CS 211 PA #1: PPM Images

This assignment is a continuation of lab 1. In this homework, you will create a program that will add simple effects to a PPM image.

Prerequisites

Before beginning this assignment, you will need some sort of imaging software that can open PPM-based images. I recommend Ifranview for Windows as it's free and lightweight.

PPM Image Format

The PPM image format is encoded in human-readable ASCII text. It might be helpful to read over the formal PPM specification document. A PPM document has two pieces: the header and the body. The header is always the top three uncommented lines in the file:

```
P3
4 4
255
```

The first line specifies the type of image that is contained within the file. We will always use the "P3" specification. The second line specifies the number of columns and rows present in the image. In this example, we have a 4x4 image. The final number indicates the maximum value for each red, green, and blue element in the picture. Having a max value of 255 is quite common and is the value that we will always use. Below the header is the body of the PPM file. Each pixel has a red, green, and blue value. For example, the content of our 4x4 image might be:

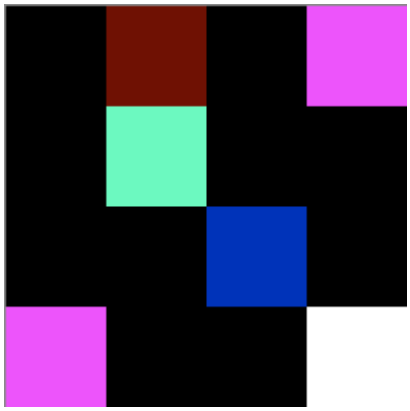
```
0  0  0       100 0  0       0  0  0       255   0 255

0  0  0       0 255 175      0  0  0        0    0  0

0  0  0       0  0  0        0 15 175       0    0  0

255 0 255     0  0  0        0  0  0       255  255 255
```

With these values, the pixel in the first column and row has a RGB value of (0,0,0) and the last column in the first row has an RGB value of (255,0,255). A blown-up image containing these values would look like:



Basic Tasks

Basic tasks require you to manipulate single pixels on a user specified image. Below is a reference value and the images that result from the image processing.

Original Image

Remove Red

Will remove all red (i.e. set red to 0) pixels in the image

Remove Green

Will remove all green pixels in the image

Remove Blue

Will remove all blue pixels in the image

Negate Red

Will negate all red pixels in an image. To negate a color, simply subtract the current pixel's red color value from 255.

Negate Green

Will negate all green pixels in an image.

Negate Blue

Will negate all blue pixels in an image.

Add Random Noise

Will add random noise to the image by randomly selecting a value ranging from -10 to 10 and adding that value to the pixel's current value. If the addition causes the pixel value to be less than 0, then set to 0. Likewise, if the addition causes the pixel's value to be greater than 255, just set the pixel's value to 255. For example, we randomly generate the number -7. Next, we add -7 to each of the current pixel's red, green, and blue values.

High Contrast

Examine each color in a pixel. If the color is greater than half of 255, then max the color out (i.e. set to 255). Otherwise, set the value to 0. For example, performing a high contrast operation on a pixel whose RGB values are 170, 50, 100 would result in an RGB value of 255, 0, 0.

Grayscale

Converts the image to grayscale by setting the value of each color in a given pixel to the average of all three values. For example, if a pixel's RGB was 177, 15, 39, the resulting RGB value would be 77, 77, 77.

Advanced Tasks

Advanced tasks require a bit more thought as they you to manipulate the several pixels at once. Below is the reference image for the advanced tasks:

Flip Horizontally

This will flip the supplied image horizontally. To accomplish, you will need to loop through your image and switch the order of every column. For example, if our picture had four columns numbered 0-3, you would need to swap columns 0 and 3, and columns 1 and 2.

Flip Vertically

This will flip the supplied image vertically. To accomplish, you will need to loop through your image and swap the order of every row. For example, if our picture had four rows numbered 0-3, you would swap row 0 with row 3, and row 1 with row 2.

Rotate 90 Degrees

This one is probably the most conceptually difficult. Essentially, you will need to turn the first row into the last column, the 2nd row into the 2nd to last column, etc. until you turn the last row into the first column. Consider the following grid:

1	1	1	1
2	2	2	2
3	3	3	3

A 90 degree rotation would result in:

3	2	1
3	2	1
3	2	1
3	2	1

NOTE: In order for this to properly work, you are going to have to alter the specified dimensions of your PPM file (2nd line).

Blur

To perform a blur, you will need to do the following:

1. For each pixel in a given row, set the pixel's red value to the average of itself and its horizontally adjacent neighbors (i.e. i + 1, i - 1). For border cases, just take the average of the singular neighbor.
    a. Do the same procedure for the pixel's green and blue values
2. For each pixel in a given column, set the pixel's red value to be the average of itself and its vertically adjacent neighbors (i.e. j + 1, j - 1). For border cases, just take the average of the singular neighbor.
    a. Do the same procedure for the pixel's green and blue values.

Do this 10 times to get an extreme blur like the one depicted in my image.

Pixelate

To pixelate an image, take a reference pixel P and a distance D and make all pixels adjacent to P in a DxD block the same as P. Then, skip over D pixels and repeat the process for a new P. For example, given a D of 2, here is how we would pixelate the following matrix:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	1	3	3
1	1	3	3
9	9	11	11
9	9	11	11

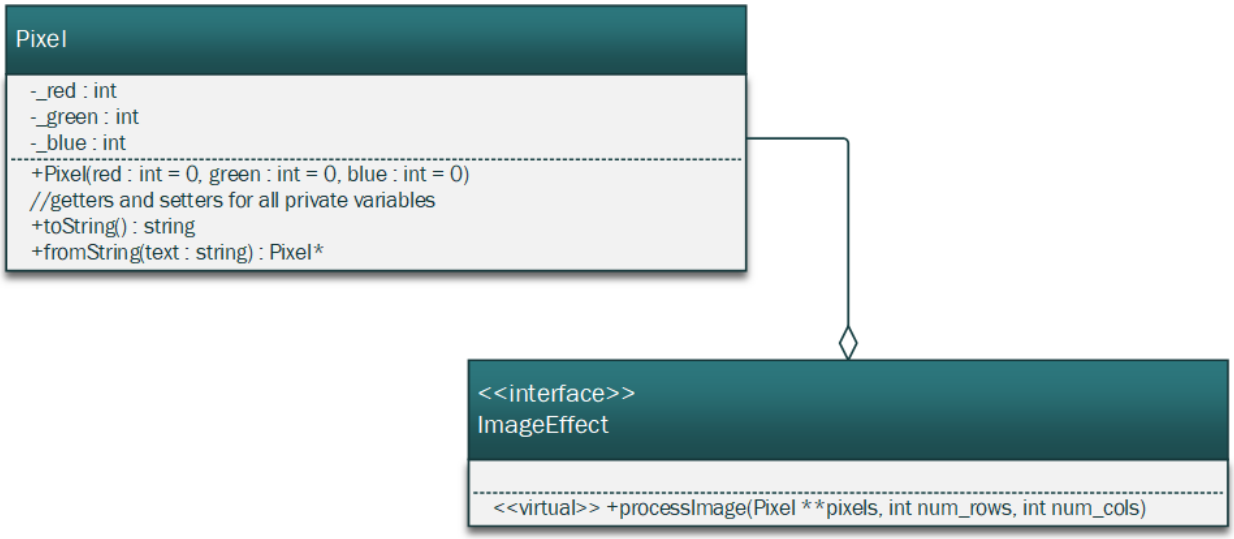
Note that for the image above, I used a distance D of 5. Try playing around with different distance values.

Chaining Effects Together

Note that manipulations can be chained together. Below is the image generated from performing a negate red, negate green, and negate blue operation.

UML Diagram

In order to complete this assignment, you **must** develop and implement and use the two classes developed in lab:



We start with the Pixel class that we developed in Lab. In the center, we have the ImageEffect interface that defines a single pure virtual function called processImage. Each effect described in the previous section should derive itself from ImageEffect, implementing the processImage method in such a way to fulfill its role (e.g. NegateRedEffect will negate all red pixels in the image).

Also note the white diamond going from ImageEffect to Pixel. This is called "aggregation" and is another form of a "has-a" relationship. The main difference between aggregation and composition (black diamond) has to deal with the lifespan of an object. A black diamond means that the object goes out of scope at the same time as the owner class. A white diamond means that the object stays alive even when the owner class goes out of scope. If you think about the lifespan of a Pixel, it is easy to see that a given Pixel will exist for a longer period than the ImageEffect. This makes sense as the Pixel still needs to be written to the output file!

Possible Strategy for Getting Started

- If you're feeling overwhelmed, try implementing one of the more simplistic image effects (e.g. remove red) inside of main. Once you get that working on a PPM file, put the necessary code into its own class and .h file.
- Use our labs as a template. Most of the ideas we covered in lab can be directly applied to the solving of this assignment.

Header Comment, and Formatting

1. Be sure to modify the file header comment at the top of your program to indicate your name, student ID, completion time, and the names of any individuals that you collaborated with on the assignment.
2. Remember to follow the basic coding style guide. A basic list of rules is included with this document.

Reflection Essay

In addition to the programming tasks listed above, your submission must include an essay that reflects on your experiences with this homework. This essay must be at least 350 words long. Note that the focus of this paper should be on your reflection, ***not*** on structure (e.g. introductory paragraph, conclusion, etc.). The essay is graded on content (i.e. it shows deep though) rather than syntax (e.g. spelling) and structure. Below are some prompts that can be used to get you thinking. Feel free to use these or to make up your own.

- Describe a particular struggle that you overcame when working on this programming assignment.
- Conversely, describe an issue with your assignment that you were unable to resolve.
- Provide advice to a future student on how he or she might succeed on this assignment.
- Describe the most fun aspect of the assignment.
- Describe the most challenging aspect of the assignment.
- Describe the most difficult aspect of the assignment to understand.
- Provide any suggestions for improving the assignment in the future.

Checkin

On Monday, January 29, 2018, you must demonstrate a working program that implements all "Basic Task" functionality (see appropriate header section).

Deliverables

You must upload your program and reflection as a ZIP file through Gradescope no later than midnight on Friday, February 2, 2018. Note that your zip file should **ONLY** contain *.cpp and *.h files!

Grading Criteria

Your assignment will be judged by the following criteria:

Reflection essay (5pts)

- Your reflection meets the minimum requirements as specified earlier in this document.

Checkin (10pts)

- Your program meets the above checking requirements

Style (10pts)

- Your project contains good structure and implements the required classes. Your program intelligently uses classes when appropriate and generally conforms to good OOP design (i.e. everything isn't slapped into main).
- You correctly implement and use ImageEffect and Pixel classes.
- You correctly clean up all dynamically allocated memory in your program.

Image Manipulations (75pts; 5pts / ea)

- You correctly implement all image manipulations

Grade Distribution

Your final grade for the assignment will be determined based on the number of points earned. Note that failing to use good design (e.g. objects, appropriate data structures), regardless of score earned, may result in a lower overall grade.

Score	Points Required
100	85
90	80
80	70
70	55
60	40
50	25
40	20
25	15