Lighting and Shading

HIMANSHU RAJ (2022216), Indraprastha Institute of Information Technology Delhi, India

1 GENERATE NORMALS AND MAP TO COLORS

The provided code generates vertices of a sphere by discretizing angles in spherical coordinates.

1.1 Calculating Normals

In spherical coordinates, the position of a point on the surface of the sphere is given by

$$x = r \sin\theta \cos\phi, \quad y = r \sin\theta \sin\phi, \quad z = r \cos\theta$$

The normal to any point on the surface of the sphere is the vector from center of the sphere towards the point. We can find normalized normals by dividing this normals vector by its magnitude as

$$\vec{N} = \frac{\vec{P}}{||\vec{P}||}$$

1.2 Normal to Color mapping

The generated normals are in the normal range of [-1, 1]^3. We map them to RGB color range of [0, 1]^3 as per the mapping c = (n+1)/2, where n is the normal vector.

Fig. 1. Sphere with normals mapped to RGB

Author's address: Himanshu Raj (2022216), himanshu22216@iiitd.ac.in, Indraprastha Institute of Information Technology Delhi, India.

2 LIGHTING AND SHADING WITH A POINT LIGHT SOURCE

2.1 Add a Fixed Point Light Source

In OpenGL, a point light source is identified by the location of the light source and its color. I have hardcoded the color components of light in shaders itself and passed the value of position of light from the C++ program.

2.2 Gouraud shading

We use Phong lighting to render the object with specular highlights with Gouraud shading, where per-vertex lighting is calculated and interpolated across the surface of the object. I have calculated the three associated lighting components (ambient, diffuse and specular) in the vertex shader where values of material coefficients are hardcoded. Since we have added an ambient component, the back side of the sphere is not pitch dark and has some ambient lighting.
The lighting equation is $I = I_{ambient} + I_{diffuse} + I_{specular}$

(a) (b)

Fig. 2. Gouraud shading using Phong lighting

2.3 Phong shading

We do Phong shading this time where per-fragment lighting is calculated rather than vertices which leads to smoother lighting effects. The lighting computation is done in fragment shader. Lighting equation is the same as the one in Gouraud shading. The shade is better than what we got from Gouraud shading.

(a) (b)

Fig. 3. Phong shading using Phong lighting

3 LIGHTING AND SHADING WITH A SPOTLIGHT

3.1 Spotlight implementation

A spotlight is a light source that emits light in a cone of directions. As a result, only objects within the cone of illumination are lit. In OpenGL, spotlight is represented with a position, direction and cutoff angle (half cone angle) that defines the cone of illumination.
For each fragment, we find θ which is the angle between fragment-to-light and spotlight direction vectors. If θ is greater than the cutoff angle, the fragment is inside the spotlight cone, so this fragment has all 3 lighting components. If θ is smaller than the cutoff angle, the fragment is outside the spotlight cone, so this fragment has only an ambient component of light.

Fig. 4. Spotlight with hard edges illuminating a part of the sphere's surface

3.2 Spotlight with soft edges

We observe hard edges on the boundary of the illumination cone, so we implement soft edges by creating an outer cone as well as an inner cone. The intensity of a fragment falling in between these two cones drops down from 1 to 0 in a smooth fashion.

Fig. 5. Spotlight with soft edges illuminating a part of the sphere's surface

4 SPOTLIGHT WITH A HEADLIGHT

A headlight is a light source that moves (and is positioned) along with the camera. This is done by assigning the camera's gaze and spotlight in the same direction. This ensures that the region we are currently observing is always illuminated. To implement this, we need to update the spotlight position as we move around the camera position inside the rendering loop.

(a) (b)

Fig. 6. Spotlight with a Headlight