Korakot Santiudommongkol
CSC 412

Program Assignment #3 Report

imageIO_TGA.c

I have not add any new functions from this file, I have only alter the **readTGA function** and the **writeTGA** function. I alter the two function to take in the Raster Image Structure. I alter the **readTGA function** to take return a Raster Image struct that is completely filled out in the struct. As for the **writeTGA function** it takes the outfile directory to write into that file, a unsigned char raster, and an Raster Image. I added the unsigned char raster to the parameter because of the program **Split.c** where it alters the raster at 3 different instances and i found it would be simpler to add the extra parameter that way. All the functions takes in a log file also to write to it if any errors are to occured.

RasterImage.c

For this file, I put in 3 functions that I used in the 3 of the operators. One of the function is **getinFileName** which takes in a input file to a tga file, what they function does is obtain the filename without the extension and return it. The method I used was to copy the infile parameter so I am not directly changing the infile directory. I then used a for loop until I get the filename by checking for '/' character. The second function is **newOutFile** which takes in string directory, filename, chr which represents the string that comes after the filename, and type represents the extension. The function would first allocate enough space for the char pointer, then copy the directory on the allocated char pointer, and then concatenate the rest. The last function is **copyRaster** which copies the raster and all the pixels that contains all the color channels. In the bytesperPixel is 4 because since we are only working witch colored tga, I do not need to check if its a gray raster or color one. The function takes a char pointer that is allocated already and a char pointer to the raster. The function just traverse through entire raster and pixel of the raster copying the raster.

Dimensions.c

For the main function I check for arguments before proceeding to anything. Depending on the arguments the program behaves differently. I have 3 functions in the program depending on the arguments. One function is simple and print the width then the height with only 2 arguments. The next function **getDimensionMod** is when there are 3 arguments and 1 argument is the modification, depending on the modification the function either gives the word width and height and the value of the width and height, or it will just print the value of width or height, or if the modification is unknown prints an error. **getDimensionMod2** is when there are 3 arguments, 2 of them being modification, the function will first check if either 2 of the function has the mod

"-v" before proceeding because the whole point of the function is the assumption that "-v" is a mod.
The one thing that I did not account for the is the order of the arguments.

Split.c, Rotate.c, Crop.c main function
The main function for 3 of the program is similar since I need to alter the filename of the infile to get the outfile. All of the program has 2 common variable word and ext which represents what string comes after the filename and the extension of the outfile. The logic behind my method is to use the RasterImage.c functions I created to build up the outfile.

Split.c
The point of the function is to separate the RGB color channels. I stored the ending of the file in a char pointer array, I stored it in the array so that I could use a for loop and it works plenty fine because I had sure that RGB is stored in the right order because 0 represents the red color channel, 1 represents green, and 2 represents blue. I would create a newoutfile for each of the colors and run the function **alterRaster** which takes the Raster Image and the index. What the function does is allocate the same amount of size as the raster to a char pointer and copy the raster to the tempras so that I do not alter the raster stored in the struct in anyway. The function just traverse through every single pixel of the raster killing all the color channels except the one at +index.

Rotate.c
The way I get the output file is similar to that of Split. This function has 2 functions **RotateRight, RotateLeft.** Which by the name Rotate the image left or right. For both function I allocated enough space for the raster. Both functions traverse through entire pixel of the raster and moving it correctly so it would rotate left or right. The logic behind rotating left is making sure the first column moves to the last row of the raster, and for right is to make sure the first row moves to the last col of the raster. The main function takes in several arguments, one being a string which contains L or R, with number of rotation. I traverse through the string making Rotation as needed and for each loop I would switch the numRows and numCols each time. During the loop if there is a random specs in the string I made it so my program exited. I implemented the **EC** for the specification on checking for uppercase and lowercase arguments.

Crop.c
The method in getting the outfile is same, the only difference in this is that I implemented the **EC** for the outfile adding the index to the name starting from 0 and checking whether the outfile exists or not and will loop until a outfile that does not exists is found. The function I implemented is **CropImage** which tries to crop the image, but does not work exactly how I wanted it to, the image created is not perfect. The way I attempt to crop the image is by creating

a char pointer to copy the raster and changing the value of the raster. I would traverse through the raster the same amount of row and col as the new dimension. Since I know that the (x,y) correlate to the bottom left part of the image, but when traversing through the image we are working from the top left. For the x I would just simply add x. As for y since it represents the y coordinate from the bottom, I get the row value by subtracting the new height from the original height and subtracting y from that and then adding the current row traversing through.
The limitation for this program, the program will crash if the (x,y) arg input is not within the image dimension, or if the new dimension is bigger than the original image dimension, or if the combination of (x,y) and new dimensions are out of the original dimensions of the image.

Myscript.sh

The very first thing done is to make sure the output directory exists if it doesn't then the script will make it. The next thing done is to compile all of the operators. The script will traverse through the directory checking for any tga file by checking the extensions of each of the file. To crop the image to four quadrant I must used the dimension program to get the correct dimension and store it into the variable. Then to get the new dimensions of each quadrant is by dividing the width and height by 2 for an even split. Then create 2 list one of each containing the x and y coordinates of the origin point of the image. The origin point is for x and y is either x=0,x=half the width, y=0, y=half the height. The (x,y) would be the combination of the x and y values making 4 points in total. The script with traverse through the list calling the crop program with the different (x,y) point and the new dimension, then the last thing the script does is use the split program and the rotate program.