INTERACTIVE GRAPHICS HW1

Ahmed El Sheikh
1873337

1. Setup

I created the cube to rotating -by increasing/decreasing value of theta- around X-axis by default, however, this can be manipulated using different buttons specifying the axis of rotation, as well as, toggling rotation, all was implemented inside the `render()` function.

Model viewing and projection calculations were implemented using `lookAt()` functionality implemented in one of the scripts provided `MV.js`

Viewer position which is referred to as `eye` is controlled in three different axes, which are needed to calculate the model view matrix.

2. Scaling & Transformation

Scale the cube and translate them, as requested was done using range sliders, and the corresponding values -as per slider- were used by the transformation matrices, the ones I constructed using `rotate()`, `translate()`, `scalem()`, of course to perform rotation, translation, scaling. All of them were multiplied to CTM using `mult()`.

All of the mentioned functions are from `MV.js`, as well.

3. Projections

- As per the requirements, I implemented 2 projections perspective `perspective()`, as well as, orthogonal `orth()`. Also found in `MV.js`

- Added to range sliders to control `Near` & `Far` planes, not to mention that, they control both projections

4. Different Projections

To project two different views -each in a separate half-, I needed to split our canvas vertically which was done using ‘gl.scissor()’ with ‘gl.viewport()’ hand in hand. Each of the projections are controlled using the same range sliders.

5. Material & Lighting

I introduced directional white colored point light source, specifying its intensity and position, in addition to, the ambient light illuminating the whole canvas. Object shading depends on the orientation of the faces with respect to the light source, thus, I needed to compute normal vector as per each vertex inside the function named `quad()` which basically calculates the cross product of 2 vectors laying on the same face.

6. Shading

I used two shading models Phong & Gouraud, both were implemented using Vertex and Fragment shaders. Phong interpolates the normal across the triangle and compute color as per fragment unlike Gouraud, which computes the color between each vertex and interpolates it.

How I applied the different shading models? By multiplication of ambient, diffuse, specular properties of light with properties of material, as well as, taking into consideration both shining property of the material, added to, light position.

7. Final Overview