COSC 4370 – Homework 4

Name: Grishma Vemireddy

October 2024

1 Objective

In this assignment, we are to implement texture mapping in OpenGL. The uv data is hard corded in the main function. Required to write code to transfer the uv data to OpenGL buffer, just like what is done for vertex position. Also, need to write code for binding texture in the rendering loop and shader code to draw the texture.

2 Methods

To start with, some of the code needed was similar to what was implemented in HW3 so I reused some of the code from HW3 for main.cpp, camera.h, and texture.vs. I referenced the material found on OpenGL website to learn how to implement the code using glGenTextures(), glBindTextures(), texture(), glGenBuffers() to set up UV buffer in main.cpp and texture.vs/.frag
Reference Material: **https://learnopengl.com/Getting-started/Textures**

3 Implementation

Projection Matrix (main.cpp): Implemented using glm:perspective(glm::radians(45.0f), (float)WIDTH / (float)HEIGHT, 0.1f, 100.0f); which is similar to the previous homework.
reference material: LearnOpenGL - Coordinate Systems

UV buffer (main.cpp): To set up the UV buffer, I first generated a buffer ID using glGenBuffers(1, &UVBO). Then I bind the newly created buffer using glBindBuffer(GL_ARRAY_BUFFER, UVBO). After that we make a call to the glBufferData(GL_ARRAY_BUFFER, sizeof(uv), uv, GL_STATIC_DRAW) which copies the previously defined vertex data into the buffer's memory. glBindVertexArray(containerVAO) bind vertex array object and glBindBuffer(GL_ARRAY_BUFFER, UVBO) copies vertices array in a buffer for opengl to use. We then set the vertex attributes using glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0) and glEnableVertexAttribArray(1) to interpret the vertex data and enable it by giving the vertex attribute location as its argument. The code for creating this buffer is same as the one already given in the starter code for VBO from line 155 till

line 166, all I had to do was replicate this code and pass-in appropriate values. reference material: LearnOpenGL - Hello Triangle

Bind texture (main.cpp): To bind the texture I used the function glBindTexture(GL_TEXTURE_2D, texture); reference material: LearnOpenGL - Textures

GetViewMatrix() (Camera.h): Implemented this function using the glm::lookAt function that was mentioned in OpenGL section going over Camera. This code was reused from hw3 as it is the same implementation. reference material: LearnOpenGL - Camera

Texture Vertex shader (texture.vs): I set gl_Position = gl_Position = projection * view * model * vec4(position, 1.0f) which is same as phong shader assignment and set UV = vec2(vertexUV.x, 1.0 - vertexUV.y) as the texture coordinates are inverted in compressed textures. This would correctly fetch the correct texture coordinates. reference material: Tutorial 5 : A Textured Cube

Texture fragment shader (texture.frag): I used the built-in texture() function that takes in the texture sample and the corresponding texture coordinates as arguments to sample the color of the texture, color = texture(myTextureSampler, UV); reference material: LearnOpenGL - Textures

4 Results

These are screenshot of the resulting cube