COSC 4370

Name: Robert Tumaliuan PSID: 0878829

October 29th, 2021

# 1 Problem

The assignment requires the replication of a given image using OpenGL's texture mapping skills. The image consists of a white cube floating in a black background. Numbers in large font are shown on two of its faces: an orange number 4 on the top face and a mirror image of a light blue number 6 on one of the cube's side faces.

# 2 Method

The functions used to replicate the image were provided by OpenGL.

For the main.cpp source code, the following functions were used: glGenBuffers(), glBindBuffer(), glBufferData(), glAttribVertexPointer(), and glEnableVertexAttribArray(), glActiveTexture(), and glBindTexture().

For the Camera.h header file, the function glm::lookAt(Position, Position + Front, Up) was used as a return value for the function glm::mat4 getViewMatrix(), similar to its implementation in HW3.

For the texture.frag fragment shader file, the texture() function was used to define the variable 'color'. The uniform sampler 2D 'myTextureSampler' variable as well as the vec2 input variable 'UV' were used as parameters for the function.

For the texture.vs vertex shader file, the vec4() constructor was used as one of the multiplication values to define the variable gl_Position. The vec3 input variable 'position' and the value of 1.0 were used as parameters for the vec4() constructor. The uniform mat4 variables 'projection', 'view', and 'model' were also used as values to define the variable gl_Position.

# 3 Implementation

The first step in implementation was modifying the Camera.h header file. In the TODO portion of the code, the glm::lookAt() function was added along with its parameters to the glm::mat4 getViewMatrix() function in the same manner as HW3.

Next, the texture.frag fragment shader file and texture.vs vertex shader file were also modified with the help of the LearnOpenGL web tutorial Getting Started: Textures and opengl-tutorial.org's Tutorial 5: A Textured Cube. In place of a solid surface color and a lighting position (as required in HW3), a texture was provided on this assignment. Both texture.frag and texture.vs were modified to accommodate its implementation, similar to how the phong.frag and

phong.vs shader files were modified to accommodate the color and lighting details for the cube in HW3.

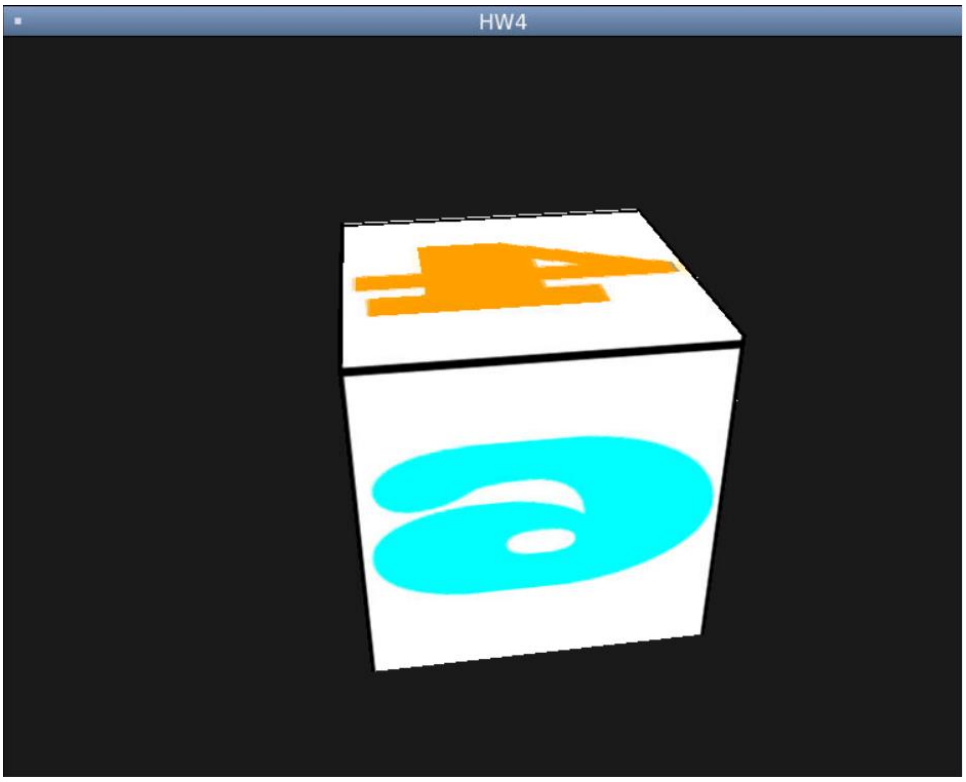
The only challenge encountered with this stage of implementation was the order in which the uniform mat4 variables model, view, and projection were to be multiplied for gl_Position. The comments in the texture.vs code were accurate in emphasizing how a black screen would show if this step was not done correctly. After a few moments of trial and error, the solution was multiplying the projection, view, model, and vec4 constructor in that exact order. The result was a gray cube rotating in a black background.

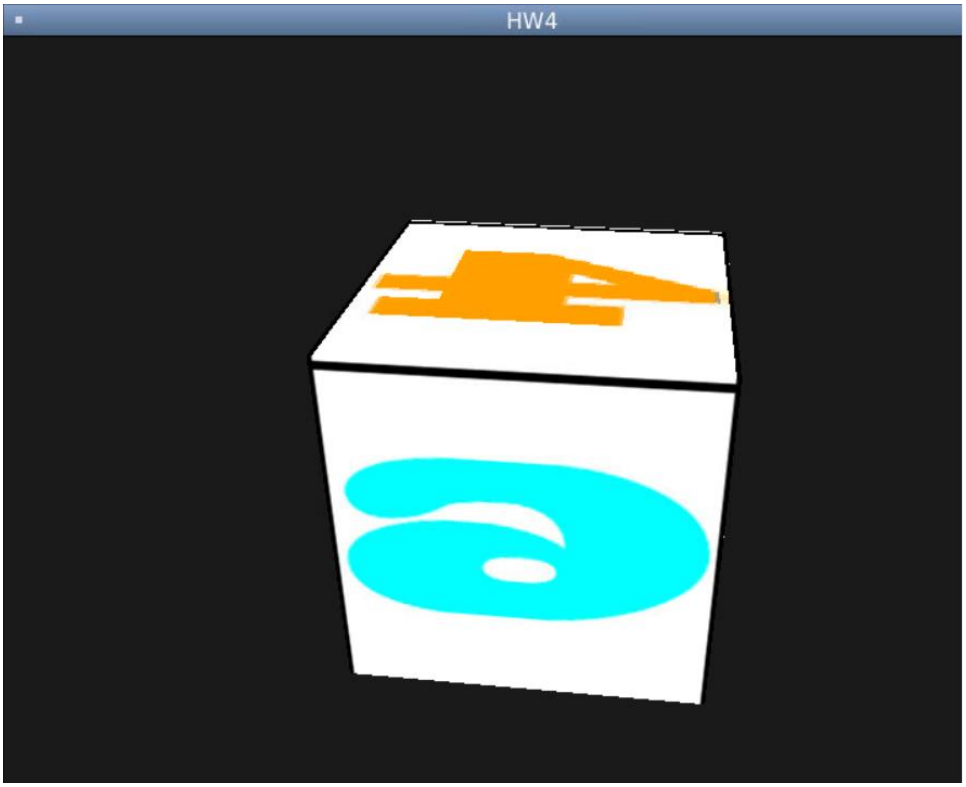
The next step in implementation was the modification of the main.cpp code for the setup of the UV buffer and the binding of the texture to the cube. Using the LearnOpenGl tutorial on Textures, the binding of the texture was first applied using the glActiveTexture() and glBindTexture() functions in the appropriate TODO segment of the code. This modification still resulted in a floating gray cube rotating in a black space, so the only step that still remained was setting up the UV buffer.

Another challenge was encountered involving this step of implementation. Much thought was put into this process only to be met with no success in how the buffer was meant to be properly implemented. From writing code based on loading BMP textures to using the glTexParameteri() function for mipmapping, the result was unsuccessful. Upon closer inspection of the code, the solution was in the way the vertex buffer was created as well as the ‘UVBO’ variable declared above the TODO segment. Similar to how the vertex buffer object (VBO) was created, the UV buffer was set up in the same manner, using the same functions as mentioned above in the **Method** section.

# 4 Result

After applying all the changes and modifications to the appropriate files, the result was a rotating textured cube in a black background, as shown in the images below.


HW4


HW4