GRAPHICS IN VISUAL BASIC

1 MEASUREMENT UNITS

Before starting discussion regarding the graphics command in VB, it is recommended first one should know about the graphics measurement units. Default units are Twips.

```
1 Twip = 1/20 Point

1 Point = 1/72 Inch   (1 inch = 1440 Twips)
```

Other measurement units are Point, Pixel, Character, Inch, Millimeter and Centimeter that may be selected from the ***ScaleMode*** property of object from properties window.

2 THE COORDINATE SYSTEM

Graphics are measured from origin 0,0 for the x and y coordinates starting from upper-left corner. The x is the horizontal and y is the vertical measurement.

The starting point depends on where the graphic is being placed. If the graphic is directly going on a ***Form***, the 0,0 coordinates are the upper-left corner of the Form object. Graphics can also be placed in other objects, like ***PictureBox***. Then ***PictureBox*** will have its own 0,0 that is its upper-left corner.

3 COLORS IN VISUAL BASIC

The colors can be assigned to graphics in VB by following three methods;

- The RGB Function
- The visual basic Intrinsic Color Constants
- The QBColor Function

3.1 The RGB Function

RGB stands for red, green and blue respectively. Following is the syntax of this function;

```
RGB (Red, Green, Blue)
```

For each of three indices in the brackets may have value from 0-255. **Table 1** is showing RGB function for some standard colors.

Table 1 *RGB Funtion for Standard Colors*

RGB Function	**Color**
RGB (0, 0, 0)	Black
RGB (255, 255, 255)	White
RGB (255, 0, 0)	Red
RGB (0, 255, 0)	Green
RGB (0, 0, 255)	Blue
RGB (0, 255, 255)	Cyan
RGB (255, 0, 255)	Magenta
RGB (255, 255, 0)	Yellow

3.2 The visual basic Intrinsic Color Constants

In this method colors can be specified using prefix vb followed by color name. Following are the color constants;

- vbBlack
- vbBlue
- vbGreen
- vbCyan
- vbRed
- vbMagenta
- vbYellow
- vbWhite

3.3 The QBColor Function

Using this function, QB color indices 0-15 may be used. QB color indices along with the color they are representing are given in **Table 2**. Following is the syntax of this function;

```
QBColor (Color Index)
```

Table 2 *QB Color Indices*

Index	Color	Index	Color
0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Yellow	14	Light Yellow
7	White	15	Bright White

4 GRAPHIC COMMANDS

Following are the graphics commands in VB;

- PSET
- LINE

4.1 PSET Command

PSET command is used to turn on a single point on the ***Form***, ***PictureBox*** or ***Image*** object. Following is the syntax;

```
PSET [STEP] (x,y) [,COLOR]
```

This will turn ON a single point at intersection of x column and y row on ***Form*** object. If it is required to turn ON a point on some other object like ***PictureBox*** following syntax is used;

```
Picture1.PSET [STEP] (x,y) [,COLOR]
```

In above commands, **STEP** and **COLOR** are optional parameters. If one uses **PSET** without **STEP** keyword, x and y co-ordinates are considered with reference to origin (upper left corner of the object). However, when **STEP** is included, point is turned ON having x and y co-ordinates with respect to immediate previous point. Hence, **STEP** keyword is always used when one is using the relative co-ordinate system instead of absolute co-ordinate system.

When **COLOR** keyword is omitted, color in the ***ForeColor*** property is assigned.

4.1.1 EXAMPLES

Example 1

It is required to turn ON four corner points of a rectangle on ***Form*** object as shown in **Fig. 1**. Consider the co-ordinates of point "A" as (50, 50).

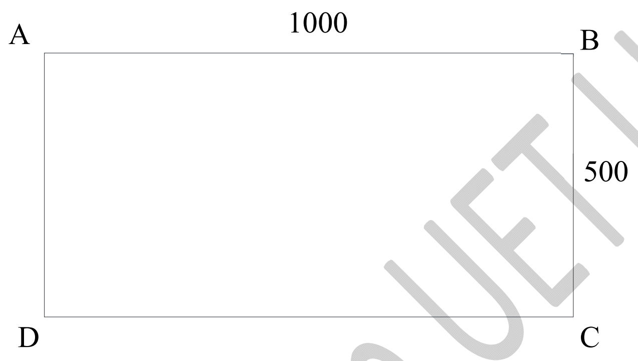

Fig. 1 Rectangle

Program

Double click the ***Form*** object and add following lines of coding in ***Form's*** click event.

Using absolute co-ordinate system

```
Private Sub Form_Click()
PSET (50, 50)       'draws point A
PSET (1050, 50)     'draws point B
PSET (1050, 550)    'draws point C
PSET (50, 550)      'draws point D
End Sub
```

Using relative co-ordinate system

```
Private Sub Form_Click()
PSET (50, 50), RGB (0,0,0)          'draws point A in black
PSET STEP (1000, 0), QBColor (2)    'draws point B in green
PSET STEP (0, 500), vbRed           'draws point C in red
PSET STEP (-1000, 0), vbBlue        'draws point D in blue
End Sub
```

Add a ***Command Button*** on the ***Form*** and change its ***Caption*** property to Clear and its ***Name*** property to cmdClear. Double click the Clear button and add following coding line in button's click event.

```
Private Sub cmdClear_Click()
CLS
End Sub
```

Here it is important to note that **CLS** command will only clear the ***Form*** object. If it is required to clear other objects like ***PictureBox*** and ***Image***, one must follow the dot notation as follows;

```
Picture1.CLS
Image1.CLS
```

Try to change the ***DrawWidth*** property of ***Form*** object from 1 to 5 and see what happens when you run the program.

Example 2

Program to place random dots in random colors on form

Program

In Form object's click event write the following coding lines;

```
Private Sub Form_Click()

Dim counter As Integer
Dim col As Integer
Dim x, y

For counter = 1 To 1000
    x = Rnd * ScaleWidth
    y = Rnd * ScaleHeight
    col = Rnd * 15
    PSet (x, y), QBColor(col)
Next counter

End Sub
```

Add a ***Command Button*** on the ***Form*** and change its ***Caption*** property to Clear and its ***Name*** property to cmdClear. Double click the Clear button and add following coding line in button's click event.

```
Private Sub cmdClear_Click()
CLS
End Sub
```

5 LINE COMMAND

LINE command is used to draw lines and rectangles. Following is the syntax;

```
   LINE [[STEP] (x1,y1)] - [STEP] (x2,y2) [,COLOR] [,[B][F]]
```

This will draw a line from point having coordinates (x1,y1) to point with coordinates (x2,y2) ***Form*** object. If it is required to draw a line on some other object like ***PictureBox*** having name Pic1, following syntax is used;

```
 Pic1. LINE [[STEP] (x1,y1)]-[STEP] (x2,y2) [,COLOR] [,[B][F]]
```

The keywords in square brackets are optional. The **STEP** keyword is used when one is using relative coordinates system. Keyword "**B**" is used to draw a rectangle by specifying (x1,y1) and (x2,y2) in line command as the end points of the diagonal. When keyword "**BF**" is used then filled rectangle can be drawn with color specified in specified in the **COLOR** keyword.

In order to draw lines with different styles, such as dotted line, dash line, center line or section line, at design time set the object's ***DrawStyle*** property to the required style. Also, this objective could be achieved by setting/changing this property at run time through coding.

NOTE: If ***DrawWidth*** property of the object is set to any value other than 1, ***DrawStyle*** property will not have any effect. Hence, in order to use line style other than solid, ***DrawWidth*** property of the object must be set to 1.

Following coding lines show how ***DrawStyle*** property can be set at the run time for ***Form*** object as well as for other objects like ***PictureBox*** and ***Image***.

Changing DrawStyle property at run time for Form object

```
DrawStyle = 0
Line (500, 500) - (3000, 500), vbRed    'solid red line

DrawStyle = 1
Line (500, 1000) - (3000, 1000), vbRed  'dashed red line

DrawStyle = 2
Line (500, 1500) - (3000, 1500), vbRed  'dotted red line
```

```
DrawStyle = 3
Line (500, 2000) - (3000, 2000), vbRed  ‘dash-dot red line

DrawStyle = 4
Line (500, 2500) - (3000, 2500), vbRed  ‘dash-dot-dot red line
```

Changing DrawStyle property at run time for PictureBox object

If ***Form*** contains a *PictureBox* object having its ***Name*** property set to Pic1. Following coding lines will be used to set the ***DrawStyle*** property of ***PictureBox*** object.

```
Pic1.DrawStyle = 0
Line (500, 500) - (3000, 500), vbRed    ‘solid red line

Pic1.DrawStyle = 1
Line (500, 1000) - (3000, 1000), vbRed  ‘dashed red line

Pic1.DrawStyle = 2
Line (500, 1500) - (3000, 1500), vbRed  ‘dotted red line


Pic1.DrawStyle = 3
Line (500, 2000) - (3000, 2000), vbRed  ‘dash-dot red line

Pic1.DrawStyle = 4
Line (500, 2500) - (3000, 2500), vbRed  ‘dash-dot-dot red line
```

5.1.1 EXAMPLES

Example 1

Draw rectangle shown in **Fig. 1**.using line command. Consider the co-ordinates of point “A” as (50, 50).

Program

Double click the ***Form*** object and add following lines of coding in ***Form’s*** click event.

Using absolute co-ordinate system

```
Private Sub Form_Click()
Line (50, 50)-(1050, 50)      'Draws AB line
Line (1050, 50)-(1050, 550)   'Draws BC line
Line (1050, 550)-(50, 550)    'Draws CD line
Line (50, 550)-(50, 50)        'Draws DA line
End Sub
```

Using relative co-ordinate system

```
Private Sub Form_Click()
Line (50, 50)-(1050, 50)       'Draws AB line
Line – STEP (0, 500)           'Draws BC line
Line – STEP (-1000, 0)         'Draws CD line
Line – STEP (0, -500)          'Draws DA line
End Sub
```

Using direct rectangle option

```
Private Sub Form_Click()
Line (50, 50)-(1050, 550),,B        'Draws Rectangle
End Sub
```

Add a ***Command Button*** on the ***Form*** and change its ***Caption*** property to Clear and its ***Name*** property to cmdClear. Double click the Clear button and add following coding line in button's click event.

```
Private Sub cmdClear_Click()
CLS
End Sub
```

Applying hatch patterns to shapes

If it is required to apply different hatch patterns, then object's ***FillStyle*** property should be set to desired pattern at design time as well as at run time through coding. In **LINE** command it is only available when rectangle option is used by including keyword "**B**". Color of the hatching is controlled by the object's ***FillColor*** property.