CG Assignment- 3

Rahul Oberoi (2021555)

Answer 1.

In Part 1 of the assignment, we are tasked with implementing Phong shading on a parametric surface. The objective is to create a visually appealing rendering of an object with realistic lighting effects. To achieve this, we will use a parametric surface and perform shading calculations using vertex shaders.

<u>Steps and Implementation:</u>

1. Choose a Parametric Surface:

We started by selecting a parametric surface of our choice. In our code, we used a sphere to demonstrate the application of Phong shading. We defined this sphere's geometry as a set of vertices and normals.

2. Create Vertex Shaders:

We modified the existing code to include vertex and fragment shaders. Vertex shaders perform calculations on the vertices of our parametric surface, and we used the following key shader components:

- vVertex: This variable holds the vertex positions of the object.
- vNormal: We computed per-vertex normals and passed them to the fragment shader.

3. Calculate Per-Vertex Normals:

To compute per-vertex normals, we utilized the method described in the assignment using partial derivatives. By taking the partial derivatives with respect to the surface parameters, we obtained the normal vector for each vertex. These normals are crucial for realistic lighting calculations in the Phong model.

4. Implement Phong Lighting in the Vertex Shader:

Our vertex shader performed lighting calculations based on the Phong lighting model. We computed ambient, diffuse, and specular lighting components for each vertex. Key shader variables and calculations included:

- ka, kd, ks: Material coefficients for ambient, diffuse, and specular components.
- La, Ld, Ls: Light source coefficients for ambient, diffuse, and specular components.
- spec_exp: Phong exponent, controlling the size of specular highlights.
- Normalization of the normals, vectors, and reflectance.

5. Pass Values to Fragment Shader:

In the vertex shader, we passed the computed lighting values and normalized normals to the fragment shader. The fragCol variable was used to store the computed light color for each vertex.

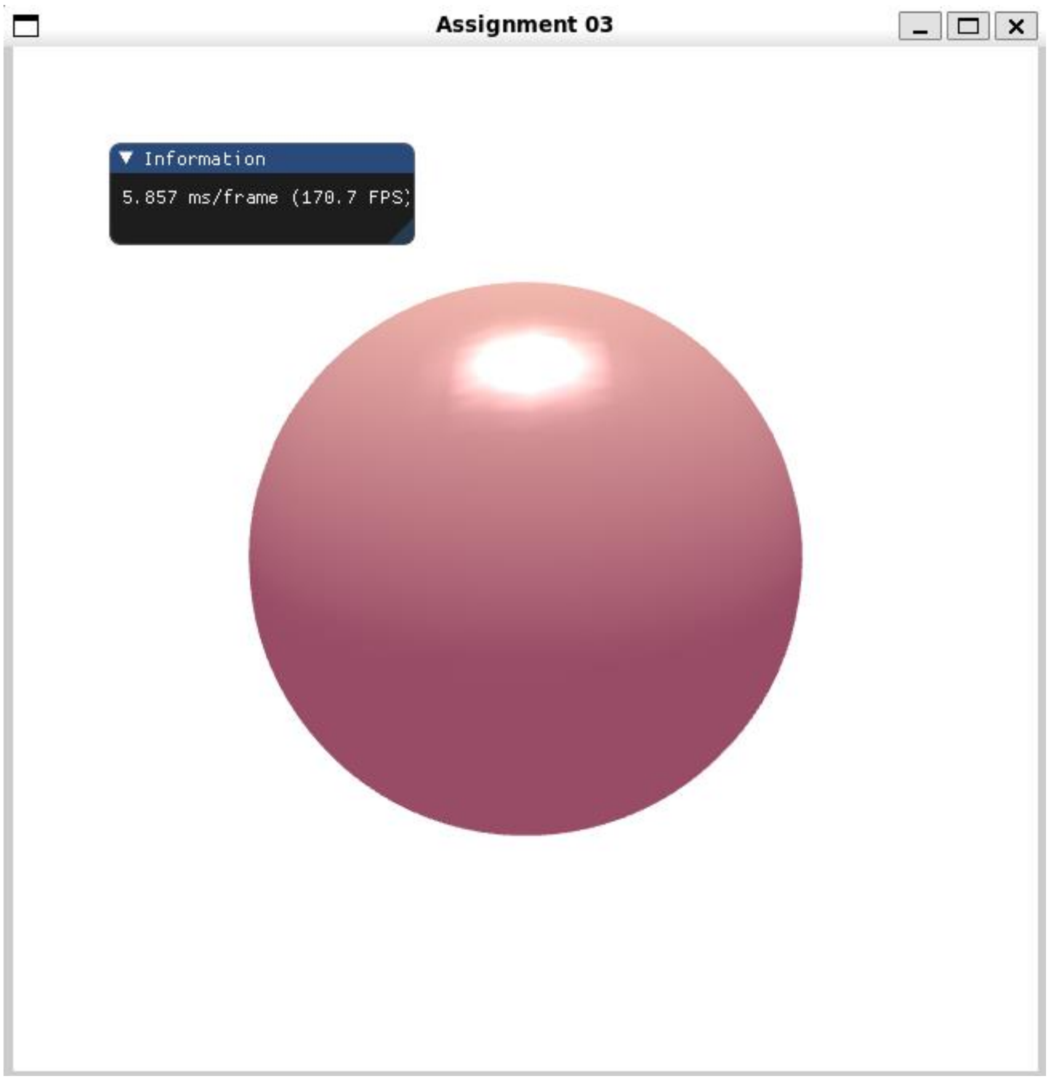
6. Render with Phong Shading:

We made the necessary changes to the fragment shader to accept the lighting values and normals from the vertex shader. The fragment shader performed the final lighting calculations based on the interpolated values received from the vertex shader.

7. Visualization:

The result was a visually appealing rendering of the parametric surface with realistic shading effects, including specular highlights. The added ambient component ensured that the back side of the object was not completely dark.

Screenshot:



```
5.857 ms/frame (170.7 FPS)
```

Answer 2.

In answer 2 of the assignment, we are tasked with implementing a spotlight. In this part, I implemented a spotlight in such a way that only part of the object falls inside the cone of illumination. To improve realism, I added smooth soft edges to the spotlight.

Steps and Implementation:

1. Implementing a Spotlight:

To create a spotlight, we started with the existing codebase and made modifications to the shaders.

2. Vertex Shader Changes:

In the vertex shader, we added the necessary variables to represent the spotlight. These included lightPosition, lightDirection, and lightCutoff. The lightPosition is the position of the spotlight, lightDirection is the direction it's pointing, and lightCutoff is the cutoff angle.

3. Lighting Calculations in the Fragment Shader:

We performed all lighting calculations in the fragment shader. To achieve this, we modified the fragment shader to take into account the spotlight's properties and the object's position.

4. Implementing the Spotlight Cone:

To ensure that only a part of the object is illuminated, we calculated the angle between the spotlight's direction and the vertex's normal vector. We then used the lightCutoff angle to determine if a fragment was inside the cone of illumination.

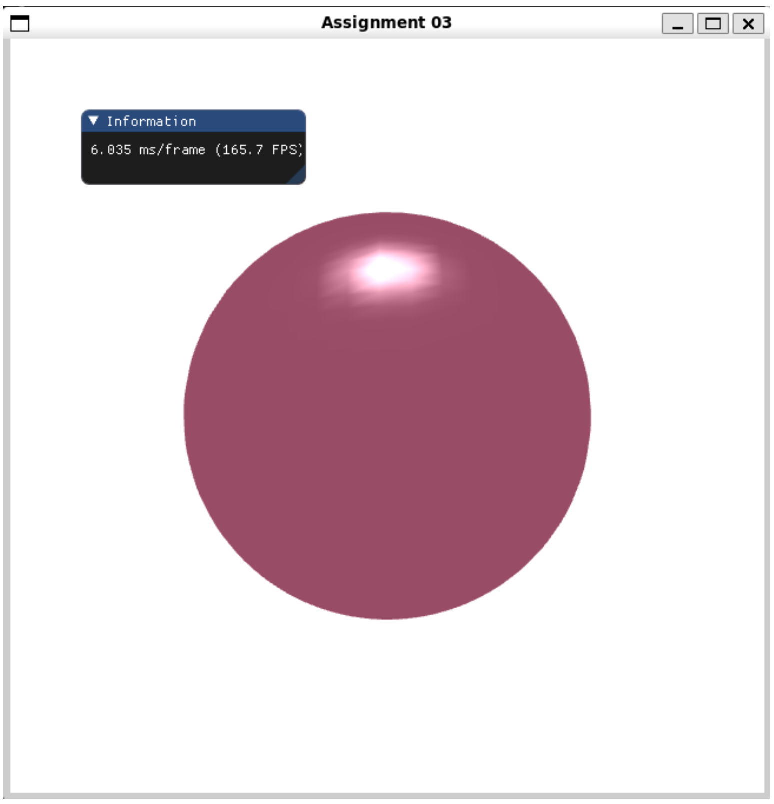
5. Smooth Soft Edges:

We implemented smooth soft edges for the spotlight by creating two cones: an inner cone with a lightCutoff - 5 degrees angle and an outer cone with a lightCutoff angle. If a fragment fell within these two cones, we linearly interpolated the intensity to smoothly transition it from 1 to 0. This resulted in a realistic falloff of light intensity at the spotlight's edges.

6. Adjusting the Light Intensity:

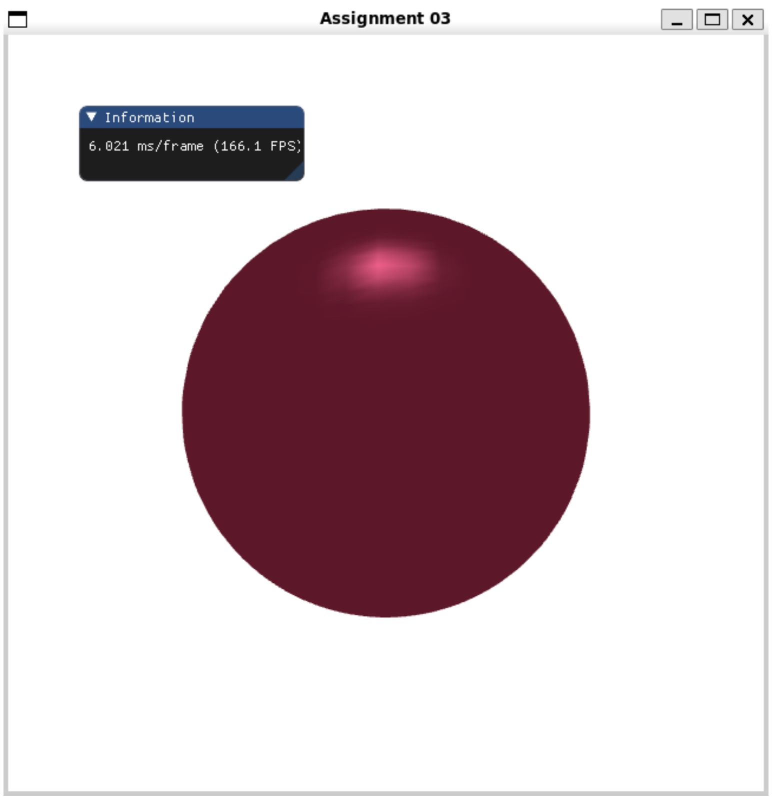
We used this interpolated intensity to modulate the final light intensity calculated for each fragment.

Part A Screenshot:



6.035 ms/frame (165.7 FPS)

Part B Screenshot:



6.021 ms/frame (166.1 FPS)