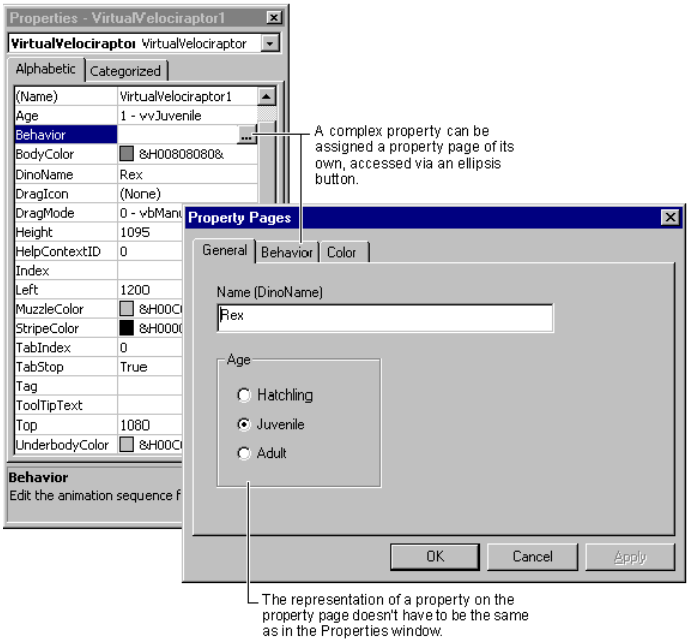
Property pages offer an alternative to the Properties window for viewing ActiveX control properties. You can group several related properties on a page, or use a page to provide a dialog-like interface for a property that's too complex for the Properties window.

In Visual Basic, property pages are displayed in the Property Pages tabbed dialog box, as shown in Figure 10.1.

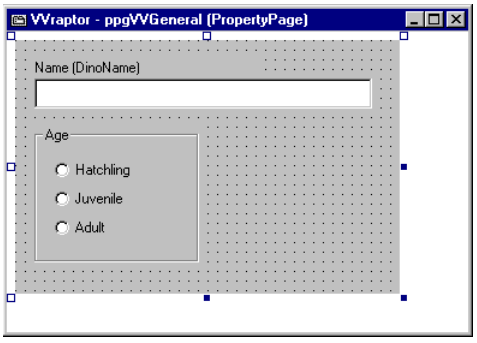
Figure 10.1 The Properties window and the Property Pages dialog box



1

Each tab in the Property Pages dialog box represents one PropertyPage object. For example, the PropertyPage object that provides the General tab shown in Figure 10.1 was created using the PropertyPage designer shown in Figure 10.2.

Figure 10.2 The PropertyPage designer for the General tab



2

In the designer, the PropertyPage object doesn't display a tab. Nor does it display the OK, Cancel, and Apply buttons. These are provided automatically by the Property Pages dialog, and are not part of any individual PropertyPage object. The Property Pages dialog uses the Caption property of the PropertyPage object as the text for the tab.

In this example, the General page groups two properties, DinoName and Age. The page is laid out to simplify localization; the length of the captions can change without affecting the layout. This is discussed in "Property Page Design Guidelines."

The following topics explain how property pages work, and the different ways you can use them in your ActiveX controls.

Contents

1

For More Information A simple property page example can be found in the step by step procedures in "Creating an ActiveX Control." Control creation is covered in depth in "Building ActiveX Controls." The Property Page Wizard can automate much of the work of creating property pages.

2

How Property Pages Work

Property pages look a lot like forms, and designing them is somewhat similar to designing forms. The way property pages work, however, is quite different from the way forms work.

For example, when the Property Pages dialog box creates an instance of a property page, the Initialize event is the first event the PropertyPage object gets — just as it would be for a form. However, unlike a form, the PropertyPage object doesn't get a Load event. The key event for PropertyPage objects is the SelectionChanged event.

This topic examines the three things your PropertyPage object must do:

- In the SelectionChanged event, obtain the property values to be edited.
- Set the PropertyPage object's Changed property whenever the user edits a property value.
- In the ApplyChanges event, copy the edited property values back to the selected control (or controls).

3

The SelectionChanged Event

The SelectionChanged event occurs when the property page is displayed, and when the list of currently selected controls changes.

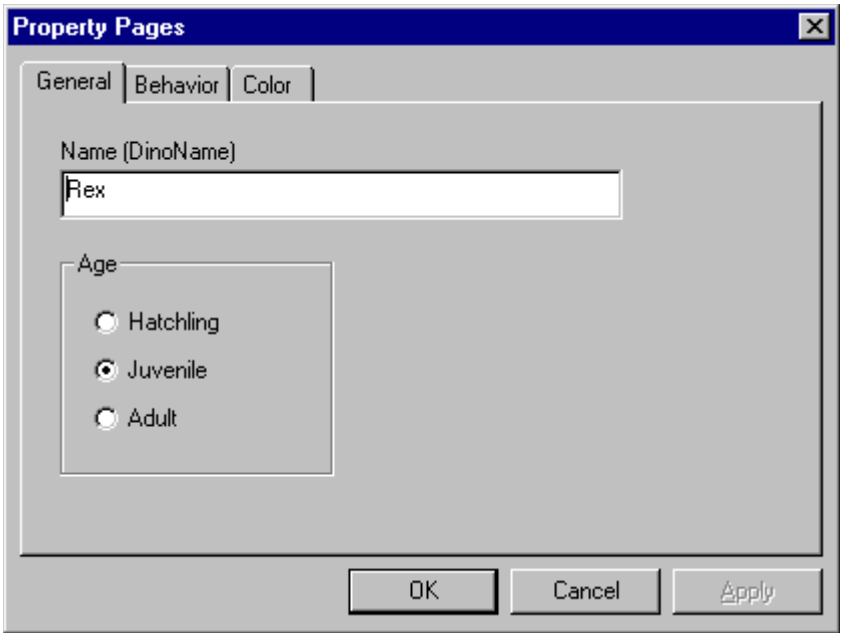
For example, after selecting an instance of your control and opening the Property Pages dialog box, a developer might realize that she needed to change the properties of two instances of your control. By clicking the second instance while holding down the CTRL key, she could add the second instance to the list of selected controls. Each of your property pages would then receive a SelectionChanged event.

Important You should always treat the SelectionChanged event as if your property page is being loaded for the first time. As you'll see, changing the selection fundamentally changes the state of the property page.

3

Coding the SelectionChanged Event for a Single Control

The most important thing you need to do in the SelectionChanged event is to set the values of the controls that display property values for editing. For example, consider the General page for the VirtualVelociraptor control (originally shown in Figure 10.1):



4

Suppose that the Age property of the VirtualVelociraptor control uses the following public Enum:

```
Public Enum DinoAge
    vvHatchling
    vvJuvenile
    vvAdult
End Enum
```

4

The SelectionChanged event of the property page might look like this:

```
Private Sub PropertyPage_SelectionChanged()
    ' Place the value of the DinoName property for the
    '   first selected control in the txtDinoName text
    '   box for display and editing.
    txtDinoName = SelectedControls(0).DinoName
    ' Use the value of the Age property of the first
    '   selected control to select the appropriate
    '   option button in the Age frame.
    optAge(SelectedControls(0).Age).Value = True
    ' (The code above depends on the fact that the
    '   elements of the DinoAge Enum have the values
    '   0, 1, and 2.)
End Sub
```

5

Tip The Property Page Wizard will populate your property page with text box controls and check boxes (for Boolean properties), and generate default code for the SelectionChanged event.

6

The SelectedControls Collection

The SelectedControls collection contains all the controls currently selected in the container the developer is working on. The collection may contain several instances of your control; if the property page is shared by more than one control in your control component, the collection may contain controls of multiple types.

Note You don't need to worry about the collection containing controls other than your own — text boxes, for example — because the Property Pages dialog box only displays those pages that are used by *all* of the currently selected controls.

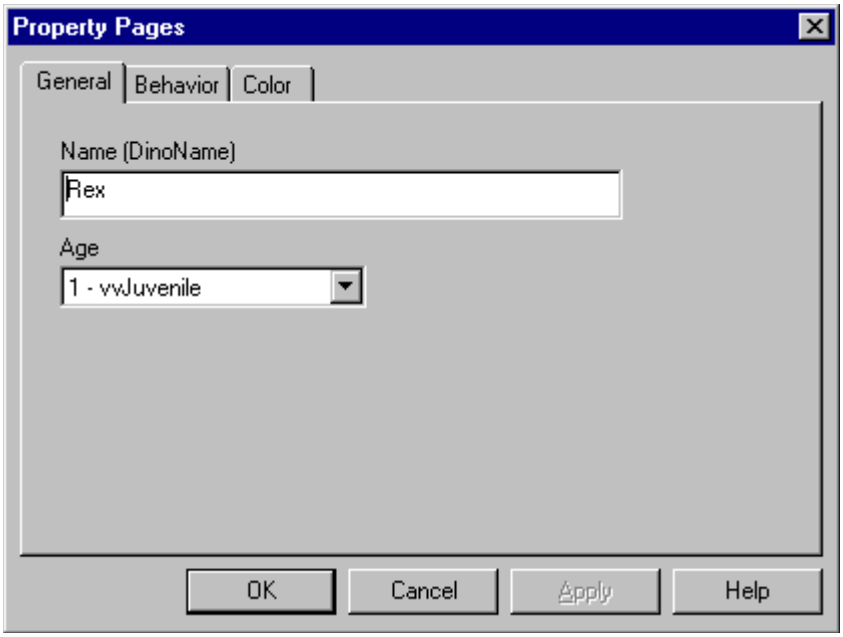
7

For the moment, ignore the possibility that multiple controls might be selected. What the code in the SelectionChanged event shown above is doing is taking the value of each property for the *first* control in the collection and assigning it to the appropriate control on the property page.

In the case of a single selected control, this places all of the control's property values in fields where the user can edit them.

Different Ways to Edit Properties

Instead of showing the property value of the Age property as a set of option buttons, you could use a drop-down list showing the elements of the enumeration:



5

The drop-down list takes up less space than the option buttons did (an advantage that grows larger as the number of possible values increases), and it shows the names of the constants that would be used in code.

The following code fragment shows how you might set up such a list.

```
Private Sub PropertyPage_SelectionChanged()
    txtDinoName = SelectedControls(0).DinoName
    ' Create a drop-down list containing the values and
    '   names of the Enum elements for the Age
    '   property, and select the one that corresponds
    '   to the current value of the Age property.
    cboAge.AddItem vvHatchling & " - vvHatchling"
    cboAge.AddItem vvJuvenile & " - vvJuvenile"
    cboAge.AddItem vvAdult & " - vvAdult"
    cboAge.ListIndex = SelectedControls(0).Age
    ' (The index of each Enum element in the drop-down
    '   list is the same as the element's value.)
End Sub
```

8

Tip While you can choose any editable representation that makes sense for a property, remember that the more space each property takes up, the more tabs you'll need. Minimizing the number of tabs makes the property pages for your control easier to use. For most enumerations, a drop-down list will make the most efficient use of space.

9

Coding the SelectionChanged Event for Multiple Controls

To determine whether multiple controls are selected, you can test the Count property of the SelectedControls collection to see whether it's greater than one.

In order to deal with multiple selected control instances, it's useful to divide the properties of your control into two groups:

- Properties that can sensibly be set to the same value for multiple controls. For example, it's very convenient to be able to set the BackColor property of several Label controls to the same value.
- Properties that do not make sense to set to the same value for multiple controls. For example, it's not particularly helpful to set the Caption property of several Label controls to the same value. In fact, it might be quite annoying to the user to do so by accident.

6

One approach you might take in your SelectionChanged event is to disable the edit fields for properties of the second sort whenever multiple controls are selected. In the discussion of the ApplyChanges event, an alternate technique will be shown.

Shared Property Pages

If you have multiple controls in your project, and two such controls share a property page, make sure that you provide error trapping for the code that reads the property values. If the first control selected doesn't include all of the properties shown on the page, an error will occur when you try to read that property value.

Enabling the Apply Button by Setting Changed = True

In order to tell Visual Basic that the user has edited one or more properties on a property page, you must set the PropertyPage object's Changed property to True. Because there's no way to know which property a user might decide to change, you must do this for every property displayed on the page.

For example, to notify the PropertyPage of changes in the DinoName or Age properties from the previous example, you would use the following code:

```
Private Sub txtDinoName_Changed()
    Changed = True
End

Private Sub cboAge_Change()
    Changed = True
End
```

10

Note that this is exactly the same as coding PropertyPage.Changed = True.

Notifying the PropertyPage object that values have changed enables the Apply button on the Property Pages dialog box, and causes the ApplyChanges event to occur when the Apply button is pressed, when the user changes tabs, or when the dialog box is dismissed.

Note You may wish to keep track of which properties have changed, so that you don't have to write them all out.

11

The ApplyChanges Event

The second most important event in a PropertyPage object is the ApplyChanges event. In this event you copy the edited property values back to the currently selected controls.

The ApplyChanges event occurs when the user:

- Clicks the OK button to dismiss the dialog.
- Clicks the Apply button.
- Selects another tab in the Property Pages dialog box.

7

The following code for the ApplyChanges event assumes that the SelectionChanged event was coded using a drop-down list for the Age property, as shown earlier.

```
Private Sub PropertyPage_ApplyChanges()
    Dim vv As VirtualVelociraptor
    ' Set the DinoName property of the FIRST selected
    '   control only.
    SelectedControls(0).DinoName = txtDinoName

    For Each vv In SelectedControls
        ' Transfer the value currently selected in the
```

—7

```
            '   drop-down list for the DinoAge property to
            '   all of the selected controls.
            vv.DinoAge = cboAge.ListIndex
            ' (The code above works because the value of
            '   each of the three elements of the Enum is
            '   the same as its index number in cboAge.)
        Next
End Sub
```

Because it generally doesn't make sense to give all of the virtual velociraptors the same name, the DinoName property is applied only to the first selected control. The Age property, on the other hand, is applied to all the selected controls.

Note As the control author, it's up to you to decide which properties make sense to set for multiple selected controls.

Dealing with Errors in ApplyChanges

In the case shown above, there's no chance of error in the ApplyChanges event. The text property is a simple string, and the drop-down list limits user input for the Age property to only those values that are valid.

If your property page allows the user to enter values that may be rejected by the Property Let (or Property Set) procedure, you should use error trapping in the ApplyChanges event. The simplest scheme is to use On Error Resume Next, and test Err.Number after each property that may raise an error.

When an error occurs:

- Stop processing the ApplyChanges event.
- Display an error message, so the user understands what went wrong.
- Set the focus to the property that caused the error.
- Set the Changed property of the PropertyPage object to True.

  Important Setting Changed = True performs two functions. First, it re-enables the Apply button. Second, it prevents the Property Pages dialog box from being dismissed if the user clicks OK. *This is the only way to prevent the dialog box from closing*.

Connecting a Property Page to an ActiveX Control

Once you've added property pages to your ActiveX control project, you can use the Connect Property Pages dialog box to establish a connection between a control and the property pages you want it to use.

When a user shows the Property Pages dialog box for an instance of one of your controls, each of the pages you connected to the control will appear as one tab in the dialog box.

To connect property pages to a control

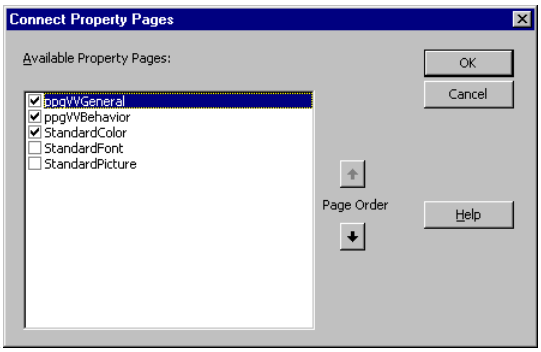
1 In the **Project** window, double-click the control to open its designer.

2 Press F4 to open the **Properties** window.

3 Double-click the **PropertyPages** property (or select the property and then click the ellipsis button) to open the **Connect Property Pages** dialog box.

4 Place a check mark beside each property page you want to appear when the developer who uses your control opens the **Property Pages** dialog box.

5 Use the **Page Order** buttons to put the pages in the order you want them to appear in the **Property Pages** dialog box, then click **OK**.



1 10

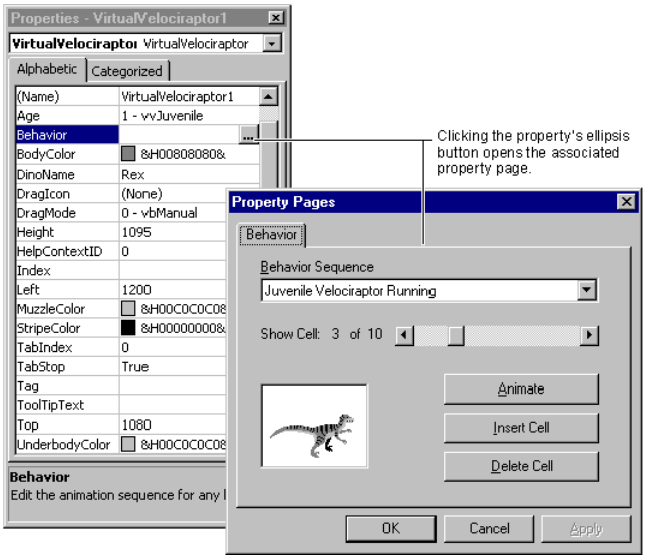
For More Information The standard property pages for fonts, colors, and pictures are discussed in "Using Standard Property Pages."

Associating a Property Page with a Property

Sometimes a property is too complicated to set from the Properties window. A property might be an object, such as the Font object, with properties of its own. A property might consist of an array of values, or even a collection of objects — such as a collection of Toolbar buttons.

For example, Figure 10.3 shows how a property page might look for the Behavior property of a hypothetical VirtualVelociraptor control. This page allows cells to be added to and deleted from animation sequences for the control's various behaviors.

Figure 10.3 A property too complex for the Properties window



11

Notice that only the tab associated with the property is displayed. If you compare this picture with Figure 10.1, you'll notice that the page size used by the Property Pages dialog box is smaller in this picture. When multiple pages are displayed, the size of the largest page is used.

Note If you declare a property of type Font, OLE_COLOR, or Picture, Visual Basic will automatically associate the property with a StandardFont, StandardColor, or StandardPicture page, as described in "Standard Property Pages."

14

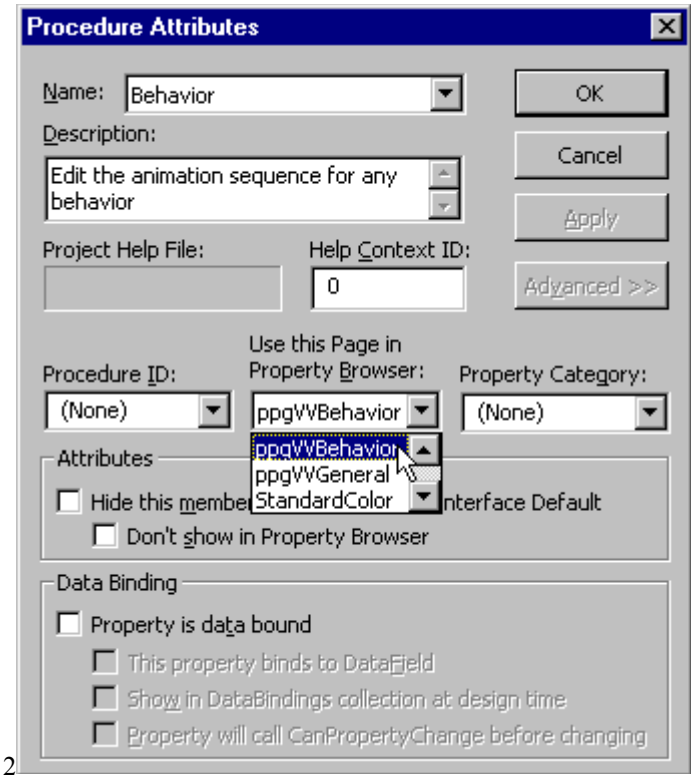
To associate a property page with an individual property

6 In the **Project** window, right-click the **UserControl** to open the context menu, and select **View Code** to open the code window.

7 On the **Tools** menu, select **Procedure Attributes** to open the **Procedure Attributes** dialog box, and click **Advanced** to expand the dialog box.

8 In the **Name** box, select the property you wish to associate with a property page.

9 Select the desired property page in the **Use this Page in Property Browser** list, as shown below, then click **Apply** or **OK**.



2

12

The EditProperty Event

Visual Basic allows you to associate multiple properties with the same property page. You might want to do this if your control has more than one property that uses the same property page layout, or if one property uses part of another property's layout. In the latter case, you can use the EditProperty event to enable only the necessary parts of the property page.

When the user clicks the ellipsis button for a property that's associated with a property page, the page receives the EditProperties event in addition to the events it normally receives. You can use the PropertyName argument of the EditProperties event to identify the property whose ellipsis button was clicked.

The EditProperties event gives you the opportunity to do a number of things to the property page, depending on the nature of your control and the complexity of the property being edited. For example, you might:

- If the property being edited is displayed in a single control on the PropertyPage, set the focus to that control.
- Enable and disable controls on the property page so that only those fields applicable to the specified property are enabled.

13

Using Standard Property Pages

Visual Basic provides three standard property pages: StandardFont, StandardColor, and StandardPicture. If you declare properties of type Font, OLE_COLOR, or Picture, Visual Basic's Properties window will automatically associate these properties with the appropriate standard property page.

Visual Basic will not automatically add these pages to the Property Pages dialog box, however. Use the procedure shown in "Connecting a Property Page to an ActiveX Control" to add standard property pages to the list of pages that will be displayed in the Property Pages dialog box.

Note The format used to display standard property pages in the Property Pages dialog box differs from the format used by the Properties window. For example, the Color page is in an entirely different format, and the Font page does not display all Font object properties.

15

Tip If you use the Property Page Wizard to create property pages for a control with properties declared Font, OLE_COLOR, or Picture, the wizard will automatically add the appropriate standard property pages to the list of pages to be displayed.

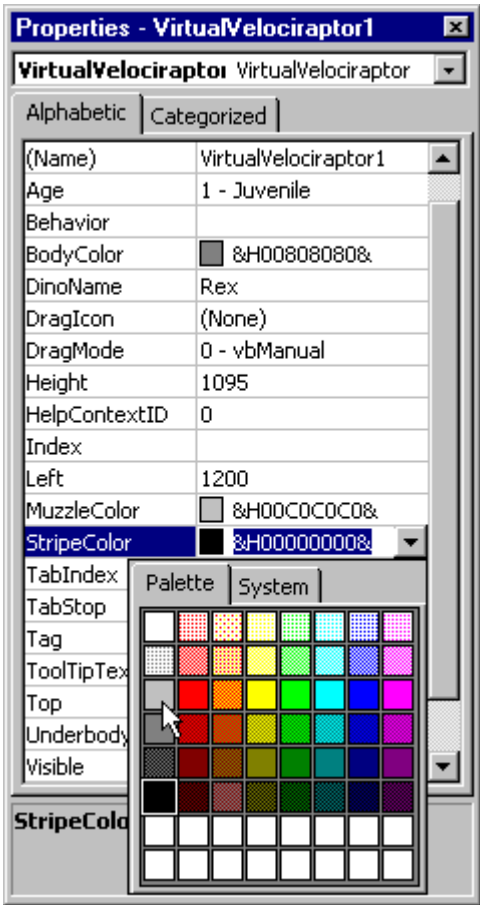
16

Standard Property Pages and Multiple Properties

If your control has more than one property that uses a standard property page, and you add the page to your control's PropertyPages property, the standard page will include a list of properties the user can select from.

For example, Figure 10.4 shows the Properties window and Property Pages dialog box for the hypothetical VirtualVelociraptor control, which has several properties of type OLE_COLOR:

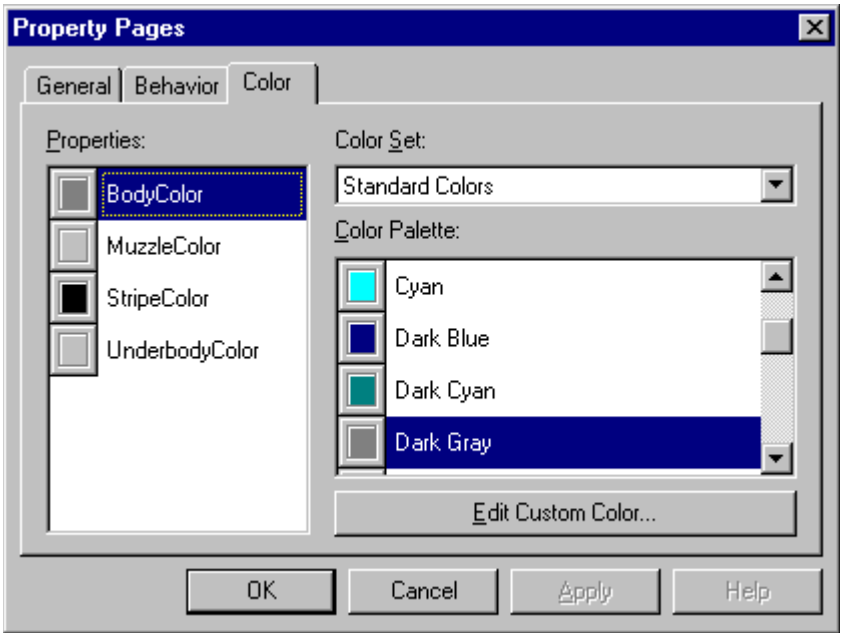
Figure 10.4 The Properties window for a control with multiple color properties



14

As Figure 10.5 shows, the Color page displayed by the Property Pages dialog box for the VirtualVelociraptor control has a list box containing the four color properties of the control.

Figure 10.5 A standard property page showing multiple properties



15

Figure 10.5 also shows that the Property Pages dialog box uses a very different format than that used by the Properties window.

Note When a standard property page displays multiple properties, the ApplyChanges event is raised each time the user selects a different property from the list.

17

The following code fragment shows how the StripeColor property of the hypothetical VirtualVelociraptor control should be declared in order to work with the Properties window and Property Pages dialog box:

```
Private mStripeColor As OLE_COLOR

Public Property Get StripeColor() As OLE_COLOR
    StripeColor = mStripeColor
End Property

Public Property Let StripeColor( _
        ByVal NewColor As OLE_COLOR)
    mStripeColor = NewColor
End Property
```

18

For More Information See “Adding Properties to Controls” in “Building ActiveX Controls.”

Property Page Design Guidelines

You may find the following guidelines helpful in designing property pages that are professional-looking and easy to use.

- When the property pages for your control are shown, focus should be on the first field of the first tab. To do this, give the first control on each PropertyPage the lowest TabIndex number for the page.
- If properties are arranged in multiple columns on a page, tab order should usually be top to bottom in the first column, then move to the top of the next column.
- Provide access keys (keyboard shortcuts) for all the fields on a property page.

  1**Note** The letter 'A' is not available as an access key, because the Property Pages dialog box uses it for the Apply button.

  16

- Place standard property pages for fonts, colors, and pictures at the end of the list of property pages for a control.
- Group similar properties together — for example, all properties that affect a control's appearance should go on the same page.
- Use a uniform size for your property pages. When the Property Pages dialog box displays multiple pages, it uses the same width and height for all of them.

  1**Tip** The standard pages for fonts, colors, and pictures are all the same size. If you use these pages, set your other property pages to this standard size. A quick way to set the standard size is to set the StandardSize property of the PropertyPage to Large at design time.

  17

- For ease of use, keep the number of tabs to a minimum. For example, don't add a tab for your control's About box.
- Most control components (.ocx files) that provide more than one control have a page with the caption "General" which contains items that are common to many controls.
- Keep your pages simple and fast. Your users will be generally be developers employing your control to get a job done — animation and fancy graphics will just get in their way.
- The Property Pages dialog box will show a Help button if you've used the General tab of the Project Options dialog box to add a Help file to your project. You can set the HelpContextID properties of your property pages and of the controls they contain.
- When you add labels to your property pages, make sure you include the property name in the label. You may want to localize your control for international use, or add user-friendly captions for the benefit of desktop tools such as Microsoft Office, but remember that the user must be able to identify the actual property names in order to write code for them, or look them up in your Help file.

- When you create strings for the named constants in an enumeration, always include the actual constant name, because that's what the person using your control will have to use when programming.
- If you plan to localize your control component for international use, make sure that combo boxes, text boxes, and so on are wide enough to accommodate translated strings. If you plan to localize for locales — such as Japan — that use DBCS characters, make sure the font you're using is available on DBCS systems, and that the height of your text boxes will accommodate DBCS characters.
- Avoid showing dialog boxes from property pages. (The File Open dialog is an exception to this rule.)
- Cross-tab property dependencies. If two properties interact — for example, if setting the value of one property limits the valid values for another property — place the properties on the same page.

18

For More Information Rules for layout that simplifies localization can be found in "International Issues."