**Programming Assignment #4 (50 pts)**

Assigned: Wednesday, October 16

Due: Wednesday, October 26, 2016

**Assignment 4A – Digital Art**

**ASSIGNMENT 4A**

According to Wikipedia, digital art is a general term for a range of artistic works and practices that use digital technology as an essential part of the creative and/or presentation process. Your assignment is to create an original piece of digital art based upon a color picture of yourself (or someone or something close to you like a pet or your car, for instance). Using the algorithms described below, you should produce a gray scale and a negative of the picture. Based upon what you learn as you create the gray scale and negative versions of the picture, you should write method myArtwork which takes your picture and modifies it in some artistic way. You could, for example, introduce a random number generator that replaces some pixel value with a random number. Alternatively, you might choose to mesh two (or more) pictures together. Another idea might be to create a digital mirror image of your picture. You are free to be creative and to think artistically.

By the due date, you should print a copy of your artwork on the color printer in the Taylor Computer Lab on a size of paper other than 8.5 X 11 (your choice, it just has to be different than the default). You must title your work, and sign it. You must also print a copy of the code that produced the image and be prepared to explain how your code produced the image. All of your work should have a professional look and feel (i.e. no wrinkles, smudges, folds, etc.)

PPMPicture

- int pixels []//array of pixels

- firstLine: String

- secondLine: String

- thirdLine: String

- x, y: int

During the following week, we will schedule a one hour art show where you can stand beside your work and answer questions about it.

+ Picture (filename:String)

//filename is a PPM file

+ save (fileName:String):void

//save the pixel array to a file

+ setToGrayScale ():void

//turn the pic to black and white

+ setToNegative ():void

//turn a gray scale image to //negative

+ myArtwork ():void

//Create an original piece of art

**SOLUTION**

The algorithm for turning a color pixel into a black and white pixel may be found in the references below.

From the black and white photograph, you should also produce a negative of the original image.

**ADDITIONAL INFORMATION**

You will be provided with the PPMPicture class whose API is shown to the right. You can use the PPMPicture constructor, along with the name of your PPM file to read the RGB pixels of your PPM file into the array named pixels.

NOTE: Make sure you place your PPM file at the highest level of your project folder.

You will need to write the code for setToGrayScale, setToNegative, and myArtwork methods. You should read the name of your photo into a dialog box at the beginning of your program. You should be prepared to run your program in lab on Thursday, October 30th for full credit.

**References:**

Color chart: http://www.tayloredmktg.com/rgb/

Algorithm help: http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

**Programming Assignment #4B (25 pts)**

The Tortoise and the Hare

**Description\***

In this program, you are asked to recreate the classic race of the Tortoise and the Hare. The race begins with both contestants at the first of 70 squares, each square representing a position along the race course. The finish line is the seventieth square. (Hint: consider the concept of an array.) The first contestant to reach or pass the finish line is declared to be the winner. Each contestant performs in a particular fashion, moving along the squares of the race course, sometimes forward and sometimes backward. See the chart below.

|  |  |  |
| --- | --- | --- |
| TORTOISE MOVEMENTS | PERCENTAGE OF MOVEMENTS | ACTUAL MOVEMENT |
| Fast Plod | 50% | 3 Squares to the Right |
| Slip | 20% | 6 Squares to the Left |
| Slow Plod | 30% | 1 Square to the Right |
| HARE MOVEMENTS | PERCENTAGE OF MOVEMENTS | ACTUAL MOVEMENT |
| Sleep | 20% | No movement |
| Big Hop | 20% | 9 Squares to the Right |
| Big Slip | 10% | 12 Squares to the Left |
| Small Hop | 30% | 1 Square to the Right |
| Small Slip | 20% | 2 Squares to the Left |

The movements should be determined through the use of randomly generated numbers. This can be done by producing a random integer in the range 1 through 10. For example, the tortoise would perform a “fast plod” if his movement integer is in the range 1 through 5 and a slip when the number is in the range 6 through 7 and so on. A similar technique could be used to move the hare. Because of the uniqueness of random numbers, it could be possible that an animal that could slip past the first square. If this happens, simply return the animal to the first square.

The race itself should be a series of iterations. With each iteration, the position of each animal should be adjusted according to the chart above. Output should consist of a 70-position line showing the letter T in the position of the tortoise and the letter H in the position of the hare. If the participants of the race land on the same square, display an exclamation point in that position, indicating surprise that the contestants are tied. All other positions should have the underscore character.

After each iteration, decide if the contest should continue, based on whether one of the animals has reached or passed the final square. When this occurs, display the name of the winner or, in rare cases, indicate a tie.

MCj02152940000[1]MCj02153060000[1]