**Image Processing**

Java Project Report

**Project Overview:-**

The project is a java program which can read, process and write image files of various image formats, ex – PNG, JPEG, BMP,etc.

The project is split into two packages, Image.class and Pixel.class.

**How it works:-**

**External libraries used:-**

* java.io.File – file handling
* Java.io.IOException – file exception handling
* java.awt.image.BufferedImage – work with BufferedImages in RAM
* java.imageio.ImageIO – image read/write

**Image.class** handles the process of reading/writing image files and the various filters and effects which can be applied to the image.

**Pixel.class** handles the different operations of fetching pixel data from the image(including its 32-bit color data, coordinates, etc) and editing pixel color data.

Using basic functions from **Pixel.class** in **Image.class**, the program has different filters defined as functions of the Image object, ex-

* **filterInvert( ):** generates the color negative of the image
* **filterRed( ):** removes all colors except for the shades of red
* **greyscale( ):** converts the image to greyscale
* **selectImage( ):** crops the image given the area to crop as parameters

**Class Diagram:-**

**Image Class:-**

File file

BufferedImage Image

height, width

*readImage()*

*writeImage()*

*FflterInvert()*

*filterInvertAlpha()*

*greyscale()*

*selectRegion()*

*filterRed()*

*filterGreen()*

*filterBlue()*

*flip()*

**Pixel Class:-**

Int p,a,r,g,b

Int x,y

*getPixelData()*

*seperateChannels()*

*combineChannels()*

**How Pixel color data is modified:-**

Each pixel in the image is a ***32 bit*** integer. The highest order 8 bits hold the ***alpha*** or transparency value, the next ***24 bits*** hold ***Red***, ***Green***, ***Blue*** values from ***0 to 0xFF*** or ***0 to 255*** in decimal.

First, we fetch pixel data using the **java.image.BufferedImage** library function getRGB(BufferdImage image,int x,int y) and store it in Pixel.p

Now, the ***separateChannels( )*** function separates the alpha, r g and b channels:-

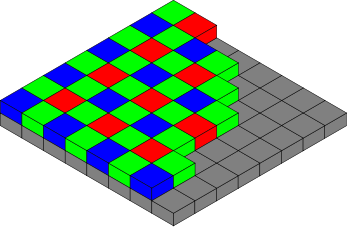
* a = (p>>24) & 0xff;
* r = (p>>16) & 0xff;
* g = (p>>8) & 0xff;
* b = (p) & 0xff;

We can now modify the data in a, r, g, and b to change the way the pixel looks and write it to the image with the ***setPixel( )*** function.

***Examples:-***

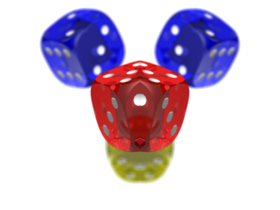


**filterInvert( )**

******

**selectRegion(75,75,150,150)**

******

******

**flip( )**

******

**greyscale( )**

***Conclusion:-***

The Image class and Pixel class together provide a lot of functionality for processing Image files and this project can be further expanded to include more filters, cropping techniques, editing methods and a GUI to make a simple Photoshop like application.