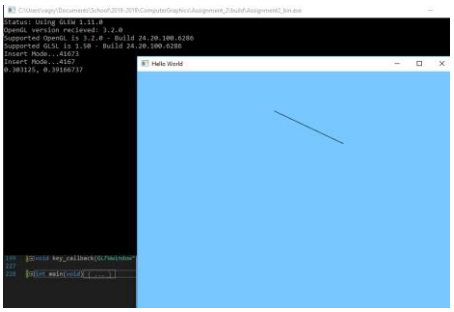
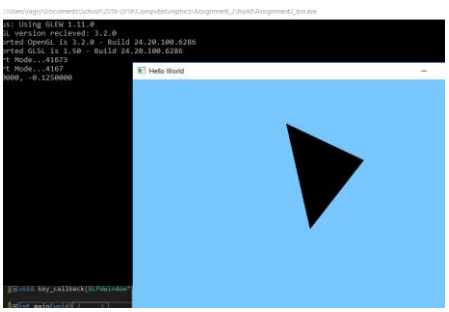
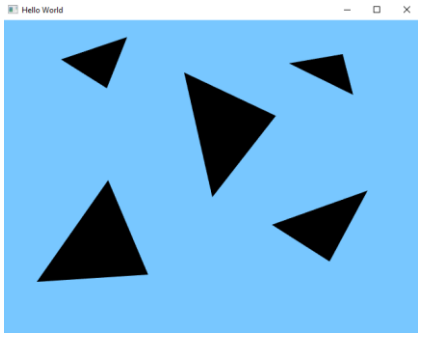
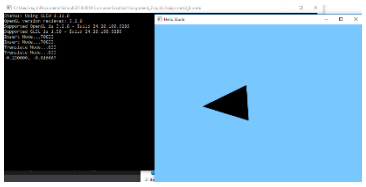
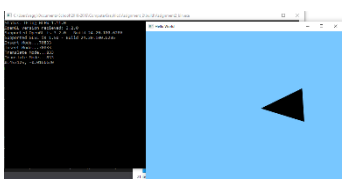
Assignment_2 – pyrokuna (Ryan Gutierrez)

When producing the source code for Assignment 2, I was able to properly implement part 1.1. Note that I made no attempts to implement either of the extra credit parts. For parts 1.2, 1.3, and 1.5, I made attempts to implement them, but unfortunately could not get them working. To compile this assignment, use CMake Gui to create a build of the assignment (using visual studios) the way we were instructed for the first assignment. Then build in Visual Studios and execute the executable.

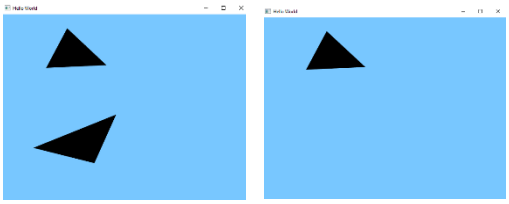
Part 1.1

Part 1.1 was implemented properly—as far as I can tell—except for triangle highlighting when doing translation. I was also able to get my triangles to draw before they are placed (something I am proud of as I had not seen many students accomplish this). In order to do this, I had to store my current mouse position in the vertex at every frame (you will see that when you run my executable, the mouse position—with respect to the window—is constantly updated in the terminal) and draw a line between the previous vertex and the mouse (after one click [first picture]) and a triangle between both previous vertices and the mouse (after two clicks [second picture]). I can also insert multiple triangles [picture 3]. To see this in full effect, refer to the program as a still does not do it justice.

Translation works properly as well (There isn't much to say about this other than I just calculated the position by subtracting the position of the cursor when the mouse was clicked by the mouse's current position in a given frame.



Here you can see deletion on display. Unfortunately, sometimes my delete function will delete two triangles at a time and I was not able to figure out why.

Part 1.2

In attempting to do rotation, I was able to calculate the area in order to figure out whether or not the cursor clicked within a triangle and also set which triangle it was that was clicked. I attempted to find the barycentric center of the triangle next but failed there. Had I succeeded I would have attempted to rotate through these procedures: (1) move the triangle to the origin of the window by subtracting each vertex by the center while simultaneously storing its original center, (2) do the rotation by multiplying by the rotation matrix, (3) move the triangle back to its original position using the previously stored center.

I never made attempts to do any scaling, but I understand that it would have used the same method for triangle selection and center location as the rotation would have used. Had I attempted to implement this part of the project, I would have followed these procedures: (1) move the triangle to the origin as in rotation, (2) find a line going from each vertex to the triangles center, (3) increase that line (or distance) by 25%, (4) move the triangle back to its original position as in rotation.

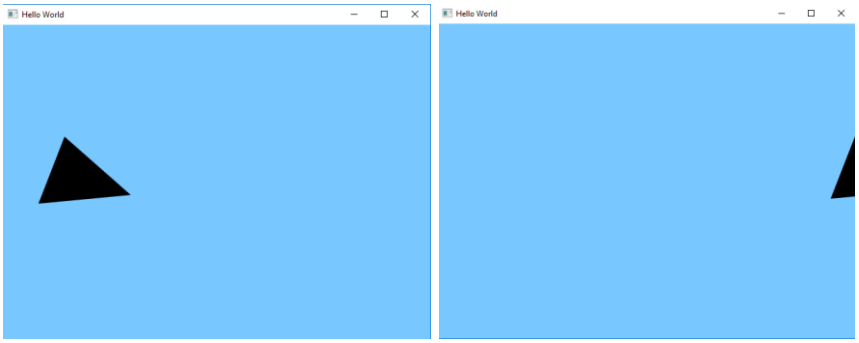
Part 1.3

Unfortunately, I was not able to implement color either. In an attempt to do so, I was able to find the nearest vertex to the cursor at any given click. Had I attempted to implement this further, I would have followed these procedures: (1) create a separate color matrix, (2) update the VBO to show the addition of the matrix.

Part 1.4

Unfortunately, I was not able to implement view control either. Honestly, I wouldn't even know where to begin with this.

Part 1.5



Unfortunately, as with the last three sections, I was not able to fully implement the animation. I did make it so that when holding A (AFTER DRAWING ONE TRIANGLE INITIALLY), it would traverse the screen to the right. It was supposed to oscillate back and forth between the left and right sides of the window, but I could not get it to go back to the left (it would just stop as seen in the right-hand image above).