Interactive Graphics Homework 1

Michał Ostyk-Narbutt (1854051)
Prof. Prof. Marco Schaerf

April 27, 2019

SAPIENZA
UNIVERSITÀ DI ROMA

Contents

1 Introduction

This is a documentation report describing the techniques used in the First Homework for Interactive Graphics.

2 Documentation

2.1 point 1

Starting with the first point of the homework, 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, in this case an isometric view of a cube. Subsequently, sliders were added to the HTML file, which include the following transformations:

- radius
- theta (angle of rotation)
- phi (angle of rotation)
- fov - field of view using the perspective function
- aspect

2.2 point 2

In this point I introduced scaling, which scales ('scalem' function) the desired cube uniformly in size. Also, a translation ('translate' function') matrix, in all directions X, Y, and Z was implemented controllable by sliders. These matrices are stored in the JS file, then are sent to the vertex shader using GL's 'uniformMatrix4fv' function, which then leads an update in the overall render function.

2.3 point 3

In point 3, we were tasked with defining an orthographic projection with the planes near and far controlled by sliders. For this, I utilized orthogonal projection matrices, which were subject to normalization. This allows to convert all projections to orthogonal projections with the default view volume, instead of deriving a different projection matrix for each type of projection. For this, the 'ortho' function was used which applies normalization which finds a transformation to convert specified clipping volume to default. This is done in two steps, by moving center to the origin and then scaling to have the sides of length 2.

2.4 point 4

In this part, I split up the windows into 2 parts, which one showing the orthographic projection meanwhile the second shows perspective projection.

2.5 point 5

In this point, the task was to a light source, and replace the colors by the properties of a material and also to assign to each vertex a normal. Light from a light source is needed in order to shade objects so their images appear three-dimensional. This is because real objects do not appear uniformly colored, as light-material interactions cause each point to have a different color or shade. To introduce a light source, we simply use place its location source in the system using the 'getUniformLocation'

function. To replace colors by a material, the shininess coefficient which in my homework is set to 100 which is a metal. Then in the 'quad' function defined in my JS file, I assign ('push') to each vertex a normal.

2.6 point 6

In the this stage, I set about implementing two models, the Phong, and the Gouraud and shading models (setting up a button to switch between them). First the Phong model has three components (material properties) Diffuse, Specular, and Ambient. As well as four vectors: To source, To viewerm Normal, Perfect reflector. In code terms, ...

Then the Gouraud shading is made up following steps; Finding the average normal at each vertex (vertex normals), applying modified Phong model at each vertex and last of all Interpolating vertex shades across each polygon.

2.7 point 7

"Add a procedural texture (your choice) on each face, with the pixel color a combination of the color computed using the lighting model and the texture"