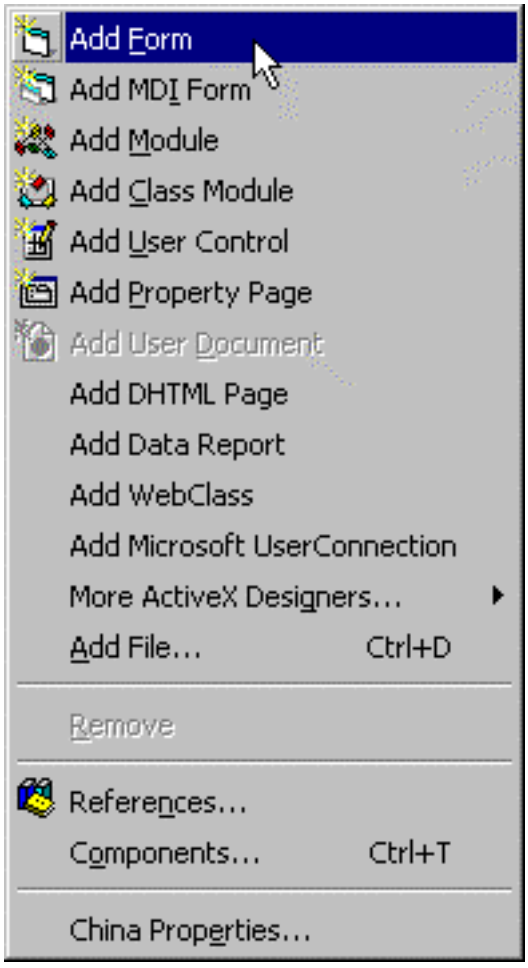
Multiple Form Projects in Visual Basic 6

Those of you who have read my first book, Learn to Program with Visual Basic, know that the China Shop Case Study consisted of a Visual Basic Project with a single form.

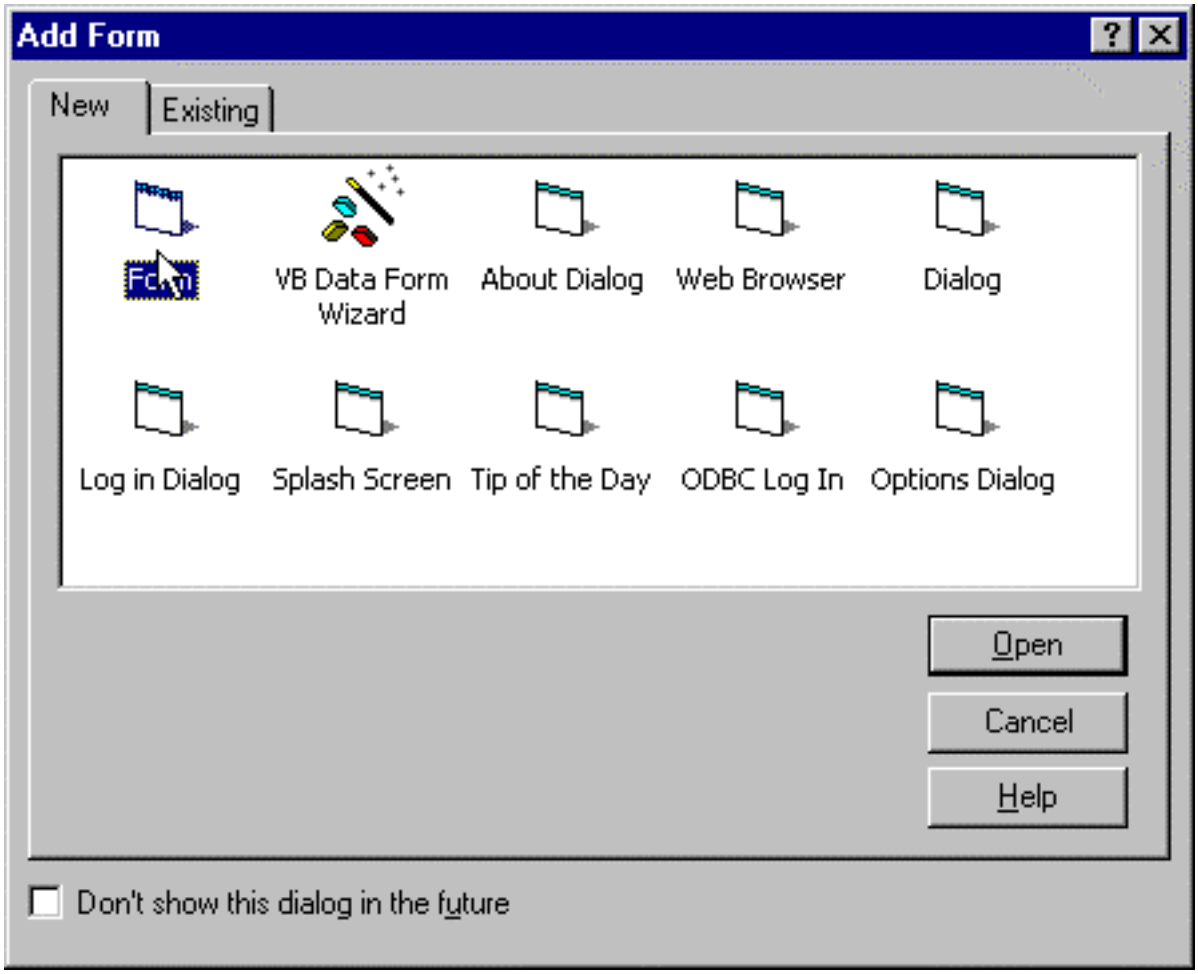
Most real world projects will have more than one form, however, and how to create a project with more than one form is the topic of this month's Web Article. Since most of you are familiar with the China Shop, I thought that it would be a good idea to add a second form to the China Shop Project which we will use to display what I call a 'Help About' form---a good opportunity for a programmer to pat themselves on the back after a great job!

Step 1: Add a New Form to the Project

By default, Visual Basic creates a Startup form for us when we create a new Project. To add a second (or additional) form to a project, all we need to do is select Project-Add Form from the Visual Basic Menu Bar.

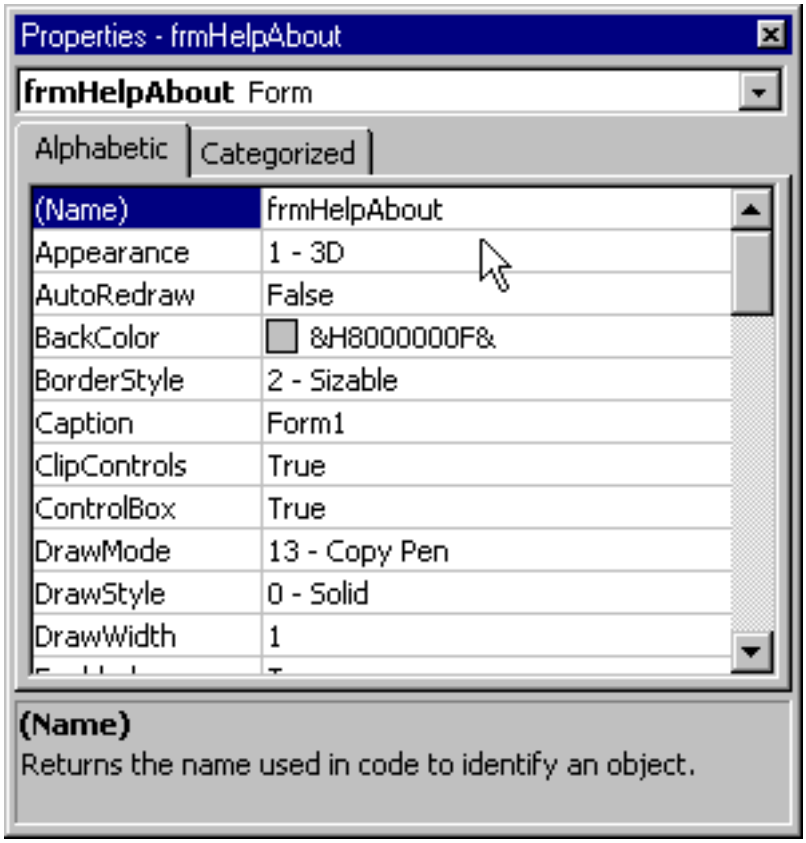


The following screen shot will appear. ..



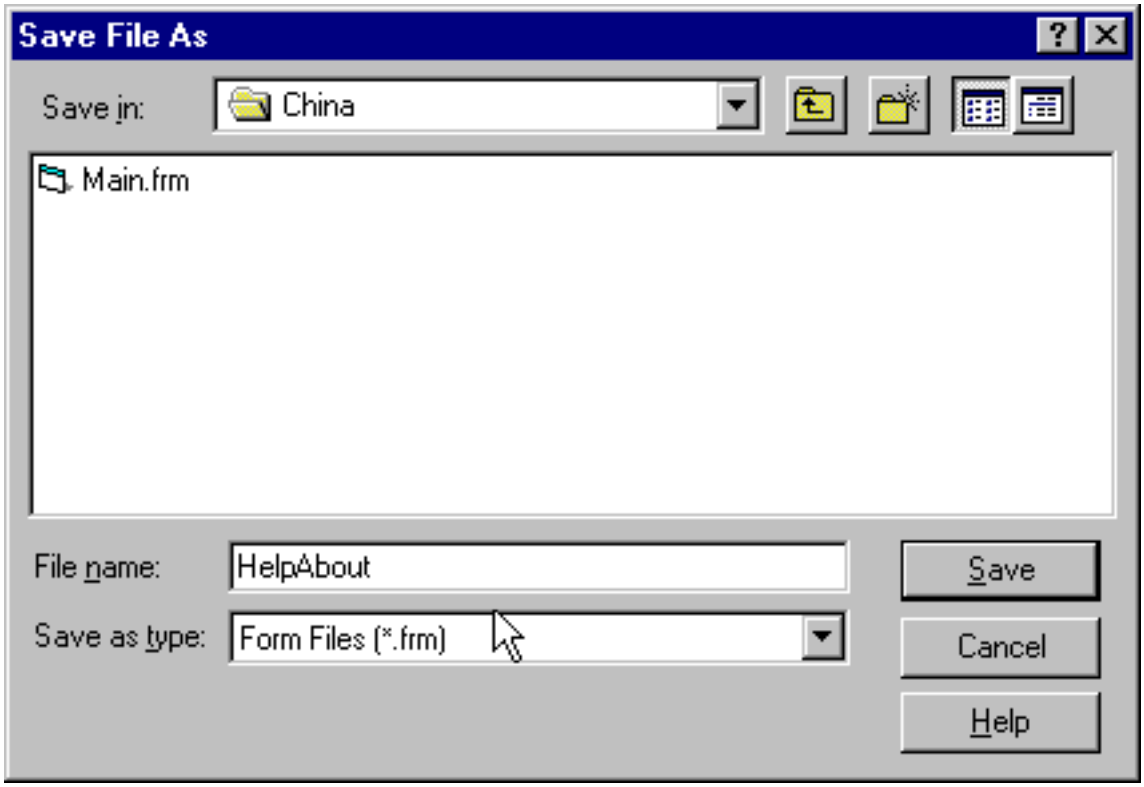
At this point, we just need to make sure that the 'New' tab is selected, and then either double-click on 'Form' or select 'Form' and click on the 'Open' Button. Visual Basic will then display a new form for us in the IDE, with a Caption reading 'Form1'.

We now have two forms loaded into the IDE. The smart thing to do now is to save the form, but before we do that, we should give it a meaningful name. To do that, select the form in the IDE by clicking on it with the mouse, and then bring up its Properties Window. Find the Name Property, and change it from Form1 to **frmHelpAbout**.

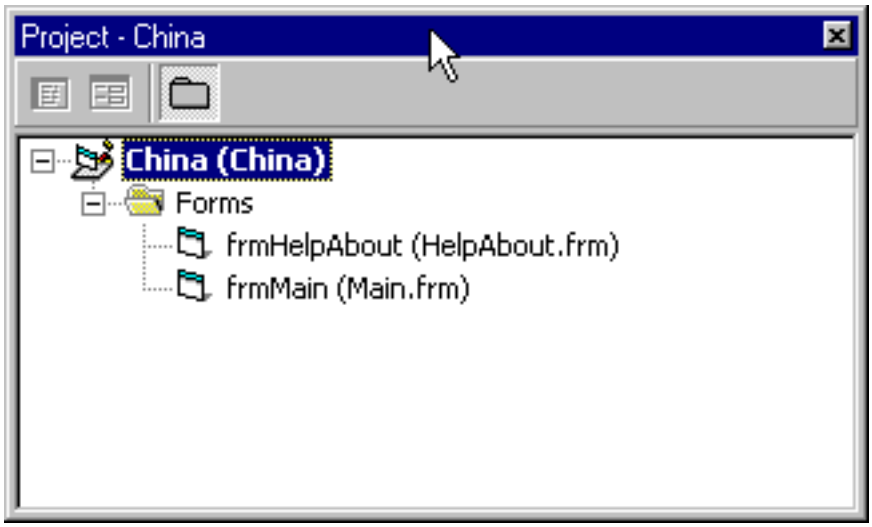


Now click on the Save button on the Toolbar. Visual Basic will immediately prompt you to save the new form with a disk file name of 'frmHelpAbout.'

We can save the form with that name, or change it if we wish. My personal preference is to drop the 'frm' from the disk file name, and save it as HelpAbout. Let's do that by clickiing on the Save button, also ensuring that the form will be saved in the correct folder or directory (\VBFILES\CHINA).



Once the form has been saved, if we bring up the Visual Basic Project Explorer Window, we should now see two forms---Main (with a Name Property of frmMain) and HelpAbout (with a Name Property of frmHelpAbout).

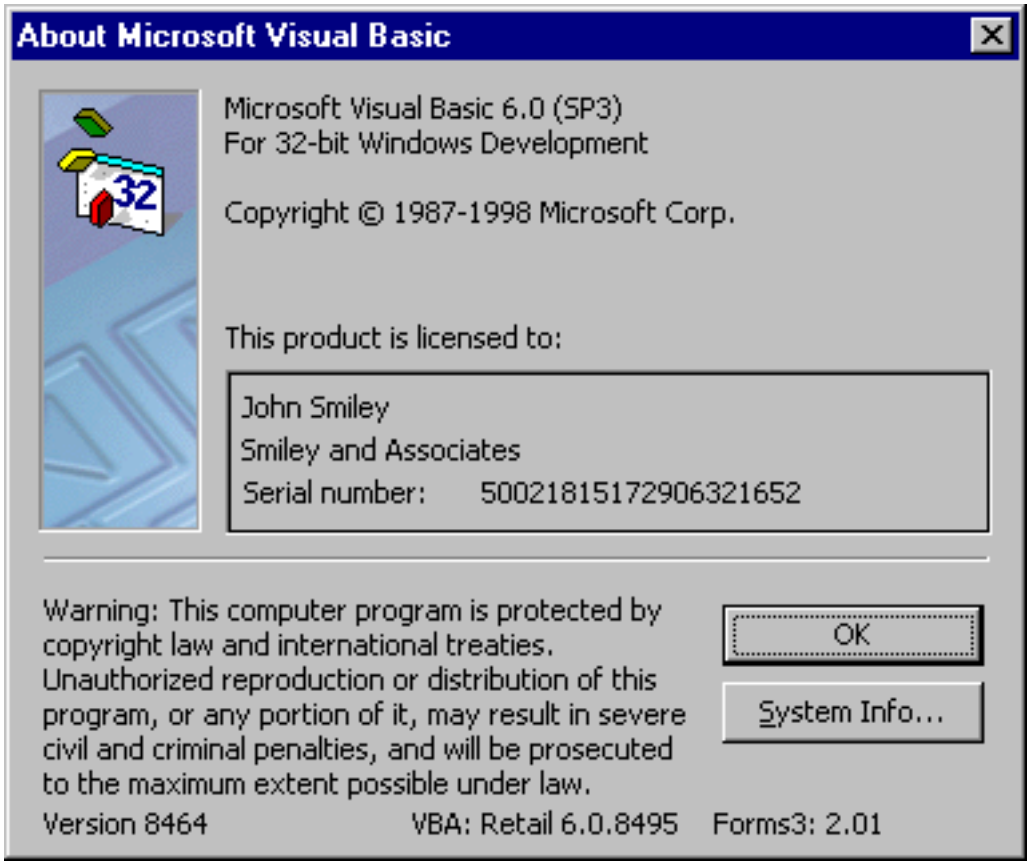


Step 2: Design the New Form

Our project now has two forms--at this point, we need to design the second form, and then we'll add code to display it from the Main form of the China Shop Project.

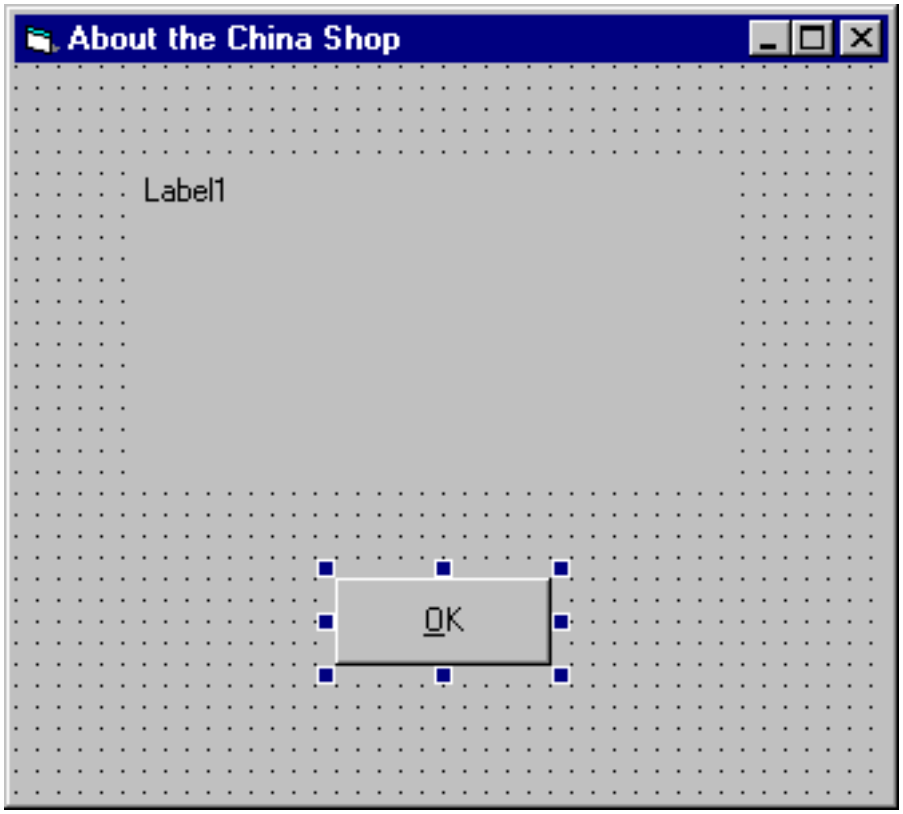
A Help-About form needn't be too fancy---basically, it should say something about the

person who wrote it, and if you're smart, you'll include some contact information in the event that the person viewing it wants to hire you for some work! You can always use the Microsoft Help-About forms as a sample For instance, here's the Help-About form for my copy of Visual Basic 6.



As you can see, the Visual Basic Help-About Form contains information about the program, and also two buttons---one is a System Information button (more on that in a future Web article) and one is an OK button. We need to include an OK button as we need a way to close the form when the user is done viewing it!

Let's resize the frmHelpAbout form now, and add a single Label control and a Command Button. We'll change the name of the Command Button to cmdOK, and change its caption Property to OK. Finally, change the name of the Label to lblTootYourHorn. Finally, change the Caption Property of the form to About the China Shop. If you are following along with me, your form should look similar to this.



You may be wondering about all that great information I told you we would be placing on the form. The easiest way to do that is through code---which is our next step.

Step 3: Write Code for the New Form

Our next step is to write some code so that when the Help-About Form is displayed (more on how to do that in Step 4), the Caption of the label control displays the information that we wish to display about us (and the program). We will also need to write code that will permit the user to close the form when he or she clicks on the OK button.

Let's start with the Caption of the label--we'll place that code in the Load Event Procedure of the form. Here's the code.

```
Private Sub Form_Load()

lblTootYourHorn.Caption = "This program written by John Smiley" & _
  vbCrLf & vbCrLf & "Email: johnsmiley@johnsmiley.com" & _
  vbCrLf & vbCrLf & "Web Site: http://www.johnsmiley.com"

End Sub
```

All we're doing here is displaying some information about the person who wrote the program, and providing an email address and a website. VbCrLF is the Visual Basic Intrinsic Constant for a Carriage Return and Line Feed, which enables us to separate

those three pieces of information with blank lines.

Finally, here's the code to close the Help-About form, which we'll place in the Click Event Procedure of cmdOK.

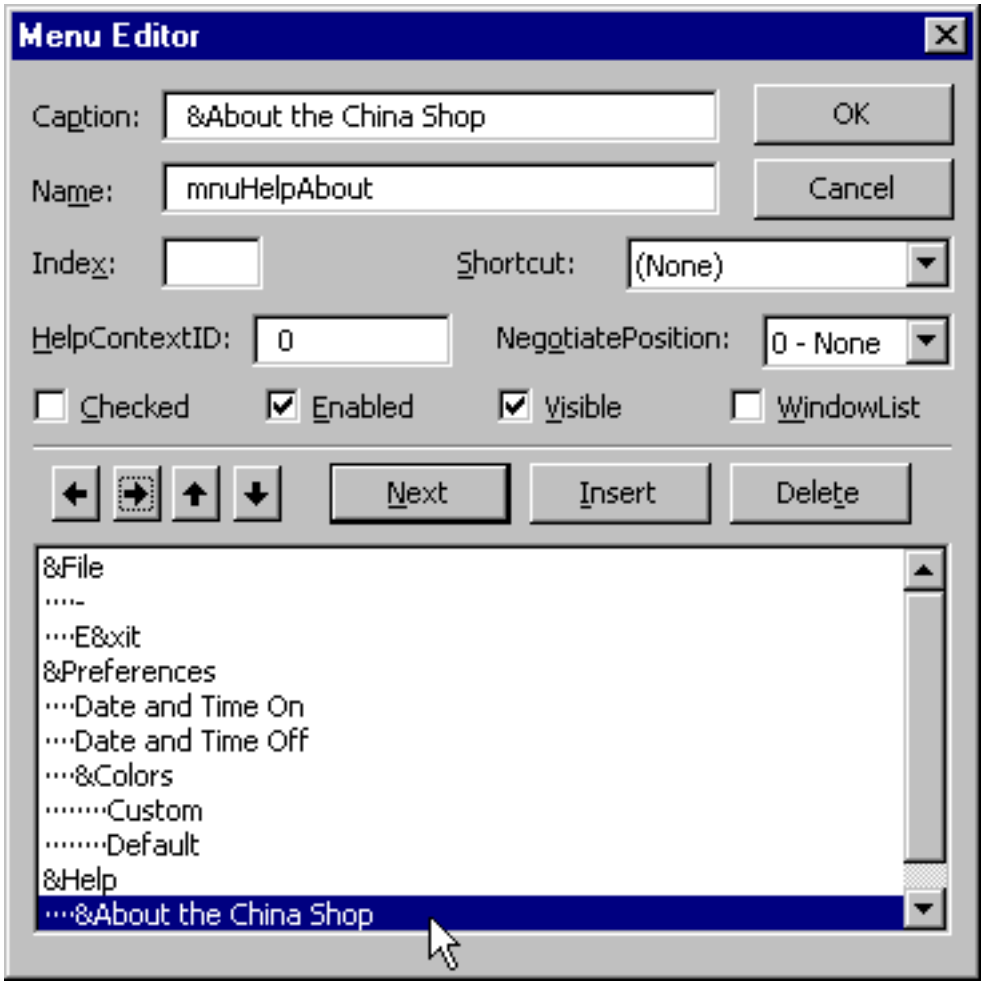
```
Private Sub cmdOK_Click()

Unload Me
Set frmAbout = Nothing

End Sub
```

Simple enough---Unload Me (where Me is just a shortcut reference to the Current Form) will unload the Help-About Form. And the Microsoft recommendation is to set the name of the form you are unloading (with the exception of your startup form) to Nothing---this will free RAM of the remnants of the form and its code.

Now we need to write the code to display the Help About Form.

Step 4: Write Code to display the New Form

What we'll do here is create a Menu entry to display the Help About Form. Not surprisingly, we'll use the Menu Editor to create a menu item called Help, with a submenu item called About. The Menu structure, as seen from the Visual Basic Menu Editor, should look like this…



We'll place the code necessary to display the Help-About form in the Click Event procedure of mnuHelpAbout.

Many beginners I know have been able to get this far on their own---but you're about to see the mistake they make--typically, beginners will read about the Load statement in Visual Basic Help and figure that this statement will do the trick…

```
Private Sub mnuHelpAbout_Click()

Load frmHelpAbout

End Sub
```

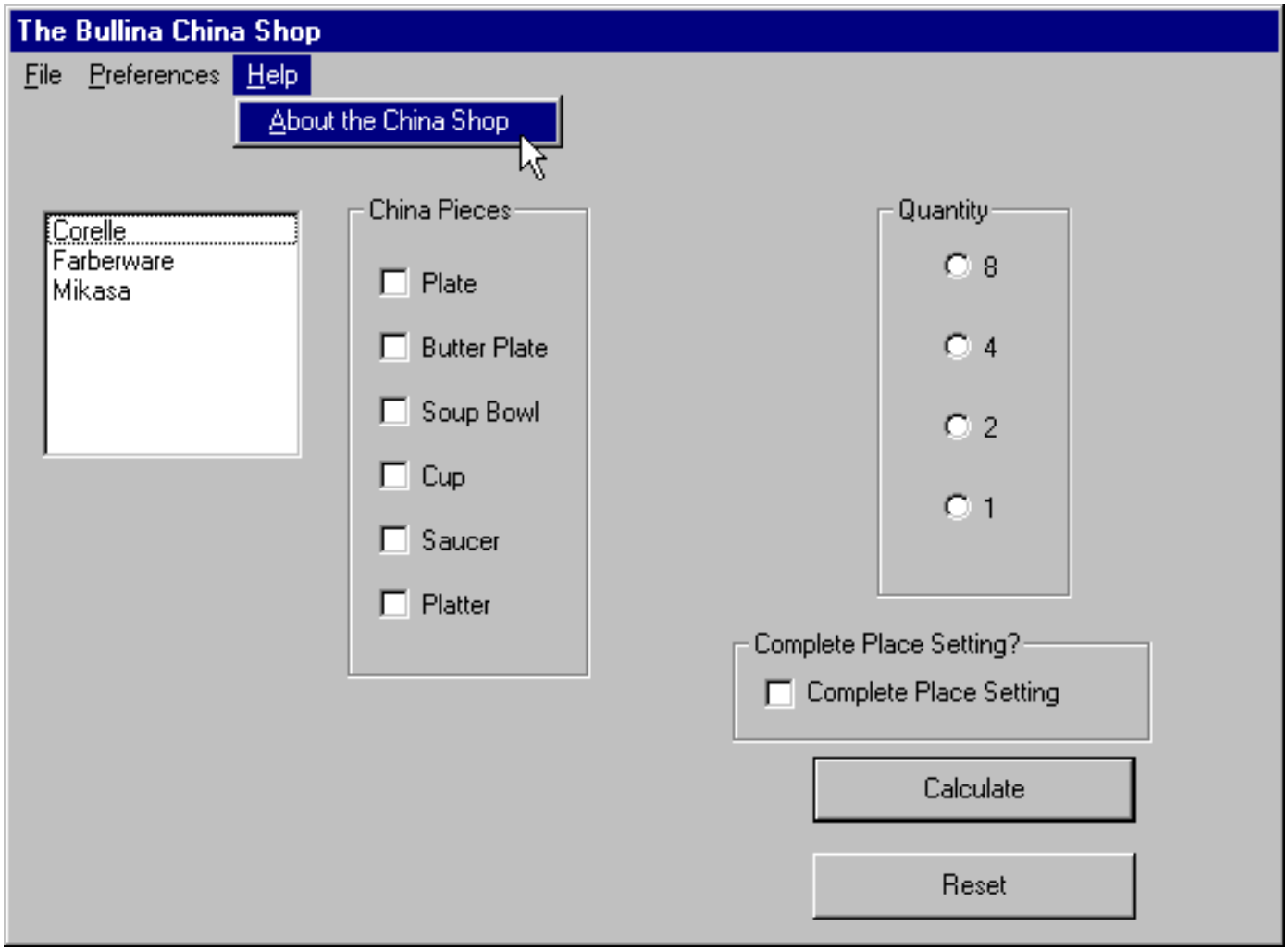
Now before you go off and execute the modified China Shop program, expecting to see the new Help About form displayed when you click on Help-About the China Shop, I should tell you that there's a problem with this code.

The problem with the code is that while it loads the Help-About form, it doesn't display it!

Merely loading a form doesn't make it visible--by default, a loaded form is not visible---it must explicitly be made visible. There are two ways to do that---either set the Visible Property of the form you wish to load to True, or execute its Show Method. As it turns out, the quickest and most efficient way to both load and make the form visible is to just execute its Show Method---which will ***both*** load the form and make it visible. Therefore, this is the code to use.

```
Private Sub mnuHelpAbout_Click()

frmHelpAbout.Show

End Sub
```

Now if you click on the Help-About the China Shop menu item…



the Help-About form will be displayed…

Let's take some time to admire our work, and then click on the OK button. The form should then disappear.

Some nuances…

I should make you aware of some nuances of multiform projects before finishing this month's article.

First, some forms (particularly Help-About forms) should be displayed as ***modal*** forms. A modal form is one that prevents further processing within the application until the form is closed. In terms of the Help-About form, that means that while the form is displayed, the user cannot interact with the rest of the China Shop project. Choosing to display a form as a modal form is a good idea if the form is meant to be an attention getter---one that you want the user to interact with before doing anything else. Displaying a form as a modal form is easy enough---just use the vbModal argument of the Show Method like this…

```
Private Sub mnuHelpAbout_Click()

frmHelpAbout.Show vbModal
```

```
End Sub
```

Now if you run the program, and click on the Help-About form, you'll see that you can't interact with the main form of the China Shop Project until the Help-About form is closed.

Another nuance to keep in mind is that there are occasions when you display the second form that you want to make the first form disappear.

Now that can mean one of two things.

First, you are done working with the first form, and you won't be needing it again. An example of such a form would be a Login form, which permits secure access to an application. Once the user has been authenticated, the Login form is no longer needed, and the main form of the application is displayed. To code this scenario, you would just unload the first form using the Unload Me statement, and then display the second form. Like this…

```
Private Sub mnuHelpAbout_Click()

Unload Me
frmHelpAbout.Show vbModal

End Sub
```

Obviously, we don't want to do that in the China Shop program, because we still want access to the main form of the China Shop available when the user is done reading the Help-About form.

Beginners, at this point, will sometimes make the mistake of unloading the first form, and then re-loading it from the OK button of the Help-About form---like this…

```
Private Sub cmdOK_Click()

Unload Me
frmMain.Show

Set frmAbout = Nothing

End Sub
```

In general, this technique is not a good one.

Loading forms in Visual Basic is a strain on the PC, so if you want to make the first form invisible while the second is displayed, and then redisplay it later, in general, it's a better

idea to Hide the first form instead of unloading it---like this…

```
Private Sub mnuHelpAbout_Click()

Me.Hide
frmHelpAbout.Show vbModal

End Sub
```

Then, when you are done with the Help-About form, use this code to Show the hidden main form…

```
Private Sub cmdOK_Click()

Unload Me
frmMain.Show
Set frmAbout = Nothing

End Sub
```

The main form will instantly appear without having to be re-loaded.

Summary

Now that you know how to add a second form to your project, adding additional forms shouldn't be a problem.