VALERIO FRANCIONE

2047712

1)

In order to realize the chair, I defined the new points(40 in total) used to create other cubes and configure the coordinates in order to obtain the requested figure, I have also increased the value of the positions at each cube(+36 each one). To calculate the normals I simply used that method:

```
var t1 = subtract(vertices[b], vertices[a]);
var t2 = subtract(vertices[c], vertices[b]);
var normal = cross(t1, t2);
normal = vec3(normal);
```

that is the representation of the mathematical formula to calculate the normal vector.

2)

In order to rotate the figure I have first of all created the variables of the axis and after initialized the rotation matrices which was multiplied to the position of the chair. In order to change in real time the rotation I have added the buttons with the appropriate listener which are going to change the value of the axis.

3)

```
//prospective
var near = 1.0;
var far = 4.0;
var radius = 3.0;
var phi = 0.0;
var thetaP = 2.5;

var  fovy = 80.0;
var  aspect = 1.0;

var modelViewMatrix, projectionMatrix;
var modelViewMatrixLoc, projectionMatrixLoc;
var eye;
const at = vec3(0.0, 0.0, 0.0);
const up = vec3(0.0, 1.0, 0.0);

eye = vec3(radius*Math.sin(thetaP)*Math.cos(phi), radius*Math.sin(thetaP)*Math.sin(phi), radius*Math.cos(thetaP));

modelViewMatrix = lookAt(eye, at, up);
projectionMatrix = perspective(fovy, aspect, near, far);
```

In order to create the prospective, I have setted the parameter as below,I am going to calculate the view of the spectator. With the slider we are going to change that parameters and recalculate the matrices with the newest one.

4)

```
//Light
var hx=0.0;
var hy=3.0;
var hz=0.05;
var lightPosition = vec4(hx, hy, hz, 1.0);
var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0);
var lightDiffuse = vec4(20.0, 20.0, 20.0, 1.0);
var lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);
var lightDirection = vec3(0.0, -1.0, 0.0);

var angle = 5;
var attenuation = 20;
```

To add the spotlight that are the parameters. I am going to pass that to the html file, and here use it to calculate the lights as follow:

```
vec3 L;
if(uLightPosition.w == 0.0)  L = normalize(uLightPosition.xyz);
else  L = normalize((uLightPosition - vPosition).xyz);

vec3 E = normalize(uEyePosition);

// Transform vertex normal into eye coordinates

vec3 N = normalize((uModelViewMatrix*vNormal).xyz);

vec3 H = normalize(L + E);

vec4 ambient = uAmbientProduct;

float Kd = max( dot(L, N), 0.0 );
vec4  diffuse = Kd*uDiffuseProduct;

float Ks = pow(max(dot(N, H), 0.0), uShininess );
vec4  specular = Ks * uSpecularProduct;

if( dot(L, N) < 0.0 ) {
    specular = vec4(0.0, 0.0, 0.0, 1.0);
}

if(dot(L, normalize(-lightDirection))>= angle){

        float spot = pow(dot(L, normalize(-lightDirection)), attenuation);
        float fPercentage = 1.0 - percentage;
        fColor =  percentage*vColor + fPercentage*(ambient + spot*(diffuse + specular));
        fColor.a = 1.0;

}else{
        fColor = ambient;
        fColor.a = 1.0;
   }
```

As we can see, in order to render the light I am going to add it to the fColor (in a similar way in the vertex shader).

5)

As material I selected the gold because I liked it. The parameters are setted as follow:

```
//Material
var materialAmbient = vec4(0.24725, 0.1995, 0.0745, 0.4);
var materialDiffuse = vec4( 0.75164, 0.60648, 0.22648, 0.4);
var materialSpecular = vec4(0.628281, 0.555802, 0.366065, 0.4);
var materialShininess = 20.0;
var ctm;
var ambientColor, diffuseColor, specularColor;
```

6)

To obtain the selection between pre-Vertex and pre-Fragment, I have created a variable called “percentage”, that derives from the slider and multiplied with the vertex and fragment variable gives us the quantity of each one. Obviously in order to obtain the mix of the two I have added in the fragment variable also the value of the vertex shader (as we can see in the image in point 4).

7)

In order to implement the quantization, I have changed the value of “fColors” with the closest color of the selected one by the user. The algorithm implemented is this one:

```
if(quantization){
    int indice=0;
    float dist = distance(colors[0], fColor);
    for(int i=1; i<8; ++i){
        if(distance(colors[i], fColor) < dist){
        dist = distance(colors[i], fColor);
        indice = i;
        }
    }
    fColor =  colors[indice];
    fColor.a = 1.0;
}
```

The user can modify each parameter (r,g,b) for each color by sliders.