



Lecture 25 – Surface Appearance

Notes – not for display
Mike Gleicher
October, 2007

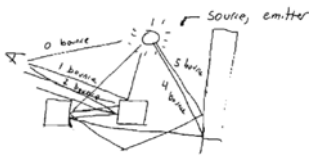


What color are things?

- How do things appear?
- Influenced by object and lighting
- Qualities of the surface
 - Shape + Material
- Qualities of the light / interaction



Lighting in the real-world



- Light bounces off everything
- All objects influence all others
- Appearance (illumination) is a **global** problem



Global Illumination

- All objects affect all other objects
- Interactions, interdependence, ...
 - Spill, reflections, shadows, ...
- Important for advanced effects!
- Very hard to do
 - CG illumination needs to make simplifications
 - Global illumination is done – but simplified
 - Still an advanced topic



Local Lighting

- What happens when light interacts with an object
 - Arguably, part of global lighting
- Again, surface interaction could be really complex – simplify to make easy
- Use only local lighting for most CG



Local Lighting in CG

- How to light **1 point** on a surface
- Surface at that point
 - (material, local geometry)
- Eye Position
- Lights
 - Position/direction, color, intensity
- Appearance of object only depends on the point – not other points

Local Lighting



- What can't we do?
 - No shadows
 - No self-shadows
 - No color spill
 - No inter-reflection / reflection / refraction
 - No area light sources
- Add these effects in with hacks

The generalized model



- Color of a point is determined by **shading**
- Shading considers all local information, gives color
 - Old days, fixed function (lighting)
 - Now/future - programmable

The fanciest local models



- BRDF
 - Bi-Directional, Reflectance, Distribution Function
- Any direction in, any direction out, what colors are transmitted

The common model



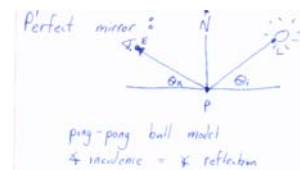
- Simplified, time-tested
- Originally built into hardware (and OpenGL)
- Now, fixed function is going away
 - (but this still gets used a lot)
- Can use with anything...

Simplified local lighting model



- 3 parts per light
 - Specular (direct reflection)
 - Diffuse (scattering)
 - Ambient (hack for indirect lighting)
- This is a hack – but a well established, common hack that gets the main phenomena
- Fancier local models exist

Specular (Direct Reflection)

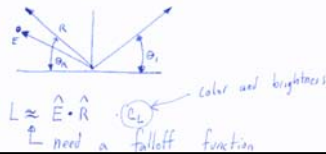


- Mirror reflection
- Light gets to eye if it bounces perfectly

Specular Lighting



- Real surfaces aren't perfectly smooth
 - So don't need to be exact
 - Shinier (smoother) more exact bounces
- Fall off as get away from direct reflection
 - Rate of falloff depends on "shininess"



Specular Lighting – Phong Model



Phong Model $L \approx (\hat{E} \cdot \hat{R})^p C_L$ specular Co-efficient

Easier Way $H = \text{half-way angle}$

$$L = (\hat{N} \cdot \hat{H})^p C_L$$

Diffuse Lighting



- Really rough surfaces (chalk)
- Matte objects
- Light is scattered in all directions equally
 - Randomness of surface direction at a micro-level, still does mirrors, but they are small
- Lambertian reflector

Diffuse Lighting



- Light scatters in all directions equally
- Eye position doesn't matter
- Light position does matter

Diffuse Lighting (Lambertian)



consider fixed sized object:



amount of light that hits is $\approx \cos \theta$ where $\theta = \angle$ between light and normal

$$D_L \approx \hat{n} \cdot \hat{i}$$

Ambient Lighting



- What about indirect lighting?
 - Room isn't totally black
 - Should have SOME color
- Ambient light is indirect light that is just bouncing around
- HACK: add in a constant light source that just lights up everything

The entire lighting Model



- Eye position (direction vector)
- Object local geometry (normal)
- Each light source
 - Position (maybe infinity) + color
- Ambient light has a color (and amount?)
- Surface has a color for each light type and shininess (Cd, Cs, Ca, s)

The final model



$$color = A * C_A + \sum_{i \in \text{lights}} (I_i * (C_D * (\hat{N} \cdot \hat{L}) + C_S (\hat{N} \cdot \hat{H})^S))$$

- Ca, Cd, Cs = object colors
- La, Ld, Ls = light colors
- S = shininess
- N = normal, L = light direction, H = half angle