8 Basic Lighting and Reflection

Up to this point, we have considered only the geometry of how objects are transformed and projected to images. We now discuss the *shading* of objects: how the appearance of objects depends, among other things, on the lighting that illuminates the scene, and on the interaction of light with the objects in the scene. Some of the basic qualitative properties of lighting and object reflectance that we need to be able to model include:

Light source - There are different types of sources of light, such as point sources (e.g., a small light at a distance), extended sources (e.g., the sky on a cloudy day), and secondary reflections (e.g., light that bounces from one surface to another).

Reflectance - Different objects reflect light in different ways. For example, diffuse surfaces appear the same when viewed from different directions, whereas a mirror looks very different from different points of view.

In this chapter, we will develop simplified model of lighting that is easy to implement and fast to compute, and used in many real-time systems such as OpenGL. This model will be an approximation and does not fully capture all of the effects we observe in the real world. In later chapters, we will discuss more sophisticated and realistic models.

8.1 Simple Reflection Models

8.1.1 Diffuse Reflection

We begin with the diffuse reflectance model. A diffuse surface is one that appears similarly bright from all viewing directions. That is, the emitted light appears independent of the viewing location. Let $\bar{p}$ be a point on a diffuse surface with normal $\vec{n}$, light by a point light source in direction $\vec{s}$ from the surface. The reflected intensity of light is given by:

$$L_d(\bar{p}) \quad = \quad r_d \, I \, \max(0, \vec{s} \cdot \vec{n}) \tag{1}$$

where I is the intensity of the light source, r_d is the diffuse reflectance (or albedo) of the surface, and $\vec{s}$ is the direction of the light source. This equation requires the vectors to be normalized, i.e., $||\vec{s}|| = 1$, $||\vec{n} = 1||$.

The $\vec{s} \cdot \vec{n}$ term is called the *foreshortening term*. When a light source projects light obliquely at a surface, that light is spread over a large area, and less of the light hits any specific point. For example, imagine pointing a flashlight directly at a wall versus in a direction nearly parallel: in the latter case, the light from the flashlight will spread over a greater area, and individual points on the wall will not be as bright.

For color rendering, we would specify the reflectance in color (as $(r_{d,R}, r_{d,G}, r_{d,B})$), and specify the light source in color as well (I_R, I_G, I_B). The reflected color of the surface is then:

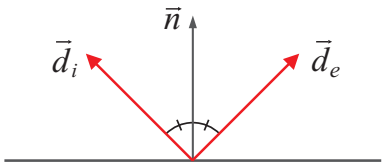
$$L_{d,R}(\bar{p}) = r_{d,R}\, I_R\, \max(0, \vec{s} \cdot \vec{n}) \quad (2)$$

$$L_{d,G}(\bar{p}) = r_{d,G}\, I_G\, \max(0, \vec{s} \cdot \vec{n}) \quad (3)$$

$$L_{d,B}(\bar{p}) = r_{d,B}\, I_B\, \max(0, \vec{s} \cdot \vec{n}) \quad (4)$$

8.1.2 Perfect Specular Reflection

For pure specular (mirror) surfaces, the incident light from each incident direction $\vec{d_i}$ is reflected toward a unique emittant direction $\vec{d_e}$. The emittant direction lies in the same plane as the incident direction $\vec{d_i}$ and the surface normal $\vec{n}$, and the angle between $\vec{n}$ and $\vec{d_e}$ is equal to that between $\vec{n}$ and $\vec{d_i}$. One can show that the emittant direction is given by $\vec{d_e} = 2(\vec{n} \cdot \vec{d_i})\vec{n} - \vec{d_i}$. (The derivation was



covered in class). In perfect specular reflection, the light emitted in direction $\vec{d_e}$ can be computed by reflecting $\vec{d_e}$ across the normal (as $2(\vec{n} \cdot \vec{d_e})\vec{n} - \vec{d_e}$), and determining the incoming light in this direction. (Again, all vectors are required to be normalized in these equations).

8.1.3 General Specular Reflection

Many materials exhibit a significant specular component in their reflectance. But few are perfect mirrors. First, most specular surfaces do not reflect all light, and that is easily handled by introducing a scalar constant to attenuate intensity. Second, most specular surfaces exhibit some form of *off-axis specular reflection*. That is, many polished and shiny surfaces (like plastics and metals) emit light in the perfect mirror direction and in some nearby directions as well. These off-axis specularities look a little blurred. Good examples are *highlights* on plastics and metals.

More precisely, the light from a distant point source in the direction of $\vec{s}$ is reflected into a range of directions about the perfect mirror directions $\vec{m} = 2(\vec{n} \cdot \vec{s})\vec{n} - \vec{s}$. One common model for this is the following:

$$L_s(\vec{d_e}) = r_s I \max(0, \vec{m} \cdot \vec{d_e})^\alpha, \quad (5)$$

where r_s is called the specular reflection coefficient (often equal to $1 - r_d$), I is the incident power from the point source, and $\alpha \geq 0$ is a constant that determines the width of the specular highlights. As α increases, the effective width of the specular reflection decreases. In the limit as α increases, this becomes a mirror.

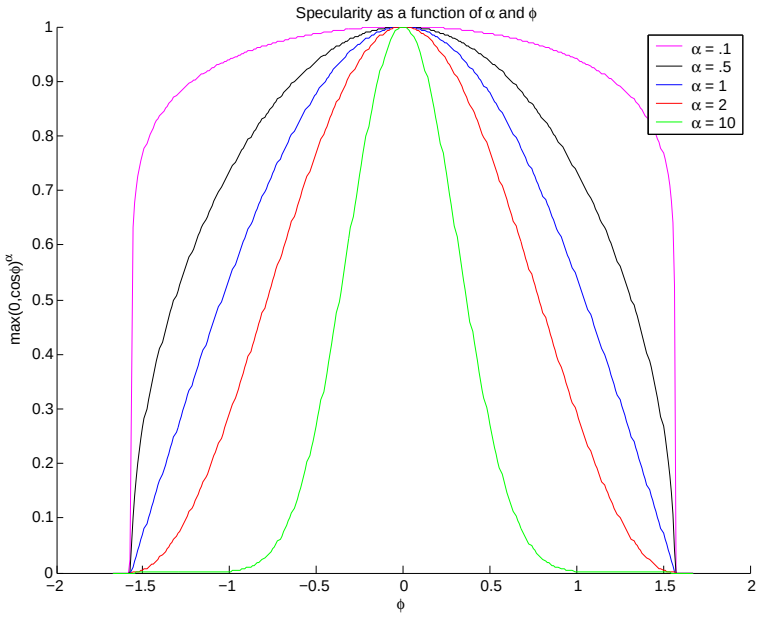


Figure 1: Plot of specular intensity as a function of viewing angle ϕ.

The intensity of the specular region is proportional to $\max(0, \cos\phi)^\alpha$, where ϕ is the angle between $\vec{m}$ and $\vec{d}_e$. One way to understand the nature of specular reflection is to plot this function, see Figure 1.

8.1.4 Ambient Illumination

The diffuse and specular shading models are easy to compute, but often appear artificial. The biggest issue is the point light source assumption, the most obvious consequence of which is that any surface normal pointing away from the light source (i.e., for which $\vec{s} \cdot \vec{n} < 0$) will have a radiance of zero. A better approximation to the light source is a uniform *ambient* term plus a point light source. This is a still a remarkably crude model, but it's much better than the point source by itself. Ambient illumintation is modeled simply by:

$$L_a(\bar{p}) = r_a \, I_a \tag{6}$$

where r_a is often called the ambient reflection coefficient, and I_a denotes the integral of the uniform illuminant.

8.1.5 Phong Reflectance Model

The **Phong reflectance model** is perhaps the simplest widely used shading model in computer graphics. It comprises a diffuse term (Eqn (1)), an ambient term (Eqn (6)), and a specular term

(Eqn (5)):

$$L(\bar{p}, \vec{d_e}) \;=\; r_d\, I_d\, \max(0, \vec{s} \cdot \vec{n}) \;+\; r_a\, I_a \;+\; r_s I_s \max(0, \vec{m} \cdot \vec{d_e})^{\alpha}, \tag{7}$$

where

- I_a, I_d, and I_r are parameters that correspond to the power of the light sources for the ambient, diffuse, and specular terms;
- r_a, r_d and r_s are scalar constants, called reflection coefficients, that determine the relative magnitudes of the three reflection terms;
- α determines the spread of the specurlar highlights;
- $\vec{n}$ is the surface normal at $\bar{p}$;
- $\vec{s}$ is the direction of the distant point source;
- $\vec{m}$ is the perfect mirror direction, given $\vec{n}$ and $\vec{s}$; and
- and $\vec{d_e}$ is the emittant direction of interest (usually the direction of the camera).

In effect, this is a model in which the diffuse and specular components of reflection are due to incident light from a point source. Extended light sources and the bouncing of light from one surface to another are not modeled except through the ambient term. Also, arguably this model has more parameters than the physics might suggest; for example, the model does not constrain the parameters to conserve energy. Nevertheless it is sometimes useful to give computer graphics practitioners more freedom in order to acheive the appearance they're after.

8.2 Lighting in OpenGL

OpenGL provides a slightly modified version of Phong lighting. Lighting and any specific lights to use must be enabled to see its effects:

```
glEnable(GL_LIGHTING);  // enable Phong lighting
glEnable(GL_LIGHT0);    // enable the first light source
glEnable(GL_LIGHT1);    // enable the second light source
...
```

Lights can be directional (infinitely far away) or positional. Positional lights can be either point lights or spotlights. Directional lights have the w component set to 0, and positional lights have w set to 1. Light properties are specified with the `glLight` functions:


```
GLfloat direction[] = {1.0f, 1.0f, 1.0f, 0.0f};
GLfloat position[] = {5.0f, 3.0f, 8.0f, 1.0f};
Glfloat spotDirection[] = {0.0f, 3.0f, 3.0f};
Glfloat diffuseRGBA[] = {1.0f, 1.0f, 1.0f, 1.0f};
Glfloat specularRGBA[] = {1.0f, 1.0f, 1.0f, 1.0f};

// A directional light
glLightfv(GL_LIGHT0, GL_POSITION, direction);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseRGBA);
glLightfv(GL_LIGHT0, GL_SPECULAR, specularRGBA);

// A spotlight
glLightfv(GL_LIGHT1, GL_POSITION, position);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseRGBA);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spotDirection);
glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0f);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 30.0f);
```

OpenGL requires you to specify both diffuse and specular components for the light source. This has no physical interpretation (real lights do not have “diffuse” or “specular” properties), but may be useful for some effects. The `glMaterial` functions are used to specify material properties, for example:

```
GLfloat diffuseRGBA = {1.0f, 0.0f, 0.0f, 1.0f};
GLfloat specularRGBA = {1.0f, 1.0f, 1.0f, 1.0f};
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseRGBA);
glMaterialfv(GL_FRONT, GL_SPECULAR, specularRGBA);
glMaterialf(GL_FRONT, GL_SHININESS, 3.0f);
```

Note that both lights and materials have ambient terms. Additionally, there is a global ambient term:

```
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
glMaterialfv(GL_FRONT, GL_AMBIENT, ambientMaterial);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientGlobal);
```

The material has an emission term as well, that is meant to model objects that can give off their own light. However, no light is actually cast on other objects in the scene.

```
glMaterialfv(GL_FRONT, GL_EMISSION, em);
```

The global ambient term is multiplied by the current material ambient value and added to the material’s emission value. The contribution from each light is then added to this value.

When rendering an object, normals should be provided for each face or for each vertex so that lighting can be computed:


```
glNormal3f(nx, ny, nz);
glVertex3f(x, y, z);
```

