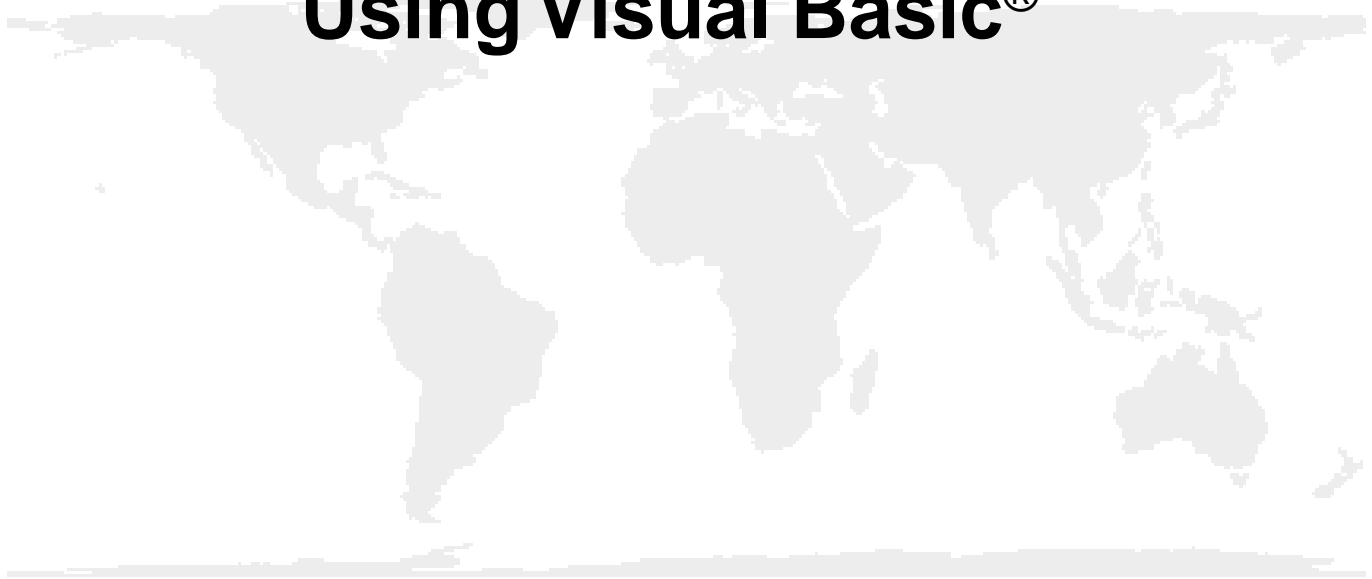


Getting Started with the Map Control Using Visual Basic®



Copyright © 2001 ESRI
All Rights Reserved.
Printed in the United States of America.

The information contained in this document is the exclusive property of ESRI. This work is protected under United States copyright law and the copyright laws of the given countries of origin and applicable international laws, treaties, and/or conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage or retrieval system, except as expressly permitted in writing by ESRI. All requests should be sent to Attention: Contracts Manager, ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

The information contained in this document is subject to change without notice.

U. S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

SAMPLE DATA LICENSE AND DISCLAIMER

The sample data included in this package is the exclusive property and copyright of ESRI, and the respective data publishers. The sample data is protected under United States copyright law and other international copyright treaties and conventions. The sample data is provided under license from each of the respective data publishers for the Licensee's own internal use. Licensee shall not sell, rent, lease, sublicense, lend, assign, time-share, or transfer, in whole or in part, or provide unlicensed third parties access to the sample data or portions of the data, documentation or metadata, any updates, or Licensee's rights under this Agreement. The sample data herein has been obtained from sources believed to be reliable, but their accuracy and completeness, and the opinions based thereon, are not guaranteed. ESRI is not inviting reliance on the sample data, and you should always verify actual data and information. The sample data contained in this package is subject to change without notice. **THE SAMPLE DATA IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.** ESRI shall not be liable for indirect, special, incidental, or consequential damages related to Licensee's use of the sample data, even if ESRI is advised of the possibility of such damage.

ESRI and the ESRI globe logo are trademarks of ESRI, registered in the United States and certain other countries; registration is pending in the European Community. ArcObjects, ArcSDE, ArcInfo, ArcMap, and ArcGIS are trademarks of ESRI. Other companies and products mentioned herein are trademarks or registered trademarks of their respective trademark owners.

Contents

Introduction	5
Loading the Map control	6
Using the Map control	8
Adding pan and zoom controls	10
Adding a toolbar	12
Creating a find tool	15
Handling resize	17
Displaying map layers based on scale	18
Adding a spatial query tool	19
Event tracking	20
Adding layers programmatically	23
Working with RasterLayer objects	24
Using ESRI tools and commands	25
Congratulations	28

Getting Started

IN THIS BOOK

- **Loading the Map control into Visual Basic**
- **Adding simple display tools**
- **Creating basic spatial query tools**
- **Displaying map layers**
- **Tracking events**
- **Adding an image layer**
- **Working with ESRI commands and tools**

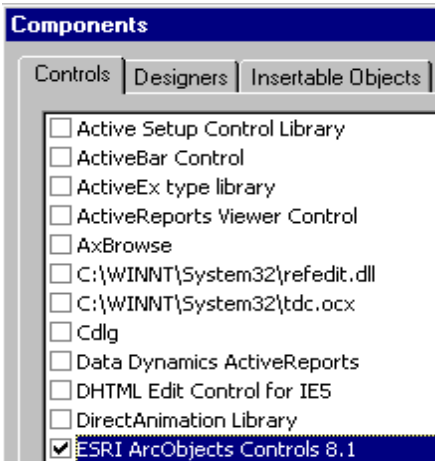
In this introductory document you will use the ESRI® Map control and Microsoft® Visual Basic® to build an application that uses maps. Along the way you will learn how to

- Display a map with multiple layers.
- Control panning and zooming.
- Create a Toolbar control.
- Display map layers based on scale.
- Perform spatial and logical queries.
- Draw simple graphics on the map.
- Display features with thematic renderers.
- Dynamically display real-time event tracking.
- Programmatically add data to a map.
- Add Identify and Query commands from the sample AfCommandsVB library.

Note: If you accepted the defaults when installing ArcObjects™, the geographic data this tutorial refers to can be found in C:\Arcexe81\ArcObjects Developer Kit\Samples\Data\Usa. The bitmaps you can use are in the C:\Arcexe81\Bin\Icons folder.

Loading the Map control

Start Visual Basic and select New project from the dialog box. Select Components from the Project menu or press Ctrl+T.



Tip: You can also add controls by selecting Components from the Project menu or by pressing Ctrl+T.

Find ESRI ArcObjects Controls 8.1 in the list of available controls and check the box beside it. Click OK to close the dialog. Notice that a new tool appears in the Visual Basic Toolbox. This new tool is the ArcObjects Controls Version 8.1 Map control.

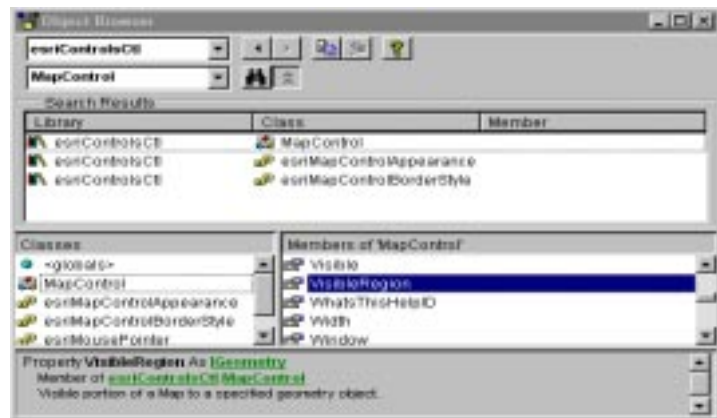


ESRI Map Control 8.1

Getting help

The Map control is an ActiveX control that is provided by ESRI as part of the ArcObjects development toolkit.

To find out about the available methods, properties, and events that are provided with the Map control, click the Object Browser button in the Visual Basic toolbar (or hit F2). Pull down the Project/Library box and choose “esriControlsCtl”.



The esriControls objects and constants are listed in the Classes bottom left-hand list.

Click on the Map control object in the list, and its properties and methods are listed in the Members list to the right. Likewise, click on a method in the right-hand list to see the

Tip: The simplest way to get help is to select the Map control and press F1.

Using the Map control

The Map control allows you to load map data from a variety of data sources including ArcSDE™ layers, ArcInfo™ coverages, shapefiles, and raster files. You can also load ArcMap™ documents (*.mxd) into your Map control. Loading an ArcMap document into the Map control will add each of the referenced layers into the control, complete with saved definitions such as extents and renderers. You can also load individual layer files (*.lyr) in this way.

The Map control also provides you with a number of helper methods such as TrackRectangle and AddLayer. We shall use some of these as we progress through this book. The Map control also has properties that expose references to the internal interfaces used by the Map control such as the IMap, IActiveView, and ILayer interfaces. Using these properties will give you a “hook” into the rich functionality available within the ArcObjects Object Model.

Add the Map control to the form

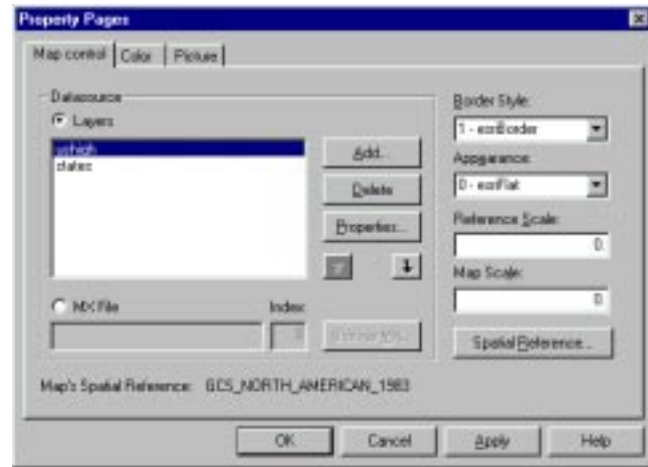
1. Double-click the MapControl button in the toolbox to add a new map to the form.
2. Resize the map to fill the form.



Select the data to display on the map

You can specify the data that is displayed on the map by setting properties in the MapControl's Property Page.

1. Right-click the mouse on the map to display the context menu.
2. Choose Properties to display the Property Pages.
3. Click the Add button and locate the folder containing the states sample data.
4. Click the states.shp file and then click Add.
5. Add the file ushigh.shp in the same manner.
6. Use the Up and Down arrows to ensure that the ushigh layer is at the top of the list.



You can also specify how the data is displayed on the Map by using this Property Page.

Set properties for the layers

1. Click the ushigh layer in the Layers list and then click Properties.

2. Modify the ushigh layer symbology:

Click on the Symbology tab.

Click the Symbol button.

Select a red color and click OK to dismiss the Layer Properties dialog.

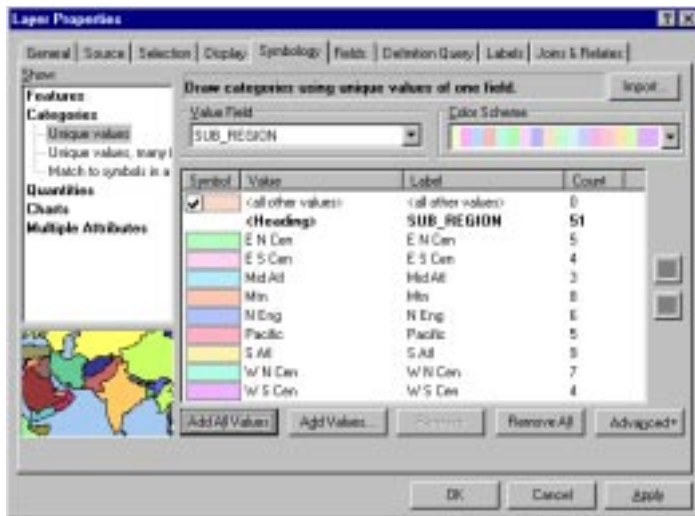
3. Now define a renderer for the States layer:

Select “States” from the list of layers.

Click on the Symbology tab.

In the Symbology tab select Categories and then click on Unique values.

Make SUB_REGION the Value field and click on the “Add All Values” button.



You can double-click on the colors to display the Color dialog box.

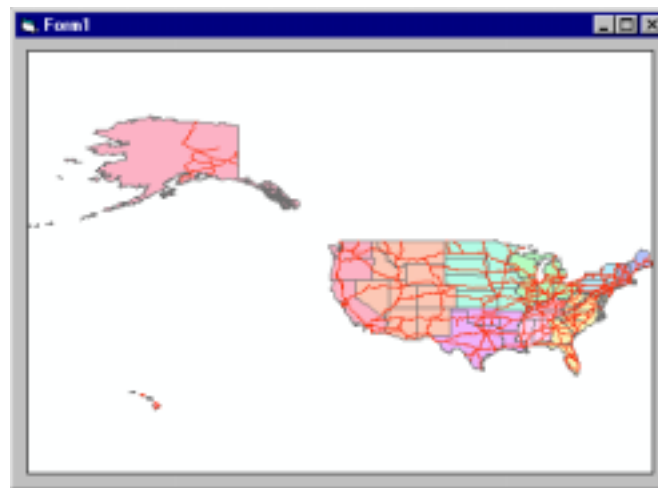
4. Click OK to close the Property Page.

Save the project

1. Click the File menu and then click Save Project.
2. Browse to a suitable folder, then in the File Name box type StarterMap.frm.
3. Click Save.
4. In the second Save dialog, type StarterMap.vbp in the File Name box.
5. Click Save.

Test your application

1. Click the Run button in the Visual Basic toolbar.
2. To stop running your application and return to design mode, click the Stop button in the Visual Basic toolbar.



Adding pan and zoom controls

At this point your application can display the map at its full extent. In this section you will add some simple pan and zoom controls that your application will activate in response to mouse clicks inside the map. You will write some code that the application will execute in response to the `OnMouseDown` event on the map.

Respond to the `MouseDown` event

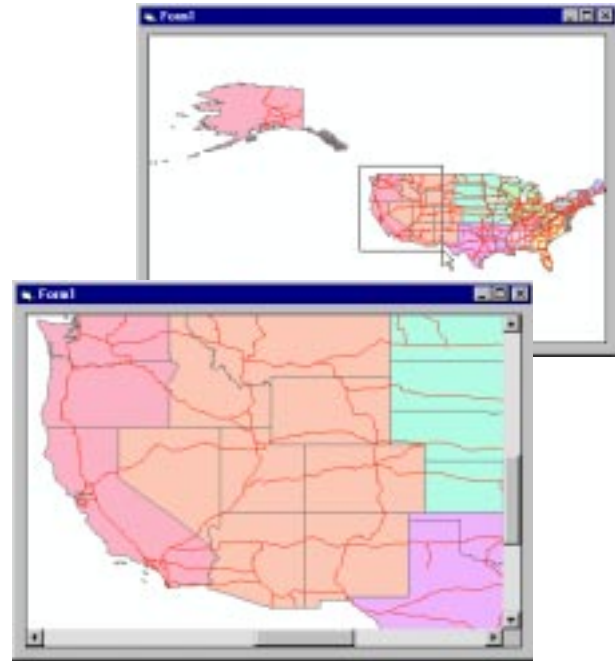
1. Double-click on the Map control to display the Visual Basic code window.
2. Add code to `MapControl1`'s `OnMouseDown` procedure.

```
Private Sub MapControl1_OnMouseDown(  
    ByVal button As Long, _  
    ByVal shift As Long, _  
    ByVal x As Long, _  
    ByVal y As Long, _  
    ByVal mapX As Double, _  
    ByVal mapY As Double)  
    MapControl1.Extent = _ MapControl1.TrackRectangle  
End Sub
```

`TrackRectangle` is a method that applies to the Map control. It tracks the movement of the mouse while the user keeps the mouse button pressed, rubber-banding a rectangle at the same time. When the user releases the mouse button, the `TrackRectangle` method returns an `IEnvelope` object that the application then assigns to the `Extent` property of the map, causing the map to be redrawn with a new map extent.

Test your change

1. Click the Run button in the Visual Basic toolbar.
2. Click the map with the left mouse button and drag out a rectangle.



3. Release the mouse button and notice that the map is redrawn at the location you specified.
4. Click the Stop button in Visual Basic to return to design mode.

Add panning

1. Double-click the map to display the Visual Basic code window again.
2. Change the code for the `OnMouseDown` event.

```

Private Sub MapControl1_OnMouseDown(
ByVal button As Long, _
ByVal shift As Long, _
ByVal x As Long, _
ByVal y As Long, _
ByVal mapX As Double, _
ByVal mapY As Double)
If Button = vbLeftButton Then
    MapControl1.Extent = _
MapControl1.TrackRectangle
ElseIf Button = vbRightButton Then
    MapControl1.Pan
End If
End Sub

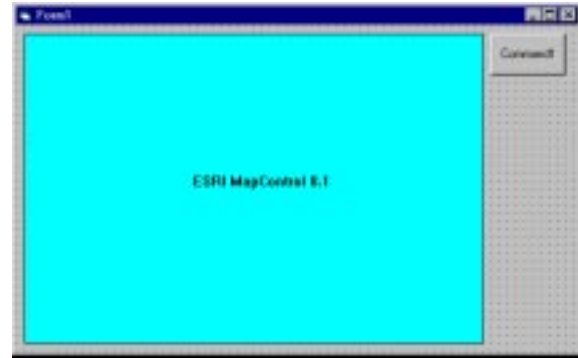
```

If the Button parameter is equal to vbLeftButton when the OnMouseDown event occurs, then the zooming code from the previous step will be executed. Otherwise, the code will call another Map control method, Pan. If the user clicks the left mouse button, the Button parameter will be vbLeftButton. If the user clicks the right mouse button, the value of Button will be vbRightButton.

Add a FullExtent button

Your application now supports panning and zooming but, once the user has zoomed into the map, there is no way to get back to the full extent again. In this section you will add a button to the form that zooms the map to the full extent.

1. Double-click the CommandButton button in the Toolbox to add a button to the form.
2. Move the button to the upper right of the form.
3. Press F4 to display the Properties window.
4. Click in the Caption box and type Full Extent to change the button's caption.



5. Resize the Map control so that it is not obscured by the button.
6. Double-click the Full Extent button to display the code window.
7. Add code for the Click event.

```

Private Sub Command1_Click()
    MapControl1.Extent = MapControl1.FullExtent
End Sub

```
8. Finally, when you are in Design mode, use the Visual Basic Properties box to change two properties of the Map control—set ShowScrollBars to False and choose a color for MapControl's BackColor property.

The FullExtent property of the Map returns a reference to an IEnvelope that defines the bounding box of *all* the layers in the Map.

Test your change

1. Click the Run button in the Visual Basic toolbar.
2. Click the map with the left mouse button and drag out a rectangle.
3. Release the mouse button to redraw the map.

- Click the map with the right mouse button and drag to pan the map.
- Release the mouse button to redraw the map.
- Click the Full Extent button to redraw the map at the full extent.

Save the project

- Click the Stop button in the Visual Basic toolbar to return to design mode.
- Click the Save Project button in the Visual Basic toolbar to save your changes.

Adding a toolbar

Visual Basic includes a Toolbar control that can be used in conjunction with an ImageList control to display a collection of buttons at the top of a form. We shall use the Visual Basic Toolbar for this starter application. The current application's pan and zoom functionality is somewhat hidden from the user. In this section, you will create a toolbar with Pan and Zoom buttons.

Adding a toolbar

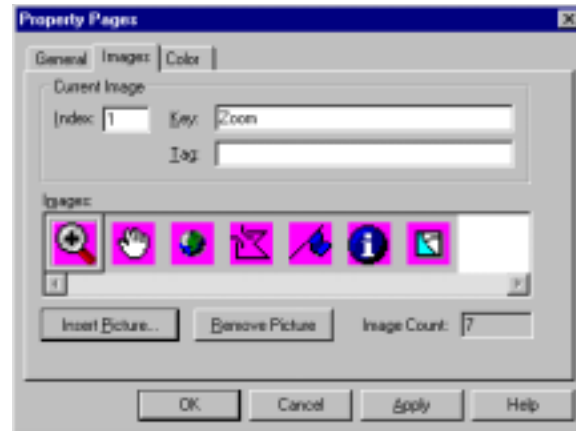
- Delete the Full Extent button from the form.
- Bring up the Components dialog by pressing Ctrl+T. Select Microsoft Windows Common Controls. You will notice new tools are added to your toolbox.
- Double-click the Toolbar button in the toolbox to add a Toolbar control to the form.
- Double-click the ImageList button in the toolbox to add an ImageList control to the form.
- Resize the map so that it is not obscured by the toolbar.

The ImageList control may obscure the map; however, this is not a problem because the ImageList control will not be visible when your application is running.



Adding images to the ImageList control

- Right-click the ImageList control to display the context menu.
- Click Properties to display the Property Pages.
- Click the Images tab.



4. Click Insert Picture and locate the folder that contains the sample bitmaps. This should be the Icons directory within the ArcGIS Bin folder.
5. Click the zoom_in_tool_1.bmp file and then click Open.
6. Add a descriptive Key, for example, "Zoom". This will correspond with Keys that will be used when adding buttons to the Toolbar control.
7. Add the files pan_1.bmp, globe_3.bmp, select_by_polygon.bmp, flag.bmp, information.bmp, and selection_1.bmp in the same manner. The Keys should be "FullExtent", "SelectByPolygon", "TrackEvent", "Identify", and "Query", respectively.

Set the MaskColor of the ImageList

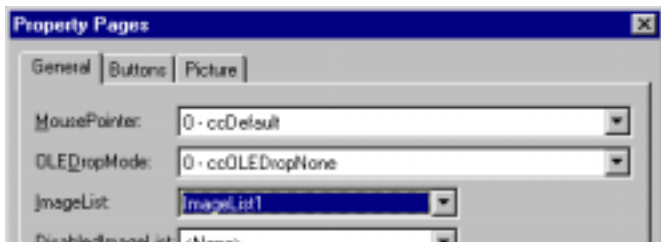
Setting the MaskColor property of an ImageList control specifies a color that will act as a mask for any images contained by the control. The mask color will not be drawn, resulting in an image with a transparent background.

1. Click the Color tab.
2. In the Properties list, click MaskColor.
3. Click "Magenta" in the Color Palette list and then click OK to dismiss the Property Pages.

Associate the ImageList with the toolbar

You can associate the Toolbar control with an ImageList control to provide the graphic images for the buttons.

1. Right-click the Toolbar control to display the context menu.

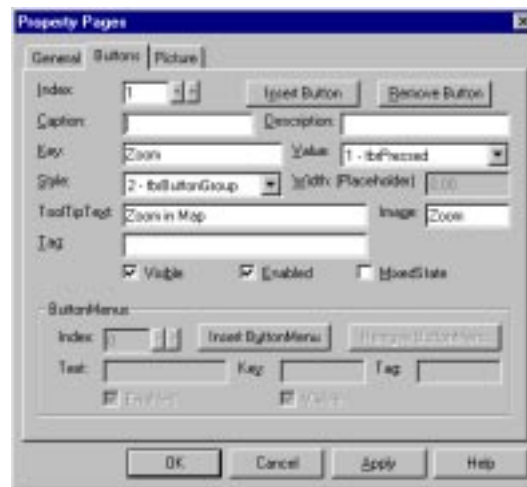


2. Click Properties to display the Property Pages.
3. In the ImageList box, click the arrow and then click ImageList1. This associates the toolbar with the ImageList control.

Adding buttons to the Toolbar control

In this section you will add seven buttons to the toolbar.

1. In the Toolbar Control Properties dialog, click the Buttons tab, then click Insert Button.
2. Set the button's style to ButtonGroup, its Image to "Zoom", its Key to "Zoom", and its Value to Pressed. Add suitable Tool Tips.
3. Add a second button and set its Style to ButtonGroup, its Image to 2, and its Key to "Pan".
4. Add five more buttons and associate them with the remaining images in the ImageList, setting their Style to ButtonGroup (add Separators if you wish). The Keys should be "FullExtent", "SelectByPolygon", "TrackEvent", "Identify", and "Query", respectively.
5. Click OK to dismiss the Property Pages.



Change the OnMouseDown event

1. Double-click the map to display the Visual Basic code window.
2. Completely replace the code in the MapControl1's OnMouseDown event handler with the following:

```
With Toolbar1.Buttons
    If .Item("Zoom").Value = tbrPressed Then
        MapControl1.Extent = MapControl1.TrackRectangle
    ElseIf .Item("Pan").Value = tbrPressed Then
        MapControl1.Pan
    End If
End With
```

Selecting the first button in the toolbar sets the mouse to be a zoom tool; selecting the second button lets you use the mouse to pan.

The Toolbar ButtonClick Event Handler

In this section you will implement the handler for the ButtonClick event that is generated whenever a click occurs on a button in the toolbar.

We will add code to change the MapControl's MousePointer depending on what tool has been selected, providing useful feedback to the user. We shall also reimplement the Full Extent button that you deleted.

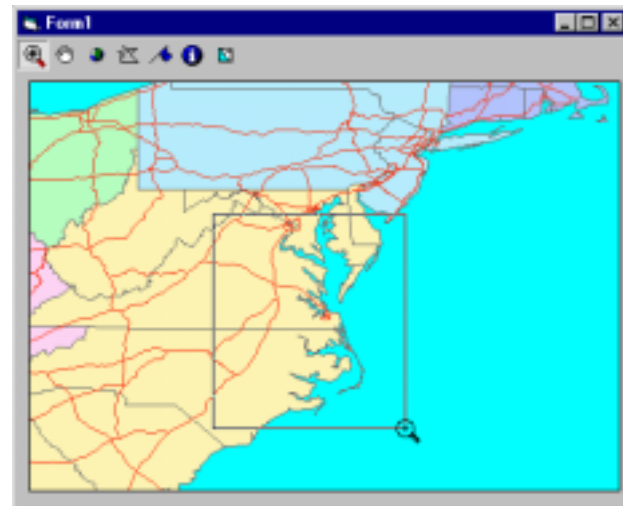
Within the event handler we will be using a Case statement with switches, depending on the Key value of the button that you defined earlier.

1. Double-click the toolbar to display the code window.
2. Add the following code to Toolbar1's ButtonClick event.

```
Private Sub Toolbar1_ButtonClick(ByVal Button _
    As MSComctlLib.Button)
    With MapControl1
        Select Case Button.Key
            Case "Zoom"
                .MousePointer = esriPointerZoomIn
            Case "Pan"
                .MousePointer = esriPointerPan
            Case "FullExtent"
                .MousePointer = esriPointerDefault
                .Extent = MapControl1.FullExtent
        End Select
    End With
End Sub
```

Test and save your changes

1. Click the Run button in the Visual Basic toolbar.
2. Drag a rectangle to zoom in.
3. Click the pan button in your application's toolbar.



- Click somewhere on the map and then drag to pan.
- Click on the Full Extent button (the globe) in your application's toolbar.

Creating a find tool

In this section you will add additional controls to your application to implement a simple function for locating a state by name. Once the State is located you will zoom to it.

Add controls to the form

- Double-click the Label button in the toolbox to add a label to the form.
- Reposition the label in the lower left corner of the form.
- Display the Properties window by pressing F4 and set the caption of the label to be 'State:'.



- Double-click the TextBox button in the toolbox to add a TextBox to the form and position it next to the label.
- Clear the Text property of the TextBox using the Properties window.
- Resize the map so that it is not obscured by the new controls.

Attach code to the TextBox

You will use the text the user types into the TextBox to perform a logical query.

- Double-click the TextBox to show the code window.
- Add code to Text1's KeyPress procedure by selecting KeyPress from the right-hand dropdown list:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii = vbKeyReturn Then
        ' Find the States layer
        Dim i As Integer
        Dim pFeatLyr As IFeatureLayer
        For i = 0 To MapControl1.LayerCount - 1
            Set pFeatLyr = MapControl1.Layer(i)
            If pFeatLyr.Name = "states" Then
                ' Found it and exit loop
                Exit For
            End If
        Next i
        ' Create a string to use in the query
        Dim queryStr As String
        queryStr = "STATE_NAME = '" & Text1.Text & "'"

        ' Create the query filter
        Dim pQueryFltr As IQueryFilter
        Set pQueryFltr = New QueryFilter
        pQueryFltr.whereClause = queryStr
    End If
End Sub
```

```

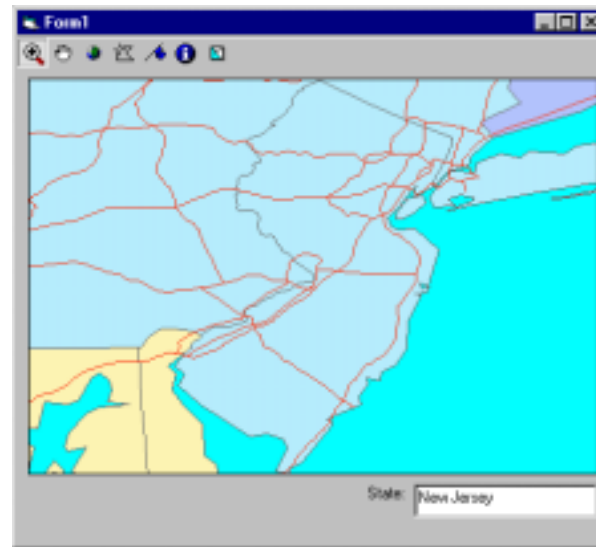
' Perform the selection
Dim pFeatSeln As IFeatureSelection
' QI for IFeatureSelection from the
' IFeatureLayer
Set pFeatSeln = pFeatLyr
pFeatSeln.SelectFeatures _
pQueryFltr, esriSelectionResultNew, False
' Get the selection set
Dim pSelSet As ISelectionSet
Set pSelSet = pFeatSeln.SelectionSet

' Get the cursor from the selection set
Dim pFeatCursor As IFeatureCursor
pSelSet.Search Nothing, True, pFeatCursor
' Assume only one feature
Dim pFeature As IFeature
Set pFeature = pFeatCursor.NextFeature
If Not pFeature Is Nothing Then
' Get the extent of the selected feature
Dim pExtent As IEnvelope
Set pExtent = pFeature.Shape.Envelope
' And set the Map control's extent
MapControl1.Extent = pFeature.Shape.Envelope
End If
End If
End Sub

```

Test and save your changes

1. Run your application.
2. Type the name of a state, for example, New Jersey, into the TextBox.
3. Press the Enter key.
4. When you are finished running your application, click the Stop button in the Visual Basic toolbar and Save your application.



The code locates the States layer and defines a simple SQL query expression using the value of the Text property of the TextBox. The States layer's IFeatureSelection uses this query to search for the specified value. You can obtain an IFeatureCursor that contains all features from the IFeatureSelection. Here we assume the cursor contains only one feature, from which we will derive the geometry and its extent envelope. We will use this envelope to reset the MapControl's Extent.

Handling resize

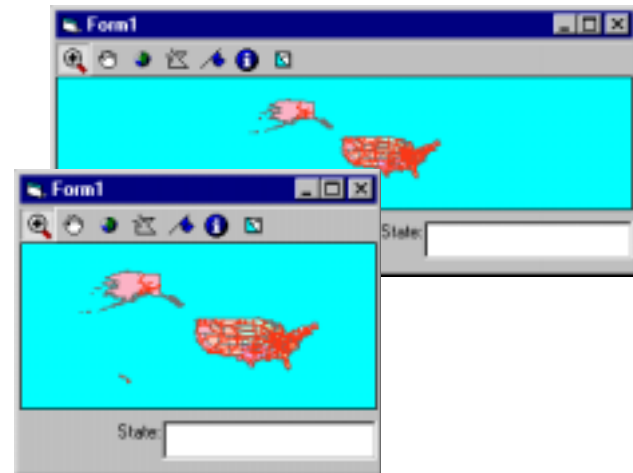
When you run your application and resize the form, you will notice that the map is not automatically resized.

Respond to the Resize event

1. Double-click the form to display the code window.
2. Add code to the form's Resize procedure by selecting Resize from the right-hand dropdown list:

```
Private Sub Form_Resize()  
    If (ScaleWidth <> 0) Then  
        ' y coord for Text and Label  
        Dim yFind As Integer  
        ' a constant spacing  
        Dim space As Integer  
        space = Text1.Top - (MapControl1.Top _  
        + MapControl1.Height)  
        yFind = ScaleHeight - Text1.Height - space  
  
        ' x coord for Text and Label  
        Dim xFind As Integer  
        xFind = ScaleWidth - Text1.Width  
  
        Dim mapTop As Integer  
        mapTop = Toolbar1.Top + Toolbar1.Height  
        Dim mapHeight As Integer  
        mapHeight = yFind - space - mapTop  
        If (mapHeight > 0) Then  
            ' move all the controls  
            Text1.Move xFind, yFind  
            Label1.Move xFind - _  
            Label1.Width - 20, yFind  
            MapControl1.Move 0, mapTop, _
```

```
        ScaleWidth, mapHeight  
    End If  
End If  
End Sub  
3. Double-click the form to show the code window.  
4. Add code to the form's Load procedure.  
Private Sub Form_Load()  
    Form_Resize  
End Sub
```



When the user resizes the form, the code resizes the controls using the Move method.

Notice that when you run your application it redraws the map twice initially. This is due to the fact that controls on the form are initially displayed using the size and position specified during design time. To fix this problem, you will resize the controls when the form is initialized. You have already written the code to resize the controls, so you just need to call the procedure.

Displaying map layers based on scale

In this section you will add a new layer to your map and add code that controls whether or not that layer is visible at a given time.

Add another layer

1. Right-click the map to display the context menu. Click Properties to show the Property Pages.
2. Click the Add button and locate the folder where the sample data is stored.
3. Click the counties.shp file and then click Add.
4. Use the arrows to put the ushigh layer at the top of the list and the counties in the middle.
5. Use the Layer Properties to render the counties layer:
Click on the Symbology tab.
Select Categories, Unique Values.
Make STATE_NAME the Value Field and click on “Add All Values” button.
6. Click OK to dismiss the Layer Properties dialog.
7. Click OK to dismiss the Property Page.

If you run your application now, you will notice that it displays every county in the United States. At the full extent there is no need to display that much detail. You can use the MinimumScale and MaximumScale properties on the ILayer interface to selectively make the Counties and States layers visible or invisible, depending on the current scale of the map.

Setting scale breaks

1. Display the code window.
2. Add the following code to the Form_Load procedure (after the Form_Resize):

```
Dim pLayer As ILayer
Dim i As Integer
' Iterate through all layers
For i = 0 To MapControl1.LayerCount - 1
    Set pLayer = MapControl1.Layer(i)
    ' Note case sensitivity
    If UCase(pLayer.Name) = "COUNTIES" Then
        pLayer.MaximumScale = 0#
        pLayer.MinimumScale = 15000000#
    ElseIf UCase(pLayer.Name) = "STATES" Then
        pLayer.MaximumScale = 14999999#
        pLayer.MinimumScale = 0#
    End If
Next i
```

Because this code executes in the application startup phase, the scale breaks are set up once. If the code were added in response to the OnBeforeDraw event, the layers would be made visible or invisible just before each drawing operation. This may unnecessarily invalidate any drawing caches, which in turn will make the redraw inefficient. Think carefully about what code you place in the OnBeforeDraw event.

Test and save your changes

1. Run your application. Notice that it does not draw the Counties layer.
2. Zoom into New England and the Counties layer becomes visible.



3. Click the FullExtent button and the Counties are no longer visible.



Adding a spatial query tool

In this section you will add a new tool to the toolbar that performs spatial queries on the map and also add code that draws the results. We have already added the button to the Toolbar, so we just need to add code to implement its functionality. We will use the SelectByPolygon button, the fourth button on the toolbar. When the SelectByPolygon button is clicked, we will handle the OnMouseDown event differently, calling a procedure called SearchShape to actually perform the spatial selection.

Implement the SearchShape procedure

```
Private Sub SearchShape()  
Dim pSearchShape As IPolygon  
' Create the search shape  
Set pSearchShape = MapControl1.TrackPolygon  
' Do the actual selection  
With MapControl1  
    .Map.ClearSelection  
    .Refresh esriViewGeoSelection  
    .Map.SelectByShape pSearchShape, Nothing, False  
    ' And refresh the map  
    .Refresh esriViewGeoSelection  
End With  
End Sub
```

Call the SearchShape procedure

1. Modify MapControl1's OnMouseDown procedure and insert the following:

```
ElseIf .Item("SelectByPolygon").Value =_
tbrPressed Then
    SearchShape
```

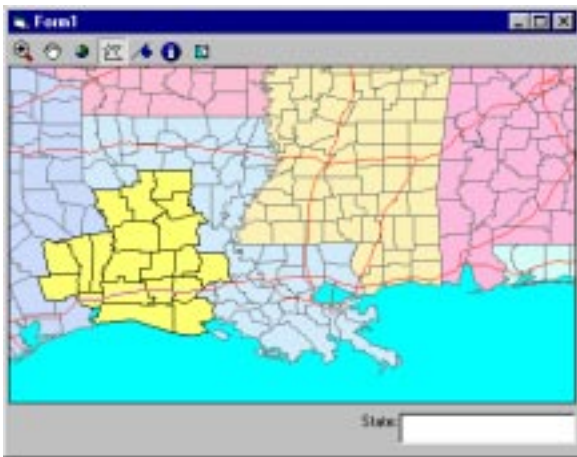
2. Finally, modify the Toolbar_ButtonClick procedure to change the MapControl's Mousepointer.

```
Case "SelectByPolygon"
    .MousePointer = esriPointerCrosshair
```

Test and save your changes

1. Run your application and zoom into an area so that the Counties layer becomes visible.
2. Click the spatial query tool, then digitize a polygon. Double-click to end. You should see that counties that intersect with the polygon are highlighted.

Optionally, from the Property Pages, display the Layer Properties dialog to modify the default Selection Symbol for each layer. Set it to a yellow solid fill.



Event tracking

Some applications must display geographic entities on top of the map, especially if those entities have a tendency to move.

For example, a vehicle tracking system would want to display vehicles on the map at the appropriate locations and update those locations over time without redrawing all the underlying layers of the map each time a vehicle changes location.

In this section, you will add event tracking layer elements to your application to facilitate this requirement. We have already added the required toolbar button (the Flag at toolbar button number five), and now we shall implement the tool.

Implement the Event tool

1. Add a private member variable to the General section of the Form's code window. This will be an ElementCollection that will be used to hold your events.

```
Private m_pElementCollection As IElementCollection
```

2. In the Form_Load method, initialize the collection:

```
Set m_pElementCollection = New ElementCollection
```

3. Modify the Toolbar1_ButtonClick procedure to change the MapControl's Mousepointer.

```
Case "TrackEvent"
```

```
MapControl1.MousePointer = esriPointerHotLink
```

4. Modify the MapControl1's OnMouseDown procedure to call AddGeoEvent, passing in the mapX and mapY coordinates. These are already in Map units.

```
ElseIf .Item("TrackEvent").Value = tbrPressed Then
    AddGeoEvent mapX, mapY
```

Implement the AddGeoEvent procedure

The AddGeoEvent procedure adds a new element where the mouse click occurred. Add the following code to the General section of the Form's code window.

```
Private Sub AddGeoEvent(mapX As Double, mapY As Double)
' Make the point
Dim pPoint As IPoint
Set pPoint = New Point
pPoint.PutCoords mapX, mapY
' Make the element
Dim pElement As IElement
Set pElement = New MarkerElement
pElement.Geometry = pPoint
' Add the element to the collection
m_pElementCollection.Add pElement
' And finally redraw the map
MapControl1.Refresh esriViewForeground
End Sub
```

Drawing the events

After the layers draw, we iterate through the collection of events to draw each, in turn, using the MapControl's DrawShape method. Add this code to the OnAfterDraw Event procedure.

```
Private Sub MapControl1_OnAfterDraw(ByVal display As
esriCore.IDisplay, ByVal phase As
esriCore.esriViewDrawPhase)
If (phase = esriViewForeground) Then
Dim pElement As IElement
Dim i As Integer
For i = 0 To m_pElementCollection.Count - 1
m_pElementCollection.QueryItem i, pElement
MapControl1.DrawShape pElement.Geometry
```

```
Next
End If
End Sub
```

Test and save the Event tool

1. Run your application.
2. Zoom into an area.
3. Click the event tool, then click in the map to add events.
4. Click the Stop button on the Visual Basic toolbar and save your project.

Add a timer to your form

To trigger the movement of the events, a Timer control will be used. The Visual Basic function 'Rnd' is used to generate a random number to move the events in a random manner.

1. Double-click the Timer button in the toolbox to add a timer to the form. The Timer control will not be visible when your application is running.
2. Double-click the timer to display the code window.
3. Add code to the timer procedure.

```
Private Sub Timer1_Timer()
Dim maxDist As Double
Dim nEventCount As Integer
Dim pt As IPoint
maxDist = MapControl1.Extent.Width / 20
nEventCount = m_pElementCollection.Count
Dim newX As Double
Dim newY As Double
Dim i As Integer
' If the collection is not empty
If (nEventCount > 0) Then
Dim pGeometry As IPoint
```

```

Dim pElement As IElement
' Iterate the collection
For i = 0 To nEventCount - 1
    m_pElementColn.QueryItem i, pElement
    Set pGeometry = pElement.Geometry
' Generate a new random position
    newX = pGeometry.x - (maxDist * (Rnd - 0.5))
    newY = pGeometry.y - (maxDist * (Rnd - 0.5))
' And update the element
    pGeometry.PutCoords newX, newY
    pElement.Geometry = pGeometry
Next i
MapControl1.Refresh esriViewForeground
End If
End Sub

```

Add a CheckBox to your form

To turn the timer on or off, you will add a CheckBox control to your application.

1. Double-click the CheckBox button in the toolbox to add a CheckBox to the form.
2. Move the CheckBox to the lower left corner of the form.
3. Open the Properties window and set the Caption to 'Data Collection'.
4. Double-click the CheckBox control to open the code window.
5. Add code to Check1's Click procedure.

```

Private Sub Check1_Click()
    Timer1.Interval = Check1.value * 500
End Sub

```

Modify Form_Resize

Modify the Form_Resize method to move the CheckBox when the form is resized. Just before the Text1 and Label1 controls are moved, insert the code for moving the Check1 control.

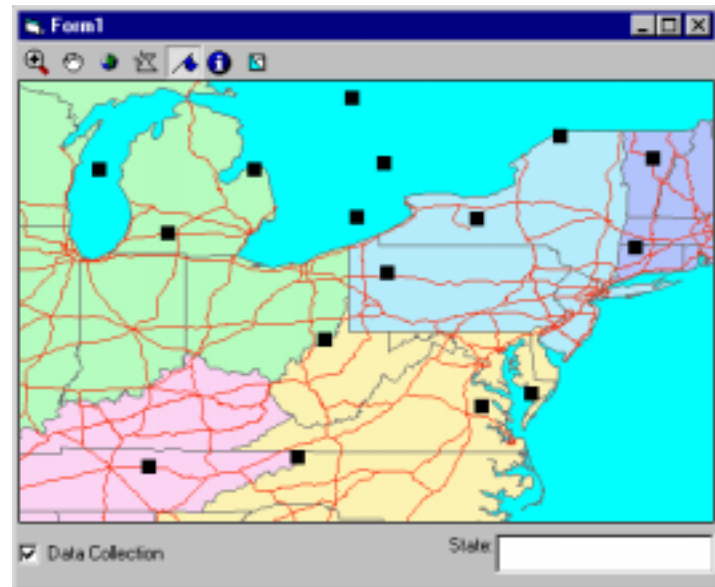
```

' Move all the controls
Check1.Move 0, yFind
Text1.Move xFind, yFind

```

Test your changes

1. Run your application.
2. Zoom into an area.
3. Click the event tool, then click in the map to add events.
4. Click the Data collection check box. Notice that the events start moving randomly on top of the map.
5. Click the check box again to stop the events.



Adding layers programmatically

In each of the previous sections, you have worked with Layer objects that were specified interactively using the MapControl's Property Pages. In this section, you will add code to your application that creates Layer objects programmatically.

There are a number of methods that you can use to programmatically add layers to your Map control:

LoadMxFile—This method will load an ArcMap document (an MXD file). If the ArcMap document contains multiple maps, then it is possible to state which one you want to load by passing in an index. Each map can contain multiple layers, and all layer properties (such as renderers) will be retained. Note that if you want to use an ArcMap document to add layers to your Map control, then this method should be called *before* any additional layers are added.

AddLayerFromFile—Using this method you can add a layer that has previously been saved to a layer file (lyr). You can also optionally specify an index to force the new layer's position within currently loaded layers. All symbology for the layer that was defined within ArcMap is also honored within the Map control.

AddShapeFile—This method allows you to add shapefiles by specifying the path and the shapefile name. The new layer is added on top of any previously loaded layers. Default symbology is provided by ArcObjects.

AddLayer—This method takes as parameters a reference to an ILayer and an optional index value. This allows you to create and load layers that come from different data formats—for example, Coverages, TIN, Raster, or ArcSDE. See the *ArcObjects Developers* online reference, the *Exploring ArcObjects* book, or the *ArcObjects Class Help* online for further details on creating layers.

The example code that follows loads three layers using two of the above methods. Ensure that you change the paths to the data to match your installation.

Remove the existing layers

1. Right-click the mouse on the map to display the context menu.
2. Choose Properties to display the Property Pages.
3. Click on the ushigh layer, then click Delete to remove the layer.
4. Remove counties and states in the same manner, then click OK.

Add a procedure that will initialize the map

1. Double-click the form to display the code window.
2. In the General section, declare a procedure.

```
Sub InitializeMap()  
' Change the paths to reflect your installation  
Dim pFactory As IWorkspaceFactory  
Dim pWorkspace As IFeatureWorkspace  
Dim pFeatLayer As IFeatureLayer  
' Create a new ShapefileworkspaceFactory object  
Set pFactory = New ShapefileworkspaceFactory  
Set pWorkspace = pFactory.OpenFromFile("C:\Data\USA", 0)  
' Create a new FeatureLayer and assign a shapefile to it  
Set pFeatLayer = New FeatureLayer  
Set pFeatLayer.FeatureClass =  
pWorkspace.OpenFeatureClass("States")  
pFeatLayer.Name = pFeatLayer.FeatureClass.AliasName  
' Add the FeatureLayer to the map  
With MapControl1  
.AddLayer pFeatLayer  
.AddShapeFile "C:\Data\USA", "counties"
```

```
.AddShapeFile "C:\Data\USA", "USHigh"
End With
End Sub
```

Modify the Form_Load procedure

1. Immediately after the call to Form_Resize and before the section of code that sets the minimum and maximum display scales for the layers, insert the following line:

```
InitializeMap
```

2. To ensure that the MapControl's MousePointer correctly reflects the startup status of the Toolbar, insert the following line after the call to InitializeMap:

```
MapControl1.MousePointer = esriPointerZoomIn
```

Test and save your changes

1. Run your application. The map should appear as before, but default colors and renderers are used and therefore may be different than the colors you previously selected in the Layer Properties dialog box.
2. Click the Stop button in the Visual Basic toolbar and save your project.

Working with RasterLayer objects

In each of the previous sections, you worked with MapLayer objects based on vector data sources. In this section, you will see how to add layers to your map that are based on images. ArcObjects allows you to use a wide range of image types as RasterLayers including such common image types as windows bitmaps (.bmp), tagged image file format (.tiff), and CompuServe bitmaps (.gif).

Select an image layer to display on the map

You can specify an image to display as a RasterLayer by setting properties in the MapControl's Property Pages.

1. Right-click the mouse on the map to display the context menu.
2. Choose Properties to display the Property Pages.
3. Click the Add button and then select "Raster datasets" from the "Show of type" combo box.
4. Navigate to a data folder that contains raster datasets. If you have the "ESRI Data & Maps" CD, you can use the sample image "wsiearth.tif".
5. Click on the raster dataset file, then click Add.
6. If you still have other datasets listed in your Property Pages, use the arrow buttons to ensure that the raster data is at the bottom of the list.

Test your changes

1. Click the Run button in the Visual Basic toolbar.
2. To stop running your application and return to design mode, click the Stop button on the Visual Basic toolbar.



Adding a RasterLayer programmatically

Previously, you added feature layers programmatically; now you will add a RasterLayer programmatically.

Remove the existing layers

1. Right-click the mouse on the map to display the context menu and click on Properties to display the Property Pages.
2. Select wsiearth.tif (or the raster dataset you selected), then click Delete.

Modify the procedure that initializes the map

1. Double-click the form to display the code window.
2. Modify the InitializeMap procedure by inserting the following after all existing code:

```
' Add RasterLayer to the map underneath all  
' existing layers  
Dim pRasterLayer As IRasterLayer  
Set pRasterLayer = New RasterLayer  
pRasterLayer.CreateFromFilePath _"E:\World\wsiearth.tif"  
' Note that we are passing in an index value  
MapControl11.AddLayer pRasterLayer,  
_MapControl11.LayerCount
```

Test and save your changes

1. Click the Run button in the Visual Basic toolbar.
2. Click the Stop button in the Visual Basic toolbar and save your project.

Using ESRI tools and commands

The ArcGIS™ installation provides you with a set of sample commands and tools that you can use to customize ArcMap or in conjunction with your Map control. These tools implement ICommand and/or ITool, providing functionality to identify and select features and to produce attribute reports, editing commands (including StartEditing, StopEditing, Undo, Redo), and tools to add and reorganize layers.

In the Bin subfolder of your ArcGIS installation, you will see two DLLs called “AfCommands” and “AfCommandsVB”. These two DLLs contain the library of supplied sample tools that can be used directly with the Map control. The source code for the sample tools is available as part of the ArcObjects Developers Kit. The following section illustrates how to use two of these commands to provide an Identify tool and a Query tool.

Adding an Identify tool

Add the “ESRI AF Commands (VB)” reference to your VB project.

1. Pull down the Project menu and select References.
2. In the list of available references, find and check “ESRI AF Commands(VB)”.
3. Ensure that the “ESRI Object Library” appears above “ESRI AF Commands(VB)”.

We have already added an Identify button on the toolbar. Now we need to make three modifications. Firstly, we define a Private member variable to hold our Command. The second change is in the Form_Load event where we instantiate the private member variable, and the final change involves handling the clicking of the toolbar.

1. Firstly, add a private member variable to your project by adding the following code to the Form's code window General section.

```
Private m_pCommand As ICommand
```

2. Immediately after the call to InitializeMap, insert the following lines to create an Identify tool and pass it a reference to the Map control.

```
Set m_pCommand = New AfCommandsVB.Identify
' Pass the Map control as the hook
m_pCommand.OnCreate MapControl1.Object
```

3. Modify the Toolbar1_ButtonClick procedure to associate the Map control with the Identify command using the CurrentTool property.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As
MSComCtlLib.Button)
```

```
Set MapControl1.CurrentTool = Nothing
With MapControl1
```

```
Select Case Button.Key
```

```
Case "Zoom"
```

```
.MousePointer = esriPointerZoomIn
```

```
Case "Pan"
```

```
.MousePointer = esriPointerPan
```

```
Case "FullExtent"
```

```
.MousePointer = esriPointerDefault
```

```
.Extent = MapControl1.FullExtent
```

```
Case "SelectByPolygon"
```

```
.MousePointer = esriPointerCrosshair
```

```
Case "TrackEvent"
```

```
.MousePointer = esriPointerHotLink
```

```
Case "Identify"
```

```
.MousePointer = esriPointerDefault
```

```
Set MapControl1.CurrentTool = m_pCommand
```

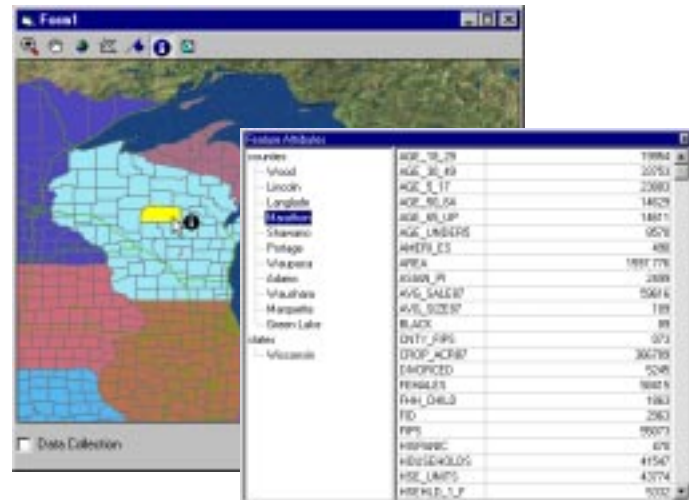
```
End Select
```

```
End Sub
```

For the Map control to be able to set the CurrentTool, the target tool must support the ITool interface, which the supplied Identify tool does. Internally, once the Map control has been associated with the CurrentTool, it forwards all MouseDown events directly to the CurrentTool first and fires the OnMouseDown event as before. If another operation has been selected from the Toolbar, for example, the Pan, we need to reset the CurrentTool to Nothing to indicate that no tool should be used. We do this at the beginning of the procedure.

Test your changes

1. Click the Run button in the Visual Basic toolbar.
2. Select the Identify tool from the Toolbar. Notice that the MousePointer has changed. Click on an individual feature or drag a rectangle and an attribute dialog box will pop up.
3. To stop running your application and return to design mode, click the Stop button in the Visual Basic toolbar.



Adding a Query tool

The AF Commands library also comes with a very useful Query tool. This tool enables you to query a specified layer using a spatial and/or attribute query. The results can be Added, Kept, or Discarded to or from the Map's SelectionSet.

To correctly use the Query tool (and all other tools that only implement the ICommand interface), we need to create our tool and load it into a global variable. If we create the command using the OnCreate method and then fire its OnClick method entirely within one procedure, it will quickly go out of scope, and an error will be raised. Storing the command in a global variable solves the problem. Alternatively, you can use a Collection to manage your tools instead of a global variable.

1. Firstly, add a private member variable to your project by adding the following code to the Form's code window General section:

```
Private m_pQuery as ICommand
```

2. Next, modify the Form_Load procedure and add the following lines to create a new Query command and pass it a reference to the Map control.

```
' Use the AfCommandsVB Query tool
```

```
Set m_pQuery = New AfCommandsVB.Query
```

```
m_pQuery.OnCreate MapControl1.Object
```

3. Modify the Toolbar1_ButtonClick procedure as shown below.

```
Case "Query"
```

```
.MousePointer = esriPointerDefault
```

```
' Fire the query command
```

```
m_pQuery.OnClick
```

Change the Feature Selection Color

4. Finally, modify the Form_Load procedure to change the default color that is used to highlight selected features. We shall change it so that selected features are shown in red. Add the following code in the Form_Load procedure after all layers have been added and their MinimumScale and MaximumScale properties have been set.

```
' Create a red color object
```

```
Dim pColor As IColor
```

```
Set pColor = New RGBColor
```

```
pColor.RGB = 255
```

```
Dim pFeatSelN As IFeatureSelection
```

```
' Iterate layers and set the color
```

```
For i = 0 To MapControl1.LayerCount - 1
```

```
Set pLayer = MapControl1.Layer(i)
```

```
' Work only with feature layers
```

```
If (TypeOf pLayer Is IFeatureLayer) Then
```

```
    ' QI for the FeatureSelection
```

```
    Set pFeatSelN = pLayer
```

```
    Set pFeatSelN.SelectionColor = pColor
```

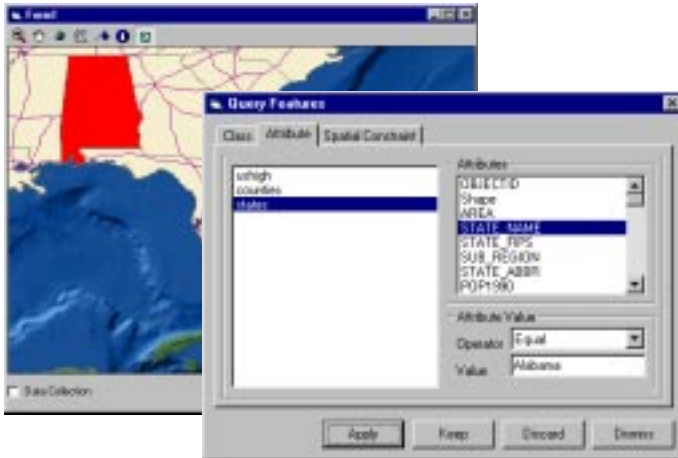
```
End If
```

```
Next i
```

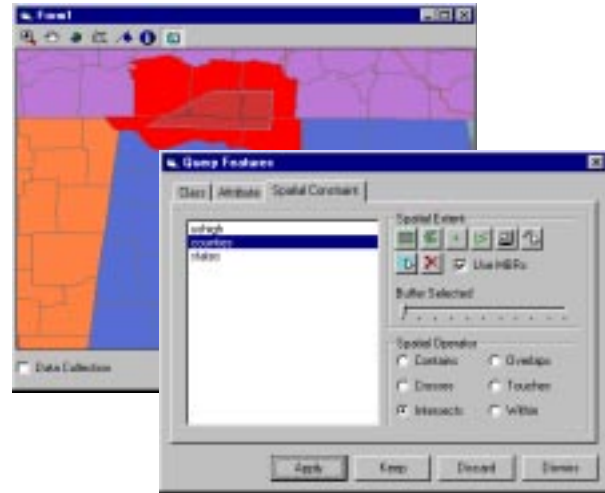
Only feature layers have a SelectionColor. To check what type of layer we have, we use the TypeOf keyword. After we have worked out if the current map layer is an IFeatureLayer, we QI directly to the IFeatureSelection interface and set its SelectionColor property.

Test and save your changes

1. Click the Run button in the Visual Basic toolbar.
2. Select the Query tool from the Toolbar and a dialog box will pop up. Build an attribute query by first clicking on the states layer in the Class tab. Then click on the Attribute tab and select STATE_NAME from the list of available Attributes. Enter "Alabama" in the Attribute Value box and click Apply.



3. Now try the Spatial Constraint. First, zoom into the Map so that the counties are drawn. In the Query dialog, click on the counties layer in the Class tab, then click on the Spatial Constraint tab. For the Spatial Extent, click on the icon for digitizing an area and then click on the map to draw a polygon. Select the "Intersects" Spatial Operator and finally click on the Apply button.
4. To stop running your application and return to design mode, click the Stop button in the Visual Basic toolbar.



Congratulations

You have built a simple application using the Map control. Although these exercises demonstrated many of the capabilities of the Map control, there is still much more to discover in ArcObjects. To learn more, read the book *Exploring ArcObjects*, consult the *ArcObjects Developers* online reference, or use the *ArcObjects Component Help* online for the most up-to-date information.