Split Your Windows

Click & Retrieve Source CODE!

Add a Splitter class to your application to give it a Windows 95 Explorer-style splitter bar.

by Chris Barlow

In the last few columns I've looked at the RichTextBox, CommonDialog, ToolBar, and StatusBar controls that you can use to give your application that Windows 95 look and feel. If you're like me, you've gradually made the switch to using the Windows 95 environment, put away the File Manager, and begun to enjoy using the Windows Explorer.

One of the features I enjoy about the Windows Explorer, and many other Windows 95 applications, is the ability to use the splitter bar to change the width of the left and right windows. I always seem to drag the splitter bar to the right so I can see more of the left windows as I get into some long folder names, then drag it back to the left so I can see the document data in the right window.

When I first sat down with Visual Basic 4.0, I searched for a "splitter" control that would let me add this functionality to my applications—but no luck! Unfortunately there is no such control. You have to write code to resize each of the controls in the left and right windows. I've gotten so used to letting Visual Basic handle my user interface that I guess I've gotten lazy. It seemed like this functionality would take a lot of code in resize and mouse move events.

Then I began thinking that this functionality would fit nicely into a class. This way, I would only have to write the code once and I could easily reuse it in other projects. In this month's column I'll take you through the steps to design and create a class that lets you add a Windows 95-style window splitter to your VB4 applications.

CREATING THE SAMPLE APP

You need to create a sample application to test my class. Because this is Visual Basic, why not just sit down and start creating a form to experiment with? Start a new project and experiment with an application to display folders in the left window and documents in the right window (or, directories and files to readers still using pre-Windows 95 terminology).

Draw a CommandButton control on the form, make it tall and skinny with no caption, and name it "butSplit." This will be the splitter bar. To the left of the splitter bar draw a DriveListBox, under that draw a DirListBox. Finally, put a Label at the bottom and size these controls to fill up the left side of the form. On the right side of the splitter bar draw a Label control, add a FileListBox under it, add another label below the FileListBox, and size these controls to fill up the right side of the form.

If you've used the DriveListBox, DirListBox, and FileListBox, you remember that you need to add a few lines of code to link these controls together. In the DriveListBox Change event, set the DirListBox Path property to the Drive property:

```
Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub
```

Then, in the DirListBox Change event, set the FileListBox Path property. Using this code, put the path in the Label control so that the full path is displayed:

```
Private Sub Dir1_Change()
File1.Path = Dir1.Path
lbLeft = Dir1.Path
End Sub
```

In the FileListBox PathChange event, select the first file (if there is one):

```
Private Sub File1_PathChange()
If File1.ListCount Then File1.ListIndex = 0
End Sub
```

Finally, in the Click event, put the number of files in the top label control and the size of the selected file in the bottom label:

```
Private Sub File1_Click()
```

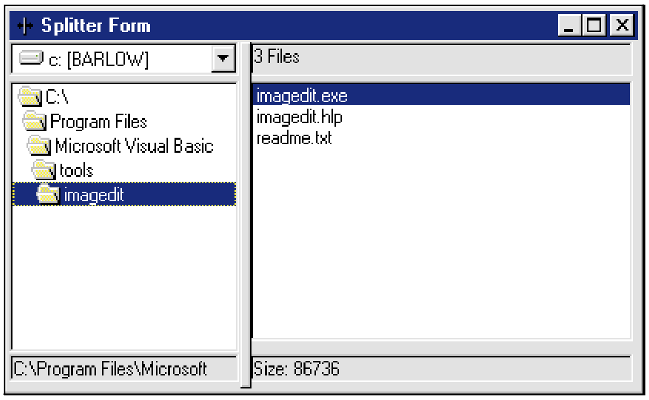

FIGURE 1 ***A Split Personality.*** *You can use your CSplitter class any time you need to divide a form into two windows.*

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications including the ObjectBank and the ObjectJob Systems, where he and Ken Henderson hold a software patent related to decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College where he worked with Drs. Kemeny and Kurtz on the BASIC language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.

```
lbList = File1.ListCount & " Files"
lbRight = "Size: " & FileLen(Dir1.Path & "\" & File1)
End Sub
```

When you run this application, you'll see a form that lets you select and display the documents in different folders and drives (see Figure 1).

DESIGNING THE CLASS

Now that you've got your form in place, it's time to design the CSplitter class. First you need to think about what you want to happen as you move the splitter bar.

Moving the bar 144 twips to the right adds 144 to the Width property of all the controls on the left side of the splitter bar. Controls on the right of the splitter bar should have 144 added to their Left property and 144 subtracted from their Width property.

Your class should have a Split method that changes these properties for all the affected controls. But how does the class know how far the splitter bar has moved? One way to convey this information is to create a Register method in the class that "registers" the splitter control with the class and saves certain information. This way, when the Split method was called by the application, you can easily get the current location of this splitter control.

Finally, you need a way to know what controls were affected by the Split method and whether they were on the left or right

VB4

```
Option Explicit
Private CurrentSplit As Integer
Private Splitter As Control
Private Splittees As New Collection
Private LStretch As String
'ID of stretchable control
Private RStretch As String
Private ContainerLeft As Integer
Private ContainerWidth As Integer
Private ContainerTop As Integer
Private ContainerHeight As Integer

Public Sub Split()
'splits to new left
Dim Diff%
Dim SC As CSplittee
Diff = (Splitter.Left + Splitter.Width \ 2) -
CurrentSplit
For Each SC In Splittees
  If SC.IsLeft Then
     SC.Ctrl.Width = SC.Ctrl.Width + Diff
  Else
     SC.Ctrl.Left = SC.Ctrl.Left + Diff
     SC.Ctrl.Width = SC.Ctrl.Width - Diff
  End If
Next
CurrentSplit = CurrentSplit + Diff
Set SC = Nothing
End Sub

Public Sub ChangeWidth(NewWidth As Integer)
'changes the width
Dim Diff%
Dim SC As CSplittee
Diff = ContainerWidth - NewWidth
For Each SC In Splittees
  If Not SC.IsLeft And SC.Ctrl.Width > Diff Then
'only change Right controls
     SC.Ctrl.Width = SC.Ctrl.Width - Diff
  End If
Next
ContainerWidth = NewWidth
Set SC = Nothing
End Sub

Public Sub ChangeHeight(NewHeight As Integer)
'changes the height
Dim Diff%
Dim SCL As CSplittee
Dim SCR As CSplittee
Dim SC As CSplittee
Diff = ContainerHeight - NewHeight
Splitter.Height = Splitter.Height - Diff
Set SCL = Splittees(LStretch)
If SCL.Ctrl.Height > Diff Then
' change left stretch control
     SCL.Ctrl.Height = SCL.Ctrl.Height - Diff
End If
Set SCR = Splittees(RStretch)
If SCR.Ctrl.Height > Diff Then
' change Right stretch controls
     SCR.Ctrl.Height = SCR.Ctrl.Height - Diff
End If
'now fix top of other controls
For Each SC In Splittees
  If SC.IsLeft Then
     If SC.Ctrl.Top > SCL.Ctrl.Top Then  'if below
        SC.Ctrl.Top = SC.Ctrl.Top - Diff
     End If
  Else
     If SC.Ctrl.Top > SCR.Ctrl.Top Then  'if below
        SC.Ctrl.Top = SC.Ctrl.Top - Diff
     End If
  End If
Next
ContainerHeight = NewHeight
Set SC = Nothing
Set SCL = Nothing
Set SCR = Nothing
End Sub

Public Sub Register(SplitControl As Control)
'register the control that acts as splitter
'saves the initial container dimensions
ContainerLeft = SplitControl.Parent.Left
ContainerTop = SplitControl.Parent.Top
ContainerWidth = SplitControl.Parent.ScaleWidth
ContainerHeight = SplitControl.Parent.ScaleHeight
SplitControl.Top = 0
SplitControl.Height = ContainerHeight
CurrentSplit = SplitControl.Left + _
  SplitControl.Width \ 2
Set Splitter = SplitControl
End Sub

Public Sub Add(ByVal Ctrl As Control, IsLeft As
Boolean, IsStretch As Boolean)
'adds a control object and save if the stretch control
Dim MySplittee As New CSplittee
MySplittee.IsLeft = IsLeft
Set MySplittee.Ctrl = Ctrl
Splittees.Add MySplittee, Ctrl.Name
If IsStretch Then
  If IsLeft Then
     LStretch = Ctrl.Name
  Else
     RStretch = Ctrl.Name
  End If
End If
Set MySplittee = Nothing
End Sub
```

LISTING 1 ***The CSplitter Class.*** *I've added the additional ChangeWidth and ChangeHeight methods to this listing. It's also available on the VBCD, in the Magazine Library (#3) of the* VBPJ *Forum on CompuServe, the* VBPJ *Development Exchange World Wide Web site, or the* VBPJ *site on The Microsoft Network.*

side of the splitter control. Let's call these controls "splittees." Because you will probably want to save additional information about these controls, create a simple CSplittee class to hold the controls and their added properties. Also, you can create an Add method for the CSplitter class that will create new instances of the CSplittee class and save them in a collection.

Start with the CSplittee class by inserting a class module into your project and adding this code:

```
Option Explicit
'contains info about a single control
Public IsLeft As Boolean
Public Ctrl As Control
```

Then right-click on the class module or press F4 to view the properties and change the name to CSplittee.

Now insert another class module, name it CSplitter, and add these properties:

```
Option Explicit
Private CurrentSplit As Integer
Private Splitter As Control
Private Splittees As New Collection
Private ContainerLeft As Integer
Private ContainerWidth As Integer
Private ContainerTop As Integer
Private ContainerHeight As Integer
```

The CurrentSplit property keeps the splitter control's current location that is saved in the Splitter property. The Splittees collection contains all instances of the CSplittee objects. You can make all these properties private because they'll only be accessed internally by the Split method. The ContainerLeft, ContainerWidth, ContainerTop, and ContainerHeight properties store the dimensional information about the form.

The Register method will have a single argument to pass the control used by the application as the splitter control—in this

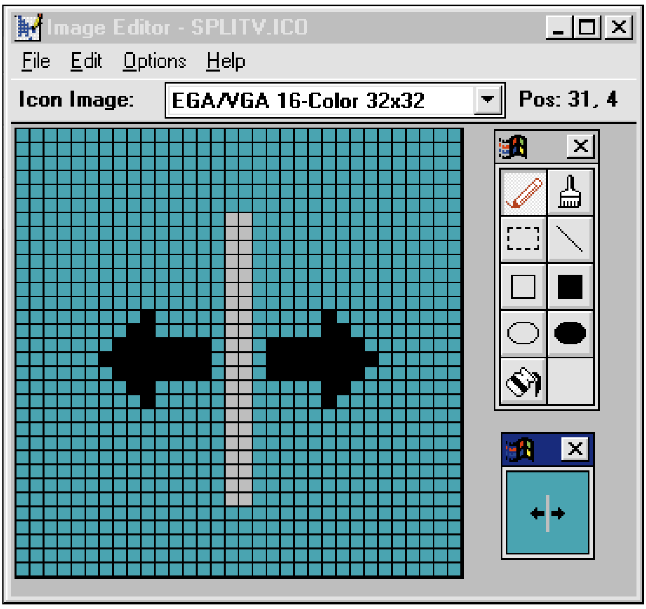

FIGURE 2 ***XDesign Your Own Pointer.*** *Use VB's ImageEdit tool to design a custom pointer that replaces the regular pointer when the user moves the mouse over the splitter bar. The ImageEdit application comes on the Visual Basic CD-ROM in the Tools folder—don't forget to copy it to your hard drive.*

case, a CommandButton control set to a very narrow width. In the Register method you will want to set the Splitter property of your class to the control that will be used as the splitter control. This is also a good place to save the initial dimensions of the form and to size the splitter control to the full form height:

```
Public Sub Register(SplitControl As Control)
'register the control that acts as
'splitter
'saves the initial container dimensions
ContainerLeft = SplitControl.Parent.Left
ContainerTop = SplitControl.Parent.Top
ContainerWidth = _
   SplitControl.Parent.ScaleWidth
ContainerHeight = _
   SplitControl.Parent.ScaleHeight
SplitControl.Top = 0
SplitControl.Height = ContainerHeight
CurrentSplit = SplitControl.Left + _
   SplitControl.Width \ 2
Set Splitter = SplitControl
End Sub
```

Note that the CurrentSplit property adds half the width of the splitter control to its left property to store the middle of the splitter control. Also note that the ScaleWidth and ScaleHeight properties of the form are used because these return the interior dimensions of the form.

One thing I've learned when creating classes is to test the code as you write it. Now go back to the Form_Load event and add the code to call this Register method. You also need to create an instance of the CSplitter class, called MySplitter, for your form:

```
Option Explicit
Dim MySplitter As New CSplitter

Private Sub Form_Load()
MySplitter.Register butSplit
End Sub
```

Now single-step through this code and make sure your Register method works. Stop the program before it returns from the Register method and use the Debug Window to look at the properties of the Splitter object. Try typing "Print Splitter.Left" in the Debug Window and you should see the value of the Left property of the butSplit control.

KEEPING TRACK OF CONTROLS

Now create the Add method of the CSplitter class that will create new instances of the CSplittee class and store them in the Splittees collection. You'll want to pass two arguments with the Add method—the control to be added and a Boolean value, IsLeft, to indicate whether this control is on the left or right of the splitter bar. Note that, for simplicity, you can use the control name as the unique index for the collection. But if you want to split control arrays, you'll need to expand this to add the control's Index property. Note that the last line of code sets the MySplittee object back to Nothing—always do your housekeeping!

```
Public Sub Add(ByVal Ctrl As Control, IsLeft As Boolean)
Dim MySplittee As New CSplittee
MySplittee.IsLeft = IsLeft
Set MySplittee.Ctrl = Ctrl
Splittees.Add MySplittee, Ctrl.Name
```

```
Set MySplittee = Nothing
End Sub
```

Time for more testing. Add code to the Form_Load event to call the Add method of the CSplitter class to add each of the form's controls to the CSplitter class:

```
MySplitter.Add Drive1, True
```

VB4

```
Option Explicit
Dim MySplitter As New CSplitter
Private Splitting As Boolean

Private Sub butSplit_MouseDown(Button As Integer, _
  Shift As Integer, X As Single, Y As Single)
'start capturing for move
Splitting = True
End Sub

Private Sub butSplit_MouseMove(Button As Integer, _
  Shift As Integer, X As Single, Y As Single)
'move and resize all
If Splitting Then butSplit.Left = X + _
  butSplit.Left - (butSplit.Width \ 2)
End Sub

Private Sub butSplit_MouseUp(Button As Integer, _
  Shift As Integer, X As Single, Y As Single)
'release resize
MySplitter.Split
Splitting = False
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
lbLeft = Dir1.Path
End Sub

Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub

Private Sub File1_Click()
lbList = File1.ListCount & " Files"
lbRight = "Size: " & FileLen(Dir1.Path & "\" & _
  File1)
End Sub

Private Sub File1_PathChange()
If File1.ListCount Then File1.ListIndex = 0
End Sub

Private Sub Form_Load()
'setup splitter
MySplitter.Register butSplit
MySplitter.Add Drive1, True, False
MySplitter.Add lbList, False, False
MySplitter.Add Dir1, True, True
MySplitter.Add File1, False, True
MySplitter.Add lbLeft, True, False
MySplitter.Add lbRight, False, False

End Sub

Private Sub Form_Resize()
MySplitter.ChangeWidth ScaleWidth
MySplitter.ChangeHeight ScaleHeight
End Sub
```

LISTING 2 ***The Splitter Application.*** *Use this code to test your new CSplitter class. It's also available on the VBCD, in the Magazine Library (#3) of the* VBPJ *Forum on CompuServe, the* VBPJ *Development Exchange World Wide Web site, or the* VBPJ *site on The Microsoft Network.*

```
MySplitter.Add lbList, False
MySplitter.Add Dir1, True
MySplitter.Add File1, False
MySplitter.Add lbLeft, True
MySplitter.Add lbRight, False
```

Now set a breakpoint on the last line and single-step through your Add method. When you get to the end of the procedure, before exiting the routines, go to the Debug Window and prove to yourself that these controls have really been saved within your class by typing these "print" lines and you will see the count property of the splittees collection and the name property of the third splittee control:

```
print splittees.count
 6
print splittees(3).ctrl.name
Dir1
```

DO THE SPLIT!

You're ready to write the Split method. First calculate how much the splitter control has moved by comparing the Left property to the CurrentSplit property you saved. You'll want to step through the Splittees collection of CSplittee objects and check the IsLeft property. If the IsLeft property is True, the control is on the left of the form and you need to adjust the width of the control. If the control is on the right, then you'll need to adjust both the Left and the Width properties of the control.

Notice how you can use the new For Each construct to step through the collection of CSplittee objects. Simply define a variable, SC, as a CSplitee object. Each iteration through the loop assigns the next CSplittee object to the SC variable. Then you can change the Width property of the Ctrl object within the SC object by referring to SC.Ctrl.Width:

```
Public Sub Split()
Dim Diff%
Dim SC As CSplittee
Diff = (Splitter.Left + Splitter.Width \ 2) - CurrentSplit
For Each SC In Splittees
   If SC.IsLeft Then
      SC.Ctrl.Width = SC.Ctrl.Width + Diff
   Else
      SC.Ctrl.Left = SC.Ctrl.Left + Diff
      SC.Ctrl.Width = SC.Ctrl.Width - Diff
   End If
Next
CurrentSplit = CurrentSplit + Diff
Set SC = Nothing
End Sub
```

To use the Split method, you have to add a little code to the form to track the mouse events over the splitter control. Add a private Boolean variable, called Splitting, to flag when the user is dragging the splitter control. Then, in the MouseDown event of the splitter control, set the Splitting variable to True when the user presses the mouse over the butSplit control:

```
Private Splitting As Boolean

Private Sub butSplit_MouseDown(Button _
   As Integer, Shift As Integer, X _
   As Single, Y As Single)
'start capturing for move
Splitting = True
```

```
End Sub
```

In the MouseUp event, call the Split method of the MySplitter object and set the Splitting variable to False:

```
Private Sub butSplit_MouseUp(Button _
   As Integer, Shift As Integer, _
   X As Single, Y As Single)
'release resize
MySplitter.Split
Splitting = False
End Sub
```

Finally, add some code under the MouseMove event to move the butSplit control as the user moves the mouse:

```
Private Sub _
   butSplit_MouseMove(Button _
   As Integer, Shift As Integer, _
   X As Single, Y As Single)
'move and resize all
If Splitting Then butSplit.Left _
   = X + butSplit.Left - _
   (butSplit.Width \ 2)
End Sub
```

When you run your application and drag the splitter control, you should see each window resize. Neat! But notice right away that the default mouse pointer does not give the user any visual feedback that they can use this control to "split" the window. You can use the MousePointer property of the butSplit control to choose a different mouse pointer, but none of the standard mouse pointers match the mouse pointer normally used in the Windows Explorer and other applications. Not a problem, though—you can set the MousePointer property to "Custom" and load any icon into the MouseIcon property. I loaded Visual Basic's ImageEdit program (you'll find it in the Tools folder on the Visual Basic CD-ROM) and quickly created my own icon to imitate the standard splitter mouse pointer (see Figure 2).

To make a more complete class, you would probably want to add a few more methods. For example, if the form is resized so that the Width changes, then the width of all the controls on the right side of the form need to be adjusted. You could easily handle this by adding a ChangeWidth method to your CSplitter class. I included this method in the code listing and sample code file (see Listing 1).

Resizing the height of the form is a little more complex because you probably do not want the height of all the controls to change proportionally. Typically, only one of the controls on each side of the splitter control "stretches" to fit the changed form height and the other controls maintain their relative positions. You'll need to let the application developer who will be using this class specify which control can "stretch" on each side of the form in the Add method of the CSplitter class. Then the ChangeHeight method can adjust the height of these controls and relocate the other controls to fit. I've included the additional properties and methods in the code listing, and I've also included the test application (see Listing 2).

You can see that it is not difficult to create your own reusable class. Now you won't have to write this code again. When you need the splitter functionality, just include the CSplitter class and you're all set. The code discussed in this column, contained in a file called CSPLIT.ZIP, is available on FTP's online sites as well as on the VBCD.