Steven Jin

A11268464

**CSE 163 Final Project: Basic High Quality Rendering**

Video Link: <https://youtu.be/6zmrrWsl48E>

**Documentation:**

**Application Controls:**

Mouse wheel to zoom camera in/out.

Left mouse button click + drag to rotate camera.

Cascading shadow map settings – press 0 to change select shadow map #0, 1 for shadow map #1.

Up/Down arrow keys to change thresholds for each shadow map.

Thresholds:

Shadow map #0 – between 10.0 to 100.0, default 100.0.

Shadow map #1 – between 100.0 to 500.0, default 500.0.

Shadow map #2 – From max threshold of shadow map #1 to 1000.0 (Cannot be changed).

**Implemented Features:**

Shadow Mapping – Implemented a cascading shadow map with three levels, based on the z distance of the vertex from the camera. Cascading shadow maps are an add-on from normal shadow maps. This makes objects closer to the camera have higher quality shadows than objects further away. PCF is also implemented for antialiasing by taking the values of the 8 surrounding pixels.

Environment Mapping – A basic environment mapping from the environment cube map of the skybox can be found on the sphere in the scene.

Water – I implemented water in OpenGL as an add-on to environment mapping. I implemented it using two textures – a reflection texture and a refraction texture. I simulated reflection by placing the camera below the water surface and loading the result into a texture, then placed the camera above the water surface and doing the same for the refraction texture. I then used a du/dv map texture to simulate the water distortion, and used a float incremented every cycle to simulate water ripples. I then mixed the reflection and refraction textures to create the final water surface texture.

**Project Files:**

Cube.h / Cube.cpp – Creates a basic 1 x 1 cube and renders it with textures and shadows.

Floor.h / Floor.cpp – Creates and renders a basic quad, used to render water textures.

Sphere.h / Sphere.cpp – Creates and renders a basic sphere, used for environment mapping

Terrain.h / Terrain.cpp – Loads and renders a mesh from a given height map (located in the textures folder). The SOIL library is used to load the greyscale image, and a mesh is generated based on the width/height of the image and the greyscale value of the image at each pixel location complete with normals and texture coordinates.

Skybox.h/ Skybox.cpp – Creates and renders a skybox, using textures located in the skybox folder.

Main.h / Main.cpp – Creates the window and sets OpenGL settings, taken from CSE167 base code.

Shader.h / Shader.cpp – Creates and loads OpenGL vertex and fragment shaders from file, taken from CSE167 base code.

Window.h / Window.cpp – Responsible for setting up textures and calling render a various objects

Important Functions:

initialize\_objects() – initializes all objects in the scene and places them in the proper location

initialize\_textures() – Initializes textures in the textures folder

genWaterBuffers() – create FBO and textures for both the reflection map and the refraction map for the water.

writeWaterBuffers() – write the scene into the reflection buffer and then the refraction buffer. For the reflection texture, the camera is placed below the water surface and any objects below the water is culled. For refraction, the camera is placed above the water surface and any objects above the water is culled.

genDepthMap(width, height) – Creates the depth map frame buffer and 3 textures, for each level of the cascading shadow map.

writeDepthMap(width, height, index) – writes the scene into the depth map specified by the index.

display\_callback() – Called every cycle and renders scene to depth maps and water textures. Then renders the skybox, followed by the terrain / boxes with shadowing, then the water and finally the sphere with environment mapping.

**Shaders:**

depthMapShader.vert/.frag – Stores depth value from light’s POV into a texture.

environShader.vert/.frag – Gets reflection direction and use it as lookup into the cube map texture of the skybox for environment mapping. Used to showcase environment mapping on the sphere.

shader.vert/.frag – General shader, responsible from calculating shadows from the correct cascading shadow map level, getting the texture RGB values using the texture coordinates, and calculating directional lighting, the summing all of these up to output a final value. Used for terrain and buildings.

skyboxShader.vert/.frag – Loads cube map texture based on texture coordinates for the skybox.

waterShader.vert/.frag – Used for the water textures. Calculates the distortion and ripple effects for the water textures, uses it to offset texture coordinates, and then mixes the reflection and refraction coordinates to produce a final result.