CS370 - Assignment 3

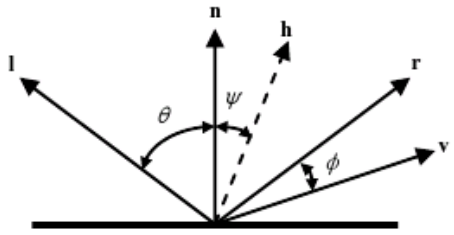
1. Whenever large polygons are rendered with lighting in OpenGL, instead of having relatively uniform shading they are bright in one area and dimmer in others. Explain why this occurs and how this problem can be avoided. Furthermore, why are the light sources applied to **every** object instead of considering whether or not the light is blocked from one object to another, i.e. our lights do not cast **shadows**?

In the basic Gouraud lighting model, the shading occurs at the **vertices**, i.e. where the normals are defined. Thus for large surfaces, the vertex colors determined by the lighting model will simply be interpolated producing a gradient effect (possibly with artifacts due to tesselation). To improve the lighting effect, we can move the lighting computations to the *fragment* shader where the colors will be determined by the interpolated normals on a *per fragment* basis. Since there are many more fragments than vertices, the lighting will be much more accurate but with a *significant* performance impact from the additional computations.

As far as global effects, since the pipeline processes each object individually, all the lights are applied to every object. When each object is rendered, it has no knowledge of other objects in the scene (other than the depth of the closest object to the camera for hidden-surface removal) and thus cannot determine if it is in the shadow of another object. Later we will see how to address this issue to some extent through a multi-pass rendering technique.

2. For specular reflections, the equation in the Phong model involves the angle between the *reflection* vector and the *viewer* vector, i.e. $r \cdot v$. To avoid this computation, if v is in the same plane as l, n, and r we can compute the *halfway vector* (the vector that is halfway between l and v) as $h = (l + v)/(|l + v|)$. Show that the angle ψ between n and h is *half* the angle ϕ between r and v. In otherwords, show $2\psi = \phi$

Using the diagram below



By the definition of the half angle, we see that the angle from l to v is twice the angle from l to h giving

$$l \cdot v = cos(2(\theta + \psi))$$

Furthermore, since the angle of incidence equals the angle of reflection, the angle from l to n is the same as the angle from n to r $(= \theta)$ giving the angle from l to v as twice the angle from l to n plus the angle from r to v

$$l \cdot v = cos(2\theta + \phi)$$

Combining these two equations and equating the angles gives

$$2(\theta + \psi) = 2\theta + 2\psi = 2\theta + \phi \Rightarrow 2\psi = \phi$$

Thus we can approximate the specular attenuation factor using only the light vector l and the camera v which are easily computed by the information passed in by the application avoiding the need to compute the reflection vector.

3. Assume that at a given pixel in the framebuffer that contains color channel values (0.1f, 0.4f, 0.7f, 1.0f), we render *two* translucent objects with the following colors

- (0.7f, 0.3f, 0.4f, 0.6f)
- (0.4f, 0.2f, 0.8f, 0.4f)

What is the final blended color for that pixel using blend factors

GL_SRC_ALPHA/GL_ONE_MINUS_SRC_ALPHA
Blending in the first object with $\alpha_s = 0.6$ gives

$$\begin{aligned} r_1 &= 0.6 * 0.7 + (1 - 0.6) * 0.1 = 0.42 + 0.04 = 0.46 \\ g_1 &= 0.6 * 0.3 + (1 - 0.6) * 0.4 = 0.18 + 0.16 = 0.34 \\ b_1 &= 0.6 * 0.4 + (1 - 0.6) * 0.7 = 0.24 + 0.28 = 0.52 \\ \alpha_1 &= 0.6 * 0.6 + (1 - 0.6) * 1.0 = 0.36 + 0.4 = 0.76 \end{aligned}$$

Thus the framebuffer color after the first object is (0.46f, 0.34f, 0.52f, 0.76f).

Blending in the second object with $\alpha_s = 0.4$ gives

$$\begin{aligned} r_2 &= 0.4 * 0.4 + (1 - 0.4) * 0.46 = 0.16 + 0.276 = 0.436 \\ g_2 &= 0.4 * 0.2 + (1 - 0.4) * 0.34 = 0.08 + 0.204 = 0.284 \\ b_2 &= 0.4 * 0.8 + (1 - 0.4) * 0.52 = 0.32 + 0.312 = 0.632 \\ \alpha_2 &= 0.4 * 0.4 + (1 - 0.4) * 0.76 = 0.16 + 0.456 = 0.616 \end{aligned}$$

Thus the final color of the pixel is **(0.436f, 0.284f, 0.632f, 0.616f)**. **Note:** Had object 2 been rendered first, the final color for the pixel would be (0.508f, 0.308f, 0.536f, 0.648f) which is different from the first ordering. However, all the color channels always remain in the range $[0, 1]$, regardless of the number of objects rendered.

GL_SRC_ALPHA/GL_ONE
Blending in the first object with $\alpha_s = 0.6$ gives

$$
\begin{aligned}
r_1 &= 0.6 * 0.7 + 1 * 0.1 = 0.42 + 0.1 = 0.52 \\
g_1 &= 0.6 * 0.3 + 1 * 0.4 = 0.18 + 0.4 = 0.58 \\
b_1 &= 0.6 * 0.4 + 1 * 0.7 = 0.24 + 0.7 = 0.94 \\
\alpha_1 &= 0.6 * 0.6 + 1 * 1.0 = 0.36 + 1 = 1.36
\end{aligned}
$$

Thus the framebuffer color after the first object is (0.52f, 0.58f, 0.94f, 1.36f).

Blending in the second object with $\alpha_s = 0.4$ gives

$$
\begin{aligned}
r_2 &= 0.4 * 0.4 + 1 * 0.52 = 0.16 + 0.52 = 0.68 \\
g_2 &= 0.4 * 0.2 + 1 * 0.58 = 0.08 + 0.58 = 0.66 \\
b_2 &= 0.4 * 0.8 + 1 * 0.94 = 0.32 + 0.94 = 1.26 \\
\alpha_2 &= 0.4 * 0.4 + 1 * 1.36 = 0.16 + 1.36 = 1.52
\end{aligned}
$$

Thus the final color of the pixel is **(0.68f, 0.66f, 1.26f, 1.52f) = (0.68f, 0.66f, 1.0f, 1.0f)** when rendered as color channels over 1.0 will be clamped, which is brighter and contains more green and blue than the previous pair of blend factors (although not as much blue as there should be). Note: These colors would be the same if object 2 was rendered first, but we see that the blue and alpha channels are outside the $[0, 1]$ range, and other channels could go out of range if additional objects were rendered.