**Lab 3**: PPM Image Files-Lab Report

**Name:** Andrew Marima Mambondiumwe ([mambondiumwem@berea.edu](mailto:mambondiumwem@berea.edu))

**Purpose:** to work with Image PPM files

# **INITIAL DESIGN PLAN**

**Design for the PPM Image files**

* Open a PPM image file that the user inputs
* Converts the PPM image to red.
* Converts the PPM image to grey scale
* Flips the PPM image horizontally
* Rotates the PPM image clockwise
* Zooms or enlarges the PPM image by an amount selected by the user.
* Turns the PPM image into a negative of that image.
* Allows the user to select whether or not to implement more than one of the above actions to the same PPM image
* Displays the image in a PPM viewer after all the actions or changes have taken place

**I will complete the task with the following methods in the PPM class:**

* PPM\_greyscale () - this method should go through each pixel in the rows and update them all to the same greyscale integers, thus changing the PPM Image to greyscale.
* PPM\_flip\_horizontal () - this method reads through the pixels backwards so that they can be displayed in the opposite order.
* PPM\_rotateclockwise () – this method steps through the list one pixel at a time and adds the original pixel list to the pixel list to get a get a rotated PPM image.
* zoom\_PPM (self, zoom\_amount) – this method uses user input to update the width and the height of the original PPM image.
* PPM\_negative (self) – this method goes through each pixel in the row list and subtracts the pixel from 255 to get the new pixel of the negative. It then displays the negative of the PPM image.
* Main () function – this function asks the user to input in a file name. It will then prompt the user to choose an action that is to be done on the image, it gives the user the options to choose from. It allows the user to also have the option of doing multiple actions to the same PPM image. Imports and uses the methods from the PPM class to do the actions.

# **SUMMARY**

I worked through this lab assignment, step by step as outlined in my design plan. Breaking the task into small tasks, method by method was critical in completing the assignment successfully. The final code was exactly like I had designed and intended it to be like. The only slight modification was to put the user-input for the zoom amount together with the zoom method so that when the zoom method is called, the user will be prompted for the zoom amount first.

Overall I spent like 10 hours (2 hours a day) from the design to the implementation of the program and making it to be a workable program.

# **IMPLEMENTATIONS**

I was able to successfully achieve my initial design which is written below, I was able to:

1. Open a PPM image file that the user inputs
2. Converts the PPM image to red.
3. Converts the PPM image to grey scale
4. Flips the PPM image horizontally
5. Rotates the PPM image clockwise
6. Zooms or enlarges the PPM image by an amount selected by the user.
7. Turns the PPM image into a negative of that image.
8. Allows the user to select whether or not to implement more than one of the above actions to the same PPM image
9. Displays the image in a PPM viewer after all the actions or changes have taken place

# **TESTING:**

# I primarily used two ppm files to test my code, which are the bc-sign.ppm and the bc-flowers.ppm. I tested each method on each of these files, one method at a time and all the methods were working for both of the two ppm images that I used in testing.

**FILES:**

The following were the files used:

* bc-sign.ppm
* bc-flowers.ppm

# **ERRORS:**

The errors that I experienced during this lab assignment included syntax errors and run time errors as I pondered and tried out several ways that I could structure the code so that it would work.

**COMMENTS:**

The difficult methods were the two additional methods that I had to create myself, which were the zoom action and the turn into a negative action. With a bit of help from a T.A, It then began making sense on how I should go about tackling the problem. I then managed to fix all the bugs and get the code working. Overall, the lab was quite challenging but it improved my knowledge and was fun at the same time.

**Updated CRC Card for the PPM class**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Class name:** | **PPM** | | **Class Methods:** | **Class Collaborations (other classes, etc):** | | * **\_\_init\_\_ ()** initializer/constructor for the class, opening a provided PPM-P3 file and setting all member attributes. * **PPM\_makeoutputfiles ()**given self.inasciifile, sets self.ascii and creates both ascii and binary files for output * **PPM\_partition ()** Given input parameter strng, the string to partition and ch, the character to use as the delimiter returns a triple with all characters before the delimiter, the delimiter itself if present and all of the characters after the delimiter (if any). * **PPM\_clean ()**removes all single line comments, whitespace, and newline characters present in the input parameter string strng. * **PPM\_load ()** takes string input parameter inasciifile as the name of the ASCII PPM-P3 (non-binary) file to load. * **PPM\_makepixellist ()** creates self.pixellist, a nested list of rows of [red, green, blue] pixels from input color\_list which contains an unnested list of strings. * **PPM\_updatefrompixellist()** updates image object data and related files from input pixellist * **PPM\_convert2bin ()** converts PPM-P3 to PPM-P6 using self.pixellist. * **PPM\_set\_title ()** setter for title of display window. * **PPM\_make\_red (**) colorizes current image to red by using self.pixellist. * **PPM\_greyscale ()** 'changes the picture into a grey scale image. * **PPM\_flip\_horizontal ()** flips image horizontally. * **PPM\_rotateclockwise ()** rotates image clockwise. * **zoom\_PPM (amount)** zooms or enlarges PPM image by a figure inputted by the user. * **PPM\_negative ()** this turns the PPM image into the negative of that image. | * **PPM\_set\_up ()** is a helper function which must be called at the beginning of any program which uses the PPM class, but it is not a part of the class. * **PPM\_render ()** is a helper function which renders all PPM images. It is not part of the class. * **PPM\_Exception** is a Python class which enables meaningful error messages on exceptions. | | **Class Data:** | **Class Collaborations (other classes, etc):** | | * **self.root** # provided master Tkinter instance created using helper function PPM\_set\_up() * **self.inasciifile** # string which represents filename is used only for reading the provided PPM-P3 as input * **self.outasciifile** # string of filename of human readable modifications to the PPM file * **self.outbinfile**  # string of binary ppm filename needed for viewing * **self.title**  # used for the title of the display window * **self.magic** # ppm file type is often called the "magic number." It needs to be P3 to be readable * **self.comment** # creates a comment for the PPM file * **self.width** # image width in pixels * **self.height** # image height in pixels * **self.colormax** # should be set to 255 * **self.ascii**  # will store the color intensities in P3 format * **self.pixellist** # will store nested list containing pixel colors * **self.image** # reference to image window * **self.label**  #  used to place image in window | global**tkintertoggle** is needed as global to ensure a single Tkinter instance which is needed to render image | |