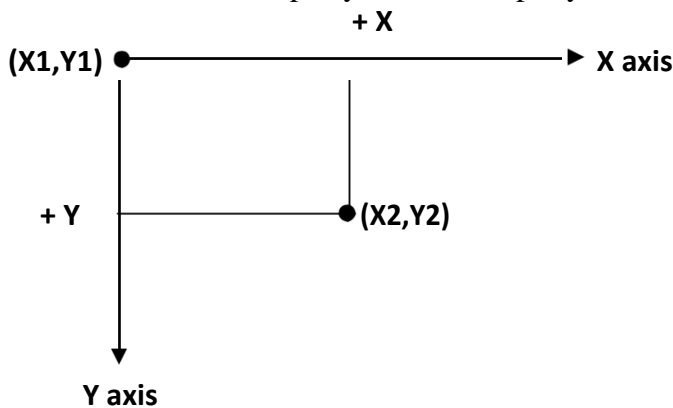

- Graphics in Visual Basic

1 Introduction: Graphics are the elements of a picture. Colors, lines, rectangles, patterns, text, etc. are all graphics. Graphics are visual. Visual Basic provides graphics capabilities for drawing shapes in different colors and patterns. Visual basic is also capable of displaying many popular image formats. Although the graphics capabilities may not be as feature rich as graphics software programs, visual basic's graphic capabilities are integral to creating polished windows applications.

2 Coordinate Systems: To draw in visual basic, we must understand Visual Basic's coordinate system as shown in figure below, that identifies points on the screen (such as forms or pictureboxes). By default, the upper-left point on the screen has coordinate (0,0), which is commonly called the origin. A coordinate pair is composed of an *x coordinate* (the horizontal coordinate) and *y coordinate* (the vertical coordinate). The x coordinate is the horizontal distance on the x axis from the origin. The y coordinate is the vertical distance on the y axis from the origin. The unit that a coordinate system is measured in is *called a scale*. Visual basic provides eight coordinate system scales. Most controls as well as the form use twips by default. Property *ScaleMode* specifies the scale.



User-defined coordinates are defined using method `Scale`. Two set of coordinates define the scale. The first coordinate set defines the upper-left corner and the second coordinate set defines the lower-right corner. The statement,

`Scale (xx1 , yy1) - (xx2 , yy2)`

For example:

- `Scale(0,0)-(100,100)`
- `Scale (100,100)-(0,0)`
- `Scale(-100,100)-(100,-100)`

1.2 Graphics Method:

Visual basic provides several methods for creating graphics. The graphics methods, summarized in the following table, apply to forms.

Method	Description
Line	Draws lines on a form. Can also be used to draw rectangles.
Circle	Draws circles on a form. Can also be used to draw ellipses
Pset	Sets a point's color
Print	Draw text on a form.

❖ **Method Line:** draws lines and rectangles between two sets of coordinates. The first set of coordinates is the starting point and the second is the ending point.

Line (x1,y1)-(x2,y2),color



For example:

- **Line (0,0)-(100,100),VbBlue**
- **Line(100,50)-(50,50),QbColor(5)**
- **Line(50,50)-(50,100),RGB(45,100,10)**

For rectangles (also called boxes) the first coordinate set specifies the upper-left corner and the second specifies the lower-right corner.

Line (x1,y1)-(x2,y2),color, B [or BF]

The visual basic constant (Vb) which represents the color name, the third argument (**B**), indicates that the method should draw a rectangle. A third argument of (**BF**) would indicate that the rectangle should be filled (solid). For example

- **Line (0,0)-(55,21), , B** 
- **Line(25,50)-(75,100), , Bf** 

➤ **Note:** There are three ways to specify a color value at run time.

1- Use RGB(1To 255, 1To 255,1 To 255) function

2- Use the QBColor(1 to 15) function to choose one of 15 Microsoft QuickBasic color as shown in table below

3- Enter a color value directly (VbColor) as shown in table below.

Vb Code	Color	Constant	Vb Code	Color	Constant
0	Black	vbBlack	8	Grey	vbGrey
1	Blue	vbBlue	9	Light Blue	vbLightBlue
2	Green	vbGreen	10	Light Green	vbLightGreen
3	Cyan	vbCyan	11	Light Cyan	vbLightCyan
4	Red	vbRed	12	Light Red	vbLightRed
5	Magenta	vbMagenta	13	Light Magenta	vbLightMagenta
6	Brown	vbBrown	14	Yellow	vbYellow
7	White	vbWhite	15	Bright White	vbBrightWhite

❖ **Method Circle:** draws circles, ellipses, arcs, and sectors. A circle's radius is the distance from the circle's center to any circle point. An ellipse differs from a circle in that its aspect ratio (the ratio of height to width) is not 1. Arcs is the curved portion of sectors. Sectors are wedge shaped pieces of a circle. Radians (from **0 to 2π**) must be used for sector and arc angles.

Circle (x1,y1), radius, color, start angle, end angle, proportion

For Example

Scale (0, 0)-(100, 100)

pi = 3.14156

- **Circle (50, 25), 5**

❖ **Method Pset:** turns on a point by changing the color at the point for example, the statement

Pset (x,y),color

Pset(40,40),VbRed

❖ **Method Print:** To draw text on the form. The default X coordinate is 0 and visual Basic automatically increments the y coordinate to draw on the next line. The current drawing coordinates are stored in properties **currentX** and **currentY**. For Example:

CurrentX=1

CurrentY=3

Print "Visual Basic"

Graphics Properties: Several drawing properties can be used with drawing methods. In this section, we introduce properties:

1- **DrawWidth:** The draw width property specifies the width of line for output from the graphics methods. For Example :

```
Private Sub Form Activate ()
```

```
Scale(0,0)-(100,200)
```

```
Drawwidth=1
```

```
Line(20,20)-(50,20)
```

```
Drawwidth=5
```

```
Line(20,40)-(50,40)
```

```
Drawwidth=8
```

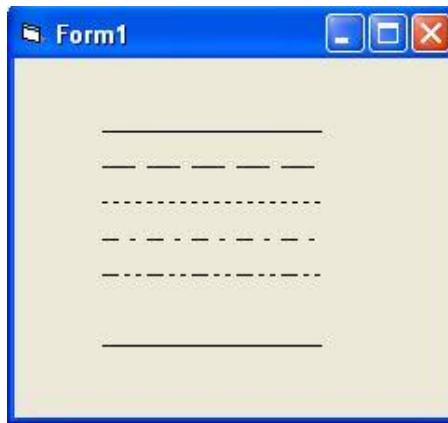
```
Line(20,60)-(50,60)
```



2- **DrawStyle**: the draw style property specifies whether the lines created with graphics methods are solid or have a broken pattern control. There are seven different draw style values (from 0 to 6).

For Example:

```
Private Sub
Form_Activate() Scale (0,
0)-(100, 100) DrawWidth =
1 Y = 10
For I = 0 To 6
DrawStyle = I
Y = Y + 10
Line (20, Y) - (70, Y)
Next I
```

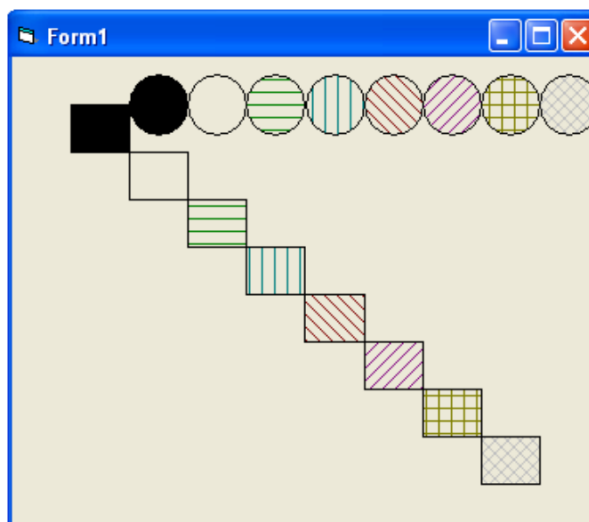


3- **FillStyle**: As long as you don't change the setting of the fill style Property, the box appears empty. (The box does get filled with default *FillStyle* and Settings, But *FillStyle* default to 1-Transparent). You can change the *FillStyle* property to any the settings listed in the following table:

Setting	Description
0	Solid. Fills in box with the color set for the <i>FillColor</i> Property
1	Transparent (the default). Graphical object appears empty, no matter what color is used
2	Horizontal lines
3	Vertical lines
4	Upward diagonal lines
5	Downward diagonal lines
6	Crosshatch
7	Diagonal Crosshatch

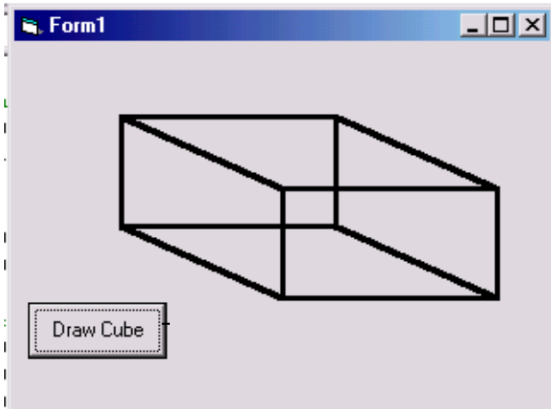
For Example:

```
Private Sub Form_Activate()
Scale (0, 0)-(100, 100)
For i = 0 To 7
Y = Y + 10
FillStyle = i
FillColor = QBColor(i)
Line (Y, Y)-(Y + 10, Y + 10), , B
Circle (Y + 15, 10), 5
Next i
```



Example: Write a code program to draw the figures below.

1- Solution:



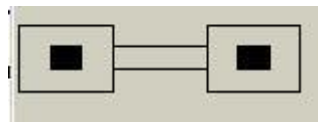
```
Project1 - Form1 (Code)
Command1 Click
Private Sub Command1_Click()
' Assign a scale
Scale (0, 0)-(100, 100)
DrawWidth = 3

'Draw 2 rectangles
Line (20, 20)-(60, 50), , B
Line (50, 40)-(90, 70), , B

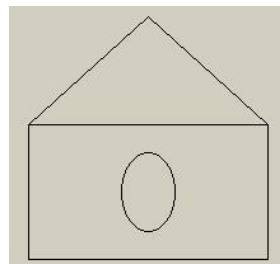
' Draw 4 connecting lines (the last is dotted)
Line (20, 20)-(50, 40)
Line (60, 20)-(90, 40)
Line (20, 50)-(50, 70)
Line (60, 50)-(90, 70)
DrawStyle = 2
' DrawStyle =1 means solid line
' DrawStyle =2 means dotted line
End Sub
```

Example: The following statements represent of visual Basic program that are used to generate the graph. Draw the figure and write all the necessary coordinates position into the graph.

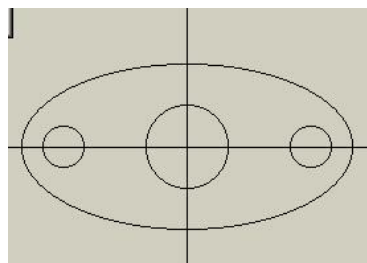
```
1-Private Sub Command1_Click()
Scale (0, 0)-(100, 100)
For i = 0 To 2 Step 2
Line (15 * i, 5)-(15 * (i + 1), 20), , B
Line (5 + 15 * i, 10)-(10 + 15 * i, 15), , BF
Next
Line (15, 10)-(30, 15), , B
End Sub
```



```
2-
Scale (0, 0)-(100, 100)
Line (20, 45)-(80, 95), , B
Line (20, 45)-(50, 5) Line
(50, 5)-(80, 45) Circle
(50, 70), 10, , , 1.5
```



```
3-Private Sub Command1_Click()
Scale (-10, 10)-(10, -10)
Circle (0, 0), 8, , , , 0.5
Circle (0, 0), 2
Circle (-6, 0), 1
Circle (6, 0), 1
```



Line (10, 0)-(-10, 0)

Line (0, 10)-(0, -10)

4-

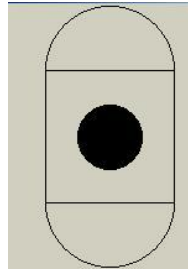
Scale (0, 0)-(40, 40) Line

(20, 15)-(30, 35), , B

Circle (25, 15), 5, , 0, 3.14

FillStyle = 0

Circle (25, 12), 1, , , , 1.5



5-

Scale (0, 0)-(120, 120)

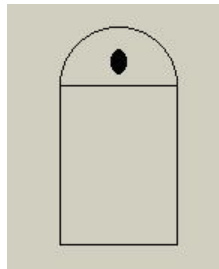
Line (30, 30)-(70, 90), , B

FillStyle = 0

Circle (50, 30), 20, , 0, 3.14 Circle

(50, 90), 20, , 3.14, 2 * 3.14

Circle (50, 60), 10



6-

Scale (-10, 10)-(10, -10)

For i = 0 To 6 Step 2

DrawStyle = 0

Line (0, 0)-(i, i), , B

Line (0, 0)-(-i, -i), , B

Next

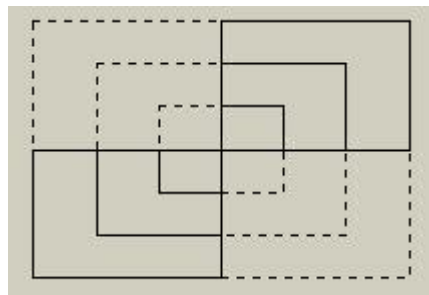
For i = 0 To 6 Step 2

DrawStyle = 2

Line (0, 0)-(i, -i), , B

Line (0, 0)-(-i, i), , B

Next



7.4 Graphics Controls: The Visual Basic provides three controls designed to create graphical effects in an application:

- The line Control
- The Shape Control
- The image control

These controls don't have event procedure. We first discuss drawing lines with the **Line** control. Unlike method line, which must be used at run-time, the line control can be used at design time. Lines can also be drawn at run-time with the **Line** control.

A line's color is specified using **Line** control's **BorderColor** property and a line's style is specified by setting the **Line** control's **BorderStyle** property. **Line** control line width(or thickness) is specified by setting the **BorderWidth**. Line length and position are specified using properties **X1, Y1, X2, and Y2**. **X1** and **Y1** specify the starting coordinate. **X2** and **Y2** specify the ending coordinate.

- Line1.BorderStyle=1
- Line1.BorderColor=Vbwhite
- Line1.X1=10
- Line1.Y1=15

The **Shape** control can be used to draw **rectangles, ellipses, rounded rectangles, squares, circles and rounded squares**. The **Shape** property specifies which **Shape** is drawn.

Value	Description
0	Rectangle
1	Square
2	Oval (i.e., an ellipse)
3	Circle
4	Rounded Rectangle
5	Rounded square

Shape control property *FillStyle* specifies how the shape is to be filled. The *BorederStyle* property specifies the style using the values from 0 to 6. *BackColor* , *FillColor* and *BorderColor* specify coloring. Note that *FillColor* and *BackColor* are ignored when either *FillStyle* or *BackStyle* is (Transparent). Property *BorderWidth* changes the width of lines.