Aidan Mellin

Prof. Bayliss

CSCI 510

Reflecting on a flat moon

Overall, I had more fun than expected with this project. Going into it, I was a smidge worried that it might be a project that would keep me up for long nights, but that wasn’t the case. Some parts were tough but when developed, most parts fell into place.

For a project breakdown, we go point by point. “Scene has basic camera movement”: It has more than basic camera movement. It has a right-click function to look, left-right-forward-back movement, and the FOV is changeable. I’m particularly proud of this part as it was the most freeing aspect of this project. “2+ textures must be shown”: While one might argue that the stars are more of a particle field or system than a texture – as textures are a 2D to 3D translation – this system works better with the constraints of the project. To have stars visible from or on more than one side of the terrain, they need to be in a box or sphere. Without the use of sphere libraries, that’s out. For a box, or more technically six flat rectangles with points spaced out on them surrounding the camera, there could be disruption if zoomed out far enough and seams or other things that could interrupt the viewing. The way of rendering it as a particle field allows me to randomly size them and place them at different distances such that they can be interacted with. The stars are procedurally generated, which allows for more customizability and lack of repetition. “Scene should use procedural generation”: the in-house Perlin noise generation runs every time the page loads, which determines the height mapping of the vertices in the grid. The stars are also procedurally generated. “The scene should not contain models”: check.

As mentioned in the presentation, there wasn’t much in the code design. We like to keep our functions simple and doing only one task, so any function that would need multiple steps (such as CreateGrid) would have the individual steps like noise calculation and texture mapping pulled into their own functions and then back into the grid creation. The project was small, which allowed it to be in one script.js without having to separate all of the sections. I would’ve if there was more to it.

My favorite parts included developing the camera controls and the wireframe generation. Both were a step to the side (a side quest, if you will) but made me enjoy my time much more. Deciding how to actually connect the vertices after the height generation led me to research the marching cubes algorithm and boids (for some reason). For the camera controls, it would’ve been nice to flush them out a bit more. I definitely think my favorite part of the camera control (besides the right click + dragging to look) is the fact that the cursor hides when holding the right click. That was a quality-of-life uptick.

I had a particular issue with the initial generation of the wireframe; I couldn’t figure out how to get it to connect to the next row down when there was a change in the Z or Y (depending on your point of view), which was fun to figure out. I admittedly relied on Luke to help me with the texture mapping. I could not for the life of me get it to work with the texture and fragment shaders. He helped me out on that part, so no sweat.

I am both proud of this project and surprised at how well it came together. This pride is in no small part to having Luke as a partner. There were some surprises with how aspects were rendered, but there were not many code-breaking bugs. Most issues were localized to specific functions. Much of the project was built small such that there was easy testing and tweaking before implementation. While I would’ve preferred to use libraries (we did technically use a library for matrix multiplication), I understand why we didn’t. We probably could’ve been a bit more creative with our concept, but we are still proud.