Interactive Graphics Homework 1

Michal Ostyk-Narbutt (1854051)
Prof. Marco Schaerf

May 2, 2020

SAPIENZA
UNIVERSITÀ DI ROMA

1 Introduction

This is a documentation report describing the techniques used in the First Homework for the Interactive Graphics course. Given a baseline file of a cube, the task was to modify it by

- expanding the number of vertices (20-30) with each having a normal and texture coordinates.
- adding a viewer position and a a perspective projection
- computing the ModelView and Projection matrices in the Javascript application
- adding two lights sources
    1. Directional
    2. Spotlight
- assigning to the object a material with the relevant properties
- Implementing a per-fragment shading model
- adding a texture loaded from file, with the pixel color a combination of the color computed using the lighting model and the texture.

2 Documentation

2.1 Creating an irregular object

My idea was to create some octagon Hourglass as seen in Figure 1. In order to accomplish that task I decied to create out of a series of triangles (`tri function`) and quadrilaterals (`quad function`). However, unlike in Figure 1, mine would have a more visible inner part.

Figure 1: Octagon Hourglass [1]

I modeled the vertices based 4 faces of different widths, with the top and bottom sharing one value for the width, and the inner "neck" of the hour composed of two octagons also sharing one value. To accomplish this task I decided to run the following python code which allowed me obtain various initial sizes.

```
def getshapevalues(height_value, vertices=8, width=1):
    coords = []
    for ind, value  in enumerate(range(vertices), start=1):
        x = cos(2*value*pi/vertices) * width
        y = sin(2*value*pi/vertices) * width
        coords.append([x, y])
       return coords
```

Listing 1: Getting values for the vertices

The code snippet in Listing 1 in Python is just a visual version of obtaining a polygon coordinates. These in reality would be checked for value and using strings formatted into JavaScript ready code. The result is show in Listing 2 below.

```
vec4(0.6, 0.0, -1.0, 1.0), // point 0
vec4(0.3*Math.sqrt(2), 0.3*Math.sqrt(2), -1.0, 1.0), // point 1
vec4(0.0, 0.6, -1.0, 1.0), // point 2
vec4(-0.3*Math.sqrt(2), 0.3*Math.sqrt(2), -1.0, 1.0), // point 3
vec4(-0.6, 0.0, -1.0, 1.0), // point 4
vec4(-0.3*Math.sqrt(2), -0.3*Math.sqrt(2), -1.0, 1.0), // point 5
vec4(0.0, -0.6, -1.0, 1.0), // point 6
vec4(0.3*Math.sqrt(2), -0.3*Math.sqrt(2), -1.0, 1.0), // point 7
vec4(0, 0, -1.0, 1.0), // point (centroid) 8
```

Listing 2: Bottom Octagon example vertex defintions

Hence, using this procedure, from the four octagon faces, and inner points of the bottom and top face, a total of 34 vertices (inner octagon centroids were not needed). The points were connected using.

2.2 ModelView and Projection matrices

Next, I added a viewer position, a projection and computed the ModelView and Projection matrices using a Javascript application. The ModelView was computed using the `lookAt` function which can concatenate with modeling transformations. Subsequently, sliders were added to the HTML file, which include the following transformations: radius, theta, phi, Near, Far, Scale, FOV (field of view), aspect. Additionaly for animation and asthetic reasons enabled buttons which through automatic increments of theta, and appropriate rotation matrices allow the rotation of the hourglass around the roll pitch yaw angles. Moreover, the shape can be slided along each o the X, Y, and Z axes.

2.3 Lighting: Directional and spotlight

TO DO, make x y and z sliders work change it up a bit

2.4 Material

descrive the random material, get rid of the specular

2.5 per-fragment shading model

equations

2.6 Texture from file

Taken from the internet, to simulate the surface of the moon.

3 Conclusions

3.1 Advantages

3.2 Disadvantages

References

[1] `http://evershinegift.com/Index.asp?Product420/742.html`