SAPIENZA
UNIVERSITÀ DI ROMA

SAPIENZA UNIVERSITY OF ROME

INTERACTIVE GRAPHICS

Homework 1

Report

Worked by: Alkid Halili
Matricola : 1956856

Excercise 1:

My object is composed by 20 vertices and to draw it I used the GL.TRIANGLES function to which, in order to draw, I pass 3 points for each triangle.

Then I create two texture images and the normal are calculated by dot product of 2 vectors that create the triangles so I could get the normal of the texture.

Excercise

Here I am doing barycenter calculation and rotation around 3 axes

I first start by calculating it for each of the 3 triangles in the faces of the object. I proceed by creating triangles from the barycenters that I found and then repeat again till I converge to 3 cases:

1 barycenter, 2 barycenter (here I do barycenter of segment),3 barycenter (in this case I do barycenter of triangle that is created by these 3 vertices) depending on the object.

To calculate it I have added 2 buttons that enable and disable the calculation.

Excercise 3:

For perspective projection I calculate first the model view matrix by adding the position of the viewer then passing this position as “at” position for the “lookAt” function.Using “phi”,” theta” and “radius” I also calculate the “Eye”.
Then I calculate the projection matrix by passing the parameters Field of view, screen aspect ratio, near and far. In my calculation I have also used the hidden surface removal method.

Excercise 4:

For the neon light I am using the library geometry.js and I am specifying the parameter of my choice and inside it I have out three lights each one with the specified emissive color. I have added ambient ,specular and diffuse light for object and the same also for the material.
Then I am calculating the normal array, position array, tangent array and passed and used this for shading calculation.
Then after having the surface normal and the direction vectors the color for each of the vertices is updated. In order to get the RGB value for the my object (both vertices and fragments) I just do a dot product of the Normal matrix with the direction vectors.

Excercise 5:

I assigned the material properties of my object, (ambient,specular,diffuse lights and the shininess of it).

Ambient lighting- contribution from the white light bouncing off multiple surfaces in the room, that is almost the same at every point in the room.

Specular lighting identifies the bright **specular** highlights that occur when **light** hits an object surface and reflects back toward the camera. **Specular lighting** is more intense than diffuse **light** and falls off more rapidly across the object surface.
I have added shinines to my geometry just to give a small reflection on the surface.
These properties will be crucial for the per-fragment.per-vertex shading and also for the texture mapping, because the differences between these will be more visible by these lights.

Excercise 6 :

I have done both Per-vertex and Per-fragment shading, which is basically the calculation of the light.

Per-vertex-shading

I sent the properties that I calculated before to the vertex shader and I got the calculation of all the object's shades. The calculations are done considering the viewer perspective.

The vertex shading is not smooth and it can be easily detected in the object when it is activated.

The computations of Is,Id and Ia are performed in vertex shader so when we process the vertex we also compute the color for it .

Vertex colors become vertex shades and can be sent to the vertex shader as vertex attribute .I have sent positional viewer in position of the light and then I have computed the vectors in the shaders .

The color is computed in vertex shader and in this case the rasterizer will compute different colors for each fragment on them the interpolation of the the color of vertices is done so that the color of each fragment is generated.

Per-fragment shading

The lighting computations are done in the fragment shader. The fragment shader needs the position of the fragment, the light's position (or the direction to the light from that position), and the surface normal of the fragment at that position and calculated the shades for each fragment.

I have proceded like this :

1-find the vertex normals

2-interpolate vertex normals across edges

3-interpolate edge normals across polygons

Normals are computed in vertex and computation of color is done at fragment shader.

If the light source and viewer are relatively far from the surface, so
that each triangle makes only a small contribution to the rendered image, we might not notice a significant difference between per-vertex and per-fragment shading

By looking at the object it can be easily seen that the fragment shading looks more smooth compared to the vertex shading which shows more edges

Excercise 7:

Bump mapping is a texture mapping technique which we use for creating bumps and wrinkles on the surface of an object.

To achieve it I have perturbated the surface normals of the object and using the perturbed normal during lighting calculations. The result is an apparently bumpy surface rather than a smooth surface although the surface of the underlying object is not changed.

In my case, the object I have chosen makes it difficult to reach a good bump since the direction of the normal is the same for the a good part of the triangles, but at least the texture is quite obvious.

The Bump mapping can be activated by the button that I have added.