Create a Toolbar from within Visual Basic 6

Most beginner Visual Basic programmers I teach in my classes are very excited when they finally add a menu to their projects---to them, a menu is a sign of a professional looking program. You can take this professional look one step further by adding a toolbar to your form.

Creating a Toolbar in Visual Basic is a multistep process, and I'll be discussing each one in detail.

First, you need to add the Microsoft Toolbar control to your Toolbox, and then place it on the form. A Visual Basic toolbar, as you'll see in just a bit, is actually made up of one or more individual 'buttons', which normally contain a picture, and optionally some text.

Second, you need to add an ImageList control to your Toolbox, and then place it on the form.

Third, you need to add 'images' to the ImageList Control that will appear in your Toolbar.

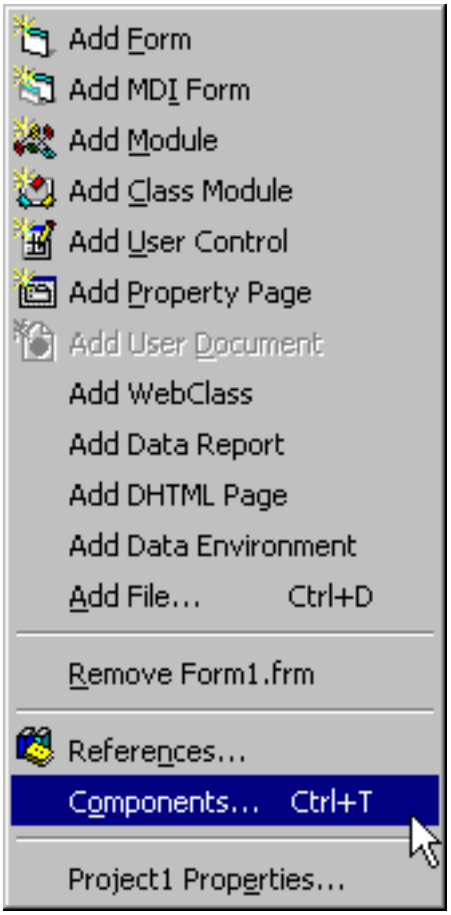
Fourth, you need to set some properties in the Toolbar control that affect the number of buttons that will appear on the toolbar, as well as the source for the icons that will appear in it (an image in the ImageList Control).

Finally, once you have the Toolbar looking the way it should, you need to place some code in the Click event procedure of the Toolbar to perform the actions you wish.

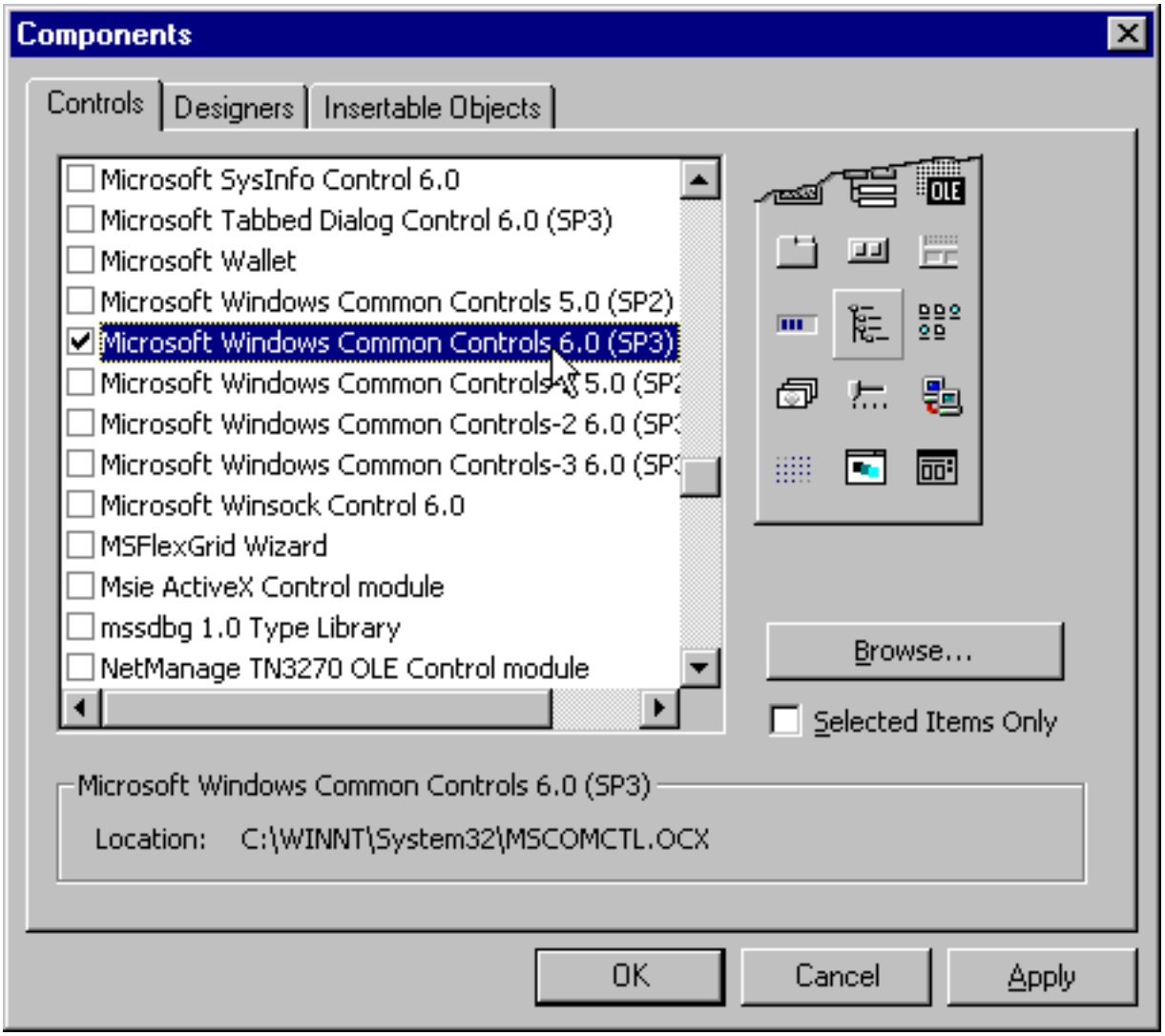
Let's look at these steps now in detail.

Add the Toolbar Control to your form

Our first step is to add the Microsoft Toolbar control to our form. First, we must select Project-Components from the Visual Basic menu bar…



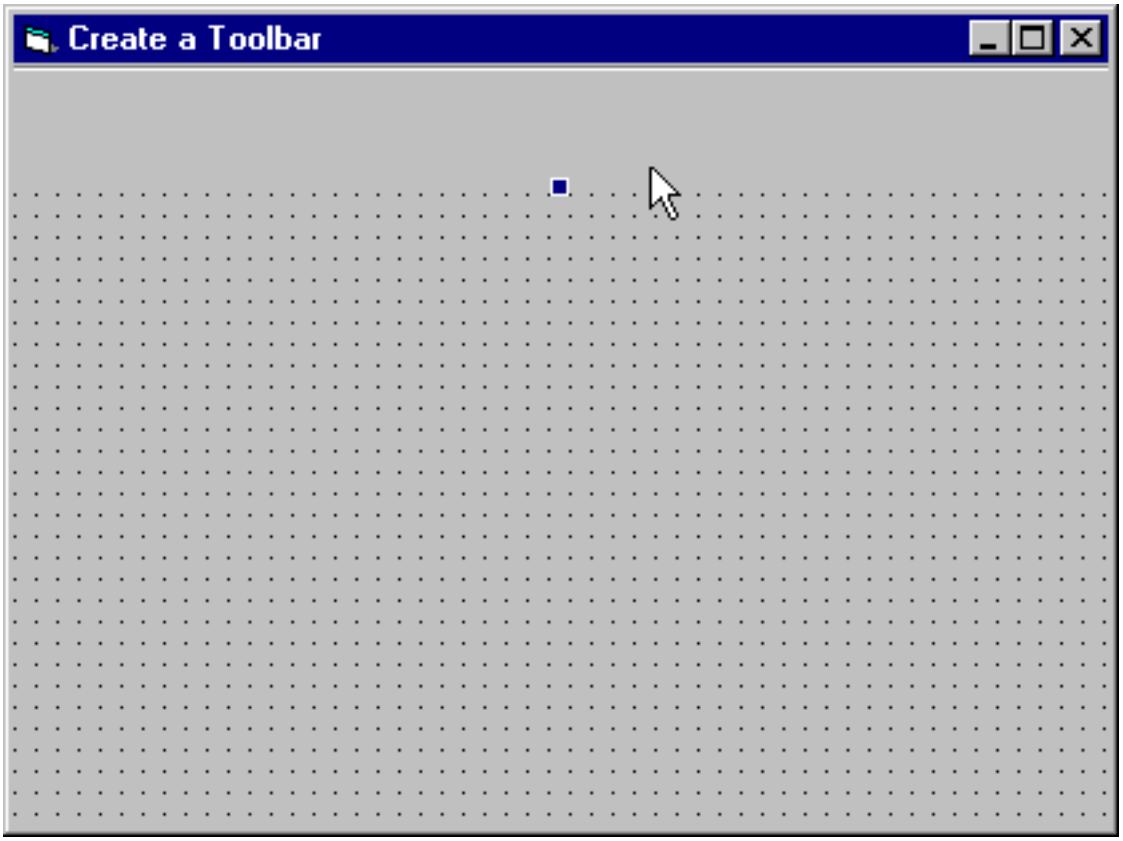
and then look for the Toolbar control. The problem is you won't find a control listed in the Components Window that says anything about a Toolbar. The toolbar control is bundled in the Microsoft Windows Common Controls (MSCOMCTL.OCX)…



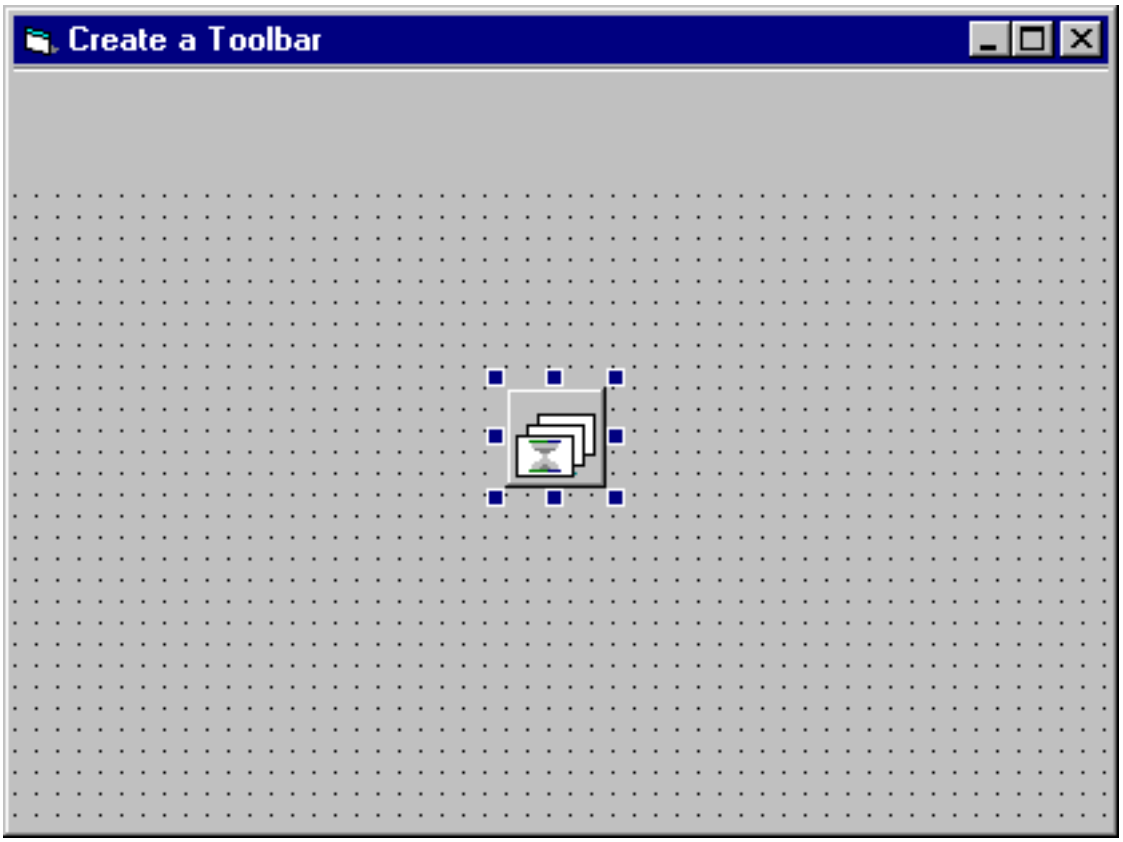
If you select this OCX, you'll see several new controls appear in your Visual Basic Toolbox. One of them is the Toolbar control…

…and one of them is the ImageList Control…

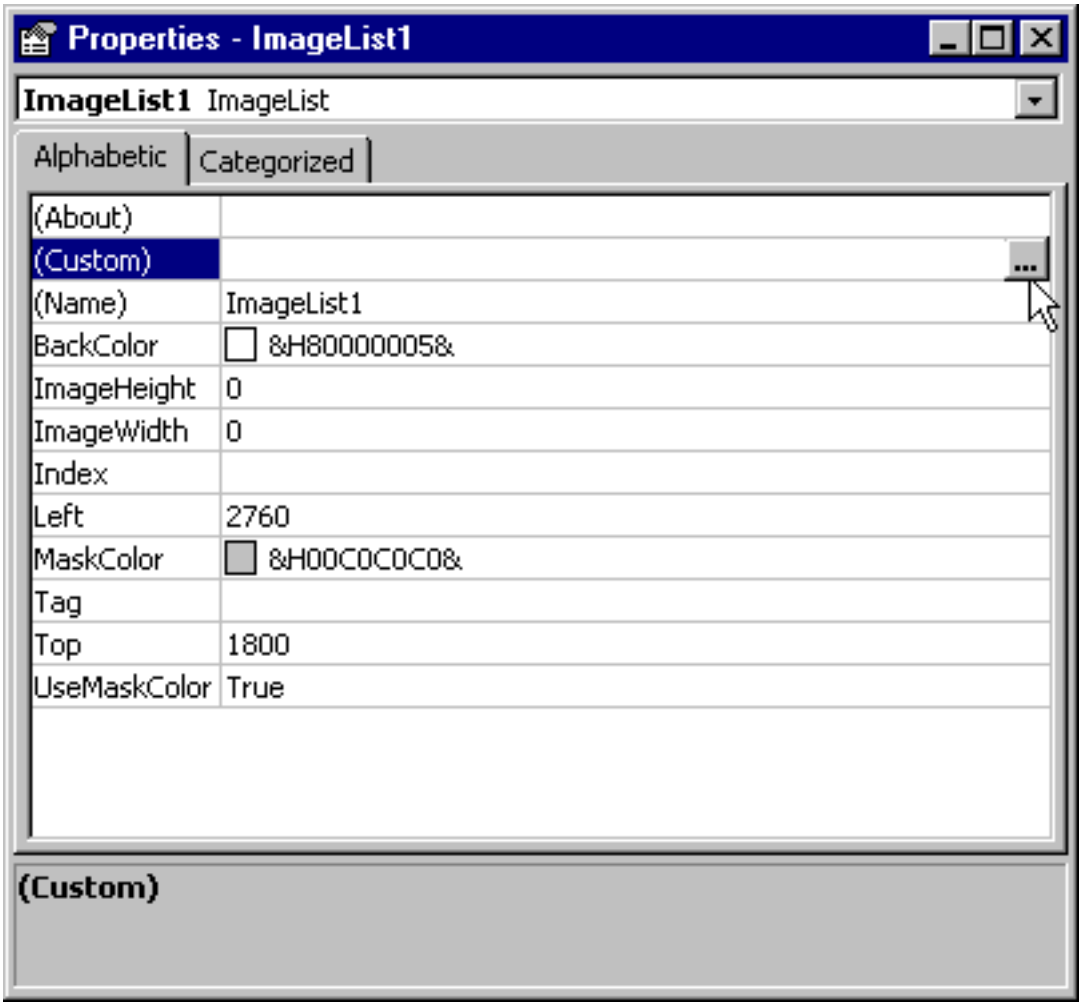
which is also necessary for any form containing a Toolbar. Let's start by placing the Toolbar control on the form. If you just double-click the Toolbar control, Visual Basic will place it at the top of the form for you…



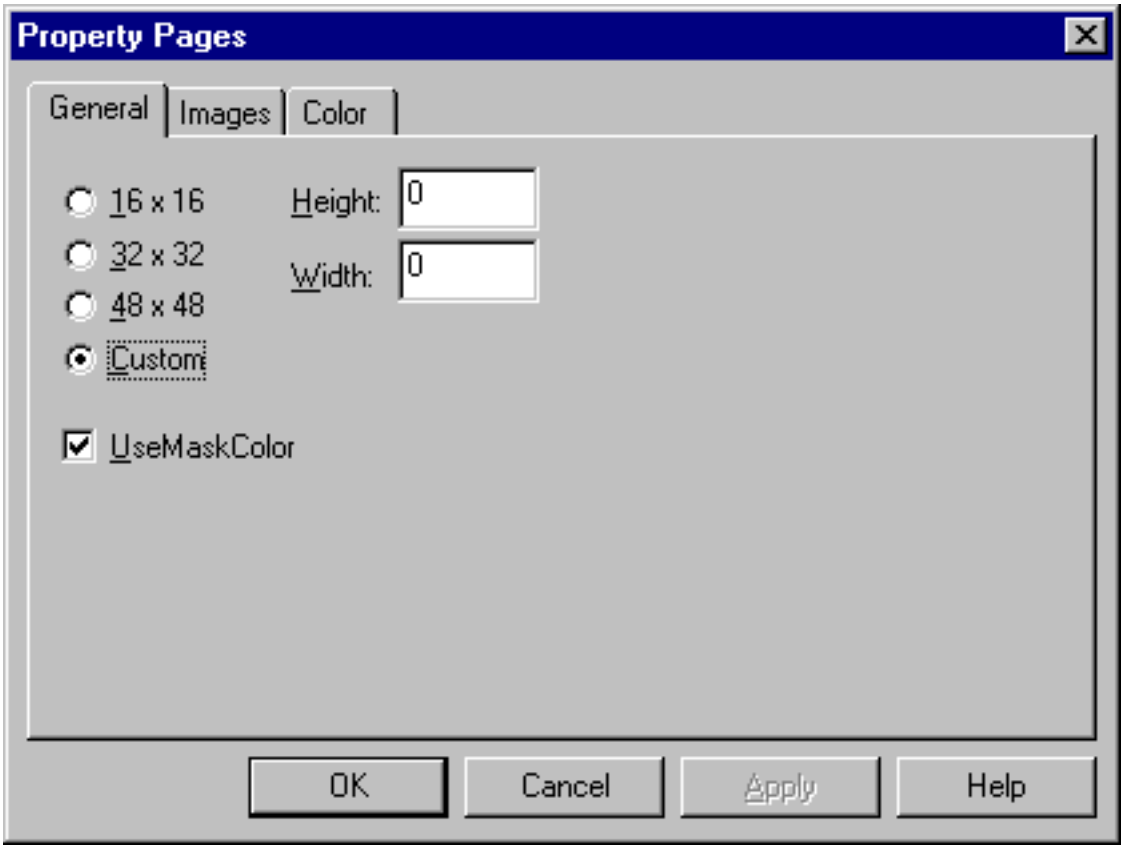
Don't be concerned with the appearance of the toolbar at this point. It's empty---we'll need to add 'buttons' to it as I mentioned earlier, but first, we need to add the ImageList control to the form. Do that by double clicking the ImageList control in the Toolbox. It doesn't matter where it appears on the form---it's a control that is invisible at runtime…



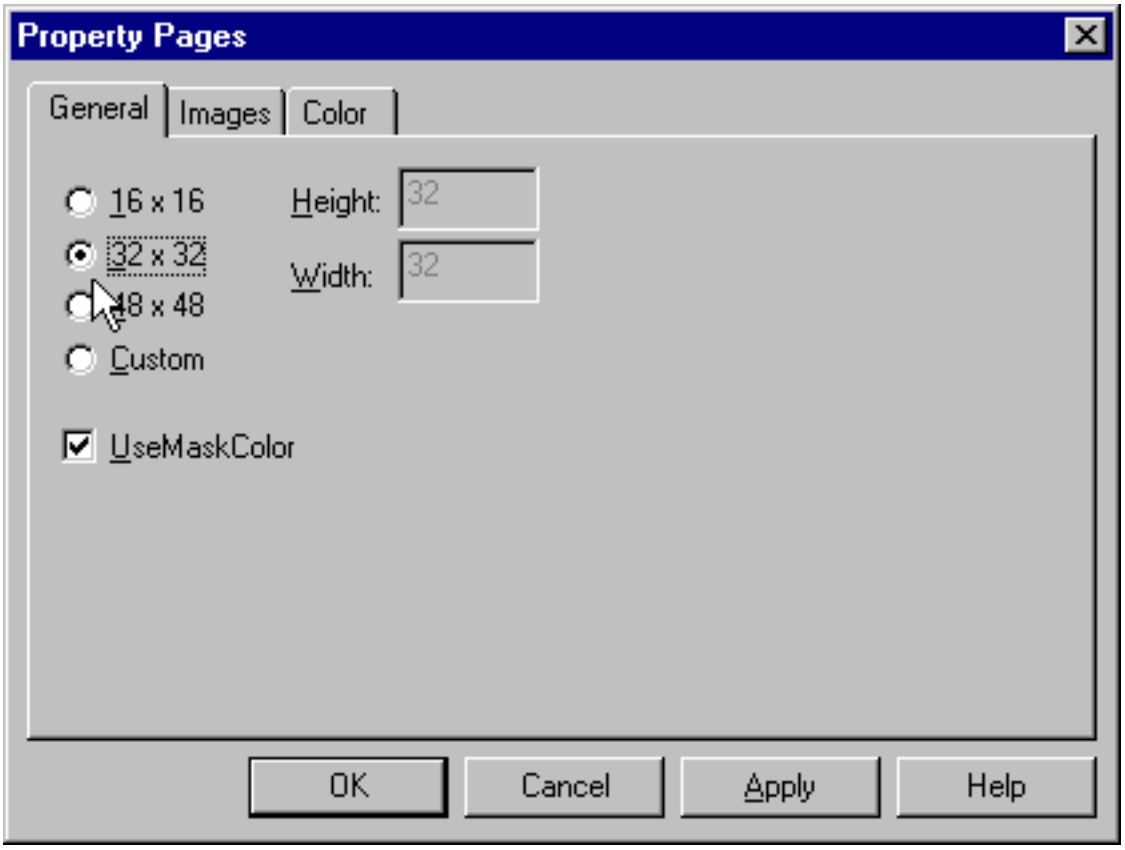
These two controls--the Toolbar and ImageList Controls---are the two controls necessary to implement the visual part of the Visual Basic Toolbar. Now it's time to select the images to appear on the toolbar, and add them to the ImageList Control. To do that, bring up the Properties Window for the ImageList Control and select (Custom)…



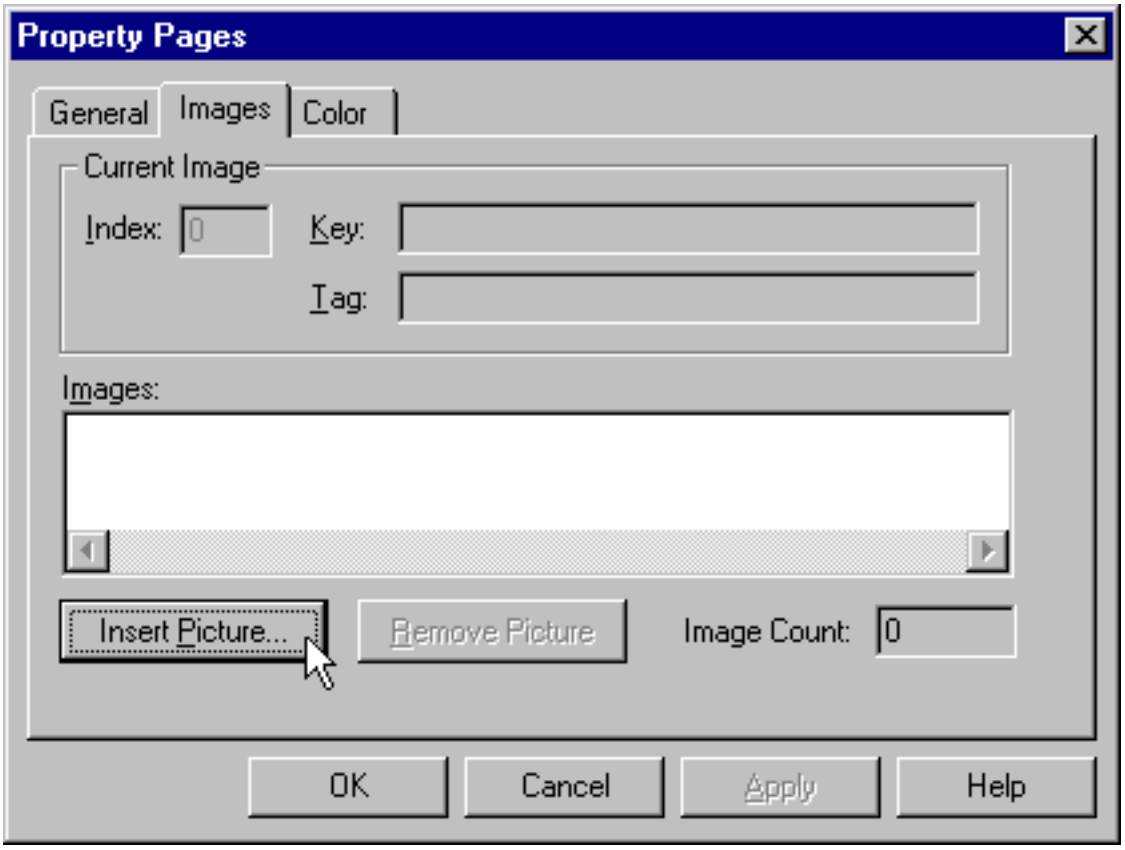
This will bring up the Property Page for the ImageList control…



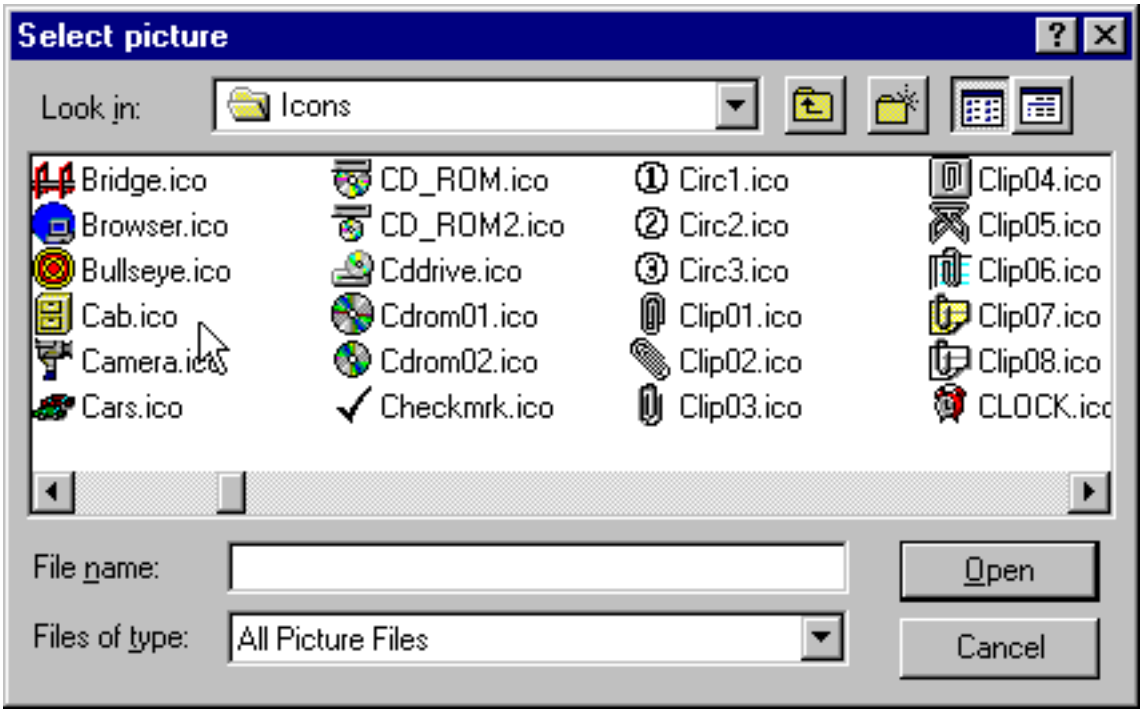
The Information that you see on the General tab here affects the size of the buttons that you will see on the Toolbar. Let's specify 32 x 32



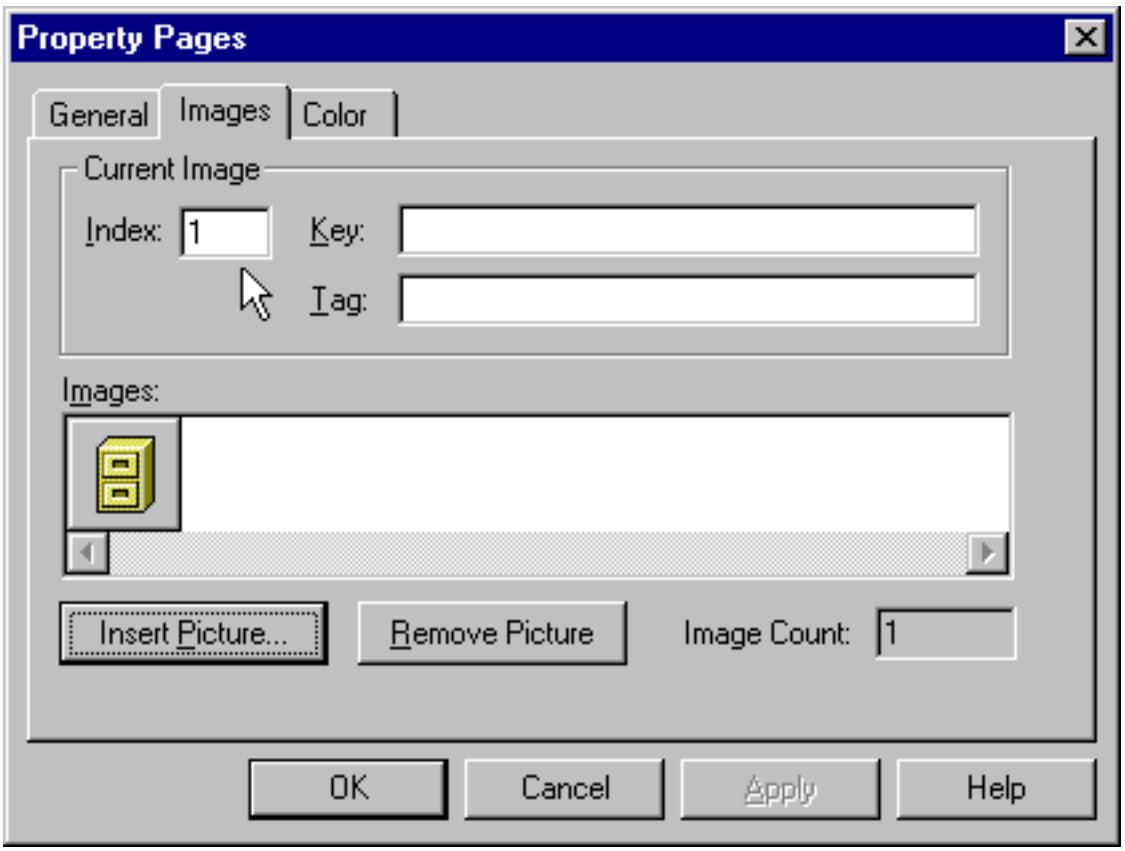
which will be a pretty large toolbar icon--but it will be easier for you to see here in this tutorial. Now we need to start adding images to the ImageList control. We do that by selecting the Images tab…



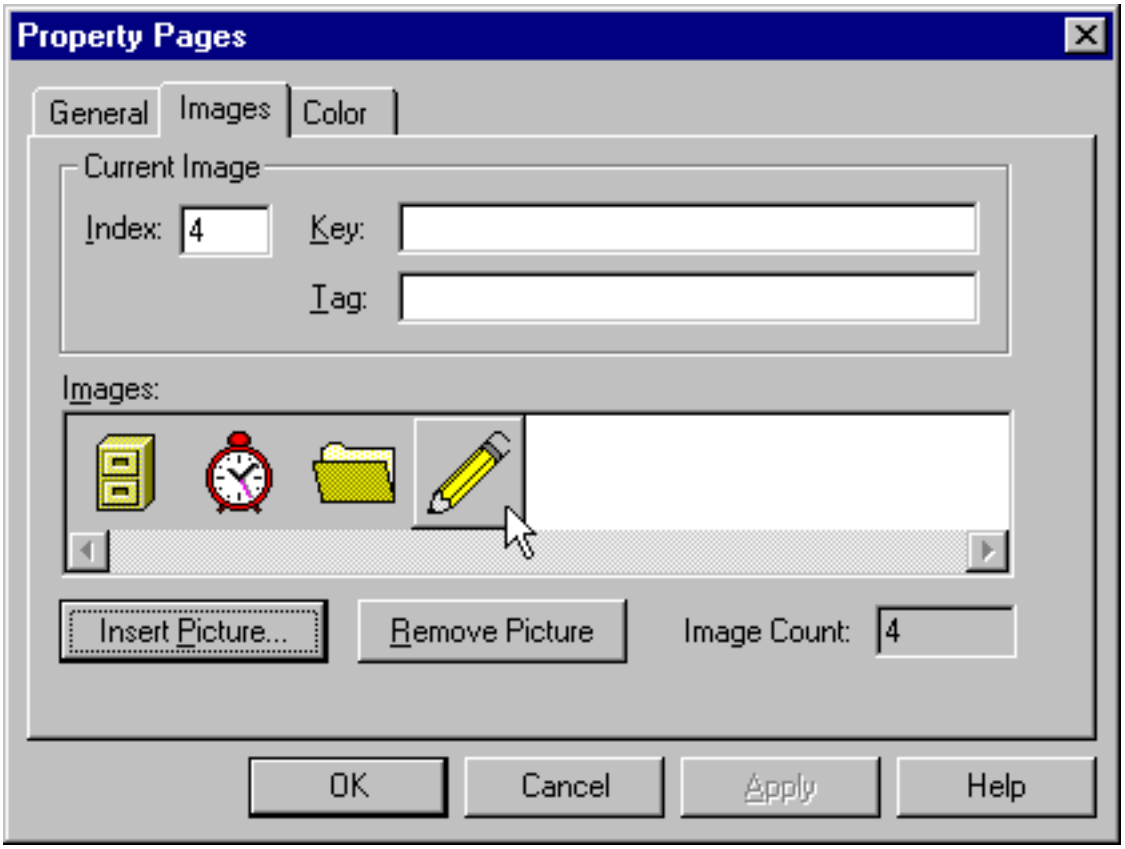
and then selecting the Insert Picture button which will cause Visual Basic to prompt us for an image to add to the ImageList control. I just happen to have a collection of images (icon format) and so, for no particular reason, I'll select the Cabinet icon here…



and as soon as I do, notice the change in the Images tab of the ImageList Control Property Pages…

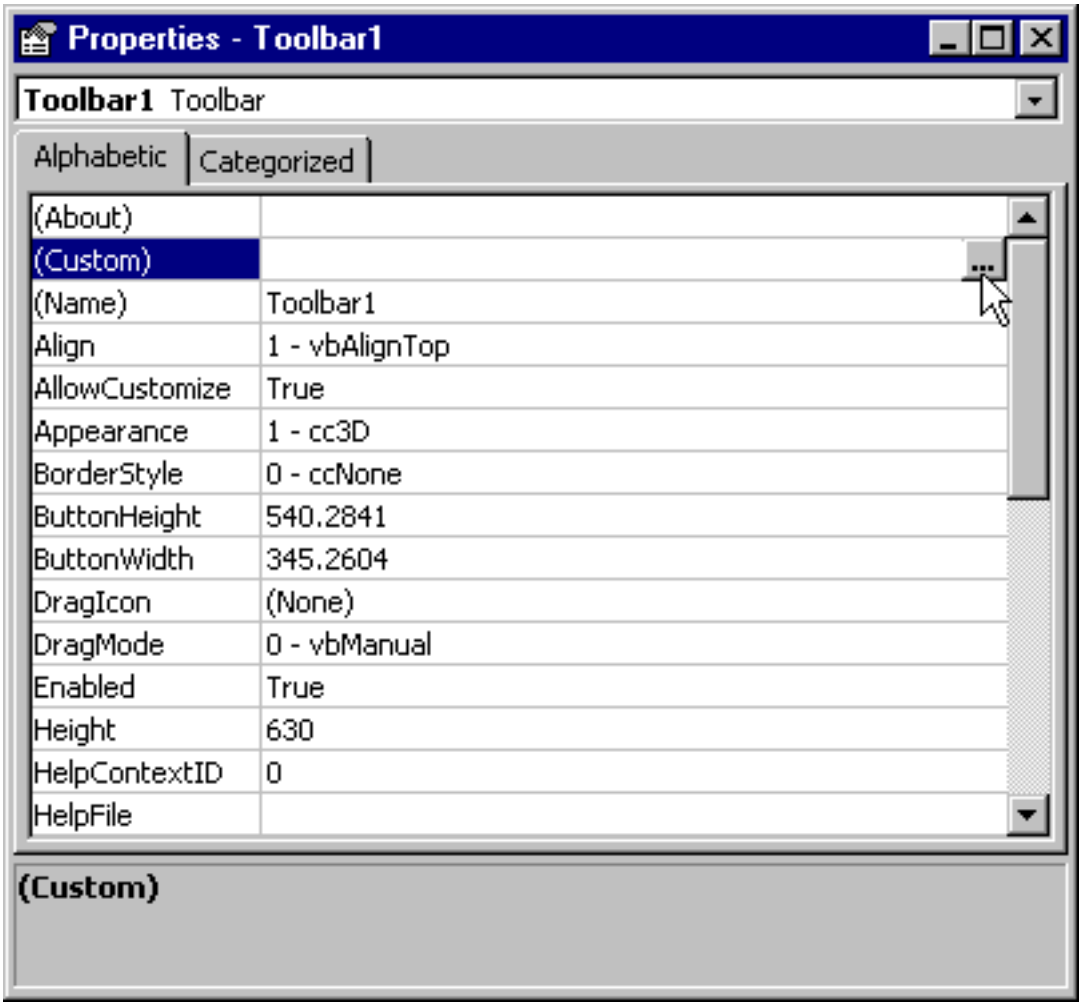


It now shows the Cabinet icon I selected, whose Index value is 1, and indicates an Image Count Property of 1. Each image in the ImageList Control can be referred to by its Index value---and we can also add an optional Key and Tag property value if we wish. The toolbar control will use the Index value to refer to each Image---I'll leave you to investigate the use of the Key and Tag Properties. At this point, we can continue to add images to the ImageList Control. Notice that we can also remove images if we wish by selecting the Remove Picture button. For now, let me add 3 more images to the ImageList control.

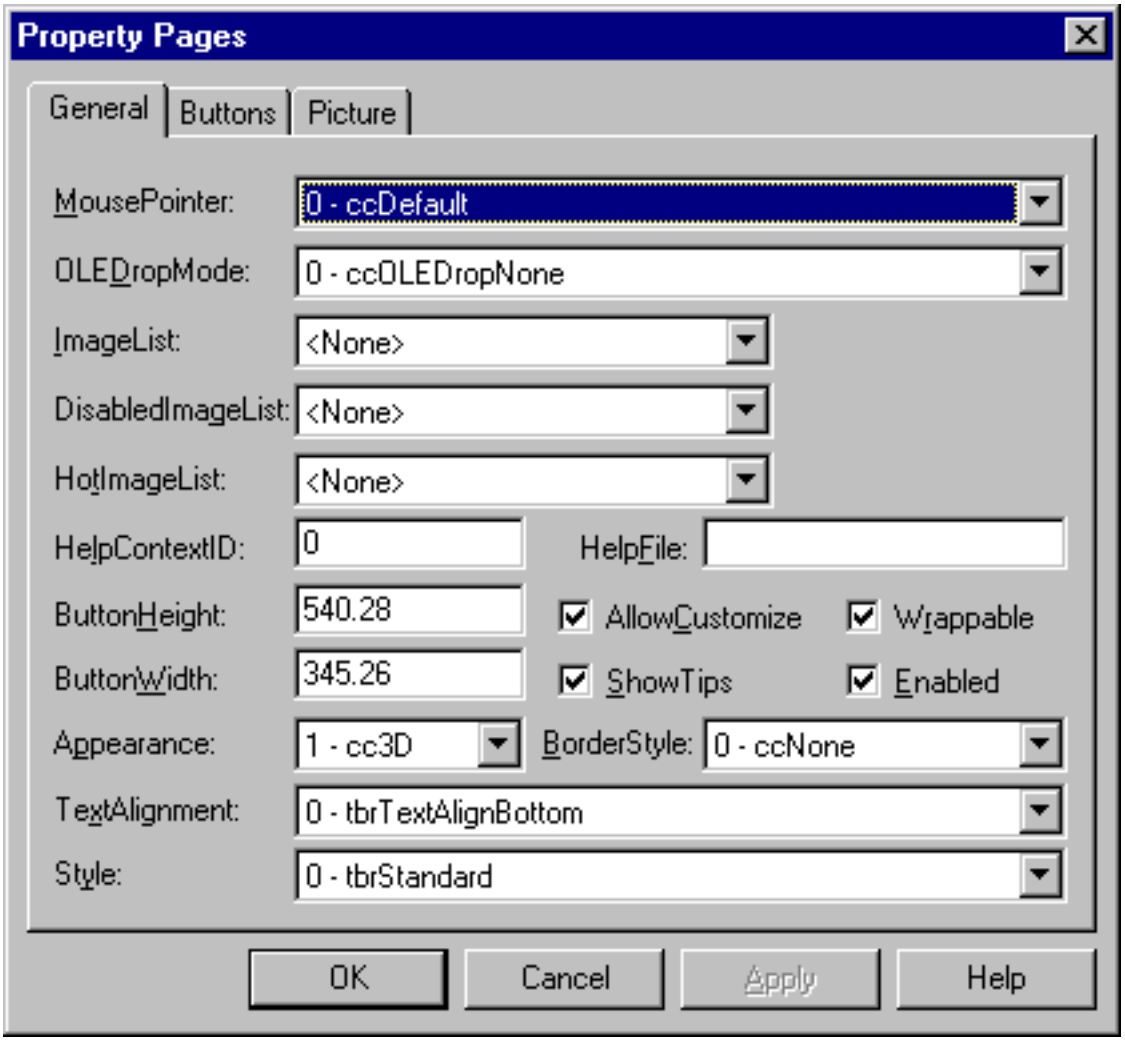


I should mention here that we can add more images to the ImageList control than we immediately foresee using with the Toolbar---with Visual Basic, we can always dynamically change the look and feel of the toolbar in code.

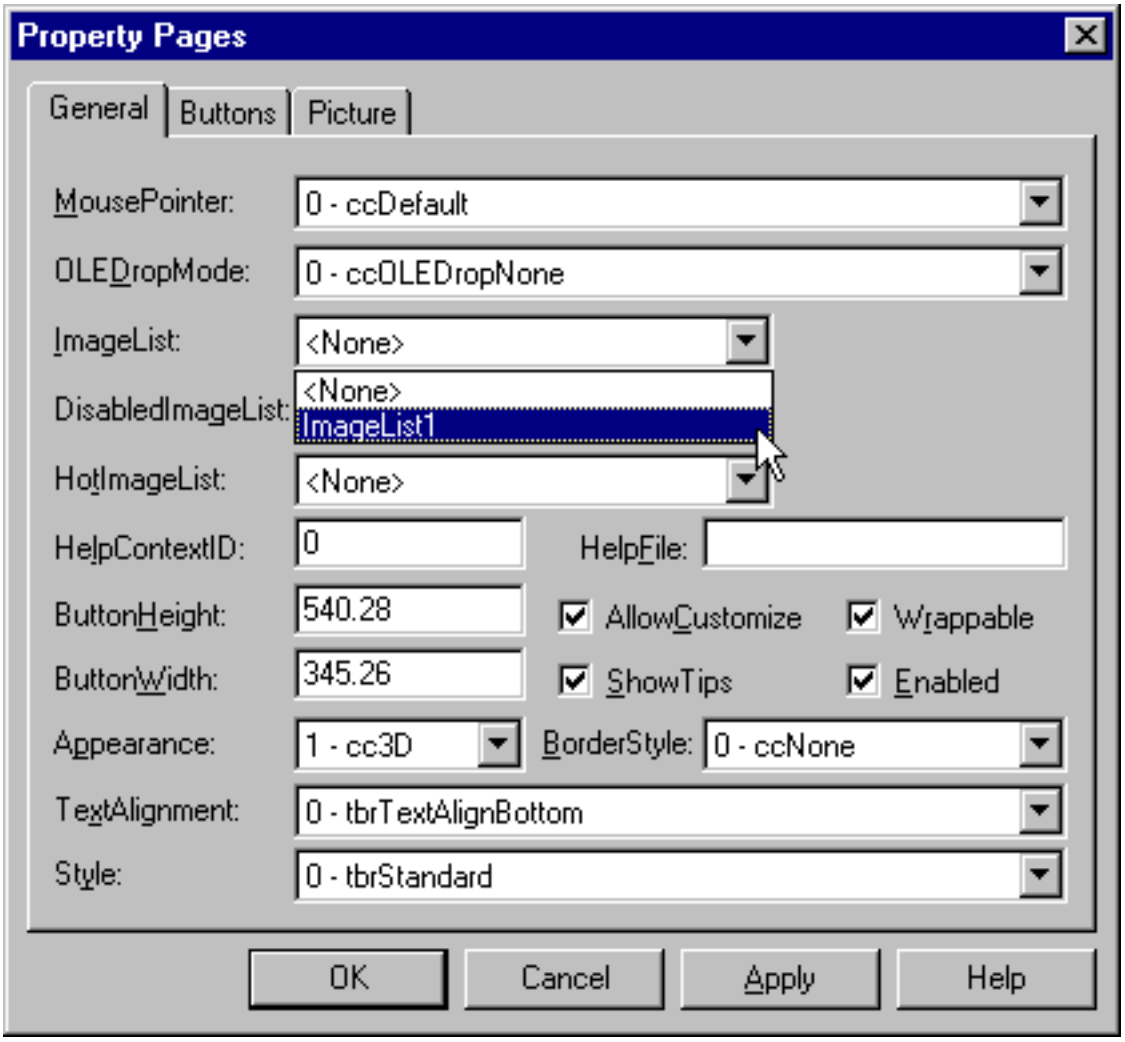
Let's close the Property Pages window now, and begin the process of associating a button on the toolbar with an image in the ImageList control. To do that, not surprisingly, we open up the Properties Window for the toolbar…



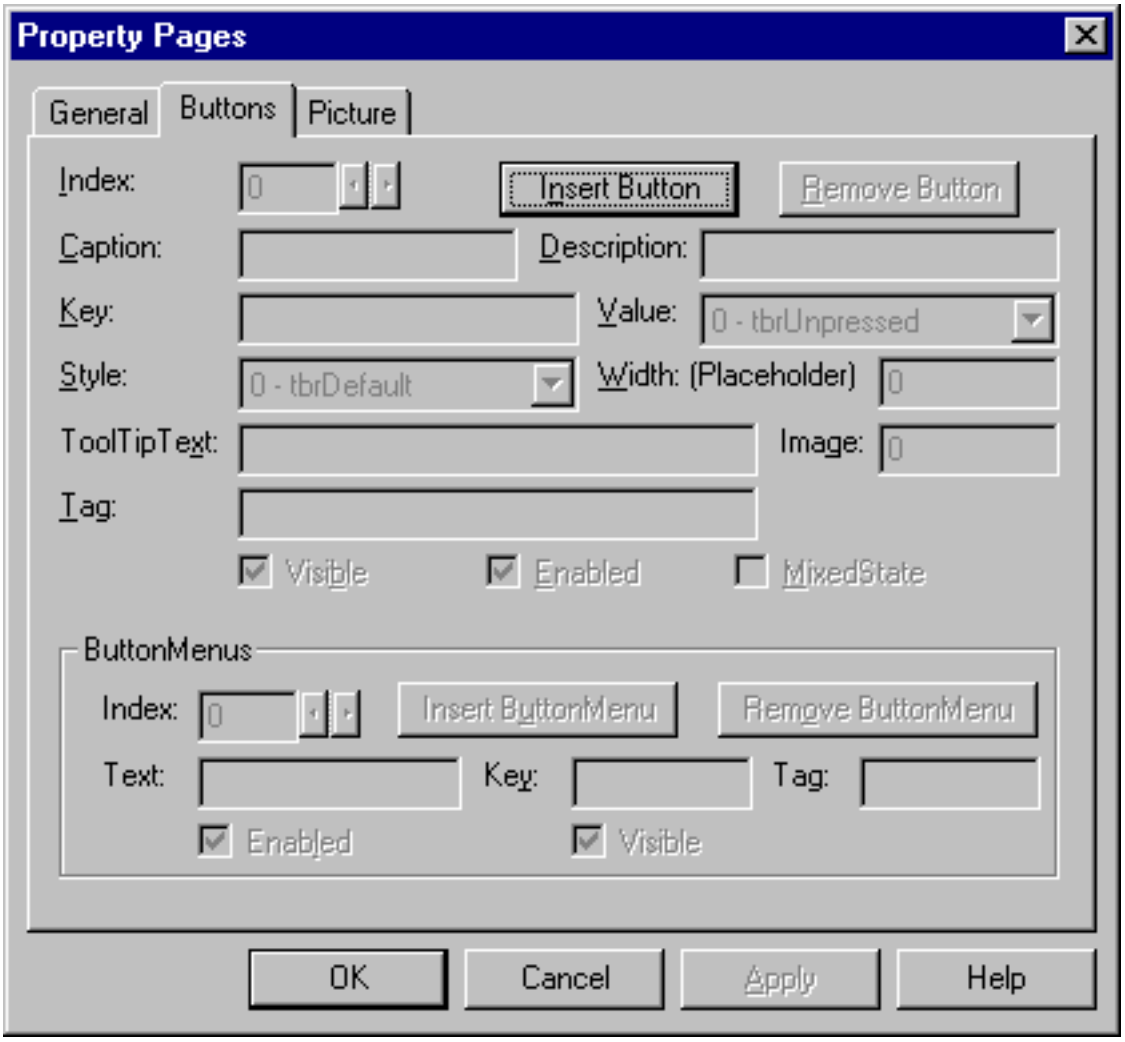
…then click on (Custom) to open up the Property Pages (you can get there quickly if you right click the toolbar on the form and select Properties)



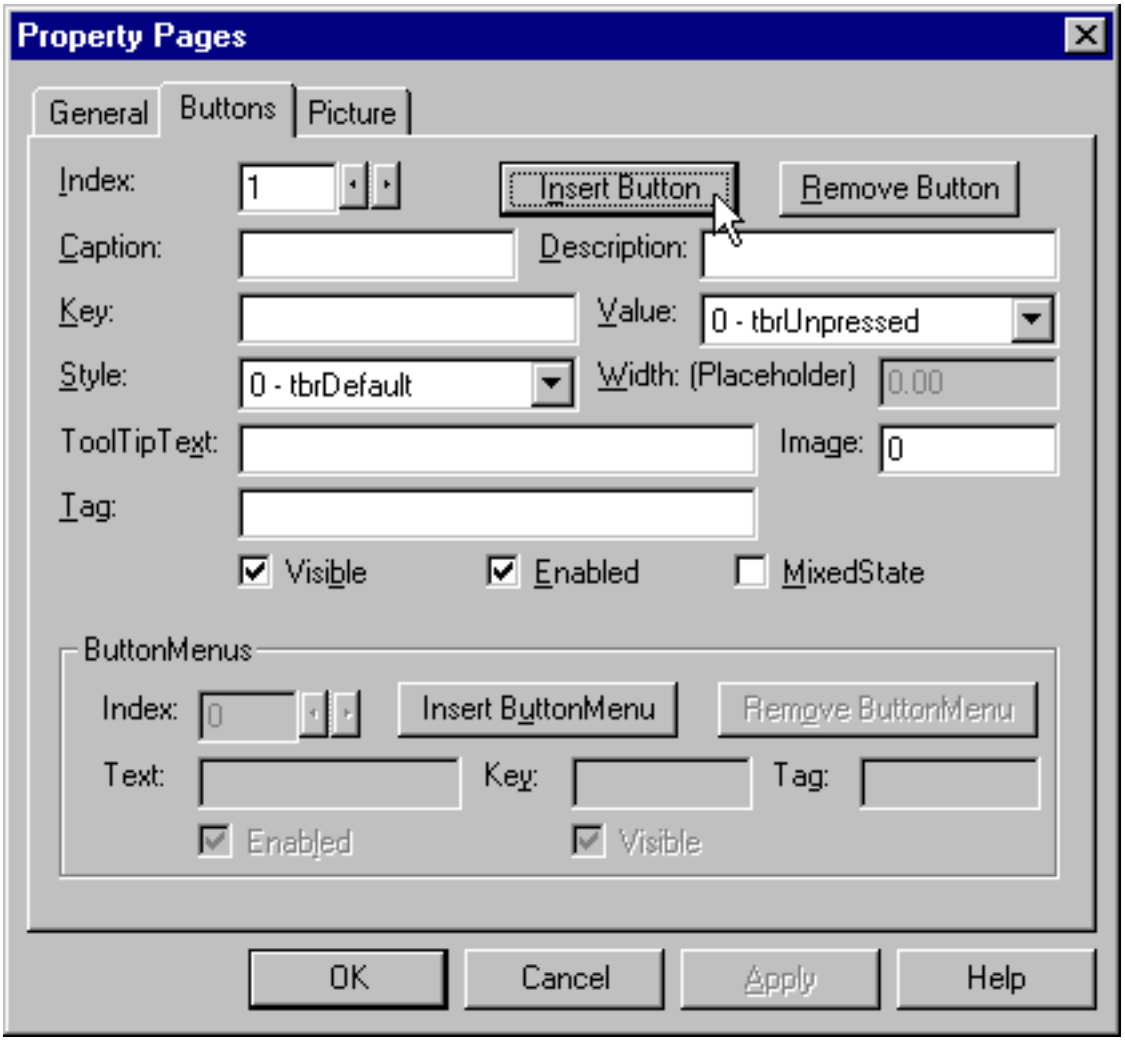
The Property Pages for the toolbar are a great deal more complicated than the ImageList control, and I won't be getting into a lot of detail on them. I'll show you how to add buttons to the toolbar, associate them with an image in the ImageList control, and leave you to experiment on your own. The first thing we need to do is tell the toolbar control where to find the images for the buttons that we're about to add. To do that, we need to click on the dropdown ListBox for the ImageList property and select the ImageList Control that contains our images…



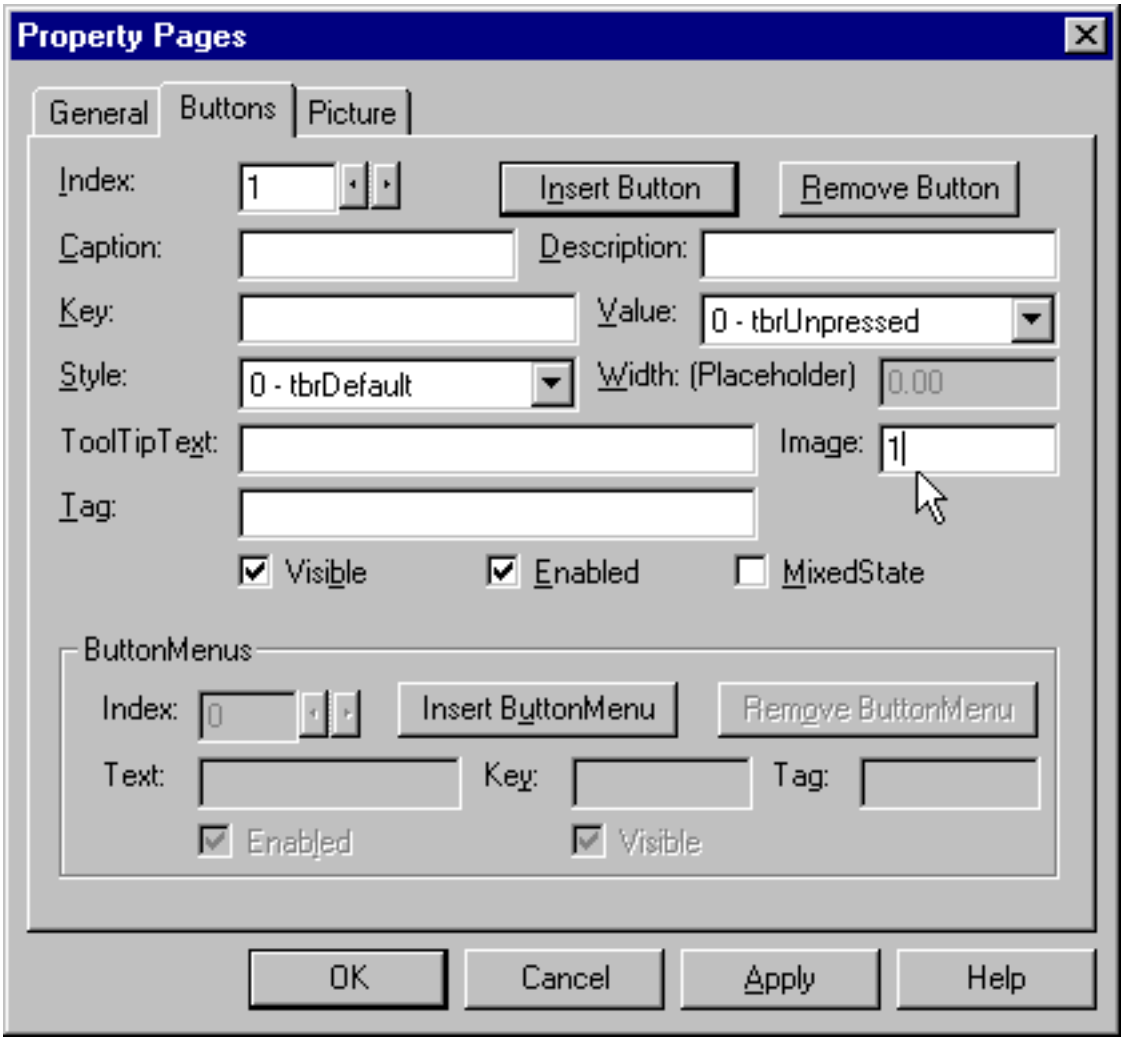
Now that we've associated our Toolbar with an ImageList control, the next step is to start adding buttons to the toolbar. We do that by selecting the Buttons tab…



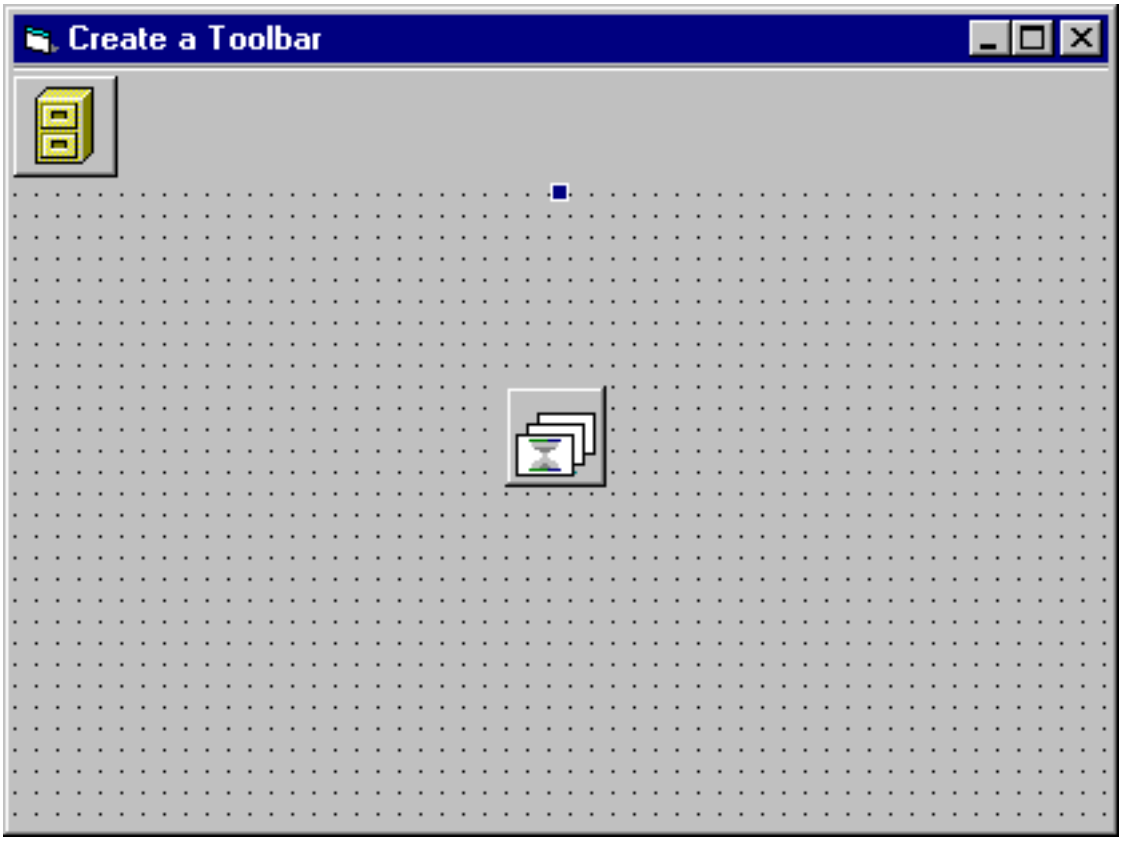
Don't worry, this looks a lot more complicated than it is. This is the Tab that allows us to specify the buttons to appear on our toolbar. Notice at this point that the Index property for the Buttons tab is dimmed---that indicates that our toolbar has no buttons. Let's click on 'Insert Button' to add our first button to the toolbar.



Once we click on 'Insert Button', the Buttons Property Page becomes undimmed---notice that the Index value now reads 1, indicating we are editing the Property values for the first button on the toolbar. We can now associate an image in the ImageList control with this button. To do that, all we need to do is specify the Index value of the appropriate image in the ImageList in the Image Property of the Buttons tab---like this…

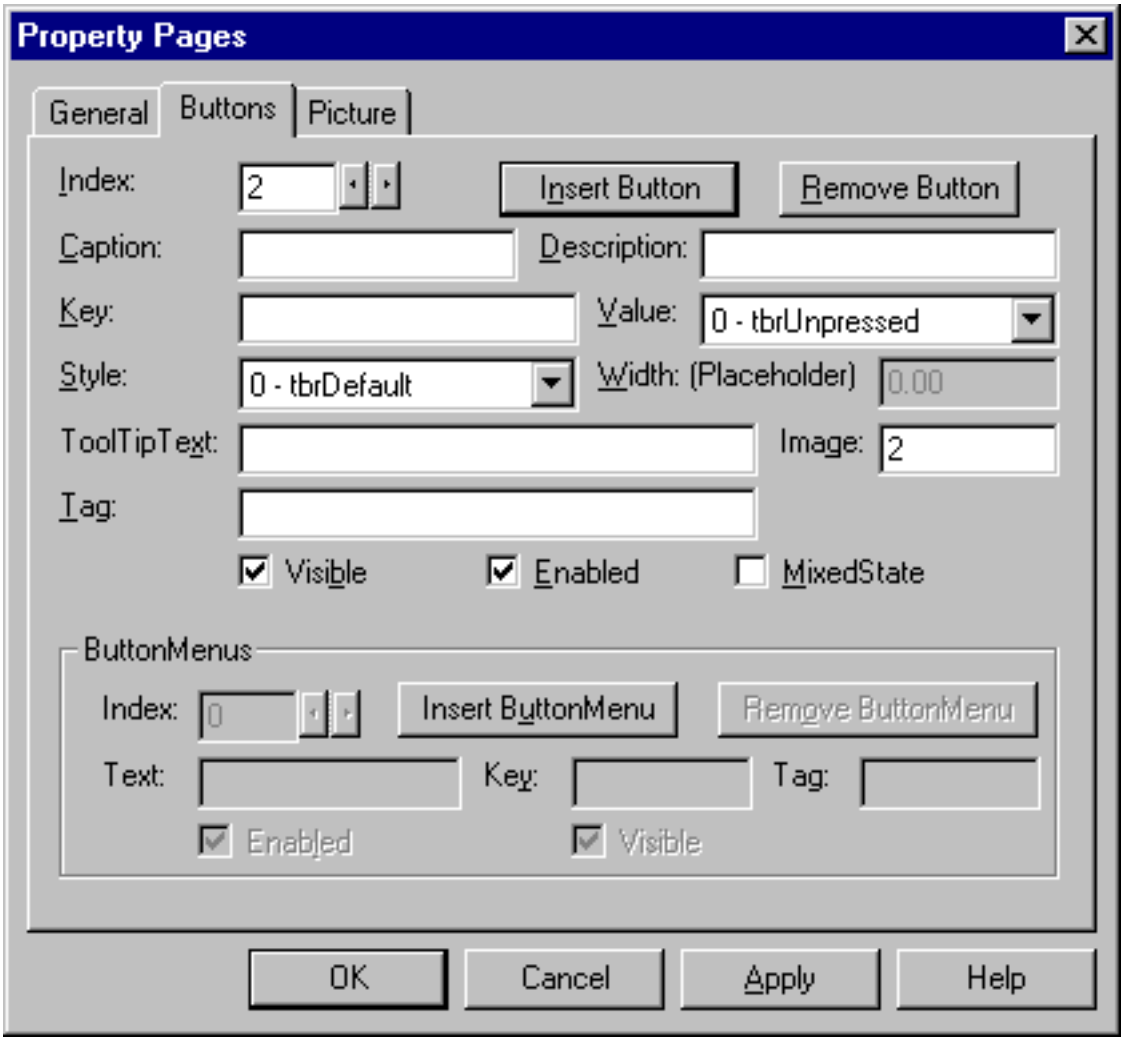


By specifying a value of 1 for the Image Property of the first button, we are telling Visual Basic to display the Cabinet icon as the image for the first button. If we now click on the Apply button, we'll see that our toolbar now looks like this…

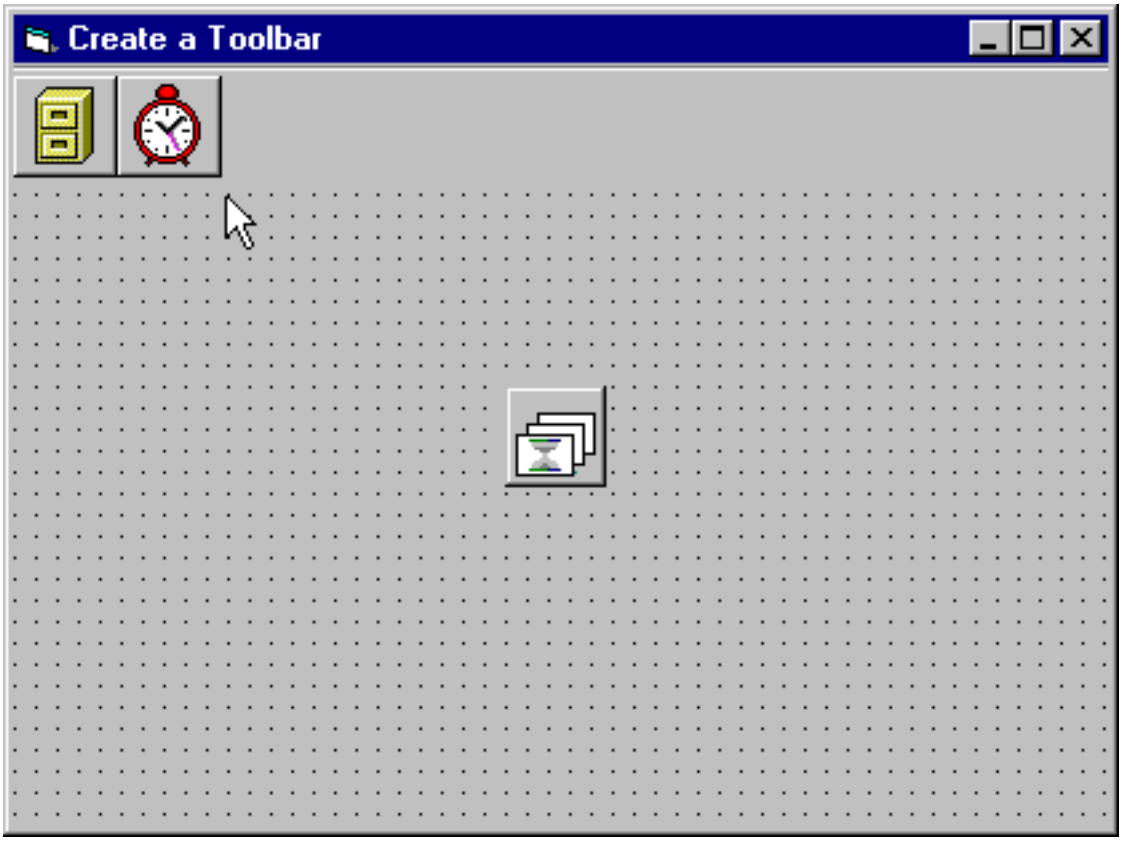


I should point out, before turning our attention to writing code for the Toolbar, that if we so desired, we could also specify ToolTipText for the buttons we add to the Toolbar by placing a value in the ToolTipText Property of the Buttons tab.

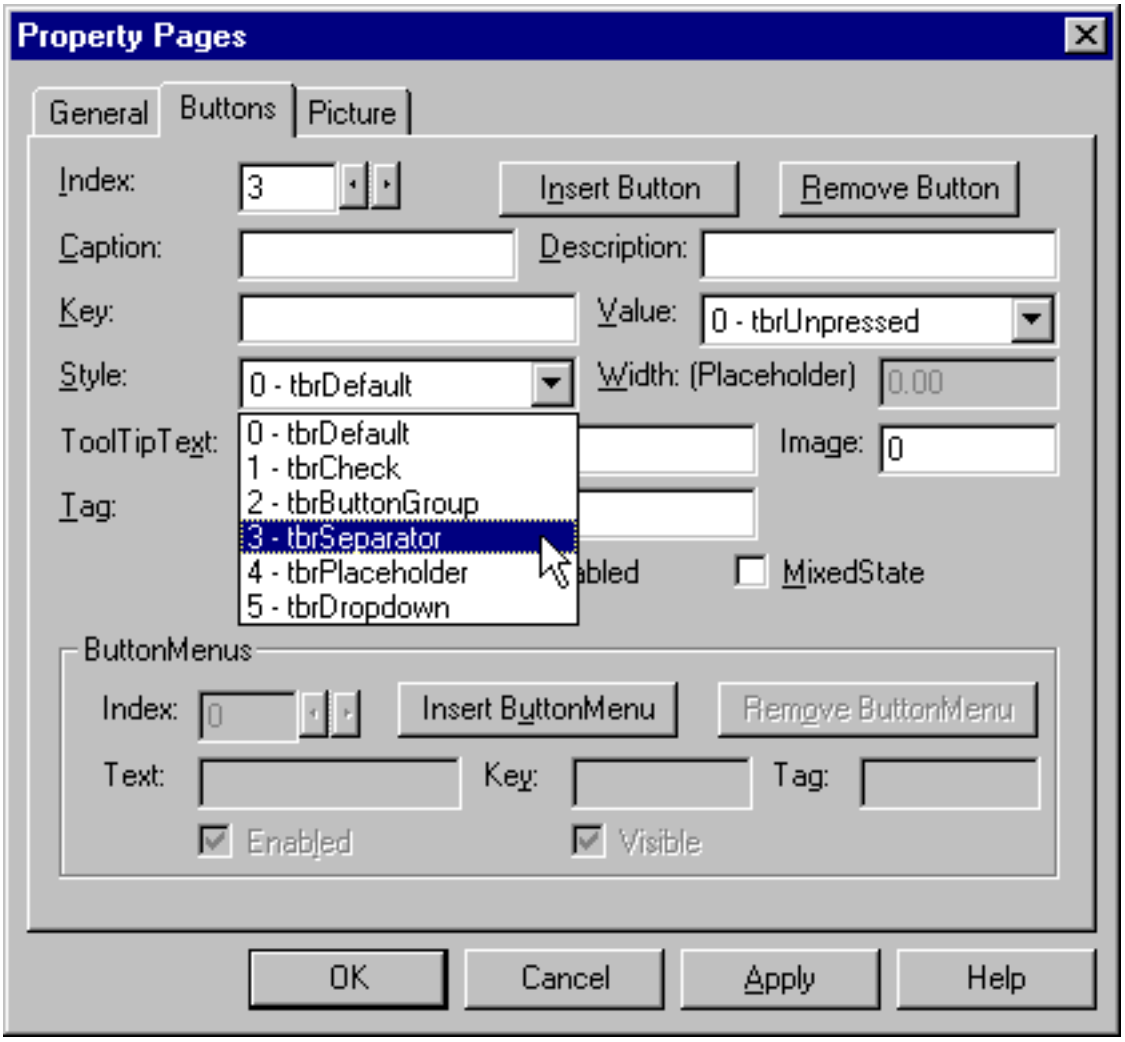
Let's add another button to the toolbar by clicking on 'Insert Button' again, and then specifying a value of 3 for the Image Property of the button (remember, we don't need to follow an order in assigning Image Properties, neither do we need to use every image contained in the ImageList Control)…



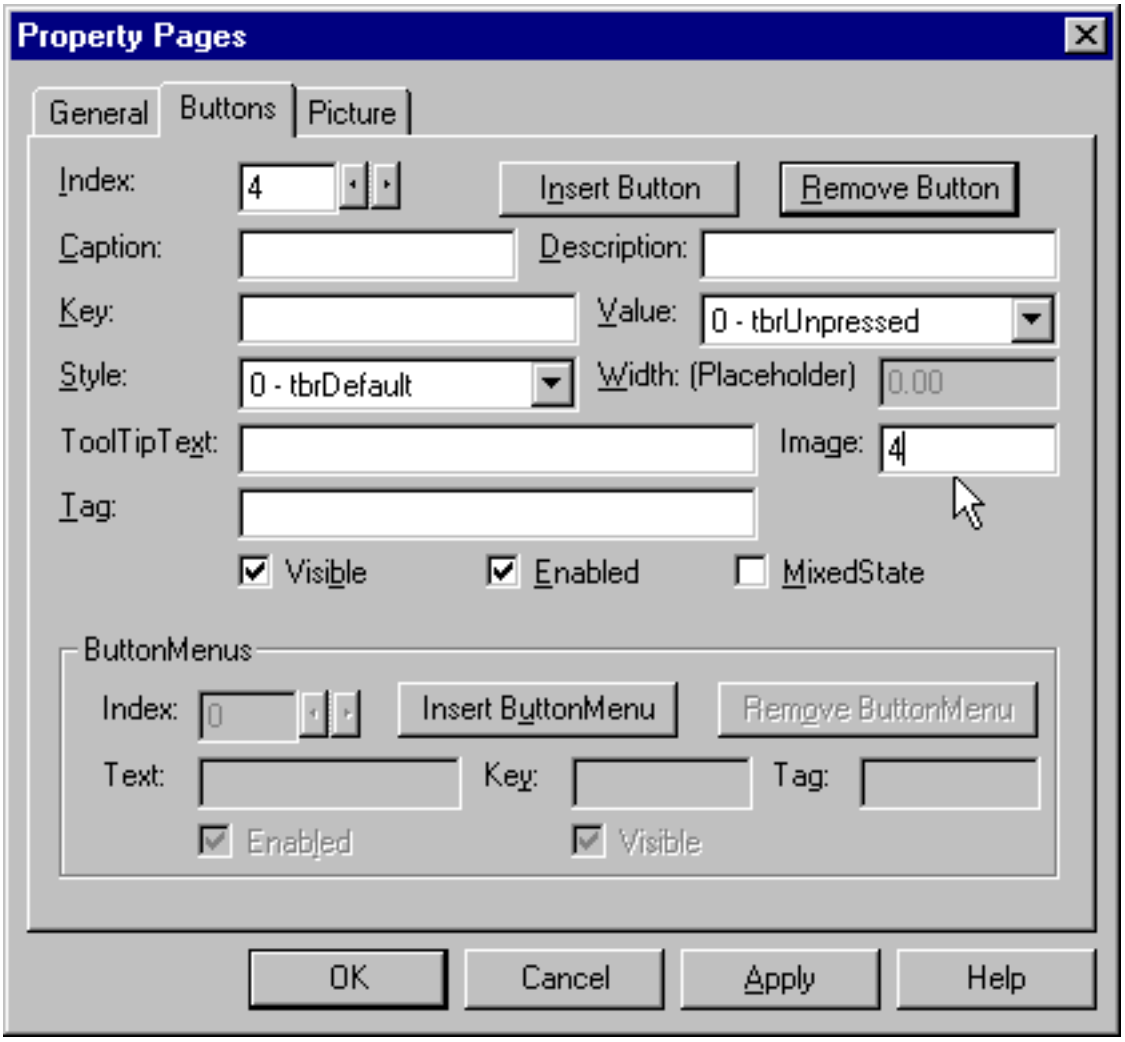
Once again, if we click on the Apply button, we'll see that the toolbar has changed once again…



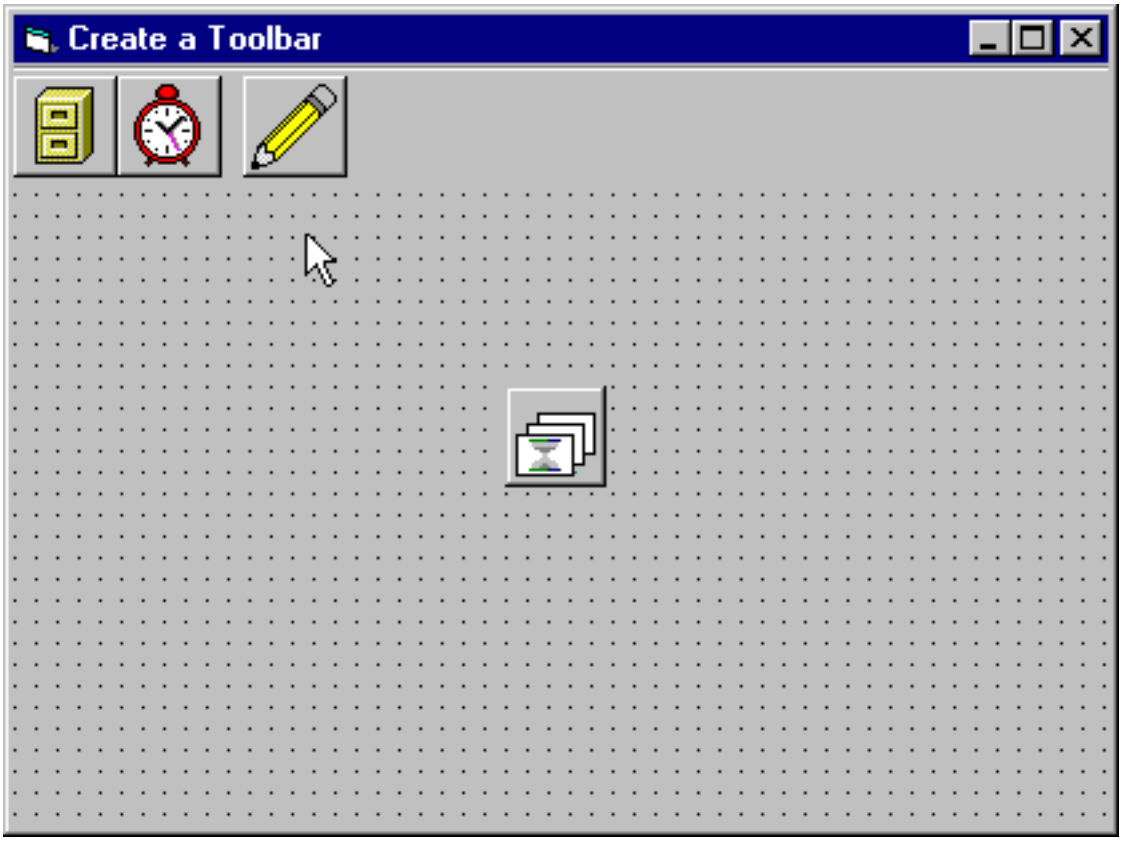
Many toolbars have a 'gap' or separator between groups of buttons. To do that, we click on 'Insert Button' once more, but this time instead of specifying a value for the Image Property, we leave it at 0, and specify a value of 3-tbrSeparator for the Style Property…



If we click on the Apply button, there won't be any obvious changes to the look of the Toolbar at this point. It won't be until we add the next button, which we'll do by once again clicking 'Insert Button' and specifying an image in the ImageList control…

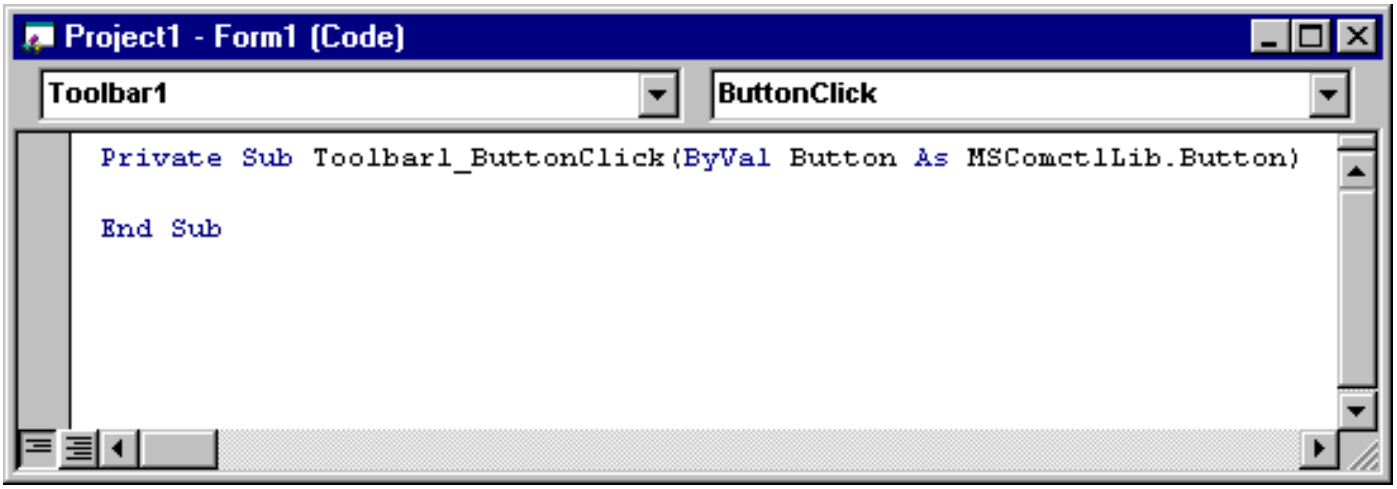


and then click on the Apply button that the effects of the separator button are realized…



At this point, we have a Toolbar that looks pretty good, but doesn't do anything. If we run this program now, the Toolbar appears, and if we click on the individual buttons on the Toolbar, they appear depressed, but that's all.

Not unlike a command button, if we want a button on the Toolbar to trigger some kind of action, we need to write code to do it. What complicates this process a bit is the fact that the individual buttons on the Toolbar do not react to their own events---there are only events associated with the Toolbar control itself. So, if we want to write code to trigger some action when a particular button on the toolbar is clicked, we first need to discern which button has been clicked. Fortunately, Visual Basic will tell us which button on the Toolbar has been clicked when the ButtonClick event procedure (NOT THE CLICK EVENT) of the Toolbar is triggered. Let's look at the ButtonClick Event Procedure of the Toolbar now…



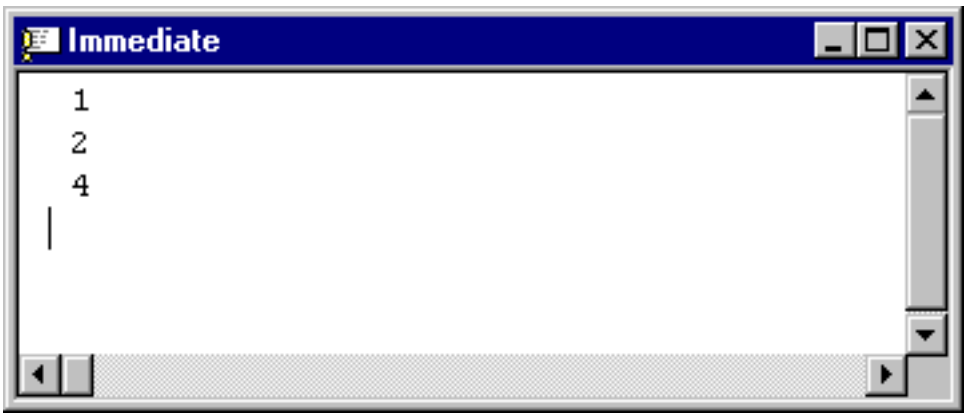
The ButtonClick Event Procedure of the toolbar is passed an Object Variable representing the button on the Toolbar that has been clicked. We can determine the Index value of that button to determine which button has been clicked. Knowing that, we can write code that is triggered when a particular button is clicked, like this code, which will display on the form the button that the user has clicked…

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)

Debug.Print Button.Index

End Sub

If we run this program, and click on the buttons from left to right, we'll see this displayed in the Immediate Window…



What about button number 3? That's the separator bar---it's a button, but it's not obviously present on the Toolbar. The Index values for the actual buttons on the Toolbar are 1, 2 and 4.

It doesn't take too much imagination to write some practical code for the Toolbar. For instance, if the first button on the Toolbar is one to end the program, then this code will do the trick…

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)

If Button.Index = 1 Then
  MsgBox "Thanks for using my program"
  Unload Me
  End
End If

End Sub
```

Summary

I hope I've demystified the concept of a Visual Basic toolbar for you. Feel free to experiment with some of the other settings of the Toolbar which I didn't discuss---I look forward to hearing from you.