note 21 views

12:15 Lab Section - Supplementary Information

Hey, Guys!

If you were in my lab section, you'll notice it got a bit hairy, and we were pressed for time. I apologize for this. Here are some things I wanted to discuss, but didn't get the chance:

fstream Review:

- File streams, or fstreams, are a really intuitive way to access a file's contents.
- There are two types of fstreams: ifstreams for reading input and ofstreams for writing output.
- To open a file for either, you use .open(char* filename). You'll notice it has the type char*. You can convert a c++ style string object into a char* using the .c_str() function.
- Whenever you open a file, you need to check whether you actually opened it. Fstream's .fail() will return true if it succeeded in opening the file, false otherwise.
- You can write to and read from an fstream using the input (>>) and output (<<) operators. Just like cin and cout.
- Always make sure to close your fstreams with.close(), especially before you leave a function.

Write (string filename):
Finish this function before any others.

- The write function takes in a string. This string is your filename. You need to create an fstream that writes output to the given file.
- Debugging tip: cat -e filename to see your output with end lines visible as '$' characters.

So with all these manipulation functions, I would really recommend drawing out what you intend to do so you can make sure that your math is right.
For each of these functions you're manipulating the Pixels vector of vectors in the Pgm class. If you're using an intermediate vector of vectors in your solution, you need to copy it to the Pixels variable.
Below, I've described what you need to implement in human terms:

Clockwise() and CClockwise():

- So with this function, you're just swapping your rows and columns in specific ways. For Clockwise, you want your first row to correspond to your right-most column, your second row to correspond to your second-most right column, and etc. For CClockwise(), you're doing the same, but to the left-most columns. You'll have to create a new vector as your target.

Create(size_t r, size_t c, size_t pv):
(Size_t is an unsigned integer type, so you can treat it as an integer for the most part, but it can never be negative)

- This function is very similar to big_white from the previous lab, but stored in a vector with no header.

Pad(size_t w, size-t pv):

- You're just adding pixels around the image, so you need to manipulate your vector of vector to reflect this.
- Your first instinct might be to prepend onto your vectors. This is very inefficient though, as it requires you to move all the vector's data around in memory. You might want to consider resizing your vector to the correct size and then moving each pixel to its correct location

Panel(size_t r, size_t c):

- So, for this function, you're wanting to create a grid of the provided image repeated r*c times. So, you can create a new vector of vectors with size (r*img_r, w*img_w), and then loop through each pixel and find it's corresponding value from the original image. (Modulo operators are your friends here!)
- You can save yourself some pain by creating a row or column of your image repeated, and then repeating that row/column for the rest of the grid.

Crop(size_t r, size_t c, size_t rows, size_t cols):

- You're just grabbing an area of your image that is rows pixels high, and cols pixels tall. The top right corner of this rectangle should be at (r,c)

If you have any questions or want any feedback, feel free to email me or come to mine or any of the other TAs office hours!

Thanks, guys!

-Kody Bloodworth and ChaoHui Zheng

lab3

~ An instructor (James S Plank) thinks this is a good note ~

Updated 12 hours ago by Kody Bloodworth

followup discussions for lingering questions and comments