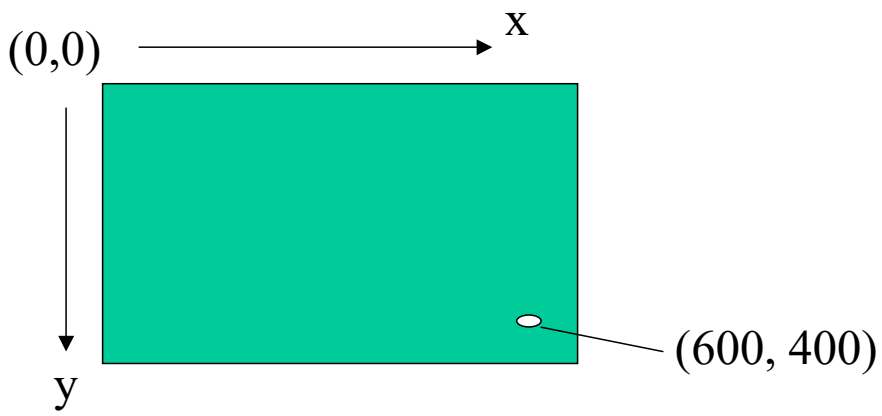
Graphics in C++ with GLUT and OpenGL

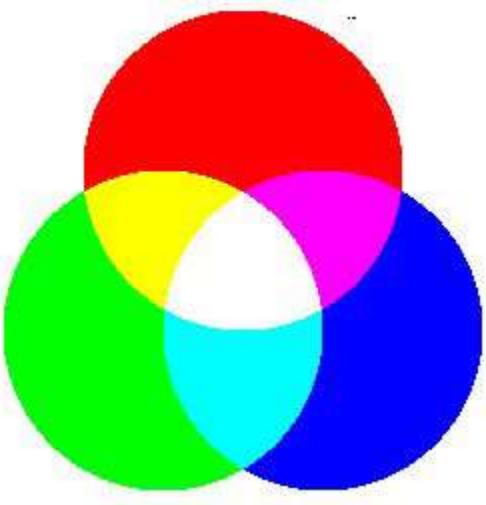
OpenGL is the Open Graphics Library and GLUT is the OpenGL Utility Toolkit. Together these tools provide a uniform interface that is commonly used to write 3D graphics programs. Here we will only provide some examples and a framework using OpenGL for 2D graphics. We'll start just by introducing how to draw simple shapes and pixels and then later how to draw sprites and animate them.

To configure OpenGL on a Visual Studio system, see lab assignment 9. You can use OpenGL on many platforms but the configuration steps are different. See http://www.opengl.org/wiki/Getting_Started for more information.

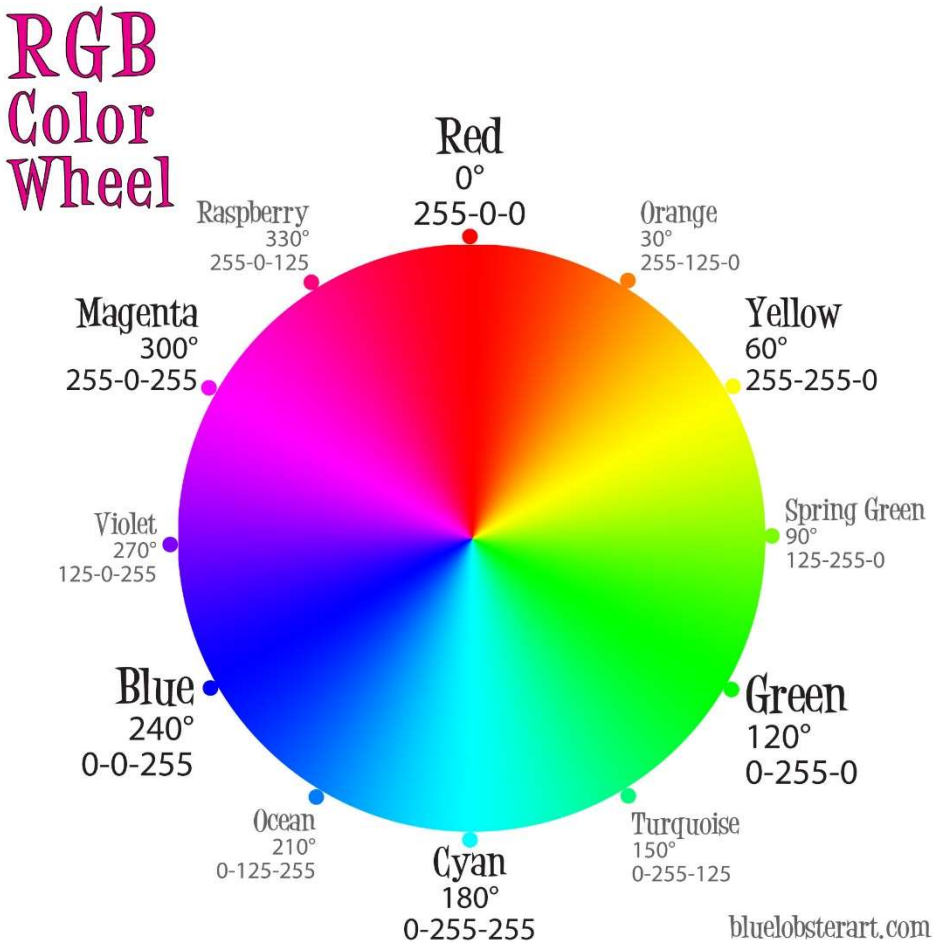
The graphics screen is set up as a grid of pixels where the upper left coordinate is 0,0. The x coordinate then grows out to the right, and the y coordinate grows down toward the bottom.



Colors are specified using the Red Green Blue (RGB) model. In this model we have a number that represents the amount of red, green, and blue. Normally we use a number from 0-255 for each color component, but in OpenGL we will use a number from 0-1. The Venn diagram below shows how you can get different colors as your intersect the red, green, and blue.



The color wheel below shows more ranges of color for different RGB values. Once again, for our case, the values range from 0-1 instead of 0-255 so substitute 1 for 255, and 0.5 for 127, etc.



There is some code setup to set up and initialize a glut window that we are skipping for now. In the Lab 9 code, focus on the code inside the display() function. Here is some code with a description of what it does:

```
// Clear the screen
glClear(GL_COLOR_BUFFER_BIT);
printText(WIDTH/5, HEIGHT/4, "This demonstrates some simple 2D drawings");

// Draw some points
glBegin(GL_POINTS);
// Change color to green  (r,g,b) where r,g,b is 0-1
glColor3f(0, 1, 0);
glVertex2f(100, 100);
glColor3f(1, 1, 1);  // Color to white
glVertex2f(10, 20);
glEnd();
```

```
// Draw a line
glBegin(GL_LINES);
glVertex2f(50, 50);
glVertex2f(75, 150);
glEnd();

// Draw a rectangle given the four corners
glBegin(GL_QUADS);  // Use GL_LINE_LOOP for hollow
glVertex2f(200, 10); // x1,y1
glVertex2f(300, 10); // x2, y1
glVertex2f(300, 50); // x2, y2
glVertex2f(200, 50); // x1, y2
glEnd();

// Draws a polygon given the coordinates
glBegin(GL_POLYGON);
glColor3f(1, 0, 0); // Change to red
glVertex2f(410, 10);
glVertex2f(510, 210);
glVertex2f(410, 210);
glVertex2f(401, 100);
glVertex2f(410, 50);
glEnd();
```

Here is a sample program to write in class: The Sierpinski Gasket

A Sierpinski Gasket or Triangle is a type of fractal named after the Polish mathematician Waclaw Sierpinski who described some of its interesting properties in 1916. It is a nice example of how an orderly structure can be created as a result of random, chaotic behavior.

One way to create the fractal is to start with a triangle. Let's say that the corners are labeled A, B, and C.

1. Set *current* equal to point A
2. Repeat the following four steps many times (you can try 50000)
    1. Randomly pick *target* as one of the three A, B, or C.
    2. Calculate the point halfway between *current* and *target*
    3. Set *current* to this halfway point
    4. Draw a pixel at location *current*.