

# Shading (Part 1)

Brian Curless  
CSEP 557  
Winter 2013

1

## Reading

Required:

- ◆ Angel chapter 5.

Optional:

- ◆ OpenGL red book, chapter 5.

2

## Introduction

So far, we've talked exclusively about geometry.

- ◆ What is the shape of an object?
- ◆ How do I place it in a virtual 3D space?
- ◆ How do I know which pixels it covers?
- ◆ How do I know which of the pixels I should actually draw?

Once we've answered all those, we have to ask one more important question:

- ◆ To what value do I set each pixel?

Answering this question is the job of the **shading model**.

Other names:

- ◆ Lighting model
- ◆ Light reflection model
- ◆ Local illumination model
- ◆ Reflectance model
- ◆ BRDF

3

## Our problem

Modeling the flow of light in a scene is very complex: photons pour out of light sources and bounce around and around before reaching a camera.

Here we focus on **local illumination**, i.e., what happens for a single bounce:

**light source** → **surface** → **viewer**

No interreflections, no shadows.

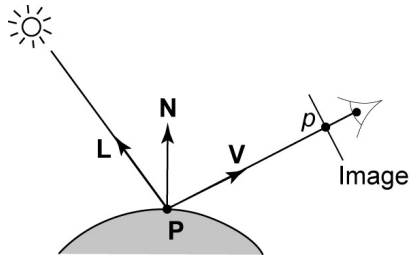
We're going to explore two models: the **Phong** and **Blinn-Phong illumination models**.

They have the following characteristics:

- ◆ physically plausible (albeit not strictly correct)
- ◆ very fast
- ◆ widely used

4

## Setup...



Given:

- ♦ a point **P** on a surface visible through pixel  $p$
- ♦ The normal **N** at **P**
- ♦ The lighting direction, **L**, and (color) intensity,  $I_L$ , at **P**
- ♦ The viewing direction, **V**, at **P**
- ♦ The shading coefficients at **P**

Compute the color,  $I$ , of pixel  $p$ .

Assume that the direction vectors are normalized:

$$\|\mathbf{N}\| = \|\mathbf{L}\| = \|\mathbf{V}\| = 1$$

5

## “Iteration zero”

The simplest thing you can do is...

Assign each polygon a single color:

$$I = k_e$$

where

- ♦  $I$  is the resulting intensity
- ♦  $k_e$  is the **emissivity** or intrinsic shade associated with the object

This has some special-purpose uses, but not really good for drawing a scene.

[Note:  $k_e$  is omitted in Angel.]

6

## “Iteration one”

Let's make the color at least dependent on the overall quantity of light available in the scene:

$$I = k_e + k_a I_{La}$$

- ♦  $k_a$  is the **ambient reflection coefficient**.
  - really the reflectance of ambient light
  - “ambient” light is assumed to be equal in all directions
- ♦  $I_{La}$  is the **ambient light intensity**.

Physically, what is “ambient” light?

[Note: Angel uses  $L_a$  instead of  $I_{La}$ .]

7

## Wavelength dependence

Really,  $k_e$ ,  $k_a$ , and  $I_{La}$  are functions over all wavelengths  $\lambda$ .

Ideally, we would do the calculation on these functions. For the ambient shading equation, we would start with:

$$I(\lambda) = k_a(\lambda) I_{La}(\lambda)$$

then we would find good RGB values to represent the spectrum  $I(\lambda)$ .

Traditionally, though,  $k_a$  and  $I_{La}$  are represented as RGB triples, and the computation is performed on each color channel separately:

$$I^R = k_a^R I_{La}^R$$

$$I^G = k_a^G I_{La}^G$$

$$I^B = k_a^B I_{La}^B$$

8

## Diffuse reflection

Let's examine the ambient shading model:

- ♦ objects have different colors
- ♦ we can control the overall light intensity
  - what happens when we turn off the lights?
  - what happens as the light intensity increases?
  - what happens if we change the color of the lights?

So far, objects are uniformly lit.

- ♦ not the way things really appear
- ♦ in reality, light sources are localized in position or direction

**Diffuse**, or **Lambertian** reflection will allow reflected intensity to vary with the direction of the light.

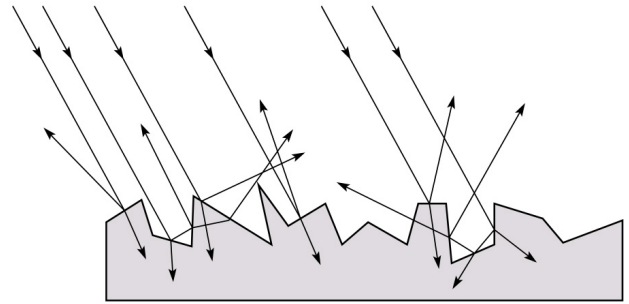
9

## Diffuse reflectors

Diffuse reflection occurs from dull, matte surfaces, like latex paint, or chalk.

These **diffuse** or **Lambertian** reflectors reradiate light equally in all directions.

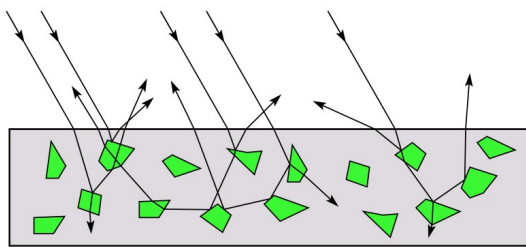
Picture a rough surface with lots of tiny **microfacets**.



10

## Diffuse reflectors

...or picture a surface with little pigment particles embedded beneath the surface (neglect reflection at the surface for the moment):



The microfacets and pigments distribute light rays in all directions.

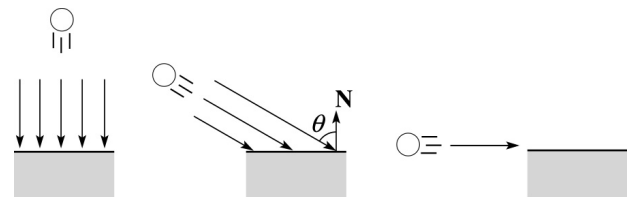
Embedded pigments are responsible for the coloration of diffusely reflected light in plastics and paints.

Note: the figures above are intuitive, but not strictly (physically) correct.

11

## Diffuse reflectors, cont.

The reflected intensity from a diffuse surface does not depend on the direction of the viewer. The incoming light, though, does depend on the direction of the light source:



12

## “Iteration two”

The incoming energy is proportional to \_\_\_\_\_, giving the diffuse reflection equations:

$$I = k_e + k_a I_{La} + k_d I_L B$$

$$= k_e + k_a I_{La} + k_d I_L B( \quad )$$

where:

- ◆  $k_d$  is the **diffuse reflection coefficient**
- ◆  $I_L$  is the (color) intensity of the light source
- ◆  $\mathbf{N}$  is the normal to the surface (unit vector)
- ◆  $\mathbf{L}$  is the direction to the light source (unit vector)
- ◆  $B$  prevents contribution of light from below the surface:

$$B = \begin{cases} 1 & \text{if } \mathbf{N} \cdot \mathbf{L} > 0 \\ 0 & \text{if } \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases}$$

[Note: Angel uses  $L_d$  instead of  $I_L$  and  $f$  instead of  $B$ ]

13

## Specular reflection

**Specular reflection** accounts for the highlight that you see on some objects.

It is particularly important for *smooth, shiny* surfaces, such as:

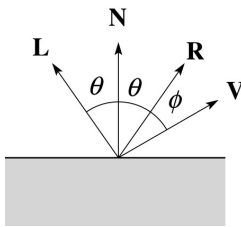
- ◆ metal
- ◆ polished stone
- ◆ plastics
- ◆ apples
- ◆ skin

Properties:

- ◆ Specular reflection depends on the viewing direction  $\mathbf{V}$ .
- ◆ For non-metals, the color is determined solely by the color of the light.
- ◆ For metals, the color may be altered (e.g., brass)

14

## Specular reflection “derivation”



For a perfect mirror reflector, light is reflected about  $\mathbf{N}$ , so

$$I = \begin{cases} I_L & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

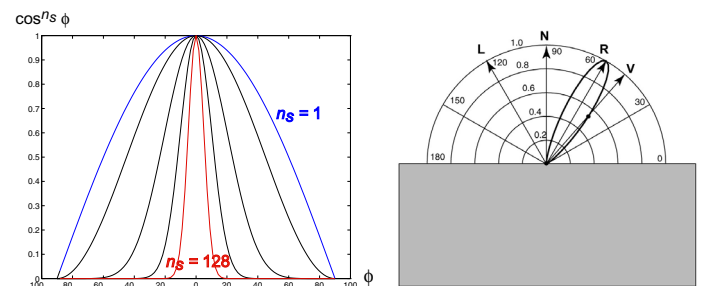
For a near-perfect reflector, you might expect the highlight to fall off quickly with increasing angle  $\phi$ .

Also known as:

- ◆ “**rough specular**” reflection
- ◆ “**directional diffuse**” reflection
- ◆ “**glossy**” reflection

15

## Phong specular reflection



One way to get this effect is to take  $(\mathbf{R} \cdot \mathbf{V})$ , raised to a power  $n_s$ .

As  $n_s$  gets larger,

- ◆ the dropoff becomes {more,less} gradual
- ◆ gives a {larger,smaller} highlight
- ◆ simulates a {more,less} mirror-like surface

Phong specular reflection is proportional to:

$$I_{\text{specular}} = B(\mathbf{R} \cdot \mathbf{V})_+^{n_s}$$

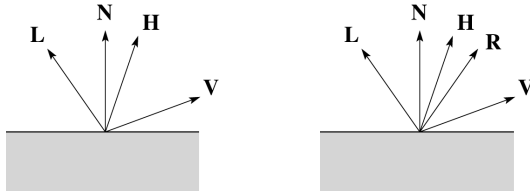
where  $(x)_+ \equiv \max(0, x)$ .

16

## Blinn-Phong specular reflection

A common alternative for specular reflection is the **Blinn-Phong model** (sometimes called the **modified Phong model**.)

We compute the vector halfway between  $\mathbf{L}$  and  $\mathbf{V}$  as:



Analogous to Phong specular reflection, we can compute the specular contribution in terms of  $(\mathbf{N} \cdot \mathbf{H})$ , raised to a power  $n_s$ :

$$I_{\text{specular}} = B(\mathbf{N} \cdot \mathbf{H})_+^{n_s}$$

where, again,  $(x)_+ \equiv \max(0, x)$ .

17

## “Iteration three”

The next update to the Blinn-Phong shading model is then:

$$\begin{aligned} I &= k_e + k_a I_{L_a} + k_d I_L B(\mathbf{N} \cdot \mathbf{L}) + k_s I_L B(\mathbf{N} \cdot \mathbf{H})_+^{n_s} \\ &= k_e + k_a I_{L_a} + I_L B \left[ k_d (\mathbf{N} \cdot \mathbf{L}) + k_s (\mathbf{N} \cdot \mathbf{H})_+^{n_s} \right] \end{aligned}$$

where:

- $k_s$  is the **specular reflection coefficient**
- $n_s$  is the **specular exponent** or **shininess**
- $\mathbf{H}$  is the unit halfway vector between  $\mathbf{L}$  and  $\mathbf{V}$ , where  $\mathbf{V}$  is the viewing direction.

[Note: Angel uses  $\alpha$  instead of  $n_s$ , and maintains a separate  $L_d$  and  $L_s$ , instead of a single  $I_L$ . This choice reflects the flexibility available in OpenGL.]

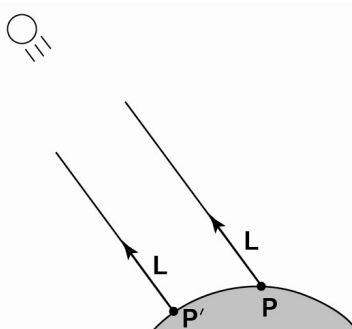
18

## Directional lights

OpenGL supports three different kinds of lights: ambient, directional, and point. Spot lights are also supported as a special form of point light.

We’ve seen ambient light sources, which are not really geometric.

**Directional light** sources have a single direction and intensity associated with them.

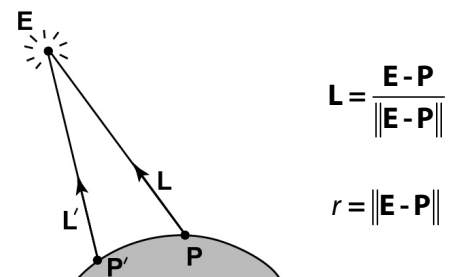


Using affine notation, what is the homogeneous coordinate for a directional light?

19

## Point lights

The direction of a **point light** sources is determined by the vector from the light position to the surface point.



$$\mathbf{L} = \frac{\mathbf{E} - \mathbf{P}}{\|\mathbf{E} - \mathbf{P}\|}$$

$$r = \|\mathbf{E} - \mathbf{P}\|$$

Physics tells us the intensity must drop off inversely with the square of the distance:

$$f_{\text{atten}} = \frac{1}{r^2}$$

Sometimes, this distance-squared dropoff is considered too “harsh.” A common alternative is:

$$f_{\text{atten}} = \frac{1}{a + br + cr^2}$$

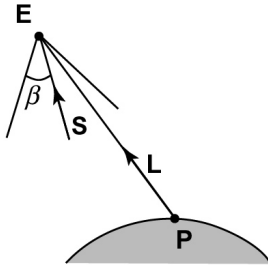
with user-supplied constants for  $a$ ,  $b$ , and  $c$ .

Using affine notation, what is the homogeneous coordinate for a point light?

20

## Spotlights

OpenGL also allows one to apply a *directional attenuation* of a point light source, giving a **spotlight** effect.



The spotlight intensity factor is computed in OpenGL as:

$$f_{\text{spot}} = (\mathbf{L} \cdot \mathbf{S})_{\beta}^e$$

where

- $\mathbf{L}$  is the direction to the point light.
- $\mathbf{S}$  is the center direction of the spotlight.
- $\beta$  is the cutoff angle for the spotlight
- $e$  is the angular falloff coefficient
- $(x)_{\beta}^e = [\max\{\cos(x) - \beta, 0\}]^e$

21

## “Iteration four”

Since light is additive, we can handle multiple lights by taking the sum over every light.

Our equation is now (for point lights):

$$I = k_e + k_a I_{La} + \sum_j \frac{1}{a_j + b_j r_j + c_j r_j^2} I_{L_j} B_j [k_d (\mathbf{N} \cdot \mathbf{L}_j) + k_s (\mathbf{N} \cdot \mathbf{H}_j)_+^{n_s}]$$

This is the Blinn-Phong illumination model.

Which quantities are spatial vectors?

Which are RGB triples?

Which are scalars?

22

## Choosing the parameters

Experiment with different parameter settings. To get you started, here are a few suggestions:

- Try  $n_s$  in the range [0,100]
- Try  $k_a + k_d + k_s < 1$
- Use a small  $k_a$  (~0.1)

|         | $n_s$  | $k_d$                    | $k_s$                 |
|---------|--------|--------------------------|-----------------------|
| Metal   | large  | Small, color of metal    | Large, color of metal |
| Plastic | medium | Medium, color of plastic | Medium, white         |
| Planet  | 0      | varying                  | 0                     |

23

## Materials in OpenGL

The OpenGL code to specify the surface shading properties is fairly straightforward. For example:

```
GLfloat ke[] = { 0.1, 0.15, 0.05, 1.0 };
GLfloat ka[] = { 0.1, 0.15, 0.1, 1.0 };
GLfloat kd[] = { 0.3, 0.3, 0.2, 1.0 };
GLfloat ks[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat ns[] = { 50.0 };
glMaterialfv(GL_FRONT, GL_EMISSION, ke);
glMaterialfv(GL_FRONT, GL_AMBIENT, ka);
glMaterialfv(GL_FRONT, GL_DIFFUSE, kd);
glMaterialfv(GL_FRONT, GL_SPECULAR, ks);
glMaterialfv(GL_FRONT, GL_SHININESS, ns);
```

Notes:

- The `GL_FRONT` parameter tells OpenGL that we are specifying the materials for the front of the surface.
- Only the alpha value of the diffuse color is used for blending. It's usually set to 1.

24

## Shading in OpenGL

The OpenGL lighting model allows you to associate different lighting colors according to material properties they will influence.

Thus, our original shading equation:

$$I = k_e + k_a I_{La} + \sum_j \frac{1}{a_j + b_j r_j + c_j r_j^2} I_{L,j} B_j \left[ k_d (\mathbf{N} \cdot \mathbf{L}_j)_+ + k_s (\mathbf{N} \cdot \mathbf{H}_j)_+^{n_s} \right]$$

becomes:

$$I = k_e + k_a I_{La} + \sum_j \frac{1}{a_j + b_j r_j + c_j r_j^2} \left[ k_a I_{La,j} + B_j \left\{ k_d I_{Ld,j} (\mathbf{N} \cdot \mathbf{L}_j)_+ + k_s I_{Ls,j} (\mathbf{N} \cdot \mathbf{H}_j)_+^{n_s} \right\} \right]$$

where you can have a global ambient light with intensity  $I_{La}$  in addition to having an ambient light intensity  $I_{La,j}$  associated with each individual light, as well as separate diffuse and specular intensities,  $I_{Ld,j}$  and  $I_{Ls,j}$  respectively.

25

## Shading in OpenGL, cont'd

In OpenGL this equation, for one light source (the 0<sup>th</sup>) is specified something like:

```
GLfloat La[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat La0[] = { 0.1, 0.1, 0.1, 1.0 };
GLfloat Ld0[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat Ls0[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat pos0[] = { 1.0, 1.0, 1.0, 0.0 };
GLfloat a0[] = { 1.0 };
GLfloat b0[] = { 0.5 };
GLfloat c0[] = { 0.25 };
GLfloat S0[] = { -1.0, -1.0, 0.0 };
GLfloat beta0[] = { 45 };
GLfloat e0[] = { 2 };
```

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, La);
glLightfv(GL_LIGHT0, GL_AMBIENT, La0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, Ld0);
glLightfv(GL_LIGHT0, GL_SPECULAR, Ls0);
glLightfv(GL_LIGHT0, GL_POSITION, pos0);
glLightfv(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a0);
glLightfv(GL_LIGHT0, GL_LINEAR_ATTENUATION, b0);
glLightfv(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c0);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, S0);
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, beta0);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, e0);
```

26

## Shading in OpenGL, cont'd

Notes:

You can have as many as GL\_MAX\_LIGHTS lights in a scene. This number is system-dependent.

For directional lights, you specify a light direction, not position, and the attenuation and spotlight terms are ignored.

The directions of directional lights and spotlights are specified in the coordinate systems of the lights, not the surface points as we've been doing in lecture.

27

## BRDF

The diffuse+specular parts of the Blinn-Phong illumination model are a mapping from light to viewing directions:

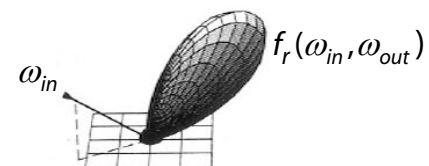
$$I = I_L B \left[ k_d (\mathbf{N} \cdot \mathbf{L}) + k_s \left( \mathbf{N} \cdot \frac{\mathbf{L} + \mathbf{V}}{\|\mathbf{L} + \mathbf{V}\|} \right)_+^{n_s} \right] \\ = I_L f_r(\mathbf{L}, \mathbf{V})$$

The mapping function  $f_r$  is often written in terms of incoming (light) directions  $\omega_{in}$  and outgoing (viewing) directions  $\omega_{out}$ :

$$f_r(\omega_{in}, \omega_{out}) \quad \text{or} \quad f_r(\omega_{in} \rightarrow \omega_{out})$$

This function is called the **Bi-directional Reflectance Distribution Function (BRDF)**.

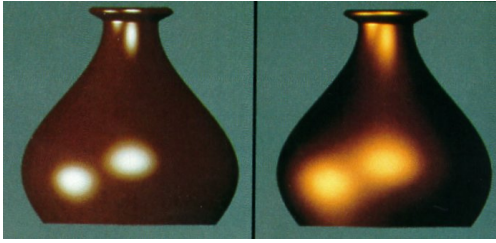
Here's a plot with  $\omega_{in}$  held constant:



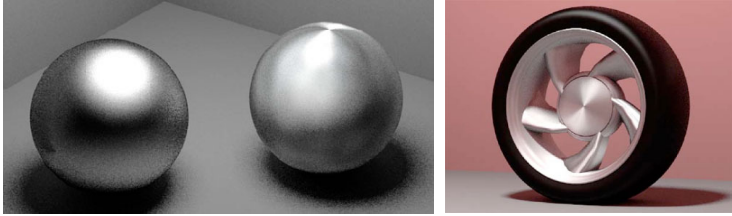
BRDF's can be quite sophisticated...

28

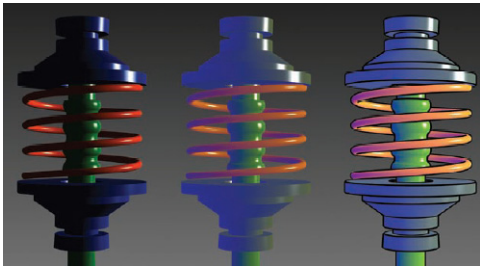
## More sophisticated BRDF's



[Cook and Torrance, 1982]



Anisotropic BRDFs [Westin, Arvo, Torrance 1992]



Artistics BRDFs [Gooch]