GEOG 490/590
GIS Programming
Lab 5: ArcMap UITool, Commands, and Vector Features Operations

Introduction: You will create a UITool that allows users to select polygons from a polygon layer in ArcMap and report the number and total area of the selected polygons. You can find a polygon shapefile to use as test data or download this subset of RLIS taxlot data.

Polygon Area Tool (PAT):
Users can use the PAT to draw a rectangular box and select polygons that fall within the rectangular area. ArcMap will zoom in to the selected features and a popup message box then shows the number of features selected, the total area of the selected features, and the linear unit. **The required output of the PAT tool is this popup message box.**

UITool Control:
Please refer to Lab 1 and Exercise 6 for instructions on how to create custom tool controls for ArcMap addIns. Don't forget to set the tooltip of the PAT UITool control and assign an appropriate button image. As in previous labs, the category must be **GEOG 4/590 Add-In Controls** so the instructor can find your work.

Setting the Cursor Type:
You should use the cursor property of the Tool object to set the cursor type. Adding the following code to the New() Sub of your UITool will change the cursor type to a crosshair. Other cursor types are available here.

```
Me.Cursor = Windows.Forms.Cursors.Cross
```

MouseDown Event:
The PAT tool requires a user to click the mouse cursor on a map and drag the cursor to select a rectangular area. That means all your code must be associated with the OnMouseDown() subroutine.

```
Protected Overrides Sub OnMouseDown(ByVal arg As
                                    ESRI.ArcGIS.Desktop.AddIns.Tool.MouseEventArgs)
  MyBase.OnMouseDown(arg)
  ' Put your VB .NET code here
End Sub
```

Drawing a Rectangle:

You use the RubberBand object to track the movement of the mouse cursor. When done, you assign the tracked rectangular geometry to a graphic element. You then display the rectangular area as a rectangular graphic element. This can be achieved by adding the code below to the OnMouseDown() subroutine.

```
    Dim pEnv As IEnvelope2
    Dim pRubberEnv As IRubberBand = New RubberEnvelope
    Dim pElem As IElement
    Dim pFillShapeElem As IFillShapeElement
    Dim pFillSymbol As IFillSymbol
    Dim pColor As IColor
    Dim pLineSymbol As ILineSymbol
    Dim pGContainer As IGraphicsContainer

    Try
        'use the RubberBand object to track the movement of mouse cursor
        pEnv = GetEnvelopeFromMouseClicks(My.Document.ActivatedView)
        'selected elements
        pElem = GetRectangleFromEnvelope(pEnv)

        'graphics container
        pGContainer = My.Document.ActivatedView
        'add pElem with default order
        pGContainer.AddElement(pElem, 0)

My.Document.ActivatedView.PartialRefresh(esriViewDrawPhase.esriViewGraphics,
pGContainer, Nothing)

    Catch ex As Exception
        MsgBox("OnMouseDown Exception: " & ex.Message)
    Finally
        pGContainer = Nothing
        pLineSymbol = Nothing
        pColor = Nothing
        pFillSymbol = Nothing
        pFillShapeElem = Nothing
        pElem = Nothing
        pRubberEnv = Nothing
        pEnv = Nothing
        GC.WaitForPendingFinalizers()
        GC.Collect()
    End Try
```

You'll also need the following two "helper" functions:

```
    Private Function GetEnvelopeFromMouseClicks(ByVal activeView As IActiveView) As
IEnvelope2
        'get the screenDisplay from the activeView which comes from
My.Document.ActivatedView
        Dim screenDisplay As IScreenDisplay = activeView.ScreenDisplay
        'use the RubberBand object to track the movement of mouse cursor
        Dim rubberBand As IRubberBand2 = New RubberEnvelope
        'RubberBand.TrackNew() returns an IGeometry object
        Dim geometry As IGeometry5 = rubberBand.TrackNew(screenDisplay, Nothing)
        'Cast IGeometry to Envelope; Envelope implements IGeometry5
        Dim env As IEnvelope2 = CType(geometry, Envelope)
        Return env
    End Function

    Private Function GetRectangleFromEnvelope(ByVal env As IEnvelope) As
IRectangleElement
        'Create new Rectangle object
        Dim pElem As IElement = New RectangleElement
        'Set IEnvelope as the geometry
        pElem.Geometry = env
        'Create object so we can set the symbology for the shape
        Dim pFillShapeElem As IFillShapeElement = pElem
        'Create object so we can set the fill symbology
        Dim pFillSymbol As IFillSymbol = pFillShapeElem.Symbol
        'Get line symbology from fill symbology
        Dim pLineSymbol As ILineSymbol = pFillSymbol.Outline
        'Create object to set the fill transparency via color property
        Dim pFillColor As IColor = pFillSymbol.Color
        'Create object to set the line color
        Dim pLineColor As IColor = pLineSymbol.Color
        'set background to transparent for fill symbol
        pFillColor.Transparency = 0
        'set line color to green
        pLineColor.RGB = RGB(0, 255, 0)
        'set line width
        pLineSymbol.Width = 2.0
        'set Color on ILineSymbol
        pLineSymbol.Color = pLineColor
        'set transparency (color) on IFillSymbol
        pFillSymbol.Color = pFillColor
        'set ILineSymbol on IFillSymbol
        pFillSymbol.Outline = pLineSymbol
        'set symbology on fill shape element
        pFillShapeElem.Symbol = pFillSymbol
        Return pElem
    End Function
```

Make sure you remember to clear your graphics container **before** the PAT draws a new rectangle. This can be done with the DeleteAllElements() method of the IGraphicsContainer interface.

Select and Zoom to Features:

You should use ArcMap built-in commands to accomplish the next tasks. First use the Edit_SelectAll command to select the rectangular graphic element, then use the Query_SelectByGraphics command to select polygon features, and then use the Query_ZoomToSelected command to zoom to the selected polygons.

The following subroutine can be used to call each of the built in commands:

```
    Private Sub FindCommandAndExecute(ByVal application As
ESRI.ArcGIS.Framework.IApplication, ByVal commandName As String)
        Dim document As ESRI.ArcGIS.Framework.IDocument = My.ArcMap.Document
        ' Get the commandBars object from the document
        Dim commandBars As ESRI.ArcGIS.Framework.ICommandBars =
document.CommandBars
        Dim uid As UID = New UIDClass()
        ' Example: "esriFramework.HelpContentsCommand" or "{D74B2F25-AC90-
11D2-87F8-0000F8751720}"
        uid.Value = commandName
        Dim commandItem As ESRI.ArcGIS.Framework.ICommandItem =
commandBars.Find(uid, False, False)
        If Not (commandItem Is Nothing) Then
            'Execute the command
            commandItem.Execute()
        End If
    End Sub
```

You pass the IApplication object along with the CLSID or ProgID of the command you want to execute when you call this subroutine. A list of CLSID and ProgID's for built in ArcMap commands can be found here.

Here is an example of calling the above subroutine for the Edit_SelectAll command. You will call the subroutine a total of 3 times: once for each command that needs to be executed

```
FindCommandAndExecute(My.ArcMap.Application, _
      "esriArcMapUI.EditSelectAllCommand")
```

Spatial Query:

You perform a spatial query to select the polygon features that fall within the rectangle area specified by the UITool. The opening part of a sample subroutine for a spatial query is below.

```
Private Function SumAreaForSelectedFeatures(ByVal pEnv As IEnvelope) As Double
    Dim pSFilter As ISpatialFilter
    Dim pFLayer As IFeatureLayer
    Dim pFCursor As IFeatureCursor
    Dim pFeature As IFeature
    Dim pArea As IArea
    Dim sumArea As Double

    Try
        'prepare a spatial filter
        pSFilter = New SpatialFilter
        pSFilter.Geometry = pEnv 'pEnv is the envelope geometry
        'you got from the IrubberBand interface
        pSFilter.SpatialRel = esriSpatialRelEnum.esriSpatialRelIntersects

        'assign the reference to a feature layer
        pFLayer = My.Document.FocusMap.Layer(0)
        pFCursor = pFLayer.Search(pSFilter, False)
        pFeature = pFCursor.NextFeature

        'This is how you get an individual feature
        'from the feature cursor. You need to use a loop
        'to access all selected features
```

Calculating Total Area:

The next step is to sum the areas of all the selected features (i.e., the features in the Feature Cursor.) You use the IArea interface to extract the area information of the shape geometry. This is done by the following code. You have to put the code in a loop to sum all area values.

```
pArea = pFeature.Shape
sumArea = sumArea + pArea.Area
```

Retrieving the Linear Unit:

The total area isn't very useful without knowing the unit of measure. You can use the code below to work your way down to the ILinearUnit object from the FeatureClass. Note that outside of this code block, we got a reference to a FeatureLayer from the Map so it could be interrogated.

You can examine the properties of the ILinearUnit object to find the unit of measure. The 'Name' property of the ILinearUnit will return the unit in String format similar to the layer information found in ArcMap.

```
Dim geoDataSet As IGeoDataset
Dim pSpRef As ISpatialReference
Dim projCoordSys As IProjectedCoordinateSystem
Dim pLinearUnit As ILinearUnit

'Get the geoDataSet from the FeatureClass
geoDataSet = pFLayer.FeatureClass
'Get the spatial reference of the geoDataSet
pSpRef = geoDataSet.SpatialReference
'Cast the spatial reference to the projected coordinate system
projCoordSys = pSpRef
'Get the linear unit from the projected coordinate system
 pLinearUnit = projCoordSys.CoordinateUnit
```

You might want to use the PanZoom_FullExtent command (see the FindCommandAndExecute() subroutine above) to reset the map display extent after the message box shows the query result.

Questions:

Q1) What is the difference between a Button and a Tool control in an ArcMap addIn? When might you choose to use one over the other?

Q2)If you wanted to make your rectangle blue rather than green, you would only need to change one line of code. What is that modified line of code?

Q3)Provide the CLSID and ProgId for the PanZoom_FullExtent command.

Q4)Your sumArea total will likely return a number that isn't user friendly. ie: A long number with no commas and many places behind the decimal point. Describe one or more VB .NET functions you could use to make this number easier to read.

Submit the answers to the questions along with your VB .NET project in a .zip file e-mailed to the instructor. Don't forget to use try/catch/finally blocks to catch errors, or to comment your code!