INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES ´
DE MONTERREY

DEPARTAMENTO DE CIENCIAS E INGENIERÍA


# GRAYSCALE IMAGE CONVERTER

MAX BURKLE GOYA

PROGRAMMING LANGUAGES FINAL PROJECT

# Context

## Digital Images

We currently live in the digital world where an extremely high percentage of the population interacts digitally in one way or another every single day. With this digital advancement in the world and society images have also developed to be this way. Images are now rarely stored and created as a physical form. With the invention of digital cameras and smartphones the digitalization of images has become more common. With the amount of software and applications that exist nowadays, image processing has also become far more popular with the use of image filters. This can be seen in social media where an enormous amount of posted images have received some kind of filtering.

A commonly used filter in images is grayscale, in which lies the focus of this project.

## Grayscale

"*Grayscale is the collection or the range of monochromic shades, ranging from pure white on the lightest end to pure black on the opposite end. Grayscale only contains luminance (brightness) information and no color information; that is why maximum luminance is white and zero luminance is black; everything in between is a shade of gray. That is why grayscale images contain only shades of gray and no color.*"

Digital images' pixels are composed of 3 channels, the red channel, green channel and blue channel. When an image is converted to grayscale the color information is removed and luminance is kept. Luminance then shows how "bright" or how white the pixel should be and then that is how we get the gray.

For image processing grayscale images function better than colored images. First of all grayscale images are "lighter" than colored ones, meaning that they contain less information.

Specifically talking about image recognition, most systems use grayscale images principally because grayscale simplifies the algorithm and reduces computational requirements.

## Converting to Grayscale

There exist several different algorithms of grayscale transformation. Most of them are fairly inexpensive and don't require much computational power. For these project the following method was used:

$$\mathcal{G}_{Luminance} \leftarrow 0.3R + 0.59G + 0.11B.$$

*"Luminance does not try to match the logarithmic nature of human brightness perception, but this is achieved to an extent with subsequent gamma correction. Luminance is the standard algorithm used by image processing software"*

Grayscale conversion may be inexpensive but as software engineers we look for maximization of resources and time of the computer when running a program. For this project java was used with the implementation of threads when processing the image. The image is divided into 10 blocks of equal sizes. The size depends on the size of the image and then each thread is in charge of converting each pixel into its gray counterpart. Following the previous equation.

# Solution

## Programming

The solution is focused on parallel or concurrent programming, used mainly for increasing software speed by having multiple computer cores working on the same operation.

## Java

The programming language used for the solution was Java. The Java platform is designed from the ground up to support concurrent programming, with basic concurrency support in the Java programming language and the Java class libraries.With the focus on parallel programming the class Thread was extended in the project making the use of threads possible in the project.

As for the implementation of the project the Eclipse window builder was used to make the creation of a runnable program that uses a window for the user.

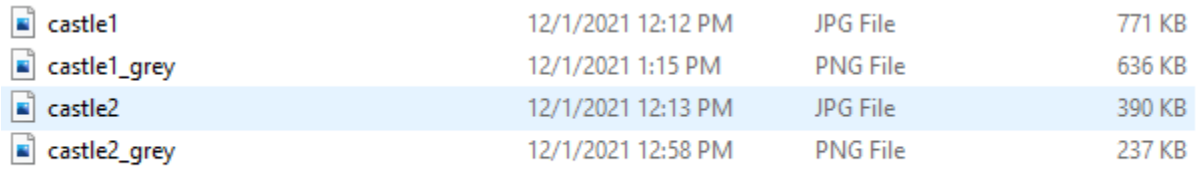
```
public static void convertImage(Shell shell, String path, int brightness) throws IOException {

    BufferedImage img = ImageIO.read(new File(path + ".jpg"));
    int height = img.getHeight();
    int width = img.getWidth();
    ImgConverter threads[];

    int block = (width * height) / 10;
    threads = new ImgConverter[10];

    for (int i = 0; i < threads.length; i++) {
        if (i != threads.length - 1) {
            threads[i] = new ImgConverter(img, brightness, width, height,
                                (i * block), ((i + 1) * block));
        } else {
            threads[i] = new ImgConverter(img, brightness, width, height,
                                (i * block), (width * height));
        }
    }

    for (int i = 0; i < threads.length; i++) {
        threads[i].start();
    }
    for (int i = 0; i < threads.length; i++) {
        try {
            threads[i].join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    ImageIO.write(img, "png", new File(path + "_grey.png"));
    MessageDialog.openConfirm(shell,"Completed","File has been succesfully converted.");
}
```

The first called method is the `convertImage()` method used when information is inputted in the window and the continue button is pressed. In this method the image is buffered and saved as a variable, and also information like the width and height of it are extracted and stored. Threads are also created in this method, an array is used to create 10 threads which will then be in charge of the image processing. Parameters for the creation of the threads include: the buffered image, width and height of the image, the brightness percentage for the grayed image, and the assigned blocks each thread will be working on.

```java
public void run() {
    int index, row, col;

    for (index = start; index < end; index++) {
        row = index / width;
        col = index % width;
        greyPixel(brightness, row, col);
    }
}

private void greyPixel(double brightness, int row, int col) {
    int rgb = img.getRGB(col, row);

    int red = rgb & 0xFF;
    int green = (rgb >> 8) & 0xFF;
    int blue = (rgb >> 16) & 0xFF;

    float L = (float) (0.3 * (float) red + 0.59 * (float) green + 0.11 * (float) blue);

    int bright = (int) Math.ceil(brightness/100 * 255);
    int color;
    color = bright * (int) L / 255;                 // R color
    color = (color << 8) | bright * (int) L / 255;  // G color
    color = (color << 8) | bright * (int) L / 255;  // B color

    img.setRGB(col, row, color);
}
```

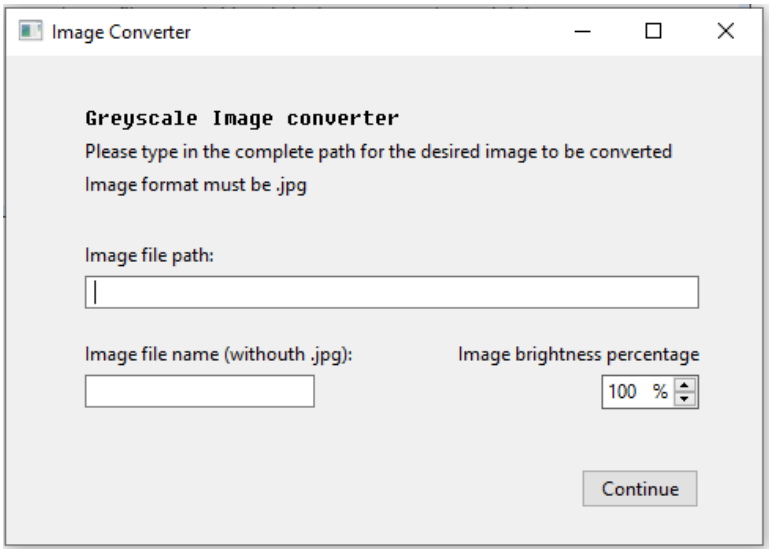
As for the methods the Threads will be in charge of is the `greyPixel()` method for the entirety of the assigned block. For each pixel the method will be called using the row and column of each pixel in the image. The rgb is extracted as information and separated using bit level operations into their three red, green and blue components. Each value then is multiplied by its constant to get the luminance value of the pixel. Then incorporating these values again to a single variable and setting the pixel for the new image. When all threads are done and the entire new image is complete then a new file is created with the new grayscale image.

The original file is conserved as a colored image and a new file with a similar name is saved in the folder.



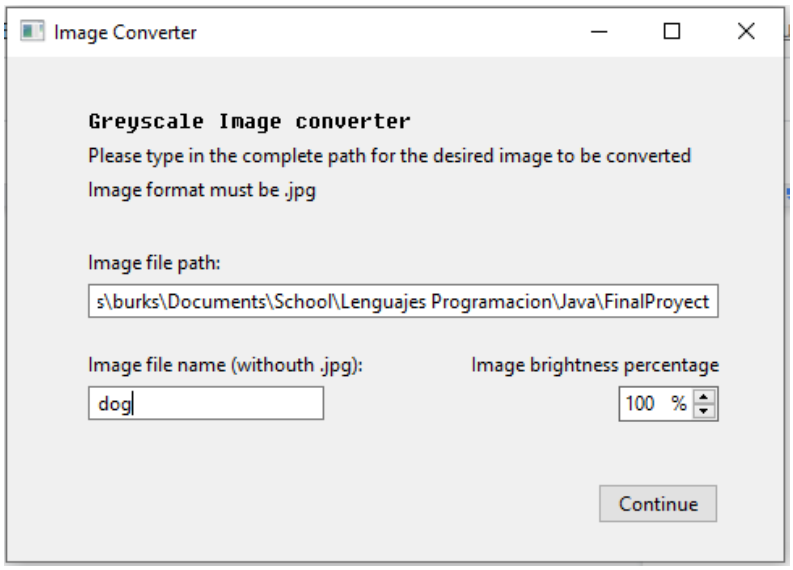
# Results

For the finished solution when the program is run a window is automatically opened:

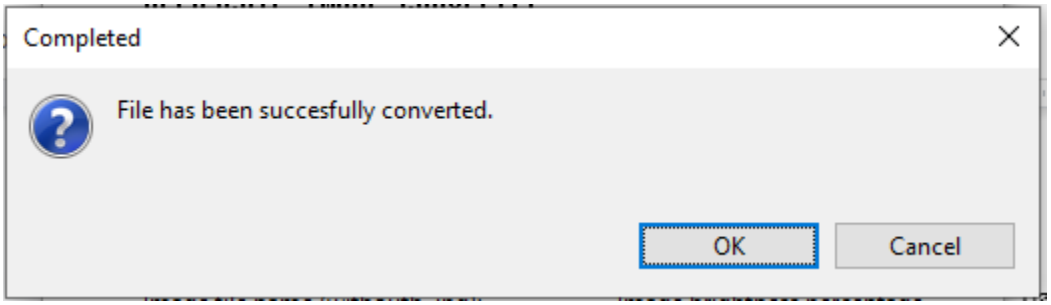



In the window the path of the image that wants to be converted must be copied and pasted. But only the path of the file not including the file. In the next text input the file name must be introduced as well but without file extension. Finally a brightness percentage may be set for the grayscale image, by default the value is set at 100% brightness but can be lowered.

Once all the information is complete then the continue button can be pressed.



If all information is correct and no error occurs then a completed notification will pop up.




With this notification the gray image will be generated in the same location as the original image. Dismissing this notification will then close the program automatically.

Converted images:

Images can have different sizes. A lot of different sizes were tested and yielded positive results, the bigger the image the longer the program can take to process it.

Also the different brightness percentages were tested yielding again positive results. The previous images are: original, 100% brightness, 75% brightness and 25% brightness. The image is still distinguishable but the luminance is reduced if the percentage is reduced.

# Conclusions

The program may be simple but it is a good solution for generating a grayscale image. The usage of threads speeds up the program substantially and this is observed much easier with larger images of 4k resolution. Also this program can also be fitted for the use of colored filters, blurring of images and even convolution of images.

# References


"GRAYSCALE & COLOR IMAGE PROCESSING." *Grayscale & Color Image Processing*, http://www.science.smith.edu/~nhowe/370/Assign/gray.html.

Kanan, Christopher, and Garrison W. Cottrell. "Color-to-Grayscale: Does the Method Matter in Image Recognition?" *PLOS ONE*, Public Library of Science, https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0029740.

"Lesson: Concurrency." *Lesson: Concurrency (The Java™ Tutorials > Essential Java Classes)*, https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html.

Techopedia. "What Is Grayscale? - Definition from Techopedia." *Techopedia.com*, Techopedia, 3 Aug. 2017, https://www.techopedia.com/definition/7468/grayscale.