By Joshua Richardson-Noyes

The Borwell - Software Challenge - Documentation

Main Aim:

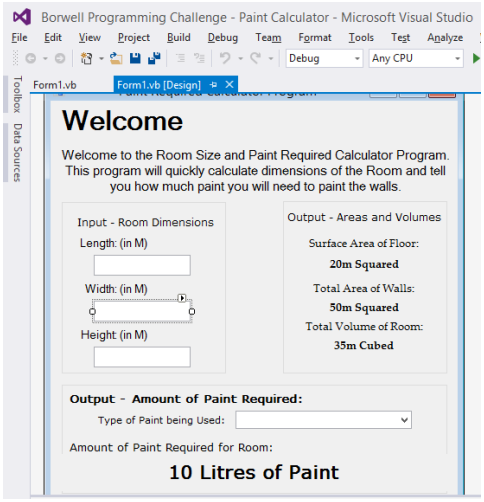
The main aim for my program will be to make it as user friendly as possible. The math to calculate how much paint will be required, alongside wall and floor areas, may prove quite difficult to an end user and so I wanted to make my program as simple to use as I could.

Choosing the language: Both Visual Basic and Python are easy enough languages to use, however I chose Visual Basic as it provides me a much easier way of making a Graphical User Interface for the user to see and use quickly and easily. Though a similar program could be coded up in Python it will require 3rd party modules that the end-user may not have installed. Furthermore, a Visual Basic program can be quickly compiled into an executable format that the end-user can simply run without any other requirements.

Part 1 – The Designer:

Before writing in the more complicated code side, it is first good practise to start by using the Designer tool in Visual Basic. The designer tool allows me to create the form that the data will later be both inputted and outputted on/to. I started off by using the Label Object. The label object is a very simple text property that cannot be changed by the user. I will use the label to display a simple welcome message and a second one to display a few helpful instructions to briefly explain to the user what they must using the program.

I quickly resized the form, to get a bit more space for all the controls and wrote in the text for the 2 labels. I centred the text so that it looked a little more professional for the instructions. I then proceeded to add in 3 Group box Controls. These meant I could group together different elements on the form as well as draw a small outline around these groups to better illustrate the different parts of the program as shown below…



As you can see here I have added 3 Text box controls, to get the dimensions in metres of the room the user will enter. Then to the left of the Windows Form, I have used a series of labels to output the dimensions of the room. By default I set each of the labels to random values just to get an idea of the program, though once the program is actually run, the program will immediately clear those label fields and, providing all measurements have been entered, will calculate each of the fields and output them to the screen. Finally, I added the bottom section of the program, this gives the user a dropdown style selection of a series of different paints they may choose to use. Once set, the program will then calculate how much paint, in litres, is required for the walls and output to the label. Finally, though invisible to the user, I have used a Timer Control. The timer control will enable me to write the program code to update each of the values at a constant rate, this will mean that each of the output values are generated seamlessly to the user, without them having to press a submit button.

By Joshua Richardson-Noyes

Part 2 – Code

As you can see I have already renamed each of the Label controls, as it makes it much easier to code

By Default, visual studio will normally name each control used after the control followed by an incrementing number. E.g. Textbox1, Textbox2. This can become very confusing when writing the code, as you won't be able to tell them apart.

For the first few lines of code, I have quickly defined all the variable that I will be using later on in the program for the mathematic equations.

```
' Quickly Defining a series of variables to store all the different types of data
Public RoomLength As Decimal
Public RoomWidth As Decimal
Public RoomHeight As Decimal

Public WallsX As Decimal
Public WallsY As Decimal

Public FloorArea As Decimal
Public TotalWallArea As Decimal
Public RoomVolume As Decimal

Public PaintRequired As Decimal
```

The second piece of code I wrote for my program was adding in some code to clear up the labels text (as the text in there was only sample text whilst in designer mode) as soon as the form loads up. This is where I call the function Form1.Load and have it assign each of the output labels a text property that is empty e.g. (“”)

```
16
17      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
18
19          ' To start with, I clear the values of each of the labels I am using for output to the user.
20          SurfaceAreaOutputLabel.Text = ""
21          WallAreaOutputLabel.Text = ""
22          TotalVolumeOutputLabel.Text = ""
23          ActualPaintOutputLabel.Text = ""
24      End Sub
```

I wanted to have the program immediately generate each of the output values (e.g. floor area, volume, wall area) as soon as the user had entered their input values so I used a “Timer” tool/control.

I didn’t bother to rename this control as it was the only Timer I would be using in the application.

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    ...
    If LengthTextbox.Text.Length > 0 And WidthTextbox.Text.Length > 0 And HeightTextbox.Text.Length > 0 Then
        ...
        If IsNumeric(LengthTextbox.Text) And IsNumeric(WidthTextbox.Text) And IsNumeric(HeightTextbox.Text) Then
            ...
            InstructionsLabel.Text = ("Welcome to the Room Size and Paint Required Calculator Program.
This program will quickly calculate dimensions of the Room and tell
you how much paint you will need to paint the walls.")
            InstructionsLabel.ForeColor = Color.Black

            ...
            RoomWidth = CDec(WidthTextbox.Text)
            RoomHeight = CDec(HeightTextbox.Text)
            RoomLength = CDec(LengthTextbox.Text)

            ...
            FloorArea = (RoomWidth * RoomLength)
            WallsX = (RoomWidth * RoomHeight) * 2   ' Multiplied by 2 as there will be 2 matching walls in the room.
            WallsY = (RoomLength * RoomHeight) * 2

            ...
            TotalWallArea = WallsX + WallsY
            ...
            RoomVolume = (RoomWidth * RoomLength * RoomHeight)

            ...
            SurfaceAreaOutputLabel.Text = FloorArea.ToString() + " Metres Squared"
            WallAreaOutputLabel.Text = TotalWallArea.ToString() + " Metres Squared"
            TotalVolumeOutputLabel.Text = RoomVolume.ToString() + " Metres Cubed"


            ...
            If PaintTypeDropdownBox.Text = "Dulux Matt Emulsion Paint ( 1 Litres / 13 Metres Squared )" Then
                ...
                PaintRequired = TotalWallArea / 13
                ...
            ElseIf PaintTypeDropdownBox.Text = "Generic Paint Type ( 1 Litres / 11 Metres Squared )" Then
                ...
                PaintRequired = TotalWallArea / 11
            End If
            ...
            ActualPaintOutputLabel.Text = Math.Round(PaintRequired, 1).ToString() + " Litres of Paint"
        Else
            InstructionsLabel.Text = "Invalid Input!
This error means you have accidentally entered a non-numeric value into the size fields."
            InstructionsLabel.ForeColor = Color.Red
        End If
    End If

End Sub
End Class
```

Next for the much longer code:

The code is explained in more detail through the comments in the code. One of the issues I faced when Debugging the program, was that if I entered any numbers into the textbox controls, the program would crash straight away, as the text cannot be divided/added/subtracted from by numbers, it would simply not make any sense to us or the computer. As a result I put all of this code, performing the math calculations, under 2 if statements. As explained before, the first IF statement checks to make sure that the textboxes actually have any text in them. The second, checks to make sure that the text entered in, is definitely numbers and does not contain any characters. With these 2 IF statements, we can stop the program from performing any mathematical calculations on the data until it is sure that the data that was entered is valid. (Numerical/Decimal only)

Another problem that occurred during testing, is that I quickly realised through several test values, e.g. 2.0 2.5, 3.2, the larger values was outputting smaller amounts of paint. Which of course wouldn't make sense as surely, the larger the wall, the more paint there is required. After looking through my code, I realised a problem with one of the mathematical calculations:

```
If PaintTypeDropdownBox.Text = "Dulux Matt Emulsion Paint ( 1 Litres / 13 Metres Squared )" Then
    ...
    PaintRequired = 13 / TotalWallArea
    ...
ElseIf PaintTypeDropdownBox.Text = "Generic Paint Type ( 1 Litres / 11 Metres Squared )" Then
    ...
    PaintRequired = 11 / TotalWallArea
End If
```

I quickly realised that I was dividing 13, (the number of square metres 1 litre will cover) by the total surface area of all the walls. Whilst both values are correct, to correctly calculate this the 2 values should be the other way around in this equation. It should be:

```
If PaintTypeDropdownBox.Text = "Dulux Matt Emulsion Paint ( 1 Litres / 13 Metres Squared )" Then
    ...
    PaintRequired = TotalWallArea / 13
    ...
ElseIf PaintTypeDropdownBox.Text = "Generic Paint Type ( 1 Litres / 11 Metres Squared )" Then
    ...
    PaintRequired = TotalWallArea / 11
End If
```

With this correction in place, I tested the values again, and this time the program calculated more paint required, the larger the numbers.

Overall, I think I have managed to create a working visual basic program that I am hoping will prove to be user friendly even to someone who doesn't know much about computers. If I was to write this program again, I think I would make the program even better by adding the ability to create custom paint types. This way, paint types that have different surface areas covered per 1 litre could be quickly and easily entered by another user. Another quick improvement to the code could be the ability to specify window dimensions to be subtracted from the surface area as these will not require painting.

DISCLAIMER: To keep the screenshots more condensed, I chose to hide the comments around the code in the screenshots. To see the comments please see the source code. It can be opened with most versions of Microsoft Visual Studio.

Controls Used in this Project:

Textbox Control – A control that can be added to a "Windows Form" in Visual Studio and is used for text input in the computer.

Timer Control – A control in Visual Studio used to repeatedly perform a set of code in regular intervals.

Label Control – Another control in Visual Studio, this one cannot be edited by the user and is generally used for instructions to the user and easy to read outputs.