Shading Basics

CS 4620

Shading

- Compute light reflected toward camera
- Inputs:
  - eye direction
  - light direction
    (for each of many lights)
  - surface normal
  - surface parameters
    (color, shininess, …)

Diffuse reflection

- Light is scattered uniformly in all directions
  - the surface color is the same for all viewing directions
- Lambert’s cosine law

Lambertian shading

- Shading independent of view direction

\[
L_d = k_d I \max(0, n \cdot l)
\]

illuminated
from source
diffuse
reflect
light
diffuse
coefficient
Lambertian shading

- Produces matte appearance

\[ k_d \]

Diffuse shading

Shadows

- Surface is only illuminated if nothing blocks its view of the light.
- With ray tracing it's easy to check
  - Just intersect a ray with the scene!
- Not so easy for rasterization pipeline
  - Since each triangle is processed separately
  - "Shadow Maps" [Williams] are raster-based alternative

Multiple lights

- Important to fill in black shadows
- Just loop over lights, add contributions
- Ambient shading
  - black shadows are not really right
  - one solution: dim light at camera
  - alternative: add a constant "ambient" color to the shading…
Specular shading (Blinn-Phong)

• Intensity depends on view direction
  – bright near mirror configuration

![Diagram showing specular shading](image1)

Specular shading (Blinn-Phong)

• Close to mirror ↔ half vector near normal
  – Measure “near” by dot product of unit vectors

\[ h = \text{bisector}(v, l) = \frac{v + l}{\|v + l\|} \]

\[ L_s = k_s I \max(0, \cos \alpha)^p = k_s I \max(0, n \cdot h)^p \]

Phong model—plots

• Increasing \( n \) narrows the lobe

![Phong model plots](image2)

Specular shading

![Specular shading](image3)
Diffuse + Phong shading

- Shading that 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

\[ L = L_a + L_d + L_s \]
\[ = k_a I_a + k_d I \max(0, n \cdot l) + k_s I \max(0, n \cdot h)^p \]

- The final result is the sum over many lights

\[ L = L_a + \sum_{i=1}^{N} [(L_d)_i + (L_s)_i] \]
\[ L = k_a I_a + \sum_{i=1}^{N} [k_d I_i \max(0, n \cdot l_i) + k_s I_i \max(0, n \cdot h_i)^p] \]

Ambient shading

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

LightMaterial Demo

OpenGL Tutors program by Nate Robins
http://www.xmission.com/~nate/tutors.html
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” material has mirror reflection and diffuse
  \[ L = L_d + L_m \]
  – where \( L_m \) is evaluated by tracing a new ray

\[
r = v + 2((n \cdot v)n - v) = 2(n \cdot v)n - v
\]