Column: Visual Basic for Dataheads
Figures: NONE
File for Companion Disk: None

Creating the User Interface: VB Forms and Controls

Whil Hentzen

Now that we have covered the language, we've got a good foundation under our belt. It's time to look at the tools that we'll use to create the most interesting part of the application – from the user's perspective, that is. Much like an automobile and an eighteen-wheeler, some of the tools and techniques will be familiar to experienced VFP developers, but other techniques and components are either brand new, or just simply different. Let's explore.

It's difficult to do justice to the topic of "user interface" in a few pages, so I'm going to blast through this, and cover the big differences in two areas. First, I'll discuss the mechanisms you use to manipulate forms and controls in Visual Basic, and then I'll go through the more common controls and compare and contrast them with their VFP brethren.

Creating a form and dropping a control on it

Silly enough, it took me the better part of a day to get the hang of putting controls on a VB form. You don't click on the control in the toolbox and then on the form, and expect the control to be drawn for you like in VFP. Instead, either click on the control, and then click and drag the control on the form – just clicking on the form doesn't do anything. Alternatively, you can just double-click on the control in the toolbox – the control will be placed in the center of the form. If you double-click on a control in the toolbox multiple times, you'll end up with several controls on top of each other in the center of your form.

Moving a control is a little different – the cursor keys don't nudge a selected control or group of controls bit by bit. Instead you have to hold the Ctrl key down in order to nudge them one way or another. You'll also want to take note of the Format, Lock Controls menu option – selecting it will turn all of the sizing handles to white, and you'll not be allowed to use the mouse to move or resize the controls from then on. Evidently accidental moves by novice VBers was a problem at some point in time. You can, however, still use the keyboard to move and resize as you want.

Alignment is one of those "half-full, half-empty" issues. You can use the Tools, Options command to display or hide the grid on a form, define the granularity of grid lines, and to have controls snapped to grid lines or not. You can also use the Format, Align menu command to align a group of selected controls, much as you do in VFP. And if you're like me, you never use the menu command in VFP, preferring to use the Layout toolbar. Unfortunately, if you look for a Layout or an Alignment toolbar in VB, you won't find one. Instead, select the Form Editor toolbar. It looks sparse, but you'll see that each button opens up into a bunch of related choices, which I think is pretty cool. It just took a while to get used to it – it's still two mouse clicks instead of one.

VB's intrinsic controls

Visual Basic is considerably different from VFP in that hardly any VB developers just accept the controls that come "in the box" as all they'll use. The whole idea behind VB was to allow the developer to extend the environment by using additional controls as they desired – truly, very few VB developers toolboxes look alike.

However, there are twenty controls that come with VB and appear automatically on the toolbox's toolbar. In this article, I'll most of these. If you dock the Toolbox to the left of the VB IDE, you'll see them as shown in Figure 1.

05VB01.TIF Visual Basic comes with twenty native controls.

The Label control is similar to VFP's label, and has many of the same properties.

The TextBox control is similar to it's VFP counterpart, but also does double duty to serve as VFP's editbox match. The Value property in VFP is called the Text property in VB for this control. You can set the MultiLine property to True and then enter more than one line of text – where you'd use an EditBox in VFP to do the same thing. If you have MultiLine set to True, you can use the ScrollBars property to display them as desired.

The CheckBox control is similar to VFP's CheckBox. It can contain a value of 0 (not checked), 1 (checked), or 2 (disabled). You have to be careful about using a value of 2, because once a checkbox's value has been set to 2, you can't tell whether it's checked or unchecked. 2 (Disabled) is also different from setting the Enabled property to False; the former keeps the checkbox's caption enabled (black) while the latter also dims the caption, turning to a fuzzy grey/white combination.

The CommandButton is VB's answer to VFP's command button – and a button is a button is a button.

The OptionButton sort of maps to an option button in VFP, but there are a couple of significant differences. If you've already placed an option button on a VB form, you may be wondering, "Where's the Option Group control?" And the answer is… There isn't any! All of the option buttons you place on a form – regardless of *where* on the form – are part of the same option group.

Blech, you say? Me too. It is a bit easier, of course, to be creative with placement, of course – don't you hate having to enlarge an option group, and then edit the group, and then align the buttons where you want them? Kind of a pain. In VB, this is easier.

But the flip side is that if you want more than one option group on a form, you have to build your own container first, using a Frame control, and then add individual option buttons. Like I said, "Blech."

A group of option buttons (I didn't say "An Option Button Group", did I?) works just like you would expect – selecting one deselects another that had been selected. You need to write code to deselect all of them, like the following fragment (you could put this in the click() method of a command button, for example):

```
Option1.Value = 0
Option2.Value = 0
Option3.Value = 0
```

The ComboBox and ListBox controls are similar to VFP's combo box and list box controls, and I think they're actually somewhat easier to use "out of the box."

The ComboBox control is similar to VFP's combo box, complete with multiple modes and a variety of properties. You can set the Style property to one of three possibilities. 0 (DropDown Combo) means you can enter text or click on the arrow to open the combo. 1 (Simple Combo) means the combo box is always open but you can also enter text if you want. It's a little unusual – you need to resize the combo box to display more than one item – and the result ends up looking like a text box with a list box placed below it on the form. See Figure 2. 2 (DropDown List) means that you can't enter text, and you have to open the control in order to select a value.

05VB02.TIF A Simple Combo box looks like a text box and a list box together.

The easiest way to add items to a ComboBox or ListBox is with the AddItem method, like so:

```
List1.AddItem "A"
List1.AddItem "B"
List1.AddItem "cccccc"
```

You can make a ListBox multi-selectable by setting the Mult-Select property to 1 (Simple) or 2 (Extended). Simple means that you can just keep clicking on items and they stay selected (clicking a second time deselects the item.) Extended means you can use the Ctrl and Shift keys to select a range of items – such as the 3rd, 4th, 5th, 9th, and 12th.

You can determine which item has been selected in a ListBox with the Text property (just as you would use the DisplayValue property in VFP.) If you have Multi-Select set to 1 or 2, the Text property displays the last item selected. In order to determine all those items selected, use the Selected property together with the ListCount property, much as you would in VFP:

```
Dim nItemNumber As Integer
nItemNumber = 0
Do While nItemNumber < List1.ListCount
  If List1.Selected(nItemNumber) = True Then
    MsgBox "Hey, I'm selected " & nItemNumber + 1
  End If
  nItemNumber = nItemNumber + 1
Loop
```

Note that the index of the array that populates the ListBox starts with zero here, not 1 as in VFP.

The Shape and Line controls allow you to create a variety of lines, circles and rectangles on your forms, much like the same controls in VFP do. The Shape property of the Shape control allows to you to choose between Rectangle, Square, Oval, Circle, Rounded Rectangle, and Rounded Square.

The Image and Picture controls allow you to place graphic files on your form. The Image control is lightweight, but doesn't allow you to overlap controls (and thus, images), and can't receive focus. The Picture control, on the other hand, can be overlapped and can receive focus – which makes it a good candidate to use when creating graphical controls.

The DirListBox control, in combination with the DriveListBox and the FileListBox controls, allow you to build file navigation dialogs quickly and easily.

The Frame control is used as a container, and I'll cover it, together with the Horizontal and Vertical Scroll Bars, the Timer, the OLE control, and the Data control, in another article.

More controls

There are, of course, more controls than just these twenty. Some ship with Visual Basic, others are produced by third parties and you have to beg, borrow, steal (or pay for!) them.

Those that ship with VB are also installed when you install VB; all you have to do is tell VB that you want access to them. In order to do so, right-click in the Toolbox and select the Components menu command. This brings forward the Components dialog as shown in Figure 3.

05VB03.TIF The Components dialog is used to add controls to the Toolbox.

As with VFP, check off the controls you want to have displayed in the Toolbox, and then select OK. Note that this dialog is much nicer than the one we get in the Controls tab of the Tools, Options dialog in VFP. How much nicer? Oh, let me count the ways. First, you can see more of the name of the control than in VFP – and if the name is too long to fit in the list box, you can scroll the list box to the right and left – in VFP, you're just hosed. Second, you can see exactly which file on disk represents the control – in VFP, you're just supposed to guess, I guess. You can also choose between Controls, Designers and Insertable Objects in three different tabs. Finally, you can choose to see just those objects that you've selected – quite nice when you've selected three of two hundred available. In VFP, get your pen and a piece of paper out unless you've got a really good memory. (I've asked for these improvements, but you know the refrain – so many ERs, so little time.)

Once you've selected some controls, they'll show up in the Toolbox along with the original twenty. You may not want your Toolbox littered with so many; right-click on the Toolbox, select the Add menu command, and add a tab to the Toolbox. Then drag your controls to the new tab. You can organize your controls any which way you want using this mechanism.

I'll discuss a couple of the more interesting ones in the next article – when we talk about data!

Conclusion

So what do you think? After months of working with VB, I feel sort of like I did when moving from 2.6 to 3.0 – there were clearly some things I liked a lot more, but I still felt much more comfortable with "the old way." Same here. Fortunately, it's not a question of choosing one over the other; remember – different tools for different purposes. Next month, it's time to let the genie out of the bottle: we'll look at data.

Whil Hentzen is editor of FoxTalk. whil@hentzenwerke.com.

<< none this month>
make this a sidebar

Stupid VB Tricks

Differences between VB and VFP, and inconsistencies in the VB environment that can "gotcha" if you're not careful or observant.

Stupid VB trick #6: