Interactive Graphics
Assignment 1

A.Y. 2017-2018

SAPIENZA
UNIVERSITÀ DI ROMA

Hossam Arafat
1803850

Introduction

This report describes the process of building a simple 3D interactive graphics project using HTML, JavaScript and WebGL. The application began with a spinning colored cube with 4 buttons. We describe below a solution that we implemented to add several features to it.

Solution

It was required to add a button that changes the direction of the cube's rotation. This was done using a simple "button" HTML tag. The change of rotation direction was implemented using a simple flag and an if-condition in the render function of the JavaScript application the incremented or decremented the value of theta accordingly.

Model-view and Projection matrices calculations were moved to the JavaScript application. The advantages of this is that computing the transformations for every vertex was a huge waste of time and computation. The Model-view matrix was implemented using the "Look At" function of the "MV.js" file.

Moreover, it was required to add sliders that can be used to scale the cube and translate it along the X, Y, and Z axes. The Sliders were added using a simple "input" HTML tag of the "range" type. Rotation, Translation, and Scaling of the cube were implemented by building Homogenous transformation matrices using the simple "rotate", "translate", and "scalem" JavaScript functions that are found in the "MV.js" file and multiplying set matrices to the Model-view matrix using the "mult" function inside the JavaScript application.

Orthographic and perspective projection were implemented using 2 functions that are found in the "MV.js" file, "ortho" and "perspective" respectively. A simple button switches between the 2 kinds of projection. Moreover, two sliders were introduced to control the viewing volume, specifically the location of the "Near" and "Far" planes. The same 2 sliders work for both kinds of projections.

The next requirement was to implement both the Gouraud and the Phong shading models. To implement these models, we need to consider the interactions between light and the different surfaces in the scene. This can be broken down into 2 independent sub-problems:

1. Modelling the Light source
2. Modelling the interactions between the different Objects / Materials and light.

The first sub-problem / step towards building a shading model was solved by introducing a “distant” or directional white-colored point light source and specifying its intensity and position co-ordinates, as well as a more “gray” ambient light that illuminates the entire scene. Since the shading of objects depends on the orientation of their surfaces with respect to the light source, we compute a normal vector to each vertex inside the “quad” function of the JavaScript application by calculating the cross product of 2 vectors that lie on a face and normalizing it.

The second sub-problem / step was tackled by removing the colors of the rotating cube and replacing it with a “chrome” material. The values that specify the properties of the material –how shiny, diffuse, and specular the material itself is and how much of the ambient light it reflects- were acquired from an open-source project on the web entitled “White House 3D”.

Both the Gouraud and Phong shading models were implemented in the Vertex and Fragment shaders respectively. The main difference between both models is that the Gouraud shading computes the color between each vertex and interpolates it while the Phong shading model interpolates the normal across the triangle and compute the color per fragment (or pixel).
We multiply the ambient, diffuse, and specular properties of the light with the same properties of material and take the shininess of the material as well as the light position from the JavaScript application and send them to the shaders.

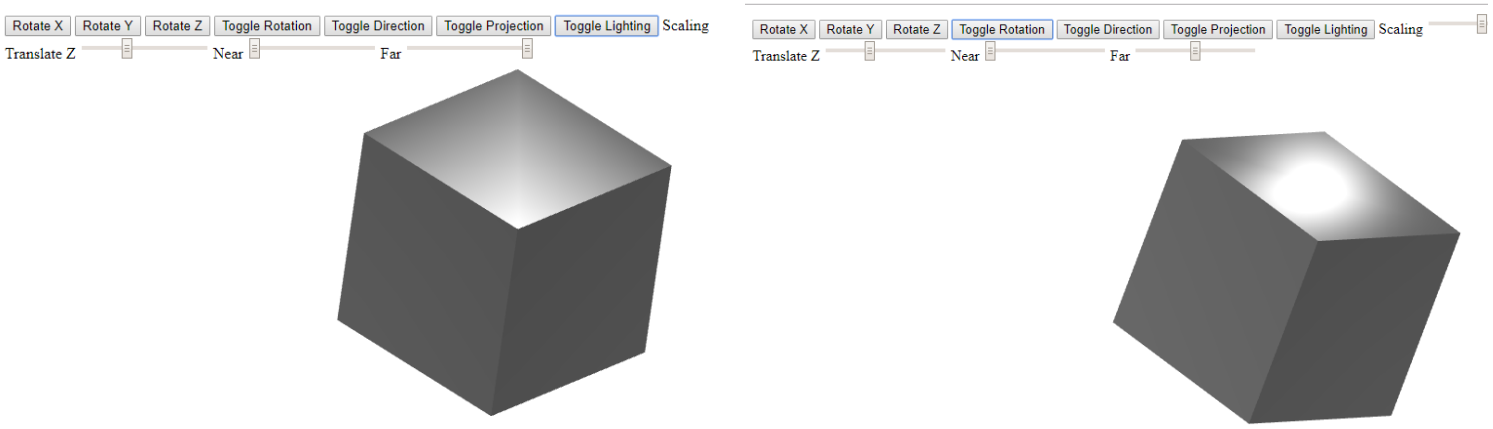

Fig 1: The difference between the Gouraud Shading and the Phong Shading model is very clear when we bring the cube closer to the light source.

Results

When selecting the perspective projection mode, the cube correctly looks closer to a parallelogram and gets smaller or larger as it is translated further away from, or, closer to the viewer. Moreover, you can see the object getting considerably lighter or darker as we translate it in any direction that is closer to, or, further away from the light source. All of these effects are desirable and speak to how realistic our model is.

This solution has the main advantage of being flexible and efficient. It correctly and efficiently implements different kinds of shading models, different kinds of projections, and allows one to readily choose between them while also having full control over the position, orientation, and rotation direction of a 3D cube in a simple and efficient manner.

The application can be improved by introducing a CSS stylesheet that organizes the elements on the webpage and gives them a more “modern” look.

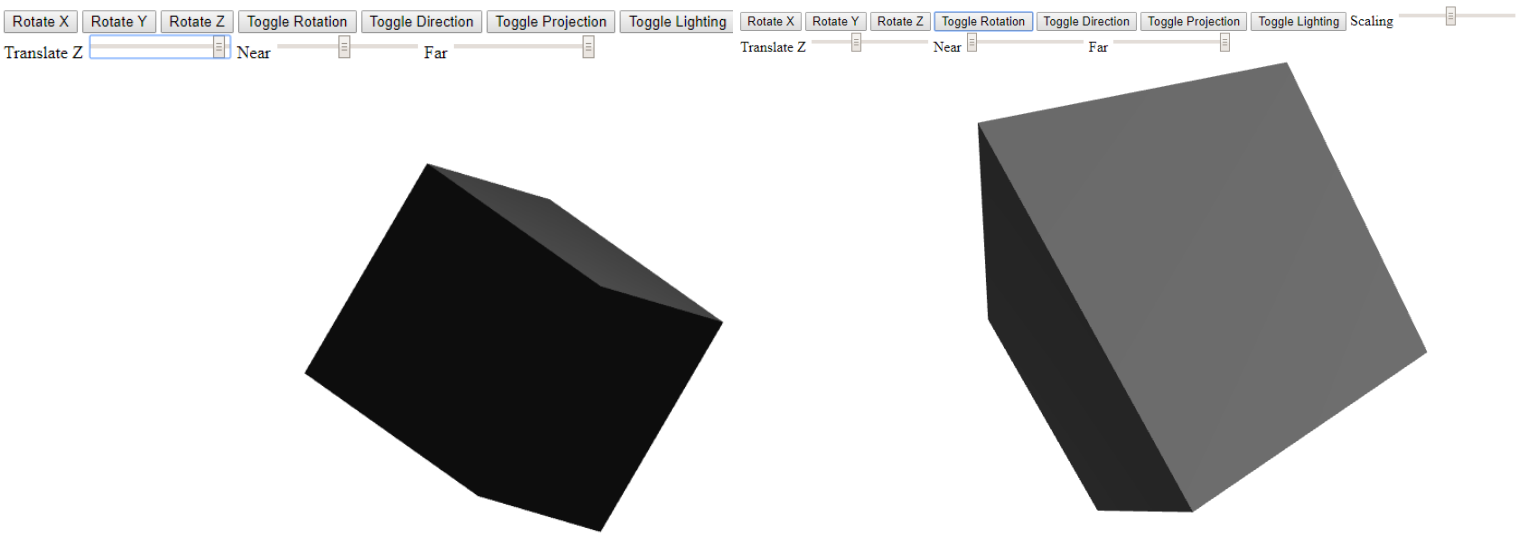

Fig 2: Cube in orthogonal projection far from the light source [left image] and cube in Perspective projection closer to the light source [right image]

The effect of the Perspective Projection as well as how the object becomes lighter or darker as we move it closer or further away from the light source is quite clear. Notice how because the light source is in the top right corner and with a Z position “out of the screen”, some of the faces of the cube that are facing the light are illuminated (lighter) while others are in shade (darker).