### Quest McClure

### Design Decisions

At the beginning of the class, I didn’t really understand the task at hand, so I chose objects that I would enjoy creating regardless of it being an assignment for school. This ended up being an assortment of weapons, as they usually consist of primitive shapes and have a history of being rendered for video games. I had a general idea of the shapes I would use but did not fully grasp the complexity of OpenGL and the steps necessary to see one’s object come to life. Starting on the hammer at the center of my scene was a fantastic foundation because it demonstrated the steps I would need to take to render it believably. At first, it was a squished cube with another cube as the handle, painted rainbow. This took hours to complete! Once texturing and lighting were added, it only added to the complexity and exhaustion of writing out each vertex of the object one by one. When I finally started to work on creating a cylinder for the handle, I studied algorithms and the different methods that OpenGL has (tons) to get one on the screen. I pigeon-holed myself into using one shader for all my objects, but it helped with understanding the concept of rendering complex shapes and fitting them into my design. When my cylinder algorithm was complete, it dawned on me that I could render a primitive shape and squish and deform it to fit what I needed it to look like. This was the basis for the scythe, as it is created almost exclusively with cylinders except for the blade. I stuck to this strategy as I moved on to my other shapes, added a sphere to the mix, and drew by hand complex shapes like blades. Anything could be created by drawing every shape’s 3D point in space, but the efficiency of primitive shape algorithms really opens up a whole different playing field of automation when rendering.

Moving around the world as the virtual camera boiled down to already understood input device functionality like using a mouse for camera pitch and yaw and keyboard WASD for movement. Additional controls I added to play with the world like turning the lights on and off and the perspective change button was mapped to the first letter the word they use is, like P for perspective and L for lights. Additionally, I added camera sensitivity and speed controls because everyone’s device and their preference is different, like mouse DPI, screen size, etc., so it is easier to adjust and easily added as a loop on my end.

The additional functions I added were to make the drawing and creating of objects a simple task. First, a Create function takes the mesh and creates a VAO for it, which is called from the main function. This happens for each object I need, like the Hammer Handle or Sword Blade. The reason I split objects up like this is to make it easier to visualize what the code is doing and help with applying different textures or modifications to a similar object. For example, the main stick part of a scythe could require different textures and be a much higher resolution cylinder than the handles that wouldn’t require much detail, as they are smaller. The Draw functions I created allow each object to have its own material, scale, rotation, and transform. The handle of the hammer is wrapped in leather, so it is not as shiny as the underlay handle, which is a very shiny metal handle. They are the same mesh but splitting how each one is drawn into their own function helped me increase how many things I could tweak and easily I could do it. In the future, I hope to understand how a professional organizes their code and what functions and practices they use to create a scene. Mine definitely needs a lot of work, and there are blatant performance issues, so it would be incredibly interesting to work with something that is industry standard. In terms of drawing objects, I came across a function that imports FBX files which is mesh data created in something like Blender straight into OpenGL. Using powerful features like that, and building something like ray-traced lighting would be so cool!