GEOG 490/590
GIS Programming
Lab 4: ArcMap Layers Display & Attribute Data Manipulation

Introduction: Now things start to get interesting as we practice using the ArcObjects VB .NET API. You will create a Layers & Attribute Manipulation (LAM) AddIn that mimics what the instructor demonstrated in class. Start by finding several small shapefiles to use for test data. The shapefiles must have numerical attribute fields. One good example is the RLIS census track dataset on the I:\ drive.

User requirements: The LAM Tool (Figure 1) allows users to add/delete GIS data layers to/from the active ArcMap map frame. A listBox shows the names of the layers that are present in the frame. The order of the items in the listBox can be changed and is linked with the display order in ArcMap (Figure 2). Once an item in the Layer listBox is selected, its attribute fields can be displayed in an attribute listBox. Users can select a numerical field from the attribute listBox to display its minimum, maximum, and average.

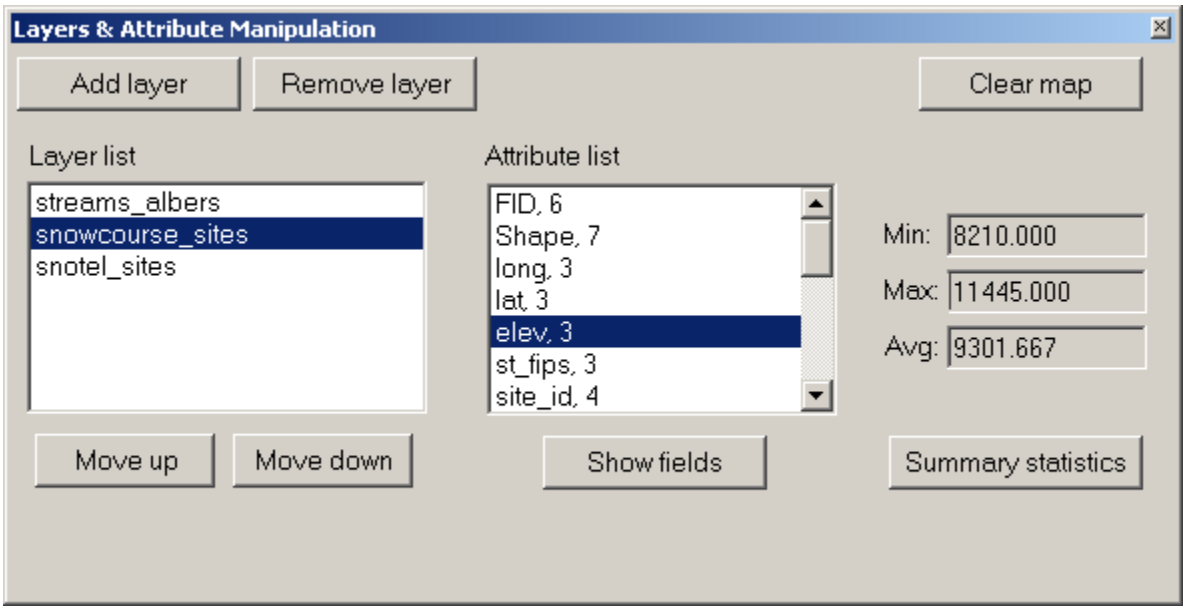

Figure 1

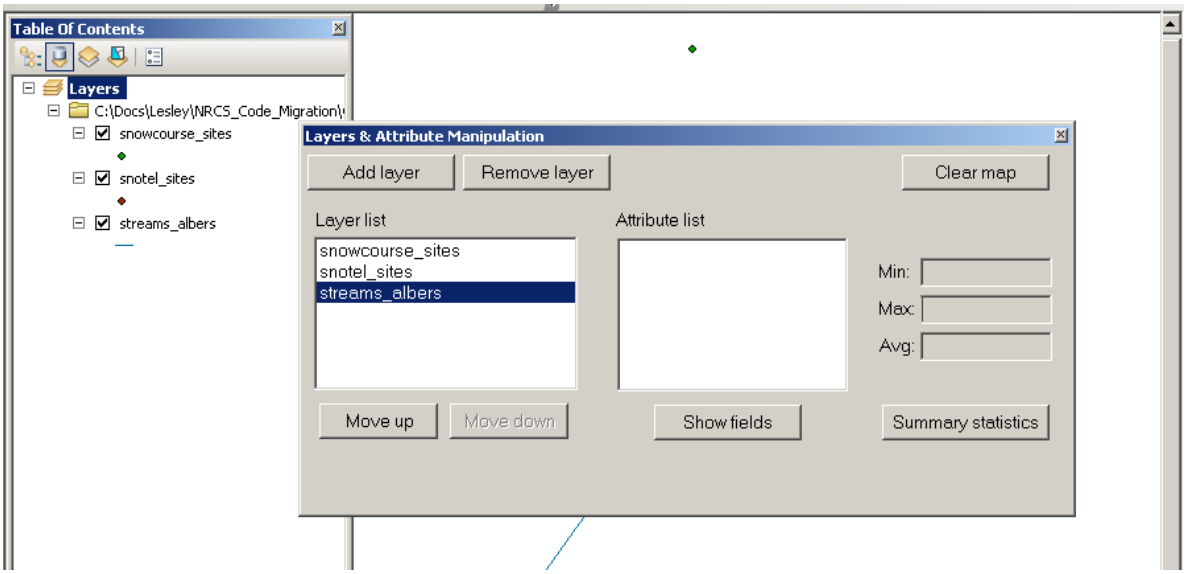

Figure 2

Creating the GUI:

1. Create a new ArcGIS Desktop Add-In project for ArcMap. Include your last name when naming your Visual Basic project: for example, BrossLab4.

2. Add a Button and a DockableWindow to your addIn. This part of Lab 4 is similar to Lab 2 so see the instructions for Lab 2 if you need instructions

3. The category for your addIn button needs to be '**GEOG 4/590 Add-In Controls**' so the instructor can find your work.

4. When the New Project Wizard has finished creating the project use File > Save All to save your project. Verify you are using the same name you used in step 1 when naming the project. Save your project in a folder in the C:\Users folder and regularly back up your data to your H:\ drive or portable device.

5. A sample layout of the DockableWindow of the AddIn is shown below in Figure 3. The form should have:
    1. Two buttons for adding / removing layers from the display list.
    2. Two buttons for moving a selected layer up and down the display list.
    3. Two listBoxes for showing the names of map layers and attribute fields.
    4. A button for showing the attribute fields.

5. A button for calculating the summary statistics of a selected numerical field.
6. Three textBoxes for displaying the minimum, maximum, and average.
7. A button to clear the layer list and reset the addIn form.
8. You also need to add labels to describe the controls (fields and buttons) on the form.

Hints:

- Remember the naming conventions: Buttons start with Btn, ListBoxes start with Lst, TextBoxes start with Txt, Labels start with Lbl
- The textBoxes on this form should be read-only since we aren't processing the user input

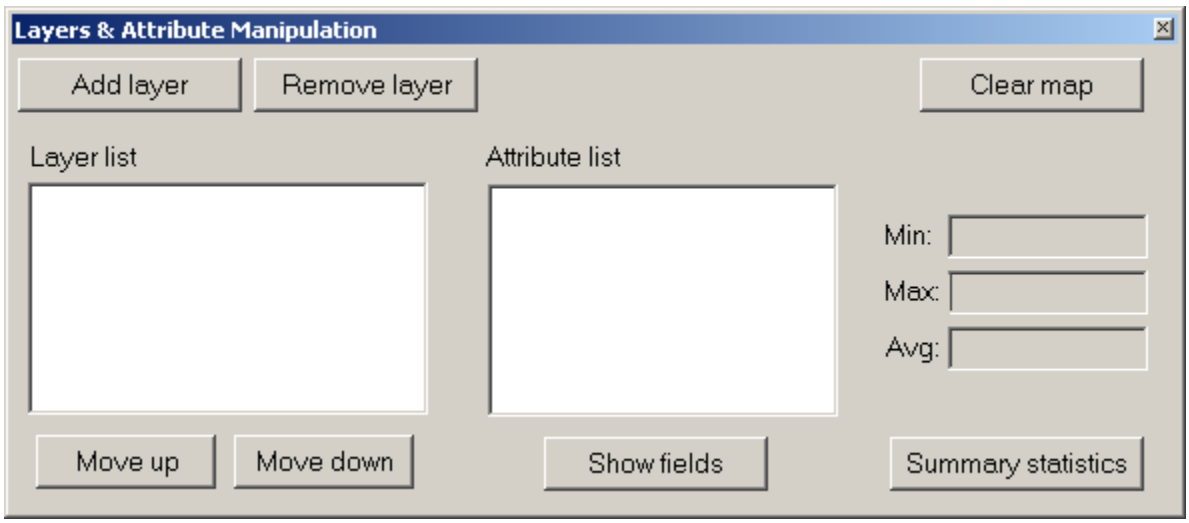

Figure 3

6. Copy and paste the following code into the OnClick() sub of your addIn Button class. This code toggles the display of the DockableWindow when you click on the addIn Button. Note that you may have to change the form name to match the name of your form.

```
Protected Overrides Sub OnClick()

      My.ArcMap.Application.CurrentTool = Nothing
      ' Declare the IDockableWindow class
      Dim dockWindow As ESRI.ArcGIS.Framework.IDockableWindow
      ' Declare the UID class
      Dim dockWinID As ESRI.ArcGIS.esriSystem.UID = New
   ESRI.ArcGIS.esriSystem.UIDClass()
      ' Set value property of UID to the form we wish to display
      dockWinID.Value = My.ThisAddIn.IDs.FrmLam
      ' Get the dockable window containing the form frm the
   DockableWindowManager
      dockWindow =
   My.ArcMap.DockableWindowManager.GetDockableWindow(dockWinID)
      ' Toggle the dockable window visible/invisible depending on current
   setting
      dockWindow.Show((Not dockWindow.IsVisible()))
   End Sub
```

7. Verify that the height and width of the DockableWindow in the Config.esriaddinx file match the height and width properties of FrmLam.

8. Start ArcMap from inside Visual Basic Express using the debug functionality. Use the Customize > Customize Mode… dialog to add the BtnLam command button to one of your toolbars. Click the button to verify that your form looks as expected.

Add layer button

1. When the 'Add Layer' button is clicked, you will use the GxDialog object to allow the user to select a shapefile layer. You need to change the GxObjectFilter type to GxFilterShapefiles. This means your dialog box will only show shapefiles. That is:

```
Dim pFilter As IGxObjectFilter
pFilter = New GxFilterShapefiles
```

   Once the GxDialog object returns a value successfully, you can use the following code to display the map.

```
pGxDataset = pGxObjects.Next
Dim pFLayer As IfeatureLayer
pFLayer = New FeatureLayer
pFLayer.FeatureClass = pGxDataset.Dataset
pFLayer.Name = pGxDataset.Dataset.Name


'add the feature layer to the active map
pMap = My.Document.Maps.Item(0)
pMap.AddLayer(pFLayer)
```

2. Note the use of the built-in ArcGIS addIn 'My.Document' object to access the MxDocument object. We then have access to the IMap object through the MxDocument collection of maps. Although an MxDocument can have a collection of maps, for the purpose of this lab assume that there is only one. Remember that collection indices are zero-based so we will always work with Item(0).

You also need to add the name of the layer to the Layer listBox. Please note that the AddLayer() method of a Map object adds a layer to the top of the TOC (Table of Contents) of the map while the AddItem() method of a listBox adds an item to the end of the list.

One easy way to synchronize the TOC and the listBox is: whenever a layer is added or removed, the program resets the listBox and displays the updated TOC. The same procedure may be used when the order of the items in the layer list is modified. This is a good reason to create a private subroutine to update the listBox using the map layer information.

Add the following subroutine to the end of your dockableWindow code (but before the End Class declaration). Please note that the sample code is just for your reference, you must read it carefully

and modify it to fit your program.

```
Private Sub RefreshLayerNames()
    Dim layercounter As Integer
    Dim i As Integer

    'reset the lstLayer list box
    LstLayers.Items.Clear()

    'add the layer name to the list box
    Dim pMap As IMap = My.Document.Maps.Item(0)
    layercounter = pMap.LayerCount - 1
    For i = 0 To layercounter
        LstLayers.Items.Add(pMap.Layer(i).Name)
    Next
End Sub
```

When you need to update the Layer list box, call the RefreshLayerNames() subroutine in your program. You can try using this subroutine in the Move Down/Up and Remove Layer sections below.

Remove layer button

When a user clicks on an item in the Layer listBox, the Remove Layer command button becomes enabled. Hint: Look at the SelectedIndexChanged event for ListBoxes.

You will use the DeleteLayer() method of a Map object to remove a layer. You also need to remove the selected item from the Layer listBox. As mentioned earlier, you can use the RefreshLayerNames() subroutine to sync the list box, or use the RemoveAt() method of a listBox to remove the item. You also need to check if there are any layers left in the listBox. If the listBox is empty, then you need to disable the Remove Layer button.

```
Dim pFLayer As IFeatureLayer
Dim pmap As IMap = My.Document.Maps.Item(0)
pFLayer = pmap.Layer(LstLayers.SelectedIndex)
pmap.DeleteLayer(pFLayer)
LstLayers.Items.RemoveAt(LstLayers.SelectedIndex)
My.Document.ActivatedView.Refresh()
```

Move Down/Move Up Buttons & Layer ListBox

1. To use the Move Down/Move Up functions, the user has to select an item from the Layer listBox first. Then, depending on which item is selected, the program enables or disables the Move Down or Move Up buttons.

   The code to control these actions needs to be put in the SelectedIndexChanged event subprocedure (i.e., LstLayers_SelectedIndexChanged, when the name of the Layer listBox is

LstLayers.) The SelectedItem property of the listBox tells you which item in the list is selected. When an item in the Layer listBox is selected, the Remove Layer and Show Attribute Fields command buttons are enabled.

```
Private Sub LstLayers_SelectedIndexChanged(ByVal sender _
      As System.Object,ByVal e As System.EventArgs) _Handles
      LstLayers.SelectedIndexChanged
        Dim item As Object = LstLayers.SelectedItem
      ' If something is selected
        If item IsNot Nothing Then
            BtnRemove.Enabled = True
            BtnAttribute.Enabled = True

            'don't enable up or down button when there's only one layer
            If LstLayers.Items.Count = 1 Then Exit Sub

            'enable the up button when the selected layer
            'is not the first layer
            If LstLayers.SelectedIndex > 0 Then
                BtnUp.Enabled = True
            Else
                BtnUp.Enabled = False
            End If
            'enable the down button when the selected layer
            'is not the last layer
            If LstLayers.SelectedIndex < _
               (LstLayers.Items.Count -1) Then
                BtnDown.Enabled = True
            Else
                BtnDown.Enabled = False
            End If
        Else
            'Disable layer-specific buttons
            BtnRemove.Enabled = False
            BtnUp.Enabled = False
            BtnDown.Enabled = False
            BtnAttribute.Enabled = False
        End If
End Sub
```

2. The MoveLayer() method of a Map object allows you to rearrange the display order of a layer in a Map object. You will need to know the position the layer is to be moved to and a pointer that points to the layer that is to be moved. Below is the sample code for the Move Down button. The Move Up button uses similar logic.

```
Private Sub BtnDown_Click(ByVal sender As System.Object, ByVal e As
  System.EventArgs) Handles BtnDown.Click
  Dim position As Long = LstLayers.SelectedIndex
  'Declare ArcObjects outside of Try/Catch so we can dispose of them in
      Finally
  Dim pFLayer As IFeatureLayer
  Dim pMap As IMap = My.Document.Maps.Item(0)

  Try
```

```
        pFLayer = pMap.Layer(position)
        'swap the layer listbox
        pMap.MoveLayer(pFLayer, position + 1)
        'refresh the active map and the layer list box
        My.Document.ActivatedView.Refresh()
        RefreshLayerNames()
        'keep the selected item being selected after the move
        LstLayers.SelectedIndex = position + 1
    Catch ex As Exception
        MessageBox.Show("BtnDown_Click Exception: " & ex.Message)
    Finally
        pFLayer = Nothing
        pMap = Nothing
        ' Call garbage collection
        GC.Collect()
        GC.WaitForPendingFinalizers()
    End Try
End Sub
```

Show Fields button

When the Show Fields button is clicked, the names and types of the attribute fields of the selected map layer are displayed in the Attribute list box. Please refer to Week 6's lecture notes for retrieving the name and type information of the fields in an attribute table. Check the notes on the IFields and IField interfaces. After you retrieve the field information, use the code below to add them to the Attribute listBox.

```
        Dim fCount As Long, i As Long
        Dim position As Long = LstLayers.SelectedIndex
        pFLayer = pMap.Layer(position)
        pFields = pFLayer.FeatureClass.Fields
        fCount = pFields.FieldCount - 1
        LstAttribute.Items.Clear()
        For i = 0 To fCount
            aField = pFields.Field(i)
            LstAttribute.Items.Add(aField.Name & ", " & aField.Type)
        Next
```

Summary statistics Button & Attribute List Box

1. The final piece of the LAM program is finding the minimum and maximum of a selected numerical attribute field and calculating its average value.

   You have to check to see if the field the user selected is a numerical field. If yes, then you enable the Summary Statistics button. Disable the Summary Statistics button otherwise. You extracted field type information when creating the Attribute list and stored this type information in the list items. You can get the type information from the list items without accessing the attribute table of the map layer again. Here is how it's done in the SelectedIndexChanged() subprocedure of the Attribute listBox.

Please read the code carefully because it contains important information. For example, you should know that when the value of the field type property is smaller then 4 (i.e., 0, 1, 2, and 3), then the field is a numeric field (see lecture notes for a complete list of field types.)

```
Private Sub LstAttribute_SelectedIndexChanged(ByVal sender As
  System.Object, ByVal e As System.EventArgs) Handles
  LstAttribute.SelectedIndexChanged
    Dim attType As Integer
    Dim attStr As String

    'check the attribute type
    attStr = CStr(LstAttribute.SelectedItem)
    'extract the attribute type information from the string
    attType = Microsoft.VisualBasic.Right(attStr, Len(attStr) - InStr(attStr,
",","))
    'numerical data types are
    'small integer(0), long integer(1), single(2), and double(3)
    If attType <= 3 Then
        BtnSummary.Enabled = True
    Else
        BtnSummary.Enabled = False
    End If
    'reset the text in the textbox
    TxtMin.Text = ""
    TxtMax.Text = ""
    TxtAverage.Text = ""
End Sub
```

2. Once you have a numeric field selected, the next step is to read the values of its records and find the min/max/mean values. Please refer to Week 6's lecture notes to see how to get the values from an attribute table. You will need to use the ArcObjects ICursor and IFeature interfaces for this task.

   You don't have to store the values you read in an array to find the min/max/mean values. The code below is an example of finding/calculating these numbers. You begin by setting the minimum (attMin) to the largest number you can get and the maximum (attMax) to the smallest number. A variable (attSum), initialized to 0, holds the sum of all numbers you retrieve from the attribute table. Then use a loop to compare the minimum and maximum values in each record with the number you retrieve and add the number to attSum.

```
'initialize values
 attMin = 2147483647
 attMax = -2147483648
 attSum = 0

 nRecord = pFLayer.FeatureClass.FeatureCount(Nothing)

 'find min, max, and sum of all values
 For i = 1 To nRecord
      aFeature = pCursor.NextRow
      tempVal = aFeature.Value(LstAttribute.SelectedIndex)
```

```
        If tempVal < attMin Then attMin = tempVal
        If tempVal > attMax Then attMax = tempVal
        attSum = attSum + tempVal
    Next

    'update output textboxes
    TxtMin.Text = Format(attMin, "0.000")
    TxtMax.Text = Format(attMax, "0.000")
    TxtAverage.Text = Format(attSum / nRecord, "0.000")
```

Clear map button

The Clear Map button removes all layers from the map and reset every control on the DockableWindow

```
'clear layers in the active map frame
Dim pMap As IMap = My.Document.Maps.Item(0)
pMap.ClearLayers()
'clear the TOC
My.Document.UpdateContents()
'refresh the map document
My.Document.ActivatedView.Refresh()
'clear attribute and layer lists
LstAttribute.Items.Clear()
LstLayers.Items.Clear()
```

Hint: In one possible solution, the following subroutines contain code:

- BtnAdd_Click()
- BtnAttribute_Click()
- BtnClear_Click()
- BtnDown_Click()
- BtnRemove_Click()
- BtnSummary_Click()
- BtnUp_Click()
- LstAttribute_SelectedIndexChanged()
- LstLayers_SelectedIndexChanged()
- RefreshLayerNames()

Questions:

Q1) What is the easiest way to begin writing code to respond to a ListBox_SelectedIndexChanged eventor a Button_Click event?

Q2)What do we mean when we say that VB .NET arrays are zero based? Give an example of an ArcObjects collection that you worked with in this lab that is also zero based.

Q3) In this lab we used the GxFilterShapefiles class with the IGxDialog so that the user could only select from shapefiles. Review the ESRI API for IGxObjectFilter. What filter class might we use to show only raster files? Both types of datasets?

Q4) What happens if you try to add the same layer twice in your LAM Tool? Provide a few lines of pseudo code that could handle this situation better. For extra credit,
implement your psuedocode in the LAM Tool.

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 comment your code!