Shading

Reading

Foley, Section16.1

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**.

(Of course, people also call it a lighting model, a light reflection model, a local illumination model, a reflectance model, etc., etc.)

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
  - cause fluorescence or phosphorescence.
- interact in a wavelength-dependent manner
- generally bounce around and around, ad nauseum

Our problem

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

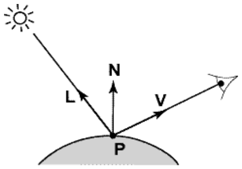
It has the following characteristics:

- *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_λ, 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: $\|\mathbf{N}\| = \|\mathbf{L}\| = \|\mathbf{V}\| = 1$

Emissivity

Assign each polygon a single color:

$$I = k_e$$

where

- I is the resulting intensity
- k_e is the intrinsic shade associated with the object

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

Often used to add color to a surface by circumventing the shading computation.

Ambient reflection

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

$$I = k_a I_a$$

Where

- 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_a and I_a are functions over all wavelengths λ.

Ideally, we would do the calculation on these functions:

$$I(l) = k_a(l) I_a(l)$$

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

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

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 directional

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

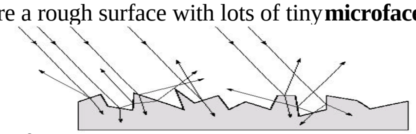
Diffuse reflectors

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

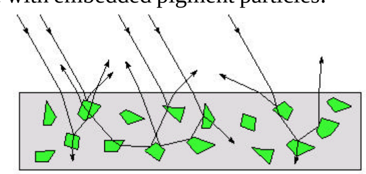
These **diffuse** or **Lambertian** reflectors reradiate light equally in all directions.

Diffuse reflectors, cont'd

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



Or a surface with embedded pigment particles:

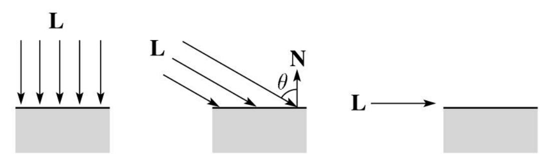


- Light may actually penetrate the surface, bounce around, and then reflect back out.
- Accounts for colorization of diffusely reflected light by plastics.

Q: Why is the North Pole cold? Why is winter cold?

Diffuse reflectors

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.



Diffuse reflectors coefficitents

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

$$I = k_e + k_a I_a + k_d I_l \cos q$$
$$= k_e + k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L})_+$$

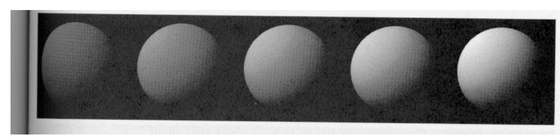
where:

- 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 $\{0, x\}$

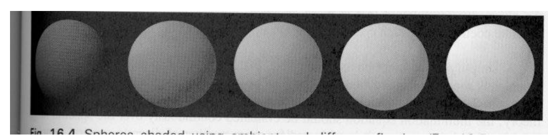
OpenGL supports different kinds of lights: point, directional, and spot. How do these work?

Ambient and Diffuse Examples

Increasing the diffuse coefficient:



Increasing the ambient term while keeping the diffuse term constant:



Intensity drop-off with distance

The laws of physics state that the intensity of a point light source must drop off with its distance squared.

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

Sometimes, this distance-squared dropoff is considered too "harsh." Angel suggests using

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

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

$$f(d) = \min\left(1, \frac{1}{a + bd + cd^2}\right)$$

Specular reflection

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

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

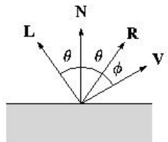
- metal
- polished stone
- plastics
- Safeway apples

Specular reflection depends on the viewing direction **V**. The color is often determined solely by the color of the light.

- corresponds to absence of internal reflections

Specular reflection derivation

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



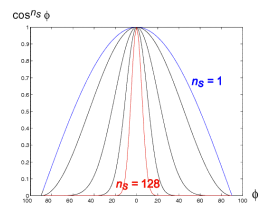
$$I = \begin{cases} I_l & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

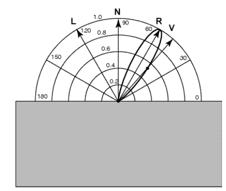
For a near-perfect reflector, you might expect the highlight to fall off quickly with increasing angle ϕ.

Also known as:

- **"rough specular" reflection**
- **"directional diffuse" reflection**
- **"glossy" reflection**

Derivation, cont.





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} glossy surface

Putting it all together

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

Our equation is now:

$$I = k_e + k_a I_a + \sum_i f(d_i)\, I_{li}\left[k_d (\mathbf{N}\cdot\mathbf{L}_i)_+ + k_s (\mathbf{V}\cdot\mathbf{R})_+^{n_s}\right]$$

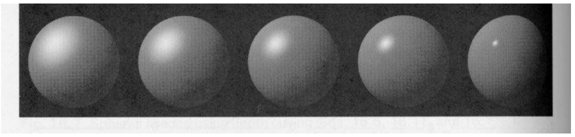
This is the Phong illumination model.

Which quantities are spatial vectors?

Which are RGB triples?

Which are scalars?

Specular Example



Effect on varying n_s

Choosing the parameters

How would I model…

- polished copper?
- blue plastic?
- lunar dust?

Choosing the Parameters

n_s in the range [0,100]

Try $k_a + k_d + k_s \le 1$

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

Choosing the parameters

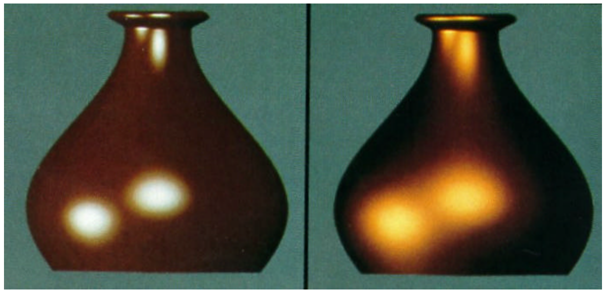


Image of Jupiter

Gouraud vs. Phong Interpolation

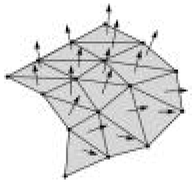
Smooth surfaces are often approximated by polygonal facets because:

- Graphic hardware generally wants polygons
- We know how to intersect rays with polygons

How do we compute the shading for such a surface?

Faceted shading

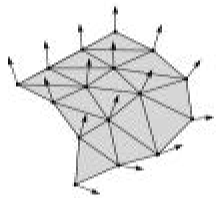
Assume each face has constant normal

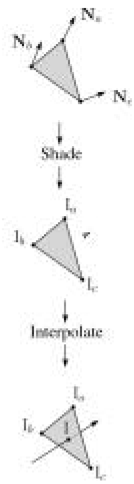


Result: facted, non non-smooth, appearance

Gouraud interpolation

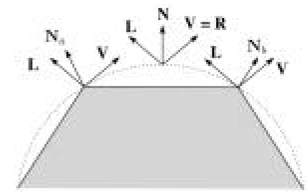
1. Compute normals at vertices
2. Shade only vertices
3. Interpolate the resulting vertex colors





Gouraud interpolation problems

If the polygonal approximation is too coarse we can miss specular highlights



We will encounter Mach banding

Phong interpolation

1. Compute normals at the vertices
2. Interpolate normals and normalize
3. Shade using the interpolated normals