COSC 4370 - Homework 4

Mauricio Perez Guzman

November 2022

1 Problem

Given template files main.cpp, Camera.h, Shader.h, texture.dds, texture.frag, and texture.vs we had to implement texture mapping in OpenGL. There was already hardcoded data in the main function. We had to write a code to transfer the UV data to the OpenGL buffer, There was already a vertex position being transferred to the OpenGL buffer to use as an example for the UV.

2 Method

Main: In order to set up UV buffer, we needed to set UVBO, in the same manner, it was already done for VBO. Later on, we had to set up the projection. After that, we had to bind the texture to the right coordinates.

Camera.h: Inside a GetViewMatrix we calculated what the camera should look at using the function lookAt that returns where to face the camera. The camera.h file implements the capability to move around, from the camera point of view, around an object, in our case an orange/red cube.

texture.vs: The vertex shader manipulates the coordinates and the main point here is that it sets up the gl_Positon variable. This is used to store the current position of the current vertex. This file also sets up the UV.

texture.frag: To calculate the color we needed a texture function that would ultimately return the numbers.

3 Implementation

Main: Using the glGenBuffers() followed by glBindBuffer, and glBufferData, I was able to set up the UVBO Gluint variable on the uv buffer. This was an implementation exactly like the VBO. For the projection section, we were able to implement a perspective function with a radian, width/height, 1.0f, and 150.0f. For the bind section I implemented a glActiveTexture() function followed by glBindTexture(), glUniform1i(), and a glEnableVertexAttribArray, glBindBuffer, glVertexAttribPointer() for each buffer. So in the end we had two attributes buffer, UVBO, and VBO.

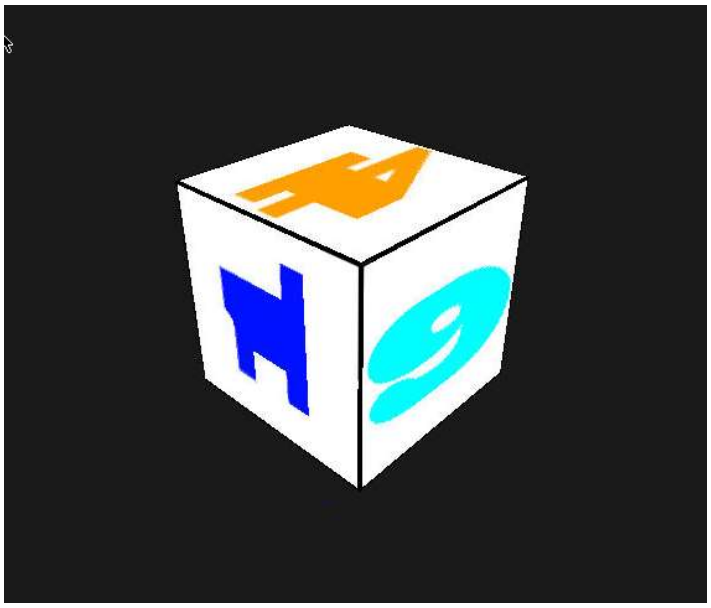
Camera.h: Inside a GetViewMatrix we calculated what the camera should look at using the function lookAt() that returns where to face the camera. The lookAt() function took in 3 parameters. The first one was the position in relation to the world. The next one told us the view we wanted to get. which was the front of the cube. The third was making sure the vector was in the positive y direction which was by default set at (0.0f, 1.0f, 0.0f).

texture.vs: gl_Position was set by multiplying the projection value by the view, model, and actual position. We also set the UV to vec2(vertexUV.s, 1.0- vertexUV.t);

texture.frag: Here we set the texture function with myTextureSampler first and UV as the second parameter, followed by (.rgba). This would be the number of textures we needed to output.

Results

The final output was a cube with numbers on all sides. The cube rotated on an axis continuously. An image of the final output is below:



References:

- https://github.com/andersonfreitas/opengl-tutorialorg/blob/master/tutorial14_render_to_texture/tutorial14.cpp
- https://learnopengl.com/code_viewer.php?code=model_loading/modelexercise1-shaders
- https://community.khronos.org/t/help-with-instanced-drawing/76346
- https://knowww.eu/nodes/59b8e93cd54a862e9d7e40e3
- https://github.com/JoeyDeVries/LearnOpenGL/blob/master/includes/learnopengl/camera.h