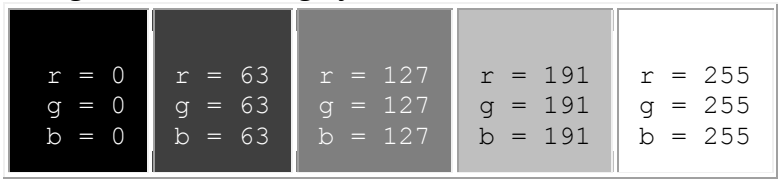
CS 111 – Program Design I, Spring 2018

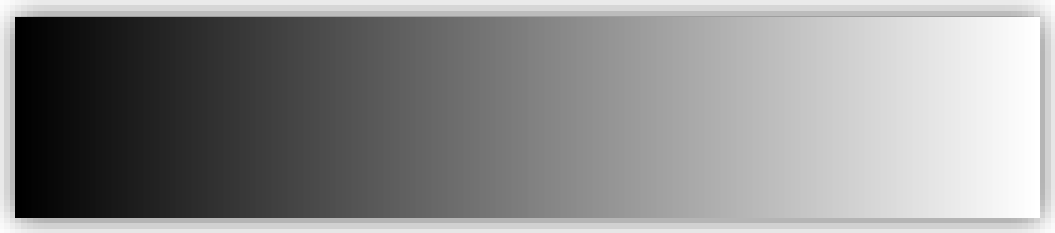
Lab 6 – Make BlueScale Images

Images Using a GrayScale

A black and white picture uses various shades of gray. These shades of grey all have the same amount of red, green and blue color. The higher the color value, the lighter the shade of gray is. The lower the color value, the darker the shade of gray is. For example, look at the following squares for various shades of gray. The amount of red, green and blue for the color is displayed as text in each square. Note that the amount of red, green and blue is the same in each square. Also note that the larger the amount of color, the lighter the color of gray.



To see the entire range of gray scale images, refer to the picture below. The left side of the picture has colors set to black (r = 0, g = 0, b=0). The right side of the picture has colors set to white (r=255, g=255, b=255). The middle colors shift from black to white based on X coordinate of the pixel.



Changing the color in a pixel

The color in a pixel can be changed by using the **setRed()**, **setGreen()** and **setBlue()** functions in the JES library.

Determining the amount of gray

One method to determine which grayscale color to use for a pixel when creating a black and white picture from a color picture is to average the amount of red, green and blue color the corresponding pixel in the colored picture. This method works fairly well and is fairly easy to understand and use. This was the code written in class for lect0220d.py. This way uses the intensity of each color to determine the grayscale value.

However, this assumes that the human eye sees the "brightness" or "luminance" of red, green and blue equally well. However, this is not the case. The human eye perceives green as brighter than red and it perceives red as brighter than blue. **So a standard average calculation is NOT the best way to determine the grayscale amount.**

A much better method is to take a weighted average of the amount of red, green and blue color instead of an evenly-weighted average. A weighted average would give more weight to certain values and less weight to other values. For example, a weighted average will be used when determining the grade for this course. Exams have a higher weight (i.e. more impact on the final grade), while lab assignments have a lower weight (i.e. less impact on the final grade).

For determining the amount of brightness for the grayscale color, the human eye needs to use a weighted average. The original amount of green used should account for almost twice as much as the 33.3% which would be used in a normal averaging. The original amount of red used should be just a bit less than the 33.3% which would be used in a normal average. The original amount of blue should only account for a third of the expected 33.3% which would be used in a normal averaging. Below us the percent amounts for each color that should be used for this lab. These values were determined according to some research done on luminance.

- Red : 29.9%
- Green : 58.7%
- Blue : 11.4%

Lab Assignment 6

Due: Wednesday, 2/28/2017 by 11:59 pm

Write three python functions that will do the following in a program file called NETIDlab6.py.

1. **Function 1: main( )**
    - The python function is to be named: main( )
    - Print out your name and your net-id
    - Allow the user to select a picture from a file stored on the local machine
    - Create a picture variable to hold the result of the JES makePicture( ) function
    - Call the function makeBlackBlue( ) with the picture given as the parameter
    - Call the function makeBlueWhite( ) with the picture given as the parameter
    - Have one of the above two calls commented out so only one is executed. Change which line that is commented out to change which function will get called when main( ) is executed. This idea is shown by the code below.

```
makeBlackBlue ( pict )
#makeBlueWhite ( pict )
```

    - Display the modified image
    - Save the modified image to a file on the local machine

2. **Function 2: Make Black And Blue**
    - The python function is to be named: makeBlackBlue( )
    - This function will take a picture as its only parameter
    - Change the picture to a "black-to-blue scale" image. This is done in a similar manner as the creation of a grayscale image, except the values for red and green are kept at zero. Only the blue value for each pixel will range from 0 to 255 based on the luminance of the original pixel. Thus, where the grayscale image is black, the blue scale image will also be black.

Where the grayscale image is white, the blue scale image will be blue. Where the grayscale image is gray, this image will range from black to blue (the middle values will be a shade of dark blue). To see the variations of color from black to blue, check out the following picture:

Note: **This is NOT done by only setting the red and green color values to 0 and leaving the blue value unmodified. You must modify all three color values in every pixel! (The red and green values get set to 0, while the blue value gets set to the weighted average of the original red, green and blue values.)**

3. **Function 3: Make Blue And White**
    - The Java program is to be named: makeBlueWhite ( )
    - This function will take a picture as its only parameter
    - Change the image to a version of a blue scale image. This is done in a similar manner as the creation of a grayscale image, but with a twist. Where the grayscale image is black, this image will be blue. Where the grayscale image is white, this image will also be white. Where the grayscale image is gray, this image will range from blue to white (the middle values will be light blue). Look at the table below to determine which color value need to change and which one need to remain constant. Also check out the following picture to see the colors range as its changes from blue to white.

The blue value should be set to the maximum value, while the red and green values are modified to reflect the luminance/grayscale amount.

Note: **This is NOT done by only setting the blue color value to 255 and leaving the red and green color values unmodified. You must modify all three color values in every pixel! (The blue values gets set to 255 while the red and green values get set to the weighted average of the original red, green and blue values.)**

4. You must write your program using good programming style which includes:
    - Good variable names
    - in-line commenting

- header block commenting for the program and each method written
  Be sure to include the following with the header block comment for the program.
  - your name
  - day and time of your CS 111 lab section (i.e. Monday at 3:00)
  - A description of the project.
- proper indentation of program statements
- use of blank lines to separate blocks of code.

5. You are to submit your python program electronically via the assignment link in Blackboard for Lab 6.

   Also, if you have any comments about your program, please write it down in the header block comment of the .py file; please do NOT write in the "Comments" field on the submission page (this will go into a different file and will be easily missed).

The discussion below is to help understand the differences between a black&white picture versus a black&blue versus a green&white picture. Using the below information, we can determine some of the before and after colors. The "lum" value below is the average for the r, g and b values listed. Notice how the lum value in the first row is used for the blue values in the 2nd and 3rd rows and for the red and green values in the 3rd row.

original color values (note: the "lum" value is the important value)	r = 0 g = 0 b = 0 lum = 0	r = 9 g = 150 b = 30 lum = 63	r = 255 g = 0 b = 126 lum = 127	r = 150 g = 168 b = 255 lum = 191	r = 255 g = 255 b = 255 lum = 255
makeBlackWhite()	r = 0 g = 0 b = 0	r = 63 g = 63 b = 63	r = 127 g = 127 b = 127	r = 191 g = 191 b = 191	r = 255 g = 255 b = 255
makeBlackBlue()	r = 0 g = 0 b = 0	r = 0 g = 0 b = 63	r = 0 g = 0 b = 127	r = 0 g = 0 b = 191	r = 0 g = 0 b = 255
makeBlueWhite()	r = 0 g = 0 b = 255	r = 63 g = 63 b = 255	r = 127 g = 127 b = 255	r = 191 g = 191 b = 255	r = 255 g = 255 b = 255

The following are some images created based on the ideas in this lab.

The following image is the original full color picture.

The following image is the black and white version of the original image.

The following image is the black and blue version of the original image.

The following image is the blue and white version of the original image.