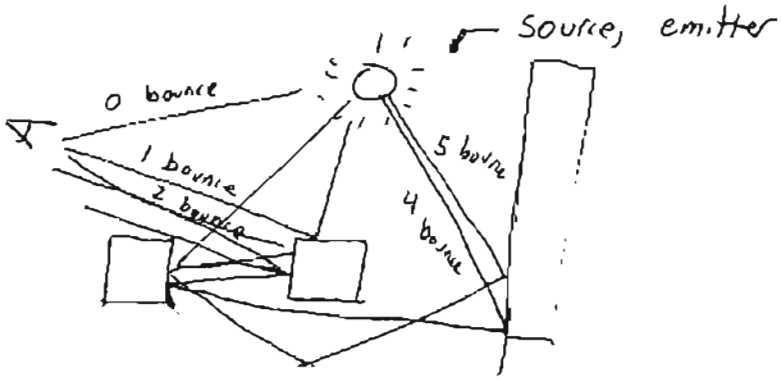
CS 559-2006 Lecture 19 - Lighting
Notes by Mike Gleicher (from previous years)

What Color do we make these solid objects?

Depends on object AND Light!

How Does Lighting Work?



In the real world .....

Light bounces off everything

All objects influence all others

Global Illumination

hard to do - must consider all objects, interactions, ... interdependence (1 depends on 2 depends...)

good for getting complex lighting effects

an advanced topic

In the CG world .....

Local Lighting -

decision of how to light a point on ~~the~~ an object depends on:

- surface AT that point
- eye position
- lights

LOCAL LIGHTING

Consider only 1 point on 1 object

- No shadows
- No self shadows
- No color spill
- No inter-reflection

} if you want these, add with a hack

- No area light sources - point sources only
  - might be at infinity

3 parts (per light)

- specular (direct reflection)
- diffuse (scattering)
- ambient (hack for indirect)

Lighting is a hack

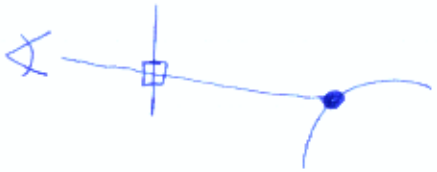
real lighting is complex
microstructure of materials

get "biggest" features of lighting correct

fancier models are still hacks + just get more features right

LIGHTING & SHADING

What color is a point?



Physics: depends on how light interacts with all objects in scene
- some of the objects reflected light goes off towards eye ⇒

CG: do some computation to determine color Shader

color = Shader (info)

↳ what info do we give the shader?

Simple Shading:
object properties (color) ← reflectance
light info (position, color, intensity)
eye position
local geometry (position, normal)

Diffuse Shading -

matte objects
rough surfaces
"micro surface texture" scatters light in all directions

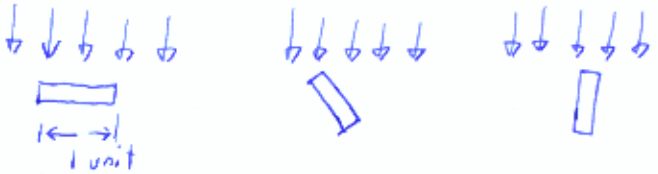
chalk, paper, unpolished wood or stone, ...

Lambertian reflector
scatters light in all directions equally

eye position doesn't matter

light position DOES matter
(relative to surface orientation)

consider fixed sized object:



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

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

One last problem -

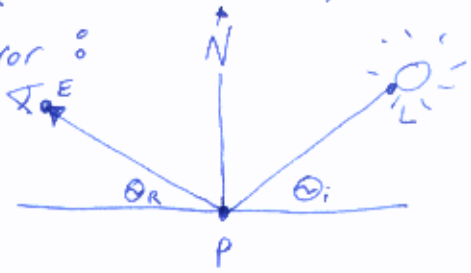
What about inter-reflected light -
room isn't totally black

 ← this side of object should have some light

"Ambient" light ≡ indirect light that is just bouncing around

Hack ∴ add in a light source that effects all objects equally - Ambient lighting

Specular (direct reflection)
Perfect mirror

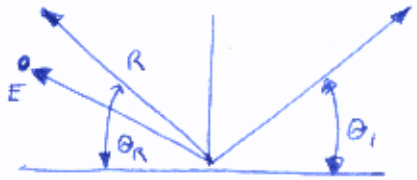


ping-pong ball model
$\measuredangle$ incidence = $\measuredangle$ reflection

light gets to eye only if things line up exactly

HACK → if it's close to the eye, that's good enough
falloff as it gets further away

~~Before~~



$L \approx \hat{E} \cdot \hat{R}$ · C_L ← color and brightness
↑ need a falloff function

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

Easier Way H = half-way angle

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

Hack Lighting Model (GL)

1. Eye Position
2. Object Local Geometry (NORMALS)

3. Each light source has a position (may be at infinity) and a brightness (color) I_i
4. Ambient light has a brightness (color) A

5. surface has a diffuse reflective color C_D
   a specular color C_S
   a shinyness S
   an ambient color (reflectivity) C_A

white or $= C_D$

these are usually the same

$$\text{color} = A * C_A + \sum_{i \in \text{lights}} \left(I_i * \left(C_D \cdot (\hat{n} \bullet \hat{l}) + C_S (\hat{N} \cdot \hat{h})^S \right) \right)$$

Some improvements :

1. falloff (brightness depends on distance)
2. more sophisticated ways of finding C_D, C_S based on position

3. more complex reflectance functions

BRDF $\triangleq$ bi-directional reflectance distribution function

given -

input direction ; output direction $\Rightarrow$ reflectance

How to use this?

Polygons are all the same color (one normal)
FLAT shading
⇒ approximation $\hat{L}$ and $\hat{E}$ do change, only a little

Problem:
polygons are an approximation to a smooth surface
normal per vertex

(1) compute color at vertices
linearly interpolate color

GOURAUD Shading

(2) linearly interpolate normals
compute lighting per-pixel

PHONG SHADING

(do not confuse with Phong LIGHTING)