# Chapter 20

# Creating and Managing Graphics

❖ Learning how to create and manage graphics

## 20.1 Introduction

Creating and managing graphics is easy in earlier versions of Visual Basic as they have built-in drawing tools. For example, In Visual Basic 6, the drawing tools are included in the toolbox where the programmer just need to drag the shape controls into the form to create rectangle, square, ellipse, circle and more. However, its simplicity has the shortcomings; you do not have many choices in creating customized drawings.

Since Visual Basic evolved into a fully OOP language under the VB.net framework, shape controls are no longer available. Now the programmer needs to write code to create various shapes and drawings. Even though the learning curve is steeper, the programmer can write powerful code to create all kinds of graphics. You can even design your own controls

VB2010 offers various graphics capabilities that enable programmers to write code that can draw all kinds of shapes and even fonts. In this Chapter, you will learn how to write code to draw lines and shapes on the VB interface.

## 20.2 Creating the Graphics Object

Before you can draw anything on a form, you need to create the Graphics object in vb2008. A graphics object is created using a CreateGraphics() method. You can create a graphics object that draw to the form itself or a control. For example, if you wish to draw to the form, you can use the following statement:

```
Dim myGraphics As Graphics =me.CreateGraphics
```

.

If you want the graphics object to draw to a picturebox, you can write the following statement:

```
Dim myGraphics As Graphics = PictureBox1.CreateGraphics
```

You can also use the textbox as a drawing surface, the statement is:

```
Dim myGraphics As Graphics = TextBox1.CreateGraphics
```

The Graphics object that is created does not draw anything on the screen until you call the methods of the Graphics object. In addition, you need to create the **Pen** object as the drawing tool. We will examine the code that can create a pen in the following section.

## 20.3 Creating the Pen object

The **Pen** object can be created using the following code:

```
myPen = New Pen(Brushes.DarkMagenta, 10)
```

In the code, myPen is a Pen variable. You can use any variable name instead of myPen. The first argument of the pen object defines the color of the drawing line and the second argument defines the width of the drawing line.

You can also create a Pen using the following statement:

```
Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)
```

Where the first argument defines the color (*here is blue, you can change that to red or whatever color you want*) and the second argument defines the width of the drawing line.

Having created the Graphics and the Pen object, you are now ready to draw graphics on the screen, which we will show you in the following section.

**20.4 Drawing a Line**

In this section, we will show you how to draw a straight line on the Form. First, launch Visual basic 2008 Express. In the startup page, drag a button into the form. Double click on the button and key in the following code.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Dim myGraphics As Graphics = me.CreateGraphics

        Dim myPen As Pen

        myPen = New Pen(Brushes.DarkMagenta, 10)

        myGraphics.DrawLine(myPen, 10, 10, 100, 10)

End Sub
```

The second line created the Graphics object and the third and fourth line create the Pen object. The fifth line draws a line on the Form using the DrawLine method. The first argument uses the Pen object created by you, the second argument and the third arguments define the coordinate of the starting point of the line, the fourth and the last arguments define the ending coordinate of the line. The general syntax to draw line is

`object.DrawLine(Pen, x1, y1, x2, y2)`

Run the program and you can a see purple line appear on the screen, as shown in Figure 20.1.

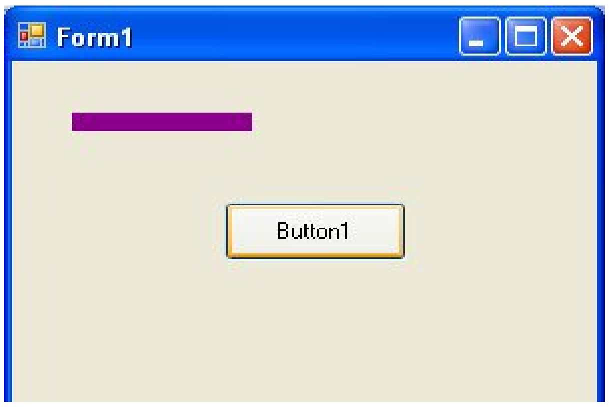

**Figure 20.1**

## 20.5 Creating a Rectangle

To draw a rectangle on the screen in VB2010, there are two ways:

(i)The first way is to draw a rectangle directly using the **DrawRectangle** method by specifying its upper-left corner's coordinates and its width and height. You also need to create a Graphics and a Pen object to handle the actual drawing. The method to draw the rectangle is **DrawRectangle**.

The syntax is:

```
myGrapphics.DrawRectangle (myPen, X, Y, width, height)
```

Where myGraphics is the variable name of the Graphics object and myPen is the variable name of the Pen object created by you. You can use any valid and meaningful variable names. X, Y is the coordinate of the upper-left corner of the rectangle while width and height are self-explanatory, i.e., the width and height of the rectangle.

The sample code is shown below:

```
Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawRectangle (myPen, 0, 0, 100, 50)
```

(ii) The second way is to create a rectangle object first and then draw this triangle using the **DrawRectangle** method. The syntax is as shown below:

```
myGraphics.DrawRectangle (myPen, myRectangle)
```

Where **myRectangle** is the rectangle object created by you, the user.

The code to create a rectangle object is as shown below:

```
        Dim myRectangle As New Rectangle

        myRect.X = 10

        myRect.Y = 10

        myRect.Width = 100

        myRect.Height = 50
```

You can also create a rectangle object using a one-line code as follows:

```
        Dim myRectangle As New Rectangle(X, Y, width, height)
```

The code to draw the above rectangle is

```
        myGraphics.DrawRectangle (myPen, myRectangle)
```

The sample code is shown below:

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click

        Dim myRect As New Rectangle

        myRect.X = 10

        myRect.Y = 10

        myRect.Width = 100

        myRect.Height = 50

        Dim myPen As Pen

        myPen = New Pen(Drawing.Color.Blue, 5)

        Dim myGraphics As Graphics = Me.CreateGraphics

        myGraphics.DrawRectangle(myPen, myRect)

    End Sub
```

## 20.6 Customizing Line Style of the Pen Object

The shapes we draw so far were drawn with solid line, we can customize the line style of the Pen object so that we have dotted line, line consisting of dashes and more. For example, the syntax to draw with dotted line is shown below:

```
myPen.DashStyle=Drawing.Drawing2D.DashStyle.Dot
```

The last argument, Dot**,** specifies a particular line DashStyle value, a line that makes up of dots. The following code draws a rectangle with red dotted line.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Dim myPen As Pen

        myPen = New Pen(Drawing.Color.Red, 5)

        Dim myGraphics As Graphics = Me.CreateGraphics

        myPen.DashStyle = Drawing.Drawing2D.DashStyle.Dot

        myGraphics.DrawRectangle(myPen, 10, 10, 100, 50)

End Sub
```

Run the program and you can see a dotted-line rectangle appears on the screen, as shown in Figure 20.2.

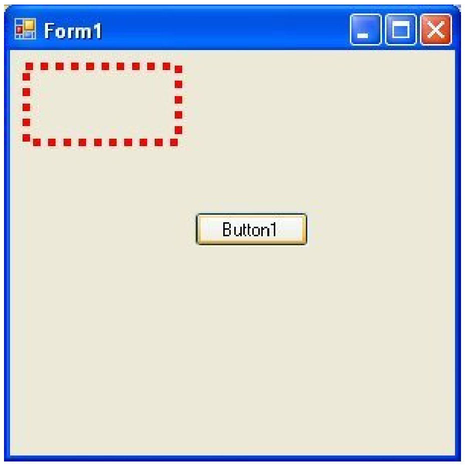

**Figure 20.2**

## 20.7 Drawing an Ellipse

First, we need to understand the principal behind drawing an ellipse. The basic structure of any shape is a rectangle. Ellipse is an oval shape that is bounded by a rectangle, as shown in Figure 20.3 below:

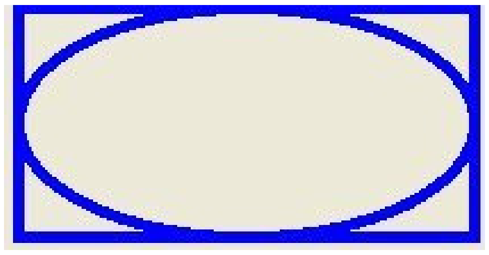

**Figure 20.3**

Therefore, you need to create a Rectangle object before you can draw an ellipse. This rectangle serves as a bounding rectangle for the ellipse. On the other hand, you can still draw an ellipse with the **DrawEllipse** method without first creating a rectangle. We will show you both ways.

In the first method, let say you have created a rectangle object known as myRectangle and a pen object as myPen, then you can draw an ellipse using the following statement:

```
myGraphics.DrawEllipse (myPen, myRectangle)
```

* Assume you have also already created the Graphics object myGraphics.

The following is an example of the full code.

```
Dim myPen As Pen
myPen = New Pen(Drawing.Color.Blue, 5)
Dim myGraphics As Graphics = Me.CreateGraphics
Dim myRectangle As New Rectangle
myRectangle.X = 10
myRectangle.Y = 10
myRectangle.Width = 200
myRectangle.Height = 100
myGraphics.DrawEllipse (myPen, myRectangle)
```

Run the program and you see the ellipse appears on the screen, as shown in Figure 20.4.

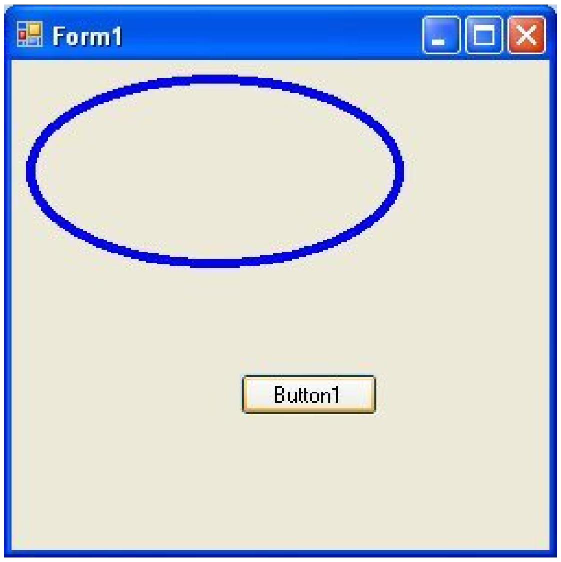

**Figure 20.4**

The second method is using the DrawEllipse method without creating a rectangle object. Off course, you still have to create the Graphics and the Pen objects. The syntax is:

```
myGraphics.DrawEllipse(myPen, X,Y, Width, Height)
```

Where (X, Y) are the coordinates of the upper left corner of the bounding rectangle, width is the width of the ellipse and height is the height of the ellipse**.**

The following is an example of the full code:

```
Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawEllipse (myPen, 10, 10, 200, 100)
```

### 20.8 Drawing a Circle

After you have learned how to draw an ellipse, drawing a circle becomes very simple. We use exactly the same methods used in the preceding section but modify the width and height so that they are of the same values.

The following examples draw the same circle.

**Example (a)**

```
Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

Dim myRectangle As New Rectangle

myRectangle.X = 10

myRectangle.Y = 10

myRectangle.Width = 100

myRectangle.Height = 100

myGraphics.DrawEllipse(myPen, myRectangle)
```

**Example (b)**

```
Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawEllipse(myPen, 10, 10, 100, 100)
```

Run the program and you can see a circle appears on the screen, as shown in Figure 20.5

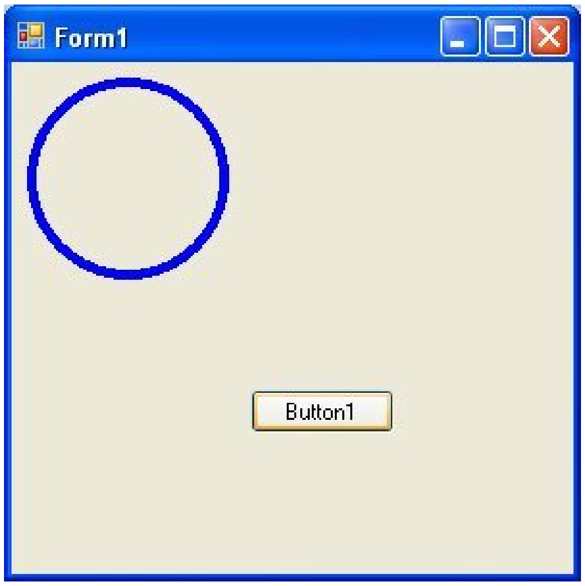

**Figure 20.5**

## 20.9 Drawing Text

In order to draw text on the screen, we can use the DrawString method. The format is as follows:

```
myGraphics.DrawString (myText, myFont, mybrush, X , Y)
```

Where myGraphics is the Graphics object, myText is the text you wish to display on the screen, myFont is the font object created by you, myBrush is the brush style created by you and X, Y are the coordinates of upper left corner of the Text.

You can create your **Font object** using the following statement:

```
myFont = New System.Drawing.Font("Verdana", 20)
```

Where the first argument of the font is the font typeface and the second argument is the font size. You can add a third argument as font style, either bold, italic, underline.

Here are some examples**:**

```
myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Bold)
myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Underline)
```

```
        myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Italic)
        myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Regular)
```

To create the **Brush** object, you can use the following statement:

```
        Dim myBrush As Brush
        myBrush = New Drawing.SolidBrush(Color.BrushColor)
```

Besides the seven colors, some of the common Brush Colors are AliceBlue, AquaMarine Beige, DarkMagenta, DrarkOliveGreen, SkyBlue and more. You do not have to remember the names of all the colors, the intelliSense will let you browse through the colors in a drop-down menu once you type the dot after the word Color.

Now we shall proceed to draw the font using the sample code below:

```
        Dim myGraphics As Graphics = Me.CreateGraphics
        Dim myFont As Font
        Dim myBrush As Brush
        myBrush = New Drawing.SolidBrush(Color.DarkOrchid)
        myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Underline)
        myGraphics.DrawString("Visual Basic 2010", myFont, myBrush, 10, 10)
```

Run the program above and you can see the text “Visual Basic 2010 “appears on the screen, as shown in Figure 20.6.

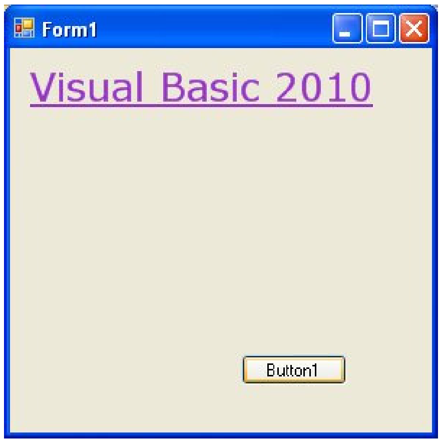

**Figure 20.6**

You can modify the preceding code if you do not want to create the Font and the Brush objects. You can use the font of an existing object such as the Form and the System Colors. Replace the last line in the preceding example with this line.

```
myGraphics.DrawString("Visual Basic 2010", me.Font,
System.Drawing.Brushes.DarkOrchid, 10, 10)
```

You can also add an InputBox, which let the user enter his or her message then displays the message on the screen.

This is the sample code is as follows:

```
Dim myGraphics As Graphics = Me.CreateGraphics
Dim myFont As Font
Dim myBrush As Brush
Dim userMsg As String
UserMsg = InputBox("What is your message?", "Message Entry Form",
"Enter your message here", 100, 200)
myBrush = New Drawing.SolidBrush(Color.DarkOrchid)
myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Underline)
myGraphics.DrawString (userMsg, myFont, myBrush, 10, 10)
```

## 20.10 Drawing a Polygon

Polygon is a closed plane figure bounded by three or more straight sides. In order to draw a polygon on the screen, we need to define the coordinates of all the points (also known as vertices) that joined up to form the polygon.

The syntax to define the points of a polygon with vertices $A_1$, $A_2$ ...$A_n$ as follows:

```
Dim A1 As New Point(X1,Y1)
Dim A2 As New Point(X2,Y2)
.
.
Dim An as New Point(Xn,Yn)
```

After declaring the points, we need to define a point structure that group all the points together using the following syntax:

```
Dim myPoints As Point() = {A1, A2, A3,....,An}
```

Finally, create the graphics object and use the DrawPolygon method to draw the polygon using the following syntax:

```
Dim myGraphics As Graphics = Me.CreateGraphics
myGraphics.DrawPolygon(myPen, myPoints)
```

Where myPen is the Pen object created using the following syntax:

```
myPen = New Pen(Drawing.Color.Blue, 5)
```

**Example : Drawing a Triangle**

A triangle is a polygon with three vertices. Here is the sample code:

```
Dim myPen As Pen

Dim A As New Point(10, 10)

Dim B As New Point(100, 50)

Dim C As New Point(60, 150)

Dim myPoints As Point() = {A, B, C}

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPolygon (myPen, myPoints)
```

Run the program and you should see a triangle appears on the screen, as shown in Figure 20.7.

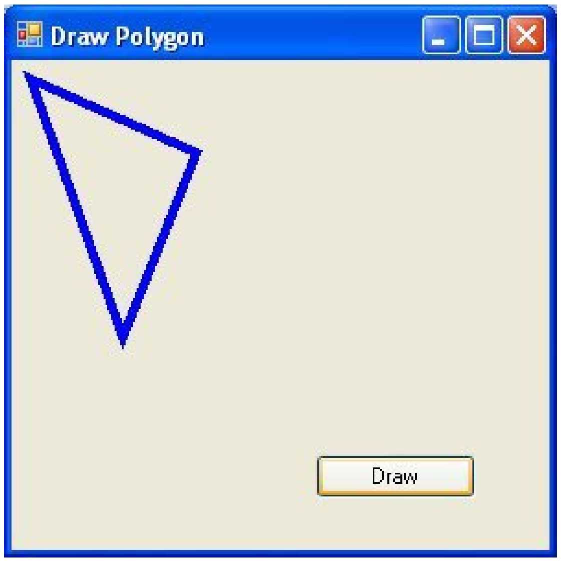

**Figure 20.7**

## Example: Drawing a Quadrilateral

A quadrilateral is a polygon consists of four sides, so you need to define four vertices. The following is the code:

```
Dim myPen As Pen

Dim A As New Point(10, 10)

Dim B As New Point(100, 50)

Dim C As New Point(120, 150)

Dim D As New Point(60, 200)

Dim myPoints As Point() = {A, B, C, D}

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPolygon (myPen, myPoints)
```

Run the program and you can see a polygon appears on the screen, as shown in Figure 20.8.

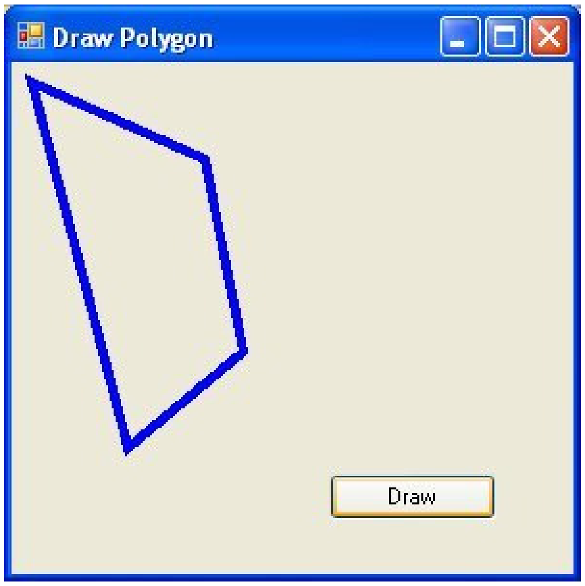

**Figure 20.8**

## 20.11: Drawing a Pie

In order to draw a pie, you can use the DrawPie method of the graphics object. As usual, you need to create the Graphics and the Pen objects. The syntax for drawing a pie is:

```
myGraphics.DrawPie (myPen, X, Y, width, height, StartAngle, SweepAngle)
```

Where X and Y are the coordinates of the bounding rectangle, other arguments are self-explanatory. Both StartAngle and SweepAngle are measured in degree. SweepAngle can take possible or negative values. If the value is positive, it sweep through clockwise direction while negative means it sweep through anticlockwise direction.

**Example:** Draw a pie that starts with 0 degree and sweep clockwise through 60 degree.

```
Dim myPen As Pen
myPen = New Pen(Drawing.Color.Blue, 5)
Dim myGraphics As Graphics = Me.CreateGraphics
myGraphics.DrawPie(myPen, 50,50, 150,150,0,60)
```

Run the program and you can see a pie appears on the screen, as shown in Figure 20.9

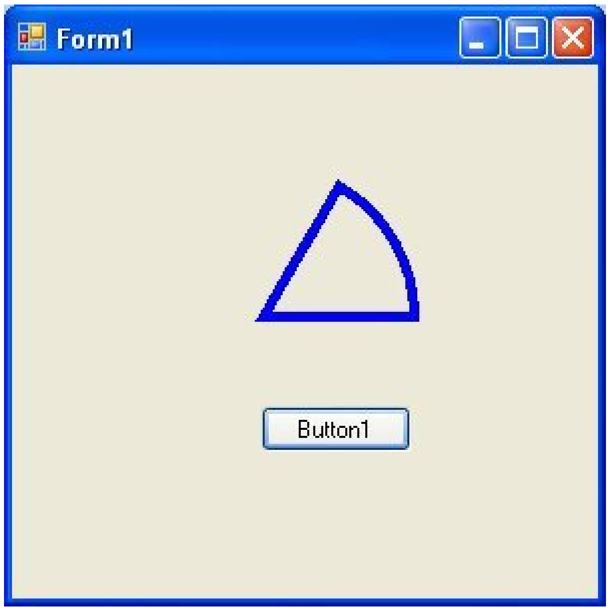

**Figure 20.9**

In previous sections, we have learned how to draw rectangle, ellipse, circle, polygon and pie with outlines only. In this section, we will show you how to fill the shapes with color, or simply solid shapes. You can use the following three methods to fill the shapes , they are **FillRectangle, FillEllipse, FillPolygon** and **FillPie**.

In order to fill the above shapes with color, we need to create the Brush object using the following syntax**:**

```
myBrush = New SolidBrush(Color.myColor)
```

Where myColor should be replaces by the names of the colors such as red, blue, yellow and more. You do not have to worry about the names of the colors because the intellisense will display the colors and enter the period after the Color key word.

```
Dim myPen As Pen
Dim myBrush As Brush
Dim myGraphics As Graphics = Me.CreateGraphics
myPen = New Pen(Drawing.Color.Blue, 5)
```

```
myBrush = New SolidBrush(Color.Coral)
myGraphics.DrawRectangle (myPen, 0, 0, 150, 150)
myGraphics.FillRectangle (myBrush, 0, 0, 150, 150)
```

Run the program and you can see a coral color square appears on the screen, as shown in Figure 20.10.

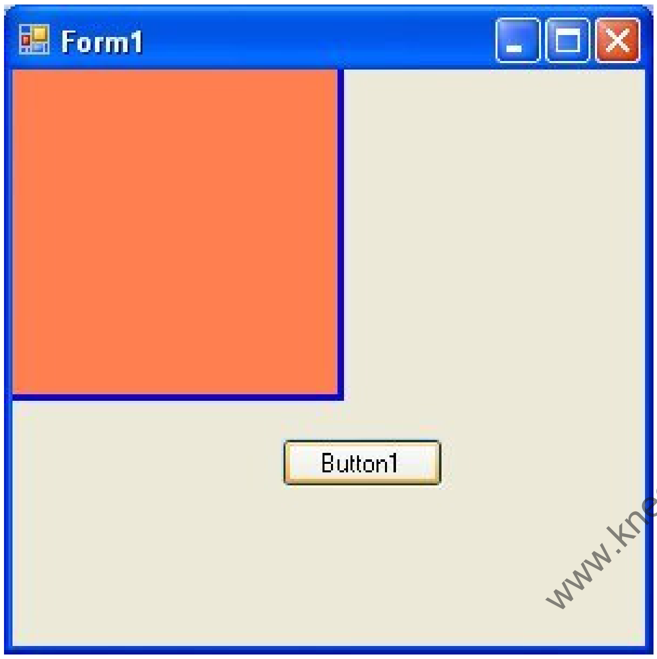

**Figure 20.10**

## 20.12 Drawing and Filling an Ellipse

The syntax to fill an ellipse with the color defined by the brush object is:

```
myGraphics.FillEllipse (myBrush, 0, 0, 150, 150)
```

The complete code is shown in the example below:

```
Dim myPen As Pen

Dim myBrush As Brush

Dim myGraphics As Graphics = Me.CreateGraphics

myPen = New Pen(Drawing.Color.Blue, 5)

myBrush = New SolidBrush(Color.Coral)
```

```
myGraphics.DrawEllipse(myPen, 0, 0, 150, 150)

myGraphics.Ellipse(myBrush, 0, 0, 150, 150)
```

Run the program and you can see a coral color ellipse appears on the screen, as shown in Figure 20.11

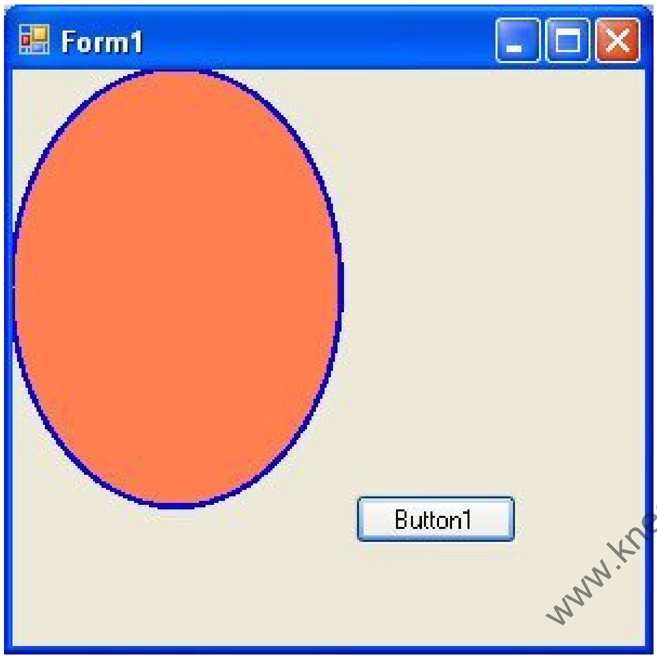

**Figure 20.11**

## 20.13 Drawing and Filling a Polygon

The syntax to fill a polygon with the color defined by the brush object is:

```
myGraphics.FillPolygon (myBrush, myPoints)
```

The complete code is shown in the example below:

```
Dim myPen As Pen
Dim myBrush As Brush
Dim A As New Point(10, 10)
Dim B As New Point(100, 50)
Dim C As New Point(120, 150)
Dim D As New Point(60, 200)
Dim myPoints As Point() = {A, B, C, D}
```

```
myPen = New Pen(Drawing.Color.Blue, 5)
myBrush = New SolidBrush(Color.Coral)
Dim myGraphics As Graphics = Me.CreateGraphics
myGraphics.DrawPolygon(myPen, myPoints)
myGraphics.FillPolygon(myBrush, myPoints)
```

Running the code produces the image as shown in Figure 20.12.

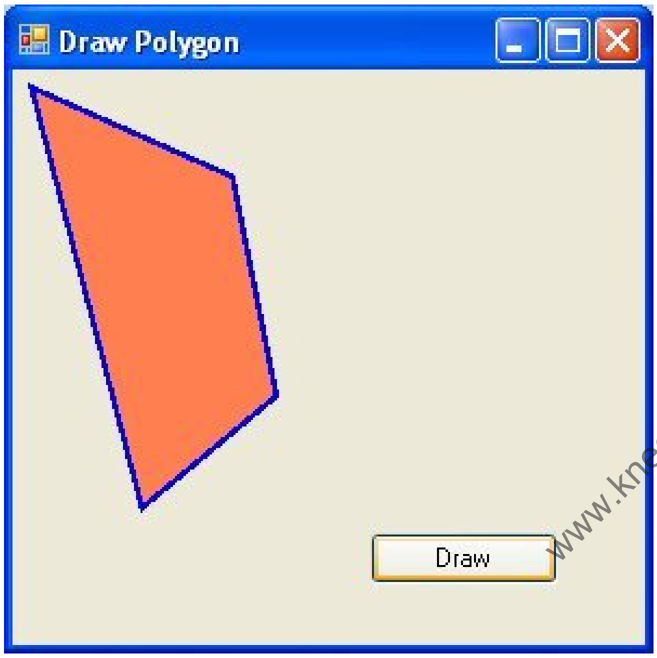

**Figure 20.12**

### 20.14 Drawing and Filling a Pie

The syntax to fill a pie with the color defined by the brush object is:

```
myGraphics.FillPie(myBrush, X, Y, width, height, StartAngle, SweepAngle)
```

The complete code is shown in the example below:

```
Dim myPen As Pen

Dim myBrush As Brush

myPen = New Pen(Drawing.Color.Blue, 5)

myBrush = New SolidBrush(Color.Coral)
```

```
Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPie(myPen, 30, 40, 150, 150, 0, 60)

myGraphics.FillPie(myBrush, 30, 40, 150, 150, 0, 60)
```

Run the program and you can see a coral color pie appears on the screen, as shown in Figure 20.13.

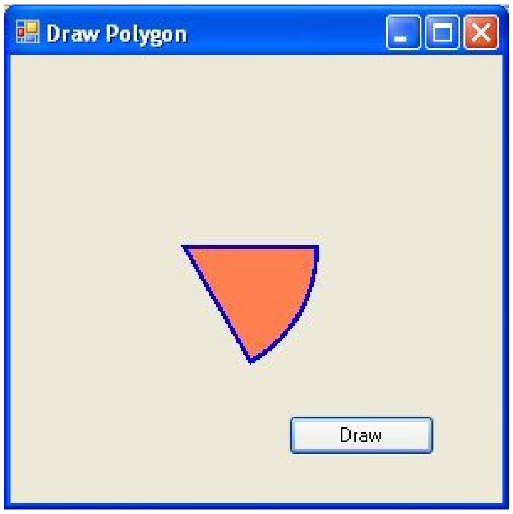

**Figure 20.13**

**Summary**

In this chapter, you learned how to draw various shapes and fill them with color.

- In section 20.1, you learned the basic concepts about graphics creation in VB2010.
- In section 20.2, you learned how to use the CreateGraphics() method to create a graphics object.
- In section 20.3, you learned how to create the Pen object
- In section 20.4, you learned how to use the DrawLine method to draw a line.
- In section 20.5, you learned how to use the DrawRectangle method to create a rectangle.
- In section 20.6, you learned how to customize the line style of the Pen object.
- In section 20.7, you learned how to use the DrawEllipse method to draw an ellipse.
- In section 20.8, you learned how to use the DrawEllipse method to draw a circle

- In section 20.9, you learned how to use the DrawString method to draw text on the screen.
- In section 20.10, you learned how to use the DrawPolygon method to draw a polygon.
- In section 20.11, you learned how to use the DrawPie method to draw a pie.
- In section 20.12, you learned how to fill an ellipse with color.
- In section 20.13, you learned how to fill a polygon with color.
- In section 20.14, you learned how to fill a pie with color.