Assignment 1: ImageMaker

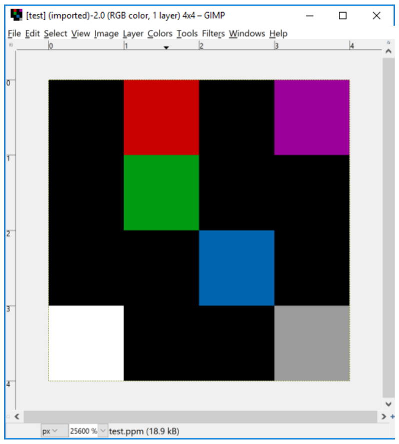
Portable Pix Map Image Format Specification

The portable pix map image format (ppm) is one of the simplest methods of creating an image file. This format is encoded as readable ascii text or viewed as an image with an image viewing program such as gimp.

In this assignment, you will make a class that create a ppm image using basic drawing methods. In addition, you will make a driver that can be used to help test the functionality of the class. **BE SURE TO READ THE DELIVERABLES!**

RGB Format

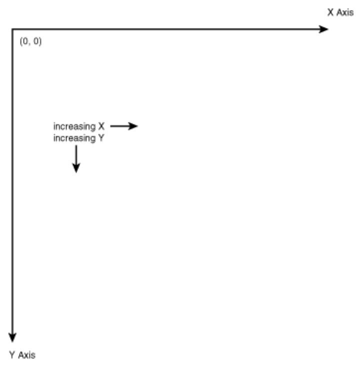
```
P3
4 4
255
0   0   0    200 0   0    0  0   0    155  0   155
0   0   0    0   155 275  0  0   0    0    0   0
0   0   0    0   0   0    0 100 175   0    0   0
255 255 255  0   0   0    0  0   0    155  155 155
```


[test] (imported)-2.0 (RGB color, 1 layer) 4x4 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
px
25600 %
test.ppm (18.9 kB)

The first three lines defines the image header. P3 indicates that the image will use RGB color. The second line indicates width and height. The 255 indicates the maximum value for the color. In the example above colors are between 0-255. Read the following wiki-link for additional details: https://en.wikipedia.org/wiki/Netpbm_format

Coordinate System

Unlike in math, the standard coordinate system for images starts in the upper left hand corner.



Class Definition

The following is the class definition:

```
const int MAX_WIDTH = 640;
const int MAX_HEIGHT = 480;
const int MAX_COLOR = 255;
enum COLOR {RED, GREEN, BLUE};

using namespace std;

class ImageMaker
{
public:
       ImageMaker();
       ImageMaker(string filename);

       void LoadImage(string filename);
       void SaveImage(string filename); // Write the matrix to file

       // Size functions
       int GetWidth();
       int GetHeight();
       void SetWidth(int width);
       void SetHeight(int height);

       // Color functions
       int GetRed();
       int GetGreen();
       int GetBlue();
       void SetRed(int newR);
       void SetGreen(int newG);
       void SetBlue(int newB);

       // Drawing methods
       void DrawPixel(int x, int y);
       void DrawRectangle(int x1, int y1, int x2, int y2);
       void DrawLine(int x1, int y1, int x2, int y2);

private:
       int width;
       int height;
       int red;
       int green;
       int blue;
       short image[MAX_WIDTH][MAX_HEIGHT][3];
};
```

```
image[x][y][RED] = red
```

Before you start looking at how to implement this class, you must determine and document the appropriate preconditions and postconditions for each operation.

Test Driver

Before this class can become a permanent part of your program library, it must be thoroughly tested. The test driver should support the following test commands:

```
DrawPixel x y
    Draw a pixel at x, y
DrawRectangle x1 y1 x2 y2
    Draw a rectangle using the points x1, y1 to x2, y2
DrawLine x1 y1 x2 y2
    Draw a line from x1, y1 to x2, y2
SetColor r/g/b value
    Sets the r, g, or b to value
PrintRGB
    Prints out the RGB
SetSize width height
    Sets the width and height
PrintSize
    Prints out the size
Load imagename
    Loads a ppm image with name imagename
Save imagename
    Saves the image using the name imagename
Quit
```

Notes

1. You can assume that if you use the load function that the file is formatted correctly
2. Be sure to refer to the other test driver examples
3. The ImageMaker class must throw an error if bad input is given. The test driver must be able to catch these errors and continue. Be sure to refer to examples given in class.
4. Clarity and formatting will be counted as part of the grade.
5. Pre and post conditions should be included for each method.

Deliverables

- Your ImageMaker class
- Your test driver class
- Your test plan as input to the test driver (See page 122 for an example)
  (THE TEST PLAN IS NOT TRIVIAL)
- The input files used with your test driver
- The output files from the test driver

Extra Credit Opportunities

The following are some ideas for extensions that can be done for extra credit. Points are additive (more features, more points).

Additional drawing methods/features (1-10%)

Extra credit awarded depending how difficult the drawing method or feature is. For example, a DrawTriangle would have a relatively low extra credit. A filled rectangle would get more points.