Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

Working with PPM images, for the most part, are fun projects. It also allows us to use some of the coding techniques you have learned.

ASSIGNMENT OVERVIEW:

Often astronomers, or individuals interested in astronomy, will take a varying number of pictures of an object in outer space such as, stars, planets, galaxies, comets, etc. NASA's Hubble Site has a ton of images taken by the Hubble Space Telescope. http://hubblesite.org/gallery/ However, often these images may be grainy or have random noise in the images. To remove this noise, the photographer or in this case the Hubble Space Telescope will take several images then process the images using a technique called image stacking. Another problem a photographer (professional or not) will sometime encounter is the photograph may have some unintended object in the picture. With the luxury of digital cameras, we have the option of taking several snapshots of the same image, then applying a filter to the images to remove the unwanted objects (bird, ball, etc.) For this assignment, you are going to write the code necessary to solve the two problems discussed above.

LEARNING OBJECTIVES:

This assignment will give you practice working with the following concepts:

- Multiple files
- Command-line arguments
- Pointers
- Array of pointers
- Structs
- Parsing files
- C – I/O concepts (fprintf, sprintf, FILE*, etc.)
- Dynamically allocating memory
- Sorting Algorithms
- Reading and writing images
- FILE I/O
- Many other concepts

ACADEMIC INTEGRITY:

This is an individual assignment. You may not receive help from anyone other than myself a lab TA. Please review the academic integrity policy provided in the syllabus.

Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

REQUIREMENTS:

I will provide a ppm_utils.h file that includes the functions you will need to implement. Outlined below are several functions you **MUST** write.

1. **void openInputFiles(char* name, FILE* input[]);** The first parameter represents a word passed to main through command line arguments. The second parameter represents an array of FILE pointers.
   This function will create and then open, for reading, the input files. You will use the function **sprintf** to create a set of ppm file names. Each of these files should be opened for reading and stored in the array of file pointers passed to the function. You **cannot** hard code these file names. You **must** generate the file names using sprintf. The file names will be in the form of <name>_001.ppm. If the parameter (name) is "average" then you will create and open, for reading, 10 files. If the parameter (name) is "median" you will create and open, for reading, 9 files. I will provide you with two sets of ppm files. One set has the name of average_001.ppm, average_002.ppm through average_010.ppm. The other set has the name of median_001.ppm, median_002.ppm through median_009.ppm. When opening the files, you must check that the file open successfully. You can use assert or a function that you create. If you choose to write a function, if a file does not successfully open you must print an error message and exit the program.

2. **image_t* removeNoiseAverage(image_t* img[]);** The parameter represents an array of image_t* that point to dynamically allocated memory for the images used in this function. This function will use the images called average_001.ppm through average_010.ppm.

   This function will remove the noise or graininess in photos taking by the hubble space telescope. For each pixel calculate an average of the RGB channels of the 10 images read in. You will need to create a local variable of type imate_t * to store the pixels for the output image. (Don't forget to dynamically allocate the memory for the local image_t*.) The calculated average should be written to output variable.

   Ex. For the first pixel of all 10 images, you will read the red values and average them, the green values and average them, and the blue values and average them. Then save the average values in the first pixel of the local image_t* created for the output. Do this for all pixels.

3. **image_t* removeNoiseMedian(image_t* image[]);**The parameter represents an array of image_t* that point to dynamically allocated memory for the images used in this function. This function will use the images called median_001.ppm through median_009.ppm.

Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

To remove an unwanted object in a series of images you can use a method similar to that used in removeNoiseAverage function. Rather than calculating the average you determine the median. This will involve, for each pixel, reading the RGB values for all 9 of the images, store these values in an array – one array for Red, Green, and Blue. Sort the 9 values in each array. Create a local variable of type image_t* for the output. Write to the output the median (middle) RGB values. (Don't forget to dynamically allocate the memory for the local image_t*.)

4. **void sort(unsigned int* arr, int n);** The function described in 3 above requires you sort the values in the array passed to the function. This is what the sort function will be used for. You can choose what flavor of sort you want to use.

5. You are to write a driver called **main.c**; The driver should have two variables of type FILE* arrays -- one for the pointers to the **average** ppm files and one for the pointers to the **median** ppm files. It should also have two variables of type image_t* arrays – one for the pointers to the **average** images and one for the pointers to the **median** images. You should open the appropriate files, read the ppm files, call either removeNoiseAverage or removeNoiseMedian.

6. **image_t* read_ppm(FILE*);** This function will read the pixel values and store them. This means you will need to read the header and store it in a local header_t (you can use the function you wrote from lab7) then call allocateMemory. Next you will read the pixel values and store them. Note that this will be stored in a 2D pointer, as opposed to the 1D pointer that was used in Lab 7.

7. **void write_p6(FILE*, image_t*);** This function will call write_header. Then use a nested for loop to write the pixels to the output file.

8. **image_t* allocateMemory(header_t*);** This function will allocate the memory for the entire image (image_t*), this is a local image pointer. Next using the header passed to the function set the newly allocated image header equal to the header_t passed to the function. Then allocate the memory for the local image_t's pixels. This must be a 2D allocation. Then return the locally allocated image_t.

Other functions/requirements.

You must have a struct called Header typedefed to header_t – contains all header information.
You must have a struct called Pixel typedeffed to pixel_t - contains all information for the pixels.

Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

You must have a struct called Image typedeffed to image_t that will have: a member of type header_t, and a member of pixel_t**

Write a function to free the allocated memory. You must free all memory when you are done with the memory.

You must close all open file pointers when you a finished with the file pointer.

Your driver file must have minimal amount of code. If your driver has more than 75 lines of code, you should create a function such as, startProgram () that can be called in main.

You are allowed to make small changes to the functions described in this document, however you are not allowed to combine multiple tasks in one function. Functions should do specific task. The rule of thumb taught to me as an undergraduate, if your whole function cannot fit on the computers screen you are doing too much in that function. I looked at all of my functions and none are more than 50 lines. (this does not include comments).

The command to execute the program should be as follows:

<executable> <string average or median> <output file name>
./a.out average output.ppm

Above are the functions you **MUST** implement for this project. Depending on how you write these functions, there may be other functions you need to write for this assignment, such as a swap to use with the sort function.

FORMATTING:

You will need to add a header to each of your files like the following:

```
/************************
 *Your name
 *CPSC 2310 your Section
 *Your email
 ************************/
```

Your program should compile with no warnings and no errors. Points will be deducted if you have warnings or errors when compiled

- Your code should be well documented. (comments)
- There should be no lines of code longer than 80 characters.
- You should use proper and consistent indention.

Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

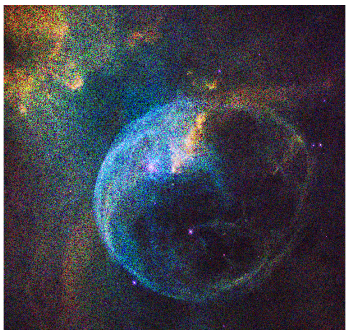
Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

SUBMISSION:

Zip your code files (.c and .h only, please DO NOT submit a makefile or the ppm files given to you/generated by you, but it is highly recommended for testing your code) **in a zip folder named <lastname>.zip to canvas.**

If you do not follow these instructions, you will lose 10 points.

Here is an example of an image that has noise in it and the output after the noise has been removed. Basically, I took an image from NASA's website, ran it through a program that introduced random noise in the image. Then ran it through the removeNoiseAverage function and the one on the right is the output.



This is the original image from NASA.

Lab 10
FUN WITH PPM IMAGES
DUE: April 15, 2024, Midnight

Here is a sampling of the images used to remove the unwanted object in the image (the person).




Here is the result after calling the removeNoiseMedian function with these images: