Rendering Engine Final Report

Ovidiu Mocanu
Nathan Holzworth

Abstract:
We managed to load in a 3D object into the canvas scene. The 3D object was supplied by a .obj file. It can be manipulated like any webgl object with rotations, scaling, and translations. In this project we were able to make a simple camera that rotates around the object and is able to play back the movement. Simple lighting was used to give the object shading.

Introduction:
For this project we used an outside source for how we read and loaded the object file into the canvas. However we had trouble modifying it to load more than one object at time into the scene. This lead to a major halt in progress to making our engine work. This delayed work on collision detection as we needed 2 objects in the scene to test the code. By the time we managed to get multiple objects loaded into the scene it was too late to work on collision detection. We only managed to add the simple camera control system. The camera control system manipulated the viewprojectmatrix which allows us to rotate the camera around the y axis using keyboard controls. There is also a playback feature in which we record all movements that occurred and then play them reversed. The overall code is split between multiple branches, the camera control is in the WIP branch while the last minute breakthrough for multiple objects in master. Sadly we didn't get to merge the two branches in time. However the two branches can be test for their features.

Background:
The project gave a lot of insight on how complicated game engines are. The WebGL framework is very in depth as such it can be very difficult to troubleshoot misunderstandings in the code. There is also a lot of linear algebra involved because of matrix multiplication. Matrix multiplication is used for every form of transformation

Methods:
We read in the .obj file by sending an XMLHttpRequest to read the file from the local folder of the javascript file. This method of loading the file to the script is unconventional as this is not how xml requests should be used. Chrome blocks this form of request because of CORS policy however on Mozilla it does not which is why we used Mozilla. After receiving the file, the script then reads through and parse information about vertices, indices, and faces. The relevant information is stored in arrays that are then fed into the draw function.

The camera control system manipulates the viewprojectmatrix with data that specifies which angle the camera is supposed to be at. This allows us to rotate the camera around the y axis using keyboard controls by incrementing or decrementing the angle. There is also a playback feature in which we record all movements that occurred in a stack and then pop them back out to play the movement in reverse. This function is not set to a key but the code should still work if applied to a key.

Resources:
We modified the OBJViewer from the intro graphics textbook to better fit the needs of this project.

Tools:
We used Javascript and Webgl like planed in the proposal. However unlike last time where we used chrome to run our code this time we had to use Mozilla because of our method of getting the .obj files.
Communication:
We communicated over discord like planned. It allowed us to share files and code snippets without the hassle of pushing and pulling from github for just one thing.

References:
Webglfundamentals.org. (2019). *WebGL 3D - Cameras*. [online] Available at: https://webglfundamentals.org/webgl/lessons/webgl-3d-camera.html
Matsuda, K. and Lea, R. (2013). *WebGL programming guide*. Upper Saddle River, NJ: Addison-Wesley.