Final Project: Design Decisions

Charles Hilton

Southern New Hampshire University

# Summary

This project certainly was a challenge. Having spent now a majority of my life CAD modeling, this course has given me a certain appreciation for the tools that implement OpenGL in a more intuitive, user-friendly way. I had great ambitions on the outset of this class, but due to time constraints and a lack of familiarity with tools like AssImp, and struggles with trying to make my own OBJ importer, I reverted to using simple, hand-coded meshes, and the primitive libraries provided.

My scene was simple, much simpler than I had planned on making it. For the base plane, I used a hand-coded set of vertices, and applied a desert texture to it. For my mouse dock, I used a conjunction of a hand-coded mesh for the base, and spheres that were stretched out using scaling, applying appropriate textures to both. The trick for the hand-coded objects was tweaking the texture coordinates and surface normals. I also approximated a mouse using a sphere, stretching it using scaling to more closely mimic a real-world mouse. The mug was interesting. I would have liked to make the handle hollow. But I don’t think there is a cut command to subract the volume using a smaller cylinder, like there is in proper CAD software.

For navigating around the scene, I added the functionality to move up and down by modyifying Camera.h and adding the needed keyboard key press events in the main program. I was rather proud of how I handled switching between orthographic and perspective projections. To prevent the system from capturing to many “P” key presses, I added a debouncer using a couple of simple techniques, that could be easily applied to other key presses if needed.

I refactored the UCreateMesh function provided in our tutorials into a UBindMesh function, to allow arrays to be passed in. This helped clean up the code, simplifying the process of trying to create individual VAO’s, VBO’s, etc. for each object I wanted to add to my scene. I would have liked to have spent more time refactoring code. It would have been nice to create a collection of objects and iterate across it to render each individual object, handle the transformations for each, and do all necessary binding. I got part of the way there. I think I might actually turn it into pet project, making a simple graphics engine per say that can load in different assets and make it easier to manage them. Depending on the file type though, I think it might be necessary to have multiple shader programs, to handle cases for file types that provide texture coordinates or surface normals. In my efforts to try importing OBJ files, I started modeling things in SolidWorks, then exporting to STL (which only has vertex XYZ locations), importing into Blender, then out to an OBJ file. Depending on the flavor of OBJ file, it can have texture coordinates or nor, as well as optional normals. I didn’t feel it was time efficient to try handling all of the different conditions, and thusly how to bind the VBO appropriately with arbitrarily changing data, so I opted to go with the old school approach of banging out arrays for the hand-coded objects in my scene.