COSC 4370 – Homework 4

Name: Rayyan Rahman

PSID: 1893113

April 4, 2023

1. Problem

   The assignment requires us to create a textured cube by implementing texture mapping in OpenGL.

2. Method

   Given the starter code, I had to modify the vertex shader and fragmentation shader files. The Camera.h and main.cpp also required me to implement functions to view the cube and bind the textures. Some of the code that needed to be implemented in the main.cpp and Camera.h was the same as the Phong shader assignment. So, I reused code and added it in their respective locations. My general methodology was using functions that load and generate textures such as glGenTextures(), glBindTexture(), glTexImage2D() or similar functions, specifying texture coordinates, passing texture data to the fragment shader, sampling the texture.

3. Implementation

   The code that needed to be modified in the vertex shader file is I needed to implement the gl_Position to be able to view the cube. I reused code from the Phong shader assignment as it was the same implementation. As for the UV, I needed to output the texture

coordinates. So, I created a vec2 type because opengl expects (0, 0) to be the bottom-left pixel of the image, whereas typical images like jpg have (0, 0) as the top-left pixel so you need to invert the y parameter of the UV in the vertex shader.

In the fragmentation shader file, I needed to implement the texture using UV coordinates. I created a 'texColor' variable that accept 4 float values (red, green, blue, and alpha) and I accepted the parameters of myTextureSampler is a texture sampler that specifies which texture to sample and the 'UV' which specifies the coordinates of where the pixels needed to be rendered.

I needed to create a view matrix in Camera.h. It was the same methodology as the Phong assignment. We use 'lookAt' to construct the view matrix and it takes 3 arguments of position of camera, target of camera and the up.

For main.cpp I needed to modify 3 things: UV buffer, projection matrix, and texture bind. For the UV buffer, the glBindBuffer function is used to bind the buffer object to the GL_ARRAY_BUFFER target. This target is used to hold vertex attribute data such as texture coordinates. The glBufferData function is used to copy the texture coordinate data to the buffer object. Finally, the vertex attribute pointer for the texture coordinates is set up using the glVertexAttribPointer function. The glGetAttribLocation function is used to get the location of the vertexUV variable in the vertex shader. The glEnableVertexAttribArray function is called to enable the attribute array for the texture coordinates, and glVertexAttribPointer is used to describe how the data is laid out in the buffer object. For projection matrix we need 4 parameters for 'perspective': field of view, aspect ratio, distance to the near clipping plane, and distance to the far clipping plane. As far as the texture bind the glActiveTexture(GL_TEXTURE0) function is used to activate

the texture unit GL_TEXTURE0, which is one of the many texture units available in modern graphics hardware. The glBindTexture(GL_TEXTURE_2D, textureID) function is used to bind a 2D texture object with ID textureID to the currently active texture unit, GL_TEXTURE0. This function tells OpenGL to use the texture with ID textureID whenever a texture lookup is performed in the shader program.

4. Result

The image below is a .png of the output. There is another file ending in .dds which shows the individual faces of the cube.