7. Shading

Reading

Required:

- Foley, et al, section 16.1.

Introduction

In these notes, we'll build up to the "Phong illumination model":

- Not physically-based
- Gives an approximation to physical light reflection
- Very fast
- Widely used

This model was first developed by Phong Bui-Thong in 1975.

The problem

Given:

- A point P on a surface
- Visible through pixel p
- A unit vector $\mathbf{V}$ from P to the viewer
- A unit vector $\mathbf{L}$ from P to a point light source

Find: The intensity and color of light radiating from P to the viewer.

Solved by developing a model of light interaction with materials.

This model is known as:

- A "shading model"
- A "lighting model"
- A "light reflection model"
- A "local illumination model"
- A "reflectance model"

Emissivity

The simplest shading rule is to just give each point a single color.

Gives the shading equation:

$$I = k_e$$

where:

- I is the resulting intensity
- k_e is the "emissivity" associated with the object

Note:

- Emissivity is something we would normally associate with a light source. However, light sources are usually specified in a special way.
- Emissivity in this context is intended to add color to a surface so that it appears as bright as you wish.

Ambient reflection

Suppose that the only light is *diffuse* and *nondirectional* – for example:

- Object suspended in a room coated with latex paint
- Light source is not directly visible to the object

Ambient reflection, cont.

In this case, light reflects from the surface according to its "reflectivity" k_a.

Gives the new ambient shading equation:

$$I = k_a I_a$$

where:

- k_a is the "ambient-reflection coefficient"
  - Really, the "reflectance of ambient light"
  - This reflectance is assumed to be "diffuse" — equal in all directions
- I_a is the "ambient intensity"
  - Really, the "intensity of the ambient light"

Note that this is a really crude approximation to the real challenge of solving for indirect lighting contributions.

Wavelength dependance

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

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

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

and then find good RGB values to respresent $I(\lambda)$.

"Traditionally," though, k_a and I_a are represented in RGB, and the computation is performed on each color component separately:

$$\begin{aligned} I_{\text{red}} &= k_{a,\text{red}}\, I_{a,\text{red}} \\ I_{\text{green}} &= k_{a,\text{green}}\, I_{a,\text{green}} \\ I_{\text{blue}} &= k_{a,\text{blue}}\, I_{a,\text{blue}} \end{aligned}$$

Diffuse reflection

Review of 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?

Problem: So far, objects are all uniformly lit.

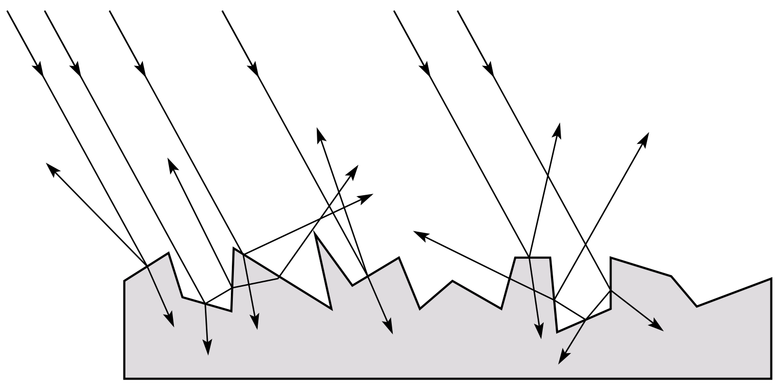
"Diffuse" or "Lambertian" reflection is the simplest kind of reflection that will allow reflected intensity to vary with the direction of the light.

Diffuse reflectors

Diffuse reflection occurs from dull, matte surfaces, like chalk or certain matte paints.

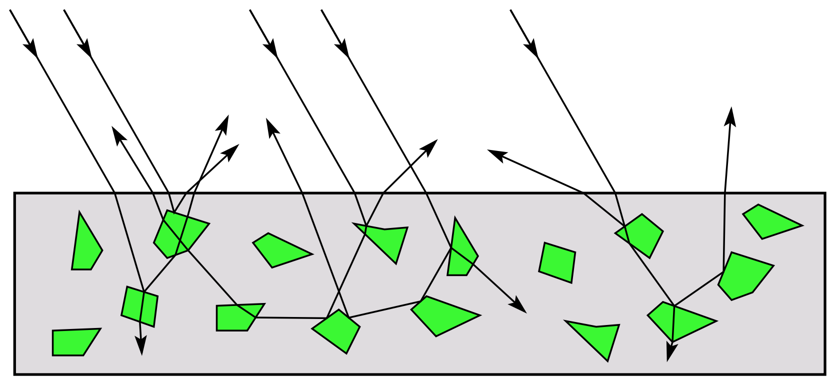
These “diffuse” or “Lambertian reflectors” reradiate light equally in all directions.

Picture a rough surface with lots of tiny “microfacets”:



Diffuse reflectors

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



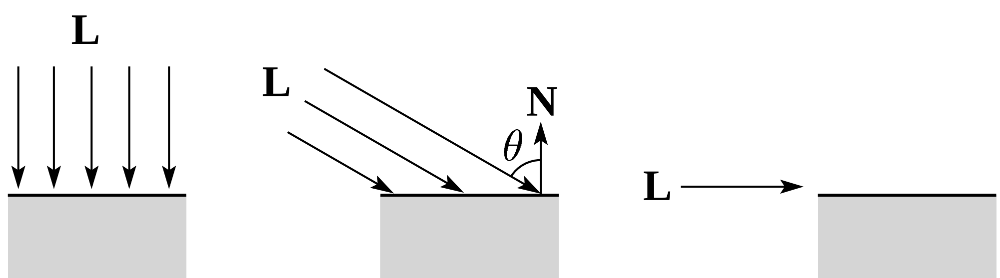
The microfacets and pigments have the effect of distributing light rays in all directions.

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

Note: the figures above give an approximate understanding of diffuse reflection. The physics is actually somewhat more complex.

Diffuse reflectors, cont.

While the reflected intensity from a diffuse surface does not depend on the direction of the viewer, the incoming energy does depend on the direction of the light source:



Diffuse reflectors, cont.

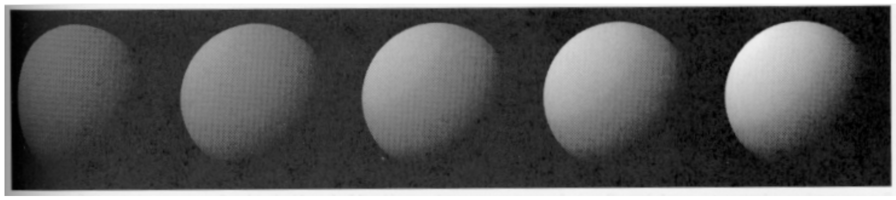
The incoming energy is proportional to ________, giving the diffuse reflection equations:

$$
\begin{aligned}
I &= k_d I_\ell \\
&= k_d I_\ell
\end{aligned}
$$

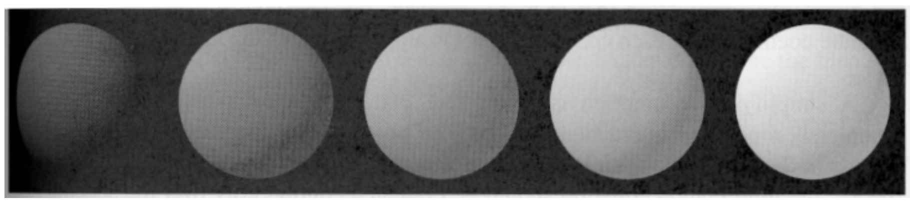
where:

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

Ambient and diffuse examples.



Increasing the diffuse coefficient.



Keeping the diffuse term constant while increasing the ambient.

Specular reflection

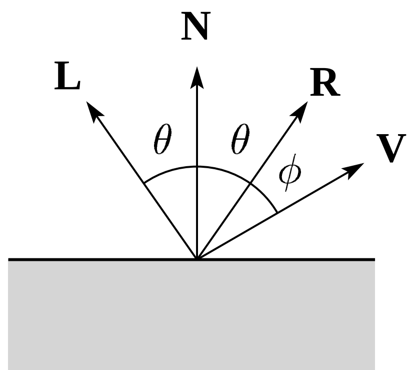
"Specular reflection" accounts for the highlight that you see on objects.

- Particularly important for smooth, shiny surfaces, such as:
    - Metals
    - Polished rocks
    - Plastics
    - Apples

Properties:

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

“Derivation”



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

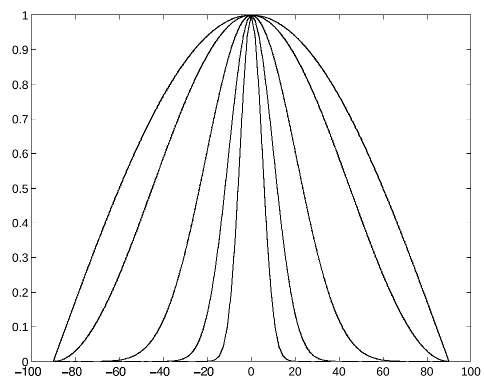
$$I = \begin{cases} I_\ell & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

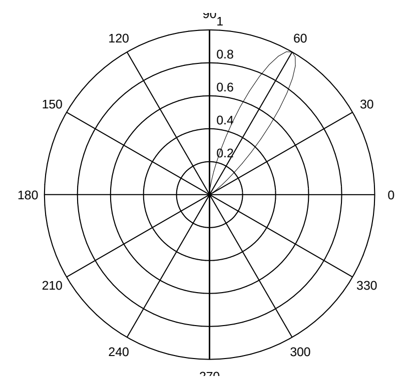
For a *near*-perfect reflector, you might expect the highlight to fall off quickly with increasing angle ϕ.

Known as:

- “Specular” reflection (Phong)
- “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,

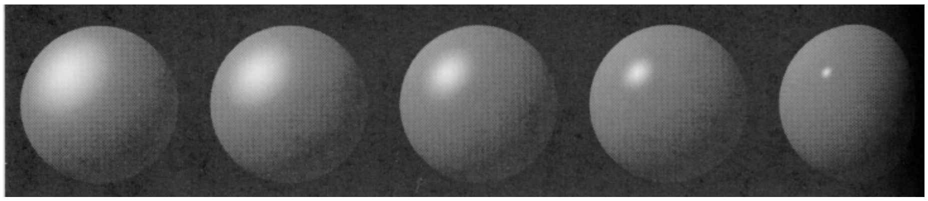
- The drop-off becomes {more, less} gradual
- Gives a {larger, smaller} highlight
- Simulates a {more, less} glossy surface

Thus, the specular component is given by:

$$I = k_s \, I_\ell \, (\mathbf{V} \cdot \mathbf{R})_+^{n_s}$$

Q: What would a polar plot of diffuse reflection look like?

Specular example



Effect of varying n_s.

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 drop-off is considered "too harsh." A slower fall-off is given by:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

for user-supplied constants a_0, a_1, a_2.

In addition, if a_0 is chosen to be small, you might put an upper limit on the intensity using:

$$f(d) = \mathrm{MIN}\left(1, \frac{1}{a_0 + a_1 d + a_2 d^2}\right)$$

The Phong shading model

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

The Phong shading model can be summarized by putting all the terms we have seen together:

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

Note: $k_e, k_a, k_d, k_s, I_a, I_{\ell i}, I$ are all RGB vectors.

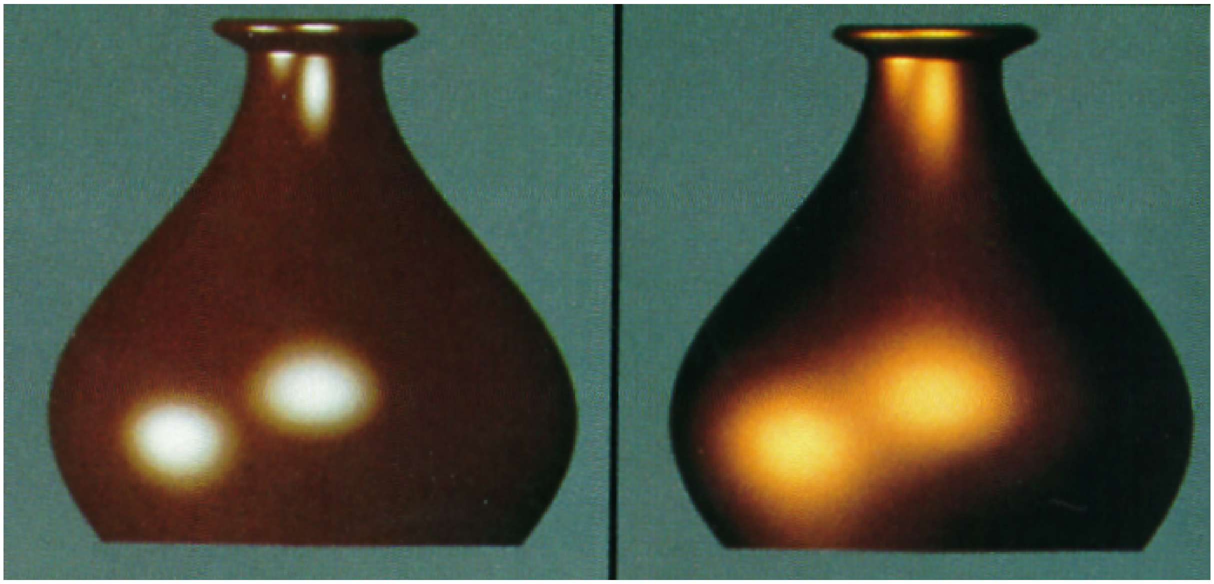
Choosing the parameters

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

- Try n_s in the range [0, 100]
- Try $k_a + k_d + k_s \leq 1$
- 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, cont'd



A vase rendered with plastic vs. copper shading

Image of jupiter

Gouraud vs. Phong interpolation

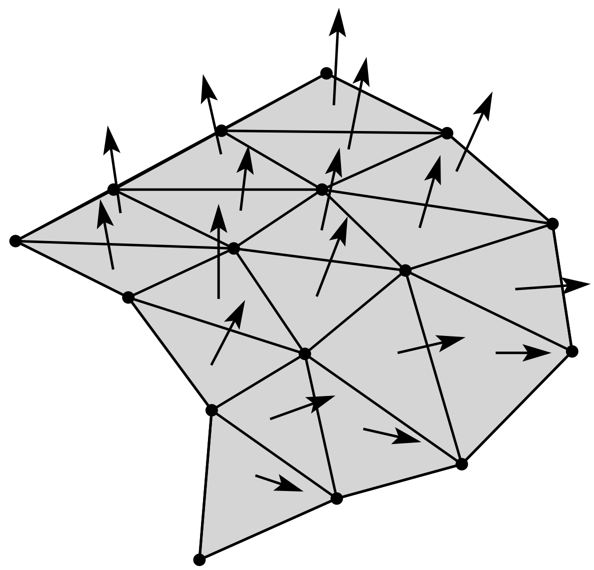
Smooth surfaces are often approximated by polygonal facets, because:

- Graphics hardware generally wants polygons (esp. triangles).
- Sometimes it easier to write ray-surface intersection algorithms for polygonal models.

How do we compute the shading for such a surface?

Faceted shading

Assume each face has a constant normal.



For a distant viewer and a distant light source, how will the color of each triangle vary?

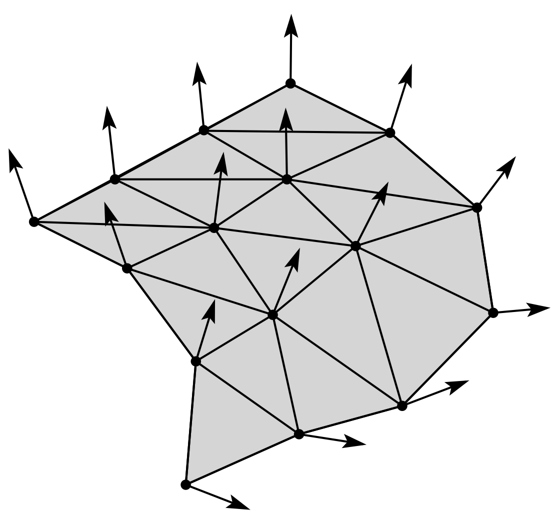
Result: facted, not smooth, appearance.

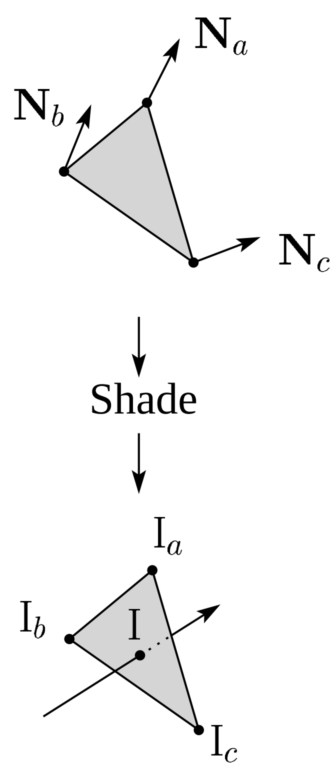
Gouraud interpolation

To get a smoother result that is easily performed in hardware, we can do "Gouraud interpolation."

Here's how it works:

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

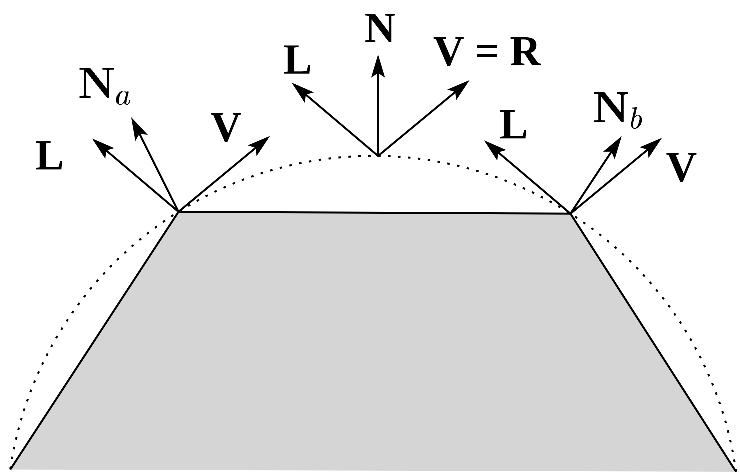




Gouraud interpolation, cont'd

Gouraud interpolation has significant limitations.

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



2. We will encounter Mach banding.

Phong interpolation

To get an even smoother result with fewer artifacts, we can perform "Phong interpolation."

Here's how it works:

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