Visual Basic 5 Tutorial

Many of us have used BASIC in some form or another, perhaps many years ago on a Commodore 64 or Sinclair Spectrum. As you might remember, the computer executed each line of your program in sequence, beginning with the first line and then working downwards towards the bottom of the program.

Programs running in the Windows environment are less predictable; the flow of execution is determined by the user who interacts with your program. For example, the user might make a choice from a menu and then click a button, or perhaps drag a scroll bar to view more of a document. Since it is not possible to predict the order in which the user will choose the various options, the traditional, sequential-style programming is not appropriate. Instead, another method of programming is used, called *event-driven* programming. Unlike other environments for example DOS, where programs are constantly busy, programs under Windows spend most of their time waiting for the user to do something. When the user carries out an action such as clicking a button, the button informs your program via an event. It is then up to your program to carry out the appropriate operation depending on what event it received

Where do these events come from? The answer is, the controls themselves. If a control causes an event to occur, it is said to have *raised* the event. If you click on a control, the control *raises* a click event.

To respond to these events, you write *event handlers*, pieces of code that Visual Basic will execute whenever the event occurs. If you write an event handler for the click event on a button control, the code contained in the handler will be executed every time the user clicks on the button. If any events occur that you didn't write event handlers for, no action is taken and the events go unprocessed.

Right, that's enough theory, let's see this in action. We will create a simple project that will let us convert inches to centimetres. Assuming you have installed VB5CCE, start it by choosing Start->Programs->Visual Basic 5 CCE->Visual Basic 5 CCE. Please note that if you are using the full VB5 retail product, you might see additional information not mentioned here. If this is the case, simply disregard it and continue throughout.

VB will display a dialog asking you whether you want to create an ActiveX Control, or a Standard EXE. Double click on the Standard EXE icon since we want to create an application (ActiveX controls will be covered in further tutorials). Ironically, creating an EXE is one thing VB5CCE *can't* do because you cannot create stand-alone applications using it. Perhaps Microsoft should have worded things differently in the case of VB5CCE.

You are now looking at a project called Project1 that contains a form called Form1. This is the default project that VB provides for us. You can see this hierarchy in the *Project Explorer* on the upper-right hand side of the screen.

Look at the *Properties Window* at the right-hand edge of the screen, below the Project Explorer. This window lists the *properties* (attributes) of the currently-selected object, which in our case, is Form1. The names of each of the properties are listed down the left-hand side of the window, and their corresponding values are shown to the right. Every object in VB has a set of properties associated with it. Some properties are specific to each type of object and so only appear when that particular type of object is selected. Other properties, such as name, are relevant to all objects and are always present in the properties window. You can change the various characteristics of an object by altering its properties.

Let's change the name of our project (Project1) to something more meaningful; click on Project(Project 1) in the Project Explorer. The properties window has changed and is now displaying *(Name)*. The only property of a project you can change is its name, which is why (Name) is the only entry in the properties window. Click on the text *Project1* that appears next to (name), replace it with *Units* and press Enter. Notice how the name of the project in the project explorer has changed to Units, along with the title bar of the window containing Form1 to reflect the new project name. Now, we want to change the name of our form as well. Click on Form1 and observe that the properties window has now changed back to displaying the properties for Form1 since we have just selected it. Click on the text Form1 next to (Name), replace it with *frmUnits* and press enter. We chose the three-letter prefix *frm* to remind us that the object is a form. VB doesn't enforce this naming convention, but it is a very useful habit to have, particularly when dealing with larger projects. Notice that VB has updated the project explorer to show the new name. Our form, frmUnits still has the default title – *Form1*. The title is different from the name, in that the title is just the descriptive text that appears in the form's title bar. The property that controls the descriptive text associated with an object, be it a form or control, is normally called *Caption*. Scroll down the properties window until you can see the Caption property. Replace the default caption, *Form1* with the text *Units Conversion*. Notice how VB immediately updates the form's title as you type so that you can see your changes straight away.

Now it's time to add some controls. Choose the *TextBox* control and then move the mouse cursor over Form1. Notice how the cursor is now a set of cross hairs to indicate that you're about to draw a control. Drag a rectangle out on the form, approximately eight dots wide by three dots high. VB creates a textbox using the dimensions of the rectangle you dragged. If you look in the properties window, you'll notice that the available properties have changed since the textbox is now selected. Now drag the textbox so that it is about half-way across the form, but close to the top. Change the name property of the textbox to *txtInches*. Next, create a similar textbox below the first one and set its name to *txtCM*. The prefix *txt* is used because we are dealing with a textbox. Change the *Text* property of both textboxes to be blank, since we want both textboxes to be initially empty. The Text property holds whatever the user types into the textbox when our program is running.

We should add some descriptions to the textboxes so that the user will know that they represent. Choose the *label* control from the toolbox and create a label to the left of

the txtInches textbox, approximately eleven dots wide by three dots high. Change the label's Caption to *Inches*. If the caption is chopped off you'll need to resize the label so that all of its text is visible. Next, create a similar label beneath the first, next to the txtCM textbox and set its caption to *Centimetres*. If there isn't enough room, you might need to drag the controls around to make room for the new labels.

Let's add the final control, a *CommandButton*, which will cause the actual conversion to occur. Create a CommandButton control to the right of the textboxes, approximately nine dots wide by three dots height. Change its name to *cmdConvert* and its caption to *Convert*. Now, all the controls are in place. Choose the *pointer tool* from the toolbox, click on a blank area of frmUnits to show its handles, and then drag the lower right-hand handle towards the top left-hand corner of the screen, so that the form fits the controls better.

Okay, we're nearly done. Now we just need to add the code that will cause the actual conversion to take place. We will place the code to accomplish this in the *Click* event handler for the button, so that the conversion will take place whenever the Convert button is clicked. Double-click on the button. VB displays the *Code Window* which shows the code in the events behind the controls. There are two combo boxes – the one on the left shows the name of the object we are writing the code for. The one on the right shows which event for that object we're dealing with. VB has correctly assumed that we wanted to create a handler for the click event. Notice that VB has already entered some code for us, *Private Sub cmdConvert_Click()* and *End Sub*. These lines tell VB which control and event combination we're dealing with. Type the following code between those two lines:

```
txtCM.Text = txtInches.Text * 2.52
```

That tells VB to multiply the contents of the txtInches textbox (held in its Text property) by 2.52 and place the result into the Text property of the txtCM textbox. The *dot* seperates the control name from its property. Now we can leave *design mode* and enter *run mode* so that we can try out the new form. Click the *Play* button on VB's toolbar. You can now interact with the form – type a number into the Inches textbox, click the Convert button, and watch the result appear in the Centimetres text box. That wasn't so hard to do, was it? Now close the form by clicking on its close box.

Can you see how the program works? When you click on the Convert CommandButton control, it *raises* a click event. The code that we wrote for the click event then performs the conversion and displays the appropriate result.

The form is far from complete, however. If you type letters into the Inches textbox, the program will fail because you can't multiply using letters of the alphabet! Next month, we'll make this more robust and add extra features using some new controls. Until then, if you want to know what so-and-so does, then *try it!*

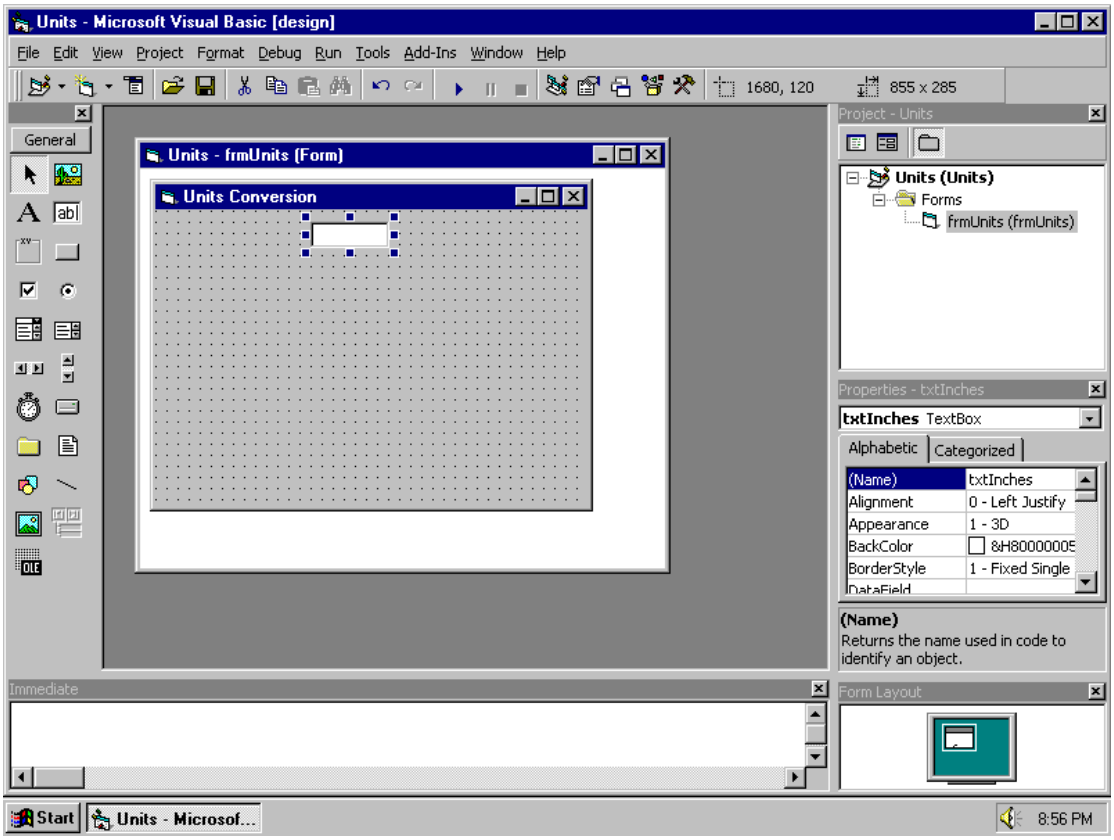
Have fun,

Nick.

Nicholas Scott is a freelance columnist who currently works for MIS Computer Services in Northwich. Nick can be contacted via email at nicks@miscs.com.

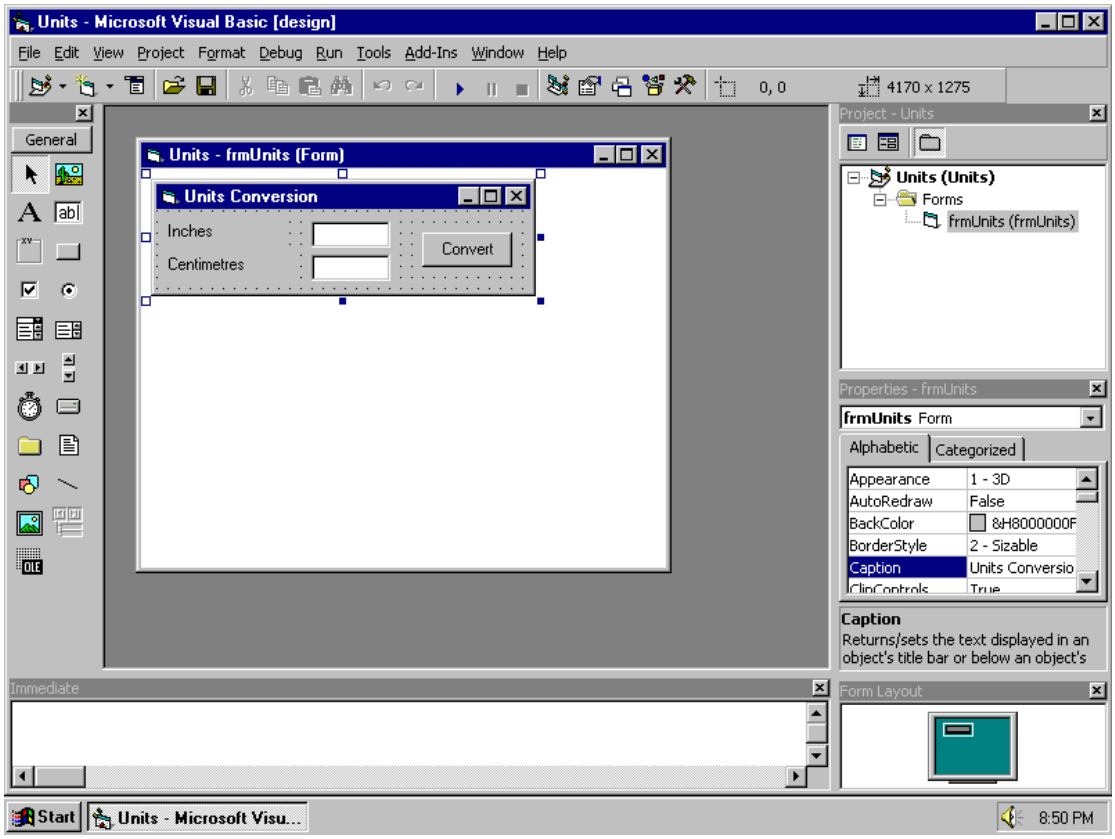
(ED: Please place the following four pictures around the text as you see fit. Please keep them in the order I present them here since they follow on from each other)

(ED: The filename for this .BMP is “Form with just a text box.bmp”)



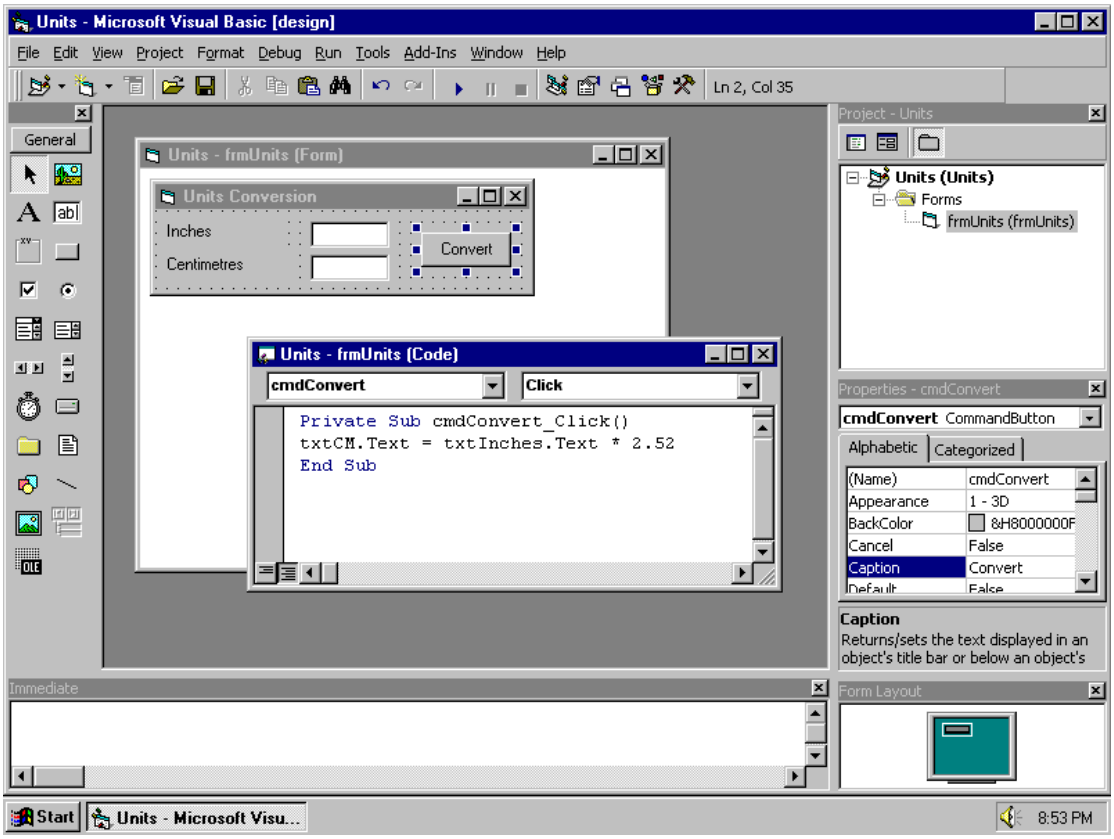
frmUnits with the txtInches textbox added to it

(ED: The filename for this image is “Form with everything on it.bmp”)



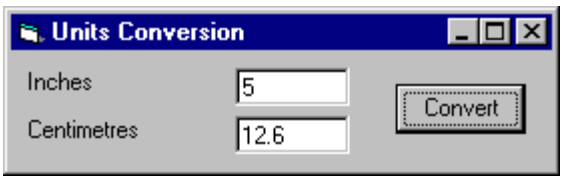
frmInches with all the controls added to it

(ED: The filename for this image is “Code Window.bmp”)



The code which performs the actual conversion, in the Click event of the CommandButton cmdConvert

(ED: The filename for this image is “Runtime.bmp”)



The finished application in action