Brendan Schneider
CAP5726
11/3/20

Assignment 2 README

Compilation Instructions

I edited the CMakeLists file slightly because it was not finding some of the libraries in the repository, but otherwise compilation should not be any different from the standard procedure.

Part 1 - 2d Modeling

Notes

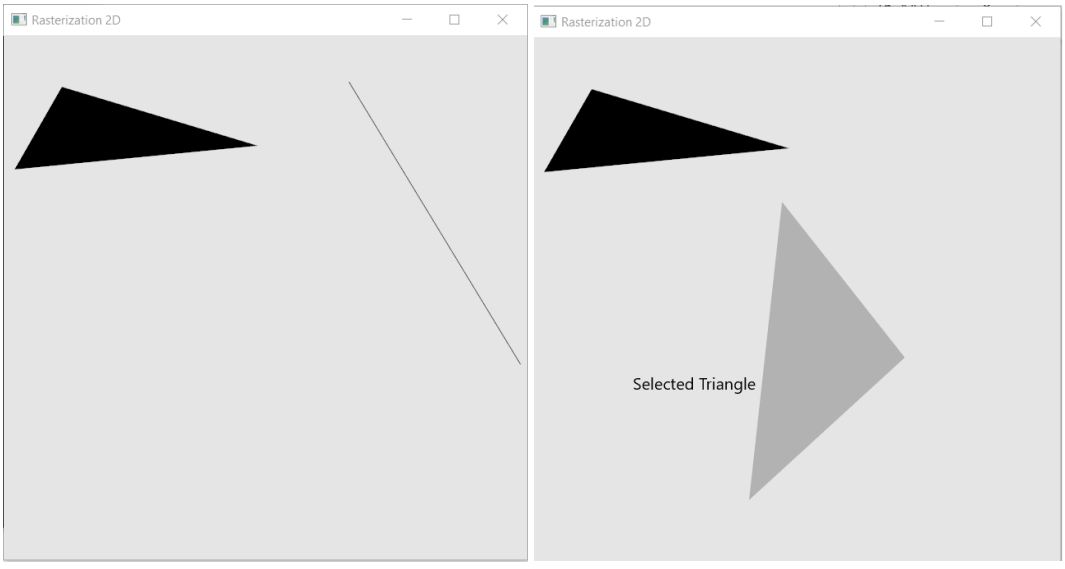
- Window resizing does not work for the 2d rasterization demo. Since it was not in the requirements for part 1, I did not spend much time on trying to fix the problem.

Triangle Soup Editor

Everything here should work as intended. If the mode is changed from insertion before the triangle is given 3 sides, the temporary lines will be erased. Translation mode works in conjunction with the options in the “rotation/scale” section. Selecting a triangle is based on draw order - that is, later triangles are considered to be “above” earlier triangles for purposes of selecting the nearest triangle.

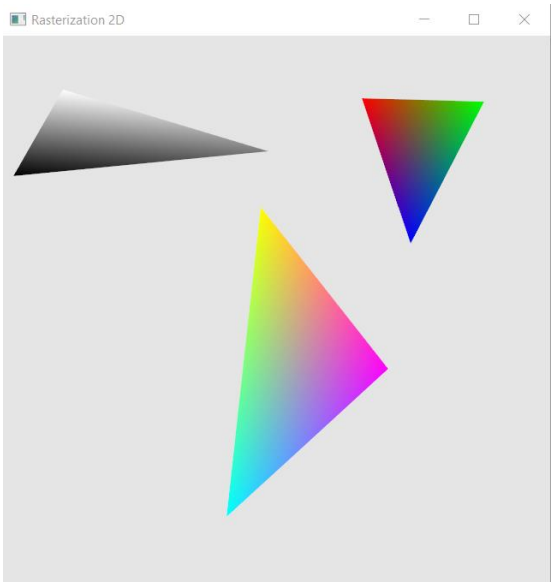
Rotation/Scale

While a triangle is selected (including while it is being moved), it can be rotated and scaled as expected. Scaling is done as a percentage of the current triangle, and not as a percentage of the original triangle.



Colors

The colors mapped to keys 1-9 are: black, white, red, green, blue, yellow, magenta, cyan, and gray, respectively.

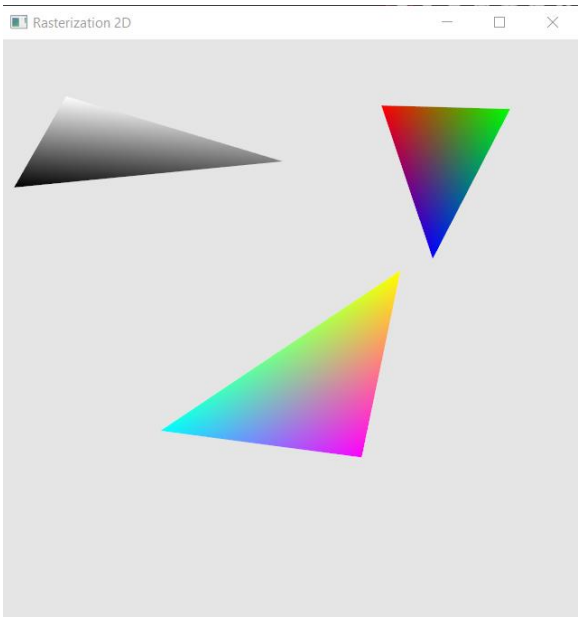


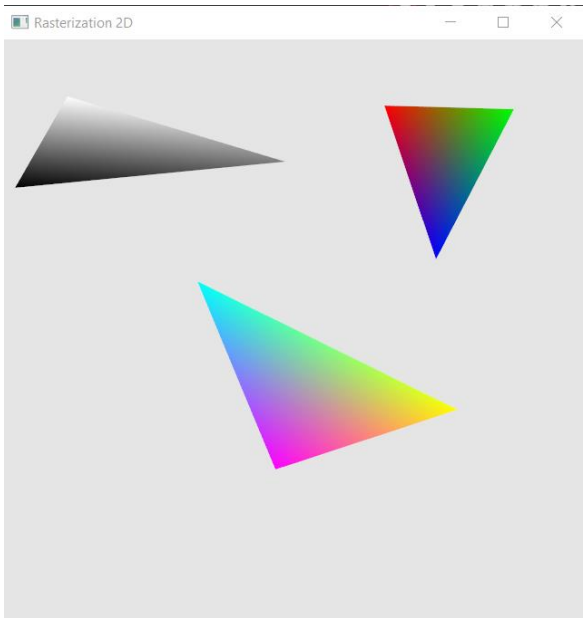
View Control

As was expected, this section was implemented within a shader. At this point in the project I was not sure how matrices interacted within GLSL, so the calculations are done by hand because of how simple they were.

Add Keyframing

After pressing the ‘z’ key, the currently selected triangle will await the next rotation, translation, or scaling. After that point, no other action may be taken on the triangle besides that. After pressing ‘z’ again, the triangle can change between 1 of 5 frames interpolated from the starting and ending points of the animation using the ‘n’ and ‘m’ keys, until either another selected triangle has the ‘z’ key pressed on it, or the mode is changed.





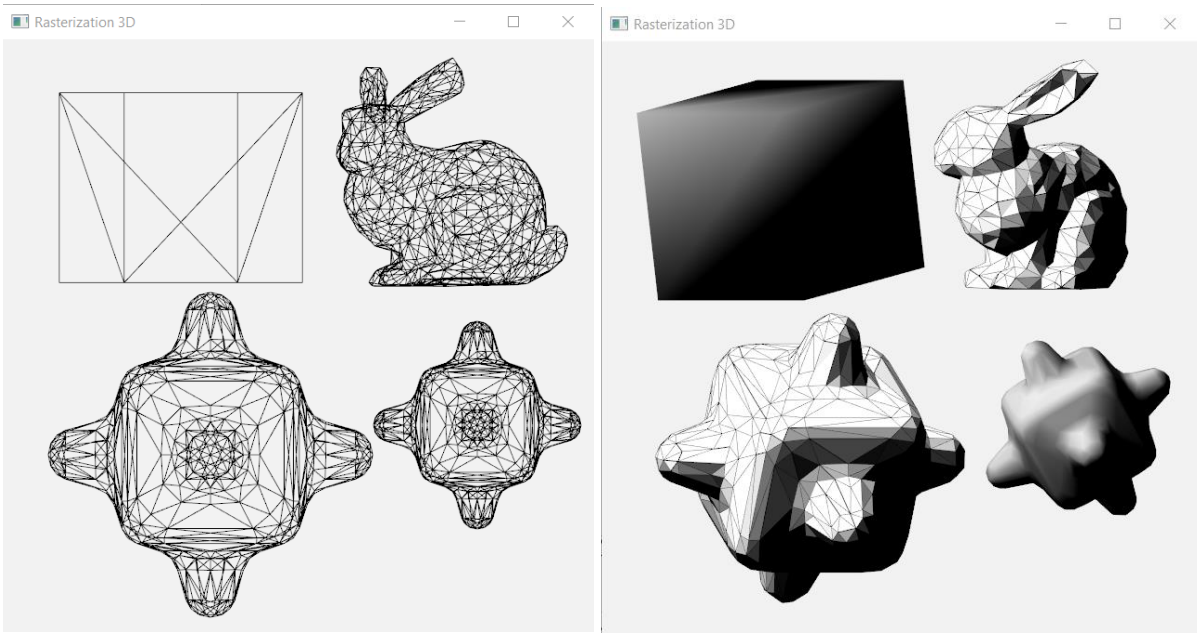
Part 2 - 3d Modeling

Notes

- The assignment asks that we only upload the VBO to the GPU once, only doing transformations within the shader. I *did* do all transformations within the shader and only linked to the VBO a single time, but because I stored the objects in different arrays I had to update buffer data quite frequently as I swapped between them. I hope this approach was okay (again, I did the transformations within the shader, this was just because of how I chose to store the objects).

Scene Editor

The scene editor allows for up to 3 of each object, created using the 1, 2, and 3 keys. Additionally, the 7, 8, and 9 keys allow for the most recent of each of these objects to be deleted. Initially, objects are rendered as wire frames, with bunnies upscaled by 5 times and bumpy cubes downscaled by 8 times.



Object Control

An object can be selected by left clicking on it on the scene. Rotation is achieved through the i, k, j, l, u, and o keys, where each press rotates 10 degrees along one of the axes. Translation is possible through the w, a, s, d, q, and e keys, where each press moves the object by 0.1 units ($1/20^{th}$ of the screen). Scaling a selected object is done through the n and m keys, scaling the object by 20% of its current size. The z, x, and c keys swap between the rendering modes (wire frame, flat shading, and Phong shading, respectively).

Camera Control

The arrow keys allow for the camera to be rotated on the unit circle around the x and y-axes. I actually did account for being able to rotate around all 3 axes, but at this point in the project I was running out of key bindings to use, so I decided to just rotate around 2. The aspect ratio of the scene is 1:1, and a window resize will maintain this, centering the scene onto the window if the window itself does not match the aspect ratio. I did not manage to implement a perspective camera, so that particular requirement was not completed.