Graphics Programming Project 2 Report

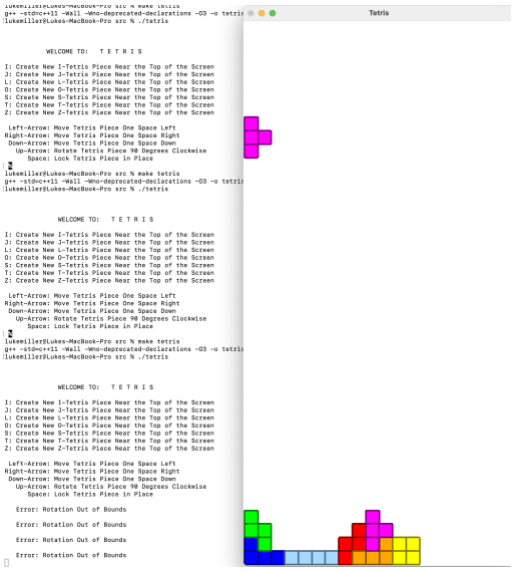
Luke Miller
010785453

Problem Statement:

The goal of the programming assignment is to generate the seven pieces of the game Tetris and display them on screen on command by shortcuts on the keyboard. This is to give us a first look into the callbacks OpenGL API for the C language. The inputs of the program are the user pressing specific keys on the keyboard and the program should draw, translate, and move the specified Tetris piece in the place/position the user desires. There are few places of error handling that keep the pieces on the board and don't accept non desired input from the user. For example, if the user presses the 'u' key the program does nothing because that is not a specified key for the program and if the user is on an edge and does an action like move or try and rotate a key then the program will test and make sure before moving said piece to make sure it won't move off the board, and if it would move off the board the program won't accept the input.

Design:

When I designed my program I first decided to tune up my first homework, which entail changing the drawSquare() to accept the RGB values as parameters not global variables. Then I changed the center point of each shape to be more in the center because some were drawing from points relative to far corners of their bodies. Then I decided what global variables I'd need to be able to manipulate and redraw old shapes, so I have two global vectors: one to hold all the locked in shapes, and one to hold the currently moving shapes. I also have to global constants for the diameter of the shapes draw and radius of the shapes drawn. Then I have two global variables for the currently moving pieces midpoint (x,y) coordinate. I then have some supporting functions for the current moving piece to rotate it clockwise, shift it left, shift it right and shift it down. Then the keyboard shortcuts have two callback functions. One for the i,j,l,o,z,s,t and space key, and one for the special keys for the left, right, up and down arrows. Then a function that takes a vector as parameter and draws the data set in the way I designed it in a repeated x, y, R, G, B form. The when the display callback is invoked it sends both vectors to draw the shapes. It also seems like a bug when trying to rotate at and edge and the program not doing anything when you want to rotate, but the rotation would mean that the piece would go off the edge of the program, so I implanted an error message that tells the user what they're doing is not allowed so know it's not a bug.



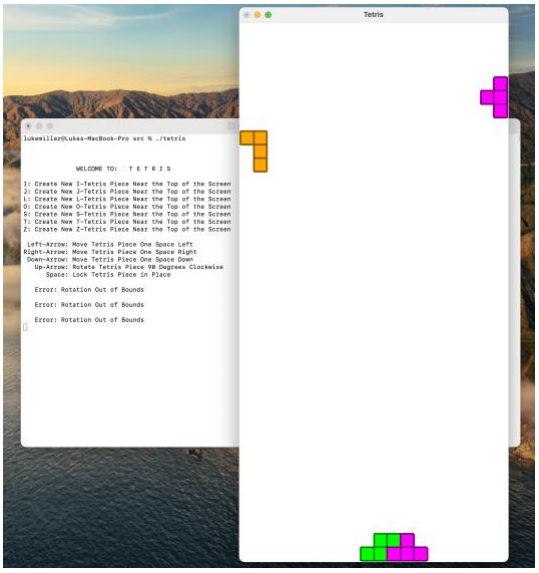
Implementation:

To get started I tuned up my first Tetris program with some design elements from the first solution to get it up to snuff. Then I copied the some of the default code from one of the 'office.cpp' files for the keyboard callbacks and

adapted them for the shortcuts I needed. After that I decided to pretty much get at it with how I implement rest trying to just get it to redraw a vector data set so I knew it would work when redrawing old shapes. The hardest part was trying to figure out the rotation stuff which I saved for last. I ended up coding my own functions for the rotation based off the Transformation slides provided, using cosine and sine. The first parts took maybe two hours then the transformations stuff got bogged down by trying to use matrixes and online research mostly confusing me, then just writing the rotation functions myself.

Testing:

After each transition was programmed, I checked to make sure it was moving the pieces correctly. Then after each shift was implemented, I retuned how each function drew its shape to keep a consistent midpoint. This was all tested for an easier implementation for the rotation. The rotation function is where I ran into most of my problems for it was doing some calculations wrong and making the current piece disappear entirely. After I discovered my math's logic bugs it worked like a charm even when rotating or shifting near an edge. This program is relatively simple but there were some special cases to test for like trying to rotate near a corner or near the top. I tried throwing every weird instance at the program and it seems to handle it well with no known bugs to report upon



Conclusions:

In conclusion, this program was a success. Everything works as expected. I am very proud of the modularity of the program and I believe this was a great next step for the Tetris assignment in the class. Knowing what I know now I wouldn't have wasted time trying to research the matrixes stuff and just made my own simple rotation function in the first place. If it weren't for that the assignment would have taken me less than 3.5 hours, especially if I wasn't reading stuff about matrixes at midnight. Three and a half hours with debugging isn't wholly that bad and I am proud of how it turned out.