CISC 1115 (Science Section)
Brooklyn College
Professor Langsam

Assignment #5

An image is made up of individual points, known as pixels. Thus if we have an image with a resolution of 100 x 100, each pixel may be considered to be a point at a specific coordinate. The origin (0, 0) is usually taken to be at the lower left corner of the image.

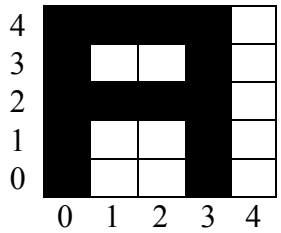
Colors may be produced by one of two methods[1].

For systems that produce color by the transmission of light (e.g., LCD panels, digital cameras, monitors, etc) the ***additive color*** approach is used. The ***additive primary*** colors are red (R), green (G), and blue (B). Adding R and G light makes yellow (Y). Similarly, G + B = cyan (C) and R + B = magenta (M). Combining all three additive primaries makes white, while the absence of any light results in black. This system is known as the ***RGB format***, in which we specify the amount of red, green and blue light. If we use a scale of 0-255 for each color, then the color of an individual ***pixel***, may be stored in 3 bytes (known as 24-bit color) and allows for 256×256×256 ≈ 16 million distinct colors.

For systems that produce color by the reflection of light (e.g., prints, film, etc.) the ***subtractive color*** approach is used. The picture is illuminated by white light (which contains all possible colors) and the dye or pigment removes the unwanted color, reflecting only the colors that are observed. The ***subtractive primaries*** are cyan (C), magenta (M) and yellow (Y). Theoretically, equal amounts of cyan, magenta and yellow, C+M+Y, should absorb all light producing black. In practice this is difficult to achieve so that a fourth pigment, black (K – in order not to confuse this with blue), is added. This system is known as the ***CMYK format***. The CYMK format represents colors on a scale from 0.0 to 1.0.

Ordinarily, editing of an image on a computer is done using RGB, which the computer converts to CMYK for printing. The conversion from RGB to CMYK can be easily done, but we will not consider it here.

As an example suppose we have a small image representing the letter A with a resolution of 5 x 5:



[1] http://en.wikipedia.org/wiki/RGB

We can then represent this image with 5 one-dimensional arrays: one for the x coordinate, one for the y coordinate and three to represent the RGB values of each pixel at location (x, y). Since all of the pixels making up the letter are black, the RGB value would be 0 for the black pixel and 255 for the white pixels. Thus for the example above we would have:

X	Y	R	G	B
0	0	0	0	0
0	1	255	255	255
0	2	255	255	255
0	3	0	0	0
0	4	255	255	255
1	0	0	0	0
1	1	255	255	255
1	2	255	255	255
1	3	0	0	0
1	4	255	255	255
2	0	0	0	0
2	1	0	0	0
2	2	0	0	0
2	3	0	0	0
2	4	255	255	255
3	0	0	0	0
3	1	255	255	255
3	2	255	255	255
3	3	0	0	0
3	4	255	255	255
4	0	0	0	0
4	1	0	0	0
4	2	0	0	0
4	3	0	0	0
4	4	255	255	255

These 5 arrays would be stored in a data file representing the image. In order to accommodate different size images, they would be preceded by a header of two integers representing the horizontal and vertical resolution. For the example above the header would be:

5 5

Drawing images in Java

In order to actually draw the image we will be using the StdDraw class provided by Princeton University.

Getting started. To use this class, you must install StdLib into NetBeans. Follow the following instructions given here:

http://www.sci.brooklyn.cuny.edu/~goetz/java/

To test that you have done this correctly, create a project named *TestStdDraw* and then type the following short program:

```
public class TestStdDraw {
    public static void main(String[] args) {
        StdDraw.setPenRadius(0.05);
        StdDraw.setPenColor(0, 0, 255);
        StdDraw.point(0.5, 0.5);
        StdDraw.setPenColor(255,0, 255);
        StdDraw.line(0.2, 0.2, 0.8, 0.2);
    }
}
```

Do not forget to add the *StdLib* library to your project and import:

```
import edu.princeton.cs.introcs.*;
```

If you compile and execute the program, you should see a window appear with a thick red line and a blue point. This program illustrates the two main types of methods in standard drawing—methods that draw geometric shapes and methods that control drawing parameters. The methods *StdDraw.line*() and *StdDraw.point*() draw lines and points; the methods *StdDraw.setPenRadius*() and *StdDraw.setPenColor*() control the line thickness and color.

You close the window and stop your program by choosing the small *stop* button on your Netbeans output window.

You can save the image by using the *File/Save* menu item in the displayed window. Choose an extension of either *.png* or *.jpg* for your filename.

You can draw a point with the following method:

```
StdDraw.point(double x, double y)
```

You can set the color of the pen by using the method:

```
StdDraw.setPenColor(int r ,int g, int b)
```

Where r, g, and b range from 0 to 255. A list of the RGB values for any color may be found by searching Google for "color code for *color* in rgb", where *color* is the color you are looking for.

The complete API for the StdDraw class may be found here:

http://siever.info/cs141/StdDraw.html

You will not need these, but you can create some fun programs by studying the document at:

https://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html

Homework:

You will be reading an image file, *mage.txt* that contains the data as described above**.**

1. First read in two integers representing the x and y resolution of the image. Then read the subsequent values into 5 one-dimensional arrays, using a method

   ```
   readImage(int[] x, int[] y, int[] r, int[] g, int[] b,
                                int xResolution, int yResolution)
   ```

2. As you read each point, use a Boolean `method`

   ```
   validateXY(int x, int y, int xResolution, int yResolution)
   ```

   that determines that the (*x*, *y*) coordinates of the point lies within the specified resolution of the header.

3. Write a method

   ```
   validateRGB(int r, int g, int b)
   ```

   that determines whether the three values are valid**.** If the values are valid it returns **`true`**; if the values are invalid it returns **`false`**. Discard any invalid points. (These may represent a damaged image.)

4. Write a method

   ```
   plotImage(int[] x, int[] y, int[] r, int[] g, int[] b,
                                          int xMax, int yMax)
   ```

   In this method, first set the *scale* and *penRadius*. Set the scale of the image to the x and y resolution of the image and allow for a small margin using the methods:

   ```
   StdDraw.setXscale(0 - 1, xMax + 1);
   StdDraw.setYscale(0 - 1, yMax + 1);
   StdDraw.setPenRadius(0.1);
   ```

   Looping through all elements of the five arrays you can then process each data pixel and use the StdDraw methods to plot the point.

Save the image that results as an *image.png* file and print it out. Be sure to comment your program, use meaningful variables and use the structured programming techniques you have learned in class.