# Desktop Notes

## PC ARC/INFO

# Anchoring Dialog Boxes

**Arron Heriford, ESRI Developer Support**

Have you ever placed a dialog box using pixel coordinates only to find it at a different position when the application is run on another PC with a different screen resolution? Here are two ways to control box placement at various screen resolutions.

**Method One**

Specify a default position for all dialog boxes when the application initializes, then use the PIN mode of operation (command syntax: WIN DB P) for dialog box placement. This causes the specified dialog box to remain at a particular location on the screen until explicitly moved. If the box is closed, the PIN mode will cause the box to reopen in the same spot.

**Method Two**

Specify a box position using a ratio of screen resolution. For example, to start the dialog box in the upper third of the screen use the ratios shown below.

```
maximum_number_of_row_pixels divided by
3maximum_number_of_column_pixels divided by 3
```

Since this method requires knowing the current screen resolution, the application could be modified to interactively determine the screen resolution. Use a Simple Macro Language (SML) program to prompt the user to enter the screen resolution (i.e., 640 x 480, 800 x 600), and store these values as SML variables in a text file to be used as input for dialog box placement. If the user does not know the screen resolution, the test shown in Figure 1 can be run to determine the approximate screen resolution. These SML instructions create a dialog box that is too large to fit on any current graphics screen (i.e., 9999 x 9999). The variable "%1" returns values that approximate the lower-left corner of the graphics screen, in x and y pixels, as well as the dialog box size in row and column pixels.

Figure 1

```
WIN DB P
WIN POS W 1 9999 9999
WIN DB C [dialog_definition_file]
WIN DB POS R 1
SETVAR 2 "%1"
WIN DB D
```

## ArcView GIS

# Getting RGB Values for Symbols Colors

**By Rex Hansen, ESRI Technical Support**

To use the same data in ArcView GIS and ArcInfo, it is often useful to have the red, green, and blue (RGB) values for all the symbol colors in a theme's legend. The script shown in Figure 2 creates a table with the labels and RGB values for each classification in the active theme.

Figure 2

```
'--- Script rgbsymbtab.ave

'--- Get legend for active theme
theView = av.GetActiveDoc
theTheme = theView.GetActiveThemes.Get(0)
theLeg = theTheme.GetLegend.Clone

'--- Make the table
aFilename =FileDialog.Put("rgb.dbf".asFilename, "*.*", "Create Attribute
Table")
theVTab = VTab.MakeNew(aFilename, dbase)

'--- Make the fields
theLBLfld = Field.Make("AVlabel", #FIELD_CHAR, 20, 0)
theRfld = Field.Make("R", #FIELD_DECIMAL, 5, 0)
theGfld = Field.Make("G", #FIELD_DECIMAL, 5, 0)
theBfld = Field.Make("B", #FIELD_DECIMAL, 5, 0)
theVTab.SetEditable(true)
theVTab.AddFields({theLBLfld, theRfld, theGfld, theBfld})
theNum = (theLeg.GetNumClasses - 1)
'--- Get the colors and add them to the table
for each i in 0..theNum
  theVTab.AddRecord
  theList = theLeg.ReturnClassInfo(i)
  theLBL = theList.Get(0)
  theSYM = theList.Get(2)
  theVTab.SetValue(theLBLfld, i, theLBL)
  theRGBlist = theSYM.GetColor.GetRGBlist
  theVTab.SetValue(theRfld, i, theRGBlist.Get(0))
  theVTab.SetValue(theGfld, i, theRGBlist.Get(1))
  theVTab.SetValue(theBfld, i, theRGBlist.Get(2))
end
theVTab.SetEditable(false)
theTable = Table.Make(theVTab)
theTable.GetWin.Open
av.PurgeObjects

'--- End of Script
```


## *ArcView IMS*

# Target a Frame Using JavaScript

**By Rex Hansen, ESRI Technical Support**

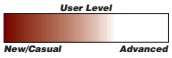
The online help for ArcView Internet Map Server (IMS) suggests using WebLink.WriteResponseHeader ("Window-tarGet: Framename"+CR+NL) to specify which frame to send information to in a multiframe HTML page. However, due to HTML limitations, some Web server/browser configurations may not load the frames correctly or may even produce error messages. These problems can be avoided by using JavaScript (not to be confused with Java, an object-oriented programming language). Security should not be a problem since JavaScript operates much like HTML on the client side.

**See software books or search the Web for sites providing more information on JavaScript.**

Figure 3

The following code creates functions ("loadIndent", "loadHelp"), tells the functions what to do ("top.ident_window.location.href='http://lanka/scripts/esrimap.dll?name=htmlmap&cmd=m3'", "top.help_window.location.href='http://www.esri.com'"), and associates the functions with a button ("<input type=button value=ident onClick=""loadIdent();"">", "<input type=button value=help onClick=""loadHelp();"">").

```
wl.writestring("<script language=JavaScript>")
wl.writestring("function loadIdent() {"++CR+NL)
wl.writestring("parent.ident_window.location.href='http://lanka/scripts
/esrimap.
dll?name=htmlmap&cmd=m3'")
wl.writestring("}"++CR+NL)

wl.writestring("function loadHelp() {"++CR+NL)
wl.writestring("parent.help_window.location.href='http://www.esri.com'")
wl.writestring("}"++CR+NL)
wl.writestring("</SCRIPT>")

wl.writestring("<input type=button value=ident onClick=""loadIdent();"">")
wl.writestring("<input type=button value=help onClick=""loadHelp();"">")
```




## *ArcCAD Version 14*

# Connecting to External Databases

**By Placido Paderes, ESRI Technical Support**

The use of the universal command syntax provided by Structured Query Language (SQL) has become the standard for reading and writing to external databases. The ability to connect to external databases is often critical for GIS applications. ArcCAD Version 14 provides a very useful tool for connecting to external databases using SQL Record themes and Microsoft's Open Database Connectivity (ODBC) software. The process is straightforward. Once the ODBC software is configured for the appropriate data source, connecting to an external database is as simple as defining a theme and issuing an SQL query. The steps for this process are given below.

**Configure ODBC software**

1. Choose Control Panel and double click on the ODBC icon.
2. In the ODBC Data Source Administrator dialog box, choose the System DSN panel.
3. If necessary, choose the Add button to add any ODBC drivers to the System DSN that are required to complete your configuration. Choose Finish when you are done.
4. Once you have the appropriate ODBC drivers added to the System DSN panel, click the Configure button.

**Connect to an External Database**

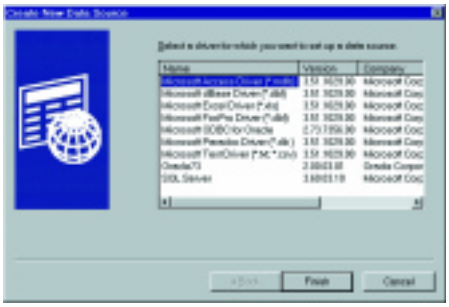
The next sequence of procedures walks you through configuring a driver for Microsoft Access.

1. Select a data source and provide a description of that source.
2. Click on the Select button in the Database section to invoke the dialog box titled Select Database.
3. Use this dialog box to navigate to the location of the desired database file and select it. Since this is an Access database, it will have a *.mdb extension.
4. Specify the data source name and description. Click OK to save the settings and exit the dialog box. For additional information regarding ODBC configurations, please consult the ODBC documentation or online help system.

**Define an SQL Theme**

The next step is to define an SQL Theme in ArcCAD that will establish the connection to the external database.

1. Choose THEME | DEFTHM. In the Define Theme dialog box, enter a theme name.
2. Choose Record (Sql) for Feature Class. In the SQL section, choose the appropriate selection from the drop-down box marked



*Add any ODBC drivers that are required for your configuration.*

Connections. This choice will be based on the ODBC data source that previously specified.
3. If the ODBC data source selected in the previous step is not available, go back and repeat the process.
4. After choosing the appropriate ODBC connection, enter a login and password if necessary.
5. Enter a valid SQL expression to complete the process. When finished, click OK.

Once a successful connection is made to the external database, view the SQL Record theme using the WBROWSE command. From the ArcCAD menu, choose Attributes | Datafiles | Wbrowse. All the records in the table should be visible.

User Level
New/Casual — Advanced

*MapObjects*

# Assigning a Projection

**By Chuck Baker,**
**ESRI Developer Support**

MapObjects Version 1.2 can project supported image formats such as TIF, BIL, and BMP but these images must have a projection assigned to them. Nongeoreferenced aerial photos or simple scanned images will not work. The ESRI Arc Automation Server DLL and the ESRI Utility Automation Server DLL must be referenced, and these DLLs require ArcInfo to be installed and loaded in Visual Basic from the References menu choice under the Project menu.

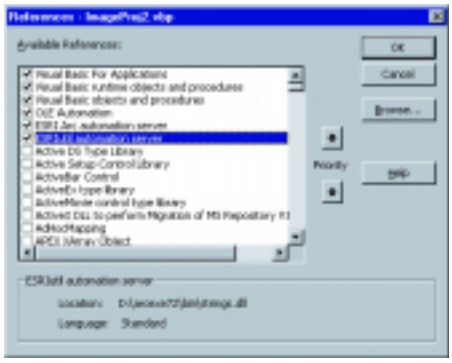
Convert the image to a grid so the data can be assigned a projection by automating the ArcInfo command IMAGEGRID. See Figure 4 for an example of Visual Basic code that will do this. Since IMAGEGRID requires an input image file name and an output grid file name, text boxes to enter the parameters can be concatenated. The results can then be displayed in a list box.

```
Figure 4
Private Sub Command1_Click()
Dim Arc As New ESRI.Arc
Dim results As New ESRIutil.Strings
Dim sev As Byte
sev = Arc.Command("imagegrid " &
Text1.Text & " " & Text2.Text,
results) * 200
Dim I As Integer
For I = 0 To results.Count - 1
List1.AddItem results.Item(I)  '—
Display the results of command
execution
Next I
End Sub
```

Next, automate the ArcInfo command PROJECT to describe the input projection and assign a new output projection. Figure 5 shows a portion of the code that prompts the user for the input and output projection parameters and puts these parameters into a list. To download the complete sample code, visit *ArcUser Online*.

The list of projection parameters is then executed using the PushString command. The Lost Focus procedure is used on the text boxes in order to skip parameters that are not needed for a particular projection.

```
Figure 5
Private Sub Command1_Click()
Dim Arc As New ESRI.Arc
Dim results As New ESRIutil.Strings
Dim sev As Byte
Dim j As Integer
For j = 0 To List2.ListCount
'— This is the total
' number of items in List
  Arc.PushString List2.List(j)
'— Returns the string for
'the specified list item.
Next j
sev = Arc.Command("project Grid "
 & Text20.Text & " "
 & Text21.Text, results) * 200
Dim I As Integer
For I = 0 To results.Count - 1
    List1.AddItem results.Item(I)
Next I
End Sub

Private Sub Command2_Click()
List2.AddItem "Input "
End Sub

Private Sub Command3_Click()
List2.AddItem "Output "
End Sub

Private Sub Command4_Click()
List2.AddItem "Parameters "
End Sub

Private Sub Command5_Click()
List2.AddItem "End "
End Sub

Private Sub Text11_LostFocus()
If Text11 = "" Then
  Exit Sub
Else
List2.AddItem "Projection " &
Text11
End If
End Sub

Private Sub Text12_LostFocus()
If Text12 = "" Then
  Exit Sub
Else
List2.AddItem "Units " & Text12
End If
End Sub

Private Sub Text13_LostFocus()
If Text13 = "" Then
  Exit Sub
Else
List2.AddItem "Zone " & Text13
End If
End Sub

Private Sub Text14_LostFocus()
If Text14 = "" Then
  Exit Sub
Else
List2.AddItem "Datum " & Text14
End If
End Sub

Private Sub Text15_LostFocus()
If Text15 = "" Then
  Exit Sub
Else
List2.AddItem "Spheroid " & Text15
```



*Load the ESRI ArcAutomation Server DLL and the ESRI Utility Automation Server DLL into Visual Basic by selecting the References from the Project menu.*

The final step is to automate the ArcInfo command GRIDIMAGE to convert the grid back to an image.

```
Figure 6
Private Sub Command1_Click()
Dim Arc As New ESRI.Arc
Dim results As New
ESRIutil.Strings
Dim sev As Byte
sev = Arc.Command("gridimage " &
 Text1.Text & " # " & Text2.Text
 & " TIFF", results) * 200
'— Specify image format in this
line
Dim I As Integer
For I = 0 To results.Count - 1
List1.AddItem results.Item(I)
Next I
```

This command requires an input grid file name, an output image file name, and an image type. Text boxes can be used for the input and output file names. The image type must be specified in the command line in the code itself. The results of the command execution can then be displayed in a list box. AU