**Lab 3 Report**

Javier Ramirez Moyano
Perception & Multimedia Computing

**Write-up Questions**

**Part A**

2.- Run your sketch and step far enough back from the screen that the pixelation is no longer obvious. What colour does the square appear to be?

It appears to be orange.

3.- Make a guess at the RGB values of the mixture colour in part A.2. Add some code to your project to draw a square of that colour next to your checkerboard. Was your guess right? By trial and error, find the RGB colour values of the single colour that best matches the mixture. Try to explain any discrepancy.

My guess for the mixture of colors red, RGB(255, 0, 0), and yellow, RGB(255, 255, 0), was RGB(255, 127, 0) because on both red and yellow the R value is 255 and the G value is 0 and 255 respectively, therefore I thought the value of G would be in the middle of 0 and 255. I think my guess was right, as the color obtained is almost identical to the orange of the mixture color.

After trying out a few other values of R and G within the orange square, I haven't been able to find a better match than RGB(255, 127, 0). I think this is indeed the correct RGB code for the combination of red and yellow.

4.- Repeat this with some other colours, at least two more. Make a comma-separated values table of the colour values in the mixture, and the closest match in RGB colour space to the mixture that you find: your table should have headings R1, G1, B1 (RGB values of the first colour), R2, G2, B2 (RGB values of the second colour, your guess), RM, GM, BM (RGB values of the matching colour); use a spreadsheet application, or a text editor.

| **Color** | **R1** | **G1** | **B1** | **R2** | **G2** | **B2** | **RM** | **GM** | **BM** |
|---|---|---|---|---|---|---|---|---|---|
| **1st** | 210 | 130 | 190 | 105 | 240 | 125 | 157 | 185 | 137 |
| **2nd** | 48 | 200 | 190 | 205 | 10 | 195 | 126 | 105 | 192 |

**Part B**

3.- Try using the HSB to RGB convert function provided by your programming framework. Do you get the same result? If not, can you try to figure out why?

In order to carry out this test, I did the following. I changed the *colorMode()* to HSB from the standard RGB. After this, I drew a square of a color HSB(251, 80, 32), and used the *get()* function to retrieve the RGB value of a random pixel within that square. The value returned was RGB(28, 16, 82), which is the same returned by my converting function with the same HSB values input.

**Part C**

2.- What do you observe?

After staring at the dot on the inverted version of the image for a while and then switching to the greyscale one, it seemed for a very short time like the grayscale image was colored, as if it was the normal one.

3.- Did you use the default invert filter function? Can you use other methods to create the same effect? Try changing the hue directly (invert hue). What do you need to do with saturation and lightness to have the same or maybe even a stronger effect? Write a paragraph (up to half an A4 page) explaining your chosen method.

Yes, I did use the built-in *filter()* function for both the grayscale and the invert filters, but it is not the only option to create these effects. Instead of using this function, I could have written my own methods. These methods would process the photo pixel by pixel, changing their RGB value for a grayscale value or the inverted color. To convert the RGB values to a grayscale value, we could simply average the RGB values: (R+G+B) / 3. To obtain the inverted color of a pixel, we could normalize the RGB values from 0 to 1 and apply the formula:

newR = 1 - R
newG = 1 - G
newB = 1 - B

I have tried the method above and it achieves the invert filter perfectly.

In order to change the hue of the colors of the photo, I first changed the color mode to HSB and then I made my program go through every pixel in the picture using a nested for loop and change their hue. The new hue (invert hue) is calculated by another function that adds or subtracts 180º to the hue, depending on the starting value. In order to achieve a closer result to the invert filter,

we must also change the saturation and lightness, so after much trial and error, I achieved the closer result by multiplying the saturation by 1.2 and the lightness by 0.8. The result still isn't exactly the inverted image, but it is somewhat close.