CS 419: Production Rendering

Shadows

Eric Shaffer
Revisiting the Phong Reflectance Model

What happens when the light is behind the object?

\[ I_p = k_a i_a + \sum_{m \in \text{lights}} (k_d (\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s (\hat{R}_m \cdot \hat{V})^\alpha i_{m,s}). \]
Revisiting the Phong Reflectance Model

- Clamping negative cosine values to zero is a typical solution

\[ I_p = k_a i_a + \sum_{m \in \text{lights}} (k_d (\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s (\hat{R}_m \cdot \hat{V})^\alpha i_{m,s}). \]
Simple Shading

- We model three types of light reflected/emitted by surfaces
  - Ambient
    - Light that has bounced around a scene…environmental light
  - Specular
    - Light reflected at surface
  - Diffuse
    - Light that has undergone transmission, absorption, scattering
Simple Shading

- Shading
  - Using an equation to compute outgoing radiance $L_o$
  - Along a view ray $\mathbf{v}$
  - Based on material properties and light sources
- We will focus on diffuse and specular shading
  - $c_{\text{diff}} = \text{diffuse color}$
  - $c_{\text{spec}} = \text{specular color}$
Diffuse Radiance (book version)

- Depends on
  - light irradiance
  - light direction vector \( l \)
  - surface normal \( n \)
  - diffuse color \( c_{\text{diff}} \)

\[
L_{\text{diff}} = \frac{c_{\text{diff}} \otimes E_L \max(n \cdot l, 0)}{\pi}
\]
Irradiance

- Sum of energies of photons passing through a surface in 1 second per unit time and unit area
- Can represent it as an RGB vector
- Surface irradiance

\[ E = E_L \cos(\theta_i) = E_L \max(n \cdot l, 0) \]
Specular Radiance (another version)

This one is from *Real-Time Rendering*

\[ M_{\text{spec}} = c_{\text{spec}} \otimes E_L \max(n \cdot l, 0) \]

\[ h = \frac{l + v}{\|l + v\|} \]

\[ L_{\text{spec}}(v) = \frac{m + 8}{8\pi} \cos^m \theta_h M_{\text{spec}} \]
Varying Smoothness
Total Radiance and End Notes

- Total radiance = ambient + specular + diffuse
- What we’ve done here is like Blinn-Phong equation (1977)
- You’ll see variations of these...all produce similar results
- This simple model is usually used in real-time applications
- We’ll look at more advanced modeling based on BRDFs later
Shadows

- Easy(?) to implement, can be computationally expensive
- Lights
  - point light has a position, emits light isotropically
  - directional lights have direction but no position
- For idealized lights (point and directional) shadows are hard-edged
Shadows

How far are the objects above the plane?

What is their distance from the cameras and relative sizes?

How many lights are there?
Real Lights...Soft Shadows

Real lights have a finite area

Umbra is the shadow where no light is visible

Penumbra is partial light

(a)          (b)          (c)
What are the relative positions of the light and eyepoint?
Real Lights...Soft Shadows
Implementation

- Determine visibility of light by ray-casting
  - Shadow ray origin is an object-primary ray hit point
  - Direction is the light direction
  - For point lights, use light position – hit point

![Diagram of ray-casting and shadow rays]

- shadow rays
- primary rays
- $b$
- $a$
What Went Wrong?
What Went Wrong?

- This shadow ray hits the plane.
- This shadow ray doesn't hit any objects.

Plane
What Went Wrong?
Rays from Objects

- Need to add an $\varepsilon$ value to ray origin
- Move it slightly in direction of ray...
- Otherwise, numerical issues can result in hitting the object surface
- Define a constant for each geometric object type
Shadows can be very expensive

arbitrary number of shadow rays

single primary ray