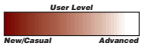
Atlas Scripting with Class

By Samuel A. Berg,
Atlas GIS Product Specialist



The AtlasMap OCX can make the functionality of Atlas GIS 4 available for use in custom applications built in Visual Basic (VB). In addition to automating procedures and modifying the user interface, VB can be used to fill dialog boxes and execute them without user interaction.

Typically, users select dialog box parameters by clicking option buttons or typing expressions in a text box. The AtlasMap OCX includes Dynamic Link Library (DLL) calls that can set dialog parameters through code. The function AtlDialogExec performs the tasks associated with a dialog box without requiring user input. However, settings for a dialog box can persist so that all parameters need to be set before each execution of the dialog box to avoid the unpredictable results that could follow executing a dialog box that had missing or inappropriate parameters.

Setting Parameters with Classes

Object-oriented programming defines an object that represents another entity. This type of code abstraction makes reading and writing code more logical and efficient. Using the AtlasMap OCX to set the parameters of a dialog box requires that each dialog parameter be set individually, spelled exactly right, and that the correct data type is specified. Incorrect parameters will not produce the desired results.

Writing VB classes that represent a particular Atlas GIS dialog box is one way to make the AtlasMap OCX easier to use. Once the VB classes are created, only the code that sets the properties of the particular class is needed. The class object will issue the appropriate commands through the AtlasMap OCX to recreate the functionality. This example of object-oriented encapsulation reduces the time needed to write new code that uses Atlas GIS functionality.

This article illustrates how to create a wrapper (encapsulation) class for one of the more complex dialogs in Atlas GIS—Geocode by Address—using the Visual Basic Class Builder Wizard. This wizard automatically adds the storage variables that are appropriate to the data type for each property to the class module. This saves the programmer a lot of typing.

The GIS VB Online Help manual that comes zipped with the Atlas Script/VB package lists 30 parameters that must be set before issuing the AtlDialogExec command to execute the Geocode by Address dialog box. To access this list of parameters, invoke the GIS VB Online Help manual, go to the Index section, and type Geocode By Address.

This article will show how to create a class based on the Geocode by Address dialog box. The 30 parameters needed to execute the dialog box will be specified as properties of the Geocode by Address class. Code will be used to set the appropriate dialog parameters through the AtlasMap OCX.

Visual Basic and Atlas GIS treat numeric data types differently, so some of the parameters specified for the dialog box must be translated when setting properties in Visual Basic. The AtlasMap OCX expects an INTEGER as a data type, while Visual Basic actually needs to pass this data as a LONG data type. The Atlas REAL data type is passed to Visual Basic as a DOUBLE data type. There is no difference in how a STRING data type is handled. Make sure that the correct data type is passed to the AtlasMap OCX.

Creating a New Class

1. From the Visual Basic menu, choose Project > Add Class Module > VB Class Builder. Name the class module atlDialogGeoByAddr. This will be the name used to refer to the class in code. Accept the default Instancing Type of MultiUse. This will allow the creation of multiple instances of the class.

2. Use the Class Builder Wizard to add all 30 of the dialog parameters as properties. Highlight the atlDialogGeoByAddr class and right-click in the Property tab view to add a property. Use the dialog setting name (i.e., RowOption, Stdize, MethZip5) as the name for each property for this class. Set the data types for these properties to reflect the requirements of VB (e.g., since the Row Option dialog setting specifies an INTEGER in the GeoCode by Address parameter list, make the property type LONG).

Note: The Target Layer setting (TgtLyr) gives the option of using either the layer name, set as a STRING data type, or layer ID, specified as an INTEGER in the dialog parameters list and set as a LONG data type.

3. The atlDialogGeoByAddr will have one method named Execute. Right-click in the Method tab view to add it.

4. After setting all the properties and adding the Execute method, choose File > Update Project to write the class properties to the

What You Will Need

Atlas GIS 4
Visual Basic 5.0 or 6.0
Atlas Script/VB package from the ESRI Web site that contains

- AtlasMap OCX (the main ActiveX control component that is loaded into the VB project)
- Atlasvb2.dll (the main scripting DLL that makes function calls to Atlas GIS)
- Agis2.dll (the updated core Atlas GIS DLL that handles function calls)
- GIS VB Online Help and other documentation zipped with the package

The AtlasMap OCX is an ActiveX control that is available free to registered users of Atlas GIS 4. Download the OCX and documentation from the Atlas GIS page (**www.esri.com/atlas**) on the ESRI Web site. Refer to the Word 97 document included in the zipped file for information on setting up the system environment. Applications created with the control require that a registered copy of Atlas GIS 4 be installed on the computer that will execute the application.

Success in scripting with Atlas GIS depends on experience with both the Atlas GIS software and Visual Basic programming.

class module. Choose File > Exit to exit the Class Builder Wizard.

This process creates an object that will store the variables that represent settings for a dialog box. Inside the Set procedure for each property, each value passed to the class will be stored in a private data member that has the appropriate data type for the property as shown in Figure 1.

Figure 1

```
Public Property Let RlxZip5(vData As Long)
     mvarRlxZip5 = vData     ' mvarRlxZip5 is a modular level LONG
End Property
```

No actions are issued to the Atlas GIS session until the AtlDialogSet command changes the settings for the dialog box and AtlDialogExec executes it. The Execute method for atlDialogGeoByAddr will gather the information about its own settings—it knows about its own private data—and will issue the AtlDialogExec command to run the dialog box.

5. Inside the Execute procedure for atlDialogGeoByAddr, write the code to set each dialog parameter. Depending on the data type, the syntax will be either AtlDialogSetInt, AtlDialogSetString, or AtlDialogSetReal for each setting stored in the class. See Figure 2 for an example.

Figure 2

```
With MainMdi.AtlasC
     .AtlDialogSetInt "GeoAddr.RowOption", Me.RowOption
     .AtlDialogSetInt "GeoAddr.Stdize , Me.Stdize
     .AtlDialogSetInt "GeoAddr.MethAddr, Me.MethAddr
     .AtlDialogSetReal "GeoAddr.FeetOffset", Me.FeetOffset
     .AtlDialogSetString "GeoAddr.DispCol", Me.DispCol
     'etc…
End With
```

Once the dialog parameters have been set, use the AtlDialogExec function to execute the dialog box without user input, as shown in Figure 3.

Figure 3

```
With MainMdi.AtlasC
     .AtlDialogExec "GeoAddr"    'Execute the Geocode by Address Dialog
End With
```

Figure 4 shows the code that uses the atlDialogGeoByAddr class to create a copy of the class, set the class properties, and issue the Execute method that runs the dialog box.

Figure 4

```
Public Sub GeocodeByAddress ()
   'First, initialize the class object:
   Dim GeoByAddr as New atlDialogGeoByAddr
   With GeoByAddr
      'Set ALL the parameters:
      .DispCol = "Name"
      .Addr1Col = "Address"
      .CityCol = "City"
      .RlxHouse = 1
       'etc…set all the properties.

        'Run the dialog
        .Execute
   End With
```

Class Benefits

Creating a class can prevent potential run-time errors caused by passing inappropriate data types directly to AtlasMap OCX commands. Data is passed to the class module first so code in the class can implement data validation before using the data passed to it.

Automation is another benefit. Instead of individually geocoding a collection of similar point tables that store address data, all the tables can be geocoded in one operation by writing a loop that will geocode each point table by changing only the TgtLayer property of the

Continued on page 44

Dialog Parameter Specifies	Set Visual Basic Property To
INTEGER	LONG
REAL	DOUBLE
STRING	STRING

Visual Basic and Atlas GIS treat numeric data types differently.

The AtlasWare GIS VB Online Help lists the 30 parameters that must be set before executing the Geocode by Address dialog box.

Atlas Scripting with Class

Continued from page 43

atlDialogGeoByAddr class and using the Execute method. The example in Figure 5 assumes that the atlDialogGeoByAddr class has been initialized and all of its default parameters are set.

Figure 5

```
Public Sub GeocodeAllPntTables ()

     Dim PntTables (255) as Long     'Array of longs to hold LayerIDs
     Dim I as Integer    'variable for looping

     'Get the layer list of all point tables:
     NumPointTables = MainMdi.AtlasC.AtlLayerGetList (A_LYRLIST_DPOINTS, A_NO_FILE_ID, PntTables() )

     'Iterate through the layer list and geocode each one:
     For I = 0 to NumPointTables-1
          GeoByAddr.TgtLayer = PntTables (I)    'Set the Point Table Layer to geocode
          GeoByAddr.Execute     'Run the dialog through the class
     Next

End Sub
```

Putting Classes to Work

The next step could be to create enumerations inside a standard module to wrap code around Atlas GIS scripting constants and make them even easier to use. See Figure 6 for an example.

Figure 6

```
'in a public standard module:
Public Enum eStatus
     IsOn = A_ON
     IsOff = A_OFF
End Enum

'in your geocode dialog box setting procedure:
MainMdi.AtlasC.RlxHouse = IsOn
MainMdi.AtlasC.MethAddr = IsOff
```

End Sub

Building classes that encapsulate the Atlas scripting code makes Atlas GIS functionality object-oriented and ensures not only that dialog parameters will be automatically and correctly set but allows the application to check for valid data types before executing the dialog box. Creating VB classes to handle setting dialog box parameters speeds the creation of code and eliminates sources of error while giving greater access to Atlas GIS functionality through the AtlasMap OCX.

The documentation in the Atlas Script/VB package describes how to register the AtlasMap OCX and set up the system environment.