Contents

The Map control

1

IN THIS CHAPTER

- **Loading MapObjects LT**
- **Getting help**
- **Adding a map**

By following the steps in this introductory document you will use ESRI® MapObjects® LT 2 and Microsoft® Visual Basic® 6 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.
- Base the display of map layers on scale.
- Perform spatial and logical queries.
- Display features with thematic renderers.
- Add vector data and images to a map programmatically.

If you are using MapObjects LT for the first time, following all the steps described in this document in order is recommended. Later, when you are more familiar with MapObjects LT, you will be able to tell which steps are necessary for particular applications.

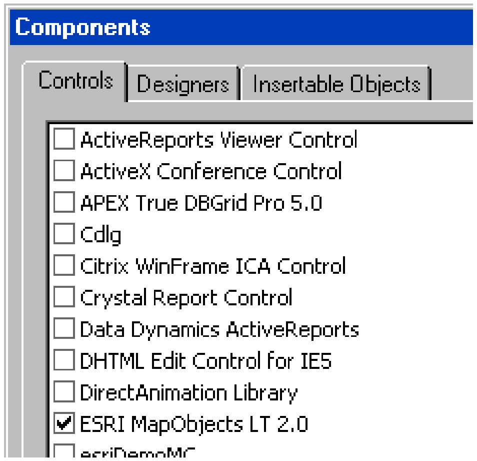
The first chapter shows how to load the Map control into Visual Basic and start to add data in design time. Adding data at design time is ideal for your first few MapObjects LT projects and for rapidly creating test applications.

Some familiarity with the Visual Basic development environment is assumed.

Note: If you accepted the defaults when installing MapObjects LT, the geographic data that this tutorial refers to can be found in C:\Program Files\ESRI\MapObjectsLT2\Samples\Data\Usa and C:\Program Files\ESRI\MapObjectsLT2\Samples\Data\Washington. The bitmaps you can use for toolbar images are in the Samples\Bitmaps folder. A complete Visual Basic project for this tutorial is available on the installation CD under \Samples\Getting Started Tutorial.

Loading MapObjects LT

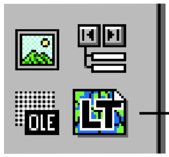
Start Visual Basic and select New project from the dialog box. Now right-click on the toolbox (the left-hand-side toolbar) and choose Components. Find ESRI MapObjects LT 2.0 in the list of available controls and check the box beside it.



- [ ] ActiveReports Viewer Control
- [ ] ActiveX Conference Control
- [ ] APEX True DBGrid Pro 5.0
- [ ] Cdlg
- [ ] Citrix WinFrame ICA Control
- [ ] Crystal Report Control
- [ ] Data Dynamics ActiveReports
- [ ] DHTML Edit Control for IE5
- [ ] DirectAnimation Library
- [x] ESRI MapObjects LT 2.0

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

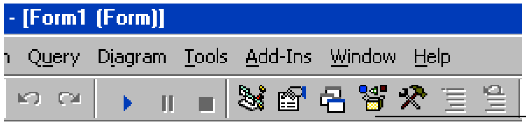
Click OK to close the dialog. Notice that a new tool appears in the Visual Basic Toolbox. This new tool is the MapObjects LT 2 Map control.



MapObjects LT Map Control

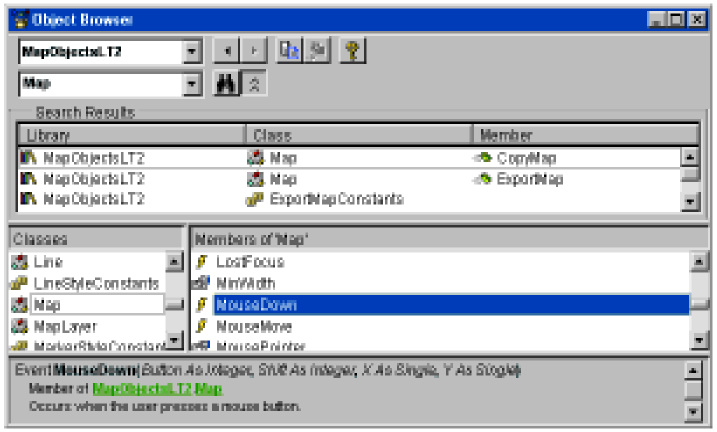
Getting help

The Map control is one of over 20 objects that make up MapObjects LT. To find out about the various objects, click the Object Browser button in the Visual Basic toolbar.



Visual Basic Object Browser

Pull down the "Libraries" combo box and choose MapObjectsLT2.



Library	Class	Member
MapObjectsLT2	Map	CopyMap
MapObjectsLT2	Map	ExportMap
MapObjectsLT2	ExportMapConstants	

Classes: Line, LineStyleConstants, Map, MapLayer

Members of 'Map': LostFocus, MinWidth, MouseDown, MouseMove

Event **MouseDown**(*Button As Integer, Shift As Integer, X As Single, Y As Single*)
Member of MapObjectsLT2.Map
Occurs when the user presses a mouse button.

MapObjects LT objects and constants are listed in the Classes list on the lower left-hand list.

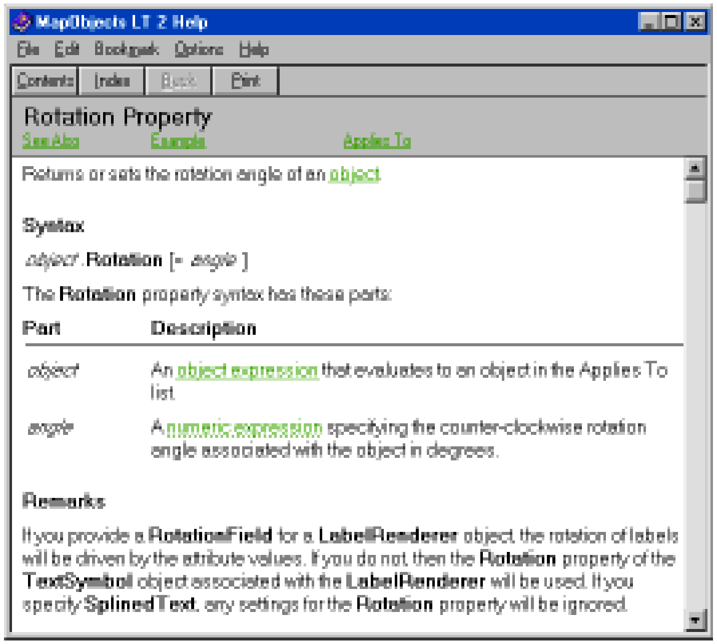
To see the properties and methods for an object, click on the object in the list. The properties and methods of that object are listed in the Members list to the right.

To see the signature of a method, click on the method in the right-hand list. The details will appear in the space at the bottom of the Object Browser.

Getting help

The following examples show how to get help with the Visual Basic Object Browser.

1. Click Symbol in the Classes/Modules list.
2. Click Rotation in the Methods/Properties list.
3. Click the help (question mark) button.


MapObjects LT 2 Help
File Edit Bookmark Options Help
Contents Index Back Print
See Also Example Applies To

Rotation Property

Returns or sets the rotation angle of an object.

Syntax

object.**Rotation** [= *angle*]

The **Rotation** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
angle	A numeric expression specifying the counter-clockwise rotation angle associated with the object in degrees.

Remarks

If you provide a **RotationField** for a **LabelRenderer** object, the rotation of labels will be driven by the attribute values. If you do not, then the **Rotation** property of the **TextSymbol** object associated with the **LabelRenderer** will be used. If you specify **SplinedText**, any settings for the **Rotation** property will be ignored.

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

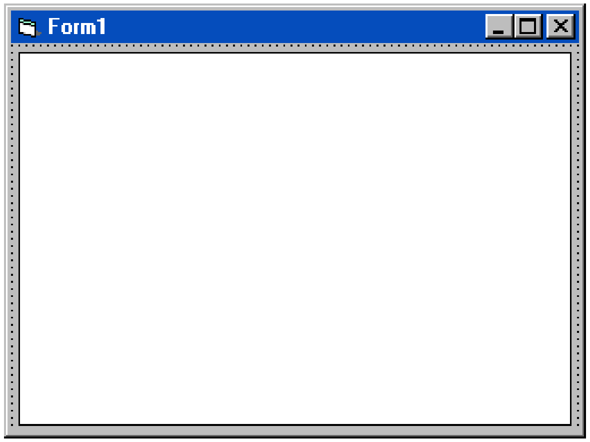
The help system provides help for every object, property, method, event, and constant in MapObjects LT. In addition to the Object Browser, the help system is accessible from the Visual Basic code window. Simply type in the name of an object, property, method, event, or constant and press F1.

Adding a map

The Map control is the object you use to display maps.

Add the Map control to the form

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



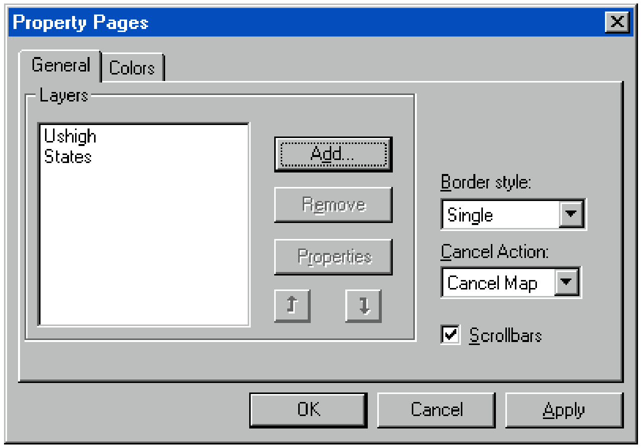
Select the data to display on the map

You can specify the data that is displayed on the map by setting properties in the Map control's property sheet.

1. Right-click the mouse on the Map to display the context menu.
2. Choose Properties to display the property sheet.
3. Click the Add button and locate the folder containing the USA sample data. If you selected the defaults when you installed MapObjects LT, this will be in

```
C:\Program Files\ESRI\MapObjectsLT2\Samples\Data
```

4. Click the States.shp file and then click Open.
5. Add the file Ushigh.shp in the same manner.


Property Pages

General | Colors

Layers:
Ushigh
States

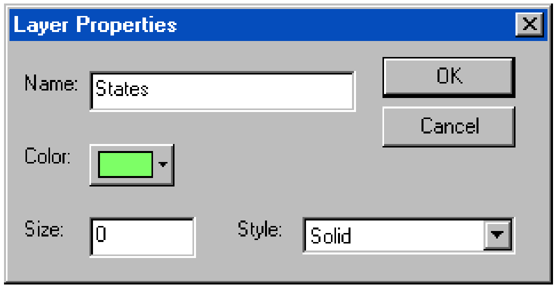
Border style: Single

Cancel Action: Cancel Map

☑ Scrollbars

Set properties for the layers

1. Click the States layer in the Layers list and then click Properties.
2. Click the Color button to select a color for the States layer.


Layer Properties

Name: States

Color: 🟩

Size: 0

Style: Solid

3. Click OK to close the dialog.
4. Select a color for the Ushigh layer in the same manner.
5. Click OK to close the property sheet.

You can also use the Properties sheet to add other data to the Map. By default, the files you can browse to include all supported vector data—shapefiles, coverages, and computer-aided design (CAD) files. However, you can browse to different types of files by changing the selected filter in the Files of Type box in the Property Pages Open dialog.

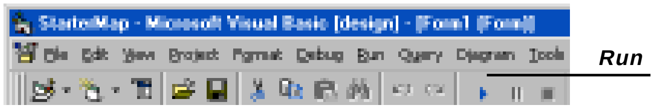
Try adding a bitmap by selecting All Supported Image Formats from the Files of Type box—if you selected the Washington sample data set sample bitmap in the \Samples\Data\Washington folder. If you selected the USA coverages during the install, try adding a coverage in a similar way, by selecting ESRI Coverage Tables from the Files of Type box.

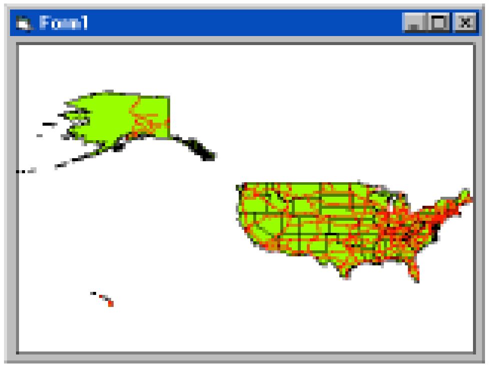
Save the project

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.

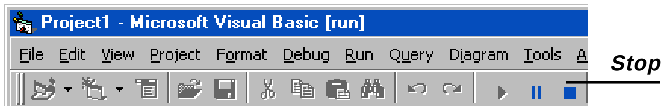
Test your application

1. Click the Run button in the Visual Basic toolbar.


StarterMap - Microsoft Visual Basic [design] - [Form1 (Form)]
File Edit View Project Format Debug Run Query Diagram Tools
Run



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


Project1 - Microsoft Visual Basic [run]
File Edit View Project Format Debug Run Query Diagram Tools A
Stop

Map tools

2

IN THIS CHAPTER

- **Adding pan and zoom controls**
- **Adding a toolbar**
- **Creating a find tool**
- **Adding a spatial query tool**

Once you have a Map control and it is displaying geographic data, you are ready to add tools and functionality to your application.

In this chapter, you will learn how to allow the user to interact with the display of the Map control by adding code to the Map event code stubs created by Visual Basic. You will also create tools that allow users to search the map by adding more controls to your form and adding code to work with the MapObjects LT ActiveX automation objects.

In this chapter, you will be working primarily with:

- The Map control's MouseDown and AfterLayerDraw events
- The Map control's Pan, SearchByDistance, SearchExpression, and SearchShape methods
- The Map control's Extent property
- The Map, Rectangle, Point, and Recordset objects

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 MouseDown event on the map.

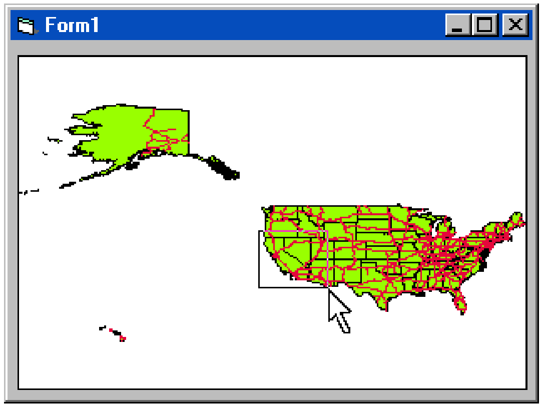
Respond to the MouseDown event

1. Double-click the map to display the Visual Basic code window.
2. Add code to Map1's MouseDown procedure.

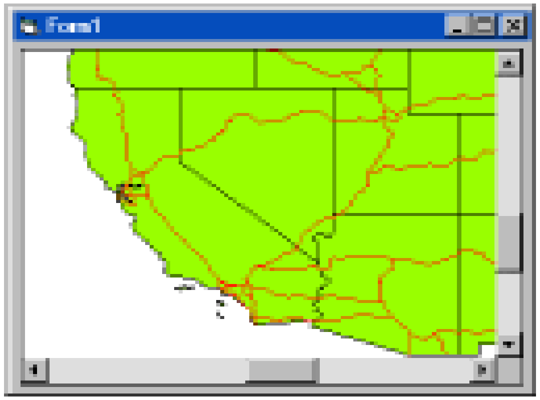
```
Private Sub Map1_MouseDown(Button As Integer_
, Shift As Integer, x As Single, y As Single)
  Set Map1.Extent = Map1.TrackRectangle
End Sub
```

Test the 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. Notice that the map is redrawn at the location you specified.



4. Click the Stop button in Visual Basic to return to design mode.

TrackRectangle is a method that applies to a map. It tracks the movement of the mouse while the user presses the mouse button, rubber-banding a rectangle at the same time. When the user releases the mouse button, the TrackRectangle method returns a Rectangle object that the application assigns into the Extent property of the map, causing the map to be redrawn with a new map extent.

Add Panning

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

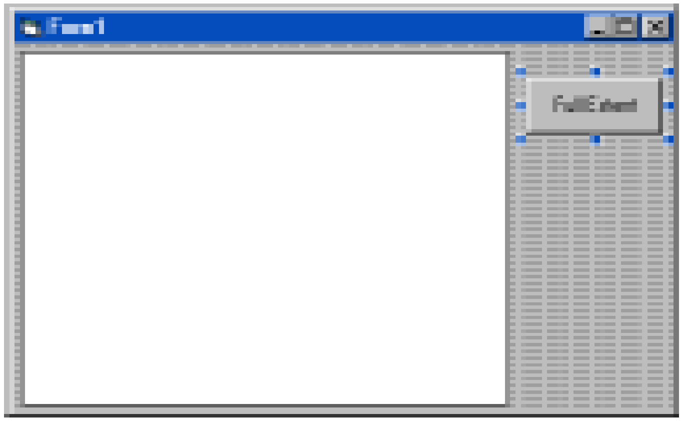
```
Private Sub Map1_MouseDown(Button As Integer, _
Shift As Integer, x As Single, y As Single)
  If Button = vbLeftButton Then
    Set Map1.Extent = Map1.TrackRectangle
  Elseif Button = vbRightButton then
    Map1.Pan
  End If
End Sub
```

If the Button parameter is equal to vbLeftButton when the MouseDown event occurs, the zooming code from the previous step will be executed. Otherwise, the code will call another map 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 zooms 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 select 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()
  Set Map1.Extent = Map1.FullExtent
End Sub
```

The FullExtent property of the map returns a Rectangle object that defines the bounding box of all the layers of the map.

Test the 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.

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

Save the project

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

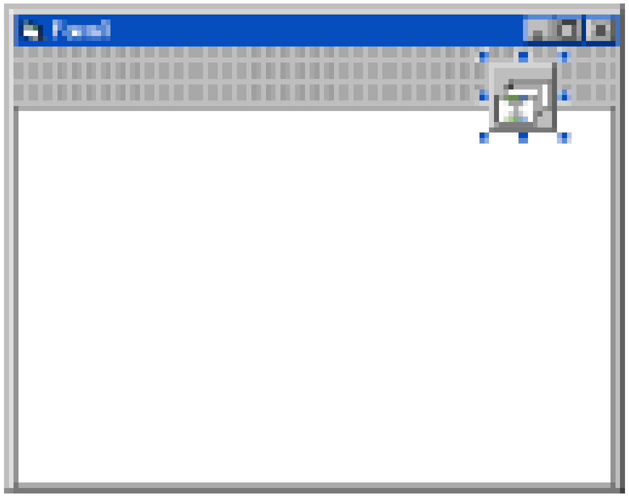
Adding a toolbar

Your application's pan and zoom capabilities are somewhat hidden from the user. In this section you will create a toolbar with pan and zoom buttons, a control that makes the functionality more visible to the user.

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.

1. Delete the Full Extent button from the form.
2. Right-click the toolbox, choose Components, and select Microsoft Windows Common Controls. You will notice that new tools are added to your toolbox.
3. Double-click the Toolbar button in the toolbox to add a Toolbar control to the form.
4. Double-click the ImageList button in the toolbox to add an ImageList control to the form.
5. 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

1. Right-click the ImageList control to display the context menu.
2. Click Properties to display the property sheet.
3. Click the Images tab.
4. Click Insert Picture and locate the folder that contains the sample bitmaps.
5. Click the Zoom.bmp file and then click Open.
6. Add the files Pan.bmp, Globe.bmp, Bex.bmp, and LayerAdd.bmp in the same manner.

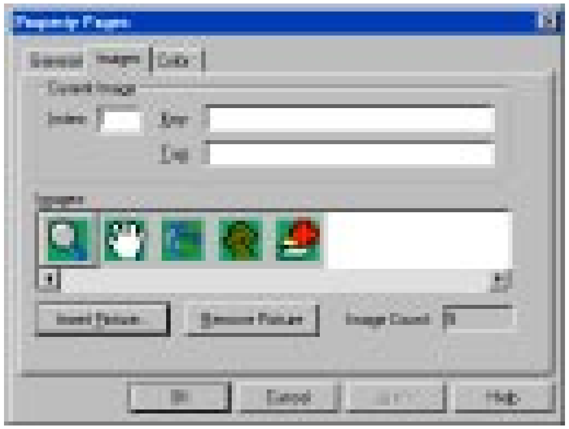
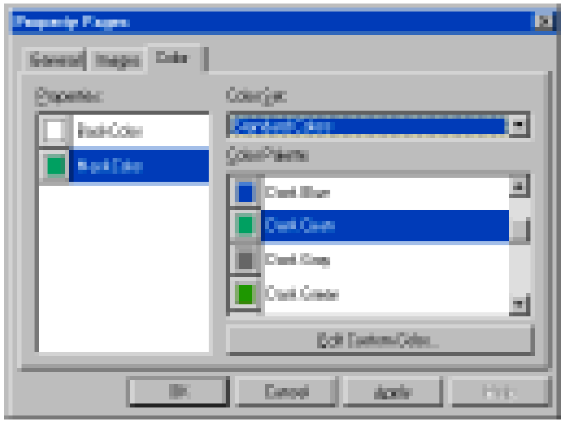


Image Count: 5

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 on the left, select MaskColor.

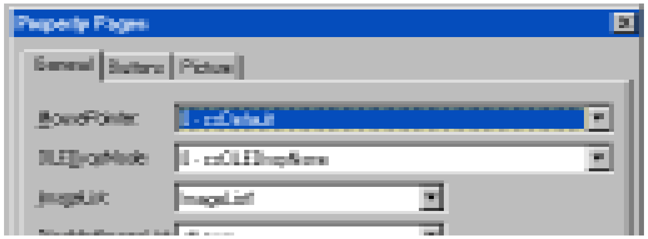


3. Click Dark Cyan in the Color Palette list on the right.
4. Click OK to dismiss the property sheet.

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 sheet.
3. In the ImageList box, click the arrow and then click ImageList1. This associates the toolbar with the ImageList control.



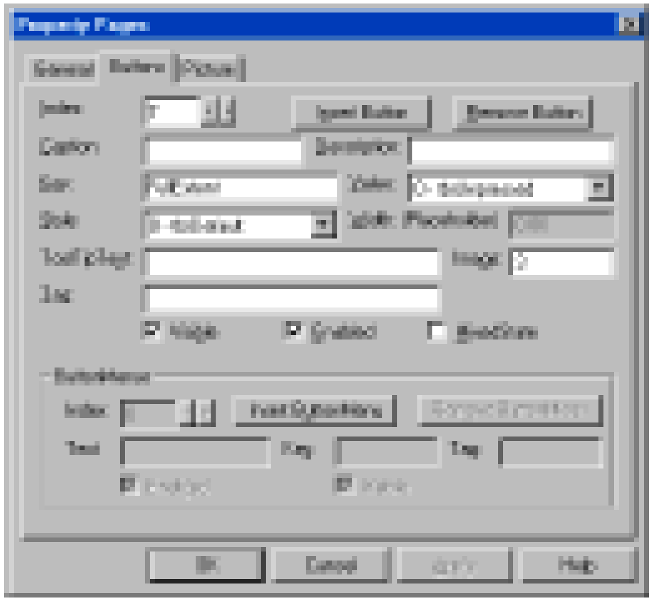
MousePointer: 0 - ccDefault
OLEDropMode: 0 - ccOLEDropNone
ImageList: ImageList1

Adding buttons to the Toolbar control

In this section you will add five buttons and two separators to the toolbar. You will set the Style of two buttons to Placeholder. The Placeholders will be used later in this document.

1. In the Toolbar property sheet, click the Buttons tab; then click Insert Button.
2. Set the button’s Style to ButtonGroup, its Image to 1, its Key to ZoomIn, and its Value to Pressed.
3. Add a second button and set its Style to ButtonGroup, its Image to 2, and its Key to Pan.
4. Add a third button and set its Style to Placeholder.

5. Add a fourth button and set its Style to Separator.
6. Add a fifth button and set its Style to Placeholder.
7. Add a sixth button and set its Style to Separator.
8. Add a seventh button and set its Image to 3 and its Key to FullExtent.



Index: 7
Key: FullExtent
Value: 0 - tbrUnpressed
Style: 0 - tbrDefault
Width: (Placeholder) [illegible]
Image: 3
Visible ✓
Enabled ✓
MixedState ☐

9. Click OK to dismiss the property sheet.

Change the MouseDown event

1. Double-click the map to display the Visual Basic code window.
2. Modify the code attached to Map1's MouseDown procedure.

```
Private Sub Map1_MouseDown(Button As Integer _
, Shift As Integer, x As Single, y As Single)

  If Toolbar1.Buttons("ZoomIn").Value _
  = tbrPressed Then
    Set Map1.Extent = Map1.TrackRectangle
  ElseIf Toolbar1.Buttons("Pan").Value _
  = tbrPressed Then
    Map1.Pan
 End If
End Sub
```

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.

Implement the FullExtent button

In this section you will reimplement the Full Extent button that you deleted.

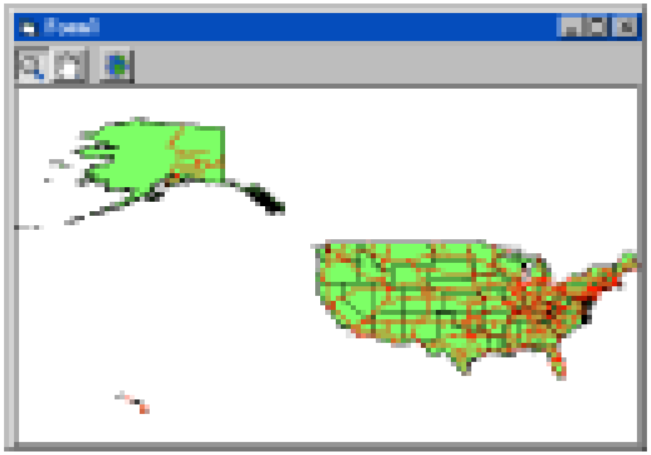
1. Double-click the toolbar to display the code window.
2. Add code to Toolbar1's ButtonClick event.

```
Private Sub Toolbar1_ButtonClick(ByVal Button _
As MSComctlLib.Button)
  If Button.Key = "FullExtent" Then
    Set Map1.Extent = Map1.FullExtent
  End If
End Sub
```

The ButtonClick event is generated whenever a click occurs on a button in the toolbar. If the Key property of the button pressed is FullExtent, the map is zoomed to its full extent.

Test the changes

1. Click the Run button in the Visual Basic toolbar.
2. Click somewhere on the map and drag a rectangle to zoom in.
3. Click the Pan button in your application's toolbar.
4. Click somewhere on the map and then drag to pan.
5. Click on the full extent button (the globe) in your application's toolbar to draw the map at its full extent.



Save the changes

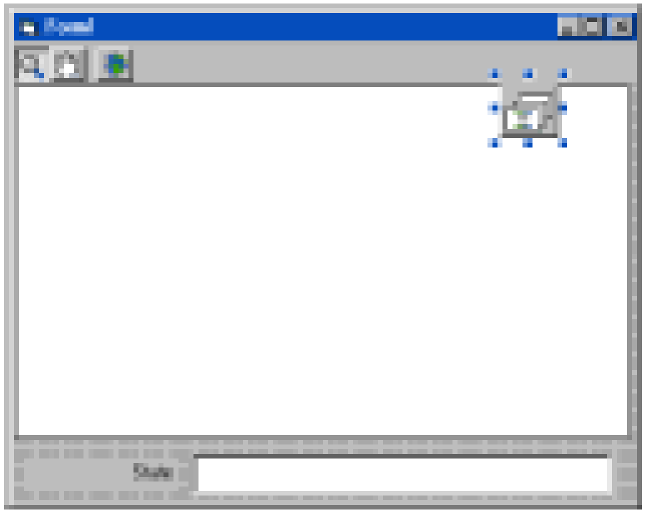
1. Click the Stop button in the Visual Basic toolbar to return to design mode.
2. Click the Save Project button in the Visual Basic toolbar to save the changes.

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.

Add controls to the form

1. Double-click the Label button in the toolbox to add a label to the form.
2. Resize the Map and reposition the label so that the label is in the lower left corner of the form.
3. In the Properties window of the new Label, set the Caption of the label to be 'State:'.
4. Double-click the TextBox button in the toolbox to add a TextBox to the form. Position the TextBox next to the label.
5. Clear the Text property of the TextBox by using the Properties window. The form should look like this.



Attach code to the TextBox

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

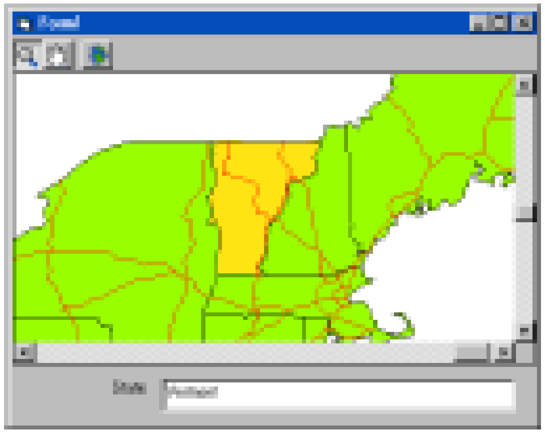
1. Double-click the TextBox to show the code window.
2. Add code to Text1's KeyPress procedure by selecting KeyPress from the right-hand dropdown list:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
  If KeyAscii = vbKeyReturn Then
    ' build a search expression
    Dim exp As String
    exp = "STATE_NAME = '" & Text1.Text & "'"
    ' perform the search
    Dim recs As MapObjectsLT2.Recordset
    Set recs = Map1.Layers("States"). _
    SearchExpression(exp)

    ' show the results, if any
    If Not recs.EOF Then
      Dim shp As Object
      Set shp = recs.Fields("Shape").Value
      Dim rect As MapObjectsLT2.Rectangle
      Set rect = shp.Extent
      rect.ScaleRectangle 2
      Set Map1.Extent = rect   ' zoom to the state
      Map1.Refresh   ' force redraw of the map
      Map1.FlashShape shp, 3  ' flash the state
    End If
  End If
End Sub
```

The code first builds a simple SQL query expression by using the value of the TextBox's Text property. It then searches the States layer using the SearchExpression method; the result is a Recordset object. If the value of the Recordset's EOF property is False, the code positions Recordset on the first record that satisfies the search expression. In that case, the code gets the value of the Shape field for the first record. The code scales the Extent of the shape and then sets it as the Extent of the map. The code then redraws the map explicitly using the Refresh method, and finally, flashes the shape three times.

Test the changes

1. Run your application.
2. Type the name of a state (e.g., Vermont) into the TextBox. Note that this search will be case sensitive.



State: Vermont

3. Press the Enter key.
4. When you are finished running your application, click the Stop button in the Visual Basic toolbar.

Adding a spatial query tool

In this section you will add a new tool to the toolbar that will perform spatial queries on the map. The spatial queries are performed on the Ushigh MapLayer and also on another MapLayer, Counties, which you will add to the Map control. You will also add code to your application that will draw the results of the spatial query on the map.

Add a new button to the toolbar

1. Right-click the toolbar to display the context menu. Click Properties to show the property sheet.
2. Click the Buttons tab.
3. Click the right arrow twice to change the Index to 3.
4. Change the button's Style to ButtonGroup, its Key to Find, and its Image to 4.
5. Click OK to dismiss the property sheet.

Add another layer

1. Right-click the map to display the context menu. Click Properties to show the property sheet.
2. Click the Add button and locate the folder where the sample data is stored.
3. Click the Counties.shp file and then click Open.
4. Click the Counties layer in the list to select it.
5. Click the down arrow to move the Counties layer below the Ushigh layer.
6. Click Properties to change the color of the Counties layer.
7. Click OK to dismiss the Layer Properties dialog and again to dismiss the property sheet.

The new MapLayer displays every county in the United States.

Add a variable to the form

1. Double-click the form to display the code window.
2. In the General section, declare a variable that will be the results of the spatial query. The type of the variable will be a Recordset.

```
Dim gSel As MapObjectsLT2.Recordset
```

It is necessary to completely qualify the Recordset type with MapObjects LT in order to avoid a name conflict with Visual Basic's built-in Recordset type.

Implement the query tool

Modify Map1's MouseDown procedure.

```
Private Sub Map1_MouseDown(Button As Integer _
, Shift As Integer, x As Single, y As Single)
  If Toolbar1.Buttons("ZoomIn").Value _
  = tbrPressed Then
    Set Map1.Extent = Map1.TrackRectangle
  ElseIf Toolbar1.Buttons("Pan").Value _
  = tbrPressed Then
    Map1.Pan

  ' spatial query
  ElseIf Toolbar1.Buttons("Find").Value _
  = tbrPressed Then
    Dim p As Point
    Set p = Map1.ToMapPoint(X, Y)
    ' search for a highway within the tolerance
    Dim recs As MapObjectsLT2.Recordset
    Set recs = Map1.Layers("Ushigh"). _
    SearchByDistance(p, Map1. _
    ToMapDistance(100), "")
    ' If nothing is found
    If recs.EOF Then
      Set gSel = Nothing
      ' Else search for counties intersecting
      Else
        Set gSel = Map1.Layers _
        ("Counties").SearchShape(recs.Fields _
        ("Shape").Value, moAreaIntersect, "")
      End If
     Map1.Refresh   ' trigger a redraw of the map
  End If
End Sub
```

When the current tool is the spatial query tool, two searches are performed. The first search is a point proximity search on the Ushigh layer. The code obtains the point by converting the x and y coordinates of the event, which are in control units, into map units. If the first search is successful, the code uses the highway it found as the input to the second search that it performs on the Counties layer. The code stores the result of the second search in the variable named gSel.

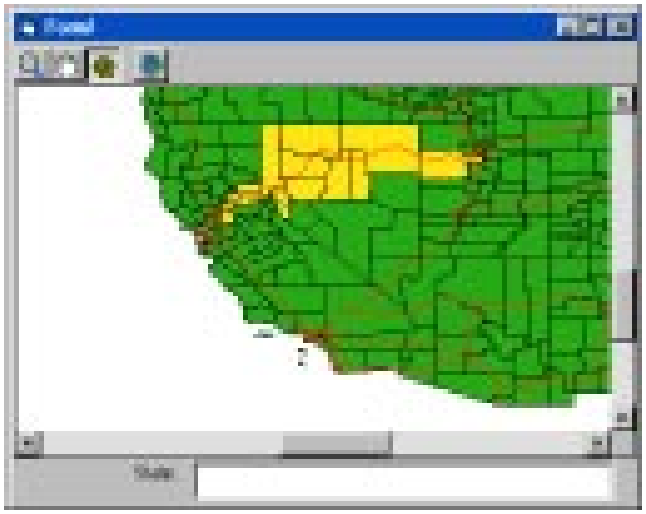
Draw the results

1. Add code to Map1's AfterLayerDraw procedure.
2. Add code to display the results of the query on top of the States layer.

```
Private Sub Map1_AfterLayerDraw(ByVal index As _
Integer, ByVal canceled As Boolean, ByVal hDC _
As stdole.OLE_HANDLE)
  If Map1.Layers(index).Name = "Counties" Then
    If Not gSel Is Nothing Then
      Dim sym as New MapObjectsLT2.Symbol
      sym.Color = moYellow
      Map1.DrawShape gSel, sym
    End If
  End If
End Sub
```

Test the changes

1. Run your application and zoom in to an area of the map that contains a highway.
2. Click the spatial query tool; then click on a highway. The corresponding country will be highlighted.



3. Click the Stop button in the Visual Basic toolbar.
4. Click the Save Project button to save the changes.

Map display

3

IN THIS CHAPTER

- **Scale-dependent display**
- **Handling the Resize event**
- **Statistical mapping**

An important part of your mapping application will be its Map control. The Map should convey its information clearly and appropriately.

In this chapter, you will learn how to handle the Resize event of the form in order to keep your Map in the intended place on the Form. You will also learn a technique for switching MapLayers on and off to improve map clarity. In addition, you will learn to implement some of the MapObjects LT renderer objects in order to symbolize your MapLayers in a more advanced way, communicating information about the different features on your Map.

In this chapter, you will use:

- The Visible and Width properties of the Map control
- The CalculateStatistics method of the Recordset object
- The DotDensityRenderer and ClassBreaksRenderer objects

Scale-dependent display

The States MapLayer displays faster than the Counties MapLayer and may provide sufficient detail when the map is at a small scale. The Counties MapLayer, on the other hand, may be the more appropriate MapLayer for large-scale displays. In this section you will add code to the BeforeLayerDraw event that uses the current Map extent to determine which MapLayer to display. Displaying one layer instead of both will speed up drawing and improve the clarity of your Map.

Respond to the BeforeLayerDraw event

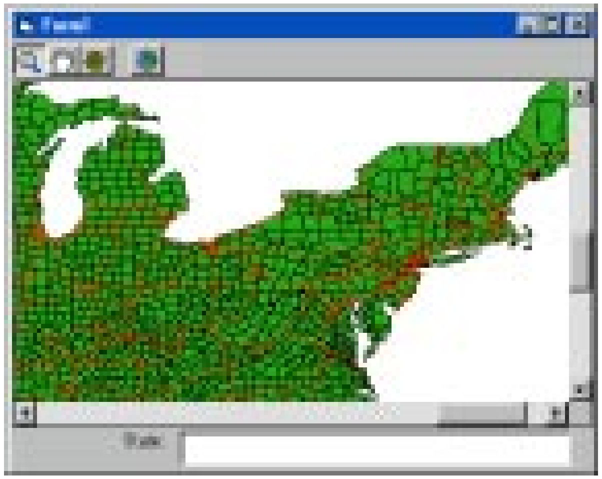
1. Double-click the Map control to display the code window.
2. Add code to Map1's BeforeLayerDraw procedure:

```
Private Sub Map1_BeforeLayerDraw(ByVal index As _
Integer, ByVal hDC As stdole.OLE_HANDLE)
  Dim layer As MapObjectsLT2.MapLayer
  Set layer = Map1.Layers(index)
  If layer.Name = "Counties" Then
    layer.Visible = Map1.Extent.Width < _
    (Map1.FullExtent.Width / 4)
  ElseIf layer.Name = "States" Then
    layer.Visible = Map1.Extent.Width >= _
    (Map1.FullExtent.Width / 4)
  End If
End Sub
```

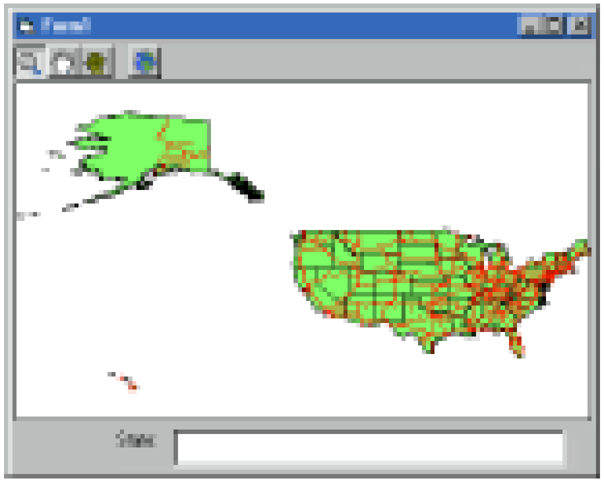
The code bases the value of the visible property of each layer on the current extent of the map. If the width of the current extent is less than one-fourth of the full extent of the map, the counties will be visible and the states will be invisible. Because this code executes in response to the BeforeLayerDraw event for each layer, the code calculates the value of the visible field before drawing occurs.

Test the changes

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



3. Click the FullExtent button. The Counties will no longer be visible.



4. Click the Stop button in the Visual Basic toolbar.
5. Click the Save Project button to save your changes.

Handling the Resize event

When you run your application and resize the form, 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()
  ' y coordinate of the find controls
  Dim yFind As Integer

  ' a constant spacing
  Dim space As Integer
  space = Text1.Top - (Map1.Top + Map1.Height)
  yFind = ScaleHeight - Text1.Height - space

  ' move the controls that make up the find tool
  Label1.Move Label1.Left, yFind
  Text1.Move Text1.Left, yFind

  ' move the map itself
  Dim mapTop As Integer
  mapTop = Toolbar1.Top + Toolbar1.Height
  Map1.Move 0, mapTop, ScaleWidth, _
  yFind - space - mapTop
End Sub
```

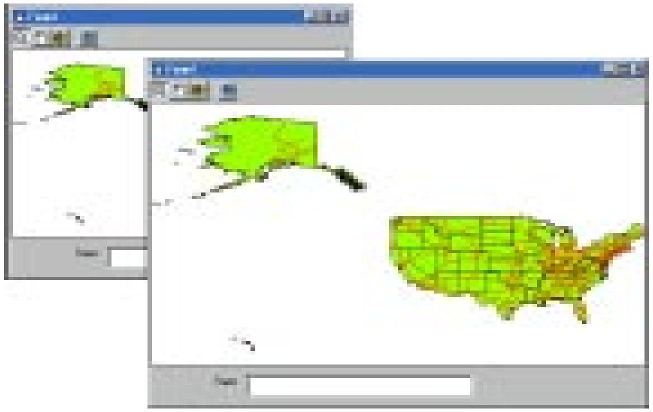
If you run your application now, it redraws the map twice initially. This is because controls on the form are initially displayed using the size and position specified during design time and then resized. To fix this problem, you will resize the controls when the form is initialized. You have already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

3. Add code to the form's Initialize procedure by selecting Resize from the right-hand dropdown list:

```
Private Sub Form_Initialize()
  Call Form_Resize
End Sub
```

Test the changes

1. Run your application.
2. Resize the form by right-click and dragging the edge of the form.
3. Release the mouse button. The controls on the form are moved and resized appropriately.



4. Click the Stop button in the Visual Basic toolbar.
5. Click the Save Project button to save your changes.

Statistical mapping

In this section you will modify your application so that it uses the underlying attribute information when it draws the MapLayers.

Attach a renderer to the Counties layer

Each MapLayer object has a Renderer property, which controls how MapObjects LT draws a MapLayer. You can use the ClassBreaksRenderer to display a MapLayer based on numeric values, as shown below.

1. Double-click the form to display the code window.
2. Add code to the Load procedure for the form.

```
Private Sub Form_Load()
  ' counties layer
  Dim rC as New MapObjectsLT2.ClassBreaksRenderer
  Set Map1.Layers("Counties").Renderer = rC
  rC.Field = "MOBILEHOME"
  Dim stats as MapObjectsLT2.Statistics
  Set stats = Map1.Layers("Counties").Records. _
  CalculateStatistics("MOBILEHOME")

  ' calculate breaks away from the mean,
  ' only add breaks within the range of values
  Dim breakVal As Double
  breakVal = stats.Mean - (stats.StdDev * 3)
  Dim i as Integer
  For i = 0 To 6
    If breakVal >= stats.Min And _
    breakVal <= stats.Max Then
      rC.BreakCount = rC.BreakCount + 1
      rC.Break(rC.BreakCount - 1) = breakVal
    End If
    breakVal = breakVal + stats.StdDev
  Next i
  rC.RampColors moLimeGreen, moRed
End Sub
```

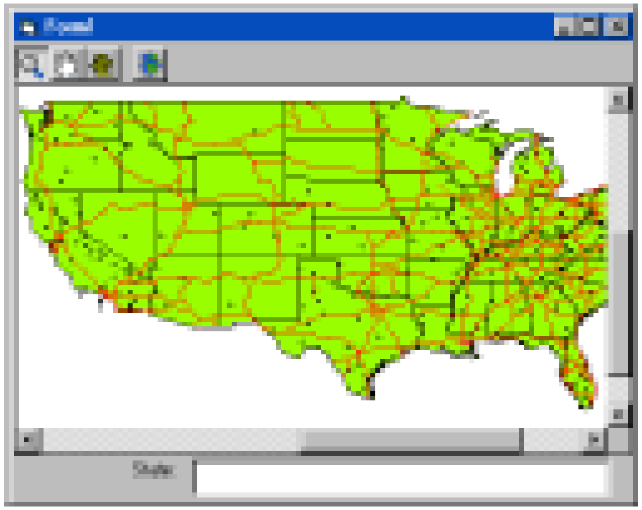
Attach a renderer to the States layer

1. Modify the form's Load procedure.
2. Append this code at the end of the form's Load procedure.

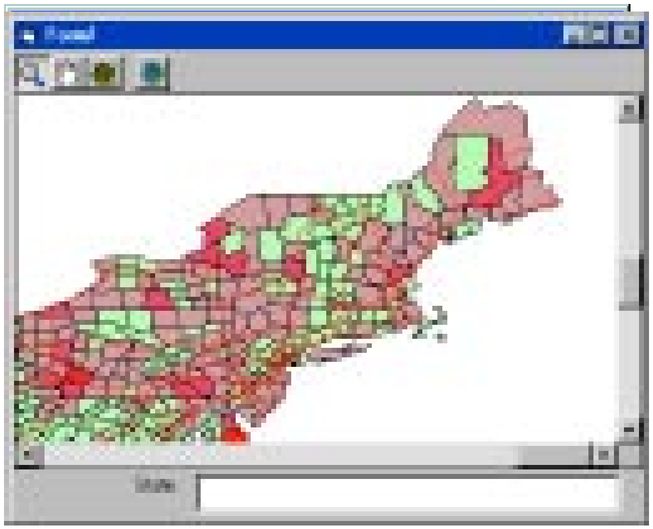
```
' states layer
  Dim rS as New MapObjectsLT2.DotDensityRenderer
  Set Map1.Layers("States").Renderer = rS
  rS.Field = "HOUSEHOLDS"
  Set stats = Map1.Layers("States").Records. _
  CalculateStatistics("HOUSEHOLDS")
  rS.DotValue = stats.Max / 40
```

Test your changes

1. Run your application and look at the States layer. Notice that the application draws polygons with dots that indicate the number of households.



2. Zoom into an area so the Counties layer becomes visible. Notice that the application now draws the counties in different colors, depending on the underlying attribute values.



3. Click the Stop button in the Visual Basic toolbar.
4. Click the Save Project button to save your changes.

Adding data at run time 4

IN THIS CHAPTER

- **Working with MapLayer objects**
- **Working with ImageLayer objects**

For many applications, users will wish to interactively add the particular raster and vector data sets they require. The required MapLayers and ImageLayers and the data files they represent may not be known in advance. Indeed, much of the power of a geographic information system (GIS) application lies in its ability to change, unlike a paper map.

In each of the previous sections, you worked with MapLayer objects that were specified using the Map control's property sheet during design time. In this section, you will add code to your application that creates MapLayer and ImageLayer objects during run time. You will add a new button to the toolbar that allows users to add data interactively during run time.

In this chapter, you will be working with:

- The MapLayer object
- The Imagelayer object
- The File and Valid properties
- The Layers collection

Working with MapLayer objects

The following section shows how to remove existing layers from your Map and demonstrates a method for adding MapLayers interactively during run time.

Remove the existing layers

1. Right-click on the map to display the context menu.
2. Choose Properties to display the property sheet.
3. Click on the Ushigh layer; then click Remove to delete the layer.
4. Remove the other layers and click OK.
5. Comment out the code in the Form_Load and Map_BeforeLayerDraw events.

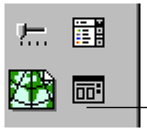
Add a new button to the toolbar

1. Right-click the toolbar to display the context menu. Click Properties to show the property sheet.
2. Click the Buttons tab and select Index number 5.
3. Change the button's Style to Default, its Key to AddLayer, and its Image to 5.
4. Click OK to dismiss the property sheet.

Add a CommonDialog control to the Form

This control will be used to select files interactively.

1. Right-click on the Visual Basic Toolbox and choose Components.
2. Find Microsoft Common Dialog Control in the list of available controls and check the box beside it.
3. Click OK to close the dialog. Notice that a new tool should appear in the Visual Basic Toolbox.
4. Double-click this new tool to place the control on the form.



Respond to the new AddLayer button

1. Double-click the toolbar to display the code window.
2. In the Toolbar1 ButtonClick event, modify the code as shown below.

```
Private Sub Toolbar1_ButtonClick(ByVal Button _
As MSComctlLib.Button)
  If Button.Key = "FullExtent" Then
    Map1.Extent = Map1.FullExtent

  ' Respond to the new AddLayer button
  ElseIf Button.Key = "AddLayer" Then
    With CommonDialog1
      .Filter = "Shapefiles (*.shp)|*.shp"

      ' Handle the user pressing Cancel
      .CancelError = True
      On Error Resume Next
      .ShowOpen
      If Err.Number = cdlCancel Then
        Exit Sub
      End If
      On Error GoTo 0

      ' Call a routine to add selected file
      If Not AddShapefile(.FileName) Then
        MsgBox "Failed to add shapefile"
      End If
    End With
  End If
End Sub
```

Add a function to add MapLayers

1. In the General section of the code window, declare a procedure.

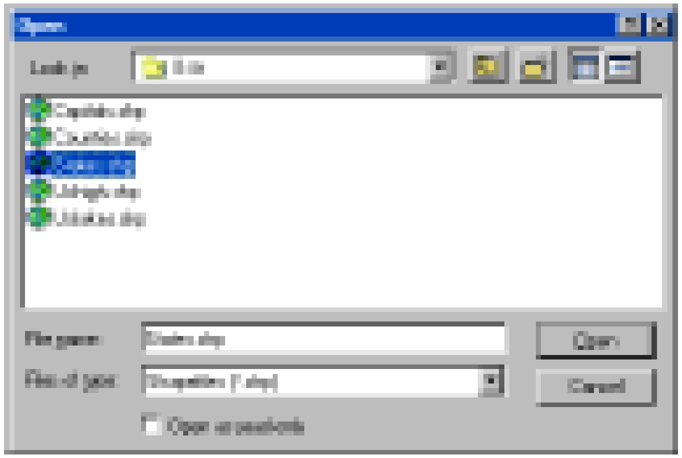
```
Private Function AddShapefile(ByVal FilePath _
As String) As Boolean
  ' Initialize return value.
  AddShapefile = False

  ' Create new MapLayer.
  Dim lyrNew As MapObjectsLT2.MapLayer
  Set lyrNew = New MapObjectsLT2.MapLayer

  ' The FilePath variable is the full
  ' path name of the selected file.
  lyrNew.File = FilePath
  If lyrNew.Valid Then
    ' If the layer is Valid then add to Map
    If Map1.Layers.Add(lyrNew) Then
      AddShapefile = True
    End If
  End If
End Function
```

Test the changes

1 Run your application to test the changes. First click the new AddLayer button on the toolbar.

2. In the dialog that opens, navigate to a directory containing shapefiles.

3. Select a shapefile and click Open. You should then see the selected shapefile being added to the Map.



4. Click the Stop button and save the changes.

You can now add shapefiles interactively to the Map. You may also wish to add CAD files or coverages to the Map interactively. CAD files may simply be added in a similar way to shapefiles. When setting the CommonDialog control's Filter property, note the CAD files extensions; many have nonstandard extensions.

ArcInfo™ coverages may also be added as MapLayers. The structure of a coverage is more complex than that of a shapefile—coverages consist of many files, and each coverage can contain one or more feature classes. For more information about adding ArcInfo coverages as MapLayers, look in the online reference index for "coverages, adding to a Map."

Working with ImageLayer objects

In each of the previous sections, you worked with MapLayer objects that were based on vector data. In this section, you will see how to add layers to your map that are based on image data. MapObjects LT allows you to use a wide range of image types as ImageLayers, including such common image types as Windows® bitmaps (.bmp), tagged image file format (.tiff), and JPEG images (.jpg, .jpeg). (For a full list of supported image formats, look in the MapObjects LT online help.) You will add code that allows the new AddLayer toolbar button to add either bitmaps as ImageLayers or shapefiles as MapLayers.

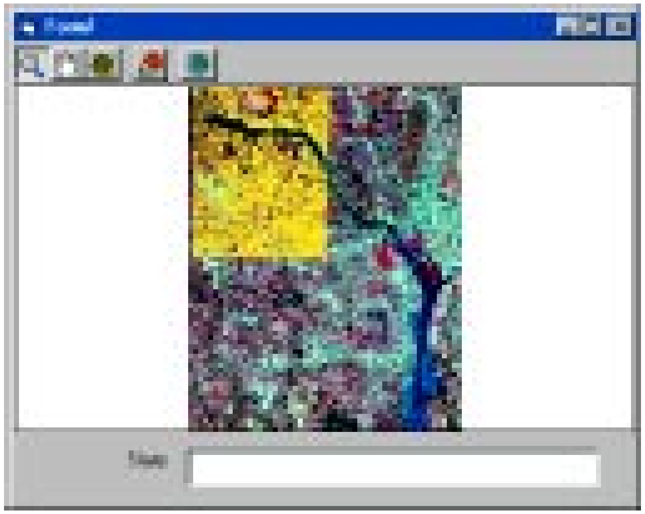
Select an image layer to display on the map

You can specify that an image is to be displayed as an ImageLayer by setting the properties in the Map control's property sheet similar to those used to add MapLayers.

1. Right-click the mouse on the map to display the context menu.
2. Choose Properties to display the property sheet.
3. Click the Add button and then select “All supported Image formats” from the “Files of type” combo box.
4. Navigate to the Washington sample data folder.
5. Click the Wash.bmp file and then click Open.
6. Click Add again and then choose the Roads shapefile.
7. Click OK to dismiss the property sheet.

Test the changes

1. Click the Run button in the Visual Basic toolbar. You should see that the bitmap of Washington is added to the Map along with the Roads shapefile. Right-click to zoom in on the top-left corner.



The roads in the shapefile should follow the roads in the bitmap. The sample bitmap has a world file (.wld) and is therefore georeferenced correctly.

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

Adding an ImageLayer in code

Previously, you added a MapLayer programmatically by using the CommonDialog control. Now you will add an ImageLayer programmatically, by adding to the existing code.

Respond to the new AddLayer button

1. Double-click the form to display the code window.
2. In the Toolbar1 ButtonClick event, modify the existing code.

```
Private Sub Toolbar1_ButtonClick(ByVal Button _
As MSComctlLib.Button)
  If Button.Key = "FullExtent" Then
    Map1.Extent = Map1.FullExtent
  ' Respond to the new AddLayer button
  ElseIf Button.Key = "AddLayer" Then
    With CommonDialog1
      .Filter = "Shapefiles (*.shp)|*.shp" & _
      "|Bitmaps (*.bmp)|*.bmp"
      .CancelError = True
      On Error Resume Next
      .ShowOpen
      If Err.Number = cdlCancel Then
        Exit Sub
      End If
      On Error GoTo 0
      ' Add a MapLayer or ImageLayer
      If .FilterIndex = 1 Then
        If Not AddShapefile(.FileName) Then
          MsgBox "Failed to add shapefile"
        End If
      ElseIf .FilterIndex = 2 Then
        If Not AddBitmap(.FileName) Then
          MsgBox "Failed to add Bitmap"
        End If
      End If
    End With
  End If
End Sub
```

Add a function to add MapLayers

1. In the General section of the code window, declare a procedure.

```
Private Function AddBitmap(ByVal FilePath _
As String) As Boolean

  ' Initialize return value.
  AddBitmap = False
  ' Create new ImageLayer.
  Dim imlNew As MapObjectsLT2.ImageLayer
  Set imlNew = New MapObjectsLT2.ImageLayer

  ' The FilePath variable is the full
  ' path name of the selected file.
  imlNew.File = FilePath
    If imlNew.Valid Then
      ' If the layer is Valid then add to Map
      If Map1.Layers.Add(imlNew) Then
        AddBitmap = True
      End If
  End If
End Function
```

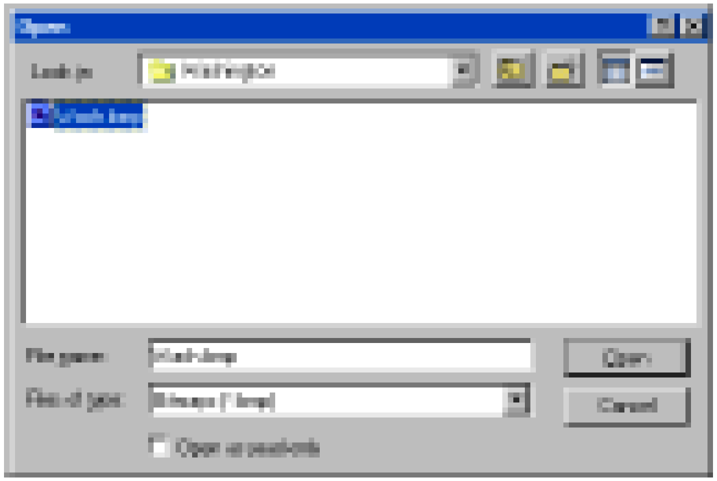
Test the changes

1 Run your application to test the changes. First, click the new AddLayer button on the toolbar.

2. In the dialog that opens, select Bitmaps from the Files of Type dropdown list.

3. Browse to a directory containing georeferenced bitmaps.

4. Select a bitmap and click Open. You should then see the selected bitmap added to the Map.



5. Click the Stop button and save the changes.

You can also specify MapLayers and ImageLayers at design time and add them at run time. Create a layer object in a similar way, but specify a path to the file in your code. Try to use relative paths, if possible.