5. Shading

Reading

Required:

- Watt, sections 6.2-6.3

Optional:

- Watt, chapter 7.

Introduction

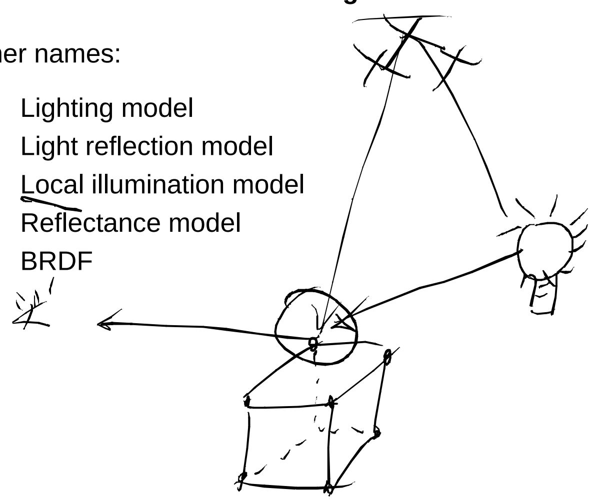
Affine transformations help us to place objects into a scene.

Before creating images of these objects, we'll look at models for how light interacts with their surfaces.

Such a model is called a **shading model**.

Other names:

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



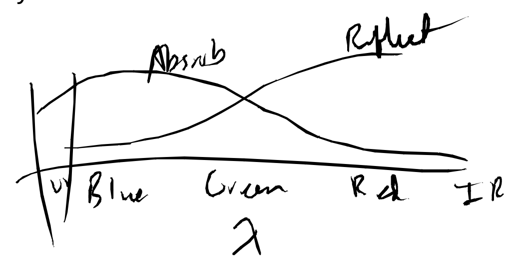
An abundance of photons

Properly determining the right color is *really hard.*

Look around the room. Each light source has different characteristics. Trillions of photons are pouring out every second.

These photons can:

- interact with the atmosphere, or with things in the atmosphere
- strike a surface and
  - be absorbed
  - be reflected (scattered)
  - cause fluorescence or phosphorescence.
- interact in a wavelength-dependent manner
- generally bounce around and around



Our problem

We're going to build up to an *approximation* of reality called the **Phong illumination model**.

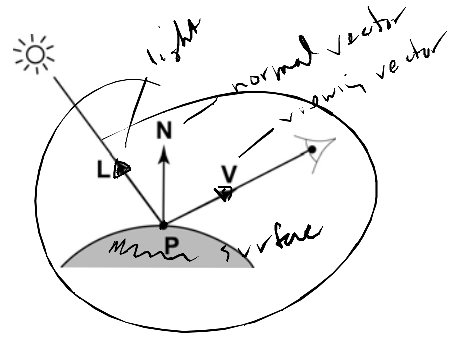
It has the following characteristics:

- *not* physically based
- gives a first-order *approximation* to physical light reflection
- very fast
- widely used

In addition, we will assume **local illumination**, i.e., light goes: light source -> surface -> viewer.

No interreflections, no shadows.

Setup…



Given:

- a point **P** on a surface visible through pixel *p*
- The normal **N** at **P**
- The lighting direction, **L**, and 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:

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

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 Watt.]

Iteration one

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

$$I = k_e + k_a I_a$$

- k_a is the **ambient reflection coefficient.**
  - really the reflectance of ambient light
  - "ambient" light is assumed to be equal in all directions
- I_a is the **ambient intensity**.

Physically, what is "ambient" light?

Wavelength dependence

Really, k_e, k_a, and I_a are functions over all wavelengths λ.

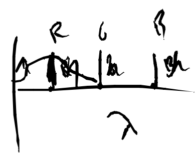
Ideally, we would do the calculation on these functions. For the ambient shading equation, we would start with:

$$I(\lambda) = k_a(\lambda) I_a(\lambda)$$

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

Traditionally, though, k_a and I_a are represented as RGB triples, and the computation is performed on each color channel separately:

$$I_R = k_{a,R}\ I_{a,R}$$
$$I_G = k_{a,G}\ I_{a,G}$$
$$I_B = k_{a,B}\ I_{a,B}$$

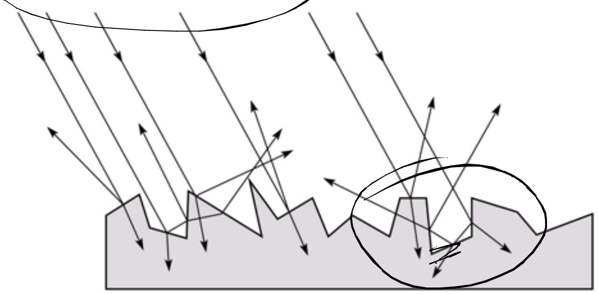


Diffuse reflectors

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

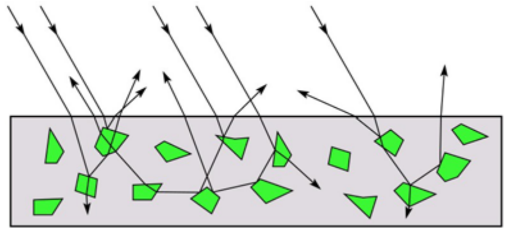
These **diffuse** or **Lambertian** reflectors reradiate light equally in all directions.

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



Diffuse reflectors

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



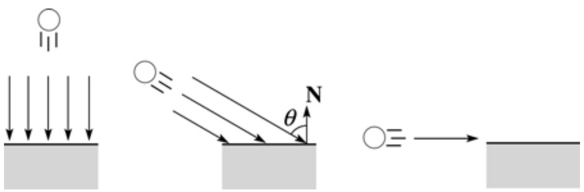
The microfacets and pigments distribute light rays in all directions.

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.

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:



Iteration two

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

$$I = k_e + k_a I_a + k_d I_\ell \,\_\_\_\_$$

$$= k_e + k_a I_a + k_d I_\ell (\qquad)$$

where:

- k_d is the **diffuse reflection coefficient**
- I_ℓ is the intensity of the light source
- ***N*** is the normal to the surface (unit vector)
- ***L*** is the direction to the light source (unit vector)
- $(x)_+$ means max $\{0,x\}$

[Note: Watt uses I_i instead of I_ℓ.]

Specular reflection

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

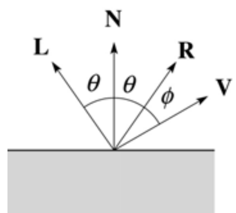
It is particularly important for *smooth, shiny* surfaces, such as:

- metal
- polished stone
- plastics
- apples
- skin

Properties:

- Specular reflection depends on the viewing direction ***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)

Specular reflection “derivation”



For a perfect mirror reflector, light is reflected about ***N***, so

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

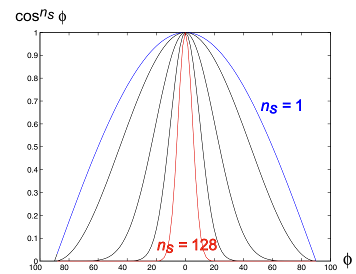
For a near-perfect reflector, you might expect the highlight to fall off quickly with increasing angle ϕ.

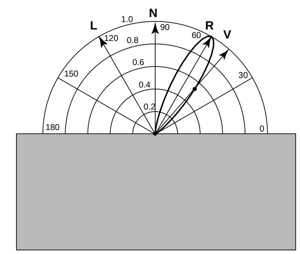
Also known as:

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

15

Derivation, cont.





One way to get this effect is to take (***R·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

16

Iteration three

The next update to the Phong shading model is then:

$$I = k_e + k_a I_a + k_d I_\ell (\mathbf{N} \cdot \mathbf{L})_+ + k_s I_\ell (\mathbf{V} \cdot \mathbf{R})_+^{n_s}$$

where:

- k_s is the **specular reflection coefficient**
- n_s is the **specular exponent** or **shininess**
- ***R*** is the reflection of the light about the normal (unit vector)
- ***V*** is viewing direction (unit vector)

[Note: Watt uses *n* instead of n_s.]

Intensity drop-off with distance

OpenGL supports different kinds of lights: point, directional, and spot.

For point light sources, the laws of physics state that the intensity of a point light source must drop off inversely with the square of the distance.

We can incorporate this effect by multiplying I_l by $1/d^2$.

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

$$f_{atten}(d) = \frac{1}{a + bd + cd^2}$$

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

[Note: not discussed in Watt.]

Iteration four

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

Our equation is now:

$$I = k_e + k_a I_a + \sum_j f_{atten}(d_j) I_{\ell j} \left[k_d (\mathbf{N} \cdot \mathbf{L}_j)_+ + k_s (\mathbf{V} \cdot \mathbf{R}_j)_+^{n_s} \right]$$

This is the Phong illumination model.

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

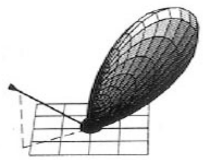
BRDF

The Phong illumination model is really a function that maps light from incoming (light) directions to outgoing (viewing) directions:

$$f_r(\omega_{in}, \omega_{out})$$

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

Here's a plot with ω_{in} held constant:



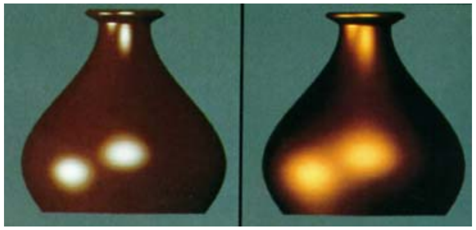
Physcally valid BRDF's obey Helmholtz reciprocity:

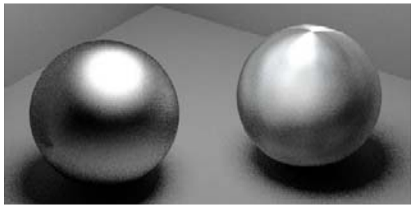
$$f_r(\omega_{in}, \omega_{out}) = f_r(\omega_{out}, \omega_{in})$$

and should conserve energy (no light amplification).

More sophisticated BRDF's

Cook and Torrance, 1982





Westin, Arvo, Torrance 1992