Graphics

Drawing devices

- Drawings or text can be placed on forms and controls
- Scale - the unit the coordinate system is measured in

![Diagram of axes (x, y)]

Form/control properties related to scale

- **ScaleLeft** - set the x coordinate for origin
- **ScaleTop** - set the y coordinate for origin
- **ScaleWidth** - set number of units for horizontal measurement of interior of object
- **ScaleHeight** - set the number of units for the vertical measurement of interior of object
- **ScaleMode** - set scale mode
Scale mode

- vbUser 0 User-defined. If you set ScaleWidth, ScaleHeight, ScaleTop, or ScaleLeft directly, the ScaleMode property is automatically set to 0.
- Twips 1 Default scale. 1,440 twips to one inch.
- Points 2 72 points to one inch.
- Pixels 3 pixels/inch a function of resolution of the device.
- Characters 4 When printed, a character is 1/6 of an inch high and 1/12 of an inch wide.
- Inches 5
- Millimeters 6
- Centimeters 7

Twip

- a screen-independent unit of screen measurement
- used to ensure that placement and proportion of screen elements in your screen application are the same on all display systems.
- equal to 1/20 of a printer's point.
- approximately 1440 twips to a logical inch or 567 twips to a logical centimeter

Changing scales

- object.Scale (x1, y1) - (x2, y2)
  object.Scale (-500, 250) - (500, -250)
  equivalent to
  object.ScaleTop = -500
  object.ScaleLeft = 250
  object.ScaleWidth = 500
  object.ScaleHeight = -250
- use of these methods sets ScaleMode to 0
Converting scales

[object.]ScaleX (width, fromscale, toscale)
[object.]ScaleY (height, fromscale, toscale)

→ Converts the value for the width/height of a Form, PictureBox, or Printer from one of the ScaleMode property’s unit of measure to another.

→ object - object expression that evaluates to an object in the Applies To list. If omitted, Form with the focus is assumed to be object.

Drawing methods

→ Print - draw text on a form or control
→ Line - draw lines, rectangles
→ Circle - draw circles, ellipses, arcs, sectors
→ Pset - set color of a point

Printing

→ Current drawing coordinates stored in properties CurrentX and CurrentY
→ CurrentX and CurrentY not available at design Time (you do not see them in property list)
→ Print “VB6 How to Program”
  → CurrentX = 0 by default
  → VB automatically increases y after each print
→ TextWidth property - width of text
→ TextHeight property - height of text
Print example 1
' Figure 9.5, page 375
' Printing to the form
Option Explicit
Private Sub cmdPrint_Click
    Call Randomize
    ' Randomly pick coordinates at which to draw
    CurrentX = Rnd()
    CurrentY = Rnd()
    ' Print at (CurrentX, CurrentY)
    Print "VB6 How to Program"
End Sub

Question: How to make sure entire text appear on form?

Print example 2
Private Sub Form_Click()
    Dim HalfHeight, HalfWidth, Msg   ' Declare variables.
    AutoRedraw = -1     ' Turn on AutoRedraw.
    BackColor = QBColor(4)     ' Set background color.
   ForeColor = QBColor(15)     ' Set foreground color.
    Msg = "Visual Basic"     ' Create message.
    FontSize = 48     ' Set font size.
    HalfWidth = TextWidth(Msg) / 2   ' Calculate one-half width.
    HalfHeight = TextHeight(Msg) / 2 ' Calculate one-half height.
    CurrentX = ScaleWidth / 2 - HalfWidth   ' Set X.
    CurrentY = ScaleHeight / 2 - HalfHeight   ' Set Y.
    Print Msg      ' Print message.
End Sub

Question: what if screen origin is not at (0, 0)?

Draw points
= [object.]Pset[Step](x, y), [color]
= Step (optional) - keyword specifying that (x, y) are relative to (CurrentX and CurrentY)
= (x, y) (required) - coordinates of the point to set
= color (optional) - Long integer value indicating the RGB color specified for point. If omitted, the current ForeColor property setting is used.
Pset (100, 100), vbRed
Pset Step (100, 100), vbGreen
Drawing lines

Line [Step] (x1, y1) [Step] - (x2, y2),
[color], [B|F]

- **Step** (optional, keyword) - specifying that starting point coordinates are
  relative to the current graphics position of (CurrentX, CurrentY).
- **x1, y1** (optional) - coordinates of the starting point for line or rectangle.
- If omitted, line begins at position (CurrentX, CurrentY).
- **Step** (optional, keyword) - specifying that end point coordinates are
  relative to the line starting point.
- **x2, y2** (required) - coordinates of the end point for the line.

- **Color** (optional) - Long integer value indicating the RGB color used to
draw the line. If omitted, ForeColor property setting is used. Can be
specified using the RGB function or QBColor function.
- **B** (optional) - If included, causes a box to be drawn using the coordinates
to specify opposite corners of the box.
- **F** (optional) - If the B option is used, the F option specifies that the box is
  filled with the same color used to draw the box.
  - You cannot use F without B.
  - B is used without F, the box is filled with the current FillColor and
    FillStyle.
  - The default value for FillStyle is transparent.

Drawing lines - examples

- **Line (0, 0) - (100, 100)**
  draw a line between (0, 0) and (100, 100)
- **Line (0, 0) - (100, 100), vbRed, B**
  draw a box between (0, 0) and (100, 100) using red color
- **Line (25, 50) - (50, 300), BF**
  draw a box with upper left corner at (25, 50) and
  lower right corner at (100, 100), filled with same color as that of line
- **Line (25, 50) - (50, 300), B**
  filled with the color and style defined in FillColor and
  FillStyle properties
Working with colors - VB color constant

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbBlack</td>
<td>&amp;H0</td>
<td>Black</td>
</tr>
<tr>
<td>vbRed</td>
<td>&amp;HFF</td>
<td>Red</td>
</tr>
<tr>
<td>vbGreen</td>
<td>&amp;HFF00</td>
<td>Green</td>
</tr>
<tr>
<td>vbYellow</td>
<td>&amp;HFFFF</td>
<td>Yellow</td>
</tr>
<tr>
<td>vbBlue</td>
<td>&amp;HFF0000</td>
<td>Blue</td>
</tr>
<tr>
<td>vbMagenta</td>
<td>&amp;HFF00FF</td>
<td>Magenta</td>
</tr>
<tr>
<td>vbCyan</td>
<td>&amp;HFFFF00</td>
<td>Cyan</td>
</tr>
<tr>
<td>vbWhite</td>
<td>&amp;HFFFFFF</td>
<td>White</td>
</tr>
</tbody>
</table>

Working with colors - QBColor function

QBColor(color) returns a long representing the RGB color code corresponding to the specified color number:

color - a whole number in the range 0–15.

<table>
<thead>
<tr>
<th>Number</th>
<th>Color</th>
<th>Number</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Black</td>
<td>8</td>
<td>Gray</td>
</tr>
<tr>
<td>1</td>
<td>Blue</td>
<td>9</td>
<td>Light Blue</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>10</td>
<td>Light Green</td>
</tr>
<tr>
<td>3</td>
<td>Cyan</td>
<td>11</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>12</td>
<td>Light Red</td>
</tr>
<tr>
<td>5</td>
<td>Magenta</td>
<td>13</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>6</td>
<td>Yellow</td>
<td>14</td>
<td>Light Yellow</td>
</tr>
<tr>
<td>7</td>
<td>White</td>
<td>15</td>
<td>Bright White</td>
</tr>
</tbody>
</table>

Working with colors - RGB function

- Assign each of the three primary colors (red, green, and blue) a number from 0 to 255, with 0 denoting the least intensity and 255 the greatest.
- Give these three numbers as input to the RGB function, using the order red-green-blue.
- Assign the result to the color property or color argument.

Example:
- Set background to green: Form1.BackColor = RGB(0, 128, 0)
- Set background to yellow: Form2.BackColor = RGB(255, 255, 0)
- Set point to dark blue: PSet(100, 100), RGB(0, 0, 64)
Drawing a yellow rectangle

- Line (0, 0) - (100, 100), vbYellow, B
- Line (0, 0) - (100, 100), QBColor(6), B
- Line (0, 0) - (100, 100), RGB(255, 255, 0), B

Drawing circles

\[
\text{Circle}\{x, y, \text{radius}, \text{color}\}
\]

Step (optional) - keyword indicating (x, y) is relative to (CurrentX, CurrentY)
(x, y) (required) - center of circle
radius (required) - radius of circle
color (optional) - color of circle

Circle (1200, 1000), 750
Circle ((ScaleWidth + ScaleLeft) / 2, (ScaleHeight + ScaleTop) / 2), ScaleWidth / 4

Drawing arcs

\[
\text{Circle}\{x, y, \text{radius}, \text{color}, \text{start, end}\}
\]

Step (optional) - keyword indicating (x, y) is relative to (CurrentX, CurrentY)
(x, y) (required) - center of circle
radius (required) - radius of circle
color (optional) - color of circle
start, end (optional) - (in radians between -2p, 2p) beginning and end positions of arc. Default 0 and 2p. 4 * Atn(1) = 4 * p /4 = p.
Circle (1200, 1000), 750
Circle ((ScaleWidth + ScaleLeft) / 2, (ScaleHeight + ScaleTop) / 2), ScaleWidth / 4
**Drawing ellipses**

```plaintext
[object.]Circle [Step](x, y, radius[, color, start, end, aspect])
```

- **Step** (optional) - keyword indicating (x, y) is relative to (CurrentX, CurrentY)
- (x, y) (required) - center of circle
- radius (required) - radius of circle
- color (optional) - color of circle
- **start, end** (optional) - specify (in radians between -2π, 2π) beginning and end positions of arc. Default 0 and 2π.
- aspect (optional) - aspect ratio of the circle. If not 1.0, an ellipse is drawn.

---

**Methods that change current X,Y**

<table>
<thead>
<tr>
<th>This method</th>
<th>Sets CurrentX, CurrentY to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>The center of the object.</td>
</tr>
<tr>
<td>Cts</td>
<td>0, 0.</td>
</tr>
<tr>
<td>EndDoc</td>
<td>0, 0.</td>
</tr>
<tr>
<td>Line</td>
<td>The end point of the line.</td>
</tr>
<tr>
<td>NewPage</td>
<td>0, 0.</td>
</tr>
<tr>
<td>Print</td>
<td>The next print position.</td>
</tr>
<tr>
<td>Pset</td>
<td>The point drawn.</td>
</tr>
</tbody>
</table>

---

**Drawing Properties**

- **Autodraw** - Boolean. Determine if a copy of graphics drawn is stored in memory (making them persistent).
  
  ```plaintext
  object.AutoRedraw = True
  ```

- **DrawMode** - determines appearance of graphics.
  
  ```plaintext
  object.DrawMode [= number]
  ```

- **DrawStyle** - determines the line style
  
  ```plaintext
  object.DrawStyle [= number]
  ```
DrawStyle constants

- **VbSolid** 0 (Default) - Solid
- **VbDash** 1 - Dash
- **VbDot** 2 - Dot
- **VbDashDot** 3 - Dash-Dot
- **VbDashDotDot** 4 - Dash-Dot-Dot
- **VbInvisible** 5 - Transparent
- **VbInsideSolid** 6 - Inside Solid

**Note:** DrawWidth > 1. DrawStyle settings 1 through 4 produce a solid line (the DrawStyle property value isn’t changed).

If DrawWidth = 1, DrawStyle produces the effect described in the above table for each setting.

Drawing Properties

- **DrawWidth** - Drawing width in pixels (integer 1 to 32767)
  
  `object.DrawWidth [= size]`

- **FillStyle** - Returns or sets the pattern used to fill Shape controls as well as circles and boxes created with the Circle and Line graphic methods.
  
  `object.FillStyle [= number]`
### FillStyle property constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VbFSSolid</td>
<td>0</td>
<td>Solid</td>
</tr>
<tr>
<td>VbFSTransparent</td>
<td>1 (Default)</td>
<td>Transparent</td>
</tr>
<tr>
<td>VbHorizontalLine</td>
<td>2</td>
<td>Horizontal Line</td>
</tr>
<tr>
<td>VbVerticalLine</td>
<td>3</td>
<td>Vertical Line</td>
</tr>
<tr>
<td>VbUpwardDiagonal</td>
<td>4</td>
<td>Upward Diagonal</td>
</tr>
<tr>
<td>VbDownwardDiagonal</td>
<td>5</td>
<td>Downward Diagonal</td>
</tr>
<tr>
<td>VbCross</td>
<td>6</td>
<td>Cross</td>
</tr>
<tr>
<td>VbDiagonalCross</td>
<td>7</td>
<td>Diagonal Cross</td>
</tr>
</tbody>
</table>

### Shape control

- A graphical control displayed as a rectangle, square, oval, circle, rounded rectangle, or rounded square.
- Use **Shape** controls at design time instead of, or in addition to, invoking **Circle** and **Line** methods at run time.
- **Shape** control can't act as a container.
- **Shape** control used to create static graphics or graphics to be displayed initially.

### Timer control

- Allow execution of code at regular intervals by causing a Timer event to occur.
- Invisible to the user (person who runs the application)
- Visible at design time
- Time interval set in Time.Interval in milliseconds
- $1000 = 1$ second