, 4363.28, pDataArray
209:   AdData 498285, 3489247, 490.775, 4387.79, pDataArray
210:   AdData 498310, 3489247, 490.775, 4412.31, pDataArray
211:   AdData 498334, 3489247, 490.775, 4436.82, pDataArray
212:   AdData 498359, 3489247, 490.775, 4461.33, pDataArray
213:   AdData 498383, 3489247, 490.775, 4485.84, pDataArray
214:   AdData 498408, 3489247, 490.775, 4510.36, pDataArray
215:   AdData 498432, 3489247, 491.39, 4534.87, pDataArray
216:   AdData 498457, 3489247, 492.544, 4559.38, pDataArray
217:   AdData 498481, 3489247, 495.351, 4583.9, pDataArray
218:   AdData 498506, 3489247, 499.202, 4608.41, pDataArray
219:   AdData 498530, 3489247, 503.886, 4632.92, pDataArray
220:   AdData 498555, 3489247, 510.254, 4657.43, pDataArray
221:   AdData 498579, 3489247, 517.434, 4681.95, pDataArray
222:   AdData 498604, 3489247, 525.448, 4706.46, pDataArray
223:   AdData 498628, 3489247, 527.497, 4730.97, pDataArray
224:   AdData 498653, 3489247, 528.527, 4755.49, pDataArray
End Sub
Private Sub MakeData4(pDataArray As esriSystem.IVariantArray)
227:   AdData 498677, 3489247, 528.527, 4780, pDataArray
228:   AdData 498702, 3489247, 528.527, 4804.51, pDataArray
229:   AdData 498726, 3489247, 528.527, 4829.02, pDataArray
230:   AdData 498751, 3489247, 528.527, 4853.54, pDataArray
231:   AdData 498775, 3489247, 528.527, 4878.05, pDataArray
232:   AdData 498800, 3489247, 528.527, 4902.56, pDataArray
233:   AdData 498824, 3489247, 528.527, 4927.08, pDataArray
234:   AdData 498849, 3489247, 528.527, 4951.59, pDataArray
235:   AdData 498874, 3489247, 528.527, 4976.1, pDataArray
236:   AdData 498898, 3489247, 528.527, 5000.61, pDataArray
```

```
237:   AdData 498923, 3489247, 528.527, 5025.13, pDataArray
238:   AdData 498947, 3489247, 528.527, 5049.64, pDataArray
239:   AdData 498972, 3489247, 528.527, 5074.15, pDataArray
240:   AdData 498996, 3489247, 528.527, 5098.66, pDataArray
241:   AdData 499021, 3489247, 528.527, 5123.18, pDataArray
242:   AdData 499045, 3489247, 528.527, 5147.69, pDataArray
243:   AdData 499070, 3489247, 528.527, 5172.2, pDataArray
244:   AdData 499094, 3489247, 528.527, 5196.72, pDataArray
245:   AdData 499119, 3489247, 528.527, 5221.23, pDataArray
246:   AdData 499143, 3489247, 528.527, 5245.74, pDataArray
247:   AdData 499168, 3489247, 528.527, 5270.25, pDataArray
248:   AdData 499192, 3489247, 528.527, 5294.77, pDataArray
249:   AdData 499217, 3489247, 528.527, 5319.28, pDataArray
250:   AdData 499241, 3489247, 528.527, 5343.79, pDataArray
251:   AdData 499266, 3489247, 528.527, 5368.31, pDataArray
252:   AdData 499290, 3489247, 528.527, 5392.82, pDataArray
253:   AdData 499315, 3489247, 528.527, 5417.33, pDataArray
254:   AdData 499339, 3489247, 528.527, 5441.84, pDataArray
255:   AdData 499364, 3489247, 528.527, 5466.36, pDataArray
256:   AdData 499388, 3489247, 528.527, 5490.87, pDataArray
257:   AdData 499413, 3489247, 528.527, 5515.38, pDataArray
258:   AdData 499437, 3489247, 528.527, 5539.9, pDataArray
259:   AdData 499462, 3489247, 528.527, 5564.41, pDataArray
260:   AdData 499486, 3489247, 528.527, 5588.92, pDataArray
261:   AdData 499511, 3489247, 528.527, 5613.43, pDataArray
262:   AdData 499535, 3489247, 528.527, 5637.95, pDataArray
263:   AdData 499560, 3489247, 528.527, 5662.46, pDataArray
264:   AdData 499584, 3489247, 528.527, 5686.97, pDataArray
265:   AdData 499609, 3489247, 528.527, 5711.49, pDataArray
266:   AdData 499633, 3489247, 528.527, 5736, pDataArray
267:   AdData 499658, 3489247, 528.527, 5760.51, pDataArray
268:   AdData 499682, 3489247, 528.527, 5785.02, pDataArray
269:   AdData 499707, 3489247, 528.527, 5809.54, pDataArray
270:   AdData 499731, 3489247, 528.527, 5834.05, pDataArray
271:   AdData 499756, 3489247, 528.527, 5858.56, pDataArray
272:   AdData 499780, 3489247, 528.527, 5883.07, pDataArray
273:   AdData 499805, 3489247, 528.527, 5907.59, pDataArray
274:   AdData 499830, 3489247, 528.527, 5932.1, pDataArray
275:   AdData 499854, 3489247, 528.299, 5956.61, pDataArray
276:   AdData 499879, 3489247, 526.543, 5981.13, pDataArray
277:   AdData 499903, 3489247, 521.095, 6005.64, pDataArray
278:   AdData 499928, 3489247, 513.164, 6030.15, pDataArray
279:   AdData 499952, 3489247, 506.795, 6054.66, pDataArray
280:   AdData 499977, 3489247, 501.315, 6079.18, pDataArray
281:   AdData 500001, 3489247, 496.689, 6103.69, pDataArray
282:   AdData 500019, 3489264, 492.552, 6128.2, pDataArray
283:   AdData 500036, 3489281, 490.828, 6152.72, pDataArray
```

```
284:   AdData 500059, 3489285, 490.775, 6177.23, pDataArray
285:   AdData 500084, 3489285, 490.775, 6201.74, pDataArray
286:   AdData 500108, 3489285, 490.775, 6226.25, pDataArray
287:   AdData 500133, 3489285, 490.775, 6250.77, pDataArray
288:   AdData 500157, 3489285, 490.775, 6275.28, pDataArray
289:   AdData 500182, 3489285, 490.775, 6299.79, pDataArray
290:   AdData 500206, 3489285, 485.344, 6324.31, pDataArray
291:   AdData 500231, 3489285, 476.902, 6348.82, pDataArray
292:   AdData 500255, 3489285, 470.044, 6373.33, pDataArray
293:   AdData 500280, 3489285, 464.212, 6397.84, pDataArray
294:   AdData 500304, 3489285, 459.238, 6422.36, pDataArray
295:   AdData 500329, 3489285, 456.2, 6446.87, pDataArray
296:   AdData 500353, 3489285, 454.113, 6471.38, pDataArray
297:   AdData 500378, 3489285, 453.094, 6495.9, pDataArray
298:   AdData 500402, 3489285, 453.023, 6520.41, pDataArray
299:   AdData 500427, 3489285, 453.023, 6544.92, pDataArray
300:   AdData 500451, 3489285, 453.023, 6569.43, pDataArray
301:   AdData 500476, 3489285, 453.023, 6593.95, pDataArray
302:   AdData 500500, 3489285, 453.023, 6618.46, pDataArray
303:   AdData 500525, 3489285, 453.023, 6642.97, pDataArray
304:   AdData 500549, 3489285, 453.023, 6667.48, pDataArray
305:   AdData 500574, 3489285, 453.023, 6692, pDataArray
306:   AdData 500598, 3489285, 453.023, 6716.51, pDataArray
307:   AdData 500623, 3489285, 448.026, 6741.02, pDataArray
308:   AdData 500647, 3489285, 440.759, 6765.54, pDataArray
309:   AdData 500672, 3489285, 433.333, 6790.05, pDataArray
310:   AdData 500696, 3489285, 427.126, 6814.56, pDataArray
311:   AdData 500721, 3489285, 421.847, 6839.07, pDataArray
312:   AdData 500745, 3489285, 418.539, 6863.59, pDataArray
313:   AdData 500770, 3489285, 413.41, 6888.1, pDataArray
314:   AdData 500794, 3489285, 406.668, 6912.61, pDataArray
315:   AdData 500819, 3489285, 398.592, 6937.13, pDataArray
316:   AdData 500843, 3489285, 389.286, 6961.64, pDataArray
317:   AdData 500868, 3489285, 378.493, 6986.15, pDataArray
318:   AdData 500892, 3489285, 364.896, 7010.66, pDataArray
319:   AdData 500914, 3489278, 351.993, 7035.18, pDataArray
320:   AdData 500932, 3489261, 344.294, 7059.69, pDataArray
321:   AdData 500950, 3489247, 341.598, 7084.2, pDataArray
322:   AdData 500975, 3489247, 340.24, 7108.72, pDataArray
323:   AdData 500999, 3489247, 333.416, 7133.23, pDataArray
324:   AdData 501023, 3489245, 322.6, 7157.74, pDataArray
325:   AdData 501040, 3489228, 308.055, 7182.25, pDataArray
326:   AdData 501058, 3489211, 302.106, 7206.77, pDataArray
327:   AdData 501082, 3489209, 302.015, 7231.28, pDataArray
328:   AdData 501106, 3489209, 300.186, 7255.79, pDataArray
End Sub
Private Sub MakeData5(pDataArray As esriSystem.IVariantArray)
```

```
331:   AdData 501131, 3489209, 295.534, 7280.31, pDataArray
332:   AdData 501149, 3489195, 279.741, 7304.82, pDataArray
333:   AdData 501167, 3489177, 273.951, 7329.33, pDataArray
334:   AdData 501189, 3489172, 280.395, 7353.84, pDataArray
335:   AdData 501212, 3489169, 284.259, 7378.36, pDataArray
336:   AdData 501230, 3489152, 269.678, 7402.87, pDataArray
337:   AdData 501247, 3489135, 271.771, 7427.38, pDataArray
338:   AdData 501271, 3489134, 272.233, 7451.89, pDataArray
339:   AdData 501293, 3489127, 269.035, 7476.41, pDataArray
340:   AdData 501310, 3489109, 273.183, 7500.92, pDataArray
341:   AdData 501327, 3489092, 283.491, 7525.43, pDataArray
342:   AdData 501345, 3489075, 277.424, 7549.95, pDataArray
343:   AdData 501362, 3489058, 286.904, 7574.46, pDataArray
344:   AdData 501387, 3489058, 276.985, 7598.97, pDataArray
345:   AdData 501408, 3489049, 267.965, 7623.48, pDataArray
346:   AdData 501425, 3489032, 268.39, 7648, pDataArray
347:   AdData 501445, 3489021, 274.838, 7672.51, pDataArray
348:   AdData 501469, 3489021, 274.838, 7697.02, pDataArray
349:   AdData 501488, 3489007, 267.623, 7721.54, pDataArray
350:   AdData 501505, 3488989, 268.85, 7746.05, pDataArray
351:   AdData 501527, 3488983, 281.41, 7770.56, pDataArray
352:   AdData 501552, 3488983, 295.246, 7795.07, pDataArray
353:   AdData 501576, 3488983, 299.898, 7819.59, pDataArray
354:   AdData 501597, 3488973, 301.818, 7844.1, pDataArray
355:   AdData 501614, 3488956, 309.753, 7868.61, pDataArray
356:   AdData 501634, 3488945, 325.46, 7893.13, pDataArray
357:   AdData 501659, 3488945, 339.435, 7917.64, pDataArray
358:   AdData 501683, 3488945, 345.184, 7942.15, pDataArray
359:   AdData 501706, 3488940, 349.652, 7966.66, pDataArray
360:   AdData 501723, 3488923, 359.451, 7991.18, pDataArray
361:   AdData 501741, 3488907, 375.668, 8015.69, pDataArray
362:   AdData 501766, 3488907, 393.578, 8040.2, pDataArray
363:   AdData 501790, 3488907, 406.461, 8064.72, pDataArray
364:   AdData 501815, 3488907, 415.112, 8089.23, pDataArray
365:   AdData 501839, 3488907, 409.482, 8113.74, pDataArray
366:   AdData 501860, 3488899, 404.262, 8138.25, pDataArray
367:   AdData 501877, 3488882, 409.949, 8162.77, pDataArray
368:   AdData 501897, 3488870, 417.896, 8187.28, pDataArray
369:   AdData 501922, 3488870, 404.409, 8211.79, pDataArray
370:   AdData 501941, 3488856, 393.337, 8236.3, pDataArray
371:   AdData 501958, 3488839, 391.863, 8260.82, pDataArray
372:   AdData 501979, 3488832, 390.626, 8285.33, pDataArray
373:   AdData 502004, 3488832, 384.551, 8309.84, pDataArray
374:   AdData 502028, 3488832, 375.991, 8334.36, pDataArray
375:   AdData 502053, 3488832, 366.608, 8358.87, pDataArray
376:   AdData 502077, 3488832, 356.438, 8383.38, pDataArray
377:   AdData 502095, 3488815, 344.273, 8407.89, pDataArray
```

```
378:   AdData 502112, 3488798, 339.734, 8432.41, pDataArray
379:   AdData 502135, 3488794, 339.767, 8456.92, pDataArray
380:   AdData 502160, 3488794, 339.421, 8481.43, pDataArray
381:   AdData 502184, 3488794, 338.055, 8505.95, pDataArray
382:   AdData 502209, 3488794, 330.662, 8530.46, pDataArray
383:   AdData 502232, 3488791, 319.815, 8554.97, pDataArray
384:   AdData 502250, 3488774, 310.25, 8579.48, pDataArray
385:   AdData 502267, 3488756, 311.282, 8604, pDataArray
386:   AdData 502291, 3488756, 317.144, 8628.51, pDataArray
387:   AdData 502313, 3488748, 309.892, 8653.02, pDataArray
388:   AdData 502330, 3488731, 299.228, 8677.54, pDataArray
389:   AdData 502349, 3488719, 302.439, 8702.05, pDataArray
390:   AdData 502374, 3488719, 303.965, 8726.56, pDataArray
391:   AdData 502398, 3488719, 296.296, 8751.07, pDataArray
392:   AdData 502421, 3488715, 285.423, 8775.59, pDataArray
393:   AdData 502439, 3488698, 281.237, 8800.1, pDataArray
394:   AdData 502456, 3488681, 287.72, 8824.61, pDataArray
395:   AdData 502481, 3488681, 298.665, 8849.13, pDataArray
396:   AdData 502505, 3488681, 303.635, 8873.64, pDataArray
397:   AdData 502530, 3488681, 302.109, 8898.15, pDataArray
398:   AdData 502554, 3488681, 302.015, 8922.66, pDataArray
399:   AdData 502579, 3488681, 302.624, 8947.18, pDataArray
400:   AdData 502603, 3488681, 304.15, 8971.69, pDataArray
401:   AdData 502628, 3488681, 308.238, 8996.2, pDataArray
402:   AdData 502652, 3488681, 313.637, 9020.71, pDataArray
403:   AdData 502677, 3488681, 320.937, 9045.23, pDataArray
404:   AdData 502701, 3488681, 330.193, 9069.74, pDataArray
405:   AdData 502726, 3488681, 340.527, 9094.25, pDataArray
406:   AdData 502750, 3488681, 352.531, 9118.77, pDataArray
407:   AdData 502775, 3488681, 363.254, 9143.28, pDataArray
408:   AdData 502799, 3488681, 372.4, 9167.79, pDataArray
409:   AdData 502824, 3488681, 376.105, 9192.3, pDataArray
410:   AdData 502848, 3488681, 377.519, 9216.82, pDataArray
411:   AdData 502872, 3488682, 377.401, 9241.33, pDataArray
412:   AdData 502890, 3488700, 379.299, 9265.84, pDataArray
413:   AdData 502907, 3488717, 387.513, 9290.36, pDataArray
414:   AdData 502931, 3488719, 400.177, 9314.87, pDataArray
415:   AdData 502955, 3488719, 413.514, 9339.38, pDataArray
416:   AdData 502980, 3488719, 427.857, 9363.89, pDataArray
417:   AdData 503004, 3488719, 436.528, 9388.41, pDataArray
418:   AdData 503029, 3488719, 444.471, 9412.92, pDataArray
419:   AdData 503053, 3488719, 453.745, 9437.43, pDataArray
420:   AdData 503078, 3488719, 464.263, 9461.95, pDataArray
421:   AdData 503101, 3488722, 475.533, 9486.46, pDataArray
422:   AdData 503118, 3488739, 489.782, 9510.97, pDataArray
423:   AdData 503136, 3488757, 506.233, 9535.48, pDataArray
424:   AdData 503153, 3488774, 496.761, 9560, pDataArray
```

```
425:   AdData 503170, 3488791, 492.436, 9584.51, pDataArray
426:   AdData 503188, 3488809, 488.502, 9609.02, pDataArray
427:   AdData 503205, 3488826, 484.626, 9633.54, pDataArray
428:   AdData 503222, 3488843, 482.245, 9658.05, pDataArray
429:   AdData 503240, 3488861, 480.935, 9682.56, pDataArray
430:   AdData 503257, 3488878, 480.91, 9707.07, pDataArray
431:   AdData 503274, 3488895, 482.205, 9731.59, pDataArray
432:   AdData 503292, 3488913, 484.538, 9756.1, pDataArray
433:   AdData 503309, 3488930, 488.399, 9780.61, pDataArray
434:   AdData 503326, 3488947, 492.077, 9805.12, pDataArray
435:   AdData 503344, 3488965, 495.199, 9829.64, pDataArray
436:   AdData 503361, 3488982, 505.821, 9854.15, pDataArray
437:   AdData 503378, 3488999, 506.677, 9878.66, pDataArray
438:   AdData 503396, 3489017, 520.723, 9903.18, pDataArray
439:   AdData 503413, 3489034, 520.245, 9927.69, pDataArray
440:   AdData 503430, 3489051, 528.398, 9952.2, pDataArray
441:   AdData 503437, 3489073, 538.096, 9976.71, pDataArray
442:   AdData 503438, 3489097, 540.478, 10001.2, pDataArray
443:   AdData 503456, 3489114, 547.106, 10025.7, pDataArray
444:   AdData 503473, 3489132, 567.839, 10050.3, pDataArray
445:   AdData 503490, 3489149, 570.242, 10074.8, pDataArray
446:   AdData 503508, 3489166, 583.814, 10099.3, pDataArray
447:   AdData 503513, 3489189, 596.887, 10123.8, pDataArray
448:   AdData 503516, 3489212, 604.641, 10148.3, pDataArray
449:   AdData 503533, 3489230, 616.226, 10172.8, pDataArray
450:   AdData 503550, 3489247, 641.253, 10197.3, pDataArray
451:   AdData 503568, 3489264, 645.384, 10221.8, pDataArray
452:   AdData 503585, 3489282, 663.687, 10246.4, pDataArray
453:   AdData 503602, 3489299, 673.864, 10270.9, pDataArray
454:   AdData 503620, 3489316, 688.339, 10295.4, pDataArray
455:   AdData 503637, 3489334, 697.047, 10319.9, pDataArray
456:   AdData 503654, 3489351, 710.984, 10344.4, pDataArray
457:   AdData 503672, 3489368, 721.118, 10368.9, pDataArray
458:   AdData 503689, 3489386, 722.663, 10393.4, pDataArray
459:   AdData 503708, 3489398, 737.057, 10417.9, pDataArray
460:   AdData 503733, 3489398, 751.973, 10442.5, pDataArray
461:   AdData 503752, 3489411, 755.639, 10467, pDataArray
462:   AdData 503769, 3489428, 767.476, 10491.5, pDataArray
463:   AdData 503791, 3489436, 782.903, 10516, pDataArray
464:   AdData 503815, 3489436, 792.802, 10540.5, pDataArray
465:   AdData 503832, 3489453, 800.102, 10565, pDataArray
466:   AdData 503850, 3489471, 820.018, 10589.5, pDataArray
467:   AdData 503873, 3489474, 829.64, 10614, pDataArray
468:   AdData 503895, 3489479, 826.45, 10638.6, pDataArray
469:   AdData 503913, 3489496, 811.936, 10663.1, pDataArray
470:   AdData 503930, 3489513, 809.58, 10687.6, pDataArray
471:   AdData 503947, 3489531, 789.197, 10712.1, pDataArray
```

```
472:   AdData 503965, 3489548, 787.758, 10736.6, pDataArray
473:   AdData 503982, 3489565, 767.999, 10761.1, pDataArray
474:   AdData 503999, 3489583, 763.055, 10785.6, pDataArray
475:   AdData 504017, 3489600, 749.346, 10810.1, pDataArray
476:   AdData 504034, 3489617, 742.29, 10834.7, pDataArray
477:   AdData 504051, 3489635, 733.566, 10859.2, pDataArray
478:   AdData 504069, 3489652, 725.482, 10883.7, pDataArray
479:   AdData 504086, 3489669, 720.529, 10908.2, pDataArray
480:   AdData 504103, 3489687, 712.589, 10932.7, pDataArray
481:   AdData 504121, 3489704, 710.068, 10957.2, pDataArray
482:   AdData 504138, 3489721, 703.503, 10981.7, pDataArray
483:   AdData 504155, 3489739, 701.981, 11006.3, pDataArray
484:   AdData 504173, 3489756, 698.05, 11030.8, pDataArray
485:   AdData 504190, 3489773, 696.199, 11055.3, pDataArray
486:   AdData 504207, 3489791, 695.046, 11079.8, pDataArray
487:   AdData 504225, 3489808, 694.226, 11104.3, pDataArray
488:   AdData 504242, 3489825, 694.284, 11128.8, pDataArray
489:   AdData 504259, 3489843, 694.239, 11153.3, pDataArray
490:   AdData 504277, 3489860, 694.361, 11177.8, pDataArray
491:   AdData 504294, 3489877, 695.27, 11202.4, pDataArray
492:   AdData 504311, 3489895, 696.927, 11226.9, pDataArray
493:   AdData 504329, 3489912, 699.629, 11251.4, pDataArray
494:   AdData 504346, 3489929, 702.898, 11275.9, pDataArray
495:   AdData 504363, 3489947, 707.354, 11300.4, pDataArray
496:   AdData 504381, 3489964, 712.205, 11324.9, pDataArray
497:   AdData 504398, 3489981, 709.777, 11349.4, pDataArray
498:   AdData 504415, 3489999, 715.248, 11373.9, pDataArray
499:   AdData 504438, 3490002, 705.18, 11398.5, pDataArray
500:   AdData 504461, 3490007, 689.706, 11423, pDataArray
501:   AdData 504478, 3490024, 680.926, 11447.5, pDataArray
502:   AdData 504496, 3490040, 682.847, 11472, pDataArray
503:   AdData 504521, 3490040, 672.577, 11496.5, pDataArray
504:   AdData 504542, 3490049, 659.219, 11521, pDataArray
505:   AdData 504559, 3490067, 654.622, 11545.5, pDataArray
506:   AdData 504579, 3490078, 655.402, 11570, pDataArray
507:   AdData 504603, 3490078, 644.742, 11594.6, pDataArray
508:   AdData 504622, 3490092, 635.259, 11619.1, pDataArray
509:   AdData 504639, 3490109, 639.878, 11643.6, pDataArray
510:   AdData 504661, 3490115, 638.037, 11668.1, pDataArray
511:   AdData 504685, 3490117, 627.263, 11692.6, pDataArray
512:   AdData 504702, 3490135, 626.057, 11717.1, pDataArray
513:   AdData 504720, 3490152, 640.223, 11741.6, pDataArray
514:   AdData 504737, 3490169, 627.109, 11766.1, pDataArray
515:   AdData 504754, 3490187, 626.174, 11790.7, pDataArray
516:   AdData 504777, 3490191, 634.895, 11815.2, pDataArray
End Sub
Private Sub MakeData6(pDataArray As esriSystem.IVariantArray)
```

```
519:   AdData 504800, 3490195, 636.809, 11839.7, pDataArray
520:   AdData 504817, 3490212, 622.4, 11864.2, pDataArray
521:   AdData 504835, 3490229, 622.312, 11888.7, pDataArray
522:   AdData 504859, 3490229, 613.294, 11913.2, pDataArray
523:   AdData 504880, 3490237, 600.354, 11937.7, pDataArray
524:   AdData 504898, 3490255, 593.627, 11962.3, pDataArray
525:   AdData 504915, 3490272, 588.664, 11986.8, pDataArray
526:   AdData 504932, 3490289, 571.076, 12011.3, pDataArray
527:   AdData 504951, 3490304, 565.826, 12035.8, pDataArray
528:   AdData 504975, 3490304, 562.543, 12060.3, pDataArray
529:   AdData 504995, 3490315, 546.904, 12084.8, pDataArray
530:   AdData 505013, 3490332, 532.619, 12109.3, pDataArray
531:   AdData 505033, 3490342, 530.97, 12133.8, pDataArray
532:   AdData 505058, 3490342, 533.58, 12158.4, pDataArray
533:   AdData 505076, 3490357, 513.711, 12182.9, pDataArray
534:   AdData 505093, 3490375, 503.897, 12207.4, pDataArray
535:   AdData 505116, 3490380, 505.6, 12231.9, pDataArray
536:   AdData 505139, 3490383, 503.963, 12256.4, pDataArray
537:   AdData 505156, 3490400, 485.617, 12280.9, pDataArray
538:   AdData 505174, 3490417, 477.574, 12305.4, pDataArray
539:   AdData 505198, 3490417, 470.781, 12329.9, pDataArray
540:   AdData 505222, 3490417, 464.749, 12354.5, pDataArray
541:   AdData 505247, 3490417, 459.754, 12379, pDataArray
542:   AdData 505272, 3490417, 456.527, 12403.5, pDataArray
543:   AdData 505296, 3490417, 451.592, 12428, pDataArray
544:   AdData 505321, 3490417, 443.333, 12452.5, pDataArray
545:   AdData 505345, 3490417, 435.76, 12477, pDataArray
546:   AdData 505370, 3490417, 428.9, 12501.5, pDataArray
547:   AdData 505394, 3490417, 423.473, 12526, pDataArray
548:   AdData 505413, 3490430, 419.788, 12550.6, pDataArray
549:   AdData 505431, 3490448, 420.451, 12575.1, pDataArray
550:   AdData 505452, 3490455, 425.147, 12599.6, pDataArray
551:   AdData 505476, 3490455, 430.625, 12624.1, pDataArray
552:   AdData 505501, 3490455, 438.051, 12648.6, pDataArray
553:   AdData 505526, 3490455, 445.881, 12673.1, pDataArray
554:   AdData 505550, 3490455, 454.139, 12697.6, pDataArray
555:   AdData 505575, 3490455, 457.464, 12722.1, pDataArray
556:   AdData 505599, 3490455, 461.295, 12746.7, pDataArray
557:   AdData 505624, 3490455, 466.29, 12771.2, pDataArray
558:   AdData 505648, 3490455, 472.897, 12795.7, pDataArray
559:   AdData 505673, 3490455, 475.302, 12820.2, pDataArray
560:   AdData 505697, 3490455, 468.443, 12844.7, pDataArray
561:   AdData 505716, 3490469, 463.419, 12869.2, pDataArray
562:   AdData 505733, 3490486, 464.688, 12893.7, pDataArray
563:   AdData 505755, 3490493, 471.126, 12918.3, pDataArray
564:   AdData 505780, 3490493, 478.1, 12942.8, pDataArray
565:   AdData 505804, 3490493, 486.702, 12967.3, pDataArray
```

```
566:   AdData 505829, 3490493, 490.775, 12991.8, pDataArray
567:   AdData 505853, 3490493, 490.775, 13016.3, pDataArray
568:   AdData 505878, 3490493, 490.775, 13040.8, pDataArray
569:   AdData 505899, 3490501, 491.964, 13065.3, pDataArray
570:   AdData 505916, 3490518, 501.11, 13089.8, pDataArray
571:   AdData 505935, 3490531, 515.395, 13114.4, pDataArray
572:   AdData 505960, 3490531, 522.527, 13138.9, pDataArray
573:   AdData 505979, 3490543, 517.834, 13163.4, pDataArray
574:   AdData 505997, 3490561, 524.74, 13187.9, pDataArray
575:   AdData 506014, 3490578, 527.121, 13212.4, pDataArray
576:   AdData 506031, 3490595, 524.186, 13236.9, pDataArray
577:   AdData 506051, 3490606, 528.527, 13261.4, pDataArray
578:   AdData 506076, 3490606, 528.527, 13285.9, pDataArray
579:   AdData 506100, 3490606, 517.384, 13310.5, pDataArray
580:   AdData 506123, 3490611, 502.452, 13335, pDataArray
581:   AdData 506140, 3490629, 491.641, 13359.5, pDataArray
582:   AdData 506158, 3490644, 491.358, 13384, pDataArray
583:   AdData 506183, 3490644, 483.747, 13408.5, pDataArray
584:   AdData 506203, 3490654, 470.417, 13433, pDataArray
585:   AdData 506220, 3490671, 463.985, 13457.5, pDataArray
586:   AdData 506241, 3490682, 463.786, 13482, pDataArray
587:   AdData 506265, 3490682, 455.753, 13506.6, pDataArray
588:   AdData 506283, 3490697, 445.005, 13531.1, pDataArray
589:   AdData 506301, 3490714, 449.525, 13555.6, pDataArray
590:   AdData 506318, 3490731, 440.347, 13580.1, pDataArray
591:   AdData 506335, 3490749, 435.473, 13604.6, pDataArray
592:   AdData 506353, 3490766, 431.778, 13629.1, pDataArray
593:   AdData 506370, 3490783, 425.941, 13653.6, pDataArray
594:   AdData 506390, 3490795, 430.438, 13678.1, pDataArray
595:   AdData 506414, 3490795, 430.438, 13702.7, pDataArray
596:   AdData 506433, 3490809, 425.856, 13727.2, pDataArray
597:   AdData 506451, 3490826, 433.226, 13751.7, pDataArray
598:   AdData 506468, 3490843, 428.479, 13776.2, pDataArray
599:   AdData 506485, 3490861, 420.338, 13800.7, pDataArray
600:   AdData 506506, 3490870, 427.102, 13825.2, pDataArray
601:   AdData 506530, 3490870, 438.907, 13849.7, pDataArray
602:   AdData 506555, 3490870, 434.613, 13874.3, pDataArray
603:   AdData 506577, 3490876, 428.455, 13898.8, pDataArray
604:   AdData 506594, 3490894, 430.238, 13923.3, pDataArray
605:   AdData 506613, 3490908, 440.435, 13947.8, pDataArray
606:   AdData 506637, 3490908, 441.484, 13972.3, pDataArray
607:   AdData 506662, 3490908, 437.217, 13996.8, pDataArray
608:   AdData 506686, 3490908, 429.965, 14021.3, pDataArray
609:   AdData 506711, 3490908, 424.538, 14045.8, pDataArray
610:   AdData 506735, 3490908, 420.269, 14070.4, pDataArray
611:   AdData 506760, 3490908, 416.976, 14094.9, pDataArray
612:   AdData 506784, 3490908, 415.864, 14119.4, pDataArray
```

```
613:   AdData 506809, 3490908, 415.79, 14143.9, pDataArray
614:   AdData 506833, 3490908, 416.902, 14168.4, pDataArray
615:   AdData 506858, 3490908, 420.05, 14192.9, pDataArray
616:   AdData 506882, 3490908, 424.178, 14217.4, pDataArray
617:   AdData 506907, 3490908, 429.606, 14241.9, pDataArray
618:   AdData 506931, 3490908, 436.725, 14266.5, pDataArray
619:   AdData 506956, 3490908, 444.094, 14291, pDataArray
620:   AdData 506980, 3490908, 451.332, 14315.5, pDataArray
621:   AdData 507005, 3490908, 453.023, 14340, pDataArray
622:   AdData 507029, 3490908, 453.023, 14364.5, pDataArray
623:   AdData 507054, 3490908, 453.023, 14389, pDataArray
624:   AdData 507078, 3490908, 453.023, 14413.5, pDataArray
625:   AdData 507103, 3490908, 453.023, 14438, pDataArray
626:   AdData 507127, 3490908, 453.023, 14462.6, pDataArray
627:   AdData 507152, 3490908, 453.023, 14487.1, pDataArray
628:   AdData 507176, 3490908, 453.023, 14511.6, pDataArray
629:   AdData 507201, 3490908, 453.023, 14536.1, pDataArray
630:   AdData 507225, 3490908, 449.217, 14560.6, pDataArray
631:   AdData 507250, 3490908, 441.978, 14585.1, pDataArray
632:   AdData 507274, 3490908, 434.555, 14609.6, pDataArray
633:   AdData 507296, 3490901, 426.538, 14634.1, pDataArray
634:   AdData 507313, 3490883, 421.887, 14658.7, pDataArray
635:   AdData 507332, 3490870, 423.532, 14683.2, pDataArray
636:   AdData 507357, 3490870, 428.96, 14707.7, pDataArray
637:   AdData 507381, 3490870, 435.841, 14732.2, pDataArray
638:   AdData 507406, 3490870, 443.232, 14756.7, pDataArray
639:   AdData 507430, 3490870, 450.471, 14781.2, pDataArray
640:   AdData 507455, 3490870, 453.023, 14805.7, pDataArray
641:   AdData 507479, 3490869, 452.518, 14830.3, pDataArray
642:   AdData 507496, 3490851, 452.241, 14854.8, pDataArray
643:   AdData 507513, 3490834, 459.972, 14879.3, pDataArray
644:   AdData 507537, 3490833, 474.482, 14903.8, pDataArray
645:   AdData 507562, 3490833, 488.569, 14928.3, pDataArray
646:   AdData 507586, 3490833, 501.694, 14952.8, pDataArray
End Sub
Private Sub MakeData7(pDataArray As esriSystem.IVariantArray)
649:   AdData 507605, 3490818, 497.11, 14977.3, pDataArray
650:   AdData 507622, 3490801, 502.836, 15001.8, pDataArray
651:   AdData 507644, 3490795, 517.475, 15026.4, pDataArray
652:   AdData 507669, 3490795, 529.759, 15050.9, pDataArray
653:   AdData 507693, 3490795, 528.885, 15075.4, pDataArray
654:   AdData 507713, 3490785, 531.995, 15099.9, pDataArray
655:   AdData 507731, 3490767, 547.029, 15124.4, pDataArray
656:   AdData 507751, 3490757, 562.915, 15148.9, pDataArray
657:   AdData 507776, 3490757, 567.015, 15173.4, pDataArray
658:   AdData 507794, 3490742, 572.853, 15197.9, pDataArray
659:   AdData 507811, 3490725, 589.556, 15222.5, pDataArray
```

```
660:   AdData 507834, 3490719, 604.872, 15247, pDataArray
661:   AdData 507858, 3490719, 615.367, 15271.5, pDataArray
662:   AdData 507883, 3490719, 620.603, 15296, pDataArray
663:   AdData 507907, 3490719, 624.894, 15320.5, pDataArray
664:   AdData 507931, 3490721, 626.609, 15345, pDataArray
665:   AdData 507948, 3490738, 610.804, 15369.5, pDataArray
666:   AdData 507966, 3490755, 608.241, 15394, pDataArray
667:   AdData 507989, 3490757, 606.681, 15418.6, pDataArray
668:   AdData 508014, 3490757, 604.937, 15443.1, pDataArray
669:   AdData 508038, 3490757, 604.172, 15467.6, pDataArray
670:   AdData 508063, 3490757, 600.257, 15492.1, pDataArray
671:   AdData 508088, 3490757, 594.966, 15516.6, pDataArray
672:   AdData 508112, 3490757, 587.925, 15541.1, pDataArray
673:   AdData 508137, 3490757, 581.969, 15565.6, pDataArray
674:   AdData 508161, 3490757, 576.704, 15590.1, pDataArray
675:   AdData 508186, 3490757, 572.676, 15614.7, pDataArray
676:   AdData 508210, 3490757, 567.002, 15639.2, pDataArray
677:   AdData 508235, 3490757, 559.367, 15663.7, pDataArray
678:   AdData 508259, 3490757, 543.35, 15688.2, pDataArray
679:   AdData 508279, 3490747, 530.04, 15712.7, pDataArray
680:   AdData 508297, 3490730, 526.836, 15737.2, pDataArray
681:   AdData 508317, 3490719, 524.198, 15761.7, pDataArray
682:   AdData 508342, 3490719, 509.931, 15786.3, pDataArray
683:   AdData 508360, 3490705, 495.995, 15810.8, pDataArray
684:   AdData 508377, 3490687, 490.581, 15835.3, pDataArray
685:   AdData 508399, 3490682, 484.929, 15859.8, pDataArray
686:   AdData 508424, 3490682, 476.014, 15884.3, pDataArray
687:   AdData 508448, 3490682, 465.166, 15908.8, pDataArray
688:   AdData 508473, 3490682, 452.817, 15933.3, pDataArray
689:   AdData 508497, 3490682, 439.558, 15957.8, pDataArray
690:   AdData 508522, 3490682, 427.754, 15982.4, pDataArray
691:   AdData 508547, 3490682, 416.86, 16006.9, pDataArray
692:   AdData 508571, 3490682, 406.809, 16031.4, pDataArray
693:   AdData 508596, 3490682, 398.679, 16055.9, pDataArray
694:   AdData 508620, 3490682, 391.534, 16080.4, pDataArray
695:   AdData 508645, 3490682, 385.595, 16104.9, pDataArray
696:   AdData 508669, 3490682, 381.73, 16129.4, pDataArray
697:   AdData 508694, 3490682, 378.963, 16153.9, pDataArray
698:   AdData 508718, 3490682, 377.741, 16178.5, pDataArray
699:   AdData 508743, 3490682, 374.485, 16203, pDataArray
700:   AdData 508767, 3490682, 369.339, 16227.5, pDataArray
701:   AdData 508792, 3490682, 360.465, 16252, pDataArray
702:   AdData 508811, 3490695, 351.073, 16276.5, pDataArray
703:   AdData 508828, 3490712, 347.468, 16301, pDataArray
704:   AdData 508850, 3490719, 348.056, 16325.5, pDataArray
705:   AdData 508874, 3490720, 347.556, 16350, pDataArray
706:   AdData 508891, 3490737, 332.521, 16374.6, pDataArray
```

```
707:   AdData 508909, 3490754, 335.543, 16399.1, pDataArray
708:   AdData 508926, 3490772, 324.771, 16423.6, pDataArray
709:   AdData 508943, 3490789, 322.404, 16448.1, pDataArray
710:   AdData 508965, 3490795, 330.325, 16472.6, pDataArray
711:   AdData 508989, 3490797, 335.95, 16497.1, pDataArray
712:   AdData 509006, 3490814, 332.332, 16521.6, pDataArray
713:   AdData 509024, 3490832, 346.773, 16546.1, pDataArray
714:   AdData 509041, 3490849, 343.859, 16570.7, pDataArray
715:   AdData 509058, 3490866, 352.307, 16595.2, pDataArray
716:   AdData 509076, 3490884, 353.879, 16619.7, pDataArray
717:   AdData 509093, 3490901, 363.654, 16644.2, pDataArray
718:   AdData 509110, 3490918, 370.754, 16668.7, pDataArray
719:   AdData 509128, 3490936, 381.027, 16693.2, pDataArray
720:   AdData 509145, 3490953, 392.718, 16717.7, pDataArray
721:   AdData 509162, 3490970, 400.418, 16742.3, pDataArray
722:   AdData 509176, 3490989, 416.984, 16766.8, pDataArray
723:   AdData 509176, 3491014, 416.984, 16791.3, pDataArray
724:   AdData 509176, 3491039, 416.212, 16815.8, pDataArray
725:   AdData 509178, 3491062, 413.838, 16840.3, pDataArray
726:   AdData 509196, 3491079, 407.573, 16864.8, pDataArray
727:   AdData 509213, 3491097, 406.561, 16889.3, pDataArray
728:   AdData 509230, 3491114, 398.225, 16913.8, pDataArray
729:   AdData 509248, 3491131, 390.224, 16938.4, pDataArray
730:   AdData 509265, 3491149, 384.196, 16962.9, pDataArray
731:   AdData 509282, 3491166, 379.221, 16987.4, pDataArray
732:   AdData 509300, 3491183, 371.595, 17011.9, pDataArray
733:   AdData 509317, 3491201, 370.165, 17036.4, pDataArray
734:   AdData 509334, 3491218, 364.777, 17060.9, pDataArray
735:   AdData 509352, 3491235, 355.41, 17085.4, pDataArray
736:   AdData 509369, 3491253, 351.869, 17109.9, pDataArray
737:   AdData 509386, 3491270, 337.59, 17134.5, pDataArray
738:   AdData 509404, 3491287, 335.375, 17159, pDataArray
739:   AdData 509421, 3491305, 318.993, 17183.5, pDataArray
740:   AdData 509438, 3491322, 321.39, 17208, pDataArray
741:   AdData 509456, 3491339, 304.138, 17232.5, pDataArray
742:   AdData 509473, 3491357, 301.782, 17257, pDataArray
743:   AdData 509490, 3491374, 294.016, 17281.5, pDataArray
744:   AdData 509508, 3491391, 291.559, 17306, pDataArray
745:   AdData 509515, 3491413, 298.262, 17330.6, pDataArray
746:   AdData 509516, 3491437, 304.222, 17355.1, pDataArray
747:   AdData 509533, 3491454, 301.733, 17379.6, pDataArray
748:   AdData 509550, 3491472, 318.028, 17404.1, pDataArray
749:   AdData 509568, 3491489, 317.814, 17428.6, pDataArray
750:   AdData 509585, 3491506, 329.352, 17453.1, pDataArray
751:   AdData 509602, 3491524, 337.224, 17477.6, pDataArray
752:   AdData 509620, 3491541, 347.514, 17502.1, pDataArray
753:   AdData 509629, 3491562, 364.158, 17526.7, pDataArray
```

```
754:   AdData 509629, 3491586, 376.737, 17551.2, pDataArray
755:   AdData 509645, 3491604, 386.352, 17575.7, pDataArray
756:   AdData 509662, 3491621, 408.138, 17600.2, pDataArray
757:   AdData 509679, 3491639, 413.541, 17624.7, pDataArray
758:   AdData 509697, 3491656, 420.837, 17649.2, pDataArray
759:   AdData 509704, 3491678, 436.902, 17673.7, pDataArray
760:   AdData 509704, 3491702, 453.023, 17698.3, pDataArray
761:   AdData 509704, 3491727, 453.023, 17722.8, pDataArray
762:   AdData 509713, 3491747, 454.916, 17747.3, pDataArray
763:   AdData 509730, 3491765, 467.827, 17771.8, pDataArray
764:   AdData 509742, 3491784, 484.998, 17796.3, pDataArray
765:   AdData 509742, 3491809, 489.747, 17820.8, pDataArray
766:   AdData 509755, 3491828, 495.469, 17845.3, pDataArray
767:   AdData 509773, 3491845, 513.582, 17869.8, pDataArray
768:   AdData 509790, 3491863, 522.916, 17894.4, pDataArray
769:   AdData 509807, 3491880, 530.497, 17918.9, pDataArray
770:   AdData 509817, 3491900, 547.674, 17943.4, pDataArray
771:   AdData 509817, 3491925, 564.396, 17967.9, pDataArray
772:   AdData 509833, 3491943, 570.143, 17992.4, pDataArray
773:   AdData 509850, 3491960, 588.697, 18016.9, pDataArray
774:   AdData 509867, 3491978, 598.82, 18041.4, pDataArray
775:   AdData 509885, 3491995, 612.118, 18065.9, pDataArray
776:   AdData 509902, 3492012, 623.286, 18090.5, pDataArray
777:   AdData 509919, 3492030, 635.789, 18115, pDataArray
778:   AdData 509937, 3492047, 650.144, 18139.5, pDataArray
779:   AdData 509954, 3492064, 655.972, 18164, pDataArray
780:   AdData 509971, 3492082, 669.662, 18188.5, pDataArray
781:   AdData 509989, 3492099, 666.06, 18213, pDataArray
782:   AdData 510006, 3492116, 680.519, 18237.5, pDataArray
783:   AdData 510023, 3492134, 674.923, 18262, pDataArray
784:   AdData 510041, 3492151, 681.282, 18286.6, pDataArray
785:   AdData 510044, 3492174, 679.254, 18311.1, pDataArray
786:   AdData 510044, 3492199, 672.423, 18335.6, pDataArray
787:   AdData 510044, 3492223, 662.023, 18360.1, pDataArray
788:   AdData 510057, 3492242, 650.305, 18384.6, pDataArray
789:   AdData 510074, 3492260, 643.737, 18409.1, pDataArray
790:   AdData 510082, 3492281, 644.101, 18433.6, pDataArray
791:   AdData 510082, 3492306, 646.284, 18458.1, pDataArray
792:   AdData 510082, 3492330, 649.849, 18482.7, pDataArray
793:   AdData 510082, 3492355, 651.699, 18507.2, pDataArray
794:   AdData 510082, 3492379, 651.699, 18531.7, pDataArray
795:   AdData 510082, 3492404, 648.345, 18556.2, pDataArray
796:   AdData 510082, 3492428, 645.346, 18580.7, pDataArray
797:   AdData 510082, 3492453, 643.192, 18605.2, pDataArray
798:   AdData 510097, 3492471, 645.719, 18629.7, pDataArray
799:   AdData 510114, 3492488, 655.011, 18654.2, pDataArray
800:   AdData 510131, 3492506, 653.631, 18678.8, pDataArray
```

```
801:   AdData 510149, 3492523, 645.618, 18703.3, pDataArray
802:   AdData 510166, 3492540, 638.489, 18727.8, pDataArray
803:   AdData 510183, 3492558, 632.271, 18752.3, pDataArray
804:   AdData 510201, 3492575, 626.81, 18776.8, pDataArray
805:   AdData 510218, 3492592, 622.494, 18801.3, pDataArray
806:   AdData 510235, 3492610, 618.79, 18825.8, pDataArray
807:   AdData 510253, 3492627, 616.457, 18850.4, pDataArray
808:   AdData 510270, 3492644, 614.604, 18874.9, pDataArray
809:   AdData 510287, 3492662, 607.819, 18899.4, pDataArray
810:   AdData 510305, 3492679, 612.207, 18923.9, pDataArray
811:   AdData 510322, 3492696, 599.873, 18948.4, pDataArray
812:   AdData 510339, 3492714, 594.834, 18972.9, pDataArray
813:   AdData 510361, 3492720, 596.91, 18997.4, pDataArray
814:   AdData 510385, 3492722, 595.493, 19021.9, pDataArray
815:   AdData 510402, 3492739, 583.638, 19046.5, pDataArray
816:   AdData 510420, 3492756, 585.148, 19071, pDataArray
817:   AdData 510443, 3492758, 581.077, 19095.5, pDataArray
818:   AdData 510468, 3492758, 576.058, 19120, pDataArray
819:   AdData 510492, 3492758, 572.031, 19144.5, pDataArray
820:   AdData 510517, 3492758, 565.955, 19169, pDataArray
821:   AdData 510541, 3492758, 559.152, 19193.5, pDataArray
822:   AdData 510566, 3492758, 551.655, 19218, pDataArray
823:   AdData 510590, 3492758, 545.303, 19242.6, pDataArray
824:   AdData 510615, 3492758, 539.681, 19267.1, pDataArray
825:   AdData 510639, 3492758, 535.375, 19291.6, pDataArray
826:   AdData 510659, 3492769, 530.701, 19316.1, pDataArray
827:   AdData 510677, 3492787, 529.121, 19340.6, pDataArray
828:   AdData 510697, 3492796, 531.121, 19365.1, pDataArray
829:   AdData 510722, 3492796, 533.731, 19389.6, pDataArray
830:   AdData 510746, 3492796, 537.932, 19414.1, pDataArray
831:   AdData 510771, 3492796, 538.813, 19438.7, pDataArray
832:   AdData 510795, 3492796, 534.506, 19463.2, pDataArray
833:   AdData 510820, 3492796, 531.655, 19487.7, pDataArray
834:   AdData 510844, 3492796, 529.596, 19512.2, pDataArray
835:   AdData 510869, 3492796, 528.722, 19536.7, pDataArray
836:   AdData 510893, 3492796, 518.115, 19561.2, pDataArray
837:   AdData 510916, 3492792, 503.374, 19585.7, pDataArray
838:   AdData 510934, 3492774, 491.449, 19610.2, pDataArray
839:   AdData 510951, 3492758, 490.493, 19634.8, pDataArray
840:   AdData 510976, 3492758, 485.744, 19659.3, pDataArray
841:   AdData 510997, 3492749, 472.734, 19683.8, pDataArray
842:   AdData 511014, 3492732, 460.825, 19708.3, pDataArray
843:   AdData 511034, 3492720, 460.977, 19732.8, pDataArray
844:   AdData 511058, 3492720, 465.972, 19757.3, pDataArray
845:   AdData 511083, 3492720, 461.344, 19781.8, pDataArray
846:   AdData 511105, 3492716, 453.145, 19806.4, pDataArray
847:   AdData 511123, 3492698, 449.195, 19830.9, pDataArray
```

```
848:   AdData 511141, 3492683, 456.741, 19855.4, pDataArray
849:   AdData 511165, 3492683, 463.758, 19879.9, pDataArray
850:   AdData 511190, 3492683, 464.253, 19904.4, pDataArray
851:   AdData 511214, 3492683, 459.262, 19928.9, pDataArray
852:   AdData 511232, 3492665, 460.253, 19953.4, pDataArray
853:   AdData 511249, 3492648, 470.33, 19977.9, pDataArray
854:   AdData 511272, 3492645, 482.534, 20002.5, pDataArray
855:   AdData 511297, 3492645, 490.775, 20027, pDataArray
856:   AdData 511321, 3492645, 490.775, 20051.5, pDataArray
857:   AdData 511346, 3492645, 482.195, 20076, pDataArray
858:   AdData 511370, 3492645, 470.789, 20100.5, pDataArray
859:   AdData 511395, 3492645, 459.834, 20125, pDataArray
860:   AdData 511419, 3492645, 449.996, 20149.5, pDataArray
861:   AdData 511444, 3492645, 440.952, 20174, pDataArray
862:   AdData 511468, 3492645, 433.525, 20198.6, pDataArray
863:   AdData 511493, 3492645, 427.267, 20223.1, pDataArray
864:   AdData 511517, 3492645, 421.933, 20247.6, pDataArray
865:   AdData 511542, 3492645, 418.625, 20272.1, pDataArray
866:   AdData 511566, 3492645, 416.423, 20296.6, pDataArray
867:   AdData 511591, 3492645, 415.311, 20321.1, pDataArray
868:   AdData 511615, 3492645, 416.343, 20345.6, pDataArray
869:   AdData 511640, 3492645, 418.386, 20370.1, pDataArray
870:   AdData 511664, 3492645, 421.695, 20394.7, pDataArray
871:   AdData 511689, 3492645, 426.876, 20419.2, pDataArray
872:   AdData 511713, 3492645, 428.573, 20443.7, pDataArray
873:   AdData 511738, 3492645, 423.145, 20468.2, pDataArray
874:   AdData 511757, 3492631, 417.289, 20492.7, pDataArray
875:   AdData 511774, 3492614, 419.151, 20517.2, pDataArray
876:   AdData 511796, 3492607, 429.465, 20541.7, pDataArray
877:   AdData 511820, 3492607, 441.056, 20566.2, pDataArray
878:   AdData 511845, 3492607, 450.363, 20590.8, pDataArray
879:   AdData 511869, 3492607, 456.25, 20615.3, pDataArray
880:   AdData 511894, 3492607, 459.256, 20639.8, pDataArray
881:   AdData 511918, 3492607, 456.218, 20664.3, pDataArray
882:   AdData 511940, 3492615, 445.898, 20688.8, pDataArray
883:   AdData 511957, 3492632, 438.201, 20713.3, pDataArray
884:   AdData 511976, 3492645, 441.574, 20737.8, pDataArray
885:   AdData 512001, 3492645, 440.525, 20762.4, pDataArray
886:   AdData 512020, 3492658, 432.065, 20786.9, pDataArray
887:   AdData 512037, 3492675, 434.49, 20811.4, pDataArray
888:   AdData 512059, 3492683, 440.857, 20835.9, pDataArray
889:   AdData 512083, 3492683, 441.576, 20860.4, pDataArray
890:   AdData 512100, 3492700, 439.013, 20884.9, pDataArray
891:   AdData 512118, 3492718, 451.456, 20909.4, pDataArray
892:   AdData 512141, 3492720, 457.179, 20933.9, pDataArray
893:   AdData 512163, 3492726, 458.56, 20958.5, pDataArray
894:   AdData 512181, 3492743, 464.722, 20983, pDataArray
```

```
895:   AdData 512199, 3492758, 481.39, 21007.5, pDataArray
896:   AdData 512223, 3492758, 488.06, 21032, pDataArray
897:   AdData 512244, 3492768, 491.296, 21056.5, pDataArray
898:   AdData 512261, 3492786, 501.767, 21081, pDataArray
899:   AdData 512281, 3492796, 518.98, 21105.5, pDataArray
900:   AdData 512306, 3492796, 532.233, 21130, pDataArray
901:   AdData 512324, 3492811, 533.48, 21154.6, pDataArray
902:   AdData 512341, 3492828, 549.213, 21179.1, pDataArray
903:   AdData 512359, 3492846, 553.852, 21203.6, pDataArray
904:   AdData 512376, 3492863, 563.859, 21228.1, pDataArray
905:   AdData 512393, 3492880, 573.799, 21252.6, pDataArray
906:   AdData 512411, 3492898, 579.769, 21277.1, pDataArray
907:   AdData 512431, 3492909, 593.913, 21301.6, pDataArray
908:   AdData 512455, 3492909, 606.268, 21326.1, pDataArray
909:   AdData 512474, 3492923, 604.026, 21350.7, pDataArray
910:   AdData 512491, 3492940, 615.54, 21375.2, pDataArray
911:   AdData 512509, 3492958, 619.829, 21399.7, pDataArray
912:   AdData 512526, 3492975, 627.732, 21424.2, pDataArray
913:   AdData 512543, 3492992, 635.249, 21448.7, pDataArray
914:   AdData 512561, 3493010, 634.066, 21473.2, pDataArray
915:   AdData 512580, 3493022, 645.734, 21497.7, pDataArray
916:   AdData 512604, 3493022, 656.393, 21522.2, pDataArray
917:   AdData 512624, 3493035, 654.43, 21546.8, pDataArray
918:   AdData 512641, 3493052, 660.394, 21571.3, pDataArray
919:   AdData 512662, 3493060, 673.532, 21595.8, pDataArray
920:   AdData 512687, 3493060, 683.644, 21620.3, pDataArray
921:   AdData 512704, 3493078, 681.222, 21644.8, pDataArray
922:   AdData 512721, 3493095, 691.151, 21669.3, pDataArray
923:   AdData 512745, 3493098, 706.583, 21693.8, pDataArray
924:   AdData 512769, 3493098, 717.287, 21718.4, pDataArray
925:   AdData 512794, 3493098, 717.287, 21742.9, pDataArray
926:   AdData 512813, 3493111, 721.76, 21767.4, pDataArray
927:   AdData 512830, 3493128, 737.389, 21791.9, pDataArray
928:   AdData 512852, 3493136, 750.269, 21816.4, pDataArray
929:   AdData 512876, 3493136, 754.933, 21840.9, pDataArray
930:   AdData 512893, 3493153, 757.54, 21865.4, pDataArray
931:   AdData 512910, 3493171, 771.596, 21889.9, pDataArray
932:   AdData 512934, 3493173, 784.69, 21914.5, pDataArray
933:   AdData 512956, 3493179, 790.751, 21939, pDataArray
934:   AdData 512974, 3493196, 792.014, 21963.5, pDataArray
935:   AdData 512992, 3493211, 804.854, 21988, pDataArray
936:   AdData 513016, 3493211, 821.313, 22012.5, pDataArray
937:   AdData 513041, 3493211, 828.962, 22037, pDataArray
938:   AdData 513065, 3493212, 830.426, 22061.5, pDataArray
939:   AdData 513082, 3493229, 833.443, 22086, pDataArray
940:   AdData 513100, 3493247, 846.151, 22110.6, pDataArray
941:   AdData 513123, 3493249, 838.698, 22135.1, pDataArray
```

```
942:   AdData 513148, 3493249, 826.254, 22159.6, pDataArray
943:   AdData 513172, 3493249, 811.012, 22184.1, pDataArray
944:   AdData 513197, 3493249, 796.347, 22208.6, pDataArray
945:   AdData 513219, 3493253, 782.385, 22233.1, pDataArray
946:   AdData 513237, 3493271, 773.435, 22257.6, pDataArray
947:   AdData 513255, 3493287, 769.656, 22282.1, pDataArray
948:   AdData 513279, 3493287, 765.431, 22306.7, pDataArray
949:   AdData 513304, 3493287, 761.846, 22331.2, pDataArray
950:   AdData 513328, 3493287, 758.807, 22355.7, pDataArray
951:   AdData 513353, 3493287, 748.957, 22380.2, pDataArray
952:   AdData 513377, 3493287, 738.423, 22404.7, pDataArray
953:   AdData 513402, 3493287, 727.09, 22429.2, pDataArray
954:   AdData 513420, 3493303, 718.483, 22453.7, pDataArray
955:   AdData 513437, 3493320, 717.346, 22478.2, pDataArray
956:   AdData 513460, 3493324, 719.709, 22502.8, pDataArray
957:   AdData 513484, 3493324, 720.603, 22527.3, pDataArray
958:   AdData 513509, 3493324, 717.384, 22551.8, pDataArray
959:   AdData 513533, 3493324, 711.56, 22576.3, pDataArray
960:   AdData 513557, 3493326, 704.305, 22600.8, pDataArray
961:   AdData 513574, 3493344, 700.692, 22625.3, pDataArray
962:   AdData 513592, 3493361, 704.796, 22649.8, pDataArray
963:   AdData 513616, 3493362, 712.015, 22674.4, pDataArray
964:   AdData 513640, 3493362, 716.293, 22698.9, pDataArray
965:   AdData 513665, 3493362, 716.293, 22723.4, pDataArray
966:   AdData 513683, 3493377, 707.239, 22747.9, pDataArray
967:   AdData 513700, 3493394, 707.056, 22772.4, pDataArray
968:   AdData 513723, 3493400, 716.797, 22796.9, pDataArray
969:   AdData 513747, 3493400, 726.173, 22821.4, pDataArray
970:   AdData 513772, 3493400, 726.173, 22845.9, pDataArray
971:   AdData 513792, 3493410, 721.802, 22870.5, pDataArray
972:   AdData 513809, 3493428, 721.943, 22895, pDataArray
973:   AdData 513829, 3493438, 731.271, 22919.5, pDataArray
974:   AdData 513854, 3493438, 743.225, 22944, pDataArray
975:   AdData 513879, 3493438, 742.929, 22968.5, pDataArray
976:   AdData 513901, 3493444, 739.719, 22993, pDataArray
977:   AdData 513918, 3493461, 742.694, 23017.5, pDataArray
978:   AdData 513935, 3493478, 752.473, 23042, pDataArray
979:   AdData 513953, 3493496, 744.567, 23066.6, pDataArray
980:   AdData 513970, 3493513, 749.246, 23091.1, pDataArray
981:   AdData 513994, 3493513, 753.009, 23115.6, pDataArray
982:   AdData 514016, 3493521, 749.757, 23140.1, pDataArray
983:   AdData 514033, 3493538, 749.683, 23164.6, pDataArray
984:   AdData 514052, 3493551, 757.065, 23189.1, pDataArray
985:   AdData 514077, 3493551, 747.222, 23213.6, pDataArray
986:   AdData 514096, 3493564, 739.06, 23238.1, pDataArray
987:   AdData 514113, 3493581, 742.128, 23262.7, pDataArray
988:   AdData 514131, 3493598, 735.385, 23287.2, pDataArray
```

```
989:   AdData 514148, 3493616, 726.389, 23311.7, pDataArray
990:   AdData 514168, 3493626, 727.192, 23336.2, pDataArray
991:   AdData 514193, 3493626, 722.088, 23360.7, pDataArray
992:   AdData 514211, 3493641, 713.058, 23385.2, pDataArray
993:   AdData 514228, 3493658, 719.138, 23409.7, pDataArray
994:   AdData 514246, 3493676, 711.753, 23434.2, pDataArray
995:   AdData 514263, 3493693, 703.625, 23458.8, pDataArray
996:   AdData 514284, 3493702, 702.413, 23483.3, pDataArray
997:   AdData 514308, 3493702, 696.469, 23507.8, pDataArray
998:   AdData 514326, 3493718, 685, 23532.3, pDataArray
999:   AdData 514343, 3493735, 690.223, 23556.8, pDataArray
1000:   AdData 514361, 3493753, 677.139, 23581.3, pDataArray
1001:   AdData 514378, 3493770, 668.225, 23605.8, pDataArray
1002:   AdData 514400, 3493777, 668.562, 23630.4, pDataArray
1003:   AdData 514424, 3493778, 666.915, 23654.9, pDataArray
1004:   AdData 514441, 3493795, 653.444, 23679.4, pDataArray
1005:   AdData 514458, 3493813, 657.44, 23703.9, pDataArray
1006:   AdData 514476, 3493830, 643.839, 23728.4, pDataArray
1007:   AdData 514493, 3493847, 639.933, 23752.9, pDataArray
1008:   AdData 514515, 3493853, 641.287, 23777.4, pDataArray
1009:   AdData 514539, 3493855, 638.033, 23801.9, pDataArray
1010:   AdData 514556, 3493873, 630.54, 23826.5, pDataArray
1011:   AdData 514574, 3493890, 641.173, 23851, pDataArray
1012:   AdData 514591, 3493907, 625.368, 23875.5, pDataArray
1013:   AdData 514608, 3493925, 625.954, 23900, pDataArray
1014:   AdData 514631, 3493928, 628.317, 23924.5, pDataArray
1015:   AdData 514654, 3493933, 625.564, 23949, pDataArray
1016:   AdData 514671, 3493950, 625.741, 23973.5, pDataArray
1017:   AdData 514689, 3493967, 640.16, 23998, pDataArray
1018:   AdData 514706, 3493985, 629.058, 24022.6, pDataArray
1019:   AdData 514723, 3494002, 638.496, 24047.1, pDataArray
1020:   AdData 514747, 3494004, 633.489, 24071.6, pDataArray
1021:   AdData 514772, 3494004, 623.799, 24096.1, pDataArray
1022:   AdData 514796, 3494004, 611.082, 24120.6, pDataArray
1023:   AdData 514821, 3494004, 599.299, 24145.1, pDataArray
1024:   AdData 514845, 3494004, 588.073, 24169.6, pDataArray
1025:   AdData 514870, 3494004, 577.755, 24194.1, pDataArray
1026:   AdData 514894, 3494004, 558.558, 24218.7, pDataArray
1027:   AdData 514917, 3494001, 536.338, 24243.2, pDataArray
1028:   AdData 514935, 3493983, 520.318, 24267.7, pDataArray
1029:   AdData 514952, 3493966, 513.249, 24292.2, pDataArray
1030:   AdData 514977, 3493966, 499.447, 24316.7, pDataArray
1031:   AdData 514998, 3493958, 480.957, 24341.2, pDataArray
1032:   AdData 515015, 3493941, 464.706, 24365.7, pDataArray
1033:   AdData 515034, 3493928, 458.364, 24390.2, pDataArray
1034:   AdData 515059, 3493928, 455.327, 24414.8, pDataArray
1035:   AdData 515083, 3493928, 441.057, 24439.3, pDataArray
```

```
1036:   AdData 515108, 3493928, 423.452, 24463.8, pDataArray
1037:   AdData 515132, 3493928, 406.696, 24488.3, pDataArray
1038:   AdData 515157, 3493928, 390.828, 24512.8, pDataArray
End Sub
```

Module 4: ErrorHandling

```
Attribute VB_Name = "ErrorHandling"
Option Explicit
'
' FILE AUTOMATICALLY GENERATED BY ESRI ERROR HANDLER ADDIN
' DO NOT EDIT OR REMOVE THIS FILE FROM THE PROJECT
'
Dim pErrorLog As New esriSystemUtility.ErrorDialog


Private Sub DisplayVersion2Dialog(sProcedureName As String, sErrDescription As String)
10:   Beep
11:   MsgBox "An error has occured in the application.   Record the call stack sequence" & vbCrLf & "and the description of the
error." & vbCrLf & vbCrLf & _
         "Error Call Stack Sequence " & vbCrLf & vbTab & sProcedureName & vbCrLf & sErrDescription, vbExclamation + vbOKOnly,
"Unexpected Program Error"
End Sub

Private Sub DisplayVersion3Dialog(sProcedureName As String, sErrDescription As String, parentHWND As Long, raiseException As Boolean)
16:   Beep
17:   MsgBox "An error has occured in the application.   Record the call stack sequence" & vbCrLf & "and the description of the
error." & vbCrLf & vbCrLf & _
         "Error Call Stack Sequence " & vbCrLf & vbTab & sProcedureName & vbCrLf & sErrDescription, vbExclamation + vbOKOnly,
"Unexpected Program Error"
End Sub

Private Sub DisplayVersion4Dialog(sProcedureName As String, sErrDescription As String, parentHWND As Long)
22:   pErrorLog.AppendErrorText "Record Call Stack Sequence - Bottom line is error line." & vbCrLf & vbCrLf & vbTab & sProcedureName &
vbCrLf & sErrDescription
23:   pErrorLog.Visible = True


End Sub

Public Sub HandleError(ByVal bTopProcedure As Boolean, _
                       ByVal sProcedureName As String, _
                       ByVal lErrNumber As Long, _
                       ByVal sErrSource As String, _
                       ByVal sErrDescription As String, _
```

```
                      Optional ByVal version As Long = 1, _
                      Optional ByVal parentHWND As Long = 0, _
                      Optional ByVal reserved1 As Variant = 0, _
                      Optional ByVal reserved2 As Variant = 0, _
                      Optional ByVal reserved3 As Variant = 0)
  ' Generic Error handling Function - This function should be called with
  ' the following Arguments
  '
  ' Boolean    -in-  True if called from a top level procedure - Event / Method / Property
  ' String     -in-  Name of function called from
  ' Long       -in-  Error Number (retrieved from Err object)
  ' String     -in-  Error Source (retrieved from Err object)
  ' String     -in-  Error Description (retrieved from Err object)
  ' Long       -in-  Version of Function (optional Default 1)
  ' parentHWND -in-  Parent Hwnd for error dialogs, NULL is valid
  ' reserved1  -in-
  ' reserved2  -in-
  ' reserved3  -in-


  ' Clear the error object
54:   Err.Clear

  ' Static variable used to control the call stack formatting
57:   Static entered As Boolean

59:   If (bTopProcedure) Then
    ' Top most procedure in call stack so report error to user
    ' Via a dialog
62:     If (Not entered) Then
63:       sErrDescription = vbCrLf & "Error Number " & vbCrLf & vbTab & CStr(lErrNumber) & vbCrLf & "Description" & vbCrLf & vbTab &
sErrDescription & vbCrLf & vbCrLf
64:     End If
65:     entered = False
66:     If (version = 4) Then
67:       DisplayVersion4Dialog sProcedureName, sErrDescription, parentHWND
68:     ElseIf (version = 3) Then
      Dim raiseError As Boolean
70:       DisplayVersion3Dialog sProcedureName, sErrDescription, parentHWND, raiseError
71:       If (raiseError) Then Err.Raise lErrNumber, sErrSource, vbTab & sProcedureName & vbCrLf & sErrDescription
72:     ElseIf (version = 2) Then
73:       DisplayVersion2Dialog sProcedureName, sErrDescription
74:     Else
75:       Beep
76:       MsgBox "An error has occured in the application.   Record the call stack sequence" & vbCrLf & "and the description of the
error." & vbCrLf & vbCrLf & _
            "Error Call Stack Sequence " & vbCrLf & vbTab & sProcedureName & vbCrLf & sErrDescription, vbExclamation + vbOKOnly,
```

```
"Unexpected Program Error"
78:     End If
79:   Else
    ' An error has occured but we are not at the top of the call stack
    ' so append the callstack and raise another error
82:     If (Not entered) Then sErrDescription = vbCrLf & "Error Number " & vbCrLf & vbTab & CStr(lErrNumber) & vbCrLf & "Description"
& vbCrLf & vbTab & sErrDescription & vbCrLf & vbCrLf
83:     entered = True
84:     Err.Raise lErrNumber, sErrSource, vbTab & sProcedureName & vbCrLf & sErrDescription
85:   End If
End Sub

Public Function GetErrorLineNumberString(ByVal lLineNumber As Long) As String
  ' Test the line number if it is non zero create a string
90:   If (lLineNumber <> 0) Then GetErrorLineNumberString = "Line : " & lLineNumber
End Function
```

Module 5: GridFunctions

```
Attribute VB_Name = "GridFunctions"
' GridFunctions:
' Jenness Enterprises
' http://www.jennessent.com
' jeffj@jennessent.com
'--------------------------------------
' AddFieldToVAT - GIVEN A RASTER, FIELDNAME, FIELD TYPE AND FIELD LENGTH, ADDS THE FIELD TO THE VAT.  IF NOT POSSIBLE
'                 (MAYBE USING FLOATING POINT GRID) THEN RETURNS NOTHING. OTHERWISE RETURNS TABLE.
' CalcGridLine - GIVEN pStartPolygon, pEndPolygon, pCorPolygon (FOR BOUNDARY REGION),
'                 pCorRaster As IRaster (FOR COST-DISTANCE), pEnv (IRasterAnalysisEnvironment),
'                 AND OPTIONAL ShouldClean, RETURNS AN IPolyline REPRESENTING LINE FROM START POLYGON TO END POLYGON
' CellValue - GIVEN A POINT AND RASTER, RETURNS CELL VALUE AS VARIANT.  MIGHT NEED TO MODIFY LATER FOR VERY LARGE GRIDS!
' CellValues - GIVEN A POINT COLLECTION AND RASTER, RETURNS CELL VALUES AS VARIANT ARRAY.  MIGHT NEED TO MODIFY LATER FOR VERY LARGE
GRIDS!
' ClipRasterToPolygon - GIVEN pRaster As IRaster, pPolygon As IPolygon, SaveInside As Boolean,
'                 OPTIONAL pClipEnvelope, OPTIONAL CellSize, OPTIONAL pEnv (IRasterAnalysisEnvironment), RETURNS IRaster
' CompareSpatialReferences - GIVEN TWO SPATIAL REFERENCES, RETURNS BOOLEAN
' DistributePointsAlongShape - GIVEN ICurve AND SEPARATION DISTANCE, RETURNS A MULTIPOINT OF POINTS ALONG THAT SHAPE
' EllipticArcToPolygon - GIVEN AN ISegmentCollection AND NumVertices, RETURNS Polygon4 BASED ON CONVEX HULL AROUND VERTICES
' EllipticArcToPolygon2 - GIVEN AN ISegmentCollection AND NumVertices, RETURNS IMultipoint
' OpenRasterWorkspace - GIVEN sPath, RETURNS A RASTER WORKSPACE
' ReturnCellCount - GIVEN A RASTER, RETURNS COUNT OF NON-NULL CELLS
' ReturnCellSize - GIVEN A RASTER, RETURNS CELL SIZE BASED ON Y-DIMENSION
' ReturnPixelHeight - GIVEN A RASTER, RETURNS PIXEL HEIGHT (Y-DIMENSION)
' ReturnPixelWidth - GIVEN A RASTER, RETURNS PIXEL WIDTH (X-DIMENSION)
' ReturnPointsByCellSize - GIVEN A RASTER AND LINE, RETURNS IPointCollection WITH POINTS DISTRIBUTED APPROXIMATELY EQUAL TO
'                 CELL SIZE
```

```
' SetSpatialAnalysisSettings - GIVEN TargetEnv AND SourceEnv, ASSIGNS PROPERTIES OF SourceEnv TO TargetEnv
' SaveRasterAs -  GIVEN A RASTER BAND COLLECTION, FOLDER PATH, RASTER FILENAME, AND RASTER TYPE, SAVES RASTER AS PERMANENT FILE

Public Enum enumRasterType
   enum_Grid_Type
   enum_Imagine_Type
   enum_TIFF_Type
   enum_JPEG_Type
   enum_JP2000_Type
   enum_BMP_Type
   enum_PNG_Type
   enum_GIF_Type
   enum_PCI_Raster_Type
   enum_X11_Pixmap_Type
   enum_PCRaster_Type
   enum_Memory_Raster_Type
   enum_HDF4_Type
   enum_BIL_Type
   enum_BIP_Type
   enum_BSQ_Type
   enum_IDRISI_Type
   enum_Geodatabase_Type
End Enum

Option Explicit
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\GridFunctions.bas"


Public Function OpenRasterWorkspace(sPath As String) As IRasterWorkspace
  On Error GoTo ErrorHandler

  Dim pWKSF As IWorkspaceFactory
58:   Set pWKSF = New RasterWorkspaceFactory

  Dim pRasterWs As IRasterWorkspace
61:   Set pRasterWs = pWKSF.OpenFromFile(sPath, 0)
62:   Set OpenRasterWorkspace = pRasterWs

  Exit Function
ErrorHandler:
  HandleError True, "OpenRasterWorkspace " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Public Sub SaveRasterAs(pRasterBandCol As IRasterBandCollection, strPath As String, strName As String, aRasterType As enumRasterType)
  On Error GoTo ErrorHandler
'  MsgBox "GridFunctions - Before: " & strPath
'  strPath = Linkages.aml_func_mod.BasicTrimAvenue(strPath, "", "/\")
```

```
'  MsgBox "GridFunctions - After: " & strPath

  Dim pSaveAs As IRasterBandCollection
75:   Set pSaveAs = pRasterBandCol
  Dim pRasterWs As IRasterWorkspace
77:   Set pRasterWs = OpenRasterWorkspace(strPath)
78:   If aRasterType = enum_Grid_Type Then
79:     pSaveAs.SaveAs strName, pRasterWs, "GRID"
80:   ElseIf aRasterType = enum_Imagine_Type Then
81:     pSaveAs.SaveAs strName, pRasterWs, "IMAGINE Image"
82:   ElseIf aRasterType = enum_TIFF_Type Then
83:     pSaveAs.SaveAs strName, pRasterWs, "TIFF"
84:   ElseIf aRasterType = enum_JPEG_Type Then
85:     pSaveAs.SaveAs strName, pRasterWs, "JPG"
86:   ElseIf aRasterType = enum_JP2000_Type Then
87:     pSaveAs.SaveAs strName, pRasterWs, "JP2"
88:   ElseIf aRasterType = enum_BMP_Type Then
89:     pSaveAs.SaveAs strName, pRasterWs, "BMP"
90:   ElseIf aRasterType = enum_PNG_Type Then
91:     pSaveAs.SaveAs strName, pRasterWs, "PNG"
92:   ElseIf aRasterType = enum_GIF_Type Then
93:     pSaveAs.SaveAs strName, pRasterWs, "GIF"
94:   ElseIf aRasterType = enum_PCI_Raster_Type Then
95:     pSaveAs.SaveAs strName, pRasterWs, "PIX"
96:   ElseIf aRasterType = enum_X11_Pixmap_Type Then
97:     pSaveAs.SaveAs strName, pRasterWs, "XPM"
98:   ElseIf aRasterType = enum_PCRaster_Type Then
99:     pSaveAs.SaveAs strName, pRasterWs, "MAP"
100:  ElseIf aRasterType = enum_Memory_Raster_Type Then
101:    pSaveAs.SaveAs strName, pRasterWs, "MEM"
102:  ElseIf aRasterType = enum_HDF4_Type Then
103:    pSaveAs.SaveAs strName, pRasterWs, "HDF4"
104:  ElseIf aRasterType = enum_BIL_Type Then
105:    pSaveAs.SaveAs strName, pRasterWs, "BIL"
106:  ElseIf aRasterType = enum_BIP_Type Then
107:    pSaveAs.SaveAs strName, pRasterWs, "BIP"
108:  ElseIf aRasterType = enum_BSQ_Type Then
109:    pSaveAs.SaveAs strName, pRasterWs, "BSQ"
110:  ElseIf aRasterType = enum_IDRISI_Type Then
111:    pSaveAs.SaveAs strName, pRasterWs, "GDB"
112:  ElseIf aRasterType = enum_Geodatabase_Type Then
113:    pSaveAs.SaveAs strName, pRasterWs, "GDB"
114:  End If


  Exit Sub
ErrorHandler:
```

```
  HandleError True, "SaveRasterAs " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Function ReturnCellSize(pRaster As IRaster) As Double
  On Error GoTo erh
  Dim pRasLayer As IRasterLayer
124:   Set pRasLayer = New RasterLayer
125:   pRasLayer.CreateFromRaster pRaster

  Dim pRasterProps As IRasterProps
128:   Set pRasterProps = pRaster

  Dim lngNumRows As Long
131:   lngNumRows = pRasLayer.RowCount

  Dim pEnvelope As IEnvelope
134:   Set pEnvelope = pRasterProps.Extent

136:   ReturnCellSize = pEnvelope.Height / lngNumRows

'  Debug.Print "By Rows = " & ReturnCellSize & ",   By Columns = " & pEnvelope.Width / pRasLayer.ColumnCount

  Exit Function
erh:
142:     MsgBox "Failed in ReturnCellSize: " & Err.Description

End Function
Public Function ReturnPixelHeight(pRaster As IRaster) As Double
  On Error GoTo erh
  Dim pRasLayer As IRasterLayer
148:   Set pRasLayer = New RasterLayer
149:   pRasLayer.CreateFromRaster pRaster

  Dim pRasterProps As IRasterProps
152:   Set pRasterProps = pRaster

  Dim lngNumRows As Long
155:   lngNumRows = pRasLayer.RowCount

  Dim pEnvelope As IEnvelope
158:   Set pEnvelope = pRasterProps.Extent

160:   ReturnPixelHeight = pEnvelope.Height / lngNumRows

'  Debug.Print "By Rows = " & ReturnCellSize & ",   By Columns = " & pEnvelope.Width / pRasLayer.ColumnCount

  Exit Function
```

```
erh:
  HandleError True, "ReturnPixelHeight " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4


End Function

Public Function ReturnCellCount(pRaster As IRaster) As Long
  On Error GoTo ErrorHandler







  Exit Function
ErrorHandler:
  HandleError True, "ReturnCellCount " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function CalcContinuousGridStats(pRaster As IRaster, pRasterStats As IRasterStatistics, _
      lngNumBins As Long) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler


  Dim pRasterAnalysisProps As IRasterAnalysisProps
  Dim pRasterProps As IRasterProps
191:   Set pRasterAnalysisProps = pRaster
192:   Set pRasterProps = pRaster

  Dim dblMaximum As Double
  Dim dblMinimum As Double
  Dim dblMean As Double
  Dim dblMedian As Double
  Dim dblMode As Double
  Dim dblStDev As Double







  Exit Function
ErrorHandler:
  HandleError True, "CalcContinuousGridStats " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Function

Public Function ReturnPixelWidth(pRaster As IRaster) As Double
  On Error GoTo erh
  Dim pRasLayer As IRasterLayer
214:   Set pRasLayer = New RasterLayer
215:   pRasLayer.CreateFromRaster pRaster

  Dim pRasterProps As IRasterProps
218:   Set pRasterProps = pRaster

  Dim lngNumCols As Long
221:   lngNumCols = pRasLayer.ColumnCount

  Dim pEnvelope As IEnvelope
224:   Set pEnvelope = pRasterProps.Extent

226:   ReturnPixelWidth = pEnvelope.Width / lngNumCols

'  Debug.Print "By Rows = " & ReturnCellSize & ",   By Columns = " & pEnvelope.Width / pRasLayer.ColumnCount

  Exit Function
erh:
  HandleError True, "ReturnPixelWidth " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4

End Function

Public Function ReturnPointsByCellSize(pRaster As IRaster, ByVal pLine As IGeometry) As IPointCollection
  On Error GoTo erh

  Dim pCurve As ICurve
  Dim dblLength As Double

242:   If TypeOf pLine Is ICurve Then
243:     Set pCurve = pLine
244:     dblLength = pCurve.length
245:   Else
246:     MsgBox "Invalid geometry type!  Must implement 'ICurve'..."
247:     Set ReturnPointsByCellSize = Nothing
    Exit Function
249:   End If
  Dim pPointCollection As IPointCollection

  ' CHECK SPATIAL REFERENCES; MIGHT NEED TO PROJECT POLYLINE
  Dim pSrcSpRef As ISpatialReference
  Dim pRasProps As IRasterProps
```

```
255:   Set pRasProps = pRaster
256:   Set pSrcSpRef = pRasProps.SpatialReference

  Dim pTrgSpRef As ISpatialReference
259:   Set pTrgSpRef = pLine.SpatialReference

261:   If Not GridFunctions.CompareSpatialReferences(pSrcSpRef, pTrgSpRef) Then
262:     pLine.Project pSrcSpRef
263:   End If

  ' GET GRID CELL SIZE
  Dim dblCellSize As Double
267:   dblCellSize = GridFunctions.ReturnCellSize(pRaster)
  Dim NumPoints As Long
269:   NumPoints = Int(dblLength / dblCellSize) + 1

  Dim pMpt As IMultipoint
  Dim pSegCol As ISegmentCollection
273:   Set pSegCol = pLine
274:   Set pMpt = GridFunctions.EllipticArcToPolygon2(pSegCol, NumPoints)
275:   Set pPointCollection = pMpt

277:   Set ReturnPointsByCellSize = pPointCollection

  Exit Function
erh:
281:     MsgBox "Failed in ReturnPointsByCellSize: " & Err.Description

End Function
Public Function CompareSpatialReferences(ByVal pSourceSR As ISpatialReference, ByVal pTargetSR As ISpatialReference) As Boolean

  On Error GoTo erh

288:   If pSourceSR Is Nothing And pTargetSR Is Nothing Then
289:     CompareSpatialReferences = True
    Exit Function
291:   ElseIf pSourceSR Is Nothing Or pTargetSR Is Nothing Then
292:     CompareSpatialReferences = False
    Exit Function
294:   End If

  Dim pSourceClone As IClone
  Dim pTargetClone As IClone
  Dim bSREqual As Boolean

300:   Set pSourceClone = pSourceSR
301:   Set pTargetClone = pTargetSR
```

```
  'Compare the coordinate system component of the spatial reference
304:   bSREqual = pSourceClone.IsEqual(pTargetClone)

  'If the comparison failed, return false and exit
307:   If Not bSREqual Then
308:     CompareSpatialReferences = False
    Exit Function
310:   End If

  'We can also compare the XY precision to ensure the spatial references are equal
  Dim pSourceSR2 As ISpatialReference2
  Dim bXYIsEqual As Boolean

316:   Set pSourceSR2 = pSourceSR
317:   bXYIsEqual = pSourceSR2.IsXYPrecisionEqual(pTargetSR)

  'If the comparison failed, return false and exit
320:   If Not bXYIsEqual Then
321:     CompareSpatialReferences = False
    Exit Function
323:   End If

325:   CompareSpatialReferences = True
  Exit Function
erh:
328:     MsgBox "Failed in CompareSpatialReferences: " & Err.Description
End Function
Sub SetSpatialAnalysisSettings(TargetEnv As IRasterAnalysisEnvironment, _
                               SourceEnv As IRasterAnalysisEnvironment)
    On Error GoTo erh
333:     If Not SourceEnv Is Nothing Then
334:         Set TargetEnv.OutWorkspace = SourceEnv.OutWorkspace
335:         If Not SourceEnv.OutSpatialReference Is Nothing Then
336:             Set TargetEnv.OutSpatialReference = SourceEnv.OutSpatialReference
337:         End If
338:         TargetEnv.DefaultOutputRasterPrefix = SourceEnv.DefaultOutputRasterPrefix
339:         TargetEnv.DefaultOutputVectorPrefix = SourceEnv.DefaultOutputVectorPrefix
340:         If Not SourceEnv.Mask Is Nothing Then
341:             Set TargetEnv.Mask = SourceEnv.Mask
342:         End If
        Dim nCellSize As Double
344:         SourceEnv.GetCellSize 3, nCellSize
345:         If nCellSize <> 0 Then
346:             TargetEnv.SetCellSize 3, nCellSize
347:         End If
        Dim pExtent As IEnvelope
```

```
349:         SourceEnv.GetExtent 3, pExtent
350:         If Not pExtent Is Nothing Then
351:             TargetEnv.SetExtent 3, pExtent
352:         End If
353:         TargetEnv.VerifyType = SourceEnv.VerifyType
354:     End If
    Exit Sub
erh:
357:     MsgBox "Failed in SetSpatialAnalysisSettings: " & Err.Description
End Sub

Public Function ClipRasterToPolygon(pRaster As IRaster, ByVal pPolygon As IPolygon, SaveInside As Boolean, _
      Optional ByVal pClipEnvelope As IEnvelope, Optional CellSize As Double, _
      Optional ByVal pEnv As IRasterAnalysisEnvironment, Optional booShowProgress As Boolean, _
      Optional pApp As IApplication) As IRaster

  On Error GoTo ErrorHandler


'  If pPolygon.IsEmpty Then
'    Dim xx As Long
'    xx = 1
'  End If

    ' PROGRESS BAR STUFF
374:   If booShowProgress Then
    Dim psbar As IStatusBar
376:     Set psbar = pApp.StatusBar
377:     psbar.ProgressBar.position = 1
    Dim pPro As IStepProgressor
379:     Set pPro = psbar.ProgressBar
380:   End If

'  MsgBox "Hello 1"

  ' FORCE OBJECT TO BE POLYGON IF THE FIRST SEGMENT IS A CURVE
  Dim booWorkingWithCurves As Boolean
  Dim pSegmentCollectionCurves As ISegmentCollection
'  Dim pSegTopoOp As ITopologicalOperator
'  Set pSegTopoOp = pPolygon
'  Dim pCheckSegLine As IPolyline
'  Set pCheckSegLine = pSegTopoOp.Boundary
391:   Set pSegmentCollectionCurves = pPolygon
  Dim pSegmentCurve As ISegment
393:   Set pSegmentCurve = pSegmentCollectionCurves.Segment(0)
  Dim pGeometryTypeA As esriGeometryType
395:   pGeometryTypeA = pSegmentCurve.GeometryType
```

```
397:   booWorkingWithCurves = (pGeometryTypeA = esriGeometryBezier3Curve) Or _
    (pGeometryTypeA = esriGeometryCircularArc) Or _
    (pGeometryTypeA = esriGeometryEllipticArc)

401:   If booWorkingWithCurves Then
    Dim pNewPoints As IPointCollection
403:     Set pNewPoints = GridFunctions.EllipticArcToPolygon2(pSegmentCollectionCurves, 100)
    Dim pNewPolygon As IPointCollection
405:     Set pNewPolygon = New Polygon
406:     pNewPolygon.SetPointCollection pNewPoints
407:     Set pPolygon = pNewPolygon
408:   End If


'  MsgBox "Hello 2"

'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument

  ' MAKE EXTRACTION OPERATOR
  Dim pExtractionOp As IExtractionOp
418:   Set pExtractionOp = New RasterExtractionOp

  Dim pRastAnalysisEnv As IRasterAnalysisEnvironment
  Dim pSpatialReference As ISpatialReference
  Dim pPolySpatRef As ISpatialReference
423:   Set pPolySpatRef = pPolygon.SpatialReference
  Dim SpatRefSame As Boolean
  Dim pEnvelope As IEnvelope

427:   If Not pEnv Is Nothing Then
428:     Set pRastAnalysisEnv = pEnv
429:     Set pSpatialReference = pRastAnalysisEnv.OutSpatialReference

    ' CHECK SPATIAL REFERENCE OF INCOMING POLYGON
432:     SpatRefSame = CompareSpatialReferences(pSpatialReference, pPolySpatRef)
433:     If Not SpatRefSame Then pPolygon.Project pSpatialReference
434:     pEnv.GetExtent esriRasterEnvValue, pEnvelope

436:   Else

    ' MAKE ANALYSIS ENVIRONMENT
439:     Set pRastAnalysisEnv = pExtractionOp
440:     pRastAnalysisEnv.RestoreToPreviousDefaultEnvironment

    ' ASSIGN CURRENT ANALYSIS ENVIRONMENT SETTINGS
```

```
    Dim theEnvType As esriRasterEnvSettingEnum
    Dim theExtEnvType As esriRasterEnvSettingEnum
    Dim pTempEnv As IEnvelope
    Dim theCellSize As Double
447:     pRastAnalysisEnv.GetCellSize theEnvType, theCellSize
448:     pRastAnalysisEnv.GetExtent theExtEnvType, pTempEnv
    Dim pRasterAnalysisProps As IRasterAnalysisProps
    Dim pRasterProps As IRasterProps
451:         Set pRasterAnalysisProps = pRaster
452:         Set pRasterProps = pRaster

    ' DETERMINE OUTPUT SPATIAL REFERENCE BASED ON INPUT RASTER

456:     Set pSpatialReference = pRasterProps.SpatialReference

    ' CHECK SPATIAL REFERENCE OF INCOMING POLYGON
459:     SpatRefSame = CompareSpatialReferences(pSpatialReference, pPolySpatRef)
460:     If Not SpatRefSame Then pPolygon.Project pSpatialReference

    ' DETERMINE ANALYSIS ENVIRONMENT TYPE AND CELL SIZE
463:     If CellSize <= 0 Then
464:       theCellSize = pRasterAnalysisProps.PixelHeight
465:       theEnvType = pRastAnalysisEnv.VerifyType
466:     Else
467:       theCellSize = CellSize
468:       theEnvType = esriRasterEnvValue
469:     End If

    Dim pTopoOp As ITopologicalOperator2
472:     If pClipEnvelope Is Nothing Then
      Dim pPolyEnvelope As IEnvelope
474:           Set pPolyEnvelope = pPolygon.Envelope
      Dim pRastEnvelope As IEnvelope
476:           Set pRastEnvelope = pRasterProps.Extent
477:           Set pTopoOp = pPolyEnvelope
478:           pTopoOp.IsKnownSimple = False
479:           pTopoOp.Simplify

481:       Set pEnvelope = pTopoOp.Intersect(pPolyEnvelope, esriGeometry2Dimension)
482:     Else
      ' CHECK SPATIAL REFERENCE OF ENVELOPE
484:       SpatRefSame = CompareSpatialReferences(pSpatialReference, pClipEnvelope.SpatialReference)
485:       If Not SpatRefSame Then
486:         pClipEnvelope.Project pSpatialReference
487:       End If
488:       Set pEnvelope = pClipEnvelope.Envelope
489:     End If
```

```
    ' SET ANALYSIS ENVIRONMENT PROPERTIES
492:     pRastAnalysisEnv.SetCellSize theEnvType, theCellSize
493:     pRastAnalysisEnv.SetExtent esriRasterEnvValue, pEnvelope
494:     Set pRastAnalysisEnv.OutSpatialReference = pSpatialReference
495:   End If


'  MsgBox "Hello 3"
499:   DoEvents

  ' CLIP INCOMING POLYGON TO ANALYSIS AREA; MIGHT HELP AVOID PROBLEMS WITH MULTIPART POLYGONS, HOLES, ETC.
  Dim pIntPolygon As IPolygon4
503:   Set pTopoOp = pPolygon
504:   pTopoOp.IsKnownSimple = False
505:   pTopoOp.Simplify
506:   Set pIntPolygon = pTopoOp.Intersect(pEnvelope, esriGeometry2Dimension)
507:   Set pIntPolygon = pPolygon
508:   Set pIntPolygon.SpatialReference = pPolygon.SpatialReference

'  ThisDocument.Graphic_MakeFromGeometry pMxDoc, pIntPolygon, "TestClipPolyGraphics"


513:   Set pTopoOp = pIntPolygon
514:   pTopoOp.IsKnownSimple = False
515:   pTopoOp.Simplify

  Dim pGeometryCollection As IGeometryCollection
  Dim pExtRing As IGeometryCollection
  Dim pIntRingBag As IGeometryCollection
520:   Set pGeometryCollection = pIntPolygon.ConnectedComponentBag
  Dim pIntGeoCol As IGeometryCollection
  Dim pIntPoly As IPolygon4
  Dim pOutGeoCol As IGeometryCollection
  Dim pOutPoly As IPolygon4

  Dim pSubPoly As IPolygon4
  Dim pSubRing As IRing
  Dim anIndex As Long
  Dim anIndex2 As Long

  Dim pClipRaster As IRaster
  Dim pOuterRaster As IRaster
  Dim pInnerRaster As IRaster

  ' MAKE A BUNCH OF NEW RASTER OPERATORS AND ASSIGN THE CURRENT ANALYSIS ENVIRONMENT
  Dim pRasMakerOp As IRasterMakerOp
```

```
537:   Set pRasMakerOp = New RasterMakerOp
538:   SetSpatialAnalysisSettings pRasMakerOp, pRastAnalysisEnv
  Dim pCondOp As IConditionalOp
540:   Set pCondOp = New RasterConditionalOp
541:   SetSpatialAnalysisSettings pCondOp, pRastAnalysisEnv
  Dim pLogicOp As ILogicalOp
543:   Set pLogicOp = New RasterMathOps
544:   SetSpatialAnalysisSettings pLogicOp, pRastAnalysisEnv
  Dim pMathOp As IMathOp
546:   Set pMathOp = New RasterMathOps
547:   SetSpatialAnalysisSettings pMathOp, pRastAnalysisEnv

  Dim pRasLayer As IRasterLayer
  Dim pTestGeometry As IGeometry
  Dim pTestGeoColl As IGeometryCollection
  Dim pSegmentCollection1 As ISegmentCollection
  Dim pSegment1 As ISegment
  Dim pGeometryType As esriGeometryType
  Dim pEllArcPolygon As IPolygon4

  Dim pFinalGrid As IRaster
558:   Set pFinalGrid = pRasMakerOp.MakeConstant(0, True)
559:   Set pClipRaster = pRasMakerOp.MakeConstant(1, True)

561:   If booShowProgress Then
562:     pPro.MaxRange = pGeometryCollection.GeometryCount + 2
563:     pPro.StepValue = 1
564:     pPro.Show
565:   End If

567:   DoEvents
'  MsgBox "Hello 4"

570:   For anIndex = 0 To pGeometryCollection.GeometryCount - 1

572:     Set pSubPoly = pGeometryCollection.Geometry(anIndex)
573:     Set pExtRing = pSubPoly.ExteriorRingBag
574:     Set pSubRing = pExtRing.Geometry(0)
575:     Set pOutGeoCol = New Polygon
576:     pOutGeoCol.AddGeometry pSubRing
577:     Set pOutPoly = pOutGeoCol
578:     Set pTopoOp = pOutPoly
579:     pTopoOp.IsKnownSimple = False
580:     pTopoOp.Simplify

'    Debug.Print anIndex & ":  Outer ring of Polygon is Circle?  " & CStr(TypeOf pOutPoly Is ICircularArc)
```

```
584:     Set pSegmentCollection1 = pOutPoly
585:     Set pSegment1 = pSegmentCollection1.Segment(0)
586:     pGeometryType = pSegment1.GeometryType

588:     If pGeometryType = esriGeometryCircularArc Then
589:       Set pOuterRaster = pExtractionOp.Circle(pClipRaster, pSegment1, Not SaveInside)
590:     ElseIf pGeometryType = esriGeometryEllipticArc Then
591:       Set pEllArcPolygon = EllipticArcToPolygon(pSegmentCollection1, 75)
592:       Set pOuterRaster = pExtractionOp.Polygon(pClipRaster, pEllArcPolygon, Not SaveInside)
593:     ElseIf pGeometryType = esriGeometryEnvelope Then
594:       Set pOuterRaster = pExtractionOp.Rectangle(pClipRaster, pOutPoly, Not SaveInside)
595:     Else
596:       Set pOuterRaster = pExtractionOp.Polygon(pClipRaster, pOutPoly, Not SaveInside)
597:     End If
598:     Set pOuterRaster = pLogicOp.IsNull(pOuterRaster)

600:     If pSubPoly.InteriorRingCount(pSubRing) > 0 Then
601:       Set pIntRingBag = pSubPoly.InteriorRingBag(pSubRing)

603:       For anIndex2 = 0 To pIntRingBag.GeometryCount - 1
604:         Set pIntGeoCol = New Polygon
605:         pIntGeoCol.AddGeometry pIntRingBag.Geometry(anIndex2)
606:         Set pIntPoly = pIntGeoCol
607:         Set pTopoOp = pIntPoly
608:         pTopoOp.IsKnownSimple = False
609:         pTopoOp.Simplify

611:         Set pSegmentCollection1 = pIntPoly
612:         Set pSegment1 = pSegmentCollection1.Segment(0)
613:         pGeometryType = pSegment1.GeometryType

       ' CHECK FOR UNUSUAL SHAPES
616:         If pGeometryType = esriGeometryCircularArc Then
617:           Set pInnerRaster = pExtractionOp.Circle(pClipRaster, pSegment1, SaveInside)
618:         ElseIf pGeometryType = esriGeometryEllipticArc Then
619:           Set pEllArcPolygon = EllipticArcToPolygon(pSegmentCollection1, 75)
620:           Set pInnerRaster = pExtractionOp.Polygon(pClipRaster, pEllArcPolygon, SaveInside)
621:         ElseIf pGeometryType = esriGeometryEnvelope Then
622:           Set pInnerRaster = pExtractionOp.Rectangle(pClipRaster, pIntPoly, SaveInside)
623:         Else
624:           Set pInnerRaster = pExtractionOp.Polygon(pClipRaster, pIntPoly, SaveInside)
625:         End If
626:         Set pInnerRaster = pLogicOp.IsNull(pInnerRaster)
627:         Set pOuterRaster = pMathOp.Times(pInnerRaster, pOuterRaster)
628:         DoEvents
629:       Next anIndex2
```

```
631:     End If

633:     Set pFinalGrid = pMathOp.Plus(pFinalGrid, pOuterRaster)
634:     If booShowProgress Then
635:       pPro.Step
636:     End If
637:     DoEvents
638:   Next anIndex

  ' FOR DEBUGGING

'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
'  Dim pMap As IMap
'  Set pMap = pMxDoc.FocusMap
'  Set pRasLayer = New RasterLayer
'  pRasLayer.CreateFromRaster pFinalGrid
'  pMap.AddLayer pRasLayer

650:   Set pFinalGrid = pCondOp.SetNull(pLogicOp.EqualTo(pFinalGrid, pRasMakerOp.MakeConstant(0, True)), pFinalGrid)
651:   If booShowProgress Then
652:     pPro.Step
653:   End If
654:   Set ClipRasterToPolygon = pMathOp.Times(pFinalGrid, pRaster)
655:   If booShowProgress Then
656:     pPro.Step
657:   End If

659:   DoEvents

  ' RESET ANALYSIS ENVIRONMENT TO PREVIOUS STATE
662:   pRastAnalysisEnv.RestoreToPreviousDefaultEnvironment

664:   If booShowProgress Then
665:     pPro.Hide
666:   End If

  Exit Function
ErrorHandler:
  HandleError True, "ClipRasterToPolygon " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function EllipticArcToPolygon(SegCollection As ISegmentCollection, NumVertices As Long) As IPolygon4
'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
```

```
'  Dim pEllArc As IEllipticArc

On Error GoTo erh

  Dim pCurve As ICurve
  Dim pGeometry As IGeometry

  Dim anIndex As Long
  Dim lngSegCount As Long
687:   lngSegCount = SegCollection.SegmentCount - 1
  Dim theLength As Double
689:   theLength = 0
  Dim theTestLength As Double
  Dim lngLengths() As Long
  ReDim lngLengths(lngSegCount)
693:   For anIndex = 0 To lngSegCount
694:     theTestLength = SegCollection.Segment(anIndex).length
695:     theLength = theLength + theTestLength
696:     lngLengths(anIndex) = theTestLength
697:   Next anIndex

  Dim pProportion As Double
  Dim lngVertices() As Long
  Dim lngNumVertices As Long
  ReDim lngVertices(lngSegCount)
703:   For anIndex = 0 To lngSegCount
704:     lngNumVertices = Int((lngLengths(anIndex) / theLength) * NumVertices)
705:     If lngNumVertices < 8 Then lngNumVertices = 8
706:     lngVertices(anIndex) = lngNumVertices
707:   Next anIndex

  Dim pMpt As IPointCollection
710:   Set pMpt = New Multipoint
  Dim pPoint As IPoint
712:   Set pPoint = New Point
  Dim pClone As IClone

  Dim pRatio As Double
  Dim anIndex2 As Long

718:   For anIndex = 0 To lngSegCount
719:     lngNumVertices = lngVertices(anIndex)
720:     pRatio = 1 / lngNumVertices
721:     Set pCurve = SegCollection.Segment(anIndex)

723:     For anIndex2 = 0 To lngNumVertices
```

```
'      If pGeometry.GeometryType = esriGeometryEllipticArc Then
725:       pCurve.QueryPoint 0, (pRatio * anIndex2), True, pPoint
726:       Set pClone = pPoint

 '   Graphic_MakeFromGeometry pMxDoc, pPoint, "DeleteMe"

730:       pMpt.AddPoint pClone.Clone
731:     Next anIndex2
732:   Next anIndex

  Dim pPoly4 As IPolygon4
  Dim pTopoOp2 As ITopologicalOperator2
  Dim pTopoOp3 As ITopologicalOperator3
737:   Set pTopoOp2 = pMpt
738:   Set pPoly4 = pTopoOp2.ConvexHull
739:   Set pTopoOp3 = pPoly4
740:   pTopoOp3.IsKnownSimple = False
741:   pTopoOp3.Simplify

743:   Set EllipticArcToPolygon = pPoly4
    Exit Function

erh:
747:     MsgBox "Failed in EllipticArcToPolygon: " & Err.Description
End Function

Public Function EllipticArcToPolygon2(SegCollection As ISegmentCollection, NumVertices As Long) As IMultipoint
'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument

'  Dim pEllArc As IEllipticArc

On Error GoTo erh

  Dim pCurve As ICurve
  Dim pGeometry As IGeometry

  Dim anIndex As Long
  Dim lngSegCount As Long
763:   lngSegCount = SegCollection.SegmentCount - 1
  Dim theLength As Double
765:   theLength = 0
  Dim theTestLength As Double
  Dim lngLengths() As Long
  ReDim lngLengths(lngSegCount)
769:   For anIndex = 0 To lngSegCount
770:     theTestLength = SegCollection.Segment(anIndex).length
```

```
771:     theLength = theLength + theTestLength
772:     lngLengths(anIndex) = theTestLength
773:   Next anIndex

  Dim pProportion As Double
  Dim lngVertices() As Long
  Dim lngNumVertices As Long
  ReDim lngVertices(lngSegCount)
779:   For anIndex = 0 To lngSegCount
780:     lngNumVertices = Int((lngLengths(anIndex) / theLength) * NumVertices)
781:     If lngNumVertices < 8 Then lngNumVertices = 8
782:     lngVertices(anIndex) = lngNumVertices
783:   Next anIndex

  Dim pMpt As IPointCollection
786:   Set pMpt = New Multipoint
  Dim pPoint As IPoint
788:   Set pPoint = New Point
  Dim pClone As IClone

  Dim pRatio As Double
  Dim anIndex2 As Long

794:   For anIndex = 0 To lngSegCount
795:     lngNumVertices = lngVertices(anIndex)
796:     pRatio = 1 / lngNumVertices
797:     Set pCurve = SegCollection.Segment(anIndex)

799:     For anIndex2 = 0 To lngNumVertices
'      If pGeometry.GeometryType = esriGeometryEllipticArc Then
801:       pCurve.QueryPoint 0, (pRatio * anIndex2), True, pPoint
802:       Set pClone = pPoint

 '   Graphic_MakeFromGeometry pMxDoc, pPoint, "DeleteMe"

806:       pMpt.AddPoint pClone.Clone
807:     Next anIndex2
808:   Next anIndex

810:   Set EllipticArcToPolygon2 = pMpt
    Exit Function

erh:
814:     MsgBox "Failed in EllipticArcToPolygon2: " & Err.Description
End Function

Public Function DistributePointsAlongShape(pCurve As ICurve, SeparationDistance As Double) As IPointCollection
```

```
On Error GoTo erh
'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
'  Dim pGeometry As IGeometry

  Dim anIndex As Long
  Dim theLength As Double
826:   theLength = pCurve.length

  Dim pMpt As IPointCollection
829:   Set pMpt = New Multipoint
  Dim pPoint As IPoint
831:   Set pPoint = New Point
  Dim pClone As IClone

  Dim dblRatio As Double
835:   dblRatio = SeparationDistance / theLength

  Dim theCurrentDist As Double
838:   theCurrentDist = 0

840:   Do While theCurrentDist < 1
841:     pCurve.QueryPoint esriNoExtension, theCurrentDist, True, pPoint
842:     Set pClone = pPoint
843:     pMpt.AddPoint pClone.Clone
'    ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPoint, "TestClipPolyGraphics"
845:     theCurrentDist = theCurrentDist + dblRatio
846:   Loop
847:   pCurve.QueryPoint esriNoExtension, 1, True, pPoint
848:   Set pClone = pPoint
849:   pMpt.AddPoint pClone.Clone

  Dim pGeometry As IGeometry
852:   Set pGeometry = pMpt
853:   Set pGeometry.SpatialReference = pCurve.SpatialReference

855:   Set DistributePointsAlongShape = pMpt

    Exit Function

erh:
860:     MsgBox "Failed in DistributePointsAlongShape: " & Err.Description
End Function

Public Function CalcGridLine(ByVal pStartPolygon As IPolygon, ByVal pEndPolygon As IPolygon, _
      ByVal pCorPolygon As IPolygon, pCorRaster As IRaster, pEnv As IRasterAnalysisEnvironment, _
```

```
      Optional ShouldClean As Boolean) As IPolyline
  On Error GoTo ErrorHandler


'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
  Dim pClone As IClone

  Dim pEnvelope As IEnvelope
874:   pEnv.GetExtent esriRasterEnvValue, pEnvelope
'  Set pEnvelope = pEnv.GetExtent

  'Create a RasterMakerOp operator
  Dim pRasMakerOp As IRasterMakerOp
879:   Set pRasMakerOp = New RasterMakerOp
880:   GridFunctions.SetSpatialAnalysisSettings pRasMakerOp, pEnv

  Dim pSpRef As ISpatialReference
883:   Set pSpRef = pEnv.OutSpatialReference

  ' SET POLYGON SPATIAL REFERENCES; PROJECT THEM IF NECESSARY
886:   If Not (GridFunctions.CompareSpatialReferences(pStartPolygon.SpatialReference, pSpRef)) Then
887:     pStartPolygon.Project pSpRef
888:   End If
889:   If Not (GridFunctions.CompareSpatialReferences(pEndPolygon.SpatialReference, pSpRef)) Then
890:     pEndPolygon.Project pSpRef
891:   End If

  ' GENERATE A SERIES OF POINTS AROUND THE BOUNDARY OF THE END POLYGON
  Dim pPoints As IPointCollection
895:   Set pPoints = GridFunctions.ReturnPointsByCellSize(pCorRaster, pEndPolygon)
  Dim anIndex As Long
'  For anIndex = 0 To pPoints.PointCount - 1
'    ThisDocument.Graphic_MakeFromGeometry pMxDoc, pPoints.Point(anIndex), "DeleteMatrix"
'  Next anIndex

  ' Create the RasterDistanceOp object
  Dim pDistanceOp As IDistanceOp
903:   Set pDistanceOp = New RasterDistanceOp

  ' SET ANALYSIS ENVIRONMENT
906:   SetSpatialAnalysisSettings pDistanceOp, pEnv

  ' Declare the input source raster object
  Dim pBaseRaster As IRaster
910:   Set pBaseRaster = pRasMakerOp.MakeConstant(1, True)
  Dim pSourceDataset As IRaster
```

```
912:   Set pSourceDataset = ClipRasterToPolygon(pBaseRaster, pStartPolygon, True, , , pEnv)

  ' Declare the input cost raster object
  Dim pCostDataset As IGeoDataset

  ' Declare the output raster object
  Dim pOutputRaster As IGeoDataset

  ' Calls the method
921:   Set pOutputRaster = pDistanceOp.CostDistanceFull(pSourceDataset, pCorRaster, True, True, False)

  ' To access the backlink raster from the output generated from the above code:
  Dim pRasterBandCollection As IRasterBandCollection
925:   Set pRasterBandCollection = pOutputRaster
  Dim pDistBand As IRasterBand        ' DISTANCE BAND
927:   Set pDistBand = pRasterBandCollection.Item(0)
  Dim pDistRaster As IRasterBandCollection
929:   Set pDistRaster = New Raster
930:   pDistRaster.Add pDistBand, 0

  Dim pDistAsRaster As IRaster
933:   Set pDistAsRaster = pDistRaster

  Dim pBacklinkBand As IRasterBand    ' BACKLINK BAND
936:   Set pBacklinkBand = pRasterBandCollection.Item(1)
  Dim pBacklinkRaster As IRasterBandCollection
938:   Set pBacklinkRaster = New Raster
939:   pBacklinkRaster.Add pBacklinkBand, 0

'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
'  Dim pMap As IMap
'  Set pMap = pMxDoc.FocusMap

'  ThisDocument.Graphic_MakeFromGeometry pMxDoc, pCleanLine, "TestClipPolyGraphics"

'  Dim pRasLayer As IRasterLayer
'  Set pRasLayer = New RasterLayer
'  pRasLayer.CreateFromRaster pCorRaster
'  pMap.AddLayer pRasLayer
'  Dim pRasLayer2 As IRasterLayer
'  Set pRasLayer2 = New RasterLayer
'  pRasLayer2.CreateFromRaster pDistAsRaster
'  pMap.AddLayer pRasLayer2
'  Dim pRasLayer3 As IRasterLayer
'  Set pRasLayer3 = New RasterLayer
'  pRasLayer3.CreateFromRaster pBacklinkRaster
```

```
'  pMap.AddLayer pRasLayer3

'  Dim pEnumVertices As IEnumVertex
'  Set pEnumVertices = pPoints.EnumVertices
'  Dim pVertex As IPoint
'  Set pVertex = New Point
'  Dim lngPlaceholder As Long
'  Dim lngPlaceholder2 As Long

  ' ArcGIS APPEARS TO OCCASIONALLY SCREW UP THE SINGLE BACK PATH FUNCTION.  THEREFORE WE'LL
  ' CREATE THEM ALL

  Dim pPathCollection As IGeometryCollection
972:   Set pPathCollection = pDistanceOp.CostPathAsPolyline(pPoints, pDistAsRaster, pBacklinkRaster)
  ' IF THIS PATH COLLECTION IS EMPTY, THEN LIKELY THAT ONE OF THE POLYGONS DOES NOT INTERSECT ANY GRID CELLS.
  ' THEREFORE CANNOT FIND PATH.
975:   If pPathCollection.GeometryCount = 0 Then
    Dim pEmptyLine As IPolyline
977:     Set pEmptyLine = New Polyline
978:     pEmptyLine.SetEmpty
979:     Set CalcGridLine = pEmptyLine
    Exit Function
981:   End If

'  Dim pixelWidth As Double, pixelHeight As Double
'  pixelWidth = GridFunctions.ReturnPixelWidth(pDistAsRaster)
'  pixelHeight = GridFunctions.ReturnPixelHeight(pDistAsRaster)
'
'  Dim lngEnvYMax As Long, lngEnvXMin As Long
'  Dim pDistEnv As IEnvelope, pDistProps As IRasterProps
'  Set pDistProps = pDistAsRaster
'  Set pDistEnv = pDistProps.Extent
'  lngEnvYMax = pDistEnv.YMax
'  lngEnvXMin = pDistEnv.XMin
'
'  pEnumVertices.Reset
'  pEnumVertices.QueryNext pVertex, lngPlaceholder, lngPlaceholder2

  ' determine which row & col were clicked on
'  Dim lRow As Long, lCol As Long
'  Dim dblMinDist As Double
'  dblMinDist = pDistEnv.Width * pDistEnv.Height
'  Dim pMinPoint As IPoint
'  Set pClone = pVertex
'  Set pMinPoint = pClone.Clone

'  Dim pSizePoint As IPnt
```

```
'  Dim pPixBlock As IPixelBlock
'  Set pSizePoint = New DblPnt
'  pSizePoint.SetCoords 1, 1
'  Dim pTLCPoint As IPnt
'  Dim pSafeArray As Variant
'  Dim dblPixVal As Double

'  Do While Not pVertex.IsEmpty
'    lCol = Round(Abs(pVertex.x - lngEnvXMin - (0.5 * pixelWidth)) / pixelWidth)
'    lRow = Round(Abs(pVertex.Y - lngEnvYMax + (0.5 * pixelHeight)) / pixelHeight)
'    Debug.Print "row: " & lRow & ", col " & lCol

    ' create a pixel block (that has only one pixel)

'    Set pPixBlock = pDistAsRaster.CreatePixelBlock(pSizePoint)
'    Set pTLCPoint = New DblPnt
'    pTLCPoint.SetCoords lCol, lRow
'    pDistAsRaster.Read pTLCPoint, pPixBlock
'    pSafeArray = pPixBlock.SafeArray(0)
'    dblPixVal = pSafeArray(0, 0)
'    If dblPixVal < dblMinDist Then
'      dblMinDist = dblPixVal
'      Set pClone = pVertex
'      Set pMinPoint = pClone.Clone
'    End If
 '   Debug.Print "row: " & lRow & ", col " & lCol & ", val = " & dblPixVal & ".   Current min = " & dblMinDist
'    pEnumVertices.QueryNext pVertex, lngPlaceholder, lngPlaceholder2
'  Loop

'  Dim pPathPointCol As IPointCollection
'  Set pPathPointCol = New Multipoint
'  pPathPointCol.AddPoint pMinPoint

'  Dim pPathCollection As IGeometryCollection
'  Set pPathCollection = pDistanceOp.CostPathAsPolyline(pPathPointCol, pDistAsRaster, pBacklinkRaster)

  Dim pPath As IPolyline
  Dim pMinPath As IPolyline
  Dim dblShortDist As Double
1045:   Set pPath = pPathCollection.Geometry(0)
1046:   Set pMinPath = pPath
1047:   dblShortDist = pPath.length
1048:   If pPathCollection.GeometryCount > 1 Then
1049:     For anIndex = 1 To pPathCollection.GeometryCount - 1
1050:       Set pPath = pPathCollection.Geometry(anIndex)
1051:       If pPath.length < dblShortDist Then
1052:         Set pMinPath = pPath
```

```
1053:         dblShortDist = pPath.length
1054:       End If
1055:     Next anIndex
1056:   End If

  ' CLEAN POLYLINE
  Dim pBoundary As IPolyline
  Dim pTopoOp As ITopologicalOperator
1061:   Set pTopoOp = pCorPolygon
1062:   Set pBoundary = pTopoOp.Boundary

  Dim pCleanLine As IPolyline
1065:   If (ShouldClean) Then
1066:     Set pCleanLine = CleanPolyline(pMinPath, pBoundary)
1067:   Else
1068:     Set pCleanLine = pMinPath
1069:   End If


'    Dim pixelWidth As Double, pixelHeight As Double
'    pixelWidth = pREnv.Width / pRLayer.ColumnCount
'    pixelHeight = pREnv.Height / pRLayer.RowCount

    ' determine which row & col were clicked on
'    Dim lRow As Long, lCol As Long
'    lCol = Round(Abs(pPoint.x - pREnv.XMin - (0.5 * pixelWidth)) / pixelWidth)
'    lRow = Round(Abs(pPoint.Y - pREnv.YMax + (0.5 * pixelHeight)) / pixelHeight)







  'pExtractOp.Polygon(pOutRaster, pCorPolygon, True)

  'Create a raster layer from the output and add it to ArcMap

'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
'  Dim pMap As IMap
'  Set pMap = pMxDoc.FocusMap

'  ThisDocument.Graphic_MakeFromGeometry pMxDoc, pCleanLine, "TestClipPolyGraphics"

'  Dim pRasLayer As IRasterLayer
'  Set pRasLayer = New RasterLayer
```

```
'  pRasLayer.CreateFromRaster pCorRaster
'  pMap.AddLayer pRasLayer
'  Dim pRasLayer2 As IRasterLayer
'  Set pRasLayer2 = New RasterLayer
'  pRasLayer2.CreateFromRaster pDistAsRaster
'  pMap.AddLayer pRasLayer2
'  Dim pRasLayer3 As IRasterLayer
'  Set pRasLayer3 = New RasterLayer
'  pRasLayer3.CreateFromRaster pBacklinkRaster
'  pMap.AddLayer pRasLayer3

'  Set pCorRaster = Nothing
1112:   Set pDistAsRaster = Nothing
1113:   Set pBacklinkRaster = Nothing
1114:   pCleanLine.ReverseOrientation
1115:   Set pCleanLine.SpatialReference = pEnv.OutSpatialReference
1116:   Set CalcGridLine = pCleanLine


  Exit Function
ErrorHandler:
  HandleError True, "CalcGridLine " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function CellValues(pPoints As IPointCollection, pRaster As IRaster) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler

    Dim pRP As IRasterProps
1129:     Set pRP = pRaster

    Dim dblCellSize As Double
1132:     dblCellSize = ReturnCellSize(pRaster)

    'Get extent
    Dim pExtent As IEnvelope
1136:     Set pExtent = pRP.Extent
    Dim X1 As Double, Y1 As Double, X2 As Double, Y2 As Double
1138:     pExtent.QueryCoords X1, Y1, X2, Y2

    'Get a PixelBloc with cellvalues
    Dim pPB As IPixelBlock3
    Dim dWidth As Double, dHeight As Double
1143:     dWidth = pRP.Width
1144:     dHeight = pRP.Height
```

```
    'Create a DblPnt to hold the PixelBlock size
    Dim pPnt As IPnt
1148:     Set pPnt = New DblPnt
1149:     pPnt.SetCoords dWidth, dHeight

    'Create a point to set origin
    Dim pOrigin As IPnt
1153:     Set pOrigin = New DblPnt
1154:     pOrigin.SetCoords 0, 0

    'Create and read entire grid into PixelBlock
1157:     Set pPB = pRaster.CreatePixelBlock(pPnt)
1158:     pRaster.Read pOrigin, pPB

    Dim lngIndex As Long
    Dim dblCellValue As Double
    Dim pPoint As IPoint
    Dim dx As Double, dy As Double
    Dim nX As Double, nY As Double
    Dim iX As Long, iY As Long

    Dim pOutArray As esriSystem.IVariantArray
1168:     Set pOutArray = New esriSystem.VarArray

    Dim vCellValue As Variant

1172:     For lngIndex = 0 To pPoints.PointCount - 1
1173:       Set pPoint = pPoints.Point(lngIndex)
      ' RETURN NULL IF OUTSIDE EXTENT
1175:       If pPoint.X < X1 Or pPoint.X > X2 Or pPoint.Y < Y1 Or pPoint.Y > Y2 Then
1176:         pOutArray.Add Null
1177:       Else

1179:         dx = pPoint.X - X1
1180:         dy = Y2 - pPoint.Y

        'Find ncells from left-top
1183:         nX = dx / dblCellSize
1184:         nY = dy / dblCellSize

        'Determine cell index by taking the int. It will also take care of the fact that the index is 0 based
1187:         iX = Int(nX)
1188:         iY = Int(nY)

1190:         If (iX < 0) Then iX = 0
1191:         If (iY < 0) Then iY = 0
1192:         If (iX > pRP.Width - 1) Then
```

```
1193:           iX = pRP.Width - 1
1194:         End If
1195:         If (iY > pRP.Height - 1) Then
1196:           iY = pRP.Height - 1
1197:         End If

        'Read cell values
1200:         vCellValue = pPB.GetVal(0, iX, iY)
1201:         Debug.Print "From CellValues function..." & vCellValue
1202:         If IsEmpty(vCellValue) Then
1203:           pOutArray.Add Null
1204:         Else
1205:           pOutArray.Add CDbl(vCellValue)
1206:         End If
1207:       End If
1208:     Next lngIndex
1209:     Set CellValues = pOutArray


  Exit Function
ErrorHandler:
  HandleError True, "CellValues " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Public Function CellValue(pPoint As IPoint, pRaster As IRaster) As Variant
  On Error GoTo ErrorHandler


'    Dim pRawPixels As IRawPixels
'    Dim pRasterBandCol As IRasterBandCollection
'    Set pRasterBandCol = pRaster
'    Set pRawPixels = pRasterBandCol.Item(0)
'create a point which determines the size for IPixelBlock
'    Dim aoiPNT As IPnt
'    Set aoiPNT = New DblPnt
'        aoiPNT.SetCoords 1, 1
'create the IPixelBlock
'    Dim aoiPixelBlock As IPixelBlock
'    Set aoiPixelBlock = pRawPixels.CreatePixelBlock(aoiPNT)



    Dim pRP As IRasterProps
1236:     Set pRP = pRaster

    Dim dblCellSize As Double
```

```
1239:     dblCellSize = ReturnCellSize(pRaster)

    'Get extent
    Dim pExtent As IEnvelope
1243:     Set pExtent = pRP.Extent
    Dim X1 As Double, Y1 As Double, X2 As Double, Y2 As Double
1245:     pExtent.QueryCoords X1, Y1, X2, Y2

    ' RETURN NULL IF OUTSIDE EXTENT
1248:     If pPoint.X < X1 Or pPoint.X > X2 Or pPoint.Y < Y1 Or pPoint.Y > Y2 Then
1249:       CellValue = Null
      Exit Function
1251:     End If

    ' GET 1-CELL PIXELBLOCK
    'Get cell index from map coordinate
    Dim pCellPoint As IPoint  'Get dx dy from left-top
    Dim dx As Double, dy As Double
1257:     dx = pPoint.X - X1
1258:     dy = Y2 - pPoint.Y

    'Find ncells from left-top
    Dim nX As Double, nY As Double
1262:     nX = dx / dblCellSize
1263:     nY = dy / dblCellSize

    'Determine cell index by taking the int. It will also take care of the fact that the index is 0 based
    Dim iX As Long, iY As Long
1267:     iX = Int(nX)
1268:     iY = Int(nY)

    'If the index is < 0, set it to 0 - do I need this adjustment?
1271:     If (iX < 0) Then iX = 0
1272:     If (iY < 0) Then iY = 0
1273:     If (iX > pRP.Width - 1) Then
1274:       iX = pRP.Width - 1
1275:     End If
1276:     If (iY > pRP.Height - 1) Then
1277:       iY = pRP.Height - 1
1278:     End If

    'Create a point from cell index and return
1281:     Set pCellPoint = New Point
1282:     pCellPoint.PutCoords CDbl(iX), CDbl(iY)

    'Get a PixelBloc with cellvalues
    Dim pPB As IPixelBlock3
```

```
    Dim dWidth As Double, dHeight As Double
1287:     dWidth = pRP.Width
1288:     dHeight = pRP.Height

    'Create a DblPnt to hold the PixelBlock size
    Dim pPnt As IPnt
1292:     Set pPnt = New DblPnt
1293:     pPnt.SetCoords dWidth, dHeight

    'Create a point to set origin
    Dim pOrigin As IPnt
1297:     Set pOrigin = New DblPnt
1298:     pOrigin.SetCoords 0, 0

    'Create and read the PixelBlock
1301:     Set pPB = pRaster.CreatePixelBlock(pPnt)

    'Read cell values
1304:     pRaster.Read pOrigin, pPB
    Dim vCellValue As Variant
1306:     vCellValue = pPB.GetVal(0, iX, iY)
1307:     Debug.Print "From function..." & vCellValue
1308:     If IsEmpty(vCellValue) Then
1309:       CellValue = Null
1310:     Else
1311:       CellValue = CDbl(vCellValue)
1312:     End If


  Exit Function
ErrorHandler:
  HandleError True, "CellValue " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
```

Module 6: modClipFunctions

```
Attribute VB_Name = "modClipFunctions"
Option Explicit
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\modClipFunctions.bas"

Public Function ClipRasterLayer(ByVal pPolygon As IPolygon, ByRef pRasterLayer As IRasterLayer, strPath As String, _
    pApp As IApplication) As IRasterLayer
  On Error GoTo ErrorHandler

'  MsgBox "ModClipFunctions - Before: " & strPath
'  strPath = Linkages.aml_func_mod.BasicTrimAvenue(strPath, "", "/\")
```

```
'  MsgBox "ModClipFunctions - After: " & strPath
  ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
13:   Set psbar = pApp.StatusBar
14:   psbar.ProgressBar.position = 1
  Dim pPro As IStepProgressor
'  Dim pProgAnim As IAnimationProgressor
'  Set pProgAnim = psbar.ProgressAnimation
'  pProgAnim.Show
19:   Set pPro = psbar.ProgressBar
20:   psbar.PlayProgressAnimation True

  Dim strCharacters As String
  Dim strNumbers As String

25:   strCharacters = "abcdefghijklmnopqrstuvwxyz_1234567890"
26:   strNumbers = "0123456789"
  Dim strCurrentName As String
28:   strCurrentName = pRasterLayer.Name

  Dim strGridName As String

 ' CALCULATE NAME FOR NEW GRID
33:   If InStr(1, strNumbers, Left(strCurrentName, 1)) > 0 Then
34:     strGridName = "z"
35:   Else
36:     strGridName = ""
37:   End If

  Dim lngIndex As Long
  Dim strChar As String
41:   For lngIndex = 1 To Len(strCurrentName)
42:     strChar = Mid(strCurrentName, lngIndex, 1)
43:     If InStr(1, strCharacters, strChar, vbTextCompare) = 0 Then
44:       strGridName = strGridName & "_"
45:     Else
46:       strGridName = strGridName & strChar
47:     End If
48:   Next lngIndex

  Dim strTestGridName As String
  Dim strBaseGridName As String
  Dim lngNameCounter As Long
  Dim strFileString As String
  Dim booGridExists As Boolean

56:   strTestGridName = Left(strGridName, 8) & "_clip"
```

```
57:   strBaseGridName = Left(strGridName, 8)
58:   lngNameCounter = 1

60:   strFileString = strPath & strGridName
61:   booGridExists = Linkages.aml_func_mod.ExistFileDir(strPath & strTestGridName)

63:   Do While booGridExists
64:     lngNameCounter = lngNameCounter + 1
65:     strTestGridName = Left(strBaseGridName, 8 - Len(CStr(lngNameCounter))) & "_clip" & CStr(lngNameCounter)
66:     booGridExists = Linkages.aml_func_mod.ExistFileDir(strPath & strTestGridName)
67:   Loop

69:   Screen.MousePointer = vbHourglass
70:   psbar.ProgressBar.Show
71:   psbar.ProgressBar.Message = "Clipping " & UCase(pRasterLayer.Name) & "..."

  Dim pRaster As IRaster
  Dim pClipRaster As IRaster

76:   Set pRaster = pRasterLayer.Raster
  ' aaa

  ' MIGHT NEED TO MAKE SURE CLIP POLYGON AND RASTER ARE IN SAME SPATIAL REFERENCE
  Dim pClipperPolygon As IPolygon
  Dim pClone As IClone
82:   Set pClone = pPolygon
83:   Set pClipperPolygon = pClone.Clone

  ' CHECK IF OBJECTS EVEN INTERSECT
  Dim pMapSpRef As ISpatialReference
  Dim pMxDoc As IMxDocument

89:   Set pMxDoc = pApp.Document
90:   Set pMapSpRef = pMxDoc.FocusMap.SpatialReference

  Dim pPolyEnv As IEnvelope
93:   Set pPolyEnv = pPolygon.Envelope
  Dim pPolyEnvPolygon As IPolygon
  Dim pPolySpatRef As ISpatialReference
96:   Set pPolySpatRef = pPolygon.SpatialReference

98:   Set pPolyEnvPolygon = Linkages.MyGeometricOperations.EnvelopeToPolygon(pPolyEnv)

  ' FOR DEBUGGING
'  MsgBox "Polygon Envelope is Nothing:  " & CStr(pPolyEnv Is Nothing) & vbCrLf & _
        "Polygon Envelope is Empty:  " & CStr(pPolyEnv.IsEmpty) & vbCrLf & _
        "Dimensions: " & pPolyEnv.Width & " x " & pPolyEnv.Height & vbCrLf & _
```

```
        "Spatial Reference = " & pPolyEnv.SpatialReference.Name
  ' END DEBUGGING

  Dim pRastEnv As IEnvelope
  Dim pRastProps As IRasterProps
109:   Set pRastProps = pRaster
110:   Set pRastEnv = pRastProps.Extent
  Dim pRastEnvPolygon As IPolygon
  Dim pRastSpatRef As ISpatialReference
113:   Set pRastSpatRef = pRastProps.SpatialReference
114:   Set pRastEnvPolygon = Linkages.MyGeometricOperations.EnvelopeToPolygon(pRastEnv)

  ' FOR DEBUGGING
'  MsgBox "Raster Envelope is Nothing:  " & CStr(pRastEnv Is Nothing) & vbCrLf & _
        "Raster Envelope is Empty:  " & CStr(pRastEnv.IsEmpty) & vbCrLf & _
        "Dimensions: " & pRastEnv.Width & " x " & pRastEnv.Height & vbCrLf & _
        "Raster Envelope Spatial Reference = " & pRastEnv.SpatialReference.Name & _
        "Raster Spatial Reference = " & pRastProps.SpatialReference.Name
  ' END DEBUGGING

  ' IF EITHER SPATIAL REFERENCE IS UNKNOWN, THEN SET REFERENCE = MAP REFERENCE

126:   If Not Linkages.GridFunctions.CompareSpatialReferences(pPolySpatRef, pRastSpatRef) And _
    (Not pPolySpatRef Is Nothing) And (Not pRastSpatRef Is Nothing) Then
128:       pClipperPolygon.Project pRastSpatRef
129:   End If

  Dim pRelOp As IRelationalOperator
132:   Set pRelOp = pClipperPolygon.Envelope

  ' FOR DEBUGGING
'  Set pPolyEnvPolygon = Linkages.MyGeometricOperations.EnvelopeToPolygon(pClipperPolygon.Envelope)
'  MsgBox "Clipper Polygon After Possible Projection Envelope is Nothing:  " & CStr(pClipperPolygon.Envelope Is Nothing) & vbCrLf & _
'        "Polygon Envelope is Empty:  " & CStr(pClipperPolygon.Envelope.IsEmpty) & vbCrLf & _
'        "Dimensions: " & pClipperPolygon.Envelope.Width & " x " & pClipperPolygon.Envelope.Height & vbCrLf & _
'        "Spatial Reference = " & pClipperPolygon.Envelope.SpatialReference.Name
'  Dim pTopoOp As ITopologicalOperator
'  Set pTopoOp = pPolyEnvPolygon
'  Dim pProxOp As IProximityOperator
'  Set pProxOp = pPolyEnvPolygon
'  MsgBox "Prox Op Distance: " & CStr(pProxOp.ReturnDistance(pRastEnv))
'  MsgBox "Relational Op 'Disjoint':  " & CStr(pRelOp.Disjoint(pRastEnvPolygon))
  ' END DEBUGGING

  Dim booDisjoint As Boolean
149:   booDisjoint = pRelOp.Disjoint(pRastEnv)
```

```
  ' FOR DEBUGGING
'  MsgBox "Disjoint = " & CStr(booDisjoint)
'  MsgBox "Envelopes Intersect = " & CStr(Not pRelOp.Disjoint(pRastEnv)) & vbCrLf & _
      "Poly Envelope = " & CStr(pPolyEnv.Width) & " x " & CStr(pPolyEnv.Height) & vbCrLf & _
      "Raster Envelope = " & CStr(pRastEnv.Width) & " x " & CStr(pRastEnv.Height)
  ' END DEBUGGING

158:   If pRelOp.Disjoint(pRastEnvPolygon) Then
159:     Screen.MousePointer = vbDefault
160:     psbar.ProgressBar.Hide
161:     psbar.PlayProgressAnimation False
    Exit Function
163:   End If

165:   Set pClipRaster = Linkages.GridFunctions.ClipRasterToPolygon(pRaster, pClipperPolygon, True, pPolyEnv, , , True, pApp)

  Dim pRasterBandCol As IRasterBandCollection
  Dim pDS As IDataset
169:   Set pRasterBandCol = pClipRaster

171:   Linkages.GridFunctions.SaveRasterAs pRasterBandCol, strPath, strTestGridName, enum_Grid_Type

  Dim pReturnRasterLayer As IRasterLayer
174:   Set pReturnRasterLayer = New RasterLayer
175:   pReturnRasterLayer.CreateFromFilePath strPath & strTestGridName
176:   Set ClipRasterLayer = pReturnRasterLayer

178:   Screen.MousePointer = vbDefault
179:   psbar.ProgressBar.Hide
'  pProgAnim.Hide
181:   psbar.PlayProgressAnimation False

183:   Set pRaster = Nothing
184:   Set pRasterLayer = Nothing
185:   Set pClipRaster = Nothing

  Exit Function
ErrorHandler:
  HandleError True, "ClipRasterLayer " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function ClipFeatureLayer(ByVal pPolygon As IPolygon, ByRef pFeatureLayer As IFeatureLayer, strPath As String, _
    pApp As IApplication) As IFeatureLayer
  On Error GoTo ErrorHandler
```

```
  Dim strName As String
198:   strName = pFeatureLayer.Name
199:   strName = Linkages.aml_func_mod.ClipExtension(strName) & "_clip.shp"

  Dim theNewName As String
202:   theNewName = Linkages.aml_func_mod.MakeUniqueFilename(strPath & strName)

  Dim pFeatureClass As IFeatureClass
205:   Set pFeatureClass = pFeatureLayer.FeatureClass
  Dim intShapeType As esriGeometryType
207:   intShapeType = pFeatureClass.ShapeType

  Dim intGeometryDim As esriGeometryDimension
  Select Case intShapeType
    Case 1, 2
      intGeometryDim = esriGeometry0Dimension
    Case 3, 6, 13, 14, 15, 16
      intGeometryDim = esriGeometry1Dimension
    Case 4, 5, 9, 11, 18, 19, 22
      intGeometryDim = esriGeometry2Dimension
    Case Else
      intGeometryDim = esriGeometry0Dimension
219:   End Select

  Dim strGeoFieldName As String
222:   strGeoFieldName = pFeatureClass.ShapeFieldName
  Dim pFields As IFields
224:   Set pFields = pFeatureClass.Fields
  Dim pGeoField As IField
226:   Set pGeoField = pFields.Field(pFields.FindField(strGeoFieldName))
227:   If (Not pGeoField Is Nothing) And (pGeoField.Type = esriFieldTypeGeometry) Then

    Dim pLayerSpRef As ISpatialReference
    Dim pGeoDef As IGeometryDef
231:     Set pGeoDef = pGeoField.GeometryDef
232:     Set pLayerSpRef = pGeoDef.SpatialReference

    Dim pCorSpRef As ISpatialReference
235:     Set pCorSpRef = pPolygon.SpatialReference
    Dim booRefSame As Boolean
237:     booRefSame = Linkages.MyGeneralOperations.CompareSpatialReferences(pLayerSpRef, pCorSpRef)

    ' MAKE SURE SPATIAL REFERENCES ARE THE SAME
    ' DO I NEED TO CHECK FOR ONE SPATIAL REFERENCE BEING UNKNOWN?
241:     If Not booRefSame Then
242:       pPolygon.Project pLayerSpRef
243:     End If
```

```
244:   Else
245:     Set pLayerSpRef = pPolygon.SpatialReference
246:   End If

' MsgBox "Polygon Spatial Reference = " & pPolygon.SpatialReference.Name

  Dim pFeatureCursor As IFeatureCursor

  Dim pSpatialFilter As ISpatialFilter
253:   Set pSpatialFilter = New SpatialFilter

255:   Set pSpatialFilter.Geometry = pPolygon
256:   pSpatialFilter.GeometryField = strGeoFieldName
257:   pSpatialFilter.SpatialRel = esriSpatialRelIntersects

  ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
261:   Set psbar = pApp.StatusBar
262:   psbar.ProgressBar.position = 1
'  Dim pProgAnim As IAnimationProgressor
'  Set pProgAnim = psbar.ProgressAnimation
'  pProgAnim.Show
  Dim pPro As IStepProgressor
267:   Set pPro = psbar.ProgressBar
268:   psbar.PlayProgressAnimation True

270:   Screen.MousePointer = vbHourglass
271:   psbar.ProgressBar.Show
272:   psbar.ProgressBar.Message = "Examining " & UCase(pFeatureLayer.Name) & "..."

  ' MAKE SELECTION ON FEATURE CLASS
  Dim pFeature As IFeature
  Dim lngFeatureCount As Long
277:   lngFeatureCount = pFeatureClass.FeatureCount(pSpatialFilter)
278:   Set pFeatureCursor = pFeatureClass.Search(pSpatialFilter, False)
279:   Set pFeature = pFeatureCursor.NextFeature
'  MsgBox lngFeatureCount & " features selected..."
281:   If pFeature Is Nothing Then
282:     Set ClipFeatureLayer = Nothing
283:     Screen.MousePointer = vbDefault
284:     psbar.ProgressBar.Hide
285:     psbar.PlayProgressAnimation False
    Exit Function
287:   End If

  ' ADD DEFAULT FIELDS TO TABLE
  Dim pField As IField
```

```
291:   Set pField = New Field
  Dim pFieldEdit As IFieldEdit
293:   Set pFieldEdit = pField
  Dim pField2 As IField
295:   Set pField2 = New Field
  Dim pFieldEdit2 As IFieldEdit
297:   Set pFieldEdit2 = pField2
  Dim strShapeName As String

  ' MAKE NEW SHAPEFILE
  Dim pClipFeatureClass As IFeatureClass
  Dim strOutputType As String
  Select Case intShapeType
    Case 1                          ' POINT
305:       Set pClipFeatureClass = aml_func_mod.CreateShapefile(Linkages.aml_func_mod.ReturnDir(theNewName), _
          Linkages.aml_func_mod.ReturnFilename(theNewName), _
          pLayerSpRef, "Point")
308:       strOutputType = "Point"
309:       With pFieldEdit
310:             .Name = "X_Coord"
311:             .AliasName = "X_Coord"
312:             .Type = esriFieldTypeDouble
313:       End With
314:       With pFieldEdit2
315:             .Name = "Y_Coord"
316:             .AliasName = "Y_Coord"
317:             .Type = esriFieldTypeDouble
318:       End With
319:       strShapeName = "Points"
    Case 2                          ' MULTIPOINT
321:       Set pClipFeatureClass = aml_func_mod.CreateShapefile(Linkages.aml_func_mod.ReturnDir(theNewName), _
          Linkages.aml_func_mod.ReturnFilename(theNewName), _
          pLayerSpRef, "Multipoint")
324:       strOutputType = "Multipoint"
325:       With pFieldEdit
326:             .Name = "clip_count"
327:             .AliasName = "clip_count"
328:             .Type = esriFieldTypeDouble
329:       End With
330:       strShapeName = "Multipoints"
    Case 3, 6, 13, 14, 15, 16       ' POLYLINE
332:       Set pClipFeatureClass = aml_func_mod.CreateShapefile(Linkages.aml_func_mod.ReturnDir(theNewName), _
          Linkages.aml_func_mod.ReturnFilename(theNewName), _
          pLayerSpRef, "Polyline")
335:       strOutputType = "Polyline"
336:       With pFieldEdit
337:             .Name = "clip_len"
```

```
338:             .AliasName = "clip_len"
339:             .Type = esriFieldTypeDouble
340:       End With
341:       strShapeName = "Polylines"
    Case 4, 5, 9, 11, 18, 19, 22   ' POLYGON
343:       Set pClipFeatureClass = aml_func_mod.CreateShapefile(Linkages.aml_func_mod.ReturnDir(theNewName), _
         Linkages.aml_func_mod.ReturnFilename(theNewName), _
         pLayerSpRef, "Polygon")
346:       strOutputType = "Polygon"
347:       With pFieldEdit
348:             .Name = "clip_area"
349:             .AliasName = "clip_area"
350:             .Type = esriFieldTypeDouble
351:       End With
352:       strShapeName = "Polygons"
    Case Else                      ' POINTS
354:       Set pClipFeatureClass = aml_func_mod.CreateShapefile(Linkages.aml_func_mod.ReturnDir(theNewName), _
         Linkages.aml_func_mod.ReturnFilename(theNewName), _
         pLayerSpRef, "Point")
357:       strOutputType = "Point"
358:       With pFieldEdit
359:             .Name = "X_Coord"
360:             .AliasName = "X_Coord"
361:             .Type = esriFieldTypeDouble
362:       End With
363:       With pFieldEdit2
364:             .Name = "Y_Coord"
365:             .AliasName = "Y_Coord"
366:             .Type = esriFieldTypeDouble
367:       End With
368:       strShapeName = "Vertices"
369:   End Select

371:   Screen.MousePointer = vbHourglass
372:   psbar.ShowProgressBar "Clipping " & strShapeName & " from " & UCase(pFeatureLayer.Name) & "...", 1, _
        lngFeatureCount, 1, False

  ' ADD GEOMETRIC MEASURE FIELD(S) TO FEATURE CLASS
  Dim pTable As ITable
377:   Set pTable = pClipFeatureClass
378:   pTable.AddField pField
379:   If strOutputType = "Point" Then pTable.AddField pField2

  ' ADD ORIGINAL FIELDS TO TABLE
  Dim intIndexArray() As Integer
  ReDim intIndexArray(pFields.FieldCount, 2)
  Dim intFieldIndex As Integer
```

```
  Dim intNewFieldIndex As Integer
386:   intNewFieldIndex = -1
  Dim pOriginalField As IField
  Dim pClone As IClone
  Dim pNewField As IField
  Dim pNewFieldEdit As IFieldEdit
  Dim lngOIDFieldIndex As String
392:   lngOIDFieldIndex = -9999
  Dim intNameCounter As Integer
  Dim strBaseName As String
  Dim strNewName As String

397:   For intFieldIndex = 0 To (pFields.FieldCount - 1)
398:     Set pOriginalField = pFields.Field(intFieldIndex)
399:     If (Not pOriginalField.Name = pFeatureClass.ShapeFieldName) Then
400:         Set pClone = pOriginalField
401:         Set pNewField = pClone.Clone
402:         Set pNewFieldEdit = pNewField
       'MsgBox "Field Name = " & pOriginalField.Name & vbCrLf & "Field Type = " & pOriginalField.Type
404:         If pOriginalField.Name = pFeatureClass.OIDFieldName Or pOriginalField.Type = esriFieldTypeOID Then
'          MsgBox "Converting to Long OID field..."
406:           Set pNewField = New Field
407:           Set pNewFieldEdit = pNewField
408:           pNewFieldEdit.Type = esriFieldTypeInteger
409:           pNewFieldEdit.Name = ("Orig_OID")
410:           lngOIDFieldIndex = intNewFieldIndex + 1
411:         ElseIf pNewField.Name = "AREA" Or pNewField.Name = "area" Or pNewField.Name = "Area" Then
412:           Set pNewField = New Field
413:           Set pNewFieldEdit = pNewField
414:           pNewFieldEdit.Type = esriFieldTypeDouble
415:           pNewFieldEdit.Precision = pOriginalField.Precision
416:           pNewFieldEdit.length = pOriginalField.length
417:           pNewFieldEdit.Name = "Orig_Area"
418:         End If
       ' MsgBox "Adding Field Name = " & pNewField.Name & vbCrLf & "Field Type = " & pNewField.Type

        ' MAKE SURE NEW FIELD NAME ISN'T ALREADY PRESENT
422:         intNameCounter = 1
423:         strNewName = pNewField.Name
424:         strBaseName = strNewName
425:         Do While pTable.FindField(strNewName) > -1
426:           intNameCounter = intNameCounter + 1
427:           strNewName = Left(strBaseName, 10 - Len(CStr(intNameCounter))) & CStr(intNameCounter)
428:         Loop
429:         pNewFieldEdit.Name = strNewName
430:         pTable.AddField pNewField
```

```
432:         intNewFieldIndex = intNewFieldIndex + 1
433:         intIndexArray(intNewFieldIndex, 0) = pFeatureClass.FindField(pOriginalField.Name)  ' ORIGINAL FEATURE CLASS
434:         intIndexArray(intNewFieldIndex, 1) = pTable.FindField(pNewField.Name)              ' NEW CLIPPED FEATURE CLASS
435:     End If
436:   Next intFieldIndex

  Dim pNewFields As IFields
439:   Set pNewFields = pClipFeatureClass.Fields
  Dim intMeasureField1Index As Long
  Dim intMeasureField2Index As Long
  Dim intIDFieldIndex As Long
  Select Case strOutputType
    Case "Point"
445:       intMeasureField1Index = pNewFields.FindField("X_Coord")
446:       intMeasureField2Index = pNewFields.FindField("Y_Coord")
    Case "Multipoint"
448:       intMeasureField1Index = pNewFields.FindField("clip_count")
    Case "Polyline"
450:       intMeasureField1Index = pNewFields.FindField("clip_len")
    Case "Polygon"
452:       intMeasureField1Index = pNewFields.FindField("clip_area")
453:   End Select
454:   intIDFieldIndex = pNewFields.FindField("unique_id")

  ' MAKE FEATURE CURSOR AND FEATURE BUFFER FOR NEW SHAPEFILE
  Dim pNewFeatBuf As IFeatureBuffer
  Dim pNewFeatCur As IFeatureCursor
459:   Set pNewFeatCur = pClipFeatureClass.Insert(True)
460:   Set pNewFeatBuf = pClipFeatureClass.CreateFeatureBuffer

  ' FOR POINTS
  Dim pPoint As IPoint

  ' FOR MULTIPOINTS
  Dim pOrigMultipoint As IMultipoint
  Dim pClipMultipoint As IMultipoint
  Dim pPointCollection As IPointCollection

  ' FOR POLYLINES
  Dim pOrigPolyline As IPolyline
  Dim pClipPolyline As IPolyline
  Dim pGeometryCollection As IGeometryCollection
  Dim pPath As IPath
  Dim pSubPolylineCollection As IGeometryCollection
  Dim pSubPolyline As IPolyline

  ' FOR POLYGONS
```

```
  Dim pOrigPolygon As IPolygon
  Dim pClipPolygon As IPolygon
  Dim pArea As IArea
  Dim pSubPoly As IPolygon
  Dim pPoly2 As IPolygon2
  Dim pPolyArray() As IPolygon

  Dim lngIndex As Long
  Dim lngCount As Long
488:   lngCount = 0

  ' pTopoOp IS THE CLIPPING BOUNDARY POLYGON
  Dim pTopoOp As ITopologicalOperator2
492:   Set pTopoOp = pPolygon
493:   pTopoOp.IsKnownSimple = False
494:   pTopoOp.Simplify

  Dim pTopoOp2 As ITopologicalOperator2

498:   Do Until pFeature Is Nothing

    Select Case strOutputType
      Case "Point"       ' ------------------------------------------------------------ POINTS
502:         Set pPoint = pFeature.ShapeCopy
503:         Set pNewFeatBuf.Shape = pPoint
504:         pNewFeatBuf.Value(intMeasureField1Index) = pPoint.X
505:         pNewFeatBuf.Value(intMeasureField2Index) = pPoint.Y
506:         lngCount = lngCount + 1
507:         pNewFeatBuf.Value(intIDFieldIndex) = lngCount
       ' ADD ORIGINAL DATA
509:         For intFieldIndex = 0 To intNewFieldIndex
510:           If intFieldIndex = lngOIDFieldIndex Then
511:             pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = CLng(pFeature.OID)
512:           Else
513:             pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = pFeature.Value(intIndexArray(intFieldIndex, 0))
514:           End If
515:         Next intFieldIndex
516:         pNewFeatCur.InsertFeature pNewFeatBuf

      Case "Multipoint"  ' ------------------------------------------------------- MULTIPOINTS
519:         Set pOrigMultipoint = pFeature.ShapeCopy
520:         Set pClipMultipoint = pTopoOp.Intersect(pOrigMultipoint, esriGeometry0Dimension)
521:         Set pNewFeatBuf.Shape = pClipMultipoint
522:         Set pPointCollection = pClipMultipoint
523:         pNewFeatBuf.Value(intMeasureField1Index) = pPointCollection.PointCount
524:         lngCount = lngCount + 1
525:         pNewFeatBuf.Value(intIDFieldIndex) = lngCount
```

```
        ' ADD ORIGINAL DATA
527:          For intFieldIndex = 0 To intNewFieldIndex
528:            If intFieldIndex = lngOIDFieldIndex Then
529:              pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = CLng(pFeature.OID)
530:            Else
531:              pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = pFeature.Value(intIndexArray(intFieldIndex, 0))
532:            End If
533:          Next intFieldIndex
534:          pNewFeatCur.InsertFeature pNewFeatBuf

       Case "Polyline"    ' ------------------------------------------------------------ POLYLINES
        ' INTERSECT POLYLINE
538:          Set pOrigPolyline = pFeature.ShapeCopy
539:          Set pClipPolyline = pTopoOp.Intersect(pOrigPolyline, esriGeometry1Dimension)

        ' EXPLODE INTERSECTED POLYLINES
542:          Set pGeometryCollection = pClipPolyline
543:          For lngIndex = 0 To pGeometryCollection.GeometryCount - 1
544:            Set pPath = pGeometryCollection.Geometry(lngIndex)
545:            Set pSubPolyline = New Polyline
546:            Set pSubPolylineCollection = pSubPolyline
547:            pSubPolylineCollection.AddGeometry pPath
548:            If Not pSubPolyline Is Nothing Then
549:              Set pNewFeatBuf.Shape = pSubPolyline
550:              pNewFeatBuf.Value(intMeasureField1Index) = pSubPolyline.length
551:              lngCount = lngCount + 1
552:              pNewFeatBuf.Value(intIDFieldIndex) = lngCount
            ' ADD ORIGINAL DATA
554:              For intFieldIndex = 0 To intNewFieldIndex
555:                If intFieldIndex = lngOIDFieldIndex Then
    '             MsgBox "OID = " & CLng(pFeature.OID) & vbCrLf & _
                   "Writing to field '" & pNewFeatBuf.Fields.Field(intFieldIndex).Name & _
                   "', Type =" & pNewFeatBuf.Fields.Field(intFieldIndex).Type

560:                 pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = CLng(pFeature.OID)
561:                Else
562:                  pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = pFeature.Value(intIndexArray(intFieldIndex, 0))
563:                End If
564:              Next intFieldIndex
565:              pNewFeatCur.InsertFeature pNewFeatBuf
566:            End If
567:          Next lngIndex

       Case "Polygon"     ' ------------------------------------------------------------ POLYGONS
        ' INTERSECT POLYGON AND SIMPLIFY IT
571:          Set pOrigPolygon = pFeature.ShapeCopy
572:          Set pClipPolygon = pTopoOp.Intersect(pOrigPolygon, esriGeometry2Dimension)
```

```
573:         Set pTopoOp2 = pClipPolygon
574:         pTopoOp2.IsKnownSimple = False
575:         pTopoOp2.Simplify

        ' EXPLODE INTERSECTED POLYGONS
578:         Set pPoly2 = pClipPolygon
        ReDim pPolyArray(pPoly2.ExteriorRingCount - 1)
580:         pPoly2.GetConnectedComponents (pPoly2.ExteriorRingCount), pPolyArray(0)

582:         For lngIndex = 0 To UBound(pPolyArray)
          'Debug.Print "Polygon number : " & i & " Length : " & ppolout(i).length
584:           Set pSubPoly = pPolyArray(lngIndex)
585:           If Not pSubPoly Is Nothing Then
586:             Set pArea = pSubPoly
587:             Set pNewFeatBuf.Shape = pSubPoly
588:             pNewFeatBuf.Value(intMeasureField1Index) = pArea.Area
589:             lngCount = lngCount + 1
590:             pNewFeatBuf.Value(intIDFieldIndex) = lngCount
          ' ADD ORIGINAL DATA
592:             For intFieldIndex = 0 To intNewFieldIndex
593:               If intFieldIndex = lngOIDFieldIndex Then
594:                 pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = CLng(pFeature.OID)
595:               Else
596:                 pNewFeatBuf.Value(intIndexArray(intFieldIndex, 1)) = pFeature.Value(intIndexArray(intFieldIndex, 0))
597:               End If
598:             Next intFieldIndex

600:             pNewFeatCur.InsertFeature pNewFeatBuf
601:           End If
602:         Next lngIndex

604:     End Select

606:     Set pFeature = pFeatureCursor.NextFeature
607:     psbar.StepProgressBar
608:   Loop

610:   pNewFeatCur.Flush

612:   psbar.PlayProgressAnimation False
'  pProgAnim.Hide
614:   psbar.HideProgressBar
615:   Screen.MousePointer = vbDefault

  Dim pNewFeatureLayer As IFeatureLayer
618:   Set pNewFeatureLayer = New FeatureLayer
619:   Set pNewFeatureLayer.FeatureClass = pClipFeatureClass
```

```
621:   Set ClipFeatureLayer = pNewFeatureLayer

  Exit Function
ErrorHandler:
  HandleError True, "ClipFeatureLayer " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
```

Module 7: modGenFunctions

```
Attribute VB_Name = "modGenFunctions"
' Declaration of a Windows routine.
' This statement should be placed in the module.

Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal _
  hWndInsertAfter As Long, ByVal X As Long, ByVal Y As _
  Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags _
  As Long) As Long

Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
  (ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

Public Const GWL_HWNDPARENT = (-8)

Option Explicit
```

Module 8: modHelpStrings

```
Attribute VB_Name = "modHelpStrings"
Option Explicit

Public Property Get Step1Polygons() As String
4:   Step1Polygons = _
    "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}{\f1\fmodern\fprq1\fcharset0 Courier New;}}" &
vbCrLf & _
    "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\b\f0\fs16 Identifying Corridor and Wildland Blocks\par" & vbCrLf & _
    "\pard\par" & vbCrLf & _
    "\b0 This function will evaluate a corridor connecting two habitat blocks.  Therefore you must identify the polygon objects that
represent the corridor and each habitat block.  You may select these polygons by either selecting the appropriate polygon layer in the
drop-down box, or by clicking on the actual polygon itself in your map.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "\b Note:\b0   Only a single polygon feature may be used for the corridor and habitat block polygons.  This single feature may be
a multipart polygon, but it must be represented by either a single record in the attribute table or a single object selected on the
screen.  Multipart polygons are considered single polygon objects, but multiple records in a polygon layer are not.  This tool will
```

```
check for this condition before allowing you to proceed.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "If you wish to select the polygon by clicking on the map, then choose the option \f1 '<-- Select by clicking on map -->' \f0 in
the drop-down box.  Choosing this option will enable the 'Select' button, which then opens a new dialog to allow you to select from
the map.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "If you wish to select the polygon by selecting the layer in the drop-down box, then make sure that the layer has only a single
polygon in it.  If there are multiple polygons in the layer, then you must select the appropriate polygon using the\f1  '<-- Select by
clicking on map -->'\f0  option.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "\b Note:\b0   The selected corridor must intersect with both habitat blocks in order to be a valid corridor.  Also, the habitat
blocks must not intersect with each other (if they intersect, then no corridor is needed!).  This tool will check for both of these
conditions before allowing you to proceed.\par" & vbCrLf & _
    "}"
End Property

Public Property Get Step2PatchesGeneral() As String

22:   Step2PatchesGeneral = _
    "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
    "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\b\f0\fs16 What are Patch Polygons?\b0\par" & vbCrLf & _
    "\pard\par" & vbCrLf & _
    "A habitat patch is a cluster of pixels that are good enough, big enough, and close enough together to support a particular
species at some important level.  The CorridorDesigner toolbox includes functions to create patch polygons from your Habitat
Suitability Model.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "Patch polygons are treated as stepping-stones through the corridor, and considered islands of high-quality habitat dispersed
through potentially marginal corridor habitat.  Users of this tool will generally have some idea of a maximum threshold distance that
a species is capable of making through marginal- or poor-quality habitat, and this tool will help the user decide whether the patch
distribution in a corridor will allow that species to move through it.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "If you include patch polygons in your analysis, then this tool will identify the route through the corridor that uses the minimum
necessary patch-to-patch distances.  If you do not include patch polygons, then this tool will identify the minimum distance required
to move from one habitat block to the other.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "Upon completion, this tool will produce a report listing the patch-to-patch segment lengths required for a species to move from
one habitat block to another, sorted in decreasing order by segment length.\par" & vbCrLf & _
    "}"

End Property
Public Property Get Step2PatchesCriteria() As String

38:   Step2PatchesCriteria = _
    "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
    "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\b\f0\fs16 Selecting Patch Polygons\b0\par" & vbCrLf & _
    "\pard\par" & vbCrLf & _
    "Patch polygons may be available in a variety of sizes, and smaller patches may not be large enough to support the species at the
```

```
level you feel is necessary for that species to be able to successfully traverse the corridor.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "The CorridorDesigner tool ""Create Patch Map"" allows you to generate patch polygons based on different size thresholds.  For
example, a small patch may be useful for providing temporary refuge while a larger patch may support more long-term breeding events.
Even larger patches may support breeding over several generations.  The ""Create Patch Map"" tool creates a polygon patch layer with
attribute values that indicate the value of the patch.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "If you feel your species will require patches that meet a minimum value  threshold, then you specify this threshold in this
dialog.  Simply select the patch attribute field containing the threshold values, then enter the selection criteria.  Only those
patches that meet your selection criteria will be considered in the analysis. \par" & vbCrLf & _
    "}"

End Property

Public Property Get BottleneckHelp() As String

53:   BottleneckHelp = _
    "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}{\f1\fmodern\fprq1\fcharset0 Courier New;}}" &
vbCrLf & _
    "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\qc\b\f0\fs16 Bottleneck Analysis\b0\par" & vbCrLf & _
    "\pard\par" & vbCrLf & _
    "Some species may not be able to move through a corridor if that corridor becomes too narrow.  Bottlenecks in a corridor, even if
they do not completely prevent movement, may still inhibit the species enough that the gene flow through the corridor is insufficient
to achieve the linkage between habitat blocks that you are after.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "Presumably you have some general sense of a threshold level of corridor constriction that your species can handle.  This may be
based on observed movement or dispersal behavior of that species, or from information on how that species behaves when in close
proximity to developed areas.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "This tool will describe your corridor in terms of how wide it is over its full length.  Based on this information, you will be
able to identify the location and relative length of narrow bottlenecks inside the corridor.  In the case of multi-strand corridors,
this function will analyze and report on the strand with the widest bottleneck, and not on the strand with the widest overall corridor
width.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "\b Note:\b0   Only a single polygon feature may be used for the corridor and habitat block polygons.  This single feature may be
a multipart polygon, but it must be represented by either a single record in the attribute table or a single object selected on the
screen.  Multipart polygons are considered single polygon objects, but multiple records in a polygon layer are not.  This tool will
check for this condition before allowing you to proceed.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "If you wish to select the polygon by clicking on the map, then choose the option \f1 '<-- Select by clicking on map -->' \f0 in
the drop-down box.  Choosing this option will enable the 'Select' button, which then opens a new dialog to allow you to select from
the map.\par" & vbCrLf & _
    "\par" & vbCrLf & _
    "If you wish to select the polygon by selecting the layer in the drop-down box, then make sure that the layer has only a single
polygon in it.  If there are multiple polygons in the layer, then you must select the appropriate polygon using the\f1  '<-- Select by
clicking on map -->'\f0  option.\par" & vbCrLf & _
    "\par" & vbCrLf & _
```

```
    "\b Note:\b0   The selected corridor must intersect with both habitat blocks in order to be a valid corridor.  Also, the habitat
blocks must not intersect with each other (if they intersect, then no corridor is needed!).  This tool will check for both of these
conditions before allowing you to proceed.\par" & vbCrLf & _
    "}"


End Property
```

Module 9: MyGeneralOperations

```
Attribute VB_Name = "MyGeneralOperations"
Option Explicit
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\MyGeneralOperations.bas"


' MyGeneralOperations
' Jeff Jenness
' Jenness Enterprises
' http://www.jennessent.com

'  CheckNumericReal - GIVEN KEYASCII AND TEXTBOX, RESTRICTS INPUT TO ANY REAL NUMBER
'  CheckNumericRealPositive - GIVEN KEYASCII AND TEXTBOX, RESTRICTS INPUT TO ANY POSITIVE REAL NUMBER
'  CheckNumericInteger - GIVEN KEYASCII AND TEXTBOX, RESTRICTS INPUT TO ANY INTEGER
'  CheckNumericIntegerPositive - GIVEN KEYASCII AND TEXTBOX, RESTRICTS INPUT TO ANY POSITIVE INTEGER
'  ReturnCurrentMapUnits - GIVE IT AN IMAP, IT RETURNS THE NAME OF THE MAP UNITS
'  CompareSpatialReferences - RETURNS A BOOLEAN STATING WHETHER PROJECTION / DATUM ARE SAME.
'  ReturnTimeElapsed - GIVEN A START AND END TIME, RETURNS A STRING DESCRIBING THE TIME ELAPSED.
'           Analysis Began: Wednesday, July 26, 2006;  4:39:01 PM
'           Analysis Complete: Wednesday, July 27, 2006;  5:47:22 PM
'           Time Elapsed: 1 day, 1 hour, 8 minutes, 21 seconds...
'  ReturnTimeElapsedRTF - GIVEN A START AND END TIME, RETURNS AN RTF-FORMATTED STRING DESCRIBING THE TIME ELAPSED.
'           Analysis Began: Wednesday, July 26, 2006;  4:39:01 PM
'           Analysis Complete: Wednesday, July 27, 2006;  5:47:22 PM
'           Time Elapsed: 1 day, 1 hour, 8 minutes, 21 seconds...
'  DoSpatialQuery - SELECTS ALL FEATURES FROM FIRST FEATURE LAYER THAT INTERSECT SELECTED FEATURES OF SECONDS FEATURE LAYER
'  DeleteGraphicsByName - GIVEN A NAME AND MAP DOCUMENT, DELETES ALL GRAPHICS WITH A PARTICULAR NAME
'  MakeColorRGB - GIVEN RED, GREEN AND BLUE, RETURN ICOLOR
'  MakeColorHSV - GIVEN HUE, SATURATION AND VALUE, RETURN ICOLOR
'  OpenFile - GIVEN A DOCUMENT FILENAME AND PATH, OPENS THAT FILE USING THE REGISTERED WINDOWS PROGRAM
'  ReturnGraphicsByName - GIVEN A MAP DOCUMENT AND NAME, RETURNS A COLLECTION CONTAINING THE GEOMETRIES OF THOSE GRAPHICS
'  ReturnGraphicsByType - GIVEN A MAP DOCUMENT AND SHAPE TYPE, RETURNS A COLLECTION CONTAINING THE GEOMETRIES OF THOSE GRAPHICS
'  GraphicsSetNameSelected - GIVEN A MAP DOCUMENT AND NAME, ASSIGNS THAT NAME TO ALL SELECTED GRAPHICS
'  Graphic_MakeFromGeometry - GIVEN A MAP DOCUMENT, GEOMETRY AND OPTIONAL NAME AND SYMBOLOGY, ADDS GRAPHIC TO MAP.
'  Graphic_ReturnElementFromGeometry - GIVEN MAP DOC, GEOMETRY, OPTIONAL NAME AND OPTIONAL ADD-TO-VIEW, RETURNS THE GRAPHIC
'           ELEMENT
'  CalcStatistics - GIVEN AN ARRAY OF DOUBLES AND AN ARRAY OF BOOLEAN STAT OPTIONS, RETURNS AN ARRAY OF STATISTICS
'  BasicStatsFromArray - GIVEN anArray, Field Name, Table Name, and Application, _
```

```
            Returns Sum, Mean, Minimum, Maximum, Range, Count, StDev, Variance, Median, Standard Error of Mean and Mode String
'  BasicStatsFromArray_Weighted - GIVEN 2-Dimensional anArray with Values and Weights, Field Name, Table Name, and Application, _
            Returns Weighted Mean, Weighted StDev and Weighted Variance
'  BasicStatsFromVAT - GIVEN 2-Dimensional Arrays (Value and Size, both sorted by Value), Field Name, Table Name, and Application, _
            Returns Sum, Mean, Minimum, Maximum, Range, Count, StDev, Variance, Median, Standard Error of Mean and Mode String
'  ReturnLayersByType - GIVEN FOCUSMAP AND TYPE, RETURNS IVariantArray OF LAYERS
'  ConvertLongBinary - GIVEN A LONG AND OPTIONAL NUMBER OF CHARACTERS, RETURNS BINARY REPRESENTATION.
'  ReturnDistanceUnitsName - GIVEN AN esriUnits, RETURNS THE NAME
'  CheckCollectionForKey - GIVEN pCollection and STRING, RETURNS BOOLEAN INDICATING WHETHER COLLECTION HAS THAT KEY OR NOT
'  EnableSelectTool - GIVEN Application, CLICKS THE "SELECT ELEMENTS" TOOL

Private Declare Function ShellExecute Lib "shell32.dll" Alias _
           "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As String, _
           ByVal lpFile As String, ByVal lpParameters As String, _
           ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Public Enum JenLayerTypes
  ENUM_jenFeatureLayers = 1
  ENUM_jenRasterLayers = 2
  ENUM_jenStandaloneTables = 4
  ENUM_jenPointLayers = 8
  ENUM_jenPolylineLayers = 16
  ENUM_jenPolygonLayers = 32
  ENUM_jenMultipointLayers = 64
  ENUM_jenTinLayers = 128
End Enum

Public Function ReturnDistanceUnitsName(lngEsriUnits As esriUnits) As String
  On Error GoTo ErrorHandler


  Select Case lngEsriUnits
    Case 0
70:       ReturnDistanceUnitsName = "Unknown"
    Case 1
72:       ReturnDistanceUnitsName = "Inches"
    Case 2
74:       ReturnDistanceUnitsName = "Points"
    Case 3
76:       ReturnDistanceUnitsName = "Feet"
    Case 4
78:       ReturnDistanceUnitsName = "Yards"
    Case 5
80:       ReturnDistanceUnitsName = "Miles"
    Case 6
82:       ReturnDistanceUnitsName = "Nautical miles"
    Case 7
```

```
84:       ReturnDistanceUnitsName = "Millimeters"
    Case 8
86:       ReturnDistanceUnitsName = "Centimeters"
    Case 9
88:       ReturnDistanceUnitsName = "Meters"
    Case 10
90:       ReturnDistanceUnitsName = "Kilometers"
    Case 11
92:       ReturnDistanceUnitsName = "Decimal degrees"
    Case 12
94:       ReturnDistanceUnitsName = "Decimeters"
95:   End Select


  Exit Function
ErrorHandler:
  HandleError True, "ReturnDistanceUnitsName " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function ConvertLongBinary(lngNumber As Long, Optional MinimumNumDigits As Long = -1) As String
  On Error GoTo ErrorHandler


  Dim strBinary As String
  Dim lngIntermediate As Long
  Dim lngRemainder As Long
110:   lngIntermediate = lngNumber
111:   Do Until lngIntermediate = 1
112:     lngRemainder = lngIntermediate Mod 2
113:     lngIntermediate = Int(lngIntermediate / 2)
114:     strBinary = CStr(lngRemainder) & strBinary
115:   Loop
116:   strBinary = "1" & strBinary

118:   If Len(strBinary) < MinimumNumDigits Then
119:     Do While Len(strBinary) < MinimumNumDigits
120:       strBinary = "0" & strBinary
121:     Loop
122:   End If

124:   ConvertLongBinary = strBinary

  Exit Function
ErrorHandler:
  HandleError True, "ConvertLongBinary " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
```

```
End Function


Public Function ReturnLayersByType(pFocusMap As IMap, enumLayerTypes As JenLayerTypes) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler


  Dim booFeatureLayers As Boolean
  Dim booRasterLayers As Boolean
  Dim booStandaloneTables As Boolean
  Dim booPointLayers As Boolean
  Dim booPolylineLayers As Boolean
  Dim booPolygonLayers As Boolean
  Dim booMultipointLayers As Boolean
  Dim booTinLayers As Boolean
  Dim booTerrainLayers As Boolean

  Dim strBinary As String
147:   strBinary = ConvertLongBinary(enumLayerTypes, 8)

149:   booFeatureLayers = Mid(strBinary, 8, 1) = "1"
150:   booRasterLayers = Mid(strBinary, 7, 1) = "1"
151:   booStandaloneTables = Mid(strBinary, 6, 1) = "1"
152:   booPointLayers = Mid(strBinary, 5, 1) = "1"
153:   booPolylineLayers = Mid(strBinary, 4, 1) = "1"
154:   booPolygonLayers = Mid(strBinary, 3, 1) = "1"
155:   booMultipointLayers = Mid(strBinary, 2, 1) = "1"
156:   booTinLayers = Mid(strBinary, 1, 1) = "1"

  Dim pEnumLayer As IEnumLayer
  Dim pFeatureLayer As IFeatureLayer
  Dim pLayer As IUnknown
  Dim pFeatureClass As IFeatureClass
  Dim pGeometryType As esriGeometryType
  Dim pFeatureLayerForValid As IFeatureLayer
  Dim booOpenDialog As Boolean
165:   booOpenDialog = False
  Dim pRasterLayer As IRasterLayer

168:   Set ReturnLayersByType = New esriSystem.VarArray

170:   If (pFocusMap.LayerCount > 0) Then
171:     Set pEnumLayer = pFocusMap.Layers(, True)
172:     pEnumLayer.Reset

174:     Set pLayer = pEnumLayer.Next
175:     Do Until pLayer Is Nothing
```

```
176:       If TypeOf pLayer Is IFeatureLayer Then
177:         Set pFeatureLayerForValid = pLayer
        ' CHECK IF FEATURE LAYER IS VALID
179:         If pFeatureLayerForValid.Valid Then
          ' CHECK IF POLYGON LAYER
181:           Set pFeatureClass = pFeatureLayerForValid.FeatureClass          ' CHECK IF FEATURE LAYER
182:           pGeometryType = pFeatureClass.ShapeType
183:           If booFeatureLayers Then
184:             ReturnLayersByType.Add pLayer
185:           Else
186:             If (pGeometryType = esriGeometryPolygon) Then                 ' CHECK IF POLYGON LAYER
187:               If booPolygonLayers Then ReturnLayersByType.Add pLayer
188:             ElseIf pGeometryType = esriGeometryPolyline Then              ' CHECK IF POLYLINE LAYER
189:               If booPolylineLayers Then ReturnLayersByType.Add pLayer
190:             ElseIf pGeometryType = esriGeometryPoint Then                 ' CHECK IF POINT LAYER
191:               If booPointLayers Then ReturnLayersByType.Add pLayer
192:             ElseIf pGeometryType = esriGeometryMultipoint Then            ' CHECK IF MULTIPOINT LAYER
193:               If booMultipointLayers Then ReturnLayersByType.Add pLayer
194:             End If
195:           End If
196:         End If
197:       ElseIf TypeOf pLayer Is IRasterLayer Then                           ' CHECK IF RASTER LAYER
198:         Set pRasterLayer = pLayer
199:         If pRasterLayer.Valid Then
200:           ReturnLayersByType.Add pLayer
201:         End If
202:       ElseIf TypeOf pLayer Is ITinLayer Then                              ' CHECK IF TIN LAYER
       Dim pTinLayer As ITinLayer
204:         Set pTinLayer = pLayer
205:         If pTinLayer.Valid Then
206:           ReturnLayersByType.Add pLayer
207:         End If
208:       End If
209:       Set pLayer = pEnumLayer.Next
210:     Loop
211:   End If

213:   If booStandaloneTables Then
    Dim pSTCollection As IStandaloneTableCollection
215:     Set pSTCollection = pFocusMap
    Dim pStTble As IStandaloneTable
217:     If pSTCollection.StandaloneTableCount > 0 Then
      Dim lngIndex As Long
219:       For lngIndex = 0 To pSTCollection.StandaloneTableCount - 1
220:         Set pStTble = pSTCollection.StandaloneTable(lngIndex)
221:         ReturnLayersByType.Add pStTble
222:       Next lngIndex
```

```
223:     End If
224:   End If


  Exit Function
ErrorHandler:
  HandleError True, "ReturnLayersByType " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function BasicStatsFromVAT(anArray() As Double, dblSizeArray() As Double, _
      theFieldName As String, theTableName As String, _
      m_App As Application, Optional lngNumberHistBins As Long = -9999) As esriSystem.IVariantArray

  On Error GoTo ErrorHandler

  ' ASSUMES ARRAYS ARE SORTED!!!! ---------------------------------------

  Dim pResponse As esriSystem.IVariantArray
242:   Set pResponse = New esriSystem.VarArray

    ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
246:   Set psbar = m_App.StatusBar
  Dim pPro As IStepProgressor
248:   Set pPro = psbar.ProgressBar

  ' IF MAKING A HISTOGRAM
  Dim booMakeHistogram As Boolean
252:   booMakeHistogram = (lngNumberHistBins > 0)
  Dim lngHistCountArray() As Long            ' WILL CONTAIN COUNTS OF VALUES LYING IN EACH HISTOGRAM BIN
254:   If booMakeHistogram Then
    Dim dblHistLow As Double
    Dim dblHistHigh As Double
257:     dblHistLow = anArray(0)
258:     dblHistHigh = anArray(UBound(anArray))
    Dim dblInterval As Double
260:     dblInterval = (dblHistHigh - dblHistLow) / lngNumberHistBins

    ReDim lngHistCountArray(lngNumberHistBins)
    Dim lngHistBinIndex As Long
264:     lngHistBinIndex = 0
    Dim dblCurrentBinThreshold As Double
266:     dblCurrentBinThreshold = dblHistLow + dblInterval
267:   End If
```

```
269:   Screen.MousePointer = vbHourglass

271:   If booMakeHistogram Then
272:     psbar.ShowProgressBar "Calculating Statistics on field [" & theFieldName & "] in " & theTableName, 1, _
          4 * (UBound(anArray) - 1), 1, True
274:   Else
275:     psbar.ShowProgressBar "Calculating Statistics on field [" & theFieldName & "] in " & theTableName, 1, _
          3 * (UBound(anArray) - 1), 1, True
277:   End If

  Dim FoundMode As Boolean
280:   FoundMode = False

  Dim theModeValList() As Double
  ReDim theModeValList(1, UBound(anArray))

  Dim theHighModeCount As Long
286:   theHighModeCount = 0

  Dim anIndex As Long
  Dim theVal As Double
  Dim theSize As Double
  Dim theValTimesSize As Double
  Dim theModeCounter As Long
  Dim theModeIndex As Long
294:   theModeIndex = -1

296:   anIndex = 0
297:   psbar.StepProgressBar

  ' IF MAKING HISTOGRAM
300:   If booMakeHistogram Then
301:     Do While anIndex <= UBound(anArray)
302:       theVal = anArray(anIndex)
303:       theSize = dblSizeArray(anIndex)
304:       theValTimesSize = theVal * theSize
305:       If theVal <= dblCurrentBinThreshold Then
306:         lngHistCountArray(lngHistBinIndex) = lngHistCountArray(lngHistBinIndex) + theSize
307:       Else
308:         Do While dblCurrentBinThreshold < theVal
309:           dblCurrentBinThreshold = dblCurrentBinThreshold + dblInterval
310:           lngHistBinIndex = lngHistBinIndex + 1
311:         Loop
312:         lngHistCountArray(lngHistBinIndex) = theSize
313:       End If
314:       anIndex = anIndex + 1
315:       psbar.StepProgressBar
```

```
316:     Loop
317:   End If

  '  PASS 1:  MODE ----------------------------------------------------
320:   anIndex = 0

322:   Do While anIndex < UBound(anArray)

324:     theModeCounter = 0
325:     theVal = anArray(anIndex)
326:     theSize = dblSizeArray(anIndex)

328:     Do While (anIndex < UBound(anArray))
      ' IF THE NEXT VALUE UP IS DIFFERENT, THEN START NEW COUNT
330:       If Not (anArray(anIndex + 1) = theVal) Then
331:         theModeCounter = theSize
332:         Exit Do
333:       End If

      ' IF NEXT VALUE UP IS THE SAME, THEN ADD NEW SIZE TO CURRENT TALLY.  THEN CONTINUE LOOKING FOR A NEW VALUE.
336:       theModeCounter = theModeCounter + theSize

338:       anIndex = anIndex + 1
339:       theSize = dblSizeArray(anIndex)

341:       psbar.StepProgressBar
342:     Loop

344:     If theModeCounter > 1 Then
345:       FoundMode = True
346:       theModeIndex = theModeIndex + 1
347:       theModeValList(0, theModeIndex) = theModeCounter
348:       theModeValList(1, theModeIndex) = theVal

'      Debug.Print "Value = " & theModeValList(1, theModeIndex) & "[" & theModeValList(0, theModeIndex) & " cases]"

352:     End If

354:     anIndex = anIndex + 1
355:     psbar.StepProgressBar
356:   Loop

  Dim theModeString As String

  ' IF ANY VALUE OCCURED > 1 TIME
  Dim theFinalModes() As Double
362:   If FoundMode Then
```

```
    ReDim Preserve theModeValList(1, theModeIndex)

    Dim theFinalModeCount As Long
366:     theFinalModeCount = 0
    Dim theTempCount As Long
368:     theTempCount = 0

    Dim theFinalModeIndex As Long

372:     For anIndex = 0 To theModeIndex
373:       theTempCount = theModeValList(0, anIndex)
374:       If theTempCount > theFinalModeCount Then
375:         theFinalModeCount = theTempCount
        ReDim theFinalModes(0)
377:         theFinalModes(0) = theModeValList(1, anIndex)
378:       ElseIf theTempCount = theFinalModeCount Then
379:         theFinalModeIndex = UBound(theFinalModes) + 1
        ReDim Preserve theFinalModes(theFinalModeIndex)
381:         theFinalModes(theFinalModeIndex) = theModeValList(1, anIndex)
382:       End If
383:     Next anIndex

385:     If UBound(theFinalModes) > 0 Then
386:       theModeString = UBound(theFinalModes) + 1 & " modes found [" & aml_func_mod.InsertCommas(theFinalModeCount) & " cases
each]: Values = "
387:       For anIndex = 0 To UBound(theFinalModes)
388:         theModeString = theModeString & theFinalModes(anIndex) & ", "
389:       Next anIndex

391:       theModeString = aml_func_mod.BasicTrimAvenue(theModeString, "", ", ")

393:     Else
394:       theModeString = "1 mode found [" & aml_func_mod.InsertCommas(theFinalModeCount) & " cases]: Value = " & theFinalModes(0)
395:     End If

397:   Else
398:     theModeString = " < No Mode Found >"
399:   End If

  Dim theSum As Double
  Dim theCount As Double
  Dim theMinimum As Double
  Dim theMaximum As Double

406:   theSum = 0
407:   theCount = 0
408:   theMinimum = anArray(0)
```

```
409:   theMaximum = anArray(0)

  '  PASS 2:  MINIMUM, MAXIMUM AND SUM ----------------------------------------------------
412:   For anIndex = LBound(anArray) To UBound(anArray)

414:     psbar.StepProgressBar

416:     theVal = anArray(anIndex)
417:     theSize = dblSizeArray(anIndex)
418:     theCount = theCount + theSize

420:     theValTimesSize = theVal * theSize

422:     If theVal < theMinimum Then
423:       theMinimum = theVal
424:     End If
425:     If theVal > theMaximum Then
426:       theMaximum = theVal
427:     End If
428:     theSum = theSum + theValTimesSize

430:   Next anIndex

  Dim theMean As Double
433:   theMean = theSum / theCount

  Dim theSumSqDev As Double
  Dim theSqDev As Double
  Dim theMedian As Double
  Dim lngMiddleIndex As Long     ' DON'T HAVE TO WORRY ABOUT DECIMAL COUNTS BECAUSE WORKING WITH VAT.  ALL COUNTS ARE INTEGER.
  Dim theRunningCount As Double
440:   theRunningCount = 0
  Dim booFoundMedian As Boolean
442:   booFoundMedian = False

444:   If theCount = 0 Then
445:     lngMiddleIndex = 0
446:   ElseIf theCount Mod 2 = 0 Then      ' EVEN NUMBER
'      theMedian = (anArray((theCount / 2) - 1) + anArray(theCount / 2)) / 2
448:     lngMiddleIndex = (((theCount / 2) - 1) + (theCount / 2)) / 2
449:   Else
'      theMedian = anArray((theCount - 1) / 2)
451:     lngMiddleIndex = (theCount - 1) / 2
452:   End If

  '  PASS 2: MEDIAN, STANDARD DEVIATION AND VARIANCE ----------------------------------------------------
455:   For anIndex = LBound(anArray) To UBound(anArray)
```

```
457:     psbar.StepProgressBar

459:     theVal = anArray(anIndex)
460:     theSize = dblSizeArray(anIndex)
461:     theRunningCount = theRunningCount + theSize
462:     If Not booFoundMedian Then
463:       If theRunningCount >= lngMiddleIndex Then
464:         theMedian = theVal
465:         booFoundMedian = True
466:       End If
467:     End If
468:     theSqDev = theSize * ((theVal - theMean) * (theVal - theMean))
469:     theSumSqDev = theSqDev + theSumSqDev

471:   Next anIndex

  Dim theVariance As Double
  Dim theStDev As Double
  Dim theStErrMean As Double

477:   If theCount > 0 Then

479:     theVariance = theSumSqDev / (theCount - 1)
480:     theStDev = Sqr(theVariance)
481:     theStErrMean = theStDev / (Sqr(theCount))

483:   Else
484:     theMedian = -9999
485:     theVariance = 0
486:     theStDev = 0
487:     theStErrMean = 0
488:   End If

  Dim theRange As Double
491:   theRange = theMaximum - theMinimum

  ' OUTPUT ARRAY; VARIANT BECAUSE OF MODE STRING
  '(0) = SUM
  '(1) = MEAN
  '(2) = MINIMUM
  '(3) = MAXIMUM
  '(4) = RANGE
  '(5) = COUNT
  '(6) = STANDARD DEVIATION
  '(7) = VARIANCE
  '(8) = MEDIAN
```

```
  '(9) = STANDARD ERROR OF MEAN
  '(10) = MODE STRING
  '(11) = DOUBLE ARRAY OF MODE VALUES
  '(12) = BOOLEAN INDICATING WHETHER MODE WAS FOUND
  '(13) = ARRAY OF HISTOGRAM BIN COUNTS

509:   pResponse.Add theSum
510:   pResponse.Add theMean
511:   pResponse.Add theMinimum
512:   pResponse.Add theMaximum
513:   pResponse.Add theRange
514:   pResponse.Add theCount
515:   pResponse.Add theStDev
516:   pResponse.Add theVariance
517:   pResponse.Add theMedian
518:   pResponse.Add theStErrMean
519:   pResponse.Add theModeString
520:   pResponse.Add theFinalModes
521:   pResponse.Add FoundMode
522:   pResponse.Add lngHistCountArray  ' WILL BE EMPTY ARRAY IF NO HISTOGRAM WAS REQUESTED

524:   psbar.HideProgressBar
525:   Screen.MousePointer = vbDefault

527:   Set BasicStatsFromVAT = pResponse
  Exit Function
ErrorHandler:
  HandleError True, "BasicStatsFromVAT " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function BasicStatsFromArray_Weighted(anArray() As Double, theFieldName As String, theTableName As String, _
      m_App As Application) As esriSystem.IDoubleArray

  On Error GoTo ErrorHandler

  ' REQUIRES 2-DIMENSIONAL INPUT ARRAY, WHERE 1ST VALUE = VALUE AND SECOND VALUE = SIZE

  Dim pResponse As esriSystem.IDoubleArray
541:   Set pResponse = New esriSystem.DoubleArray

    ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
545:   Set psbar = m_App.StatusBar
  Dim pPro As IStepProgressor
547:   Set pPro = psbar.ProgressBar
```

```
549:   Screen.MousePointer = vbHourglass

551:   psbar.ShowProgressBar "Calculating Statistics on field [" & theFieldName & "] in " & theTableName, 1, _
        2 * (UBound(anArray) - 1), 1, True

  Dim anIndex As Long
  Dim theVal As Double
  Dim theWeight As Double
  Dim theWeightedVal As Double

559:   anIndex = 0
560:   psbar.StepProgressBar

  Dim theSum As Double
  Dim theCount As Double
  Dim theMinimum As Double
  Dim theMaximum As Double

567:   theCount = 0

  '  MINIMUM, MAXIMUM AND SUM ------------------------------------------------
570:   For anIndex = LBound(anArray) To UBound(anArray)

572:     psbar.StepProgressBar

574:     theVal = anArray(anIndex, 0)
575:     theWeight = anArray(anIndex, 1)

577:     theCount = theCount + theWeight
578:     theWeightedVal = theVal * theWeight
579:     theSum = theSum + theWeightedVal

581:   Next anIndex

  Dim theMean As Double
584:   theMean = theSum / theCount

  Dim theSumSqDev As Double
587:   theSumSqDev = 0
  Dim theSqDev As Double

  '  PASS 2:  STANDARD DEVIATION AND VARIANCE --------------------------------------------------
591:   For anIndex = LBound(anArray) To UBound(anArray)

593:     psbar.StepProgressBar

595:     theVal = anArray(anIndex, 0)
```

```
596:     theWeight = anArray(anIndex, 1)
597:     theSqDev = theWeight * ((theVal - theMean) * (theVal - theMean))
598:     theSumSqDev = theSqDev + theSumSqDev

600:   Next anIndex

  Dim theVariance As Double
  Dim theStDev As Double

605:   theVariance = theSumSqDev / theCount
606:   theStDev = Sqr(theVariance)

608:   pResponse.Add theMean
609:   pResponse.Add theStDev
610:   pResponse.Add theVariance

612:   psbar.HideProgressBar

614:   Screen.MousePointer = vbDefault

616:   Set BasicStatsFromArray_Weighted = pResponse

  Exit Function
ErrorHandler:
  HandleError True, "BasicStatsFromArray_Weighted " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function BasicStatsFromArray(anArray() As Double, theFieldName As String, theTableName As String, _
      m_App As Application, Optional lngNumberHistBins As Long = -9999) As esriSystem.IVariantArray

  On Error GoTo ErrorHandler

  ' ASSUMES ARRAY IS SORTED!!!! -------------------------------------

  Dim pResponse As esriSystem.IVariantArray
631:   Set pResponse = New esriSystem.VarArray

    ' PROGRESS BAR STUFF
  Dim psbar As IStatusBar
635:   Set psbar = m_App.StatusBar
  Dim pPro As IStepProgressor
637:   Set pPro = psbar.ProgressBar

  ' IF MAKING A HISTOGRAM
  Dim booMakeHistogram As Boolean
641:   booMakeHistogram = (lngNumberHistBins > 0)
```

```
  Dim lngHistCountArray() As Long            ' WILL CONTAIN COUNTS OF VALUES LYING IN EACH HISTOGRAM BIN
643:   If booMakeHistogram Then
    Dim dblHistLow As Double
    Dim dblHistHigh As Double
646:     dblHistLow = anArray(0)
647:     dblHistHigh = anArray(UBound(anArray))
    Dim dblInterval As Double
649:     dblInterval = (dblHistHigh - dblHistLow) / lngNumberHistBins

    ReDim lngHistCountArray(lngNumberHistBins)
    Dim lngHistBinIndex As Long
653:     lngHistBinIndex = 0
    Dim dblCurrentBinThreshold As Double
655:     dblCurrentBinThreshold = dblHistLow + dblInterval
656:   End If

658:   Screen.MousePointer = vbHourglass

660:   If booMakeHistogram Then
661:     psbar.ShowProgressBar "Calculating Statistics on field [" & theFieldName & "] in " & theTableName, 1, _
          4 * (UBound(anArray) - 1), 1, True
663:   Else
664:     psbar.ShowProgressBar "Calculating Statistics on field [" & theFieldName & "] in " & theTableName, 1, _
          3 * (UBound(anArray) - 1), 1, True
666:   End If

  Dim FoundMode As Boolean
669:   FoundMode = False

  Dim theModeValList() As Double
  ReDim theModeValList(1, UBound(anArray))

  Dim theHighModeCount As Long
675:   theHighModeCount = 0

  Dim anIndex As Long
  Dim theVal As Double
  Dim theModeCounter As Long
  Dim theModeIndex As Long
681:   theModeIndex = -1

683:   anIndex = 0
684:   psbar.StepProgressBar

  ' IF MAKING HISTOGRAM
687:   If booMakeHistogram Then
688:     Do While anIndex <= UBound(anArray)
```

```
689:       theVal = anArray(anIndex)
690:       If theVal <= dblCurrentBinThreshold Then
691:         lngHistCountArray(lngHistBinIndex) = lngHistCountArray(lngHistBinIndex) + 1
692:       Else
693:         Do While dblCurrentBinThreshold < theVal
694:           dblCurrentBinThreshold = dblCurrentBinThreshold + dblInterval
695:           lngHistBinIndex = lngHistBinIndex + 1
696:         Loop
697:         lngHistCountArray(lngHistBinIndex) = 1
698:       End If
699:       anIndex = anIndex + 1
700:       psbar.StepProgressBar
701:     Loop
702:   End If

  '  PASS 1:  MODE ------------------------------------------------
705:   anIndex = 0

707:   Do While anIndex < UBound(anArray)

709:     theModeCounter = 1
710:     theVal = anArray(anIndex)

712:     Do While (anIndex < UBound(anArray))
713:       If Not (anArray(anIndex + 1) = theVal) Then
714:         Exit Do
715:       End If
716:       anIndex = anIndex + 1
717:       psbar.StepProgressBar
718:       theModeCounter = theModeCounter + 1
719:     Loop

721:     If theModeCounter > 1 Then
722:       FoundMode = True
723:       theModeIndex = theModeIndex + 1
724:       theModeValList(0, theModeIndex) = theModeCounter
725:       theModeValList(1, theModeIndex) = theVal

'      Debug.Print "Value = " & theModeValList(1, theModeIndex) & "[" & theModeValList(0, theModeIndex) & " cases]"

729:     End If

731:     anIndex = anIndex + 1
732:     psbar.StepProgressBar
733:   Loop

  Dim theModeString As String
```

```
  ' IF ANY VALUE OCCURED > 1 TIME
  Dim theFinalModes() As Double
739:   If FoundMode Then
    ReDim Preserve theModeValList(1, theModeIndex)

    Dim theFinalModeCount As Long
743:     theFinalModeCount = 0
    Dim theTempCount As Long
745:     theTempCount = 0

    Dim theFinalModeIndex As Long

749:     For anIndex = 0 To theModeIndex
750:       theTempCount = theModeValList(0, anIndex)
751:       If theTempCount > theFinalModeCount Then
752:         theFinalModeCount = theTempCount
        ReDim theFinalModes(0)
754:         theFinalModes(0) = theModeValList(1, anIndex)
755:       ElseIf theTempCount = theFinalModeCount Then
756:         theFinalModeIndex = UBound(theFinalModes) + 1
        ReDim Preserve theFinalModes(theFinalModeIndex)
758:         theFinalModes(theFinalModeIndex) = theModeValList(1, anIndex)
759:       End If
760:     Next anIndex

762:     If UBound(theFinalModes) > 0 Then
763:       theModeString = UBound(theFinalModes) + 1 & " modes found [" & aml_func_mod.InsertCommas(theFinalModeCount) & " cases
each]: Values = "
764:       For anIndex = 0 To UBound(theFinalModes)
765:         theModeString = theModeString & theFinalModes(anIndex) & ", "
766:       Next anIndex

768:       theModeString = aml_func_mod.BasicTrimAvenue(theModeString, "", ", ")

770:     Else
771:       theModeString = "1 mode found [" & aml_func_mod.InsertCommas(theFinalModeCount) & " cases]: Value = " & theFinalModes(0)
772:     End If

774:   Else
775:     theModeString = " < No Mode Found >"
776:   End If

  Dim theSum As Double
  Dim theCount As Double
  Dim theMinimum As Double
  Dim theMaximum As Double
```

```
783:   theSum = 0
784:   theCount = UBound(anArray) + 1         ' ARRAY INDEX STARTS AT 0
785:   theMinimum = anArray(0)
786:   theMaximum = anArray(0)

  '  PASS 2:  MINIMUM, MAXIMUM AND SUM ------------------------------------------------
789:   For anIndex = LBound(anArray) To UBound(anArray)

791:     psbar.StepProgressBar

793:     theVal = anArray(anIndex)

795:     If theVal < theMinimum Then
796:       theMinimum = theVal
797:     End If
798:     If theVal > theMaximum Then
799:       theMaximum = theVal
800:     End If
801:     theSum = theSum + theVal

803:   Next anIndex

  Dim theMean As Double
806:   theMean = theSum / theCount

  Dim theSumSqDev As Double
  Dim theSqDev As Double

  '  PASS 2:  STANDARD DEVIATION AND VARIANCE -----------------------------------------------
812:   For anIndex = LBound(anArray) To UBound(anArray)

814:     psbar.StepProgressBar

816:     theVal = anArray(anIndex)
817:     theSqDev = (theVal - theMean) * (theVal - theMean)
818:     theSumSqDev = theSqDev + theSumSqDev

820:   Next anIndex

  Dim theMedian As Double
  Dim theVariance As Double
  Dim theStDev As Double
  Dim theStErrMean As Double

827:   If theCount > 1 Then
828:     If theCount Mod 2 = 0 Then      ' EVEN NUMBER
```

```
829:       theMedian = (anArray((theCount / 2) - 1) + anArray(theCount / 2)) / 2
830:     Else
831:       theMedian = anArray((theCount - 1) / 2)
832:     End If

834:     theVariance = theSumSqDev / (theCount - 1)
835:     theStDev = Sqr(theVariance)
836:     theStErrMean = theStDev / (Sqr(theCount))

838:   Else
839:     theMedian = -9999
840:     theVariance = 0
841:     theStDev = 0
842:     theStErrMean = 0
843:   End If

  Dim theRange As Double
846:   theRange = theMaximum - theMinimum

  ' OUTPUT ARRAY; VARIANT BECAUSE OF MODE STRING
  '(0) = SUM
  '(1) = MEAN
  '(2) = MINIMUM
  '(3) = MAXIMUM
  '(4) = RANGE
  '(5) = COUNT
  '(6) = STANDARD DEVIATION
  '(7) = VARIANCE
  '(8) = MEDIAN
  '(9) = STANDARD ERROR OF MEAN
  '(10) = MODE STRING
  '(11) = DOUBLE ARRAY OF MODE VALUES
  '(12) = BOOLEAN INDICATING WHETHER MODE WAS FOUND
  '(13) = ARRAY OF HISTOGRAM BIN COUNTS

864:   pResponse.Add theSum
865:   pResponse.Add theMean
866:   pResponse.Add theMinimum
867:   pResponse.Add theMaximum
868:   pResponse.Add theRange
869:   pResponse.Add theCount
870:   pResponse.Add theStDev
871:   pResponse.Add theVariance
872:   pResponse.Add theMedian
873:   pResponse.Add theStErrMean
874:   pResponse.Add theModeString
875:   pResponse.Add theFinalModes
```

```
876:   pResponse.Add FoundMode
877:   pResponse.Add lngHistCountArray  ' WILL BE EMPTY ARRAY IF NO HISTOGRAM WAS REQUESTED

879:   psbar.HideProgressBar
880:   Screen.MousePointer = vbDefault

882:   Set BasicStatsFromArray = pResponse
  Exit Function
ErrorHandler:
  HandleError True, "BasicStatsFromArray " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function CalcStatistics(dblSortedNumbers() As Double, pStatOptions As esriSystem.IVariantArray, _
      Optional booReportProgress As Boolean, Optional pApp As IApplication) As esriSystem.IVariantArray
  On Error GoTo ErrorHandler


  ' FAO_WRD.Stat_CalcFieldStats
'
'Dim chkMean As Boolean
'Set chkMean = pStatOptions.Element(0)
'Dim chkSEMean As Boolean
'Set chkSEMean = pStatOptions.Element(1)
'Dim chkCIMean As Boolean
'Set chkCIMean = pStatOptions.Element(2)
'Dim ConLevel As Double
'Set ConLevel = pStatOptions.Element(3)
'Dim chkMinimum As Boolean
'Set chkMinimum = pStatOptions.Element(4)
'Dim chkQ1 As Boolean
'Set chkQ1 = pStatOptions.Element(5)
'Dim chkMedian As Boolean
'Set chkMedian = pStatOptions.Element(6)
'Dim chkQ3 As Boolean
'Set chkQ3 = pStatOptions.Element.Item(7)
'Dim chkMaximum As Boolean
'Set chkMaximum = pStatOptions.Element(8)
'Dim chkVariance As Boolean
'Set chkVariance = pStatOptions.Element(9)
'Dim chkStDev As Boolean
'Set chkStDev = pStatOptions.Element(10)
'Dim chkAvgDev As Boolean
'Set chkAvgDev = pStatOptions.Element(11)
'Dim chkSkewness As Boolean
'Set chkSkewness = pStatOptions.Element(12)
```

```
'Dim chkSkewnessFish As Boolean
'Set chkSkewnessFish = pStatOptions.Element(13)
'Dim chkKurtosis As Boolean
'Set chkKurtosis = pStatOptions.Element(14)
'Dim chkKurtosisFish As Boolean
'Set chkKurtosisFish = pStatOptions.Element(15)
'Dim chkCount As Boolean
'Set chkCount = pStatOptions.Element(16)
'Dim chkNumberNull As Boolean
'Set chkNumberNull = pStatOptions.Element(17)
'Dim chkSum As Boolean
'Set chkSum = pStatOptions.Element(18)
'Dim chkRange As Boolean
'Set chkRange = pStatOptions.Element(19)
'Dim chkMode As Boolean
'Set chkMode = pStatOptions.Element(20)
'
'' MakeHistogram = theResults.Get(21)
'
'Dim pResponse As esriSystem.IVariantArray
'Dim anIndex As Long
'For anIndex = 0 To 20
'  pResponse.Add Nothing
'Next anIndex
'
''' UNFORTUNATELY, THE HISTOGRAM FUNCTION WAS WRITTEN SEPARATELY AND THE STATS ARE IN A DIFFERENT ORDER
''theResponse = {nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil}
''theStatsForHistogramScript = {nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil, nil,
nil}
''
''If (MakeHistogram) Then
''  chkMean = True
''  chkStDev = True
''  chkCount = True
''  chkMinimum = True
''  chkMaximum = True
''  theResults.Set(0,True)
''  theResults.Set(4,True)
''  theResults.Set(8,True)
''  theResults.Set(10,True)
''  theResults.Set(16,True)
''End
'
''' HELP IT RUN FASTER IF ONLY A FEW OPTIONS ARE CHECKED
'Dim DoQuantiles As Boolean
'Dim DoSkewKurt As Boolean
'Dim MoreThanBasic As Boolean
```

```
'Dim lngMultiplier As Long
'lngMultiplier = 1   ' FOR BASIC STATS
'
'' DIMENSION STATISTIC VARIABLES
'Dim theMean As Double
'Dim theSEMean As Double
'Dim LowerCI As Double
'Dim UpperCI As Double
'Dim theMinimum As Double
'Dim theQ1 As Double
'Dim theMedian As Double
'Dim theQ3 As Double
'Dim theMaximum As Double
'Dim theVar As Double
'Dim theStdDev As Double
'Dim theAvgDev As Double
'Dim theSkew As Double
'Dim theFisherSkew As Double
'Dim theKurt As Double
'Dim theFisherKurt As Double
'Dim theCount As Double
'Dim theNumberNull As Double
'Dim theSum As Double
'Dim theRange As Double
'Dim theModeString As String
'
'' BASIC STATS ARE Count, Minimum, Maximum, Mean, Sum
'DoQuantiles = (chkQ1 Or chkMedian Or chkQ3)
'DoSkewKurt = (chkSkewness Or chkSkewnessFish Or chkKurtosis Or chkKurtosisFish)
''MoreThanBasic = (DoQuantiles Or DoSkewKurt Or chkSEMean Or chkCIMean Or chkVariance Or chkStDev Or chkAvgDev Or MakeHistogram)
'MoreThanBasic = (DoQuantiles Or DoSkewKurt Or chkSEMean Or chkCIMean Or chkVariance Or chkStDev Or chkAvgDev)
'
'If chkVariance Or chkStDev Or chkAvgDev Then lngMultiplier = lngMultiplier + 1
'
'If DoQuantiles Then lngMultiplier = lngMultiplier + 1
'If DoSkewKurt Then lngMultiplier = lngMultiplier + 1
'If MoreThanBasic Then lngMultiplier = lngMultiplier + 1
'If chkMode Then lngMultiplier = lngMultiplier + 1
'
'If booReportProgress Then
'    ' PROGRESS BAR STUFF
'  Dim psbar As IStatusBar
'  Set psbar = pApp.StatusBar
'  Dim pPro As IStepProgressor
'  Set pPro = psbar.ProgressBar
'
'  Screen.MousePointer = vbHourglass
```

```
'
'  psbar.ShowProgressBar "Calculating Statistics on field [" & theFieldName & "] in " & theTableName, 1, _
'         lngMultiplier * (UBound(dblSortedNumbers) - 1), 1, True
'End If
'
'Dim theCount As Long
'theCount = UBound(dblSortedNumbers) - 1
'
'If chkMode Then  ' ASSUMES NUMBER ARRAY IS SORTED!
'
'  Dim pFoundMode As Boolean
'  Dim theModeValList() As Double
'  ReDim theModeValList(1, UBound(dblSortedNumbers))
'
'  Dim theHighModeCount As Long
'  theHighModeCount = 0
'
'  Dim theVal As Double
'  Dim theModeCounter As Long
'  Dim theModeIndex As Long
'  theModeIndex = -1
'
'  anIndex = 0
'
'  Do While anIndex < UBound(dblSortedNumbers)
'
'    theModeCounter = 1
'    theVal = dblSortedNumbers(anIndex)
'
'    Do While (anIndex < UBound(dblSortedNumbers))
'      If Not (dblSortedNumbers(anIndex + 1) = theVal) Then
'        Exit Do
'      End If
'      anIndex = anIndex + 1
'      If booReportProgress Then psbar.StepProgressBar
'      theModeCounter = theModeCounter + 1
'    Loop
'
'    If theModeCounter > 1 Then
'      FoundMode = True
'      theModeIndex = theModeIndex + 1
'      theModeValList(0, theModeIndex) = theModeCounter
'      theModeValList(1, theModeIndex) = theVal
'
''       Debug.Print "Value = " & theModeValList(1, theModeIndex) & "[" & theModeValList(0, theModeIndex) & " cases]"
'
'    End If
```

```
'
'    anIndex = anIndex + 1
'    If booReportProgress Then psbar.StepProgressBar
'  Loop
'
'  Dim theModeString As String
'
'  ' IF ANY VALUE OCCURED > 1 TIME
'  If FoundMode Then
'    ReDim Preserve theModeValList(1, theModeIndex)
'    Dim theFinalModes() As Double
'    Dim theFinalModeCount As Long
'    theFinalModeCount = 0
'    Dim theTempCount As Long
'    theTempCount = 0
'
'    Dim theFinalModeIndex As Long
'
'    For anIndex = 0 To theModeIndex
'      theTempCount = theModeValList(0, anIndex)
'      If theTempCount > theFinalModeCount Then
'        theFinalModeCount = theTempCount
'        ReDim theFinalModes(0)
'        theFinalModes(0) = theModeValList(1, anIndex)
'      ElseIf theTempCount = theFinalModeCount Then
'        theFinalModeIndex = UBound(theFinalModes) + 1
'        ReDim Preserve theFinalModes(theFinalModeIndex)
'        theFinalModes(theFinalModeIndex) = theModeValList(1, anIndex)
'      End If
'    Next anIndex
'
'    If UBound(theFinalModes) > 0 Then
'      theModeString = UBound(theFinalModes) + 1 & " modes found [" & aml_func_mod.InsertCommas(theFinalModeCount) & " cases each]:
Values = "
'      For anIndex = 0 To UBound(theFinalModes)
'        theModeString = theModeString & theFinalModes(anIndex) & ", "
'      Next anIndex
'
'      theModeString = aml_func_mod.BasicTrimAvenue(theModeString, "", ", ")
'
'    Else
'      theModeString = "1 mode found [" & aml_func_mod.InsertCommas(theFinalModeCount) & " cases]: Value = " & theFinalModes(0)
'    End If
'
'  Else
'    theModeString = " < No Mode Found >"
'  End If
```

```
'End If
'
'theSum = 0
'theCount = UBound(dblSortedNumbers) + 1         ' ARRAY INDEX STARTS AT 0
'theMinimum = dblSortedNumbers(0)
'theMaximum = dblSortedNumbers(0)
'
''  PASS 2:  MINIMUM, MAXIMUM AND SUM ------------------------------------------------
'For anIndex = LBound(dblSortedNumbers) To UBound(dblSortedNumbers)
'
'  If booReportProgress Then psbar.StepProgressBar
'
'  theVal = dblSortedNumbers(anIndex)
'
'  If theVal < theMinimum Then
'    theMinimum = theVal
'  End If
'  If theVal > theMaximum Then
'    theMaximum = theVal
'  End If
'  theSum = theSum + theVal
'
'Next anIndex
'
'Dim theMean As Double
'theMean = theSum / theCount
'
'Dim theSumSqDev As Double
'Dim theSqDev As Double
'
''  PASS 2:  STANDARD DEVIATION AND VARIANCE ------------------------------------------
'For anIndex = LBound(dblSortedNumbers) To UBound(dblSortedNumbers)
'
'  If booReportProgress Then psbar.StepProgressBar
'
'  theVal = dblSortedNumbers(anIndex)
'  theSqDev = (theVal - theMean) * (theVal - theMean)
'  theSumSqDev = theSqDev + theSumSqDev
'
'Next anIndex
'
'Dim theMedian As Double
'Dim theVariance As Double
'Dim theStDev As Double
'Dim theStErrMean As Double
'
'If theCount > 1 Then
```

```
'  If theCount Mod 2 = 0 Then      ' EVEN NUMBER
'    theMedian = (dblSortedNumbers((theCount / 2) - 1) + dblSortedNumbers(theCount / 2)) / 2
'  Else
'    theMedian = dblSortedNumbers((theCount - 1) / 2)
'  End If
'
'  theVariance = theSumSqDev / (theCount - 1)
'  theStDev = Sqr(theVariance)
'  theStErrMean = theStDev / (Sqr(theCount))
'
'Else
'  theMedian = -9999
'  theVariance = 0
'  theStDev = 0
'  theStErrMean = 0
'End If
'
'Dim theRange As Double
'theRange = theMaximum - theMinimum
'
'' OUTPUT ARRAY; VARIANT BECAUSE OF MODE STRING
''(0) = SUM
''(1) = MEAN
''(2) = MINIMUM
''(3) = MAXIMUM
''(4) = RANGE
''(5) = COUNT
''(6) = STANDARD DEVIATION
''(7) = VARIANCE
''(8) = MEDIAN
''(9) = STANDARD ERROR OF MEAN
''(10) = MODE STRING
'
'theStatsArray(0) = theSum
'theStatsArray(1) = theMean
'theStatsArray(2) = theMinimum
'theStatsArray(3) = theMaximum
'theStatsArray(4) = theRange
'theStatsArray(5) = theCount
'theStatsArray(6) = theStDev
'theStatsArray(7) = theVariance
'theStatsArray(8) = theMedian
'theStatsArray(9) = theStErrMean
'theStatsArray(10) = theModeString
'
''If (chkMode) Then
''  FoundMode = False
```

```
''  theModeValList = {}
''  theHighModeCount = 0
''  theDictionaryOfValues = Dictionary.Make(theCount)
''End
''
''theList = {}
''theNumberNull = 0
''theSum = 0
''theCounter = 0
''av.ClearStatus
''
''av.ShowMsg ("Calculating Pass 1")
''For Each aRecord In theSelection
''  theCounter = theCounter + 1
''  av.SetStatus ((theCounter / theCount) * 100)
''  theX = theVTab.ReturnValue(theField, aRecord)
''  If (theX.IsNull) Then
''    theNumberNull = theNumberNull + 1
''  Else
''    theList.Add (theX)
''    theSum = theSum + theX
''
''    ' CALCULATE MODE
''    If (chkMode) Then
''      theModeCount = theDictionaryOfValues.Get(theX)
''      If (theModeCount = nil) Then
''        theModeCount = 1
''      Else
''        theModeCount = theModeCount + 1
''        FoundMode = True
''      End
''      theDictionaryOfValues.Set(theX, theModeCount)
''
''      If (theModeCount > theHighModeCount) Then
''        theModeValList = {theX}
''        theHighModeCount = theModeCount
''      ElseIf (theModeCount = theHighModeCount) Then
''        theModeValList.Add (theX)
''      End
''    End
''  End
''End
''av.ClearStatus
''
''If (theList.Count = 0) Then
''  msgBox.Info("No data available to calculate!  Bailing out...", "")
''  return nil
```

```
''End
''
''If (chkMode) Then
''  If (FoundMode) Then
''    theModeValList.RemoveDuplicates
''    theModeValList.Sort (True)
''    theModeString = ""
''    For Each aModeVal In theModeValList
''      theModeString = theModeString + aModeVal.AsString + ", "
''    End
''    theModeString = theModeString.Left(theModeString.Count - 2)
''  Else
''    theModeString = " < No Mode Found >"
''  End
''End
''
''theTotalCount = theCount - theNumberNull
''
''theMean = theSum / theTotalCount
''
''If (MoreThanBasic) Then
''  av.ShowMsg ("Second Pass")
''  theCounter = 0
''
''  theAvgDev = 0
''  theMoment2 = 0
''  theMoment3 = 0
''  theMoment4 = 0
''  For Each anX In theList
''    theCounter = theCounter + 1
''    av.SetStatus ((theCounter / theTotalCount) * 100)
''    theDev = anX - theMean
''    theAvgDev = theAvgDev + (theDev.Abs)
''    theMoment2 = theMoment2 + (theDev ^ 2)
''    If (DoSkewKurt) Then
''      theMoment3 = theMoment3 + (theDev ^ 3)
''      theMoment4 = theMoment4 + (theDev ^ 4)
''    End
''  End
''  av.ClearStatus
''
''  ' MEAN, STANDARD DEVIATION, SKEWNESS, KURTOSIS --------------------------------
''  theVar = theMoment2 / (theTotalCount - 1)
''  theStdDev = theVar.Sqrt
''  theAvgDev = theAvgDev / theTotalCount
''
''  theMoment2 = theMoment2 / theTotalCount
```

```
''
''  If (DoSkewKurt) Then
''    theMoment3 = theMoment3 / theTotalCount
''    theMoment4 = theMoment4 / theTotalCount
''
''    theSkew = theMoment3 / (theMoment2 ^ (3 / 2))
''    theFisherSkew  = (theSkew*((theTotalCount*(theTotalCount-1)).sqrt))/(theTotalCount-2)
''
''    theKurt = theMoment4 / (theMoment2 ^ 2)
''    theFisherKurt = ((theTotalCount + 1) * (theTotalCount - 1)) / ((theTotalCount - 2) * (theTotalCount - 3))
''    theFisherKurt = theFisherKurt * (theKurt - (3 * (theTotalCount - 1) / (theTotalCount + 1)))
''
''    If (theVar <> 0) Then
''      theSkew = theMoment3 / (theMoment2 ^ (3 / 2))
''      theFisherSkew  = (theSkew*((theTotalCount*(theTotalCount-1)).sqrt))/(theTotalCount-2)
''
''      theKurt = theMoment4 / (theMoment2 ^ 2)
''      theFisherKurt = ((theTotalCount + 1) * (theTotalCount - 1)) / ((theTotalCount - 2) * (theTotalCount - 3))
''      theFisherKurt = theFisherKurt * (theKurt - (3 * (theTotalCount - 1) / (theTotalCount + 1)))
''    Else
''      theSkew = Number.MakeNull
''      theKurt = Number.MakeNull
''    End
''  End
''
''  ' STANDARD ERROR OF MEAN, CONFIDENCE INTERVALS
''  theSEMean = theStdDev / (theTotalCount.Sqrt)
''  If (chkCIMean) Then
''    theAlphaOver2 = 1 - ((1 - ConLevel) / 2)
''    theT = av.Run("FAO_WRD.Stat_IDF_StudentsT", {theAlphaOver2, (theTotalCount-1)})
''    theFactor = theSEMean * theT
''    LowerCI = theMean - theFactor
''    UpperCI = theMean + theFactor
''  End
''
''  ' QUANTILE DATA ------------------------------------------------------------------
''  If (DoQuantiles) Then
''
''    av.ShowMsg ("Calculating Quantile Data...")
''
''    theList.Sort (True)
''    theMinimum = theList.Get(0)
''    theMaximum = theList.Get(theList.Count - 1)
''    theRange = theMaximum - theMinimum
''
''    theListCount = theList.Count
''    theQ1Index = (theListCount + 1) * 0.25
```

```
''    theQ2Index = (theListCount + 1) * 0.5
''    theQ3Index = (theListCount + 1) * 0.75
''
''    If (theQ1Index.Round = theQ1Index) Then
''      theQ1 = theList.Get(theQ1Index - 1)
''    Else
''      theFloor = theList.Get((theQ1Index-1).Floor.Max(0))    ' POSSIBLE THAT IT COULD TRY TO GET INDEX -1, SO MAX IT WITH 0
''      theCeiling = theList.Get((theQ1Index-1).Ceiling)
''      theQ1 = (theFloor + theCeiling) / 2
''    End
''
''    If (theQ2Index.Round = theQ2Index) Then
''      theMedian = theList.Get(theQ2Index - 1)
''    Else
''      theFloor = theList.Get((theQ2Index-1).Floor)
''      theCeiling = theList.Get((theQ2Index-1).Ceiling)
''      theMedian = (theFloor + theCeiling) / 2
''    End
''
''    If (theQ3Index.Round = theQ3Index) Then
''      theQ3 = theList.Get(theQ3Index - 1)
''    Else
''      theFloor = theList.Get((theQ3Index-1).Floor)
''      theCeiling = theList.Get((theQ3Index-1).Ceiling.Min(theListCount-1))  ' POSSIBLE THAT IT COULD TRY TO GET INDEX (COUNT+1), SO
MIN IT WITH COUNT
''
''      theQ3 = (theFloor + theCeiling) / 2
''    End
''  End
''End   ' END MORE THAN BASIC
''
'
'
'
'If booReportProgress Then
'  psbar.HideProgressBar
'  Screen.MousePointer = vbDefault
'End If
'
'If chkMean Then pResponse.Element(0) = theMean
'If chkSEMean Then pResponse.Element(1) = theSEMean
'If chkCIMean Then pResponse.Element(2) = LowerCI
'If chkCIMean Then pResponse.Element(3) = UpperCI
'If chkMinimum Then pResponse.Element(4) = theMinimum
'If chkQ1 Then pResponse.Element(5) = theQ1
'If chkMedian Then pResponse.Element(6) = theMedian
'If chkQ3 Then pResponse.Element(7) = theQ3
```

```
'If chkMaximum Then pResponse.Element(8) = theMaximum
'If chkVariance Then pResponse.Element(9) = theVar
'If chkStDev Then pResponse.Element(10) = theStdDev
'If chkAvgDev Then pResponse.Element(11) = theAvgDev
'If chkSkewness Then pResponse.Element(12) = theSkew
'If chkSkewnessFish Then pResponse.Element(13) = theFisherSkew
'If chkKurtosis Then pResponse.Element(14) = theKurt
'If chkKurtosisFish Then pResponse.Element(15) = theFisherKurt
'If chkCount Then pResponse.Element(16) = theCount
'If chkNumberNull Then pResponse.Element(17) = theNumberNull
'If chkSum Then pResponse.Element(18) = theSum
'If chkRange Then pResponse.Element(19) = theRange
'If chkMode Then pResponse.Element(20) = theModeString
'
'
'




  Exit Function
ErrorHandler:
  HandleError True, "CalcStatistics " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function Graphic_ReturnElementFromGeometry(ByRef pMxDoc As IMxDocument, ByRef pGeometry As IGeometry, _
    Optional strName As String, Optional AddToView As Boolean) As IElement
  On Error GoTo ErrorHandler


  Dim pMxDocument As esriArcMapUI.IMxDocument
  Dim pActiveView As esriCarto.IActiveView

  Dim pGContainer As IGraphicsContainer
1423:   Set pGContainer = pMxDoc.FocusMap

  Dim pElement As IElement
  Dim pPolygonElement As IPolygonElement
  Dim pSpatialReference As ISpatialReference
  Dim pGraphicElement As IGraphicElement
  Dim pElementProperties As IElementProperties

  Dim pClone As IClone
1432:   Set pClone = pGeometry
  Dim pNewGeometry As IGeometry
1434:   Set pNewGeometry = pClone.Clone
```

```
  Dim pGeometryType As esriGeometryType
1437:   pGeometryType = pNewGeometry.GeometryType

  'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
  Select Case pGeometryType
    Case 0:
1442:       MsgBox "Null Geometry!  No graphic added..."
    Case 1, 2:
1444:       Set pElement = New MarkerElement
    Case 3, 6, 13, 14, 15, 16:
1446:       Set pElement = New LineElement
    Case 4, 11:
1448:       Set pElement = New PolygonElement
    Case 5:
1450:       Set pElement = New RectangleElement
    Case Else:
1452:       MsgBox "Unexpected Shape Type:  Unable to add graphic..."
      Exit Function
1454:   End Select


1457:   If pGeometryType = 2 Then
    Dim pGroupElement As IGroupElement2
1459:     Set pGroupElement = New GroupElement
    Dim pSubElement As IElement
1461:     Set pSubElement = New MarkerElement
    Dim lngIndex As Long
    Dim pPtColl As IPointCollection
1464:     Set pPtColl = pNewGeometry
    Dim pPt As IPoint
1466:     For lngIndex = 0 To pPtColl.PointCount - 1
1467:       Set pPt = pPtColl.Point(lngIndex)
1468:       Set pSubElement = New MarkerElement
1469:       pSubElement.Geometry = pPt
1470:       pGroupElement.AddElement pSubElement
1471:     Next lngIndex
1472:     Set pGraphicElement = pGroupElement
1473:     Set pSpatialReference = pGeometry.SpatialReference
1474:     Set pGraphicElement.SpatialReference = pSpatialReference
1475:     Set pElementProperties = pGroupElement
1476:     pElementProperties.Name = strName
1477:     If AddToView Then
      ' ADD GRAPHIC TO GRAPHICS CONTAINER
1479:       pGContainer.AddElement pGroupElement, 0
      'Draw
1481:       pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing
```

```
1482:     End If

1484:   Else
1485:     pElement.Geometry = pNewGeometry
1486:     Set pGraphicElement = pElement
1487:     Set pSpatialReference = pGeometry.SpatialReference
1488:     Set pGraphicElement.SpatialReference = pSpatialReference
1489:     Set pElementProperties = pElement
1490:     pElementProperties.Name = strName

1492:     If AddToView Then
      ' ADD GRAPHIC TO GRAPHICS CONTAINER
1494:       pGContainer.AddElement pElement, 0
      'Draw
1496:       pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing
1497:     End If

1499:   End If


1502:   Set Graphic_ReturnElementFromGeometry = pElement


  Exit Function
ErrorHandler:
  HandleError True, "Graphic_ReturnElementFromGeometry " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number,
Err.Source, Err.Description, 4
End Function
Public Sub Graphic_MakeFromGeometry(ByRef pMxDoc As IMxDocument, ByRef pGeometry As IGeometry, Optional strName As String)
  On Error GoTo ErrorHandler


  Dim pMxDocument As esriArcMapUI.IMxDocument
  Dim pActiveView As esriCarto.IActiveView

  Dim pGContainer As IGraphicsContainer
1517:   Set pGContainer = pMxDoc.FocusMap

  Dim pElement As IElement
  Dim pPolygonElement As IPolygonElement
  Dim pSpatialReference As ISpatialReference
  Dim pGraphicElement As IGraphicElement
  Dim pElementProperties As IElementProperties

  Dim pClone As IClone
1526:   Set pClone = pGeometry
  Dim pNewGeometry As IGeometry
```

```
1528:   Set pNewGeometry = pClone.Clone

  Dim pGeometryType As esriGeometryType
1531:   pGeometryType = pNewGeometry.GeometryType

  'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
  Select Case pGeometryType
    Case 0:
1536:       MsgBox "Null Geometry!  No graphic added..."
    Case 1:
1538:       Set pElement = New MarkerElement
    Case 2:
1540:       Set pElement = New GroupElement
    Case 3, 6, 13, 14, 15, 16:
1542:       Set pElement = New LineElement
    Case 4, 11:
1544:       Set pElement = New PolygonElement
    Case 5:
1546:       Set pElement = New RectangleElement
    Case Else:
1548:       MsgBox "Unexpected Shape Type:  Unable to add graphic..."
1549:   End Select

1551:   If pGeometryType = 2 Then
    Dim pGroupElement As IGroupElement2
1553:     Set pGroupElement = New GroupElement
    Dim pSubElement As IElement
1555:     Set pSubElement = New MarkerElement
    Dim lngIndex As Long
    Dim pPtColl As IPointCollection
1558:     Set pPtColl = pNewGeometry
    Dim pPt As IPoint
1560:     For lngIndex = 0 To pPtColl.PointCount - 1
1561:       Set pPt = pPtColl.Point(lngIndex)
1562:       Set pSubElement = New MarkerElement
1563:       pSubElement.Geometry = pPt
1564:       pGroupElement.AddElement pSubElement
1565:     Next lngIndex
1566:     Set pGraphicElement = pGroupElement
1567:     Set pSpatialReference = pGeometry.SpatialReference
1568:     Set pGraphicElement.SpatialReference = pSpatialReference
1569:     Set pElementProperties = pGroupElement
1570:     pElementProperties.Name = strName
    ' ADD GRAPHIC TO GRAPHICS CONTAINER
1572:     pGContainer.AddElement pGroupElement, 0
1573:   Else
1574:     pElement.Geometry = pNewGeometry
```

```
1575:     Set pGraphicElement = pElement
1576:     Set pSpatialReference = pGeometry.SpatialReference
1577:     Set pGraphicElement.SpatialReference = pSpatialReference
1578:     Set pElementProperties = pElement
1579:     pElementProperties.Name = strName

    ' ADD GRAPHIC TO GRAPHICS CONTAINER
1582:     pGContainer.AddElement pElement, 0
1583:   End If

  'Draw
1586:   pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Graphic_MakeFromGeometry " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Sub OpenDoc(theDocFilename As String, theDocPath As String)
  On Error GoTo ErrorHandler

    ' CHECK IF FILE EXISTS
    Dim booFileExists As Boolean
    Dim theCheckFilename As String
1599:     If Right(theDocPath, 1) = "\" Or Right(theDocPath, 1) = "/" Then
1600:       theCheckFilename = theDocPath & theDocFilename
1601:     Else
1602:       theCheckFilename = theDocPath & "\" & theDocFilename
1603:     End If

1605:     booFileExists = Dir(theCheckFilename) <> ""

1607:     If booFileExists Then
1608:       Call ShellExecute(0, vbNullString, theDocFilename, vbNullString, theDocPath, 1)
1609:     Else
1610:       MsgBox "Unable to find the following file:" & vbCrLf & vbCrLf & theCheckFilename & vbCrLf & _
           vbCrLf & "Bailing out...", , "Missing File:"
1612:     End If

  Exit Sub
ErrorHandler:
  HandleError True, "OpenDoc " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub

Public Function MakeColorRGB(pRed As Integer, pGreen As Integer, pBlue As Integer) As IColor
  On Error GoTo ErrorHandler
```

```
  Dim pColor As IRgbColor
1623:   Set pColor = New RgbColor
1624:   pColor.Red = pRed
1625:   pColor.Blue = pBlue
1626:   pColor.Green = pGreen
1627:   pColor.UseWindowsDithering = True

1629:   Set MakeColorRGB = pColor


  Exit Function
ErrorHandler:
  HandleError True, "MakeColorRGB " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Public Function MakeColorHSV(pHue As Integer, pSaturation As Integer, pValue As Integer) As IColor
  On Error GoTo ErrorHandler

  Dim pColor As IHsvColor
1640:   Set pColor = New HsvColor
1641:   pColor.Hue = pHue
1642:   pColor.Saturation = pSaturation
1643:   pColor.Value = pValue
1644:   pColor.UseWindowsDithering = True

1646:   Set MakeColorHSV = pColor


  Exit Function
ErrorHandler:
  HandleError True, "MakeColorHSV " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Public Sub GraphicsSetNameSelected(ByRef pMxDoc As IMxDocument, strName As String)
  On Error GoTo ErrorHandler

  Dim pGraphicsContainerSelect As IGraphicsContainerSelect

1658:   Set pGraphicsContainerSelect = pMxDoc.FocusMap
  Dim pEnumElement As IEnumElement
1660:   Set pEnumElement = pGraphicsContainerSelect.SelectedElements
1661:   pEnumElement.Reset

  Dim pElement As IElement
  Dim pElementProperties As IElementProperties
```

```
1666:   Set pElement = pEnumElement.Next

1668:   While Not pElement Is Nothing
1669:     Set pElementProperties = pElement
1670:     pElementProperties.Name = strName
1671:     Set pElement = pEnumElement.Next
1672:   Wend

  Exit Sub


  Exit Sub
ErrorHandler:
  HandleError True, "GraphicsSetNameSelected " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Function ReturnGraphicsByName(ByRef pMxDoc As IMxDocument, strName As String, _
      Optional AsElements As Boolean) As IArray
  On Error GoTo ErrorHandler


  Dim pGraphicsContainer As IGraphicsContainer

1689:   Set pGraphicsContainer = pMxDoc.FocusMap

1691:   pGraphicsContainer.Reset

  Dim pElement As IElement
  Dim pElementProperties As IElementProperties

1696:   Set pElement = pGraphicsContainer.Next

  Dim pArray As IArray
1699:   Set pArray = New esriSystem.Array
  Dim pGeometry As IGeometry
  Dim pClone As IClone

1703:   While Not pElement Is Nothing
1704:     Set pElementProperties = pElement
1705:     If pElementProperties.Name = strName Then
1706:       If AsElements Then
1707:         pArray.Add pElement
1708:       Else
1709:         Set pGeometry = pElement.Geometry
1710:         Set pClone = pGeometry
1711:         pArray.Add pClone.Clone     ' ONLY RETURN A COPY OF THE GEOMETRY; DON'T WANT TO MODIFY ACTUAL GRAPHIC HERE
```

```
1712:       End If
1713:     End If
1714:     Set pElement = pGraphicsContainer.Next

1716:   Wend
1717:   Set ReturnGraphicsByName = pArray

  Exit Function
ErrorHandler:
  HandleError True, "ReturnGraphicsByName " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function ReturnGraphicsByType(ByRef pMxDoc As IMxDocument, intGeometryType As esriGeometryType, _
      Optional AsElements As Boolean) As IArray
  On Error GoTo ErrorHandler


  Dim pGraphicsContainer As IGraphicsContainer

1731:   Set pGraphicsContainer = pMxDoc.FocusMap

1733:   pGraphicsContainer.Reset

  Dim pElement As IElement
  Dim pElementProperties As IElementProperties

1738:   Set pElement = pGraphicsContainer.Next

  Dim pArray As IArray
1741:   Set pArray = New esriSystem.Array
  Dim pGeometry As IGeometry
  Dim pClone As IClone

  Dim pGeometryType As esriGeometryType

1747:   While Not pElement Is Nothing
1748:     Set pElementProperties = pElement
1749:     Set pGeometry = pElement.Geometry
1750:     pGeometryType = pGeometry.GeometryType

1752:     If pGeometryType = intGeometryType Then
1753:       If AsElements Then            ' IN THIS CASE RETURN THE ACTUAL GRAPHIC ELEMENT
1754:         pArray.Add pElement
1755:       Else
1756:         Set pClone = pGeometry
1757:         pArray.Add pClone.Clone     ' ONLY RETURN A COPY OF THE GEOMETRY; DON'T WANT TO MODIFY ACTUAL GRAPHIC HERE
```

```
1758:       End If
1759:     End If
1760:     Set pElement = pGraphicsContainer.Next

1762:   Wend
1763:   Set ReturnGraphicsByType = pArray

  Exit Function
ErrorHandler:
  HandleError True, "ReturnGraphicsByType " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Sub DeleteGraphicsByName(ByRef pMxDoc As IMxDocument, strName As String)
  On Error GoTo ErrorHandler


  Dim pGraphicsContainer As IGraphicsContainer
  Dim pActiveView As IActiveView

1777:   Set pGraphicsContainer = pMxDoc.FocusMap
1778:   Set pActiveView = pMxDoc.ActiveView

1780:   pGraphicsContainer.Reset

  Dim pElement As IElement
  Dim pElementProperties As IElementProperties

1785:   Set pElement = pGraphicsContainer.Next

  Dim pEnvelope As IEnvelope

1789:   While Not pElement Is Nothing
1790:     Set pElementProperties = pElement

1792:     If pElementProperties.Name = strName Then
1793:       pGraphicsContainer.DeleteElement pElement
1794:       If (pEnvelope Is Nothing) Then
1795:         Set pEnvelope = pElement.Geometry.Envelope
1796:       Else
1797:         pEnvelope.Union pElement.Geometry.Envelope
1798:       End If
1799:     End If
1800:     Set pElement = pGraphicsContainer.Next

1802:   Wend
```

```
1804:   If (Not pEnvelope Is Nothing) Then
1805:     pActiveView.PartialRefresh esriViewGraphics + esriViewGraphicSelection + esriViewGeography, Nothing, pEnvelope
1806:   End If

  Exit Sub
ErrorHandler:
  HandleError True, "DeleteGraphicsByName " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Public Sub DoSpatialQuery(pSourceFeatureLayer As IFeatureLayer, pTargetFeatureLayer As IFeatureLayer, DoAll As Boolean, _
    pSelRelationship As esriSpatialRelEnum)
  On Error GoTo ErrorHandler


  Dim pFLayer As IFeatureLayer
  Dim pFc As IFeatureClass

  ' Specify the polygon layer with currently selected features
1823:   Set pFLayer = pSourceFeatureLayer

  Dim pFeatSel As IFeatureSelection
1826:   Set pFeatSel = pFLayer

  Dim pSelSet As ISelectionSet

1830:   If DoAll Then
1831:     pFeatSel.SelectFeatures Nothing, esriSelectionResultNew, False
1832:   End If

1834:   Set pSelSet = pFeatSel.SelectionSet

  Dim pEnumGeom As IEnumGeometry
  Dim pEnumGeomBind As IEnumGeometryBind

1839:   Set pEnumGeom = New EnumFeatureGeometry
1840:   Set pEnumGeomBind = pEnumGeom
1841:   pEnumGeomBind.BindGeometrySource Nothing, pSelSet

  Dim pGeomFactory As IGeometryFactory
1844:   Set pGeomFactory = New GeometryEnvironment

  Dim pGeom As IGeometry
1847:   Set pGeom = pGeomFactory.CreateGeometryFromEnumerator(pEnumGeom)

  Dim pSpFilter As ISpatialFilter
```

```
1850:   Set pSpFilter = New SpatialFilter
1851:   With pSpFilter
1852:     Set .Geometry = pGeom
1853:     .GeometryField = "SHAPE"
1854:     .SpatialRel = pSelRelationship
1855:   End With

1857:   Set pFLayer = pTargetFeatureLayer
1858:   Set pFeatSel = pFLayer

1860:   pFeatSel.SelectFeatures pSpFilter, esriSelectionResultNew, False


  Exit Sub
ErrorHandler:
  HandleError True, "DoSpatialQuery " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Public Function ReturnCurrentMapUnits(pMap As IMap) As String
  On Error GoTo ErrorHandler


  Dim intEsriUnits As Integer
1874:   intEsriUnits = pMap.MapUnits

  Select Case intEsriUnits
    Case 0
1878:       ReturnCurrentMapUnits = "Unknown"
    Case 1
1880:       ReturnCurrentMapUnits = "Inches"
    Case 2
1882:       ReturnCurrentMapUnits = "Points"
    Case 3
1884:       ReturnCurrentMapUnits = "Feet"
    Case 4
1886:       ReturnCurrentMapUnits = "Yards"
    Case 5
1888:       ReturnCurrentMapUnits = "Miles"
    Case 6
1890:       ReturnCurrentMapUnits = "Nautical Miles"
    Case 7
1892:       ReturnCurrentMapUnits = "Millimeters"
    Case 8
1894:       ReturnCurrentMapUnits = "Centimeters"
    Case 9
```

```
1896:       ReturnCurrentMapUnits = "Meters"
    Case 10
1898:       ReturnCurrentMapUnits = "Kilometers"
    Case 11
1900:       ReturnCurrentMapUnits = "Decimal Degrees"
    Case 12
1902:       ReturnCurrentMapUnits = "Decimeters"
1903:   End Select


  Exit Function
ErrorHandler:
  HandleError True, "ReturnCurrentMapUnits " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Sub CheckNumericReal(KeyAscii As Integer, txtTextBox As TextBox)
  On Error GoTo ErrorHandler


Select Case KeyAscii
  Case Is < 8
1918:     KeyAscii = 0
'  Case 8             ' BACKSPACE
  Case 9 To 43
1921:     KeyAscii = 0
  Case 45            ' "-" CHARACTER; INSERT AT BEGINNING OR REMOVE FROM BEGINNING
                     ' RESET CURSOR POSITION TO ORIGINAL CHARACTER LOCATION
1924:     KeyAscii = 0
    Dim strText As String
    Dim lngpos As Long
1927:     strText = txtTextBox.Text
1928:     lngpos = txtTextBox.SelStart
1929:     If Left(strText, 1) = "-" Then
1930:       txtTextBox.Text = Right(strText, Len(strText) - 1)
1931:       If lngpos > 0 Then
1932:         txtTextBox.SelStart = lngpos - 1
1933:       End If
1934:     Else
1935:       txtTextBox.Text = "-" & strText
1936:       txtTextBox.SelStart = lngpos + 1
1937:     End If
  Case 44 Or 46          ' DECIMAL CHARACTER; ONLY ALLOW ONE PERIOD OR COMMA
    Dim strText2 As String
1940:     strText2 = txtTextBox.Text
    Dim lngPos2 As Long
```

```
1942:     lngPos2 = txtTextBox.SelStart
1943:     If Not IsNumeric(Left(strText2, lngPos2) & Chr(KeyAscii) & Right(strText2, Len(strText2) - lngPos2)) Then
1944:       KeyAscii = 0
1945:     End If
  Case 47          ' "/" CHARACTER
1947:     KeyAscii = 0
  Case Is > 57
1949:     KeyAscii = 0
1950: End Select


  Exit Sub
ErrorHandler:
  HandleError True, "CheckNumericReal " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub


Public Sub CheckNumericRealPositive(KeyAscii As Integer, txtTextBox As TextBox)
  On Error GoTo ErrorHandler


Select Case KeyAscii
  Case Is < 8
1965:     KeyAscii = 0
'  Case 8            ' BACKSPACE
  Case 9 To 43
1968:     KeyAscii = 0
  Case 45            ' "-" CHARACTER
1970:     KeyAscii = 0

  Case 44 Or 46          ' DECIMAL CHARACTER; ONLY ALLOW ONE PERIOD OR COMMA
    Dim strText2 As String
1974:     strText2 = txtTextBox.Text
    Dim lngPos2 As Long
1976:     lngPos2 = txtTextBox.SelStart
1977:     If Not IsNumeric(Left(strText2, lngPos2) & Chr(KeyAscii) & Right(strText2, Len(strText2) - lngPos2)) Then
1978:       KeyAscii = 0
1979:     End If
  Case 47          ' "/" CHARACTER
1981:     KeyAscii = 0
  Case Is > 57
1983:     KeyAscii = 0
1984: End Select


  Exit Sub
```

```
ErrorHandler:
  HandleError True, "CheckNumericRealPositive " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Sub CheckNumericInteger(KeyAscii As Integer, txtTextBox As TextBox)
  On Error GoTo ErrorHandler


Select Case KeyAscii
  Case Is < 8
1997:     KeyAscii = 0
'  Case 8            ' BACKSPACE
  Case 9 To 43
2000:     KeyAscii = 0
  Case 45           ' "-" CHARACTER; INSERT AT BEGINNING OR REMOVE FROM BEGINNING
                    ' RESET CURSOR POSITION TO ORIGINAL CHARACTER LOCATION
2003:     KeyAscii = 0
    Dim strText As String
    Dim lngpos As Long
2006:     strText = txtTextBox.Text
2007:     lngpos = txtTextBox.SelStart
2008:     If Left(strText, 1) = "-" Then
2009:       txtTextBox.Text = Right(strText, Len(strText) - 1)
2010:       If lngpos > 0 Then
2011:         txtTextBox.SelStart = lngpos - 1
2012:       End If
2013:     Else
2014:       txtTextBox.Text = "-" & strText
2015:       txtTextBox.SelStart = lngpos + 1
2016:     End If
  Case 44, 46, 47         ' PREVENT PERIOD OR COMMA DECIMAL CHARACTER
2018:     KeyAscii = 0
  Case Is > 57
2020:     KeyAscii = 0
2021: End Select


  Exit Sub
ErrorHandler:
  HandleError True, "CheckNumericInteger " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub

Public Sub CheckNumericIntegerPositive(KeyAscii As Integer, txtTextBox As TextBox)
  On Error GoTo ErrorHandler
```

```
' ONLY ALLOW NUMBERS AND BACKSPACE
Select Case KeyAscii
  Case Is < 8
2036:     KeyAscii = 0
'  Case 8            ' BACKSPACE
  Case 9 To 47
2039:     KeyAscii = 0
  Case Is > 57
2041:     KeyAscii = 0
2042: End Select


  Exit Sub
ErrorHandler:
  HandleError True, "CheckNumericIntegerPositive " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Function CompareSpatialReferences(ByVal pSourceSR As ISpatialReference, ByVal pTargetSR As ISpatialReference) As Boolean
  On Error GoTo ErrorHandler


  Dim pSourceClone As IClone
  Dim pTargetClone As IClone
  Dim bSREqual As Boolean

2057:   Set pSourceClone = pSourceSR
2058:   Set pTargetClone = pTargetSR

  'Compare the coordinate system component of the spatial reference
2061:   bSREqual = pSourceClone.IsEqual(pTargetClone)

  'If the comparison failed, return false and exit
2064:   If Not bSREqual Then
2065:     CompareSpatialReferences = False
    Exit Function
2067:   End If

  'We can also compare the XY precision to ensure the spatial references are equal
  Dim pSourceSR2 As ISpatialReference2
  Dim bXYIsEqual As Boolean

2073:   Set pSourceSR2 = pSourceSR
2074:   bXYIsEqual = pSourceSR2.IsXYPrecisionEqual(pTargetSR)

  'If the comparison failed, return false and exit
2077:   If Not bXYIsEqual Then
2078:     CompareSpatialReferences = False
```

```
    Exit Function
2080:   End If

2082:   CompareSpatialReferences = True


  Exit Function
ErrorHandler:
  HandleError True, "CompareSpatialReferences " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function ReturnTimeElapsed(theTimeBegan As Date, theTimeEnd As Date) As String
  On Error GoTo ErrorHandler


  Dim theElapsedTime As Double
  Dim theNumDays As Double
  Dim theNumHours As Double
  Dim theNumMinutes As Double
  Dim theNumSeconds As Double

2100:   theElapsedTime = DateDiff("s", theTimeBegan, theTimeEnd)

2102:   theNumDays = Int(theElapsedTime / 86400)
2103:   theNumHours = Int((theElapsedTime Mod 86400) / 3600)
2104:   theNumMinutes = Int((theElapsedTime Mod 3600) / 60)
2105:   theNumSeconds = theElapsedTime Mod 60

  Dim theDayString As String
  Dim theHourString As String
  Dim theMinString As String
  Dim theSecString As String

2112:   If theNumDays = 1 Then
2113:     theDayString = " day"
2114:   Else
2115:     theDayString = " days"
2116:   End If

2118:   If theNumHours = 1 Then
2119:     theHourString = " hour"
2120:   Else
2121:     theHourString = " hours"
2122:   End If

2124:   If theNumMinutes = 1 Then
```

```
2125:     theMinString = " minute"
2126:   Else
2127:     theMinString = " minutes"
2128:   End If

2130:   If theNumSeconds = 1 Then
2131:     theSecString = " second..."
2132:   Else
2133:     theSecString = " seconds..."
2134:   End If

  Dim theElapsedTimeString As String
2137:   theElapsedTimeString = "Time Elapsed: "
2138:   If theNumDays > 0 Then
2139:     theElapsedTimeString = theElapsedTimeString & theNumDays & theDayString & ", " & theNumHours & theHourString & ", " & _
    theNumMinutes & theMinString & ", " & theNumSeconds & theSecString
2141:   ElseIf theNumHours > 0 Then
2142:     theElapsedTimeString = theElapsedTimeString & theNumHours & theHourString & ", " & _
    theNumMinutes & theMinString & ", " & theNumSeconds & theSecString
2144:   ElseIf theNumMinutes > 0 Then
2145:     theElapsedTimeString = theElapsedTimeString & _
    theNumMinutes & theMinString & ", " & theNumSeconds & theSecString
2147:   Else
2148:     theElapsedTimeString = theElapsedTimeString & theNumSeconds & theSecString
2149:   End If

2151:   ReturnTimeElapsed = "Analysis Began: " & Format(theTimeBegan, "long date") & ";  " & Format(theTimeBegan, "long time") &
vbCrLf & _
                      "Analysis Complete: " & Format(theTimeEnd, "long date") & ";  " & Format(theTimeEnd, "long time") & vbCrLf & _
                      theElapsedTimeString & vbCrLf & vbCrLf



  Exit Function
ErrorHandler:
  HandleError True, "ReturnTimeElapsed " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function ReturnTimeElapsedRTF(theTimeBegan As Date, theTimeEnd As Date, lngFontSize As Long) As String
  On Error GoTo ErrorHandler

  Dim theElapsedTime As Double
  Dim theNumDays As Double
  Dim theNumHours As Double
  Dim theNumMinutes As Double
```

```
  Dim theNumSeconds As Double

2172:   theElapsedTime = DateDiff("s", theTimeBegan, theTimeEnd)

2174:   theNumDays = Int(theElapsedTime / 86400)
2175:   theNumHours = Int((theElapsedTime Mod 86400) / 3600)
2176:   theNumMinutes = Int((theElapsedTime Mod 3600) / 60)
2177:   theNumSeconds = theElapsedTime Mod 60

  Dim theDayString As String
  Dim theHourString As String
  Dim theMinString As String
  Dim theSecString As String

2184:   If theNumDays = 1 Then
2185:     theDayString = " day"
2186:   Else
2187:     theDayString = " days"
2188:   End If

2190:   If theNumHours = 1 Then
2191:     theHourString = " hour"
2192:   Else
2193:     theHourString = " hours"
2194:   End If

2196:   If theNumMinutes = 1 Then
2197:     theMinString = " minute"
2198:   Else
2199:     theMinString = " minutes"
2200:   End If

2202:   If theNumSeconds = 1 Then
2203:     theSecString = " second..."
2204:   Else
2205:     theSecString = " seconds..."
2206:   End If

  Dim theElapsedTimeString As String
2209:   theElapsedTimeString = "Time Elapsed: "
2210:   If theNumDays > 0 Then
2211:     theElapsedTimeString = theElapsedTimeString & theNumDays & theDayString & ", " & theNumHours & theHourString & ", " & _
    theNumMinutes & theMinString & ", " & theNumSeconds & theSecString
2213:   ElseIf theNumHours > 0 Then
2214:     theElapsedTimeString = theElapsedTimeString & theNumHours & theHourString & ", " & _
    theNumMinutes & theMinString & ", " & theNumSeconds & theSecString
2216:   ElseIf theNumMinutes > 0 Then
```

```
2217:     theElapsedTimeString = theElapsedTimeString & _
    theNumMinutes & theMinString & ", " & theNumSeconds & theSecString
2219:   Else
2220:     theElapsedTimeString = theElapsedTimeString & theNumSeconds & theSecString
2221:   End If

2223:   ReturnTimeElapsedRTF = _
    "\b\fszzzFontSizezzz Analysis Began:\b0\i    zzzTimeBeganzzz\par" & vbCrLf & _
    "\b\i0 Analysis Complete:\b0\i    zzzTimeEndzzz\par" & vbCrLf & _
    "\b\i0 Time Elapsed:\b0\i    zzzTimeElapsedzzz\par" & vbCrLf

'  ReturnTimeElapsedRTF = "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0 Arial;}}" & vbCrLf & _
      "{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fszzzFontSizezzz Analysis Began:\b0\i    zzzTimeBeganzzz\par" &
vbCrLf & _
      "\b\i0 Analysis Complete:\b0\i    zzzTimeEndzzz\par" & vbCrLf & _
      "\b\i0 Time Elapsed:\b0\i    zzzTimeElapsedzzz\par" & vbCrLf & _
      "}"
  Dim strTimeBegan As String
2234:   strTimeBegan = CStr(Format(theTimeBegan, "long date")) & " at " & CStr(Format(theTimeBegan, "long time"))
  Dim strTimeEnd As String
2236:   strTimeEnd = CStr(Format(theTimeEnd, "long date")) & " at " & CStr(Format(theTimeEnd, "long time"))

2238:   ReturnTimeElapsedRTF = Replace(ReturnTimeElapsedRTF, "zzzFontSizezzz", CStr(lngFontSize * 2))
2239:   ReturnTimeElapsedRTF = Replace(ReturnTimeElapsedRTF, "zzzTimeBeganzzz", strTimeBegan)
2240:   ReturnTimeElapsedRTF = Replace(ReturnTimeElapsedRTF, "zzzTimeEndzzz", strTimeEnd)
2241:   ReturnTimeElapsedRTF = Replace(ReturnTimeElapsedRTF, "zzzTimeElapsedzzz", theElapsedTimeString)

'  ReturnTimeElapsedRTF = "Analysis Began: " & Format(theTimeBegan, "long date") & ";  " & Format(theTimeBegan, "long time") & vbCrLf
& _
                      "Analysis Complete: " & Format(theTimeEnd, "long date") & ";  " & Format(theTimeEnd, "long time") & vbCrLf & _
                      theElapsedTimeString & vbCrLf & vbCrLf

  Exit Function
ErrorHandler:
  HandleError True, "ReturnTimeElapsedRTF " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function


Public Function CheckCollectionForKey(colCollection As Collection, strKey As String) As Boolean

  On Error GoTo ErrorHandler
2256:   CheckCollectionForKey = True
  Dim varTest As Variant
2258:   varTest = colCollection.Item(strKey)

  Exit Function
```

```
ErrorHandler:
2262:   CheckCollectionForKey = False

End Function

Public Sub EnableSelectTool(pApp As IApplication)
  On Error GoTo ErrorHandler

    Dim pUID As New uID
    Dim pCmdItem As ICommandItem
    ' Use the GUID of the Select Elements command
2272:     pUID.Value = "{C22579D1-BC17-11D0-8667-0000F8751720}"
2273:     Set pCmdItem = pApp.Document.CommandBars.Find(pUID)
2274:     pCmdItem.Execute

  Exit Sub
ErrorHandler:
  HandleError True, "EnableSelectTool " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
```

Module 10: MyGeometricOperations

```
Attribute VB_Name = "MyGeometricOperations"
' MyGeometricOperations
'            AsDegrees - CONVERTS RADIANS TO DEGREES
'            AsRadians - CONVERTS DEGREES TO RADIANS
'          CalcBearing - GIVEN TWO POINTS, CALCULATES THE CARTESIAN BEARING, WHERE 0 = NORTH, 360 DEGREES GOING CLOCKWISE
'   CalcCheckClockwise - CHECKS IF 3 CONSECUTIVE POINTS ARE ARRANGED COUNTERCLOCKWISE
'    CalcClosestPoints - GIVEN TWO GEOMETRIES AND OPTIONAL NUMBER OF TIMES TO GO BACK AND FORTH BETWEEN CURVES, RETURNS
'                        IArray CONTAINING:
'                        IStringArray containing either "Intersecting Shapes" or
'                           "Empty Shapes" + two booleans indicating which geometry is empty.
'                             -- OR --
'                        3 OBJECTS:
'                        0) Connector Line AS IPOLYLINE
'                        1) Closest Point on Geometry #1         AS IPOINT
'                        2) Closest Point on Geometry #2         AS IPOINT
'       CalcDistMatrix - GIVEN AN IARRAY OF SHAPES, RETURNS A COLLECTION WHERE:
'                        INDEX = IArrayIndex1 & "_" & IArrayIndex1, and Object =
'                        IArray of {Distance, Optional Line, Optional Azimuth}
'        CalcPointLine - GIVEN POINT, DISTANCE, AZIMUTH, EMPTY ENDPOINT AND EMPTY POLYLINE, REPLACES EMPTY
'                        ENDPOINT WITH ACTUAL ENDPOINT AND OPTIONALLY RETURNS A POLYLINE CONNECTOR
'       CurveToPolygon - SIMILAR TO EllipticArcToPolygon2 EXCEPT THAT IT RETURNS AN IPolygon.  DOESN'T INSERT
'                        POINTS IF SEGMENT IS A LINE
'      CurveToPolyline - SIMILAR TO EllipticArcToPolygon2 EXCEPT THAT IT RETURNS AN IPolyline.  DOESN'T INSERT
'                        POINTS IF SEGMENT IS A LINE
```

```
' EllipticArcToPolygon - Given a segment collection and number of vertices, returns a polygon4 simulating the ellipse
'                        by generating points along the arc and then calculating a convex hull around the points.
'EllipticArcToPolygon2 - GIVEN A SEG COLLECTION AND NUMBER OF VERTICES, RETURNS A MULTIPOINT WITH APPROXIMATELY THE
'                        REQUESTED NUMBER OF POINTS DISTRIBUTED ALONG THE ARC.
'    EnvelopeToPolygon - GIVEN AN ENVELOPE, RETURNS A POLYGON
'Graphic_MakeFromGeometry - GIVEN A MAP DOCUMENT, GEOMETRY AND OPTIONAL NAME AND SYMBOLOGY, ADDS GRAPHIC TO MAP.
'Graphic_ReturnElementFromGeometry - GIVEN MAP DOC, GEOMETRY, OPTIONAL NAME AND OPTIONAL ADD-TO-VIEW, RETURNS THE GRAPHIC
'                                    ELEMENT
'             PointAdd - ADDS TWO POINTS
'        PointSubtract - SUBTRACTS POINT B FROM POINT A
'         ShowVertices - GIVEN A MAP DOC, GEOMETRY AND OPTIONAL NAME, ADDS POINT GRAPHICS TO SCREEN SHOWING WHERE
'                        VERTICES ARE
Option Explicit

Const dblPi As Double = 3.14159265358979
Const c_sModuleFileName As String = "D:\arcGIS_stuff\consultation\az_linkages\VB_Code\MyGeometricOperations.bas"


Public Function EnvelopeToPolygon(pEnv As IEnvelope) As IPolygon
  On Error GoTo ErrorHandler

  Dim pPtColl As IPointCollection

47:   Set pPtColl = New Polygon
48:   With pPtColl
49:       .AddPoint pEnv.LowerLeft
50:       .AddPoint pEnv.UpperLeft
51:       .AddPoint pEnv.UpperRight
52:       .AddPoint pEnv.LowerRight
      'Close the polygon
54:       .AddPoint pEnv.LowerLeft
55:   End With

  Dim pPolygon As IPolygon
58:   Set pPolygon = pPtColl
59:   Set pPolygon.SpatialReference = pEnv.SpatialReference
  Dim pTopoOp As ITopologicalOperator
61:   Set pTopoOp = pPolygon
62:   pTopoOp.Simplify

64:   Set EnvelopeToPolygon = pPtColl

  Exit Function
ErrorHandler:
  HandleError True, "EnvelopeToPolygon " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
```

```
Public Function EllipticArcToPolygon2(SegCollection As ISegmentCollection, NumVertices As Long) As IMultipoint
'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument

'  Dim pEllArc As IEllipticArc

On Error GoTo erh

  Dim pCurve As ICurve
  Dim pGeometry As IGeometry

  Dim anIndex As Long
  Dim lngSegCount As Long
85:   lngSegCount = SegCollection.SegmentCount - 1
  Dim theLength As Double
87:   theLength = 0
  Dim theTestLength As Double
  Dim lngLengths() As Long
  ReDim lngLengths(lngSegCount)
91:   For anIndex = 0 To lngSegCount
92:     theTestLength = SegCollection.Segment(anIndex).length
93:     theLength = theLength + theTestLength
94:     lngLengths(anIndex) = theTestLength
95:   Next anIndex

  Dim pProportion As Double
  Dim lngVertices() As Long
  Dim lngNumVertices As Long
  ReDim lngVertices(lngSegCount)
101:   For anIndex = 0 To lngSegCount
102:     lngNumVertices = Int((lngLengths(anIndex) / theLength) * NumVertices)
103:     If lngNumVertices < 8 Then lngNumVertices = 8
104:     lngVertices(anIndex) = lngNumVertices
105:   Next anIndex

  Dim pMpt As IPointCollection
108:   Set pMpt = New Multipoint
  Dim pPoint As IPoint
110:   Set pPoint = New Point
  Dim pClone As IClone

  Dim pRatio As Double
  Dim anIndex2 As Long

116:   For anIndex = 0 To lngSegCount
```

```
117:     lngNumVertices = lngVertices(anIndex)
118:     pRatio = 1 / lngNumVertices
119:     Set pCurve = SegCollection.Segment(anIndex)

121:     For anIndex2 = 0 To lngNumVertices
'      If pGeometry.GeometryType = esriGeometryEllipticArc Then
123:       pCurve.QueryPoint 0, (pRatio * anIndex2), True, pPoint
124:       Set pClone = pPoint

 '   Graphic_MakeFromGeometry pMxDoc, pPoint, "DeleteMe"

128:       pMpt.AddPoint pClone.Clone
129:     Next anIndex2
130:   Next anIndex

132:   Set EllipticArcToPolygon2 = pMpt
    Exit Function

erh:
136:     MsgBox "Failed in EllipticArcToPolygon2: " & Err.Description
End Function

Public Function CurveToPolygon(pOrigGeometry As IGeometry, NumVertices As Long) As IPolygon
On Error GoTo erh

  Dim pGeometryCollection As IGeometryCollection
143:   Set pGeometryCollection = pOrigGeometry
  Dim pSpRef As ISpatialReference
145:   Set pSpRef = pOrigGeometry.SpatialReference

  Dim pOrigPolygon As IPolycurve
148:   Set pOrigPolygon = pOrigGeometry

  Dim dblFullLength As Double
151:   dblFullLength = pOrigPolygon.length

  Dim pCurve As ICurve
  Dim pGeometry As IGeometry
  Dim pPolygon As IPointCollection
  Dim pRing As IRing
  Dim pSegment As ISegment
  Dim pStartPoint As IPoint
159:   Set pStartPoint = New Point
  Dim pEndPoint As IPoint
161:   Set pEndPoint = New Point
  Dim pClone As IClone
  Dim booFoundCurve As Boolean
```

```
  Dim lngRingCount As Long
  Dim lngNumVertices As Long
  Dim pRatio As Double
  Dim anIndex As Long
  Dim anIndex2 As Long
  Dim anIndex3 As Long
  Dim lngSegCount As Long

  Dim SegCollection As ISegmentCollection
  Dim pNewSegCollection As ISegmentCollection

  Dim pPathSegColl As ISegmentCollection
  Dim pNewSegment As ISegment
  Dim pNewLine As esriGeometry.ILine

  Dim pNewPolyGeoColl As IGeometryCollection
181:   Set pNewPolyGeoColl = New Polygon
  Dim pNewRingGeometry As IGeometry
  Dim pPath As IPath
  Dim pSegmentCollection As ISegmentCollection
  Dim pNewSegCol As ISegmentCollection

187:   lngRingCount = pGeometryCollection.GeometryCount - 1
188:   For anIndex = 0 To lngRingCount
189:     If TypeOf pOrigGeometry Is IPolyline Then
190:       Set pPath = pGeometryCollection.Geometry(anIndex)
191:       Set pSegmentCollection = pPath
192:       Set pNewSegCol = New Ring
193:       pNewSegCol.AddSegmentCollection pSegmentCollection
194:       Set pRing = pNewSegCol
195:       pRing.Close
196:     Else
197:       Set pRing = pGeometryCollection.Geometry(anIndex)
198:     End If
199:     Set SegCollection = pRing
200:     Set pNewSegCollection = New Ring
201:     lngSegCount = SegCollection.SegmentCount - 1
202:     For anIndex2 = 0 To lngSegCount
203:       Set pSegment = SegCollection.Segment(anIndex2)
204:       Set pGeometry = pSegment
205:       If pGeometry.GeometryType <> esriGeometryLine Then ' IF SEGMENT IS CURVE
206:         booFoundCurve = True
207:         lngNumVertices = Int((pSegment.length / dblFullLength) * NumVertices)
208:         If lngNumVertices < 8 Then lngNumVertices = 8
209:         pRatio = 1 / lngNumVertices
```

```
211:         Set pCurve = pSegment
212:         Set pPathSegColl = New Path
213:         Set pNewSegment = New esriGeometry.Line
214:         Set pStartPoint = pCurve.FromPoint
215:         For anIndex3 = 1 To lngNumVertices
216:           pCurve.QueryPoint 0, (pRatio * anIndex3), True, pEndPoint
217:           pNewSegment.FromPoint = pStartPoint
218:           pNewSegment.ToPoint = pEndPoint

220:           Set pClone = pNewSegment
221:           pPathSegColl.AddSegment pClone.Clone

'          If anIndex3 < 4 Then
'            MsgBox "Start Point:  X = " & CStr(pStartPoint.X) & ", Y = " & CStr(pStartPoint.Y) & vbCrLf & _
'              "End Point:  X = " & CStr(pEndPoint.X) & ", Y = " & CStr(pEndPoint.Y) & vbCrLf & _
'              "Segment Length = " & CStr(pNewSegment.length) & vbCrLf & _
'              "Segment Collection Count = " & CStr(pPathSegColl.SegmentCount)
'          End If

230:           Set pClone = pEndPoint
231:           Set pStartPoint = pClone.Clone

233:         Next anIndex3
234:         pNewSegCollection.AddSegmentCollection pPathSegColl

236:       Else      ' IF SEGMENT IS ACTUALLY LINE, DON'T ADD MIDPOINTS
237:         Set pClone = pSegment
238:         pNewSegCollection.AddSegment pClone.Clone
239:       End If
240:     Next anIndex2
241:     Set pNewRingGeometry = pNewSegCollection
242:     pNewPolyGeoColl.AddGeometry pNewRingGeometry

244:   Next anIndex

  Dim pNewPolygon As IPolygon

248:   If booFoundCurve Or (TypeOf pOrigGeometry Is IPolyline) Then

250:     Set pNewPolygon = pNewPolyGeoColl
    Dim pTopoOp As ITopologicalOperator
252:     Set pTopoOp = pNewPolygon
253:     pTopoOp.Simplify
254:     Set pNewPolygon.SpatialReference = pSpRef

256:   Else
257:     Set pNewPolygon = pOrigGeometry
```

```
258:     Set pNewPolygon.SpatialReference = pSpRef
259:   End If

261:   Set CurveToPolygon = pNewPolygon
  Exit Function

erh:
265:     MsgBox "Failed in CurveToPolygon: " & vbCrLf & "Error = " & Err.Description & vbCrLf & "Line Number = " & CStr(Erl)
End Function
Public Function CurveToPolyline(pOrigGeometry As IGeometry, NumVertices As Long) As IPolyline
On Error GoTo erh

  Dim pGeometryCollection As IGeometryCollection
271:   Set pGeometryCollection = pOrigGeometry
  Dim pSpRef As ISpatialReference
273:   Set pSpRef = pOrigGeometry.SpatialReference

  Dim pOrigPolyline As IPolycurve

277:   Set pOrigPolyline = pOrigGeometry

  Dim dblFullLength As Double
280:   dblFullLength = pOrigPolyline.length

  Dim pPath As IPath

  Dim pCurve As ICurve
  Dim pGeometry As IGeometry
  Dim pSegment As ISegment
  Dim pStartPoint As IPoint
288:   Set pStartPoint = New Point
  Dim pEndPoint As IPoint
290:   Set pEndPoint = New Point
  Dim pClone As IClone
  Dim booFoundCurve As Boolean

  Dim lngPathCount As Long
  Dim lngNumVertices As Long
  Dim pRatio As Double
  Dim anIndex As Long
  Dim anIndex2 As Long
  Dim anIndex3 As Long
  Dim lngSegCount As Long

  Dim SegCollection As ISegmentCollection
  Dim pNewSegCollection As ISegmentCollection
```

```
  Dim pPathSegColl As ISegmentCollection
  Dim pNewSegment As ISegment
  Dim pNewLine As esriGeometry.ILine

  Dim pNewPolyGeoColl As IGeometryCollection
310:   Set pNewPolyGeoColl = New Polyline
  Dim pNewPathGeometry As IGeometry

  Dim pRing As IRing

315:   lngPathCount = pGeometryCollection.GeometryCount - 1
316:   For anIndex = 0 To lngPathCount
317:     If TypeOf pOrigGeometry Is IPolygon Then
318:       Set pRing = pGeometryCollection.Geometry(anIndex)
319:       Set pPath = pRing
320:     Else
321:       Set pPath = pGeometryCollection.Geometry(anIndex)
322:     End If
323:     Set SegCollection = pPath
324:     Set pNewSegCollection = New Path
325:     lngSegCount = SegCollection.SegmentCount - 1
326:     For anIndex2 = 0 To lngSegCount
327:       Set pSegment = SegCollection.Segment(anIndex2)
328:       Set pGeometry = pSegment
329:       If pGeometry.GeometryType <> esriGeometryLine Then ' IF SEGMENT IS CURVE
330:         booFoundCurve = True
331:         lngNumVertices = Int((pSegment.length / dblFullLength) * NumVertices)
332:         If lngNumVertices < 8 Then lngNumVertices = 8
333:         pRatio = 1 / lngNumVertices

335:         Set pCurve = pSegment
336:         Set pPathSegColl = New Path
337:         Set pNewSegment = New esriGeometry.Line
338:         Set pStartPoint = pCurve.FromPoint
339:         For anIndex3 = 1 To lngNumVertices
340:           pCurve.QueryPoint 0, (pRatio * anIndex3), True, pEndPoint
341:           pNewSegment.FromPoint = pStartPoint
342:           pNewSegment.ToPoint = pEndPoint

344:           Set pClone = pNewSegment
345:           pPathSegColl.AddSegment pClone.Clone
346:           Set pClone = pEndPoint
347:           Set pStartPoint = pClone.Clone
348:         Next anIndex3
349:         pNewSegCollection.AddSegmentCollection pPathSegColl

351:       Else      ' IF SEGMENT IS ACTUALLY LINE, DON'T ADD MIDPOINTS
```

```
352:         Set pClone = pSegment
353:         pNewSegCollection.AddSegment pClone.Clone
354:       End If
355:     Next anIndex2
356:     Set pNewPathGeometry = pNewSegCollection
357:     pNewPolyGeoColl.AddGeometry pNewPathGeometry

359:   Next anIndex


  Dim pNewPolyline As IPolyline

364:   If booFoundCurve Or (TypeOf pOrigGeometry Is IPolygon) Then

366:     Set pNewPolyline = pNewPolyGeoColl
    Dim pTopoOp As ITopologicalOperator
368:     Set pTopoOp = pNewPolyline
369:     pTopoOp.Simplify
370:     Set pNewPolyline.SpatialReference = pSpRef

372:   Else
373:     Set pNewPolyline = pOrigGeometry
374:     Set pNewPolyline.SpatialReference = pSpRef
375:   End If

377:   Set CurveToPolyline = pNewPolyline
  Exit Function

erh:
381:     MsgBox "Failed in CurveToPolyline: " & vbCrLf & "Error = " & Err.Description & vbCrLf & "Line Number = " & CStr(Erl)
End Function
Public Sub Graphic_MakeFromGeometry(ByRef pMxDoc As IMxDocument, ByRef pGeometry As IGeometry, Optional strName As String)
  On Error GoTo ErrorHandler


  Dim pMxDocument As esriArcMapUI.IMxDocument
  Dim pActiveView As esriCarto.IActiveView

  Dim pGContainer As IGraphicsContainer
391:   Set pGContainer = pMxDoc.FocusMap

  Dim pElement As IElement
  Dim pPolygonElement As IPolygonElement
  Dim pSpatialReference As ISpatialReference
  Dim pGraphicElement As IGraphicElement
  Dim pElementProperties As IElementProperties
```

```
  Dim pClone As IClone
400:   Set pClone = pGeometry
  Dim pNewGeometry As IGeometry
402:   Set pNewGeometry = pClone.Clone

  Dim pGeometryType As esriGeometryType
405:   pGeometryType = pNewGeometry.GeometryType

  'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
  Select Case pGeometryType
    Case 0:
410:       MsgBox "Null Geometry!  No graphic added..."
    Case 1:
412:       Set pElement = New MarkerElement
    Case 3, 6, 13, 14, 15, 16:
414:       Set pElement = New LineElement
    Case 4, 11:
416:       Set pElement = New PolygonElement
    Case 5:
418:       Set pElement = New RectangleElement
    Case Else:
420:       MsgBox "Unexpected Shape Type:  Unable to add graphic..."
421:   End Select

423:   pElement.Geometry = pNewGeometry
424:   Set pGraphicElement = pElement
425:   Set pSpatialReference = pGeometry.SpatialReference
426:   Set pGraphicElement.SpatialReference = pSpatialReference
427:   Set pElementProperties = pElement
428:   pElementProperties.Name = strName

  ' ADD GRAPHIC TO GRAPHICS CONTAINER
431:   pGContainer.AddElement pElement, 0

  'Draw
434:   pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Graphic_MakeFromGeometry " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Function Graphic_ReturnElementFromGeometry(ByRef pMxDoc As IMxDocument, ByRef pGeometry As IGeometry, _
    Optional strName As String, Optional AddToView As Boolean) As IElement
  On Error GoTo ErrorHandler
```

```
  Dim pMxDocument As esriArcMapUI.IMxDocument
  Dim pActiveView As esriCarto.IActiveView

  Dim pGContainer As IGraphicsContainer
450:   Set pGContainer = pMxDoc.FocusMap

  Dim pElement As IElement
  Dim pPolygonElement As IPolygonElement
  Dim pSpatialReference As ISpatialReference
  Dim pGraphicElement As IGraphicElement
  Dim pElementProperties As IElementProperties

  Dim pClone As IClone
459:   Set pClone = pGeometry
  Dim pNewGeometry As IGeometry
461:   Set pNewGeometry = pClone.Clone

  Dim pGeometryType As esriGeometryType
464:   pGeometryType = pNewGeometry.GeometryType

  'ADD GEOMETRY, NAME AND SPATIAL REFERENCE TO GRAPHIC ELEMENT
  Select Case pGeometryType
    Case 0:
469:       MsgBox "Null Geometry!  No graphic added..."
    Case 1:
471:       Set pElement = New MarkerElement
    Case 3, 6, 13, 14, 15, 16:
473:       Set pElement = New LineElement
    Case 4, 11:
475:       Set pElement = New PolygonElement
    Case 5:
477:       Set pElement = New RectangleElement
    Case Else:
479:       MsgBox "Unexpected Shape Type:  Unable to add graphic..."
      Exit Function
481:   End Select

483:   pElement.Geometry = pNewGeometry
484:   Set pGraphicElement = pElement
485:   Set pSpatialReference = pGeometry.SpatialReference
486:   Set pGraphicElement.SpatialReference = pSpatialReference
487:   Set pElementProperties = pElement
488:   pElementProperties.Name = strName

490:   If AddToView Then
    ' ADD GRAPHIC TO GRAPHICS CONTAINER
```

```
492:     pGContainer.AddElement pElement, 0
    'Draw
494:     pMxDoc.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing
495:   End If

497:   Set Graphic_ReturnElementFromGeometry = pElement


  Exit Function
ErrorHandler:
  HandleError True, "Graphic_ReturnElementFromGeometry " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number,
Err.Source, Err.Description, 4
End Function

Public Sub ShowVertices(pMxDoc As IMxDocument, pGeometry As IGeometry, Optional strName As String, _
        Optional DeleteCurrentGraphicsWithName As Boolean)
  On Error GoTo ErrorHandler


  Dim pPoint1 As IPoint
  Dim pPoint2 As IPoint
  Dim pPoly As IPolygon
  Dim pLine As IPolyline
  Dim pOutVertex As IPoint, lOutPart As Long, lOutVertex As Long

516:   If DeleteCurrentGraphicsWithName And (strName <> "") Then
517:     Call DeleteGraphicsByName(pMxDoc, "DeleteMe")
518:   End If

  Dim pArray As IArray
521:   Set pArray = New esriSystem.Array

  Dim pPointCollection As IPointCollection
524:   Set pPointCollection = pGeometry

  Dim pPointEnum As IEnumVertex
527:   Set pPointEnum = pPointCollection.EnumVertices

529:   pPointEnum.Reset

  Dim pVertex As IPoint
532:   Set pVertex = New Point
  'Query the next vertex - have to cocreate the point
  'QueryNext is faster than Next, because the method doesn't have
  'to create the point each time
536:   pPointEnum.QueryNext pVertex, lOutPart, lOutVertex
```

```
538:   Do While Not pVertex.IsEmpty
539:     Graphic_MakeFromGeometry pMxDoc, pVertex, strName
540:     pPointEnum.QueryNext pVertex, lOutPart, lOutVertex
'    Debug.Print lOutPart & ",    " & lOutVertex & ",  " & pVertex.IsEmpty
542:   Loop


  Exit Sub
ErrorHandler:
  HandleError True, "ShowVertices " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Function CalcBearing(ByRef Point1 As IPoint, ByRef Point2 As IPoint) As Double
  On Error GoTo ErrorHandler


  Dim dblBearing As Double

  Dim xDist As Double
  Dim yDist As Double
  Dim xyTanDeg As Double

559:   xDist = (Point1.X - Point2.X)
560:   yDist = (Point1.Y - Point2.Y)
561:   If yDist = 0 Then
562:     If xDist < 0 Then
563:       xyTanDeg = -90
564:     ElseIf xDist = 0 Then
565:       xyTanDeg = 0
566:     Else
567:       xyTanDeg = 90
568:     End If
569:   Else
570:     xyTanDeg = AsDegrees(Atn(xDist / yDist))
571:   End If

573:   If (yDist >= 0) Then
574:     dblBearing = 180 + xyTanDeg
575:   Else
576:     If (xDist <= 0) Then
577:       dblBearing = xyTanDeg
578:     Else
579:       dblBearing = 360 + xyTanDeg
580:     End If
581:   End If ' END CALCULATING BEARING

583:   dblBearing = Abs(dblBearing)
```

```
584:   CalcBearing = dblBearing


  Exit Function
ErrorHandler:
  HandleError True, "CalcBearing " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
Public Function CalcDistMatrix(pArray As esriSystem.IArray, Optional IncludeLine As Boolean, _
    Optional IncludeBearing As Boolean, Optional pApp As IApplication) As Collection
  On Error GoTo ErrorHandler

595:   Screen.MousePointer = vbHourglass

  ' RETURNS A COLLECTION OF IVariantArray OBJECTS
  ' EACH IVariantArray IDENTIFIED BY STRING; CONCATENATION OF [ORIGIN INDEX] & "_" & [DESTINATION INDEX]
  ' EACH IVariantArray OBJECT CONTAINS:
  '       0) ORIGIN SHAPE INDEX VALUE
  '       1) DESTINATION SHAPE INDEX VALUE
  '       2) DISTANCE
  '       3) CONNECTOR POLYLINE:  OPTIONAL; CONTAINS BOOLEAN "FALSE" IF NOT REQUESTED
  '       4) BEARING:             OPTIONAL; CONTAINS BOOLEAN "FALSE" IF NOT REQUESTED

  Dim pCollection As Collection
607:   Set pCollection = New Collection

  Dim pProxOp As IProximityOperator

  Dim lngIndex As Long
  Dim lngIndex2 As Long

  Dim lngArrayMaxIndex As Long
615:   lngArrayMaxIndex = pArray.Count - 1

  Dim pGeometry1 As IGeometry
  Dim pGeometry2 As IGeometry

  Dim dblDistance As Double
  Dim dblBearing As Double
  Dim dblRevBearing As Double
  Dim strID As String
  Dim strRevID As String
  Dim pConnector As IPolyline
  Dim pRevConnector As IPolyline

  Dim pVarArray As IVariantArray
  Dim pRevVarArray As IVariantArray
```

```
631:   If Not pApp Is Nothing Then
      ' PROGRESS BAR STUFF
    Dim psbar As IStatusBar
634:     Set psbar = pApp.StatusBar
    Dim pPro As IStepProgressor
636:     Set pPro = psbar.ProgressBar
    Dim lngCounter As Long
638:     lngCounter = 0
    Dim lngTotalCount As Long
640:     lngTotalCount = (((lngArrayMaxIndex + 1) * lngArrayMaxIndex) / 2)
    Dim strTotalCount As String
642:     strTotalCount = CStr(lngTotalCount)
643:     pPro.position = 1
644:     psbar.ShowProgressBar "Building Preliminary Distance Matrix:  Step 1 of " & strTotalCount & "...", 1, _
           lngTotalCount, 1, True
646:   End If

  Dim pOutputPointCollection As IPointCollection

650:   For lngIndex = 0 To lngArrayMaxIndex
651:     Set pGeometry1 = pArray.Element(lngIndex)

653:     For lngIndex2 = lngIndex To lngArrayMaxIndex

655:       strID = CStr(lngIndex) & "_" & CStr(lngIndex2)
656:       strRevID = CStr(lngIndex2) & "_" & CStr(lngIndex)

658:       Set pVarArray = New VarArray
659:       Set pRevVarArray = New VarArray

      ' FIRST ELEMENT
662:       pVarArray.Add lngIndex         ' FIRST VALUES IN THE ARRAY ARE ORIGIN NODES
663:       pRevVarArray.Add lngIndex2

      ' SECOND ELEMENT
666:       pVarArray.Add lngIndex2        ' SECOND VALUES IN THE ARRAY ARE "TO" NODES
667:       pRevVarArray.Add lngIndex

669:       If lngIndex = lngIndex2 Then   ' IF MEASURING DISTANCE TO ITSELF
670:         Set pConnector = New Polyline
671:         pConnector.SetEmpty
672:         Set pRevConnector = New Polyline
673:         pRevConnector.SetEmpty

        ' THIRD ELEMENT
676:         pVarArray.Add 0              ' THIRD VALUE IS DISTANCE
```

```
        ' FOURTH ELEMENT
679:          If IncludeLine Then
680:            pVarArray.Add pConnector   ' FOURTH VALUE IS CONNECTION LINE
681:          Else
682:            pVarArray.Add False        ' FOURTH VALUE:  JUST ADDING SMALL PLACEHOLDER ELEMENT
683:          End If

        ' FIFTH ELEMENT
686:          If IncludeBearing Then
687:            pVarArray.Add -999         ' FIFTH VALUE IS BEARING
688:          Else
689:            pVarArray.Add False
690:          End If

        ' ADD VARARRAY TO ORIGINAL COLLECTION
693:          pCollection.Add pVarArray, strID

695:        Else

697:          If Not pApp Is Nothing Then
698:            lngCounter = lngCounter + 1
699:            pPro.Message = "Building Preliminary Distance Matrix:  Step " & CStr(lngCounter) & " of " & strTotalCount & "..."
700:            psbar.StepProgressBar
701:          End If

703:          Set pGeometry2 = pArray.Element(lngIndex2)

705:          If IncludeLine Or IncludeBearing Then
          Dim pLineArray As IArray
707:            Set pLineArray = CalcClosestPoints(pGeometry1, pGeometry2, 10)

709:            If TypeOf pLineArray.Element(0) Is esriSystem.IStringArray Then      ' FUNCTION FAILED FOR SOME REASON
            Dim pStrArray As IStringArray
711:              Set pStrArray = pLineArray.Element(0)
712:              MsgBox "Failed to connect:" & vbCrLf & "Message = " & pStrArray.Element(0) & vbCrLf & _
                    "Index 1 = " & CStr(lngIndex) & " of " & CStr(lngArrayMaxIndex) & vbCrLf & _
                    "Index 2 = " & CStr(lngIndex2) & " of " & CStr(lngArrayMaxIndex)
715:              Set pConnector = New Polyline
716:              pConnector.SetEmpty
717:              Set pRevConnector = New Polyline
718:              pRevConnector.SetEmpty

            ' THIRD ELEMENT
721:              pVarArray.Add 0              ' THIRD VALUE IS DISTANCE

            ' FOURTH ELEMENT
```

```
724:             If IncludeLine Then
725:               pVarArray.Add pConnector   ' FOURTH VALUE IS CONNECTION LINE
726:             Else
727:               pVarArray.Add False        ' FOURTH VALUE:  JUST ADDING SMALL PLACEHOLDER ELEMENT
728:             End If

           ' FIFTH ELEMENT
731:             If IncludeBearing Then
732:               pVarArray.Add -999         ' FIFTH VALUE IS BEARING
733:             Else
734:               pVarArray.Add False
735:             End If
736:          Else

738:             If IncludeLine Then
739:               Set pConnector = pLineArray.Element(0)
740:               Set pRevConnector = New Polyline
741:               Set pOutputPointCollection = pRevConnector
742:               pOutputPointCollection.AddPoint pLineArray.Element(2)
743:               pOutputPointCollection.AddPoint pLineArray.Element(1)

            ' THIRD ELEMENT
746:               pVarArray.Add pConnector.length
747:               pRevVarArray.Add pConnector.length

            ' FOURTH ELEMENT
750:               pVarArray.Add pConnector
751:               pRevVarArray.Add pRevConnector
752:             Else
            ' FOURTH ELEMENT
754:               pVarArray.Add False        ' JUST ADDING SMALL PLACEHOLDER ELEMENT
755:               pRevVarArray.Add False     ' JUST ADDING SMALL PLACEHOLDER ELEMENT
756:             End If
757:             If IncludeBearing Then
758:               dblBearing = CalcBearing(pLineArray.Element(1), pLineArray.Element(2))
759:               If dblBearing < 180 Then
760:                 dblRevBearing = dblBearing + 180
761:               Else
762:                 dblRevBearing = dblBearing - 180
763:               End If
            ' FIFTH ELEMENT
765:               pVarArray.Add dblBearing
766:               pRevVarArray.Add dblRevBearing
767:             Else
            ' FIFTH ELEMENT
769:               pVarArray.Add False
770:               pRevVarArray.Add False
```

```
771:             End If
772:           End If
773:         Else
         ' THIRD ELEMENT
775:           Set pProxOp = pGeometry1
776:           dblDistance = pProxOp.ReturnDistance(pGeometry2)
777:           pVarArray.Add dblDistance
778:           pRevVarArray.Add dblDistance
         ' FOURTH ELEMENT   (DISTANCE)
780:           pVarArray.Add False
781:           pRevVarArray.Add False
         ' FIFTH ELEMENT   (BEARING)
783:           pVarArray.Add False
784:           pRevVarArray.Add False
785:         End If

787:         pCollection.Add pVarArray, strID
788:         pCollection.Add pRevVarArray, strRevID

790:       End If
791:     Next lngIndex2
792:   Next lngIndex

794:   Set CalcDistMatrix = pCollection

796:   Screen.MousePointer = vbDefault
797:   If Not pApp Is Nothing Then
798:     pPro.position = 1
799:     psbar.HideProgressBar
800:   End If

  Exit Function
ErrorHandler:
  HandleError True, "CalcDistMatrix " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
Public Function CalcClosestPoints(ByVal Shape1 As IGeometry, ByVal shape2 As IGeometry, Optional intMaxCurveRepeat As Integer) As
IArray
  On Error GoTo ErrorHandler

' CalcClosestPoints
' Jenness Enterprises (www.jennessent.com)
' Given two shapes, this script returns an IARRAY object containing the line connecting the closest points on each shape, plus the
connection points
' CURRENTLY DOES NOT GUARANTEE SUCCESS WITH TRUE CURVES BECAUSE VERTICES ARE NOT GOOD QUERY POINTS; ATTEMPTS SEVERAL RUNS BACK AND
FORTH
```

```
' Dim pRelationalOperator As IRelationalOperator
Dim pGeometryType1 As esriGeometryType
Dim pGeometryType2 As esriGeometryType
Dim pGeometry1 As IGeometry
Dim pGeometry2 As IGeometry

820: Set pGeometry1 = Shape1
821: Set pGeometry2 = shape2

823: pGeometryType2 = shape2.GeometryType

' IF SHAPE #2 HAPPENS TO BE POINT, SET THAT ONE FIRST
Dim ShouldReverse As Boolean
827: ShouldReverse = False
828: If pGeometryType2 = esriGeometryPoint Then
829:   Set pGeometry1 = shape2
830:   Set pGeometry2 = Shape1
831:   pGeometryType2 = pGeometry2.GeometryType
832:   ShouldReverse = True
833: End If

835: pGeometryType1 = Shape1.GeometryType

Dim pArray As IArray
838: Set pArray = New esriSystem.Array
Dim pOutputLine As IPolyline
840: Set pOutputLine = New Polyline
Dim pOutputPointCollection As IPointCollection
842: Set pOutputPointCollection = pOutputLine

Dim pStartPoint As IPoint
Dim pEndPoint As IPoint

Dim pPoint1 As IPoint
Dim pPoint2 As IPoint
849: Set pPoint2 = New Point

Dim pProximityOp As IProximityOperator
Dim pStringArray As IStringArray

' CHECK FOR NULL SHAPES
' CHECK FOR INTERSECTING SHAPES
' NOT SURE IF THIS WILL WORK WITH MULTIPOINTS

858: If pGeometry1.IsEmpty Or pGeometry2.IsEmpty Then
859:   Set pStringArray = New strArray
860:   pStringArray.Add "Empty Shapes"
```

```
861:   pStringArray.Add CStr(pGeometry1.IsEmpty)
862:   pStringArray.Add CStr(pGeometry2.IsEmpty)
863:   pArray.Add pStringArray
864:   Set CalcClosestPoints = pArray
  Exit Function
866: Else
867:   Set pProximityOp = pGeometry1
868:   If pProximityOp.ReturnDistance(pGeometry2) = 0 Then
869:     Set pStringArray = New strArray
870:     pStringArray.Add "Intersecting Shapes"
871:     pArray.Add pStringArray
872:     Set CalcClosestPoints = pArray
    Exit Function
874:   End If
875: End If

877: If pGeometryType1 = esriGeometryPoint Then
878:   Set pPoint1 = pGeometry1

880:   If pGeometryType2 = esriGeometryPoint Then
881:     Set pPoint2 = pGeometry2

883:     If pPoint1.X = pPoint2.X And pPoint1.Y = pPoint2.Y Then
884:       If ShouldReverse Then
885:         pArray.Add pOutputLine
886:         pArray.Add pPoint2
887:         pArray.Add pPoint1
888:       Else
889:         pArray.Add pOutputLine
890:         pArray.Add pPoint1
891:         pArray.Add pPoint2
892:       End If
893:     Else
894:       If ShouldReverse Then
895:         pOutputPointCollection.AddPoint pPoint2
896:         pOutputPointCollection.AddPoint pPoint1
897:         pArray.Add pOutputLine
898:         pArray.Add pPoint2
899:         pArray.Add pPoint1
900:       Else
901:         pOutputPointCollection.AddPoint pPoint1
902:         pOutputPointCollection.AddPoint pPoint2
903:         pArray.Add pOutputLine
904:         pArray.Add pPoint1
905:         pArray.Add pPoint2
906:       End If
907:     End If
```

```
908:   Else

910:     Set pProximityOp = pGeometry2

912:     pProximityOp.QueryNearestPoint pPoint1, esriNoExtension, pPoint2

914:     If ShouldReverse Then
915:       pOutputPointCollection.AddPoint pPoint2
916:       pOutputPointCollection.AddPoint pPoint1
917:       pArray.Add pOutputLine
918:       pArray.Add pPoint2
919:       pArray.Add pPoint1
920:     Else
921:       pOutputPointCollection.AddPoint pPoint1
922:       pOutputPointCollection.AddPoint pPoint2
923:       pArray.Add pOutputLine
924:       pArray.Add pPoint1
925:       pArray.Add pPoint2
926:     End If


929:   End If
930: Else
  Dim dblTestDistance As Double
  Dim pEnvelope As IEnvelope
  Dim pEnvelope2 As IEnvelope
934:   Set pEnvelope = pGeometry1.Envelope
935:   Set pEnvelope2 = pGeometry2.Envelope
936:   pEnvelope.Union pEnvelope2
937:   dblTestDistance = (pEnvelope.Height * pEnvelope.Width)
  Dim dblMaxDistance As Double
939:   dblMaxDistance = dblTestDistance

  Dim pPointCollection1 As IPointCollection
  Dim pPointCollection2 As IPointCollection

944:   If pGeometry1.GeometryType = esriGeometryEnvelope Then
    Dim pTempEnv As IEnvelope
946:     Set pTempEnv = pGeometry1
    Dim pTempPoly1 As IPolygon
    Dim pTempPoint1 As IPoint
949:     Set pTempPoly1 = New Polygon
950:     Set pPointCollection1 = pTempPoly1
    Dim dXmin1 As Double
    Dim dYmin1 As Double
    Dim dXmax1 As Double
    Dim dYmax1 As Double
```

```
955:     pTempEnv.QueryCoords dXmin1, dYmin1, dXmax1, dYmax1
956:     Set pTempPoint1 = New Point
957:     pTempPoint1.X = dXmin1
958:     pTempPoint1.Y = dYmin1
959:     pPointCollection1.AddPoint pTempPoint1

961:     Set pTempPoint1 = New Point
962:     pTempPoint1.X = dXmin1
963:     pTempPoint1.Y = dYmax1
964:     pPointCollection1.AddPoint pTempPoint1

966:     Set pTempPoint1 = New Point
967:     pTempPoint1.X = dXmax1
968:     pTempPoint1.Y = dYmax1
969:     pPointCollection1.AddPoint pTempPoint1

971:     Set pTempPoint1 = New Point
972:     pTempPoint1.X = dXmax1
973:     pTempPoint1.Y = dYmin1
974:     pPointCollection1.AddPoint pTempPoint1
975:   Else
976:     Set pPointCollection1 = pGeometry1
977:   End If

979:   If pGeometry2.GeometryType = esriGeometryEnvelope Then
    Dim pTempEnv2 As IEnvelope
981:     Set pTempEnv2 = pGeometry2
    Dim pTempPoly2 As IPolygon
    Dim pTempPoint2 As IPoint
984:     Set pTempPoly2 = New Polygon
985:     Set pPointCollection2 = pTempPoly2
    Dim dXmin2 As Double
    Dim dYmin2 As Double
    Dim dXmax2 As Double
    Dim dYmax2 As Double
990:     pTempEnv2.QueryCoords dXmin2, dYmin2, dXmax2, dYmax2
991:     Set pTempPoint2 = New Point
992:     pTempPoint2.X = dXmin2
993:     pTempPoint2.Y = dYmin2
994:     pPointCollection2.AddPoint pTempPoint2

996:     Set pTempPoint2 = New Point
997:     pTempPoint2.X = dXmin2
998:     pTempPoint2.Y = dYmax2
999:     pPointCollection2.AddPoint pTempPoint2

1001:     Set pTempPoint2 = New Point
```

```
1002:     pTempPoint2.X = dXmax2
1003:     pTempPoint2.Y = dYmax2
1004:     pPointCollection2.AddPoint pTempPoint2

1006:     Set pTempPoint2 = New Point
1007:     pTempPoint2.X = dXmax2
1008:     pTempPoint2.Y = dYmin2
1009:     pPointCollection2.AddPoint pTempPoint2
1010:   Else
1011:     Set pPointCollection2 = pGeometry2
1012:   End If

  Dim pClone As IClone

  Dim pVertex As IPoint
1017:   Set pVertex = New Point

  Dim pPointEnum As IEnumVertex
  Dim lngOutPart As Long
  Dim lngOutVertex As Long

1023:   Set pPointEnum = pPointCollection1.EnumVertices
1024:   pPointEnum.Reset
1025:   pPointEnum.QueryNext pVertex, lngOutPart, lngOutVertex

  ' CHECK IF CURVES; THIS CODE JUST CHECKS FIRST SEGMENT FOR CURVATURE
  Dim booWorkingWithCurves As Boolean
  Dim pSegmentCollection1 As ISegmentCollection
1030:   Set pSegmentCollection1 = pGeometry1
  Dim pSegment1 As ISegment
1032:   Set pSegment1 = pSegmentCollection1.Segment(0)
  Dim pGeometryTypeA As esriGeometryType
1034:   pGeometryTypeA = pSegment1.GeometryType

  Dim pSegmentCollection2 As ISegmentCollection
1037:   Set pSegmentCollection2 = pGeometry2
  Dim pSegment2 As ISegment
1039:   Set pSegment2 = pSegmentCollection2.Segment(0)
  Dim pGeometryTypeB As esriGeometryType
1041:   pGeometryTypeB = pSegment2.GeometryType

1043:   booWorkingWithCurves = (pGeometryTypeA = esriGeometryBezier3Curve) Or _
    (pGeometryTypeA = esriGeometryCircularArc) Or _
    (pGeometryTypeA = esriGeometryEllipticArc) Or _
    (pGeometryTypeB = esriGeometryBezier3Curve) Or _
    (pGeometryTypeB = esriGeometryCircularArc) Or _
    (pGeometryTypeB = esriGeometryEllipticArc)
```

```
1050:   Do While Not pVertex.IsEmpty
1051:     Set pProximityOp = pGeometry2
1052:     dblTestDistance = pProximityOp.ReturnDistance(pVertex)
1053:     If dblTestDistance < dblMaxDistance Then
1054:       dblMaxDistance = dblTestDistance
1055:       Set pClone = pVertex
1056:       Set pPoint1 = pClone.Clone
1057:       pProximityOp.QueryNearestPoint pVertex, esriNoExtension, pPoint2
1058:     End If
1059:     pPointEnum.QueryNext pVertex, lngOutPart, lngOutVertex
1060:   Loop

1062:   Set pPointEnum = pPointCollection2.EnumVertices
1063:   pPointEnum.Reset
1064:   pPointEnum.QueryNext pVertex, lngOutPart, lngOutVertex

1066:   Do While Not pVertex.IsEmpty
1067:     Set pProximityOp = pGeometry1
1068:     dblTestDistance = pProximityOp.ReturnDistance(pVertex)
1069:     If dblTestDistance < dblMaxDistance Then
1070:       dblMaxDistance = dblTestDistance
1071:       Set pClone = pVertex
1072:       Set pPoint2 = pClone.Clone
1073:       pProximityOp.QueryNearestPoint pVertex, esriNoExtension, pPoint1
1074:     End If
1075:     pPointEnum.QueryNext pVertex, lngOutPart, lngOutVertex
1076:   Loop

  ' FOR DEBUGGING
'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument
  ' IF WORKING WITH CURVES, GO BACK AND FORTH A FEW TIMES
1082:   If booWorkingWithCurves Then
    Dim intRepeat As Integer
    Dim pPoint1Temp As IPoint, pPoint2Temp As IPoint

1086:     Do Until (intRepeat = intMaxCurveRepeat)
 '     Graphic_MakeFromGeometry pMxDoc, pPoint1, "DeleteMe"
 '     Graphic_MakeFromGeometry pMxDoc, pPoint2, "DeleteMe"

1090:       Set pProximityOp = pGeometry2
1091:       pProximityOp.QueryNearestPoint pPoint1, esriNoExtension, pPoint2

1093:       Set pProximityOp = pGeometry1
1094:       pProximityOp.QueryNearestPoint pPoint2, esriNoExtension, pPoint1
```

```
1096:       intRepeat = intRepeat + 1

1098:     Loop

1100:   End If

1102:   If ShouldReverse Then
1103:     pOutputPointCollection.AddPoint pPoint2
1104:     pOutputPointCollection.AddPoint pPoint1
1105:     pArray.Add pOutputLine
1106:     pArray.Add pPoint2
1107:     pArray.Add pPoint1
1108:   Else
1109:     pOutputPointCollection.AddPoint pPoint1
1110:     pOutputPointCollection.AddPoint pPoint2
1111:     pArray.Add pOutputLine
1112:     pArray.Add pPoint1
1113:     pArray.Add pPoint2
1114:   End If

1116: End If

1118: Set CalcClosestPoints = pArray


  Exit Function
ErrorHandler:
  HandleError True, "CalcClosestPoints " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function CalcCheckClockwise(theP As IPoint, theQ As IPoint, theR As IPoint) As Boolean
  On Error GoTo ErrorHandler


' CalcCheckClockwise
' Jenness Enterprises <www.jennessent.com)>
' Given 3 consecutive points, this scripts calculates whether the third point lies to the right
' (clockwise) or to the left (counter-clockwise) of the line connecting the first point to
' the second point.

' CLOCKWISE IF TRUE
1137: CalcCheckClockwise = ((theQ.X * (theR.Y - theP.Y)) + (theQ.Y * (theP.X - theR.X)) - ((theP.X) * (theR.Y)) _
      + ((theP.Y) * (theR.X)) < 0)


  Exit Function
```

```
ErrorHandler:
  HandleError True, "CalcCheckClockwise " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function PointAdd(pPointA As IPoint, pPointB As IPoint) As IPoint
  On Error GoTo ErrorHandler


1150:   Set PointAdd = New Point
1151:   PointAdd.PutCoords pPointA.X + pPointB.X, pPointA.Y + pPointB.Y


  Exit Function
ErrorHandler:
  HandleError True, "PointAdd " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Public Function PointSubtract(pPointA As IPoint, pPointB As IPoint) As IPoint
  On Error GoTo ErrorHandler


1163:   Set PointSubtract = New Point
1164:   PointSubtract.PutCoords pPointA.X - pPointB.X, pPointA.Y - pPointB.Y


  Exit Function
ErrorHandler:
  HandleError True, "PointSubtract " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Public Function AsRadians(theDegrees As Double) As Double
  On Error GoTo ErrorHandler


1176:   AsRadians = dblPi * (theDegrees / 180)


  Exit Function
ErrorHandler:
  HandleError True, "AsRadians " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Public Function AsDegrees(theRadians As Double) As Double
  On Error GoTo ErrorHandler
```

```
1188:   AsDegrees = (theRadians * 180) / dblPi


  Exit Function
ErrorHandler:
  HandleError True, "AsDegrees " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function

Public Sub CalcPointLine(ptOrigin As IPoint, theLength As Double, dblAzimuth As Double, ptEndPoint As IPoint, _
    Optional pLine As IPolyline)
  On Error GoTo ErrorHandler


' Jenness Enterprises <www.jennessent.com>
' Given an origin point, distance and bearing, this script will return a new point at that distance and bearing, and a line
' connecting that new point to the origin point

'' MAKE SURE AZIMUTH IS BETWEEN 0 AND 360
Dim theAzimuth As Double
1207: theAzimuth = dblAzimuth

1209: Set ptEndPoint = New Point

1211: Do While theAzimuth < 0
1212:   theAzimuth = theAzimuth + 360
1213: Loop
1214: theAzimuth = theAzimuth Mod 360
'
'' NEW SEGMENT AND POINT DISTANCE NORTH/SOUTH AND EAST/WEST BASED ON DISTANCE AND BEARING FROM ORIGIN.
'' THERE ARE EIGHT DIFFERENT POSSIBILITIES:  THE BEARING COULD BE ONE OF THE FOUR CARDINAL DIRECTIONS OR IT
'' COULD BE IN ONE OF THE FOUR QUADRANTS.  THE BEARING IS TREATED DIFFERENTLY IN EACH OF THESE CIRCUMSTANCES.
'' THE CALCULATION TO DETERMINE THE NEW POINT LOCATION IS ESSENTIALLY A MATTER OF TAKING THE SINE OR THE
'' COSINE OF THE ANGLE (AFTER CONVERTING IT TO RADIANS), AND MULTIPLYING THAT SINE OR COSINE BY THE MEASURED
'' DISTANCE.  PLEASE CONTACT THE AUTHOR IF THIS DOESN'T MAKE SENSE, OR IF YOU WOULD LIKE FURTHER EXPLANATION.
Dim NorthSouthDistance As Double
Dim EastWestDistance As Double
Dim EastWest As Integer
Dim NorthSouth As Integer

1227: If theAzimuth = 0 Or theAzimuth = 360 Then
1228:   NorthSouthDistance = theLength
1229:   NorthSouth = 1
1230:   EastWestDistance = 0
1231:   EastWest = 1
1232: ElseIf (theAzimuth = 180) Then
1233:   NorthSouthDistance = theLength
```

```
1234:   NorthSouth = -1
1235:   EastWestDistance = 0
1236:   EastWest = 1
1237: ElseIf (theAzimuth = 90) Then
1238:   NorthSouthDistance = 0
1239:   NorthSouth = 1
1240:   EastWestDistance = theLength
1241:   EastWest = 1
1242: ElseIf (theAzimuth = 270) Then
1243:   NorthSouthDistance = 0
1244:   NorthSouth = 1
1245:   EastWestDistance = theLength
1246:   EastWest = -1
1247: ElseIf ((theAzimuth > 0) And (theAzimuth < 90)) Then
1248:   NorthSouthDistance = Cos(AsRadians(theAzimuth)) * theLength
1249:   NorthSouth = 1
1250:   EastWestDistance = Sin(AsRadians(theAzimuth)) * theLength
1251:   EastWest = 1
1252: ElseIf ((theAzimuth > 90) And (theAzimuth < 180)) Then
1253:   NorthSouthDistance = (Sin(AsRadians(theAzimuth - 90))) * theLength
1254:   NorthSouth = -1
1255:   EastWestDistance = (Cos(AsRadians(theAzimuth - 90))) * theLength
1256:   EastWest = 1
1257: ElseIf ((theAzimuth > 180) And (theAzimuth < 270)) Then
1258:   NorthSouthDistance = (Cos(AsRadians(theAzimuth - 180))) * theLength
1259:   NorthSouth = -1
1260:   EastWestDistance = (Sin(AsRadians(theAzimuth - 180))) * theLength
1261:   EastWest = -1
1262: ElseIf ((theAzimuth > 270) And (theAzimuth < 360)) Then
1263:   NorthSouthDistance = (Sin(AsRadians(theAzimuth - 270))) * theLength
1264:   NorthSouth = 1
1265:   EastWestDistance = (Cos(AsRadians(theAzimuth - 270))) * theLength
1266:   EastWest = -1
1267: End If

Dim theMovementNorth As Double
Dim theMovementWest As Double

1272: theMovementNorth = NorthSouthDistance * NorthSouth
1273: theMovementWest = EastWestDistance * EastWest

Dim startX As Double
Dim startY As Double

1278: ptOrigin.QueryCoords startX, startY
1279: ptEndPoint.PutCoords startX + theMovementWest, startY + theMovementNorth
```

```
1281: If Not pLine Is Nothing Then
  Dim pPointColl As IPointCollection
1283:   pLine.SetEmpty
1284:   Set pPointColl = pLine
1285:   pPointColl.AddPoint ptOrigin
1286:   pPointColl.AddPoint ptEndPoint
1287: End If


  Exit Sub
ErrorHandler:
  HandleError True, "CalcPointLine " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Sub
Public Function EllipticArcToPolygon(SegCollection As ISegmentCollection, NumVertices As Long) As IPolygon4
  On Error GoTo ErrorHandler

'  Dim pMxDoc As IMxDocument
'  Set pMxDoc = ThisDocument

'  Dim pEllArc As IEllipticArc
  Dim pCurve As ICurve
  Dim pGeometry As IGeometry

  Dim anIndex As Long
  Dim lngSegCount As Long
1306:   lngSegCount = SegCollection.SegmentCount - 1
  Dim theLength As Double
1308:   theLength = 0
  Dim theTestLength As Double
  Dim lngLengths() As Long
  ReDim lngLengths(lngSegCount)
1312:   For anIndex = 0 To lngSegCount
1313:     theTestLength = SegCollection.Segment(anIndex).length
1314:     theLength = theLength + theTestLength
1315:     lngLengths(anIndex) = theTestLength
1316:   Next anIndex

  Dim pProportion As Double
  Dim lngVertices() As Long
  Dim lngNumVertices As Long
  ReDim lngVertices(lngSegCount)
1322:   For anIndex = 0 To lngSegCount
1323:     lngNumVertices = Int((lngLengths(anIndex) / theLength) * NumVertices)
1324:     If lngNumVertices < 8 Then lngNumVertices = 8
1325:     lngVertices(anIndex) = lngNumVertices
1326:   Next anIndex
```

```
  Dim pMpt As IPointCollection
1329:   Set pMpt = New Multipoint
  Dim pPoint As IPoint
1331:   Set pPoint = New Point
  Dim pClone As IClone

  Dim pRatio As Double
  Dim anIndex2 As Long

1337:   For anIndex = 0 To lngSegCount
1338:     lngNumVertices = lngVertices(anIndex)
1339:     pRatio = 1 / lngNumVertices
1340:     Set pCurve = SegCollection.Segment(anIndex)

1342:     For anIndex2 = 0 To lngNumVertices
'      If pGeometry.GeometryType = esriGeometryEllipticArc Then
1344:       pCurve.QueryPoint 0, (pRatio * anIndex2), True, pPoint
1345:       Set pClone = pPoint

 '   Graphic_MakeFromGeometry pMxDoc, pPoint, "DeleteMe"

1349:       pMpt.AddPoint pClone.Clone
1350:     Next anIndex2
1351:   Next anIndex

  Dim pPoly4 As IPolygon4
  Dim pTopoOp2 As ITopologicalOperator2
  Dim pTopoOp3 As ITopologicalOperator3
1356:   Set pTopoOp2 = pMpt
1357:   Set pPoly4 = pTopoOp2.ConvexHull
1358:   Set pTopoOp3 = pPoly4
1359:   pTopoOp3.IsKnownSimple = False
1360:   pTopoOp3.Simplify

1362:   Set EllipticArcToPolygon = pPoly4


  Exit Function
ErrorHandler:
  HandleError True, "EllipticArcToPolygon " & c_sModuleFileName & " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
```

Module 11: MyVBOperations

```
Attribute VB_Name = "MyVBOperations"
```

```
Option Explicit

'  ReturnControl - GIVEN A CONTROL NAME, SEARCHES THE FORM AND RETURNS THE CONTROL

'-------------------------------------------------------------------------------------------------------

Public Function ReturnControl(pForm As Form, strName As String) As Control

  Dim pControl As Control
  Dim pIndex As Long

12:   Set ReturnControl = pForm.Controls.Item(strName)

'  For pIndex = 1 To pForm.Controls.Count
'    Set pControl = pForm.Controls.Item(pIndex)
'    If pControl.Name = strName Then
'      Set ReturnControl = pControl
'      Exit For
'    End If
'  Next pIndex

End Function
```

Module 12: QuickSort

```
Attribute VB_Name = "QuickSort"
Option Explicit
Option Compare Binary

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Copyright ©1996-2005 VBnet, Randy Birch, All Rights Reserved.
' Some pages may also contain other copyrights by the author.
' see http://vbnet.mvps.org/index.html?code/sort/qsvariations.htm
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Distribution: You can freely use this code in your own
'               applications, but you may not reproduce
'               or publish this code on any web site,
'               online service, or distribute as source
'               on any media without express permission.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' MODIFIED JAN. 4, 2006 BY JEFF JENNESS, TO SIMPLIFY IMPLEMENTATION IN ARCGIS
'-----------------------------------------------------------------------------

Public Sub ByteAscending(nArray() As Byte, inLow As Long, inHi As Long)

   Dim pivot As Byte
   Dim tmpSwap As Byte
```

```
   Dim tmpLow As Long
   Dim tmpHi  As Long

25:    tmpLow = inLow
26:    tmpHi = inHi

28:    pivot = nArray((inLow + inHi) / 2)

30:    While (tmpLow <= tmpHi)

32:       While (nArray(tmpLow) < pivot And tmpLow < inHi)
33:          tmpLow = tmpLow + 1
34:       Wend

36:       While (pivot < nArray(tmpHi) And tmpHi > inLow)
37:          tmpHi = tmpHi - 1
38:       Wend

40:       If (tmpLow <= tmpHi) Then
41:          tmpSwap = nArray(tmpLow)
42:          nArray(tmpLow) = nArray(tmpHi)
43:          nArray(tmpHi) = tmpSwap
44:          tmpLow = tmpLow + 1
45:          tmpHi = tmpHi - 1
46:       End If

48:    Wend

50:    If (inLow < tmpHi) Then ByteAscending nArray(), inLow, tmpHi
51:    If (tmpLow < inHi) Then ByteAscending nArray(), tmpLow, inHi

End Sub


Public Sub ByteDescending(nArray() As Byte, inLow As Long, inHi As Long)

   Dim pivot As Byte
   Dim tmpSwap As Byte
   Dim tmpLow As Long
   Dim tmpHi  As Long

63:    tmpLow = inLow
64:    tmpHi = inHi

66:    pivot = nArray((inLow + inHi) / 2)

68:    While (tmpLow <= tmpHi)
```

```
70:       While (nArray(tmpLow) > pivot And tmpLow < inHi)
71:          tmpLow = tmpLow + 1
72:       Wend

74:       While (pivot > nArray(tmpHi) And tmpHi > inLow)
75:          tmpHi = tmpHi - 1
76:       Wend

78:       If (tmpLow <= tmpHi) Then
79:          tmpSwap = nArray(tmpLow)
80:          nArray(tmpLow) = nArray(tmpHi)
81:          nArray(tmpHi) = tmpSwap
82:          tmpLow = tmpLow + 1
83:          tmpHi = tmpHi - 1
84:       End If

86:    Wend

88:    If (inLow < tmpHi) Then ByteDescending nArray(), inLow, tmpHi
89:    If (tmpLow < inHi) Then ByteDescending nArray(), tmpLow, inHi

End Sub
Public Sub LongAscending(nArray() As Long, inLow As Long, inHi As Long)

   Dim pivot As Long
   Dim tmpSwap As Long
   Dim tmpLow As Long
   Dim tmpHi  As Long

99:    tmpLow = inLow
100:    tmpHi = inHi

102:    pivot = nArray((inLow + inHi) / 2)

104:    While (tmpLow <= tmpHi)

106:       While (nArray(tmpLow) < pivot And tmpLow < inHi)
107:          tmpLow = tmpLow + 1
108:       Wend

110:       While (pivot < nArray(tmpHi) And tmpHi > inLow)
111:          tmpHi = tmpHi - 1
112:       Wend

114:       If (tmpLow <= tmpHi) Then
115:          tmpSwap = nArray(tmpLow)
```

```
116:          nArray(tmpLow) = nArray(tmpHi)
117:          nArray(tmpHi) = tmpSwap
118:          tmpLow = tmpLow + 1
119:          tmpHi = tmpHi - 1
120:       End If

122:    Wend

124:    If (inLow < tmpHi) Then LongAscending nArray(), inLow, tmpHi
125:    If (tmpLow < inHi) Then LongAscending nArray(), tmpLow, inHi

End Sub


Public Sub LongDescending(nArray() As Long, inLow As Long, inHi As Long)

   Dim pivot As Long
   Dim tmpSwap As Long
   Dim tmpLow As Long
   Dim tmpHi  As Long

137:    tmpLow = inLow
138:    tmpHi = inHi

140:    pivot = nArray((inLow + inHi) / 2)

142:    While (tmpLow <= tmpHi)

144:       While (nArray(tmpLow) > pivot And tmpLow < inHi)
145:          tmpLow = tmpLow + 1
146:       Wend

148:       While (pivot > nArray(tmpHi) And tmpHi > inLow)
149:          tmpHi = tmpHi - 1
150:       Wend

152:       If (tmpLow <= tmpHi) Then
153:          tmpSwap = nArray(tmpLow)
154:          nArray(tmpLow) = nArray(tmpHi)
155:          nArray(tmpHi) = tmpSwap
156:          tmpLow = tmpLow + 1
157:          tmpHi = tmpHi - 1
158:       End If

160:    Wend

162:    If (inLow < tmpHi) Then LongDescending nArray(), inLow, tmpHi
```

```
163:    If (tmpLow < inHi) Then LongDescending nArray(), tmpLow, inHi

End Sub



Public Sub SingleAscending(nArray() As Single, inLow As Long, inHi As Long)

   Dim pivot As Single
   Dim tmpSwap As Single
   Dim tmpLow As Long
   Dim tmpHi  As Long

177:    tmpLow = inLow
178:    tmpHi = inHi

180:    pivot = nArray((inLow + inHi) / 2)

182:    While (tmpLow <= tmpHi)

184:       While (nArray(tmpLow) < pivot And tmpLow < inHi)
185:          tmpLow = tmpLow + 1
186:       Wend

188:       While (pivot < nArray(tmpHi) And tmpHi > inLow)
189:          tmpHi = tmpHi - 1
190:       Wend

192:       If (tmpLow <= tmpHi) Then
193:          tmpSwap = nArray(tmpLow)
194:          nArray(tmpLow) = nArray(tmpHi)
195:          nArray(tmpHi) = tmpSwap
196:          tmpLow = tmpLow + 1
197:          tmpHi = tmpHi - 1
198:       End If

200:    Wend

202:    If (inLow < tmpHi) Then SingleAscending nArray(), inLow, tmpHi
203:    If (tmpLow < inHi) Then SingleAscending nArray(), tmpLow, inHi

End Sub


Public Sub SingleDescending(nArray() As Single, inLow As Long, inHi As Long)
```

```
   Dim pivot As Single
   Dim tmpSwap As Single
   Dim tmpLow As Long
   Dim tmpHi  As Long

215:    tmpLow = inLow
216:    tmpHi = inHi

218:    pivot = nArray((inLow + inHi) / 2)

220:    While (tmpLow <= tmpHi)

222:       While (nArray(tmpLow) > pivot And tmpLow < inHi)
223:          tmpLow = tmpLow + 1
224:       Wend

226:       While (pivot > nArray(tmpHi) And tmpHi > inLow)
227:          tmpHi = tmpHi - 1
228:       Wend

230:       If (tmpLow <= tmpHi) Then
231:          tmpSwap = nArray(tmpLow)
232:          nArray(tmpLow) = nArray(tmpHi)
233:          nArray(tmpHi) = tmpSwap
234:          tmpLow = tmpLow + 1
235:          tmpHi = tmpHi - 1
236:       End If

238:    Wend

240:    If (inLow < tmpHi) Then SingleDescending nArray(), inLow, tmpHi
241:    If (tmpLow < inHi) Then SingleDescending nArray(), tmpLow, inHi

End Sub
Public Sub DoubleAscendingWithObjects(nArray() As Double, varObjArray() As Variant, inLow As Long, inHi As Long)

  Dim pivot As Double
  Dim tmpSwap As Double
  Dim tmpSizeSwap As Variant
  Dim tmpLow As Long
  Dim tmpHi  As Long

252:   tmpLow = inLow
253:   tmpHi = inHi

255:   pivot = nArray((inLow + inHi) / 2)
256:   While (tmpLow <= tmpHi)
```

```
258:     While (nArray(tmpLow) < pivot And tmpLow < inHi)
259:        tmpLow = tmpLow + 1
260:     Wend

262:     While (pivot < nArray(tmpHi) And tmpHi > inLow)
263:        tmpHi = tmpHi - 1
264:     Wend
265:     If (tmpLow <= tmpHi) Then
266:        tmpSwap = nArray(tmpLow)
267:        nArray(tmpLow) = nArray(tmpHi)
268:        nArray(tmpHi) = tmpSwap

270:        tmpSizeSwap = varObjArray(tmpLow)
271:        varObjArray(tmpLow) = varObjArray(tmpHi)
272:        varObjArray(tmpHi) = tmpSizeSwap

274:        tmpLow = tmpLow + 1
275:        tmpHi = tmpHi - 1
276:     End If

278:   Wend

280:   If (inLow < tmpHi) Then DoubleAscendingWithObjects nArray(), varObjArray(), inLow, tmpHi
281:   If (tmpLow < inHi) Then DoubleAscendingWithObjects nArray(), varObjArray(), tmpLow, inHi

End Sub

Public Sub DoubleAscendingWithSizes(nArray() As Double, nSizeArray() As Double, inLow As Long, inHi As Long)

  Dim pivot As Double
  Dim tmpSwap As Double
  Dim tmpSizeSwap As Double
  Dim tmpLow As Long
  Dim tmpHi  As Long

293:   tmpLow = inLow
294:   tmpHi = inHi

296:   pivot = nArray((inLow + inHi) / 2)
297:   While (tmpLow <= tmpHi)

299:     While (nArray(tmpLow) < pivot And tmpLow < inHi)
300:        tmpLow = tmpLow + 1
301:     Wend

303:     While (pivot < nArray(tmpHi) And tmpHi > inLow)
```

```
304:        tmpHi = tmpHi - 1
305:     Wend
306:     If (tmpLow <= tmpHi) Then
307:        tmpSwap = nArray(tmpLow)
308:        nArray(tmpLow) = nArray(tmpHi)
309:        nArray(tmpHi) = tmpSwap

311:        tmpSizeSwap = nSizeArray(tmpLow)
312:        nSizeArray(tmpLow) = nSizeArray(tmpHi)
313:        nSizeArray(tmpHi) = tmpSizeSwap

315:        tmpLow = tmpLow + 1
316:        tmpHi = tmpHi - 1
317:     End If

319:   Wend

321:   If (inLow < tmpHi) Then DoubleAscendingWithSizes nArray(), nSizeArray(), inLow, tmpHi
322:   If (tmpLow < inHi) Then DoubleAscendingWithSizes nArray(), nSizeArray(), tmpLow, inHi

End Sub
Public Sub DoubleAscending(nArray() As Double, inLow As Long, inHi As Long)

   Dim pivot As Double
   Dim tmpSwap As Double
   Dim tmpLow As Long
   Dim tmpHi  As Long

332:    tmpLow = inLow
333:    tmpHi = inHi

335:    pivot = nArray((inLow + inHi) / 2)

337:    While (tmpLow <= tmpHi)

339:       While (nArray(tmpLow) < pivot And tmpLow < inHi)
340:          tmpLow = tmpLow + 1
341:       Wend

343:       While (pivot < nArray(tmpHi) And tmpHi > inLow)
344:          tmpHi = tmpHi - 1
345:       Wend

347:       If (tmpLow <= tmpHi) Then
348:          tmpSwap = nArray(tmpLow)
349:          nArray(tmpLow) = nArray(tmpHi)
350:          nArray(tmpHi) = tmpSwap
```

```
351:          tmpLow = tmpLow + 1
352:          tmpHi = tmpHi - 1
353:       End If

355:    Wend

357:    If (inLow < tmpHi) Then DoubleAscending nArray(), inLow, tmpHi
358:    If (tmpLow < inHi) Then DoubleAscending nArray(), tmpLow, inHi

End Sub


Public Sub DoubleDescending(nArray() As Double, inLow As Long, inHi As Long)

   Dim pivot As Double
   Dim tmpSwap As Double
   Dim tmpLow As Long
   Dim tmpHi  As Long

370:    tmpLow = inLow
371:    tmpHi = inHi

373:    pivot = nArray((inLow + inHi) / 2)

375:    While (tmpLow <= tmpHi)

377:       While (nArray(tmpLow) > pivot And tmpLow < inHi)
378:          tmpLow = tmpLow + 1
379:       Wend

381:       While (pivot > nArray(tmpHi) And tmpHi > inLow)
382:          tmpHi = tmpHi - 1
383:       Wend

385:       If (tmpLow <= tmpHi) Then
386:          tmpSwap = nArray(tmpLow)
387:          nArray(tmpLow) = nArray(tmpHi)
388:          nArray(tmpHi) = tmpSwap
389:          tmpLow = tmpLow + 1
390:          tmpHi = tmpHi - 1
391:       End If

393:    Wend

395:    If (inLow < tmpHi) Then DoubleDescending nArray(), inLow, tmpHi
396:    If (tmpLow < inHi) Then DoubleDescending nArray(), tmpLow, inHi
```

```
End Sub

Public Sub StringsAscending(sarray() As String, inLow As Long, inHi As Long)

   Dim pivot As String
   Dim tmpSwap As String
   Dim tmpLow As Long
   Dim tmpHi As Long

407:    tmpLow = inLow
408:    tmpHi = inHi

410:    pivot = sarray((inLow + inHi) / 2)

412:    While (tmpLow <= tmpHi)

414:       While (sarray(tmpLow) < pivot And tmpLow < inHi)
415:          tmpLow = tmpLow + 1
416:       Wend

418:       While (pivot < sarray(tmpHi) And tmpHi > inLow)
419:          tmpHi = tmpHi - 1
420:       Wend

422:       If (tmpLow <= tmpHi) Then
423:          tmpSwap = sarray(tmpLow)
424:          sarray(tmpLow) = sarray(tmpHi)
425:          sarray(tmpHi) = tmpSwap
426:          tmpLow = tmpLow + 1
427:          tmpHi = tmpHi - 1
428:       End If

430:    Wend

432:    If (inLow < tmpHi) Then StringsAscending sarray(), inLow, tmpHi
433:    If (tmpLow < inHi) Then StringsAscending sarray(), tmpLow, inHi

End Sub


Public Sub StringsDescending(sarray() As String, inLow As Long, inHi As Long)

   Dim pivot As String
   Dim tmpSwap As String
   Dim tmpLow As Long
   Dim tmpHi As Long
```

```
445:    tmpLow = inLow
446:    tmpHi = inHi

448:    pivot = sarray((inLow + inHi) / 2)

450:    While (tmpLow <= tmpHi)

452:       While (sarray(tmpLow) > pivot And tmpLow < inHi)
453:          tmpLow = tmpLow + 1
454:       Wend

456:       While (pivot > sarray(tmpHi) And tmpHi > inLow)
457:          tmpHi = tmpHi - 1
458:       Wend

460:       If (tmpLow <= tmpHi) Then
461:          tmpSwap = sarray(tmpLow)
462:          sarray(tmpLow) = sarray(tmpHi)
463:          sarray(tmpHi) = tmpSwap
464:          tmpLow = tmpLow + 1
465:          tmpHi = tmpHi - 1
466:       End If

468:    Wend

470:    If (inLow < tmpHi) Then StringsDescending sarray(), inLow, tmpHi
471:    If (tmpLow < inHi) Then StringsDescending sarray(), tmpLow, inHi

End Sub


Public Sub VariantAscending(sarray() As Variant, inLow As Long, inHi As Long)

   Dim pivot As Variant
   Dim tmpSwap As Variant
   Dim tmpLow As Long
   Dim tmpHi As Long

483:    tmpLow = inLow
484:    tmpHi = inHi

486:    pivot = sarray((inLow + inHi) / 2)

488:    While (tmpLow <= tmpHi)

490:       While (sarray(tmpLow) < pivot And tmpLow < inHi)
491:          tmpLow = tmpLow + 1
```

```
492:       Wend

494:       While (pivot < sarray(tmpHi) And tmpHi > inLow)
495:          tmpHi = tmpHi - 1
496:       Wend

498:       If (tmpLow <= tmpHi) Then
499:          tmpSwap = sarray(tmpLow)
500:          sarray(tmpLow) = sarray(tmpHi)
501:          sarray(tmpHi) = tmpSwap
502:          tmpLow = tmpLow + 1
503:          tmpHi = tmpHi - 1
504:       End If

506:    Wend

508:    If (inLow < tmpHi) Then VariantAscending sarray(), inLow, tmpHi
509:    If (tmpLow < inHi) Then VariantAscending sarray(), tmpLow, inHi

End Sub


Public Sub VariantDescending(sarray() As Variant, inLow As Long, inHi As Long)

   Dim pivot As Variant
   Dim tmpSwap As Variant
   Dim tmpLow As Long
   Dim tmpHi As Long

521:    tmpLow = inLow
522:    tmpHi = inHi

524:    pivot = sarray((inLow + inHi) / 2)

526:    While (tmpLow <= tmpHi)

528:       While (sarray(tmpLow) > pivot And tmpLow < inHi)
529:          tmpLow = tmpLow + 1
530:       Wend

532:       While (pivot > sarray(tmpHi) And tmpHi > inLow)
533:          tmpHi = tmpHi - 1
534:       Wend

536:       If (tmpLow <= tmpHi) Then
537:          tmpSwap = sarray(tmpLow)
538:          sarray(tmpLow) = sarray(tmpHi)
```

```
539:          sarray(tmpHi) = tmpSwap
540:          tmpLow = tmpLow + 1
541:          tmpHi = tmpHi - 1
542:       End If

544:    Wend

546:    If (inLow < tmpHi) Then VariantDescending sarray(), inLow, tmpHi
547:    If (tmpLow < inHi) Then VariantDescending sarray(), tmpLow, inHi

End Sub

Public Sub DatesDescending(nArray() As Date, inLow As Long, inHi As Long)

   Dim pivot As Long
   Dim tmpSwap As Long
   Dim tmpLow As Long
   Dim tmpHi  As Long

558:    tmpLow = inLow
559:    tmpHi = inHi

561:    pivot = DateToJulian(nArray((inLow + inHi) / 2))

563:    While (tmpLow <= tmpHi)

565:       While DateToJulian(nArray(tmpLow)) > pivot And (tmpLow < inHi)
566:          tmpLow = tmpLow + 1
567:       Wend

569:       While (pivot > DateToJulian(nArray(tmpHi))) And (tmpHi > inLow)
570:          tmpHi = tmpHi - 1
571:       Wend

573:       If (tmpLow <= tmpHi) Then
574:          tmpSwap = nArray(tmpLow)
575:          nArray(tmpLow) = nArray(tmpHi)
576:          nArray(tmpHi) = tmpSwap
577:          tmpLow = tmpLow + 1
578:          tmpHi = tmpHi - 1
579:       End If

581:    Wend

583:    If (inLow < tmpHi) Then DatesDescending nArray(), inLow, tmpHi
584:    If (tmpLow < inHi) Then DatesDescending nArray(), tmpLow, inHi
```

```
End Sub


Public Sub DatesAscending(nArray() As Date, inLow As Long, inHi As Long)

   Dim pivot As Long
   Dim tmpSwap As Long
   Dim tmpLow As Long
   Dim tmpHi  As Long

596:    tmpLow = inLow
597:    tmpHi = inHi

599:    pivot = DateToJulian(nArray((inLow + inHi) / 2))

601:    While (tmpLow <= tmpHi)

603:       While (DateToJulian(nArray(tmpLow)) < pivot) And (tmpLow < inHi)
604:          tmpLow = tmpLow + 1
605:       Wend

607:       While (pivot < DateToJulian(nArray(tmpHi))) And (tmpHi > inLow)
608:          tmpHi = tmpHi - 1
609:       Wend

611:       If (tmpLow <= tmpHi) Then

613:          tmpSwap = nArray(tmpLow)
614:          nArray(tmpLow) = nArray(tmpHi)
615:          nArray(tmpHi) = tmpSwap
616:          tmpLow = tmpLow + 1
617:          tmpHi = tmpHi - 1

619:       End If

621:    Wend

623:    If (inLow < tmpHi) Then DatesAscending nArray(), inLow, tmpHi
624:    If (tmpLow < inHi) Then DatesAscending nArray(), tmpLow, inHi

End Sub

Private Function DateToJulian(MyDate As Date) As Long

  'Return a numeric value representing
  'the passed date
632:    DateToJulian = dateValue(MyDate)
```

```
End Function
```