Homework 4 – Texture Mapping

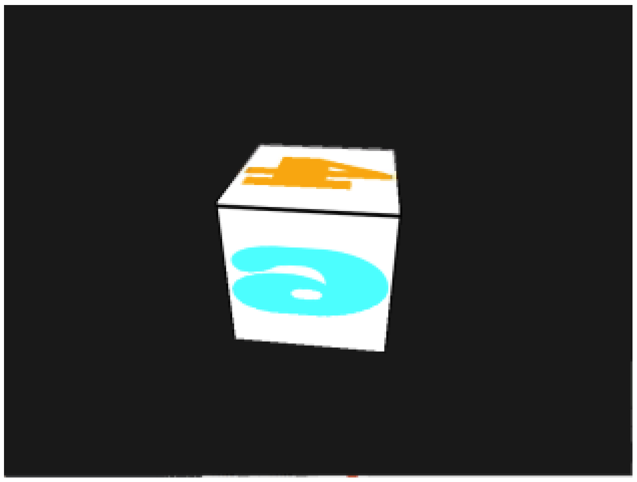
Daniel Nguyen

1479087

November 2022

1 Problem

The main problem of homework 4 is to replicate the given image on opengl using previous techniques and implementing new ones. The two files that needs changes are main.cpp and the two shader files texture.frag and texture.vs. Some of the changes includes, transferring the UV data to opengl buffer, bind texture in rendering loop and shader to draw the texture. This is the given image:



Each face having a separate number from 1 to 6.

2 Methods

Methods that are used in this homework are glGenBuffers, glBindBuffer, glBufferData, glVertexAttribPointer, glEnableVertexAttribArray, perspective, glBindTexture, and texture. These are commands that are used through this homework. Using these methods, we are able to replicate the given image.

2.1 References

http://www.opengl-tutorial.org/beginners-tutorials/tutorial-5-a-textured-cube/#inversing-the-uvs

https://learnopengl.com/Advanced-OpenGL/Cubemaps

https://learnopengl.com/Getting-started/Hello-Triangle

https://learnopengl.com/Getting-started/Textures

https://registry.khronos.org/OpenGL-Refpages/gl4/html/glVertexAttribPointer.xhtml

3 Implementation

glGenBuffers this generates a unique buffer ID, after we must bind the buffer using glBindBuffer, then we use glBufferData to copy user-defined data into the bound buffer. First, we create buffer IDs for UVBO then we bind GL_ARRAY_BUFFER and UVBO (the unique ID with the array.) lastly, we use glBufferData to copy the defined data to bound buffer, our parameters are GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW.

glVertexAttribPointer define an array of generic vertex attribute, it takes six parameters. The first is the attribute you are defining, second is the size of each vertex, third is the type, fourth is to normalize each vertex or not, fifth and sixth if there are stride or offset on the vertices. glEnableVertexAttribArray enables generic vertex attribute. This is used together with the function above glVertexAttribPointer.

perspective has been used in homework 3. This allows the user to see the camera's projection used in main.cpp. This works because the perspective uploads the matrix to the GPU each frame. The function takes four variables, field of view, screen size, near, and far. These parameters determine how the camera projects the image and allows us to see. In my case the parameters are radians(45.0f) being the FOV, (GLfloat)WIDTH / (GLfloat)HEIGHT calculating the screen size, 0.1F the nearest, 100.0f being the farthest the camera can see.

glBindTexture this function sets or "binds" the target object with the provided texture used in main.cpp to bind the texture. This function takes two parameters, the target, and the texture. In our case we provide myTextureSampler as the target and UV as texture which holds the texture.dds image.

Gl_position holds the position of the current vertex. This is important because if this is implemented incorrectly the cube will not be visible. To calculate the position, we used the following projection * view * model * vec4(position, 1.0). Also, I had to invert the vertexUV, because if I left it as is then the numbers would have overwritten each other. In order to prevent that we must invert vertexUV.

texture is a built-in texture function in GLSL used in shader. The texture function samples the corresponding color value using the texture sampler and corresponding texture coordinates. This function takes two parameters, a texture sampler and corresponding texture coordinates then outputs the color of the texture in the texture coordinates. The sampler in this case would me myTextureSampler and the coordinates would be UV.

4 Result

Using all the methods mentioned above I was able to output a 3D cube with texture mapped on it, shown below. Implementing the project matrix allows us to see from the eyes of the camera, using the vertices given in main.cpp we are able to create a cube. Using the dds image file that was given we can map that onto the cube using the methods mentioned above. Each face of the cube has a number from 1 to 6.