Use the UpDown Control in your Visual Basic 6 Projects

One of the more common requirements that my students and readers have is for the user of their program to make a numeric selection of some kind. For instance, in the case of a program where the user must select a quantity, immediately a textbox comes to mind.

One of the things I preach in my classes is the need to reduce, or possibly eliminate, the user's attempt to make entries using the keyboard---whenever possible, allow them to use their mouse to make selections, either using checkbox, option Boxes, or selections from a ListBox. Having the user type an entry into a Textbox is an invitation to disaster---there's no easy way to control what they enter.

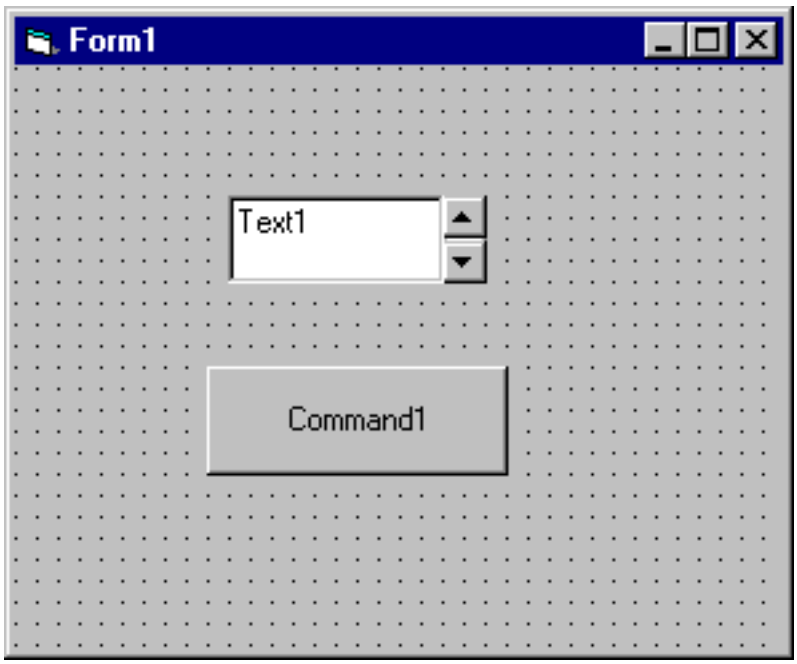
For instance, if you want to restrict the allowable entries in a textbox to a range of numbers between 1 and 10, you can do so--but you'll need to write some code and place it in the KeyPress event procedure of the Textbox. Unfortunately, while the KeyPress event procedure is a great place to 'intercept' single character strokes, and discard any keystroke that doesn't meet your requirements, as soon as you need to validate something other than a single keystroke (as in the number '10')---the KeyPress event procedure becomes more difficult to work with.

The Vertical ScrollBar Control is NOT the UpDown Control

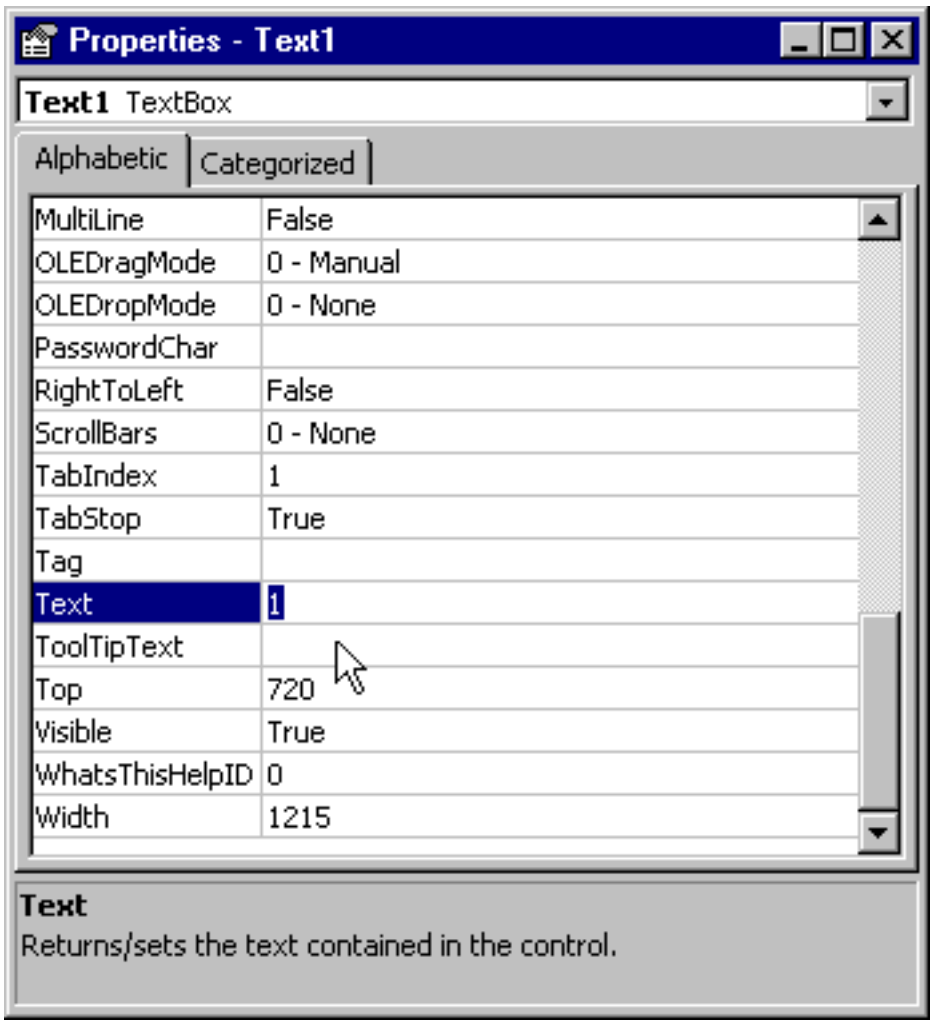
The desire to limit what a user enters into a Textbox is typically when a beginner programmer will discover the Vertical Scrollbar control--which is NOT the UpDown Control, by the way, but let's discuss it first, and then you'll see the true wonder of the UpDown Control.

The Vertical ScrollBar is typically discovered before the UpDown control because it's already visible in the Visual Basic Toolbox…

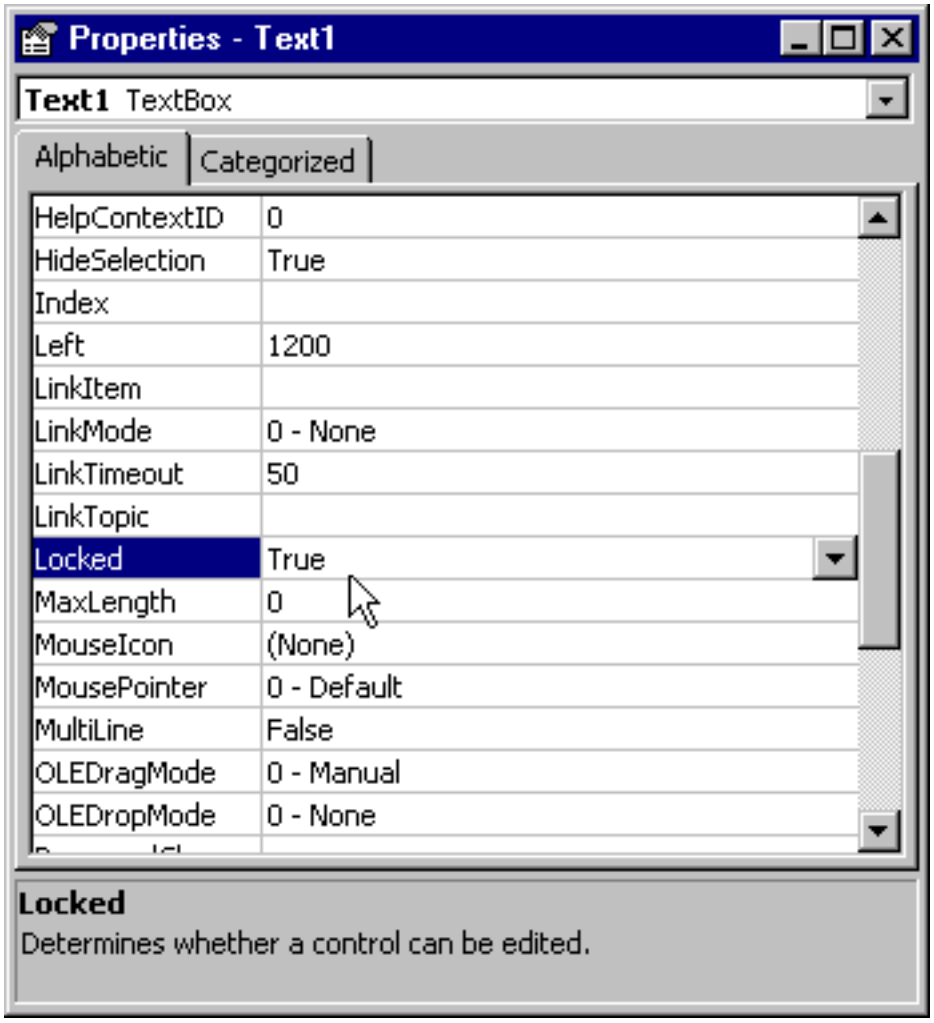
Let's place the Vertical ScrollBar Control and a Textbox on a form now, and we'll see how the Vertical Scrollbar can provide us with some control over the selection of a quantify by the user of our program…



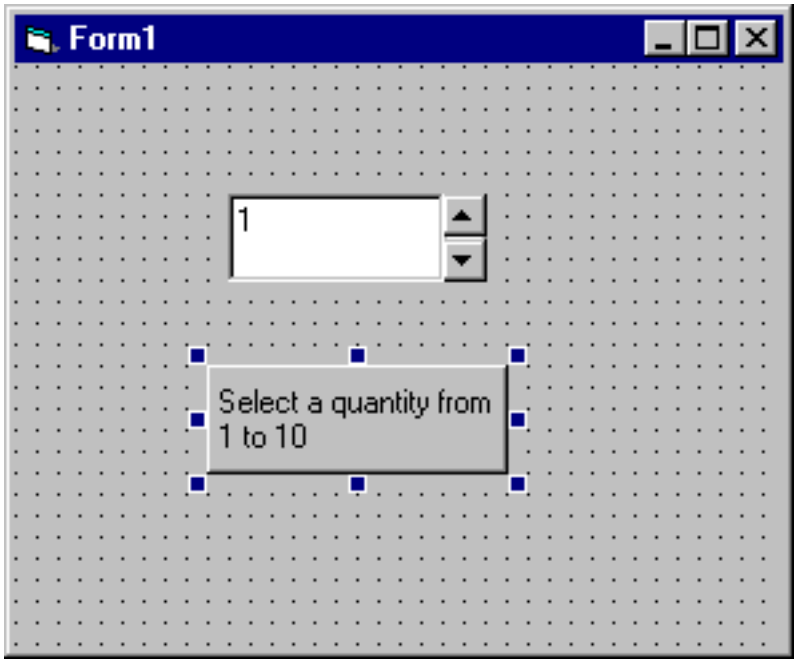
We'd like to 'initialize' the value of the Textbox to 1, indicating that this is the minimum quantity the user can select. We can do that by changing the Text Property to 1…



We need to prevent the user from typing into the Textbox directly, and we can do that by setting the Locked Property to False..



While we're at it, we'll provide a friendly caption on the Command Button for the user…



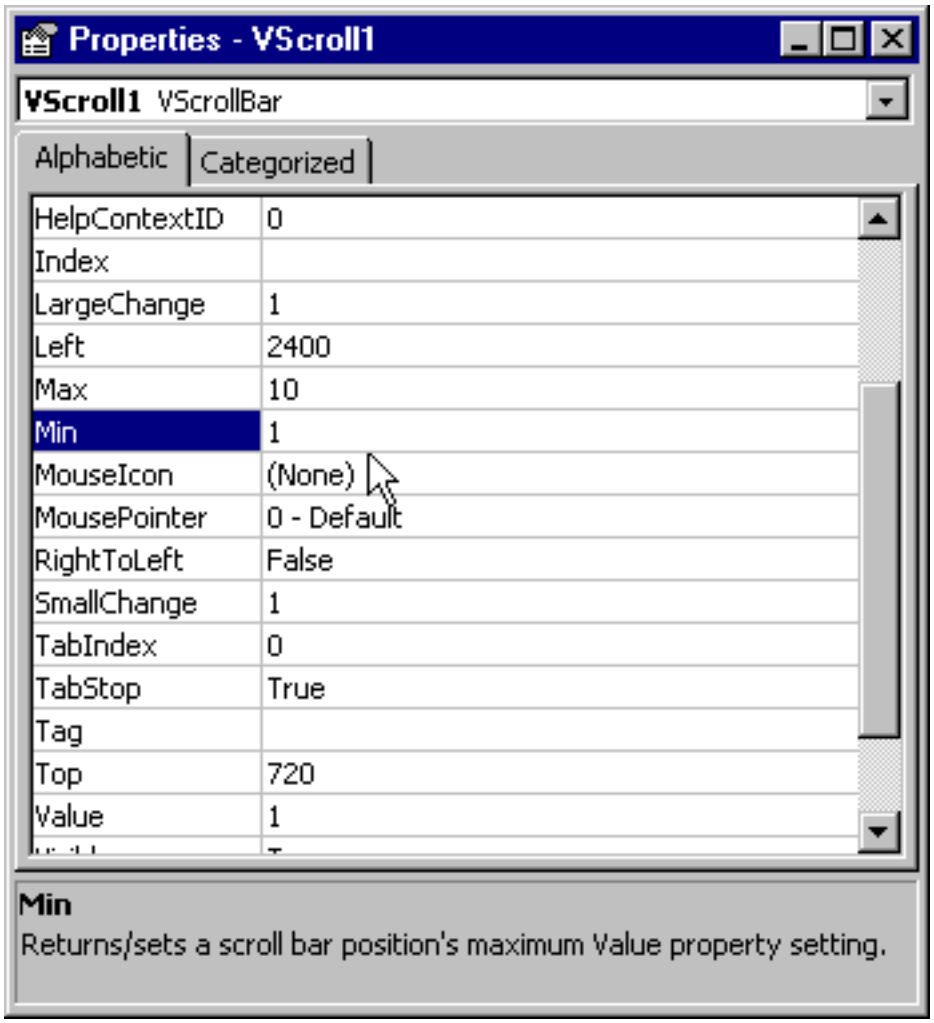
Now what about that Vertical Scrollbar? How exactly can that be used to

update the value in the Textbox?

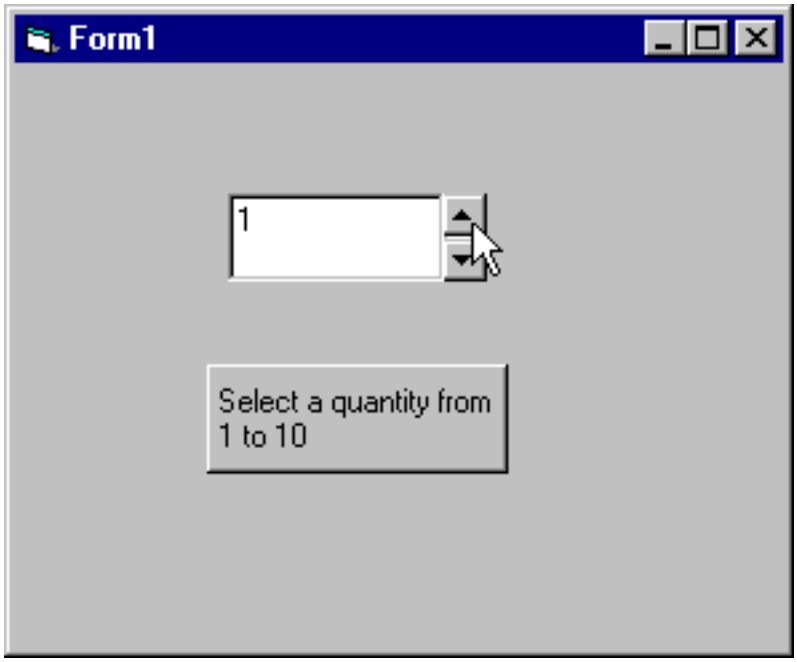
Set some Properties

The Vertical ScrollBar has several properties that will help us--they are: Min, Max, and Value. Behind the scenes of the Vertical Scrollbar, there is a Value property, which is a numeric value. By default it is 0, and when the 'up arrow' on the Vertical Scrollbar is clicked, the value ***decreases*** by one. When the 'down arrow' on the Vertical Scrollbar is clicked, the value is ***increased*** by one (yes, that's correct--it's not a typo.)

We can specify the Minimum (using the Min Property) and the Maximum (using the Max Property) that we will permit the Value Property to reach. For this example, we'll set Min to 1 and Max to 10…



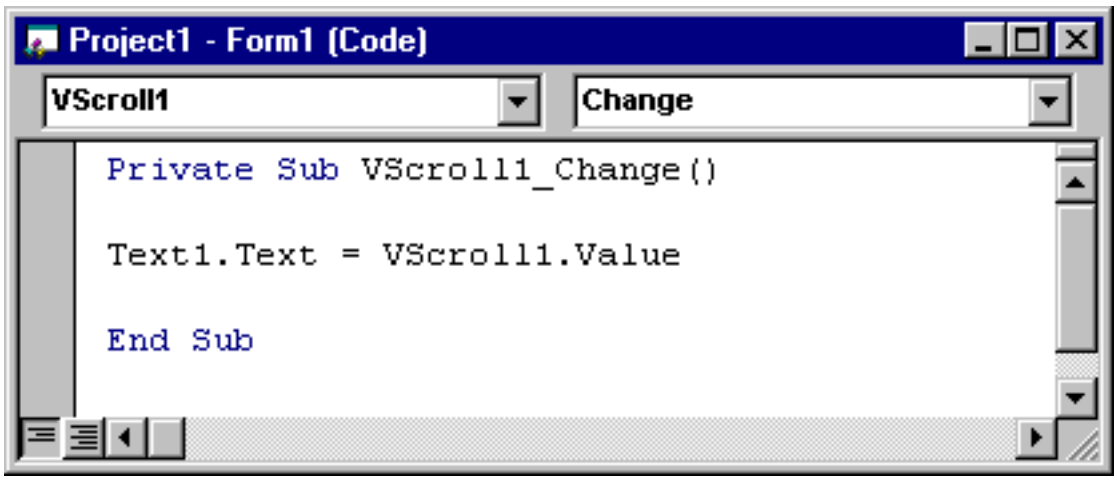
If we now run this program, and click on the scrollbars of the Vertical Scrollbar, you'll see the greatest weakness of the Vertical Scrollbar---the value in the Textbox never changes…



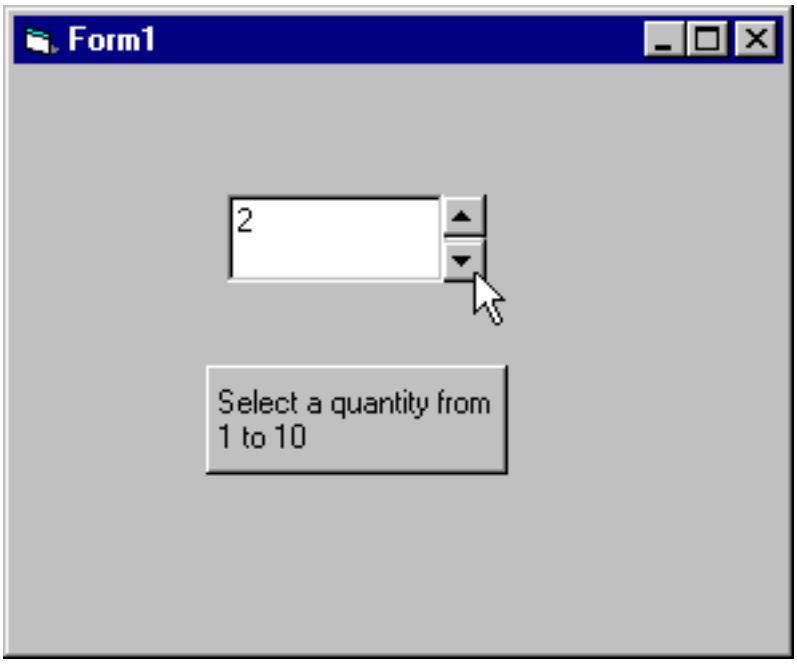
although the Value Property of the Vertical Scrollbar control is changing. (This is the enhancement the UpDown Control gives us--but more on that in a moment or two). As a result, the Textbox value remains at its default of 1 (this causes beginners quite a bit of confusion.)

Write some code…

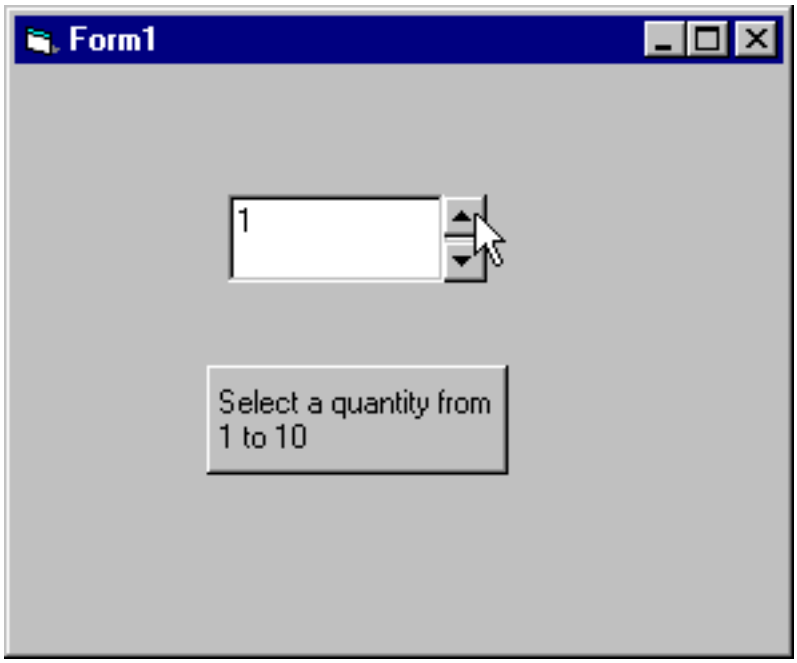
This is not a big deal---the Vertical ScrollBar contains a Change event, into which we can place code to set the Text Property of our Textbox equal to the Value Property of the Vertical ScrollBar. The Change Event is triggered whenever the Value Property changes, which occurs when the user clicks on the up or down arrow. Here's the code…



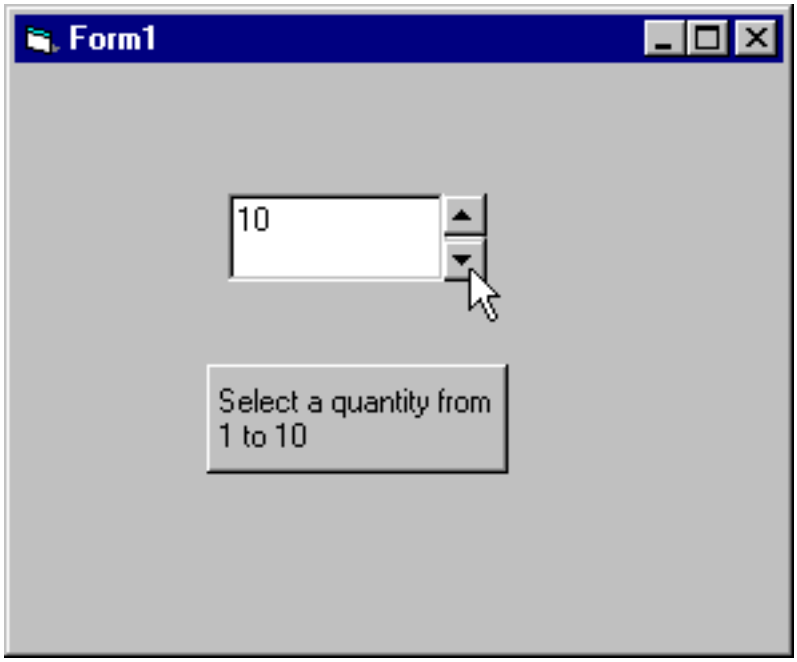
If we now run the program, and click on the up or down arrows of the Vertical Scrollbar, the Textbox value will change accordingly. Annoyingly, to advance the value by 1 we need to click on the down arrow…



To reduce the value, we need to click on the Up Arrow---seems like it should be the other way around, doesn't it?



The great thing about using the Vertical Scrollbar, is that it eliminates the need for the user to type something into the textbox, and there is built in validation as to the limits of values they can enter. If the user scrolls to 10…



and then tries to increase the value beyond that, the number stills reads 10--because of the value we placed in the Max Property of the Vertical Scroll Bar.

So what's wrong with the Vertical Scrollbar?

Well, we've already seen that the up and down arrows appear to be reversed (at least I think so). Most people would agree when the up arrow is clicked, you would expect the Value property to increase. Most people would also agree that when the down arrow is clicked, you would expect the Value property to decrease.

Something else that annoys people about the Vertical Scrollbar (and you can't see it here in the screenshots) is that if there are no other controls on your form capable of receiving focus, the Vertical Scrollbar will blink intermittently, which can be very distracting.

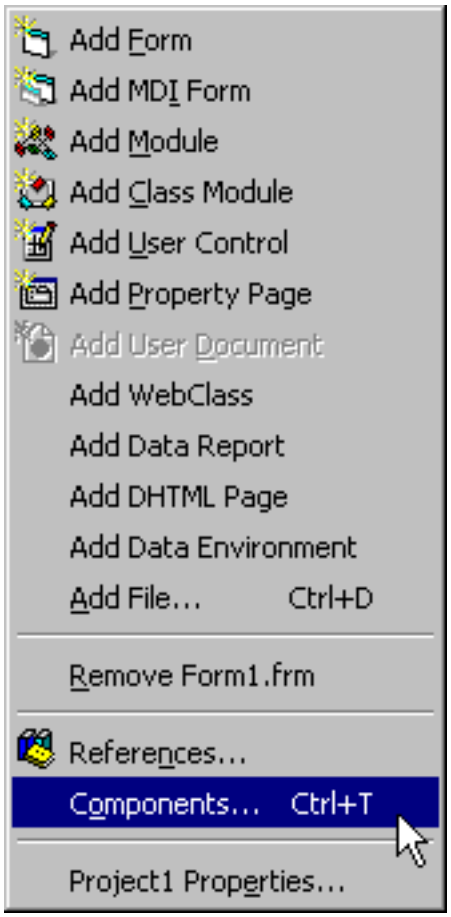
And finally, wouldn't it be nice if you didn't have to write your own code to update the Text Property of the Textbox?

The UpDown Control

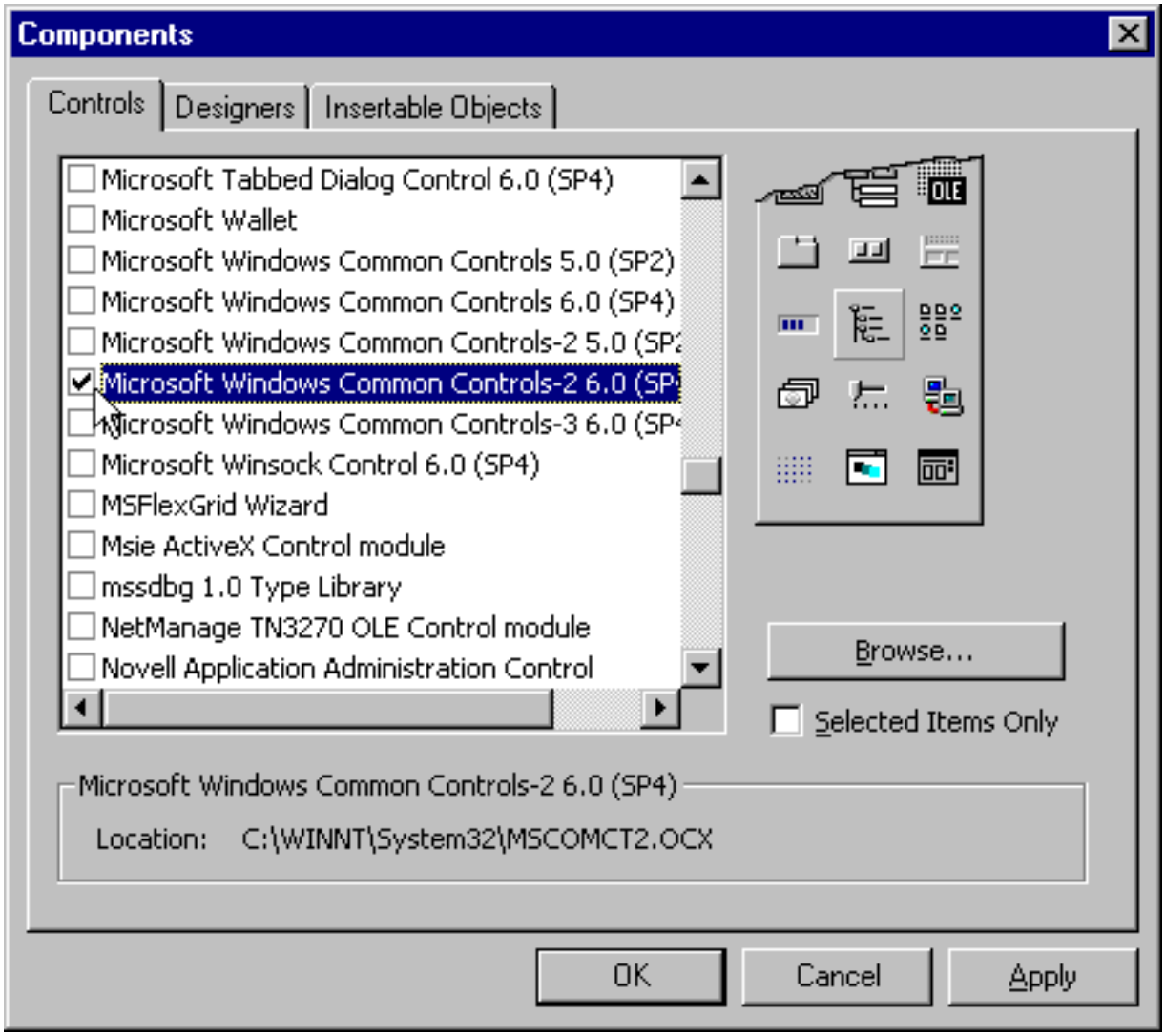
These deficiencies were all taken care of by the UpDown Control. The biggest impediment in using the UpDown Control is to find it---it doesn't appear in the Visual Basic Toolbox. It's contained in a library of additional controls that may (or may not) come with your version of Visual Basic---you'll need to check.

Let's delete the Vertical Scrollbar control from our form now, and work with the UpDown Control instead. To find the UpDown Control, select Project-

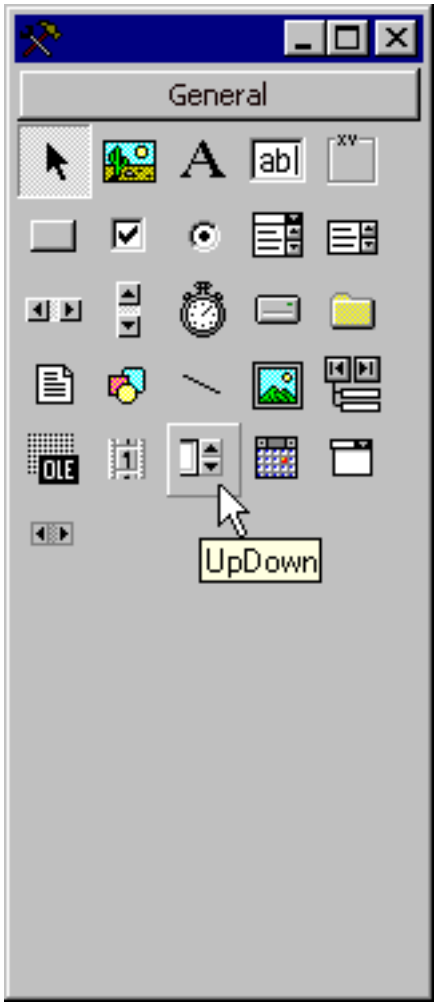
Components from the Visual Basic Menu Bar…



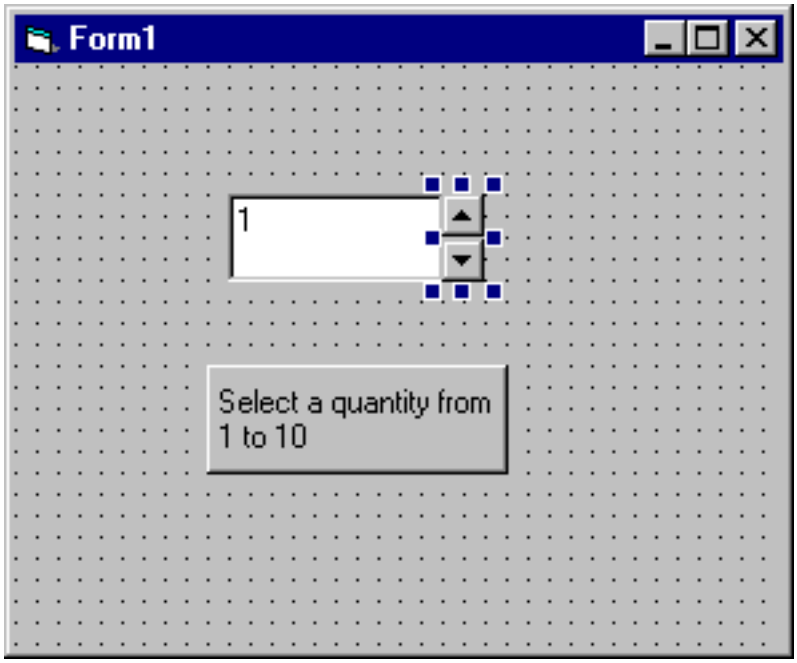
Once the Components Window appears, we need to go hunting for the Microsoft Windows Common Controls-2 Control (select the latest version you have--as you can see in the screen shot, I have the same control from Visual Basic 5 still resident on my PC)



As soon as you click on the OK button, the UpDown Control (and some others as well) will appear in the Visual Basic Toolbox...



Let's select the UpDown control, and place it on our form right next to the Textbox…



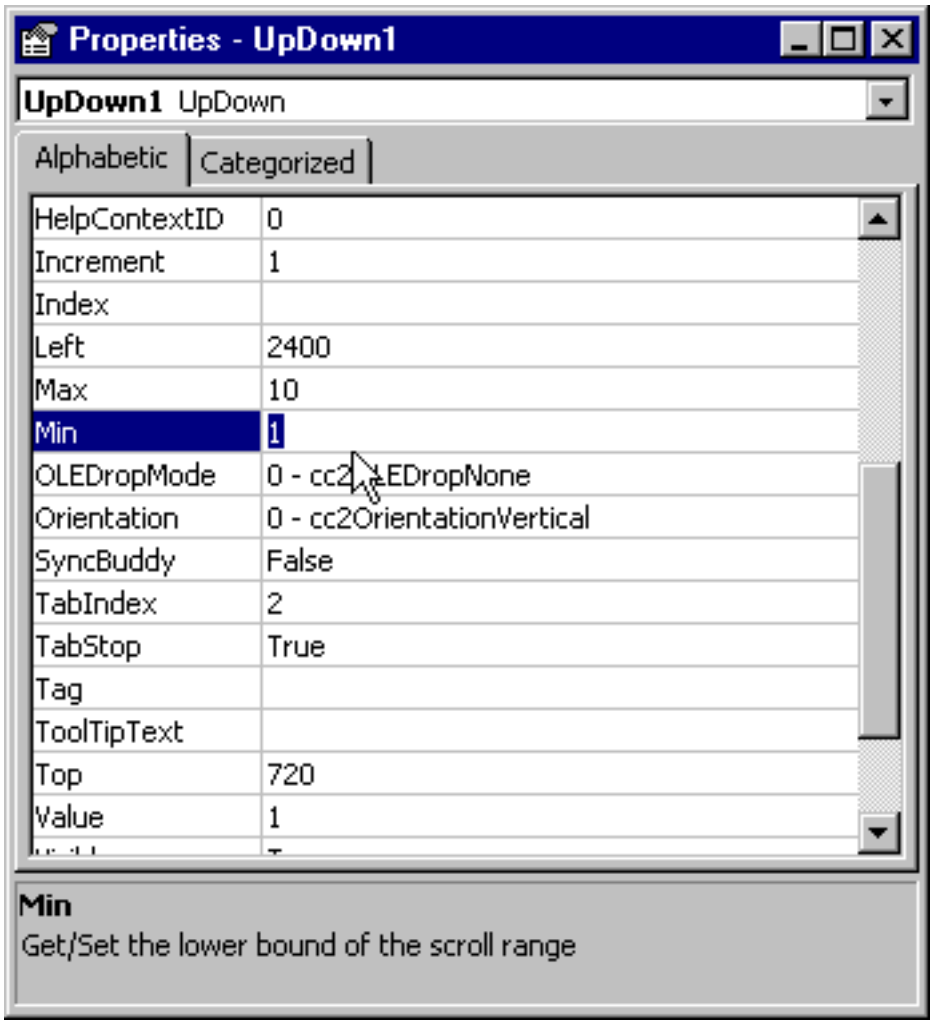
As you can see, it looks very much like the Vertical Scrollbar, but you'll see

that we have a lot less work to do with this control.

Set the Properties

Like the Vertical ScrollBar, the UpDown Control has a Min, Max, and Value Property. By default the Value Property is 0, and when the 'up arrow' on the UpDown Control is clicked, the value ***increases*** by one. When the 'down arrow' on the UpDown Control is clicked, the value is ***decreased*** by one (the opposite of the Vertical ScrollBar.)

As we did with the Vertical Scrollbar, let's specify the Minimum (using the Min Property) and the Maximum (using the Max Property) that we will permit the Value Property to reach. For this example, we'll set Min to 1 and Max to 10…

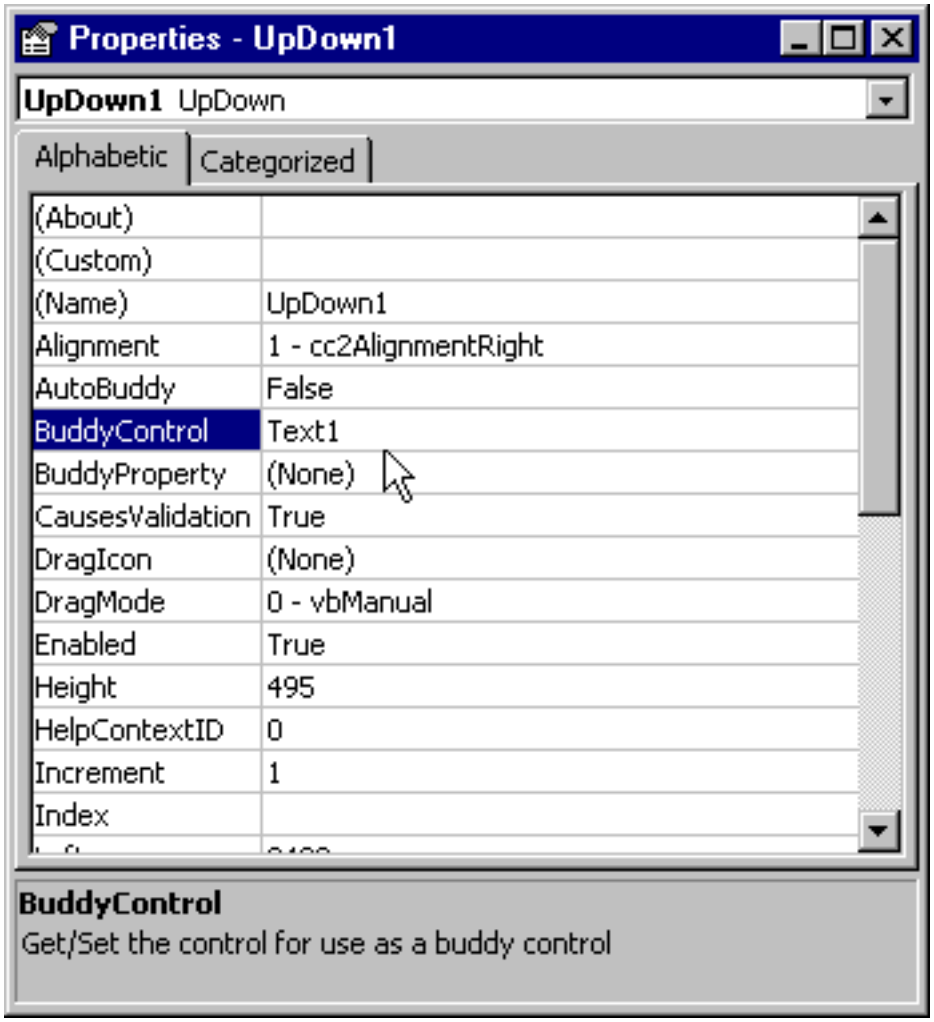


Not surprising, the UpDown Control has a Change Event, and we could write some code so that when it is triggered, the Text Property our Textbox is updated with the Value Property of the UpDown Control. But this is where the UpDown Control can save us some work---by using its Buddy Properties.

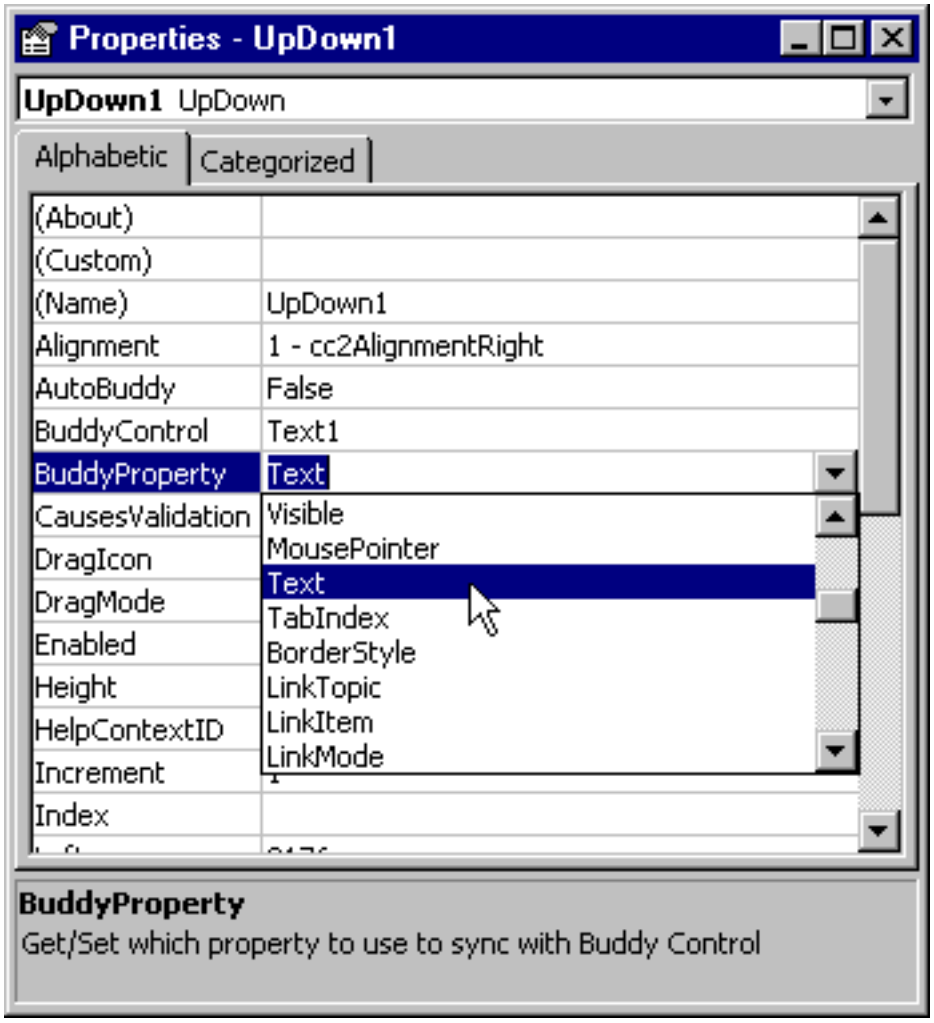
Buddy Properties? That's right. We can tell the UpDown Control to 'buddy up'

with another control on the form, and have it synchronize its Value Property with a Property on that control. To do that, we need to set values in two Properties of the UpDown Control: the BuddyControl and BuddyProperty Properties.

Let's set the BuddyControl Property to the name of our Textbox, Text1. You'll need to type this in…

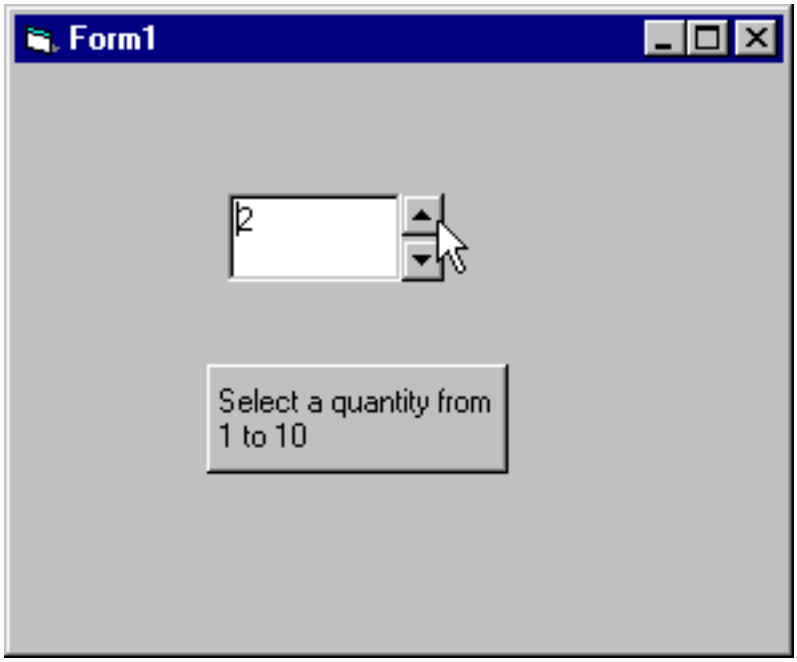


and then set the BuddyProperty Property to Text (the Text property of Text1). This is selectable via a DropDown ListBox, once you set the BuddyControl Property…

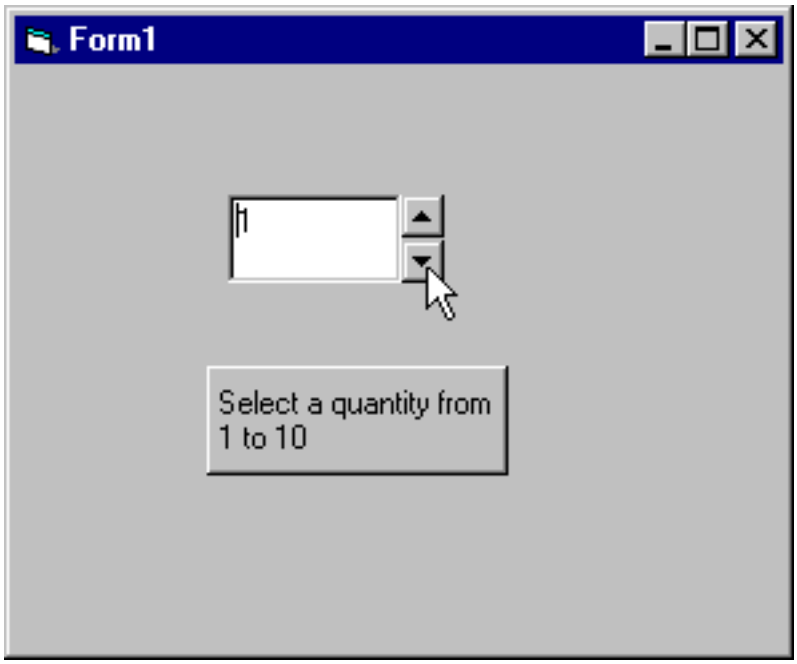


No Code to Write!

Once you've made these selections, if you run the program and click on the up arrow of the UpDown Control the value in the Textbox will automatically be incremented---no code to write…



In the same way, click on the down arrow of the UpDown Control will decrease the value in the Textbox automatically---again, no code to write…



As you can see, by using the UpDown Control instead of the Vertical ScrollBar, we've eliminated the problem with the arrow keys not behaving the way most people believe they should, we no longer need to write code to handle the update of the Textbox, and that annoying flicker of the Vertical Scrollbar is a thing of the past.

Summary

I hope you enjoyed this article on the UpDown Control. In future 'bonus'

articles, I'll try to highlight a Control that you have not know exists.