GETTING STARTED WITH VBA

Let the Wizard Guide Them

Click & Retrieve Source CODE!

Create your own wizard to handle functionality for beginning users.

by Chris Barlow

Choosing your target user marks one of the most critical decision points in the design process when you develop a new application. When you get to this point, you must ask yourself, "Is this application designed for the beginning, intermediate, or advanced user?"

Alan Cooper, in his book *About Face: The Essentials of User Interface Design,* explains the paradox facing developers. Here are a few of the points he makes:

• Most developers are expert computer users. They are intimately familiar with the man-computer interface. They love to add cool, new, advanced features to their applications. As a result, they aim the first design of their applications at the expert user.
• The people in the marketing department focus heavily on the application's "ease-of-use." They demand that developers add features to make the application simple for the first-time user.
• Most users, however, are intermediate users. They find much of the application's advanced features too tricky, while the features aimed at beginners become annoying as soon as the intermediate users learn the program.

How can you design your application to satisfy the intermediate users without intimidating the beginners? Wizards are the answer.

Wizards are a nice way to introduce a new user to the unique features of your system. For example, SunOpTech implemented its ObjectOrder System, an order entry system written in VB, in a client's office in Germany. Recognizing that a limited number of users in the order entry department would use this application daily, we intentionally designed the application's main form to satisfy the needs of these intermediate users.

The goal was to present the important information about the order on a single form so that the user could see all the fields at a glance. We made a conscious decision not to use additional forms or even a tab control to hide some of the fields. This design made it easy for the user to display an order and quickly answer the customer's question about any of the fields on the order. It also speeded up order entry because all the fields on the order were available with the Tab key.

The resulting form design was extremely busy (see Figure 1). Although regular users of the system loved the ability to quickly see any field on the order, beginning users invariably complained about the complexity of the form. Several weeks passed before they, too, were happy with the form.

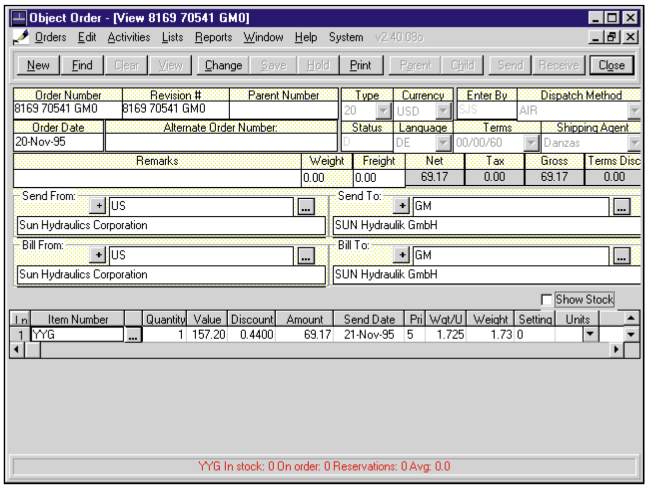
The solution was to develop an OBOrder Entry Wizard. This additional form guides the beginner through the process of filling out an order entry form. A new user can use the wizard until he or she becomes comfortable with the normal form. The code for the comprehensive OBOrder Wizard application is available to members of the Premier Club of The Development Exchange (for details, see the Code Online box at the end of this column).

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank and the ObjectJob Systems. 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.

WRITE YOUR OWN WIZARD

The wizard concept is not new. Microsoft has used wizards in many of its applications to step a user through a complex function. With some simple Visual Basic code, you can easily add your own wizard to your new application.

The function of a wizard is fairly simple. It displays a form with a graphic image and some text explaining the purpose of the wizard. When the user presses the next button, the wizard changes the frame to display various controls with simple instructions on how to fill out the data. The user can press the Back button to make a change in something already entered. On the last frame, the user can press the Finish button to complete the process. Let's step through the development of the Order Entry Wizard (see Listing 1 for the complete code for this wizard).



Orders Edit Activities Lists Reports Window Help System v2.40.08o
New Find Clear View Change Save Hold Print Parent Child Send Receive Close
Show Stock

Object Order - [View 8169 70541 GM0]

Order Number	Revision #	Parent Number	Type	Currency	Enter By	Dispatch Method
8169 70541 GM0	8169 70541 GM0		20	USD	SJS	AIR

Order Date	Alternate Order Number:	Status	Language	Terms	Shipping Agent
20-Nov-95		D	DE	00/00/60	Danzas

Remarks	Weight	Freight	Net	Tax	Gross	Terms Disc
	0.00	0.00	69.17	0.00	69.17	0.00

Send From: US — Sun Hydraulics Corporation
Send To: GM — SUN Hydraulik GmbH
Bill From: US — Sun Hydraulics Corporation
Bill To: GM — SUN Hydraulik GmbH

Ln	Item Number	Quantity	Value	Discount	Amount	Send Date	Pri	Wgt/U	Weight	Setting	Units
1	YYG	1	157.20	0.4400	69.17	21-Nov-95	5	1.725	1.73	0	

YYG In stock: 0 On order: 0 Reservations: 0 Avg: 0.0

FIGURE 1 ***Present Information at a Glance.*** *This screen, designed for the intermediate user, is intentionally full. Notice how you can see the entire order at a glance.*

Start with a new Visual Basic project. Using the default form, set the BorderStyle property to Fixed Dialog so that the form resembles a dialog without a Control Box or resizing borders. Add to the form a Frame control large enough for the graphics, text, and controls for each step. Now add an Image control on the left half of the frame for the graphic image.

For each of the steps in your wizard, add one of these frames to the form in a control array. Select the Frame control, press Ctrl-C to copy it to the clipboard, and then press Ctrl-V to paste a copy on the form. Click on the Yes button when Visual Basic asks if you want to create a control array for both the frame control and the image control.

Press Ctrl-V three more times so that you have five frames with indexes zero through 4. You can locate these forms anywhere on the form because you'll write code to position the frame in the proper location at run time. I usually change the Caption property of each frame to *Step 1, Step 2,* and so on so that I can easily determine which frame I'm working with.

As you place these frames, you may notice that one frame lies behind another. In other words, even though the display of the frame's grab handles tells you that you selected the frame, part of the frame remains hidden by another frame positioned in front of it.

Visual Basic actually maintains three graphical layers for every form or container. All nongraphical controls, like these frames, are displayed in the front layer. All controls on this layer display in front of the middle layer, which displays graphical objects and label controls. The back layer, which lies behind all controls, displays methods such as Form.Print.

Visual Basic has a method called ZOrder, which arranges the display of objects within a single layer. At run time, you can call the object's ZOrder method and pass an argument of 1 so that the object displays behind all other objects on that layer. If you pass an argument of zero or no argument, the object displays in front of all other objects on that layer. In the Visual Basic editing environment, you can use the Ctrl-J and Ctrl-K key combinations to move objects to the front and back of the ZOrder.

Place four buttons below these controls and set the captions to *Help*, < *Back*, *Next* >, and *Cancel* (see Figure 2).

Each time the user presses the Next button, you want the Wizard to display the next frame to guide the user through the process. Each frame will have different text and controls that you customize for each step in this process.

For example, the OBOrder Entry Wizard has five frames to guide the user through the five steps of creating an order:

1. Selecting the order type.
2. Choosing the Send To location.
3. Choosing the Send From location.
4. Entering miscellaneous information.
5. Entering detail order lines.

You can put any type of controls you need on each frame to complete that step. The first frame contains a RichTextBox control with an introduction to the wizard and four option button controls that allow the user to select the type of order he or she wants to create.

SunOpTech's ObjectOrder system is different from most order entry systems. Most systems make an artificial distinction between customer orders and purchase orders by maintaining them in separate data files. These systems have real trouble dealing with drop shipments: one of your vendors directly ships a product to one of your customers. The ObjectOrder system treats all orders in the same manner, and varies the sending and receiving locations.

CONCEPTION BEFORE CREATION

Even before you decide on the exact format of each frame in your wizard, you should add some code to the form to navigate through the wizard. This gives you a chance to visualize how the wizard will look and test it as you add the text, controls, and actions for each frame.

First, define a variable to store the index of the current frame being displayed and the maximum index of frames:

```
Option Explicit
Dim iCurFrame As Integer
Const MaxFrameIndx = 4
```

Then add code for the Next button to step through the frames. If the current frame equals the maximum frames, the process is complete and you'll need to add

VB4

```
Option Explicit
Dim iCurFrame As Integer
Const MaxFrameIndx = 4

Sub ProcessStep(iFrame%)
Select Case iFrame
Case 0 'Order Type
  If OptType(0) Then
  'if CO then set the SFR
      txtSFR = "Sun Hydraulik GmbH"
  ElseIf OptType(1) Then
  'if PO then set the STO
      txtSTO = "Sun Hydraulik GmbH"
  ElseIf OptType(3) Then
  'if RG then set STO
      txtSTO = "Sun Hydraulik GmbH"
  End If
Case 1 'Send To
Case 2 'Send From
Case 3 'Misc
Case 4 'Items
End Select
End Sub

Private Sub butNext_Click()
ProcessStep (iCurFrame)
If iCurFrame = MaxFrameIndx Then
  'finished so process order
  'add code to process order here
  MsgBox "Order created!"
  Unload Me
Else
  iCurFrame = iCurFrame + 1
  Frame1(iCurFrame).Left = _
      Frame1(0).Left
  Frame1(iCurFrame).Top = _
      Frame1(0).Top
  Frame1(iCurFrame).Visible = True
  Frame1(iCurFrame).ZOrder
  butBack.Visible = True
End If
If iCurFrame = MaxFrameIndx Then
  butNext.Caption = "Finish"
End If
End Sub

Private Sub butBack_Click()
Frame1(iCurFrame).Visible = False
butNext.Caption = "Next >"
iCurFrame = iCurFrame - 1
If iCurFrame = 0 Then
  butBack.Visible = False
End If
End Sub

Private Sub butCancel_Click()
Unload Me
End Sub

Private Sub butNextItem_Click()
'add code to save this item info
'then clear for next item
txtItem.Text = ""
txtQty.Text = ""
txtDate.Text = ""
End Sub

Private Sub Form_Load()
Dim i%
For i = 1 To MaxFrameIndx
  Frame1(i).Visible = False
Next
butBack.Visible = False
End Sub

Private Sub Form_Unload(Cancel _
  As Integer)
End
End Sub
```

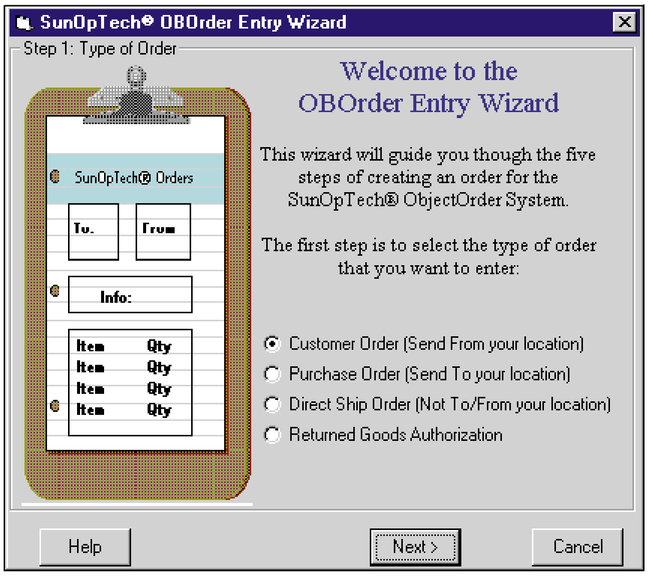
LISTING 1 ***OBOrder Entry Wizard Code.*** *You can create the functionality for a simple wizard with little code.*

code to complete the order entry process:

```
Private Sub butNext_Click()
If iCurFrame = MaxFrameIndx Then
   'finished so process order
   'add code to finis-h order
   MsgBox "Order created!"
```

However, if the last frame is not already displayed, you will need to display the next frame by making it visible and orienting it in the proper location over the first frame:

```
Else
   iCurFrame = iCurFrame + 1
   Frame1(iCurFrame).Left = Frame1(0).Left
   Frame1(iCurFrame).Top = Frame1(0).Top
   Frame1(iCurFrame).Visible = True
```



Welcome to the
OBOrder Entry Wizard

This wizard will guide you though the five steps of creating an order for the SunOpTech® ObjectOrder System.

The first step is to select the type of order that you want to enter:

(●) Customer Order (Send From your location)
( ) Purchase Order (Send To your location)
( ) Direct Ship Order (Not To/From your location)
( ) Returned Goods Authorization

FIGURE 2 ***The OBOrder Entry Wizard.*** *When the class is nearing completion, I create a more comprehensive test form to really put it through its paces.*

VB4

```
Sub ProcessStep(iFrame%)
Select Case iFrame
Case 0 'Order Type
  If OptType(0) Then
  'if CO then set the SFR
     txtSFR = "Sun Hydraulik GmbH"
  ElseIf OptType(1) Then
  'if PO then set the STO
     txtSTO = "Sun Hydraulik GmbH"
  ElseIf OptType(3) Then
  'if RG then set STO
     txtSTO = "Sun Hydraulik GmbH"
  End If
Case 1 'Send To
Case 2 'Send From
Case 3 'Misc
Case 4 'Items
End Select
End Sub
```

LISTING 2 ***Branch Your Code.*** *Use the Select Case statement to branch your code based on the frame you just processed.*

```
   Frame1(iCurFrame).ZOrder
   butBack.Visible = True
End If
```

Finally, if the last frame is now displayed, you should change the caption of the Next button to "Finish":

```
If iCurFrame = MaxFrameIndx Then
   butNext.Caption = "Finish"
End If
End Sub
```

The code under the Back button is similar—cycle through the control array and set the appropriate button captions:

```
Private Sub butBack_Click()
Frame1(iCurFrame).Visible = False
butNext.Caption = "Next >"
iCurFrame = iCurFrame - 1
If iCurFrame = 0 Then
   butBack.Visible = False
End If
End Sub
```

Now add this code in the Form_Load event to set the Visible property for all but the first frame to False:

```
Private Sub Form_Load()
Dim i%
For i = 1 To MaxFrameIndx
   Frame1(i).Visible = False
Next
butBack.Visible = False
End Sub
```

Now run your project and try the buttons. You should be able to scroll forward and backward through the frames. You have an operating wizard—simply add some meaningful code to handle each step. I suggest you add a procedure called ProcessStep and pass the current frame index as an argument. Then call this procedure at the beginning of the butNext_click and butBack_click events to handle the processing necessary for that step.

For example, you can use the Select Case statement to branch your code based on the frame just processed. Fill in the proper location information in the code (see Listing 2). It's easy to add the functionality you need for your wizard. Your beginners will thank you for doing so.

Code Online

You can find all the code published in this issue of VBPJ *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* VBPJ *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.*

Let the Wizard Guide Them

Locator+ Codes

Listings ZIP file (free Registered Level): VBPJ1196

Listings for this article plus code for the comprehensive OBOrder Wizard application (subscriber Premier Level): GS1196P