Shading

CS 465 Lecture 5

Visual cues to 3D geometry

- size (perspective)
- occlusion
- shading

Recognizing materials

- Human visual system is quite good at understanding shading
Shading for Computer Graphics

- Need to compute an image
  - of particular geometry
  - under particular illumination
  - from a particular viewpoint
- Basic question: how much light reflects from an object toward the viewer?

Simple materials

metal

dielectric

Adding microgeometry

Classic reflection behavior

ideal specular (Fresnel)

rough specular

Lambertian
Basics of local lighting

- Diffuse reflection
  - light goes everywhere
  - colored by object color
- Specular reflection
  - happens only near mirror configurations
  - needs to be spread out some for point lights
  - usually white (except colored metals: e.g. copper, gold)
- Ambient reflection
  - don’t worry about where light comes from
  - just add a constant amount of light to account for other sources of illumination

Diffuse reflection

- Light is scattered uniformly in all directions
  - the surface color is the same for all viewing directions
- Lambert's cosine law
  - Top face of cube receives a certain amount of light
  - Top face of 60° rotated cube intercepts half the light
  - In general, light per unit area is proportional to \( \cos \theta = L \cdot N \)

Shading: diffuse reflection

- Assume light reflects equally in all directions
  - therefore surface looks same color from all views: “view independent”
- Illumination on an oblique surface is less than on a normal one
  - generally: illumination falls off as \( \cos \theta \)

Lambertian shading

- Shading independent of view direction
  - illumination from source
    \[ L_d = k_d I \max(0, n \cdot v_L) \]
  - diffuse coefficient
  - diffusely reflected light
**Lambertian shading**

- Produces matte appearance

![Lambertian shading](image)

**Diffuse shading**

![Diffuse shading](image)

**Specular shading (Phong model)**

- Intensity depends on view direction
  - bright near mirror configuration

![Specular shading](image)

**Specular shading (Phong model)**

- Intensity depends on view direction
  - bright near mirror configuration

\[
v_R = v_L + 2((n \cdot v_L)n - v_L) = 2(n \cdot v_L)n - v_L
\]
### Specular shading (Phong model)

- Intensity depends on view direction
  - bright near mirror configuration

\[ \mathbf{v}_R = \mathbf{v}_L + 2(\mathbf{n} \cdot \mathbf{v}_L)\mathbf{n} - \mathbf{v}_L = 2(\mathbf{n} \cdot \mathbf{v}_L)\mathbf{n} - \mathbf{v}_L \]

\[ L_s = k_s I \max(0, \cos \sigma)^n = k_s I \max(0, \mathbf{v}_E \cdot \mathbf{v}_R)^n \]

Specularly reflected light

specularly reflected light

specular coefficient

### Phong model—plots

- Increasing \( n \) narrows the lobe

![Fig. 16.9 Different values of \( \cos^n \alpha \) used in the Phong illumination model.]

### Phong variant: Blinn-Phong

- Rather than computing reflection directly, just compare to normal bisection property

\[ \mathbf{v}_H = \text{bisector}(\mathbf{v}_L, \mathbf{v}_E) = \frac{\mathbf{v}_L + \mathbf{v}_E}{\| \mathbf{v}_L + \mathbf{v}_E \|} \]

\[ L_s = k_s I \max(0, \cos \alpha)^n = k_s I \max(0, \mathbf{n} \cdot \mathbf{v}_H)^n \]

### Specular shading

- Phong and Blinn-Phong

![Specular shading (Phong model)](Foley et al.)
**Diffuse + Phong shading**

![Image of green and blue objects](image)

- Shading does not depend on anything
  - add constant color to account for disregarded illumination and fill in black shadows

**Ambient shading**

- Shading does not depend on anything
  - add constant color to account for disregarded illumination and fill in black shadows

**Putting it together**

- Usually include ambient, diffuse, Phong in one model
  
  $L = L_a + L_d + L_s$
  
  $= k_a I_a + I (k_d \max(0, \mathbf{n} \cdot \mathbf{v}_L) + k_s \max(0, \mathbf{n} \cdot \mathbf{v}_H)^n)$

- The final result is the sum over many lights
  
  $L = L_a + \sum_i (L_d)_i + (L_s)_i$
  
  $= k_a I_a + \sum_i I_i (k_d \max(0, \mathbf{n} \cdot (\mathbf{v}_L)_i) + k_s \max(0, \mathbf{n} \cdot (\mathbf{v}_H)_i)^n)$

**Mirror reflection**

- Consider perfectly shiny surface
  - there isn’t a highlight
  - instead there’s a reflection of other objects

- Can render this using recursive ray tracing
  - to find out mirror reflection color, ask what color is seen from surface point in reflection direction
  - already computing reflection direction for Phong...

- “Glazed” surface has mirror reflection and diffuse
  
  $L = L_a + L_d + L_m$
  
  where $L_m$ is evaluated by tracing a new ray
Diffuse + mirror reflection (glazed)

(glazed material on floor)