Homework 1 - Interactive Graphics

Veronica Romano - 1580844

June 13, 2020

1 Introduction

To realize this homework I had to start from the code provided by the professor and later to modify it to realize my personal object and to implement several tasks:

1. Creation of an object (max 20-30 vertices) and association of normal and texture coordinates.
2. Adding of a viewer position and of a perspective projection
3. Adding of a directional light and of a spotlight
4. Assignment of object material properties
5. Implementation of a per-fragment shading model
6. Adding of a texture from file

2 Implementations of Tasks

2.1 Creation an Coloring of the Figure

First of all I started by choosing the object to represent, in my case will be a modified shelf, inspired by the shelf in my room (Figure 1). To have the minimum number of vertices, I had to modify its sides that were circular, by making them pentagonal. I reached 20 vertices. However I had to add 4 additional vertices to make the coloration of all sides of my figure, in the way to use the *gl.TRIANGLES* method, in particular for the boundaries of the shelf. Basing on the sketch of the real shelf, I measured the real dimensions of the shelf, and then I created the coordinates of the vertices normalizing them in the range [-1,1], considering that the center of the figure was in the center of my reference frame. In this way the rotation around y axis of an theta angle, results also around the center of the figure. After creating the vertices and positioning them in the reference frame, I colored my shelf with the *gl.TRIANGLES* method. I modified the function quad() provided by the professor by adding a fifth parameter, to set the color of my sides. The color comes from *vertexColors[...]* variable. So in *colorCube()* function I recall the quad() function with 5 indices; the first four were the indices of the vertex of a specific side in *vertices[]*, the last was the color. For the pentagonal sides, because I used the *gl.TRIANGLES*, I

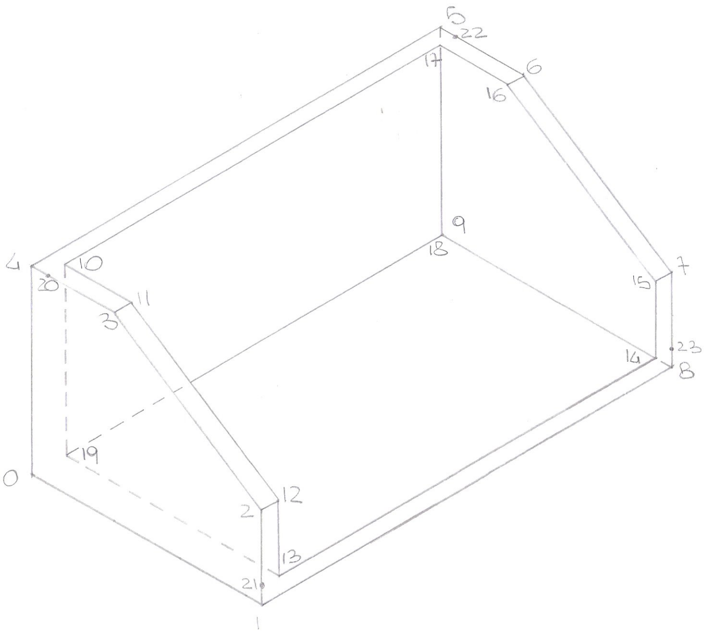

Figure 1: Model of the shelf

created 2 quadrilateral to coloring them separately. While for the boundary of the shelf, I had to create 4 additional vertices to use in such a way to making the sides as rectangles.

2.1.1 Adding Texture and Normal Coordinates

To realize this task I added a new *texCoord[]* variable, to use for the next sixth task, and I also created *texCoordsArray = []* to use in the *quad()* function in which I placed the texCoord. Later I compute for each sides the normal, always in the quad() function, that I used for the second task and a vector *normalsArray = []* in which I placed the normals computed.

2.2 Viewer Position and Perspective Projection

I proceeded with the realization of the perspective and viewer position through the creation of ModelView and Projection matrices. I used the normals created in the previous point to create a *ModelView* matrix by using the *lookAt()* method and the *projectionMatrix* matrix by using *perspective()* method. From these I created *modelViewMatrixLoc* and *projectionMatrixLoc* in the javaScript file, recalling them in the html file by using *uModelViewMatrix* and *uProjectionMatrix* that are the uniform variables whose locations are to be returned. I created also buttons to control them.

2.3 Directional Light and Spotlight

I created a directional light and a spotlight. For the directional light I chose the position in the way that the w coordinate was 0.0, so the xyz coordinates constitute the direction of the light. I defined it in the vertex-shader by using all the parameters as ambient (where I defined the color of the light), specular and diffuse coefficients. I also created a button to manage the light on and the light off. While for the spotlight, I implemented it in the fragment-shader, always by using all the parameters described in the javaScript file. I defined a position and direction for it, and I created a button to manage the on/off and its cutoff. As you can see from image, the spotlight is reflected on the surface of the shelf, and also the directional light creates shady areas.

2.4 Material Properties

For my object I chose the ambient, specular and diffuse material properties and also a shininess coefficient that I also used in the computation of the reflection of the lights, to render how the lights appear on a specific material. To do this I used the Phong reflection model, calculating the coefficients k_s and k_d for the specular and diffuse light, plus other parameters like the normal of each surface that I computed in the vertex and fragment-shader.

2.5 Per-fragment Shading Model

I implemented this shading model as a semplified version of that described in the article[1] of Lake, A., in the fragment-shader as requested. I created in the fragment-shader, both for the directional light and for the spotlight, the algorithm by calculating the illuminated diffuse color and the shadowed diffuse color, with respect to if the max[L·n, 0] be >= or < of 0.5, where n is the normal to the surface and L is the normalized vector from the vertex to the light source position. In the first case I calculated the coefficient C_i so with the adding of the diffuse component given by the product $d_l \times d_m$. In the other case this component there isn't, so I created a shadowed area without the diffuse component of the lights.

2.6 Adding a Texture from File

To implement this task, I used an image provided by the professor from the codes of our textbook, in particular this is *honolulu4.js* utilized in the chapter 8. I utilized the texture coordinates to create in the cube() function, to create the *texCoordsArray.* I applied and implemented the texture in the javaScript file, I created also a button to activate and deactivate the texture by using a flag, and in the fragment-shader I rendered the texture through the variable *fColor.*

[1] Lake, A., Marshall, C., Harris, M., & Blackstein, M. (2000, June). Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings of the 1th international symposium on Non-photorealistic animation and rendering* (pp. 13-20).