

## Variables

A variable is a named location in memory where you can store values. In the previous lesson you wrote a program to display a random color every time you clicked, but you did not know what the color was. Variables would let us assign a random value to a variable named Red for example and then use the stored value to change the color. Because the value of a variable is stored in memory, we can also display the value in the text.

**Variable Names** must start with a letter of the alphabet: after the first letter, it can have more letters, digits, and underscores. Blanks are not allowed. Two words can be separated by an underscore, or the second word can begin with a capital letter to make it readable:

HoursWorked or Hourly\_rate, for example. You cannot use *keywords* such as end, if or sub that have a special meaning in Visual Basic.

**Type:** Each variable also has a type. There are hundred of types, but we usually use one of the following:

**Integer:** whole numbers with no decimal places;

**Double:** numbers with decimal places;

**String:** for names, words, etc.

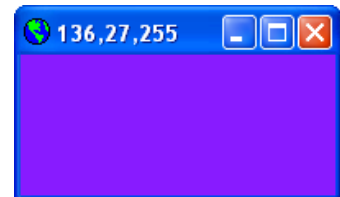
**Boolean:** for variables that can be either True or False.

Close any open project and start a new windows application called Colors.

Write code for the click event as shown below: Red, Green and Blue are declared as whole numbers.

```
Protected Overrides Sub OnClick(ByVal e As System.EventArgs)
    Dim Red, Green, Blue As Integer
    Red = Rnd() * 255
    Green = Rnd() * 255
    Blue = Rnd() * 255
    Me.BackColor = Color.FromArgb(255, Red, Green, Blue)
    Me.Text = Red & ", " & Green & ", " & Blue
End Sub 'OnClick
```

If you run the program several times, you will notice that it always gives you the same sequence of colors. This can be very useful: if something goes wrong, it makes it easy to repeat the condition that caused the error. It can also make it easier to win at a game of dice, but it isn't very interesting.



To reset the random numbers, add the following code: (*You type only randomize(!)*)

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles MyBase.Load
    Randomize()
End Sub 'Form1_Load
```

## Assigning Values to a Variable:

A variable can be assigned a value with the assignment statement.

The format is: **variable = newvalue**

### Examples:

The new value is a constant:  $x = 5$

The new value is another variable:  $x = y$

The new value is an expression:  $x = y/3$

It can even use its current value in the expression:  $x = x + 1$

*The last statement assigns a new value to x that is one more than its current value!*

## Counter

Save the Colors project, and then start a new windows application called Counter.

Declare a variable called Number in the location shown below:

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim Number As Integer
```

Declaring the variable here makes it a global variable. It will be known, and can be used in any procedure.

Add the code to add 1 to Number each time you click and display the value in the text:

```
Protected Overrides Sub OnClick(ByVal e As System.EventArgs)
    Number = Number + 1
    Me.Text = Number
End Sub 'OnClick
```

Run the program and click, click, click. Each time you click one is added to Number and the new value is displayed in the text. If the variable Number is declared inside OnClick, (made local to OnClick) it is created each time OnClick is invoked. When OnClick ends, the variable is destroyed. Clicking would show a value of 1 every time, because an integer is given a default of zero. Adding 1 will always result in one. Try it!

**Experiment:** Reset number to 0 when you double click.

**Experiment:** Add 0.5 to Number when you click. What happens?

Because Number is an integer, it is given an initial value of 0. Add  $0+0.5$  and you get 0.5, but when that value is assigned to an integer the decimal portion is dropped. Click all you want the value will stay 0! Change the type to **double** and try it again. (Erase Integer and backspace to type the space after **As** over again, this will force the pop-up window to appear.)

**Experiment:** Modify the code so that the numbers that appear are 5, 10, 15, 20, 25, etc.

**Experiment:** Modify the code so that number is multiplied by 2 each time. Nothing will appear because Number has an initial value of 0. Add code to OnLoad so that number is given a value of 1 at start. What numbers appear now? Reverse the two statements in OnClick so that the value of number is displayed before it is multiplied by two. What numbers do you see now?

## Objects

Save the project and start a new project. Call it Shapes. We would like to draw a circle every time the user clicks; right at the point they click.

The OnClick event doesn't tell us where the click took place. Instead of writing the code for OnClick, we will write the code for OnMouseDown because it does give us the X and Y values.

```
Protected Overrides Sub OnMouseDown(ByVal e As _
    System.Windows.Forms.MouseEventArgs)
    Dim MyPen As New Pen(Color.Black, 3) 'Black, width of 3
    Me.CreateGraphics.DrawEllipse(MyPen, e.X, e.Y, 10, 10)
End Sub 'OnMouseDown
```

The types integer, double, and char are called primitive types. They hold a single value. Variables declared as primitive types can be given values and used in expressions but they can't *do* anything.

Pen is a **class**. A class has methods and properties. (Form is also a class.) We declare MyPen to be an **instance** of the Pen class. MyPen is called an **object**. An object has all of the methods and properties of the class. We need a pen or brush in order to draw things.

Notice that the declaration of MyPen uses the word New. New must be used when you declare an **instance** of a class. The declaration of a pen allows you to assign values to the properties Color and Width. We have selected Black for the initial value of the color and 3 as the width. We can change these properties later if we want.

A form is also an object. Forms have lots of methods. After typing "me dot" we can select CreateGraphics, this creates a graphics object. Graphics object also have lots of methods, we have chosen to draw an ellipse.

The first set of arguments for DrawEllipse requires a rectangle: rectangle is another object.

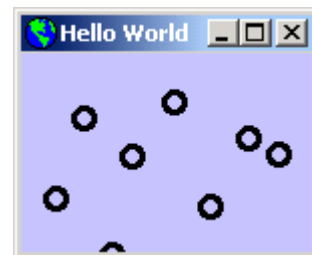
```
▲ 1 of 4 ▼ DrawEllipse (pen As System.Drawing.Pen, rect As System.Drawing.RectangleF)
pen: System.Drawing.Pen object that determines the color, width, and style of the ellipse.
```

Notice that the top left corner of the help window says 1 of 4. Click the down arrow on the side to select the second method. The first argument is the Pen you will use, then X and Y, then width and height.

```
Me.CreateGraphics.DrawEllipse (|
▲ 2 of 4 ▼ DrawEllipse (pen As System.Drawing.Pen, x As Single, y As Single, width As Single, height As Single)
pen: System.Drawing.Pen object that determines the color, width, and style of the ellipse.
```

We have specified that the top left corner of the ellipse will be at e.X and e.Y with a width of 10 and a height of 10. Because the width and height are the same we will see a circle everyplace we click.

**Experiment:** The top left corner of the circle is at the point you clicked. How would you make the center of the circle at the point you clicked?



If you would like to create a solid circle, you will need a brush:

```
me.CreateGraphics.FillEllipse (
```

▲ 2 of 4 ▼ FillEllipse (**brush As System.Drawing.Brush**, x As Single, y As Single, width As Single, height As Single)  
**brush:** System.Drawing.Brush object that determines the characteristics of the fill.

A brush is similar to a pen except it does not have a width. Change the code as shown below:


```
Protected Overrides Sub OnMouseDown(ByVal e As _
    System.Windows.Forms.MouseEventArgs)
    Dim MyBrush As New SolidBrush(Color.Red)
    Me.CreateGraphics.FillEllipse(MyBrush, e.X-5, e.Y-5, 10, 10)
End Sub 'OnMouseDown
```

This time when you run the program and click the center of the circle is at the point you click and the circle is solid red.

**Experiment:** Modify the code to create blue circles with a radius of 20.

The problem above is easier if you use variables:

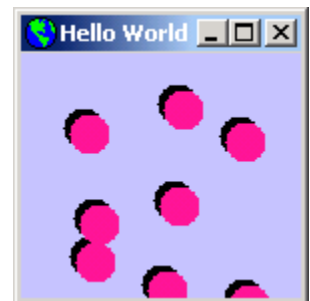
```
Protected Overrides Sub OnMouseDown(ByVal e As _
    System.Windows.Forms.MouseEventArgs)
    Dim Radius As Integer = 10
    Dim X As Integer = e.X - Radius
    Dim Y As Integer = e.Y - Radius
    Dim MyBrush As New SolidBrush(Color.Red)
    Me.CreateGraphics.FillEllipse(MyBrush,X,Y,Radius*2, Radius*2)
End Sub 'OnMouseDown
```

**Experiment:** Draw a solid circle with the brush, then the outline of the same circle with the pen to get a red circle with a black outline, . What happens if you draw the outline first, then the solid circle?

## Rectangles

Some of the graphics methods use rectangles. A rectangle is another class. When you declare an instance of the rectangle class, you give X, Y, width and height as the arguments. The code below shows how to change the properties of an object after it is declared:

```
Protected Overrides Sub OnMouseDown(ByVal e As _
    System.Windows.Forms.MouseEventArgs)
    Dim MyBrush As New SolidBrush(Color.Black)
    Dim MyRectangle As New Rectangle(e.X, e.Y, 20, 20)
    Me.CreateGraphics.FillEllipse(MyBrush, MyRectangle)
    MyRectangle.X = MyRectangle.X + 3
    MyRectangle.Y = MyRectangle.Y + 3
    MyBrush.Color = Color.DeepPink
    Me.CreateGraphics.FillEllipse(MyBrush, MyRectangle)
End Sub 'OnMouseDown
```



**Experiment:** Modify to draw a square. Change the position of the shadow.

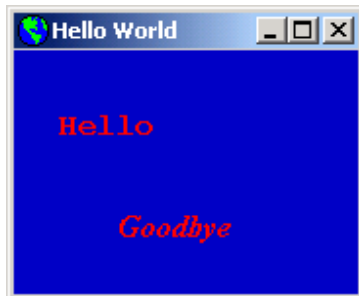
## Drawing Strings: Font Class

Another class that can be useful is the Font. You can set a font two ways:

- Set the font property of the form at design time. This lets you make choices from the common dialog for selecting font, size, and style.
- Declare an instance of the Font class in the code.

The example below uses both methods. The last two arguments give the X and Y position for the top left corner of the text area. Setting X=0 and Y=0 would print in the top left corner of the form.

```
Protected Overrides Sub OnPaint(ByVal e As System.Windows.Forms.PaintEventArgs)
    Dim MyBrush As New SolidBrush(Color.Red)
    Dim MyFont As New Font("Courier New", 12, FontStyle.Bold)
    e.Graphics.DrawString("Hello", MyFont, MyBrush, 20, 30)
    e.Graphics.DrawString("Goodbye", Me.Font, MyBrush, 50, 80)
End Sub 'OnPaint
```



## More Code, More Fun

We are going to combine several of the techniques we have learned to create a larger program. Every time we click we will generate a random color and then draw a circle at the point we clicked. When that works, we will add a counter and print the number of the click on top of the circle. To make sure we can see the text we will create a second color that is different from the first color.

Since this combines several steps we will write them one at a time. Close everything and start a new windows application. Name it dots.

Declare a global variable number: `Dim Number As Integer`

It is made global by declaring it outside of any method or procedure.

Under Inherits `System.Windows.Forms.Form` is a good place.

Next we will increment number (add one) on mouse down. Display the new value in the text so that you can test.

```
Protected Overrides Sub OnMouseDown(ByVal e As System.Windows.Forms.MouseEventArgs)
    Number = Number + 1
    Me.Text = Number
End Sub 'OnMouseDown
```



Declare an instance of the font class inside OnMouseDown, and a brush.

Draw the number on the form instead of as the text.

```
Protected Overrides Sub OnMouseDown(ByVal e As System.Windows.Forms.MouseEventArgs)
    Dim MyFont As New Font("Courier New", 12, FontStyle.Bold)
    Dim MyBrush As New SolidBrush(Color.Red)
    Number = Number + 1
    Me.CreateGraphics.DrawString(Number, MyFont, MyBrush, e.X, e.Y)
End Sub 'OnMouseDown
```

Run the program. The results should be like this.

**SAVE!! It only takes one click and it's worth it!**



Modify the code so that the color is random:

```
Protected Overrides Sub OnMouseDown(ByVal e As System.Windows.Forms.MouseEventArgs)
    Dim MyFont As New Font("Courier New", 12, FontStyle.Bold)
    Dim MyBrush As New SolidBrush(Color.Red)
    Dim Red, Green, Blue As Integer
    Red = Rnd() * 255
    Green = Rnd() * 255
    Blue = Rnd() * 255
    MyBrush.Color = Color.FromArgb(255, Red, Green, Blue)
    Number = Number + 1
    Me.CreateGraphics.DrawString(Number, MyFont, MyBrush, e.X, e.Y)
End Sub 'OnMouseDown
```

Draw the circle first, then change the color before you print the number. Each of the colors, red, green and blue must be between 0 and 255. If you subtract the current value from 255 you will get a new color that is also between 0 and 255. Unless each of the values was close to 127 this will produce a different color.

```
Protected Overrides Sub OnMouseDown(ByVal e As System.Windows.Forms.MouseEventArgs)
    Dim MyFont As New Font("Courier New", 12, FontStyle.Bold)
    Dim MyBrush As New SolidBrush(Color.Red)
    Dim Red, Green, Blue As Integer
    Red = Rnd() * 255
    Green = Rnd() * 255
    Blue = Rnd() * 255
    MyBrush.Color = Color.FromArgb(255, Red, Green, Blue)
    Me.CreateGraphics.FillEllipse(MyBrush, e.X, e.Y, 20, 20)
    MyBrush.Color = Color.FromArgb(255, 255-Red, 255-Green, 255-Blue)
    Number = Number + 1
    Me.CreateGraphics.DrawString(Number, MyFont, MyBrush, e.X, e.Y)
End Sub 'OnMouseDown
```

Make sure you save, then close the project and start a new one. Try to recreate the program without looking at your notes!