CS A201
Intro to Drawing Graphics

To draw some simple graphics, we first need to create a window. The easiest way to do this in the current version of Java is to create a JFrame object which is part of the Swing library. When we create a JFrame object we actually create a new window on the screen. Here is an example:

```
import javax.swing.JFrame;
import java.awt.Graphics;

public class GTest extends JFrame
{
	public GTest()
	{
		setSize(300, 200);	// 300 pixels across, 200 down
		setDefaultCloseOperation(EXIT_ON_CLOSE);
		setVisible(true); // Makes window visible
	}

	public static void main(String[] args)
	{
		GTest myWindow = new GTest(); // Title
	}
}
```

This program creates the following window and makes it visible:

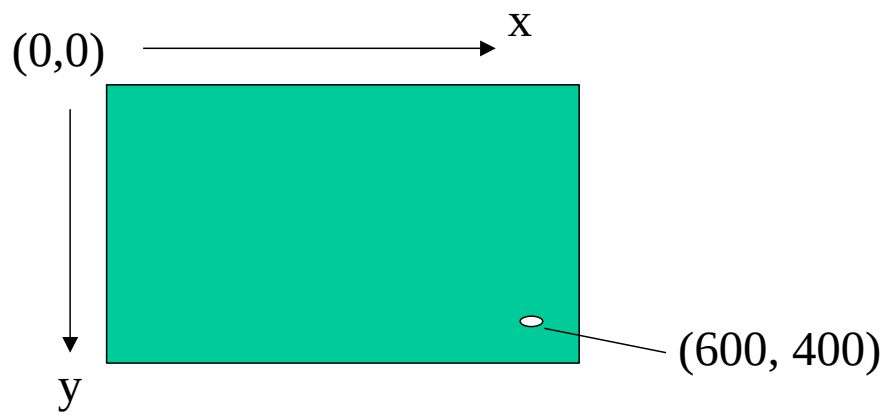
The import statements give us access to Java graphics libraries.

The code "`public GTest()`" is called a constructor. It is a place we can put initialization code, and it is invoked from the line "`GTest myWindow = new GTest();`". We'll cover more on this later, for now you can consider this boilerplate for making a GUI window.

Note that we picked the name `GTest` as short for Graphics Test. You would use whatever name you like to describe the program you are writing.

Drawing in the JFrame

Once we have a JFrame, we will get access to a graphics object on the frame.
The graphics screen is set up as a grid of pixels where the upper left coordinate is 0,0. This is relative to where the canvas is on the viewframe. The x coordinate then grows out to the right, and the y coordinate grows down toward the bottom.



Here is some sample code that illustrates how to draw a rectangle and a string of text on the screen. You can use this as a template for your own programs, where you would likely want to draw something different inside the window in the paint() method:

```
import javax.swing.JFrame;
import java.awt.Graphics;
import java.awt.Color;

public class GTest extends JFrame
{

	// The paint method below determines what to
	// draw on the screen and is invoked by Java
	public void paint(Graphics g)
	{
		super.paint(g);
		g.setColor(Color.red);  // Sets color to red
		// NOTE: Title Bar about 22 pixels down
		// Border frame about 7 pixels across

		// Draw rectangle at X=10, Y=50,
		// width=100, height = 70  pixels
		g.drawRect(10,50,100,70);
		// Write text at X = 100, Y = 150
		g.drawString("Moof.", 100, 150);
	}

	public GTest()
	{
		setSize(300, 200);	// 300 pixels across, 200 down
		setDefaultCloseOperation(EXIT_ON_CLOSE);
		setVisible(true); // Makes window visible
```

```
	}

	public static void main(String[] args)
	{
		GTest myWindow = new GTest(); // Title
	}
}
```

This code produces the following image:

In this code, we are importing java.awt.Color to get access to some predefined colors.

When Java goes to display what is in the window, it invokes the `paint()` method. The parameter is a Graphics object that supports many drawing functions. Here we are using `drawRect` and `drawString`.

`drawRect(x1,y1,width,height)` draws a rectangle with the upper left corner at (x1,y1) and the specified width and height.

`drawString(string, x1, y1)` draws the text string at coordinates x1, y1.

It is important to note that our drawing coordinates include pixels used for the title bar and the frame around the border. With my widget scheme the title bar is about 23 pixels, and the border about 7 pixels wide. However, if we try to draw something at that location, we get nothing. For example, if the drawRect is:

```
	g.drawRect(0,0,100,70);
```

Then the output will look like:

Colors

Here are different colors available using the color class:

Color.black	Color.darkGray	Color.gray
Color.blue	Color.green	Color.lightGray
Color.cyan	Color.magenta	Color.orange
Color.pink	Color.red	Color.white
Color.yellow		

To create our own color, we can specify the color we want in RGB (red, green, blue) color model. This is an additive color model that is somewhat like mixing colors of light or like mixing paints. The idea is that any color is represented as some combination of red, green, and blue. To specify a color use:

```
new Color(redvalue, greenvalue, bluevalue);
```

where each value is in the range 0-255.

For example:

```
new Color(0,255,200);
```

Creates a green-blue color, with stronger green than blue. If red, green, and blue are all 0 then this is the color black. If red, green, and blue are all 255 then this is the color white. Red and green makes yellow, red and blue makes magenta, blue and green makes cyan, etc.

More drawing functions

Here is a list of other drawing functions you can use with the graphics object:

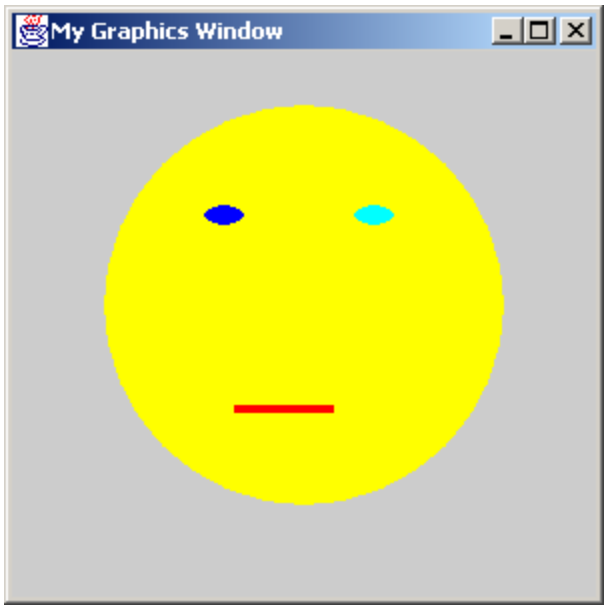
setColor(color)
Sets the current color to a color defined as above
drawLine(int x1, int y1, int x2, int y2)
Draws a line in the current color from x1,y1 to x2,y2
Can use drawLine(x1,y1,x1,y1) to draw a single pixel at x1,y1
drawOval(int x, int y, int width, int height)
Draws an oval in the current color within a box at upper left corner x,y
With specified width and height in pixels
Use width = height to draw a circle
drawRect(int x, int y, int width, int height)
Draws a rectangle in the current color with upper left corner at x,y
With specified width and height in pixels
drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
Draws an arc in the current color at the coordinates and specified angle
fillOval(int x, int y, int width, int height)
Draws an oval in the current color within a box at upper left corner x,y
With specified width and height in pixels
Use width = height to draw a circle
fillRect(int x, int y, int width, int height)
Fills a rectangle in the current color with upper left corner at x,y
With specified width and height in pixels
fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)
Fills an arc in the current color at the coordinates and specified angle

There are many more, for a full list see
http://download.oracle.com/javase/1.4.2/docs/api/java/awt/Graphics.html

Here is an example using some of these functions to draw an ambivalent yellow face:

```
public void paint(Graphics g)
{
      super.paint(g);
      g.setColor(Color.yellow);
      g.fillOval(50,50,200,200);
      g.setColor(new Color(0,0,255));     // Blue left eye
      g.fillOval(100,100,20,10);
      g.setColor(new Color(0,255,255));   // Blue-Green right eye
      g.fillOval(175,100,20,10);
      g.setColor(new Color(255,0,0));     // Red mouth
      g.fillRect(115,200,50,4);
}
```

This program draws the following image:


My Graphics Window