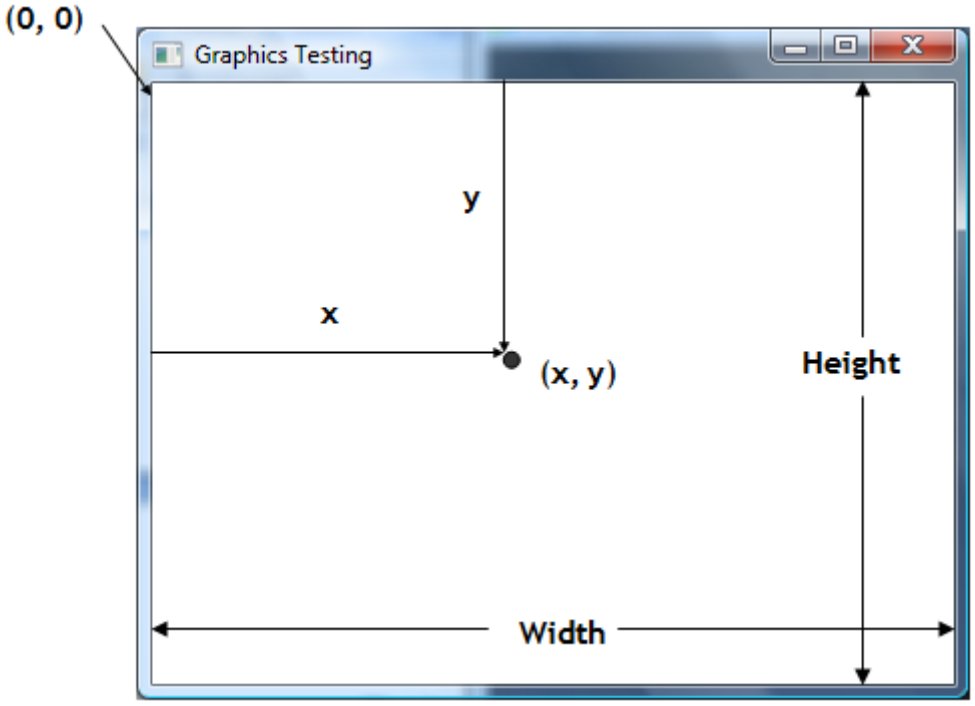
Small Basic Tutorial #2

GRAPHICS TIME!

SmallBasic has two ways of displaying output; a text mode and a graphics mode. Graphics mode can display both characters and graphics such as lines, boxes and other shapes.

You're working screen has the following layout, which is very much like a graph:

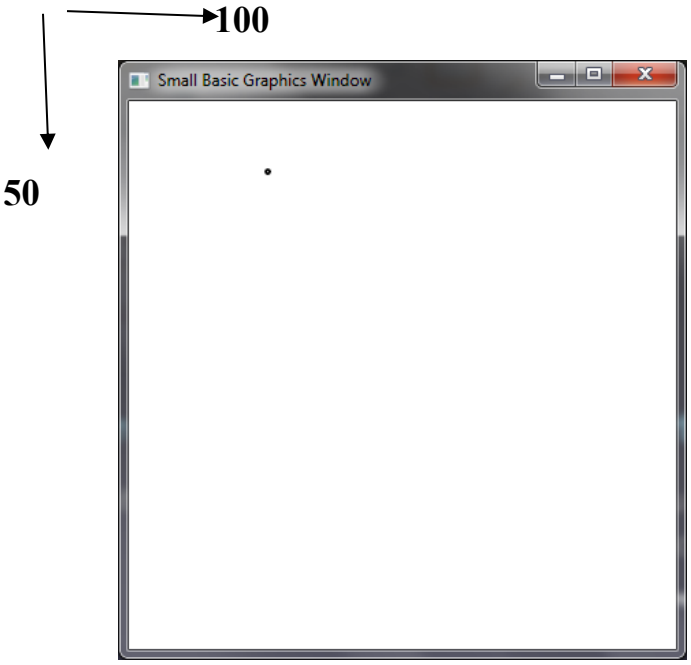


where the default is set to width = 624 and height = 444. We can easily change these dimensions to our own choices by typing the following lines:

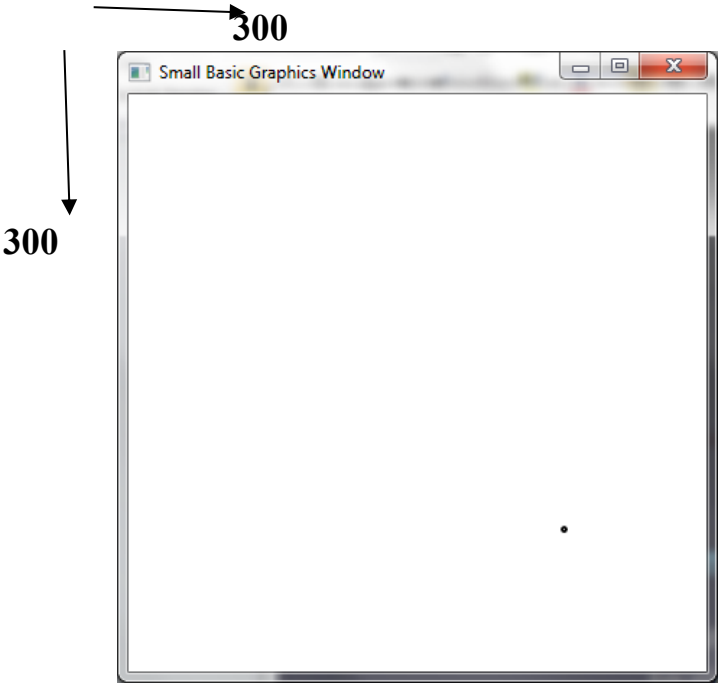
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 400
```

Or any number of your choice. I highly recommend changing the coordinates to values you understand as it will make organizing your output much, much easier.

So for example, on a window that's 400 x 400 pixels, a dot placed a x=100, y=50 looks like the following



One placed at x=300, y = 300 looks like:



Remember, the top left corner is always (0,0).

Drawing Time:

The first method I'd like to show you is DrawLine. This method does exactly what is says, it draws a line on the screen. Its code looks like the following:

```
GraphicsWindow.DrawLine(x1,y1,x2,y2)
```

Where

x1 = the position along the x-axis
y1 = the position along the y-axis
x2 = the position along the x-axis
y2 = the position along the y-axis

So for example try typing in

```
GraphicsWindow.DrawLine(10,200,200,200)
```

Pen Colour & Width:

The PenColor property will change the line color of the item you are drawing. The PenWidth property will change the border/line width of the item you are drawing.

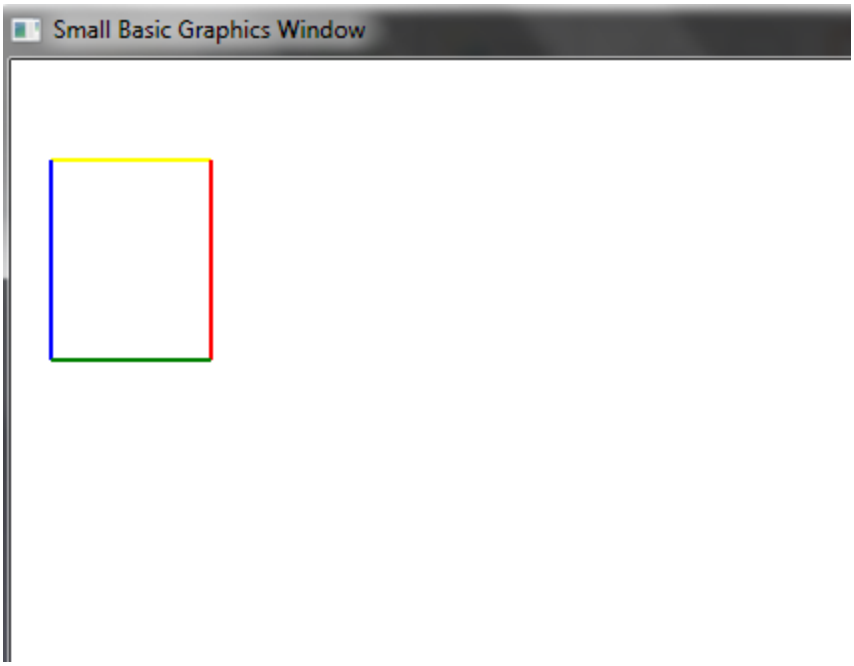
Try the following:

```
GraphicsWindow.PenColor = "blue"
GraphicsWindow.PenWidth = 10
GraphicsWindow.DrawLine(10,200,200,200)
```

Activity: Draw a vertical line and a diagonal line.

Activity: Try drawing a box using only lines commands. Create each line as a different color.

For example, mine looks like this when I run it:



Drawing Shapes:

Now that we've learned the basics of how to draw lines, lets look a bit closer at shapes. SmallBasic has commands that draw boxes and ovals.

Rectangle

The command is

```
GraphicsWindow.DrawRectangle(x,y,width,height)
```

where

x = the top left corner position along the x-axis
y = the top left corner position along the y-axis
width = determines how wide the rectangle will be
height = determines how high the rectangle will be

Oval

The command is

```
GraphicsWindow.DrawEllipse(x,y,width,height)
```

where

x = the top left corner position along the x-axis
y = the top left corner position along the y-axis
width = determines how wide the oval will be
height = determines how high the oval will be

Triangle

The command is

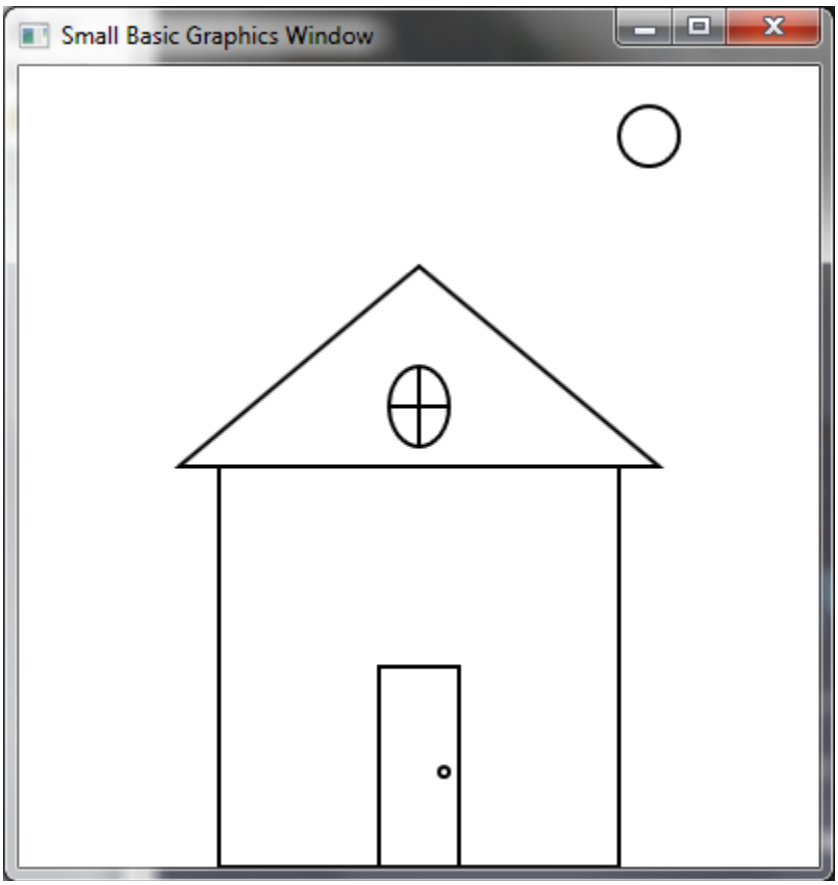
```
GraphicsWindow.DrawTriangle(x1,y1,x2,y2,x3,y3)
```

where

x1 = the x co-ordinate of the first point.
y1 = the y co-ordinate of the first point.
x2 = the x co-ordinate of the second point.
y2 = the y co-ordinate of the second point.
x3 = the x co-ordinate of the third point.
y3 = the y co-ordinate of the third point.

Activity:

Use rectangles, ovals, triangles and lines draw a house.



Brush Colour:

The BrushColor property allows you to fill a shape with a color. Try the following:

```
GraphicsWindow.BrushColor = "blue"
```

When wanting to fill a shape you have to use the BrushColor property and then one of the following fill statements:

```
GraphicsWindow.FillRectangle(x,y,width,height)

GraphicsWindow.FillEllipse(x,y,width,height)

GraphicsWindow.FillTriangle(x1,y1,x2,y2,x3,y3)
```

Colours in SmallBasic

There are several ways to create color in SmallBasic. The following will all output the same colour.

```
GraphicsWindow.BrushColor = "blue"
GraphicsWindow.BrushColor = "#0000FF"
GraphicsWindow.BrushColor = GraphicsWindow.GetColorFromRGB(0,0,255)
```

The following link will provide you with many color names and their related HEX codes.

Link → ***COLOURS***

For additional color details visit this link.

The following chart may also help you with some hex codes for the colours.

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

Drawing Font:

The font used to draw text in the graphics window is specified by four different properties:

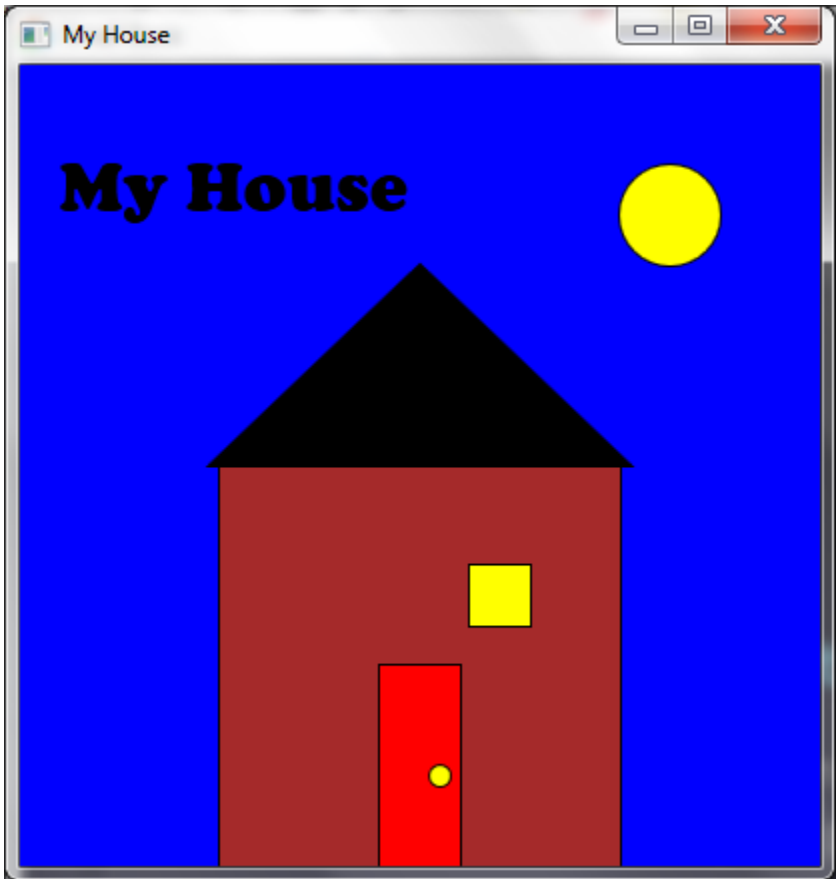
```
GraphicsWindow.FontName
GraphicsWindow.FontSize
GraphicsWindow.FontBold
GraphicsWindow.FontItalic
```

- The **FontName** property is the name of the font, or font face. The default value is "Tahoma". Other font names can be found by opening a word processor and selecting the change font option.
- The **FontSize** property defines the size of the font, where the default is 12.
- **FontBold** can have one of two values. If "true", the font will be bold. If "false", it will not be bold. The default value is "true".
- **FontItalic** indicates if a font is italicized. The default value is "false" – no italics.

Try the following code. What are all the properties in the following code doing?

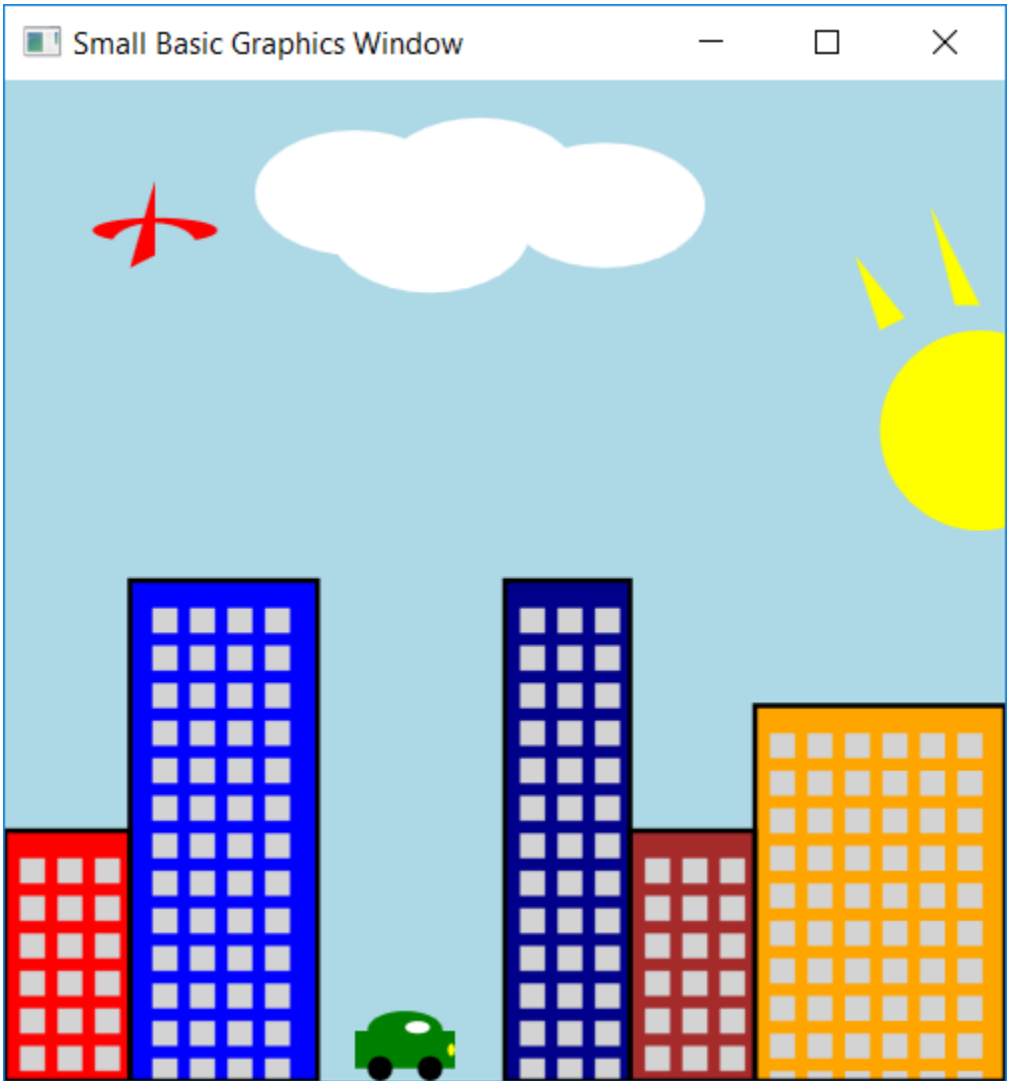
```
GraphicsWindow.Show()
GraphicsWindow.Title="Font Styles"
GraphicsWindow.Width=400
GraphicsWindow.Height=150
GraphicsWindow.BackgroundColor="Yellow"
GraphicsWindow.FontName = "Cooper Black"
GraphicsWindow.FontSize=36
GraphicsWindow.FontBold="true"
GraphicsWindow.FontItalic="true"
GraphicsWindow.BrushColor="Black"
GraphicsWindow.DrawText(20,40,"Graphics Window")
```

Activity: Recreate your house, but add colour and text. Attempt other class appropriate drawings to practice your coordinates.



CHALLENGE:

Create a scene like mine (but finish the sun rays).



Movement:

To make a shape move you need to think about your intended start and end point. For example if I want a box to move horizontal from x = 100 to x = 300 I could create the following program.

```
For x = 1 to 200
        GraphicsWindow.DrawRectangle(100+x,  190,  20,  20)
endfor
```

If you wanted your shape to move up and down which number would you change? What would you change if you wanted it to grow bigger?

Try inputting the following code. What is each line of code doing?

```
'setting the graphics window to 400 x 400
GraphicsWindow.Width = 400
GraphicsWindow.Height = 400

'preventing user from changing size of window
GraphicsWindow.CanResize = "false"

'Loop to move square from x = 100 to x = 300.
'x location increases for every iteration of the loop
For x = 1 To 200
  GraphicsWindow.Clear()  ' clear the screen
  GraphicsWindow.BrushColor = "blue" 'brush fill colour
  GraphicsWindow.DrawRectangle(100+x,190,20,20) ' draws square outline
  GraphicsWindow.FillRectangle(100+x,190,20,20) 'draws fill square
  Program.Delay(20) ' delaying program to show speed
endfor
```

DrawImage:

We have learned how to draw our own images using draw functions but in many cases we would want to include a saved image

When importing an image make sure it is saved in the same folder as your SmallBasic file. If this is the care you can use "Program.Directory" in place of the entire directory link. You MUST use this so that your programs work when you submit them to your teacher.

First save the image in a variable. In this case I created a variable called "Image". This variable is equal to the program directory.

```
Image = Program.Directory+"image.jpg"
```

Second there are two ways to put the image on the screen. Using just "DrawImage" will place the image at the specified location, and maintain the size of the image.

```
GraphicsWindow.DrawImage(ImageName,x,y)
```

Example:

```
GraphicsWindow.DrawImage(Image,100,100)
```

When using "DrawResizedImage" you have to place the image at a x and y location and then specify a width and height.

```
GraphicsWindow.DrawResizedImage(ImageName,x,y,width,height)
```

Example:

```
GraphicsWindow.DrawResizedImage(Image,100,100,10,10)
```

Example Code & Output:

```
Image = Program.Directory + "\Boom.png"
GraphicsWindow.DrawImage(image,200,100)
```

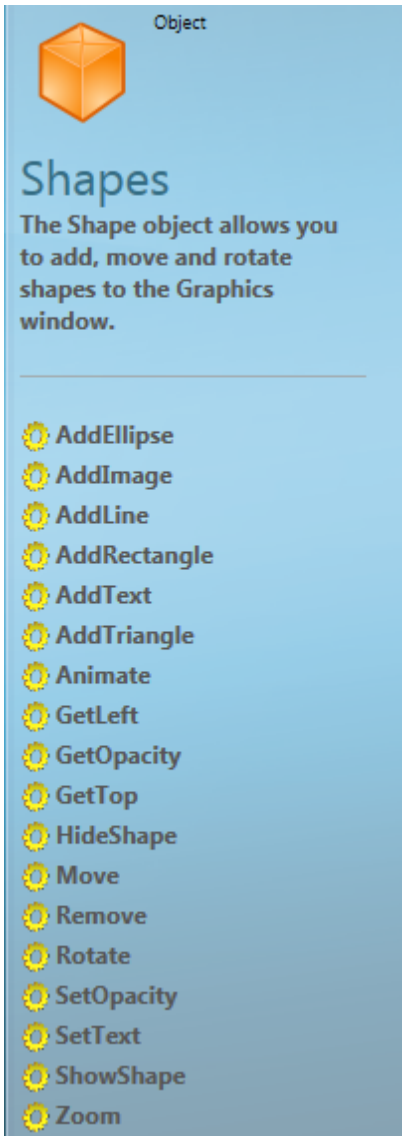

```
GraphicsWindow.DrawResizedImage(image,20,100,50,50)
```

Shapes

Although we have now learned how to draw and move shapes, there is a separate way to complete these tasks without using loops and without seeing the glitchy result.

SmallBasic has an object called "Shapes" which allows you to apply several modifications to the image including adding, moving, rotating and zooming.

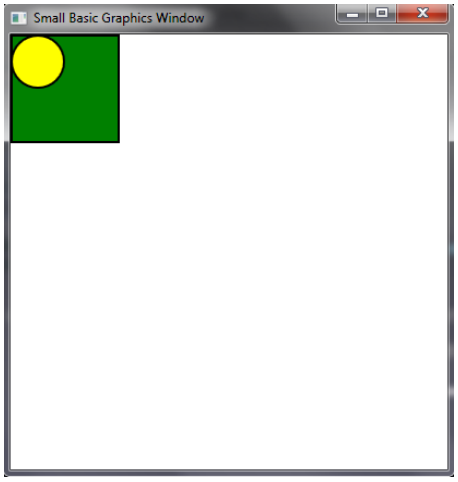
Keep in mind that shapes still work in layers. Layers are created in the same order your program is coded.



Creating Shapes

```
GraphicsWindow.BrushColor = "green"
Rectangle = Shapes.AddRectangle(100,100)

GraphicsWindow.BrushColor = "yellow"
Circle = Shapes.AddEllipse(50,50)
```

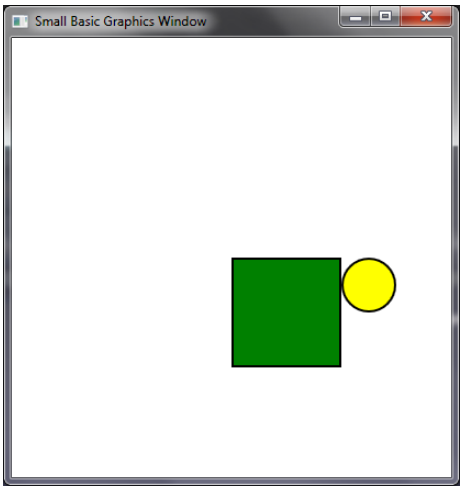


Moving Shapes

```
GraphicsWindow.BrushColor = "green"
Rectangle = Shapes.AddRectangle(100,100)

GraphicsWindow.BrushColor = "yellow"
Circle = Shapes.AddEllipse(50,50)

Shapes.Move(Rectangle, 200,200)
Shapes.Move(Circle, 300,200)
```



Animating Shapes

The “Animate” command is not allowed!

Hiding Shapes

```
GraphicsWindow.BrushColor = "green"
Rectangle = Shapes.AddRectangle(100,100)

For counter = 1 To 5
  Program.Delay(1000)
  Shapes.HideShape(Rectangle)
  Program.Delay(1000)
  Shapes.ShowShape(Rectangle)
EndFor
```

Rotating

```
'creates rectangle
GraphicsWindow.BrushColor = "Purple"
rotateshape = Shapes.AddRectangle(150, 100)
Shapes.Move(rotateshape, 200, 150)

' repeats rotation of rectangle for 12 iterations (30 * 12 = 360)
For i = 0 To 12
  'Rotataing image by 30 degree intervals
  Shapes.Rotate(rotateshape, 30 * i)

  'Printing the degree value on the screen
  GraphicsWindow.DrawText(20,20*i,30*i)
  Program.Delay(1000)
EndFor
```

Zooming & Opacity

Shapes.SetOpacity(ShapeName, level)

- Level range is 0 for completely transparent to 100 for solid

Shapes.Zoom(ShapeName, X-level, Y-level)

- Range allowed is 0.1 to 20

```
'Creating two rectangles
GraphicsWindow.BrushColor = "Purple"
rectangle1 = Shapes.AddRectangle(100, 100)
Shapes.Move(rectangle1, 50, 80)
rectangle2 = Shapes.AddRectangle(100, 100)
Shapes.Move(rectangle2, 300, 80)

For i = 1 To 4
  Program.Delay(1000)

  'Setting Opacity in increments of 25
  Shapes.SetOpacity(rectangle1, i * 25)

  'Zooming the image by 1/2 intervals.
  Shapes.Zoom(rectangle2, i * 0.5, i * 0.5)
EndFor
```

Creating repeating shapes

```
GraphicsWindow.Height = 500
GraphicsWindow.Width = 700
'loop to create 20 boxes
For i = 0 To 20
  GraphicsWindow.PenWidth = 0.5  'creates a thin border

  'determining a random colour
  GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()

  'creating a new rectangle in multiples of 20 wide, 10 high
  rectangle1 = Shapes.AddRectangle(i * 20, i * 10)

  'moves the rectangle to locations of multiples of 10
  Shapes.Move(rectangle1, i * 10, i * 10)
  Program.Delay(500)
EndFor
```

Activity

Write a program to display a graphics window, and perform the following steps:

- Add a line and a circle to the window.
- Set the color, size, and location of the shapes.
- Use loops to make the circle move on top of the line from the left side to the right side of the graphics window.