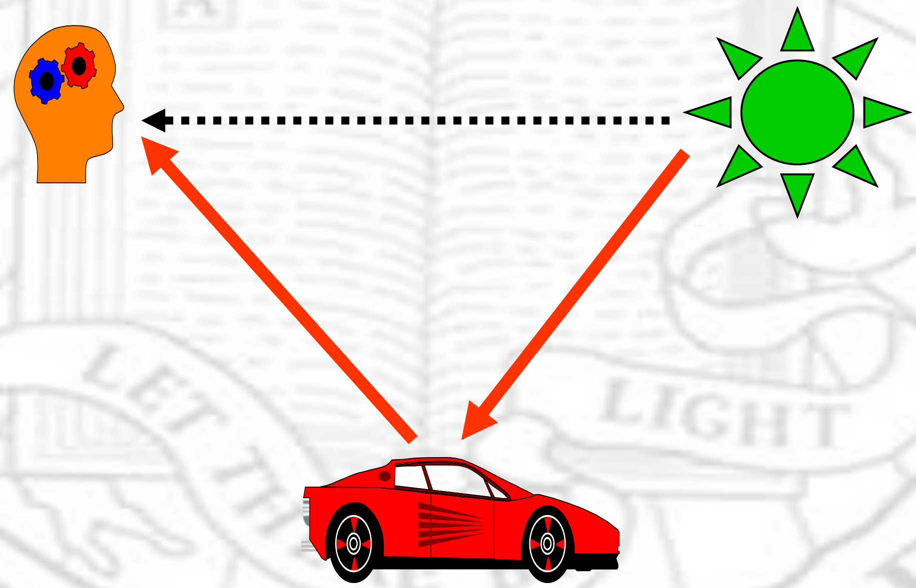
Lighting and Shading Models



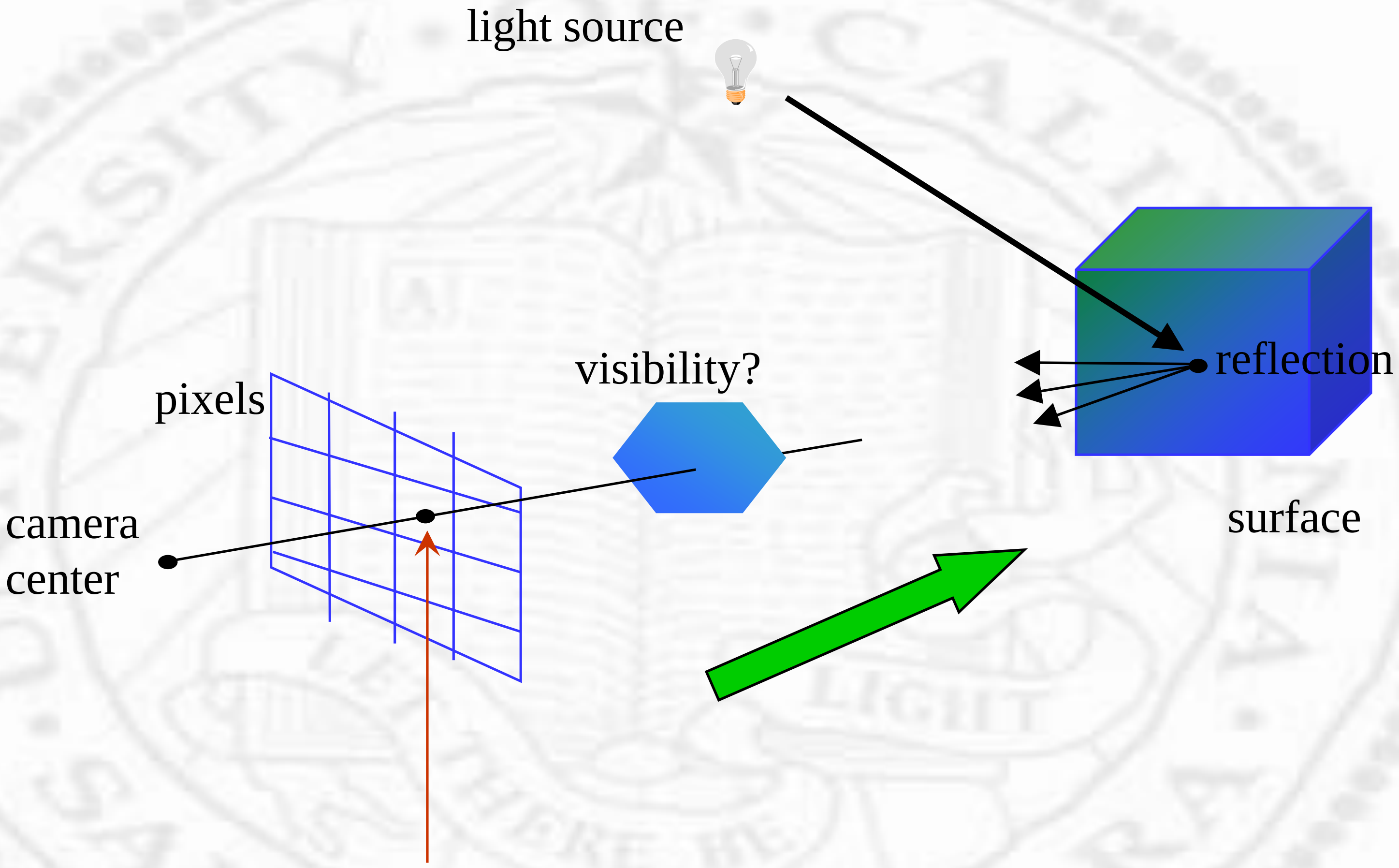
Geometry and Radiometry

- In creating and interpreting images, we need to understand two things:
  - Geometry – Where scene points appear in the image (image locations)
  - Radiometry – How “bright” they are (image values)
- **Geometric** enables us to know something about the scene location of a point imaged at pixel (u, v)
- **Radiometric** enables us to know what a pixel value implies about surface lightness and illumination

Radiometry

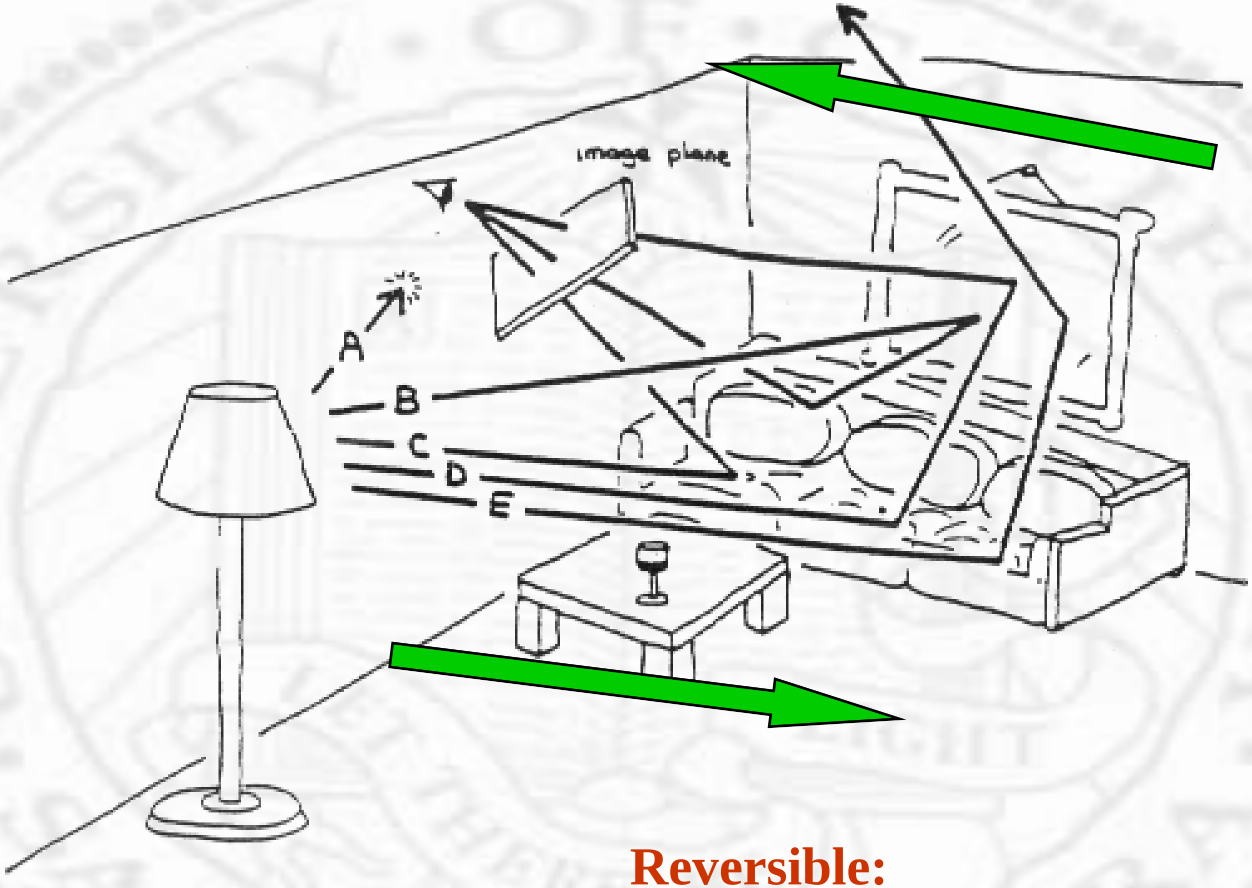
- Radiometry is the measurement of light
  - Actually, electromagnetic energy
- Imaging starts with light sources
  - Emitting photons – quanta of light energy
  - The sun, artificial lighting, candles, fire, blackbody radiators …
- Light energy interact with surfaces
  - Reflection, refraction, absorption, fluorescence…
  - Also atmospheric effects (not just solid surfaces)
- Light energy from sources and surfaces gets imaged by a camera
  - Through a lens, onto a sensor array, finally to pixel values – an image!

Computer Vision



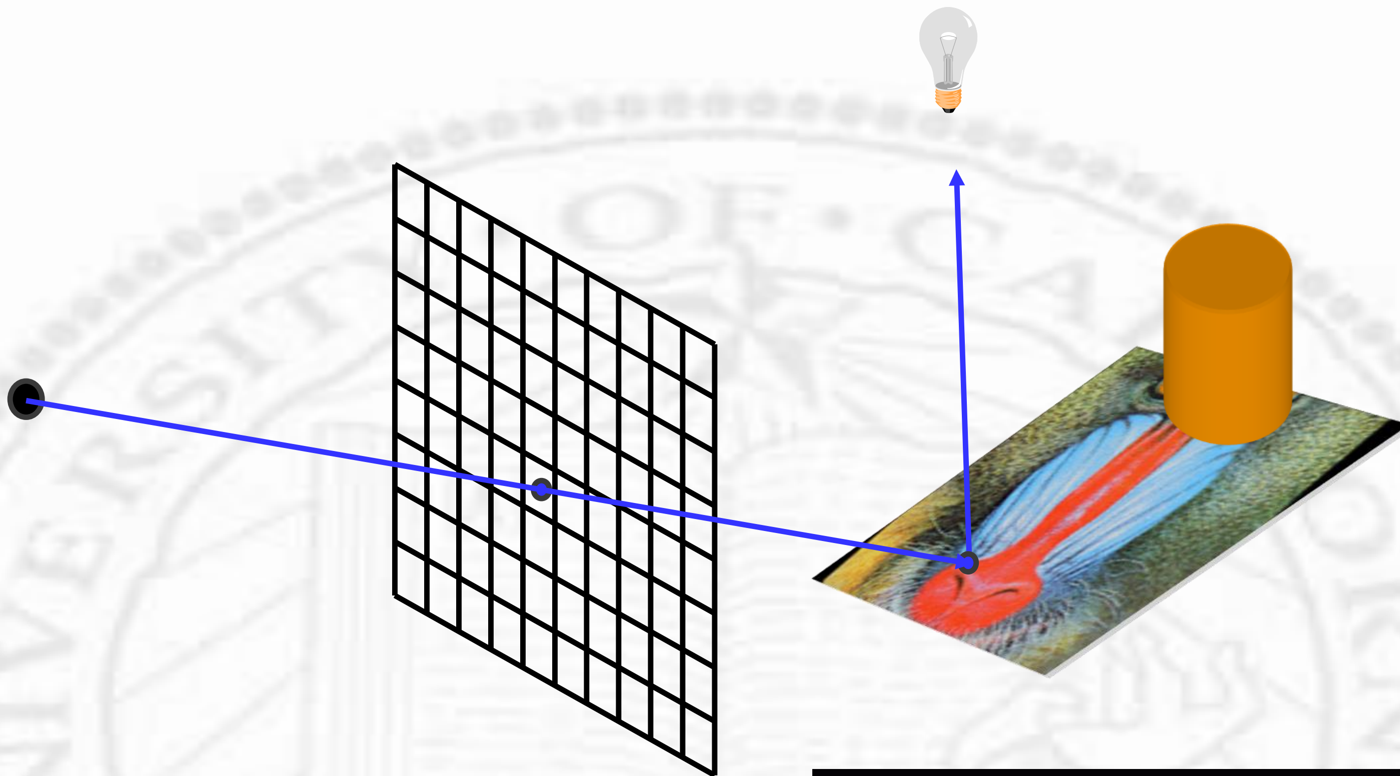
What's the intensity value (and color) at this pixel?

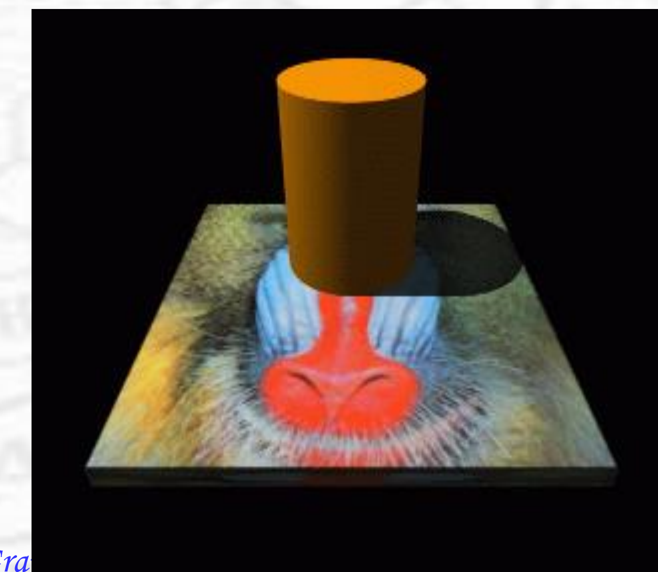
Computer Graphics



Reversible:

- **From camera to light sources**
- **From light sources to camera**





Computer graphics example

CG example: Pixar

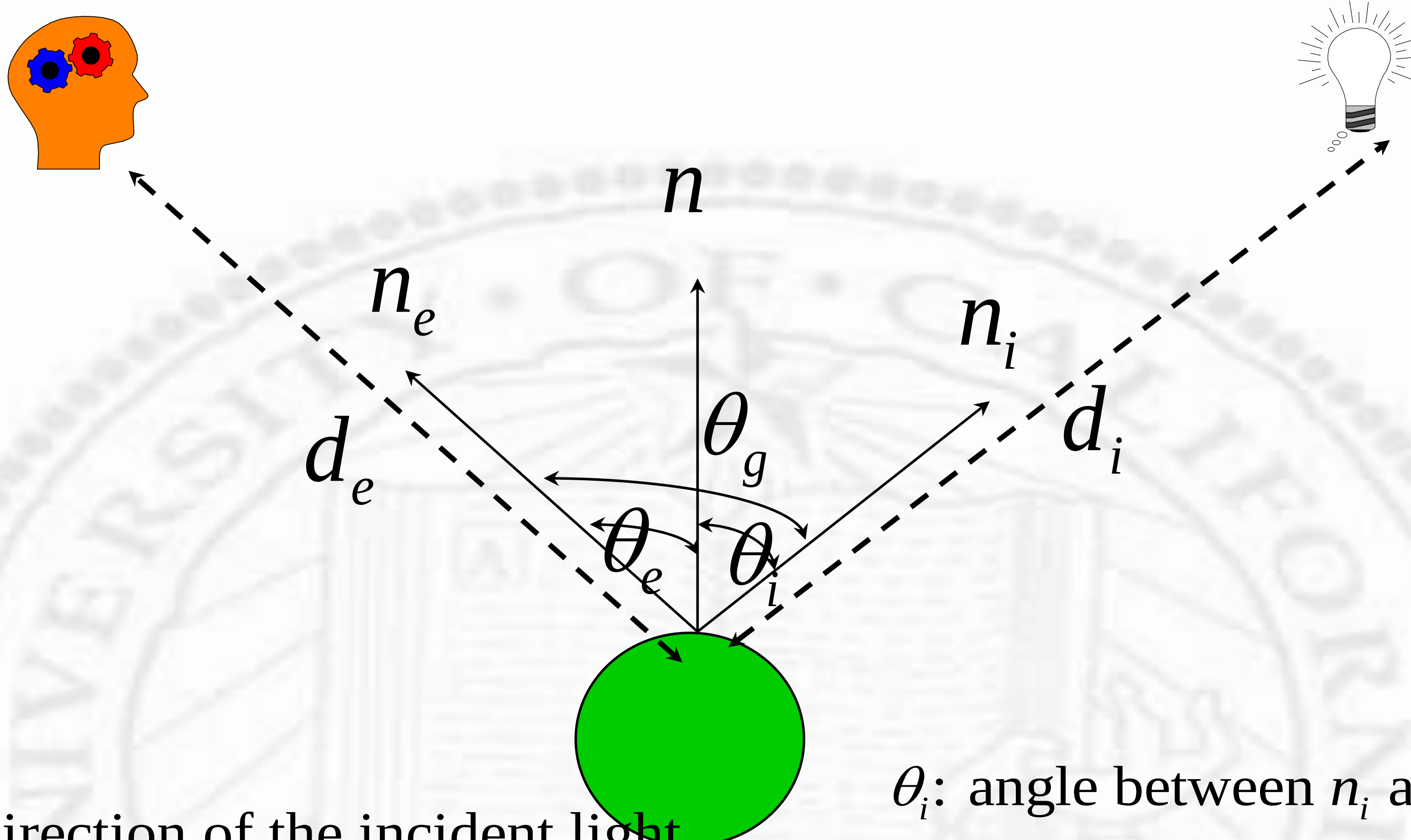
Geri's Game
1997 Oscar Award
Best Animated Short Film

Simple Shading Models

- A jumbled collection of *ad hoc* & *heuristic* techniques, developed over the past two decades
- Concerned mostly with the *primary* ray (light source *to* surface *to* viewer)
- Secondary, tertiary, etc. reflection *not* considered
- Shading individual points and polygons
- Shadow, texture, etc.

Simple Shading Models

- Color (Shading) = f (light source, surface material, geometry, viewer perception model, etc.)
  - light sources: color (spectrum distribution), position, orientation, spatial extent, etc.
  - surface material: orientation, reflectivity, transparency, roughness, etc.
  - geometry: distance, relative orientation, etc.
  - viewer perception model: color model, sensitivity, etc.



n_i: direction of the incident light

n: surface normal direction

n_e: direction to the observer (camera)

θ_i: angle between n_i and n

θ_g: angle between n_i and n_e

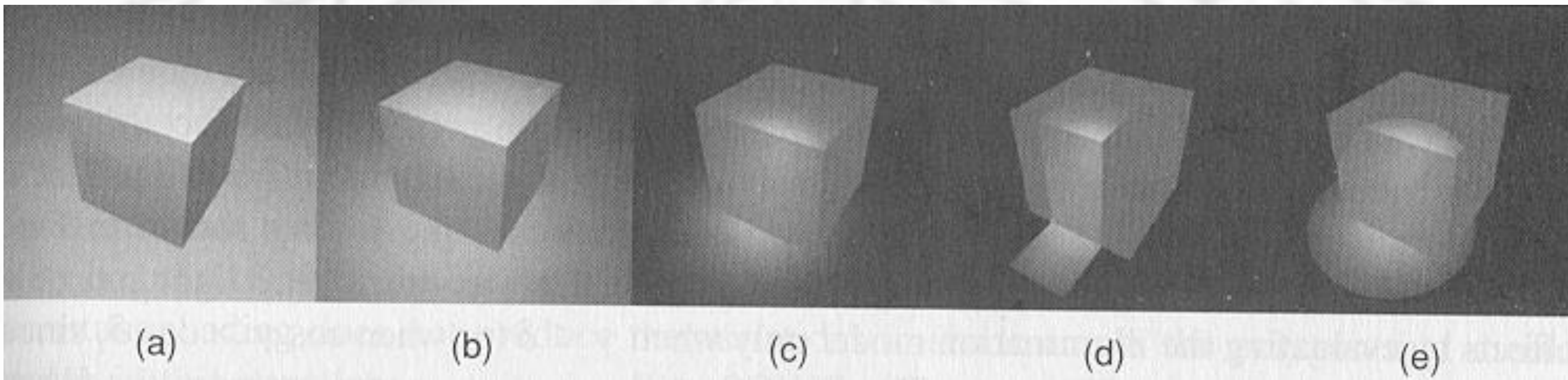
θ_e: angle between n_e and n

d_i: distance from the light source to the object

d_e: distance from the object to the camera

Light sources

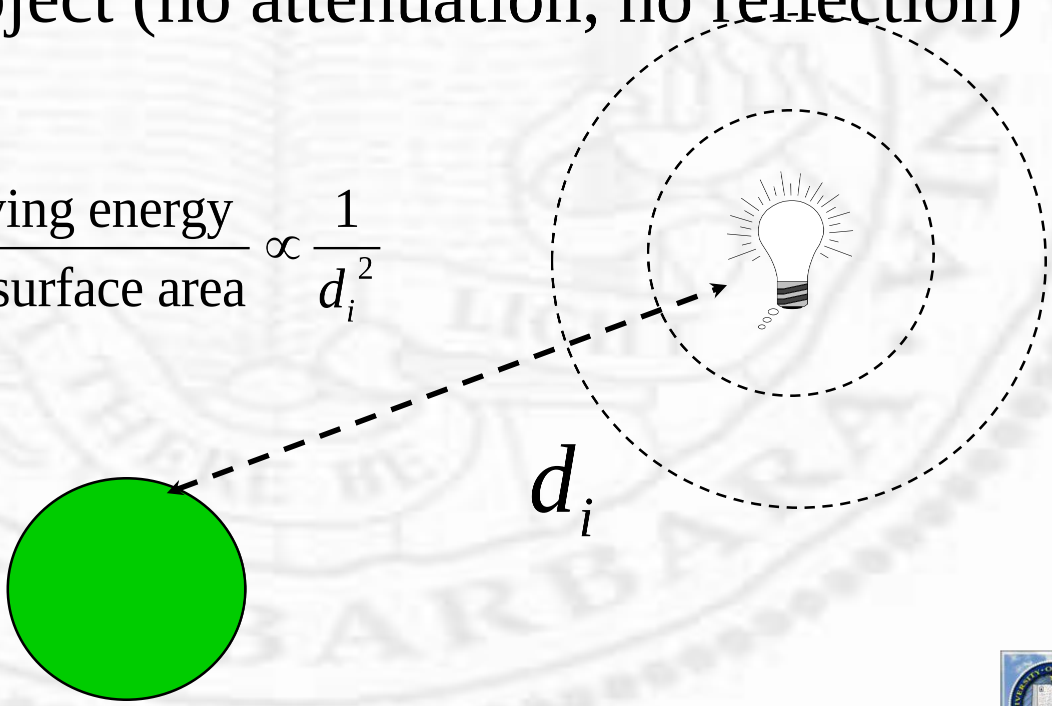
- Spectral properties: R-G-B, H-S-V, etc.
- Strength: characterized by its radiance (joules/sec m^2 sr, watts/m^2 sr, energy/unit-time-area-solid-angle)
- Geometry:
  - Point source (location only, e.g. bulb)
  - Directional source (orientation only, e.g. Sun)
  - Ambient source (no location nor orientation)
  - Spot light (point source + spread angle)
  - Flap, barn-door (directional source + spatial extent)


(a)
(b)
(c)
(d)
(e)

Arriving Light

- Light *arriving* at a surface
  - Strength: characterized by its irradiance (joules/sec m^2, watts/m^2, energy/time-area
  - Distance: how much emitted energy actually gets to the object (no attenuation, no reflection)

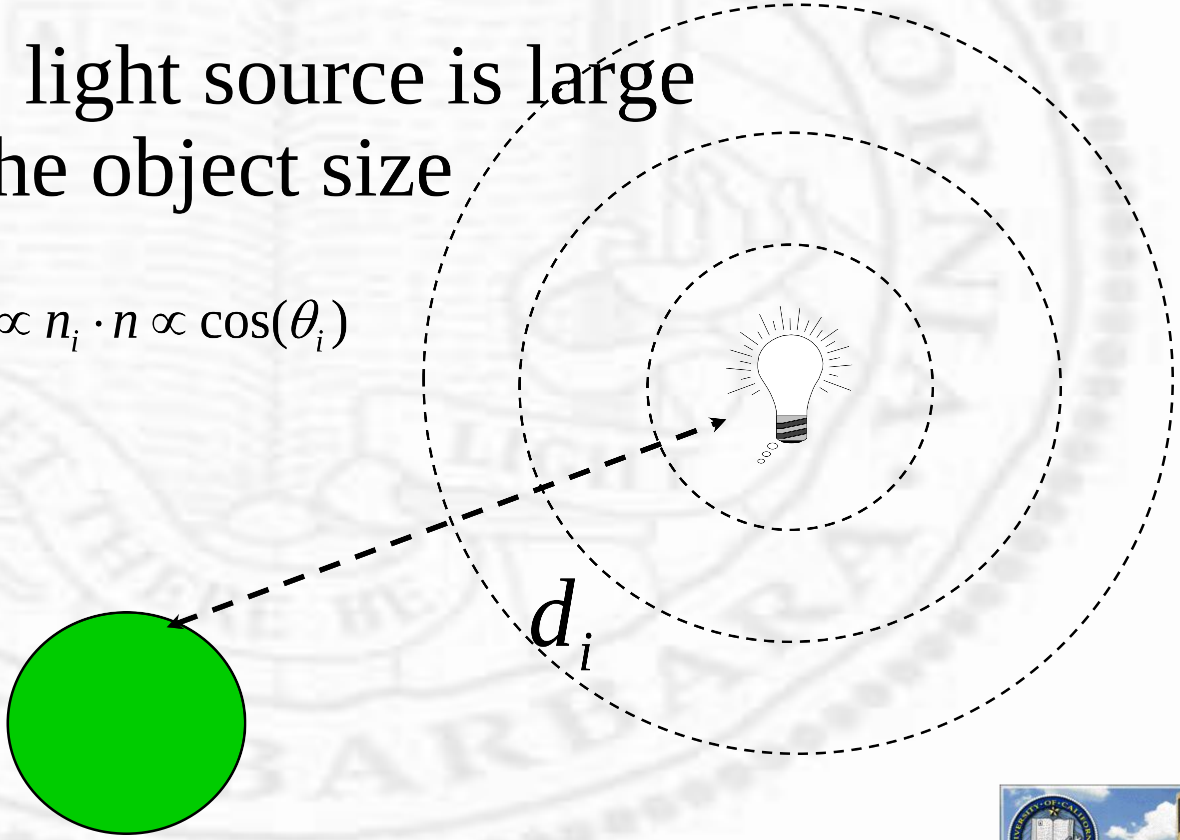
$$\frac{\text{arriving energy}}{\text{unit surface area}} \propto \frac{1}{d_i^2}$$



Incident Light

- Relative orientation: how much emitted energy actually incident on the object
- Follow cosine law $n_i \cdot n = \cos(\theta_i)$
- Distance to the light source is large comparing to the object size

$$\frac{\text{incident energy}}{\text{unit surface area}} \propto n_i \cdot n \propto \cos(\theta_i)$$



Exiting Light

- How much comes out and in what direction?
- Three things can happen
  - absorption
  - reflection (the same side)
    - diffuse (no dominant direction e.g. chalk, cloth)
    - specular (w. a dominant direction e.g. waxed apple, mirror)
  - refraction (the opposite side)
    - diffuse (no dominant direction)
    - specular (w. a dominant direction)
  - absorption + reflection + refraction = total incident

Surface reflectance function $f(\theta_i, \theta_e, \theta_g)$

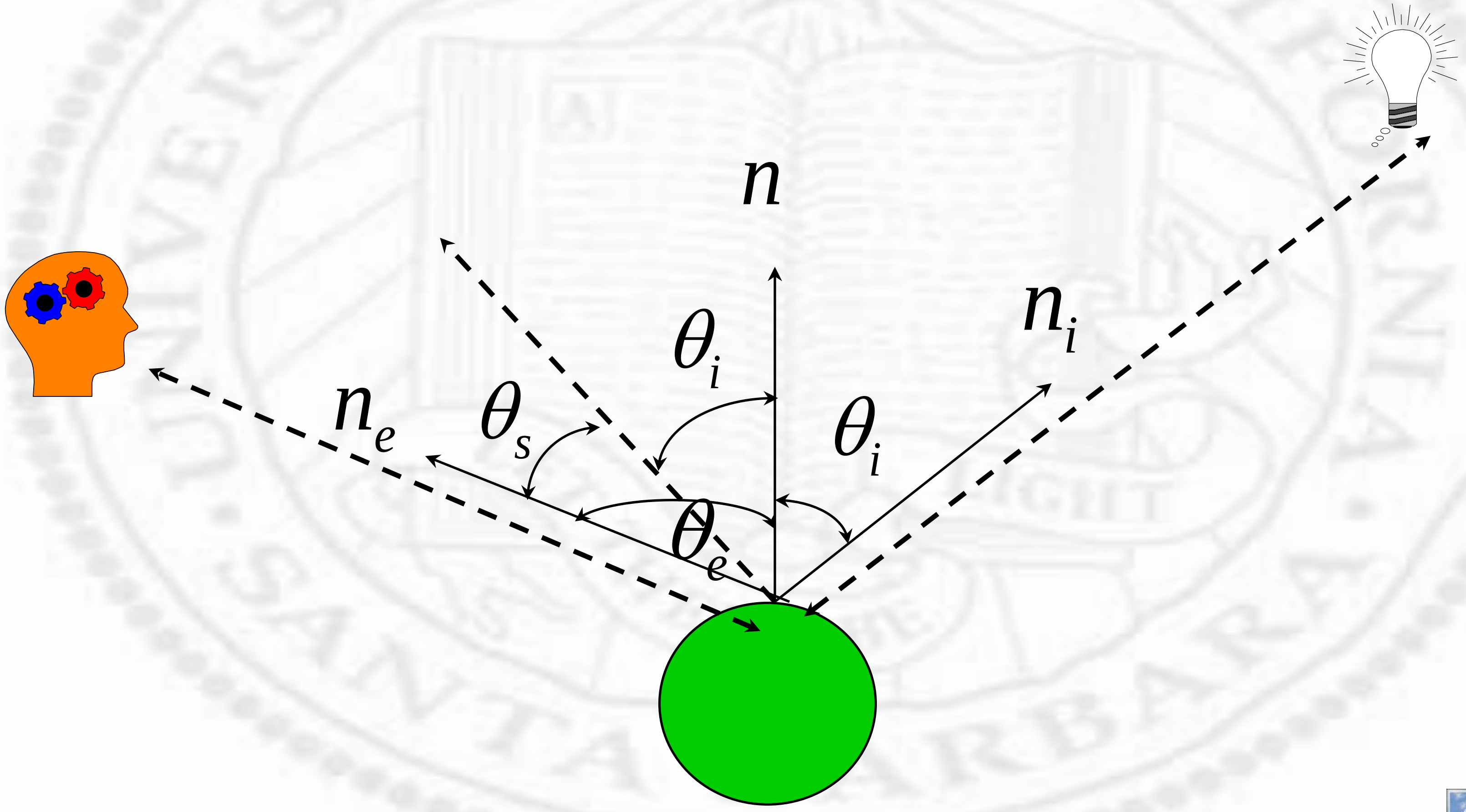
- Fraction of incident light from the incident direction to the viewing direction per unit surface area per unit viewing angle
- Diffuse (Lambertian) reflection
- Ideal specular (Mirror) reflection

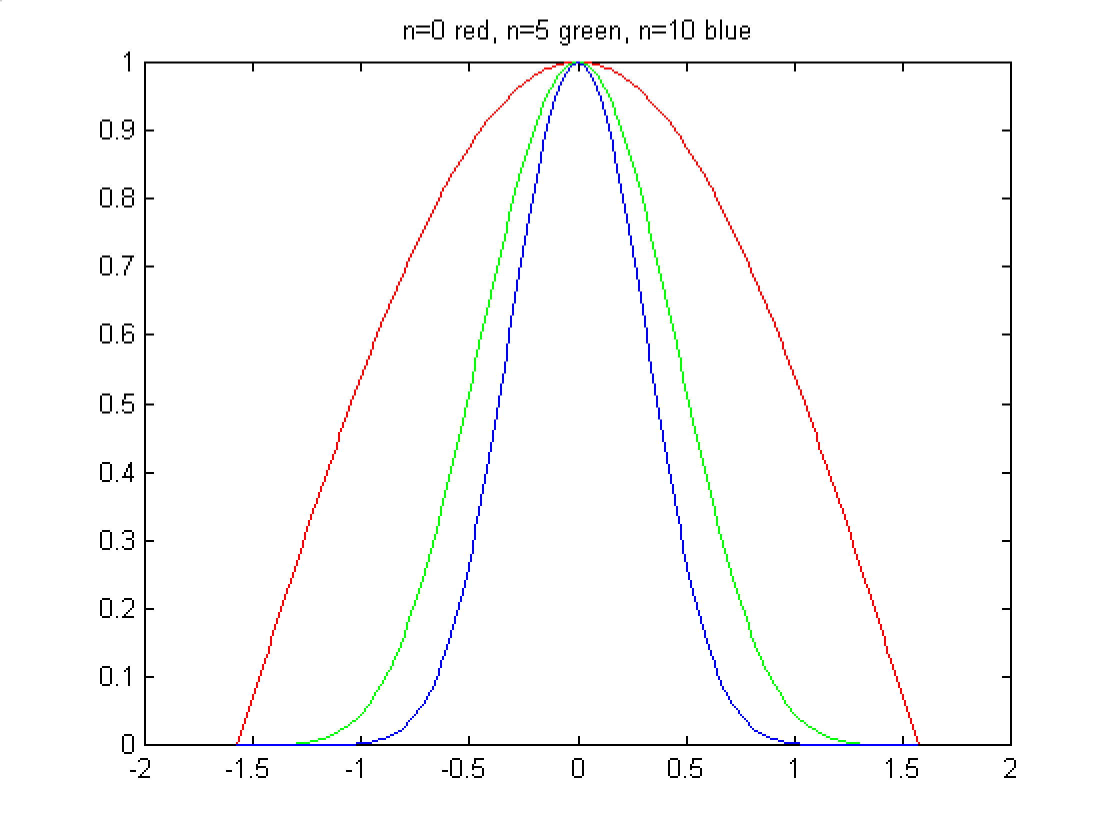
$$f(\theta_i, \theta_e, \theta_g) = k_d$$

$$f(\theta_i, \theta_e, \theta_g) = \begin{cases} k_s & \theta_i = \theta_e, \theta_g = 2\theta_i = 2\theta_e = \theta_i + \theta_e \\ 0 & otherwise \end{cases}$$

Specular (Mirror) reflection

$$f(\theta_i, \theta_e, \theta_g) = k_s \cos^n(\theta_s) \propto k_s(2\cos(\theta_i)\cos(\theta_e) - \cos(\theta_g))^n$$

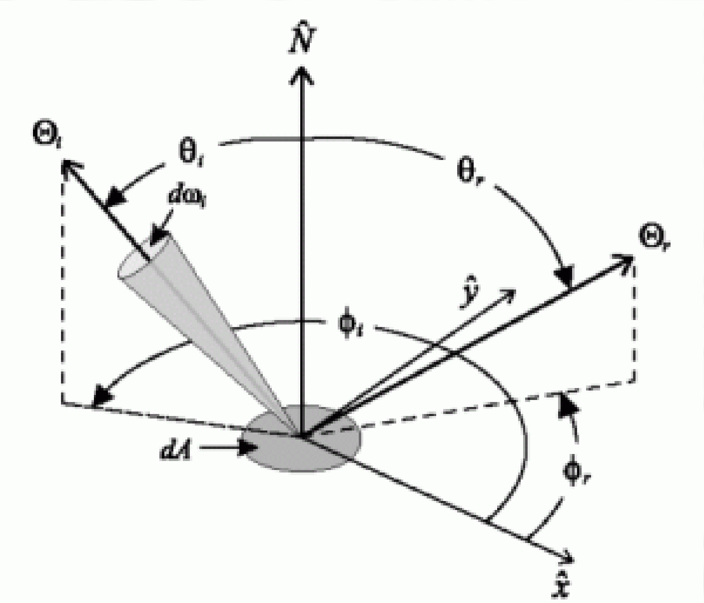


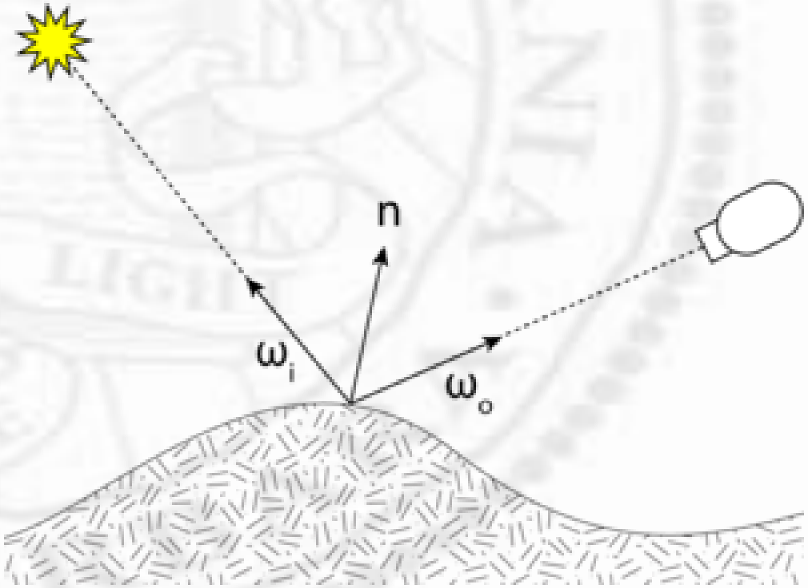

n=0 red, n=5 green, n=10 blue
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

BRDF

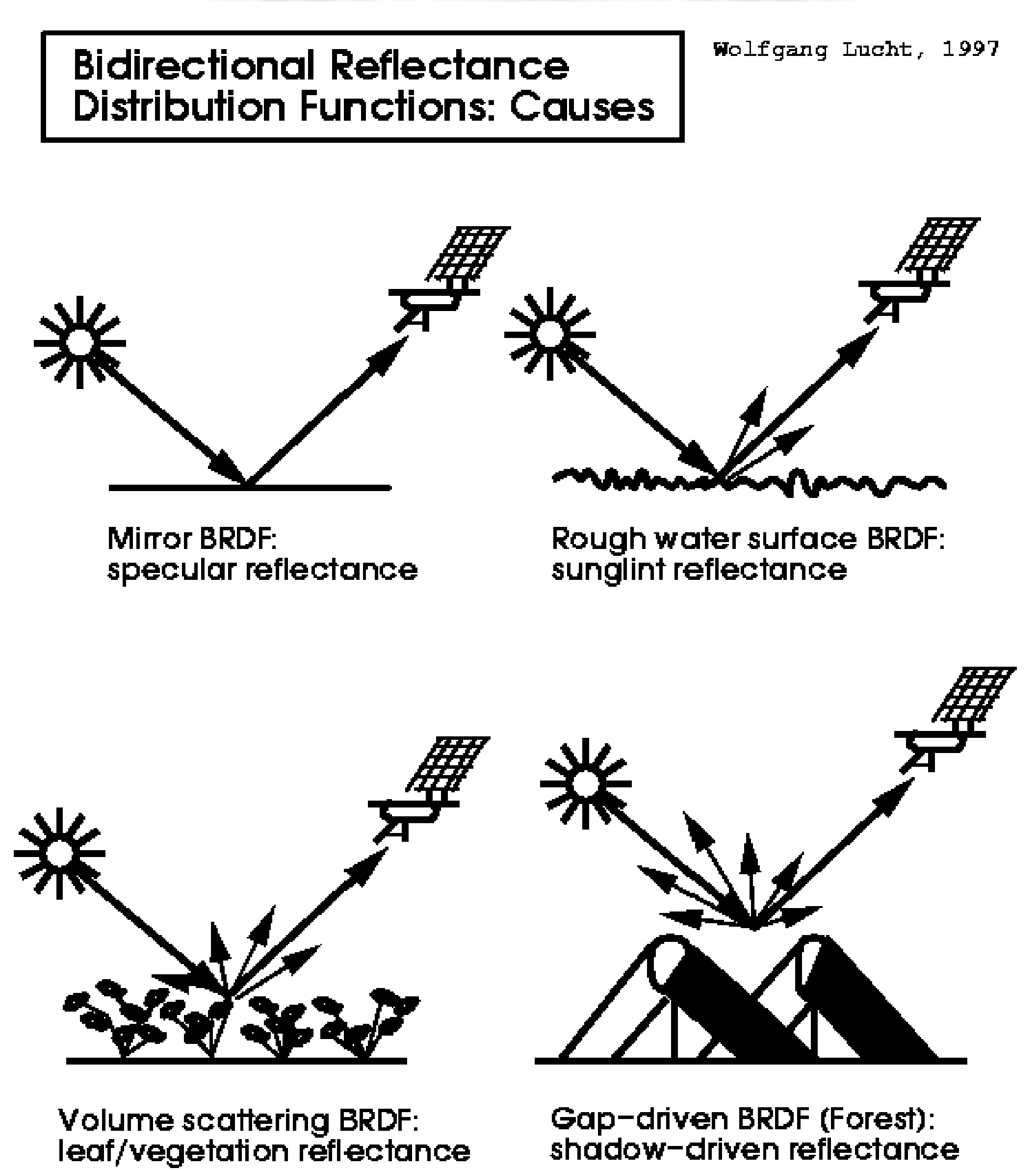
- Bi-directional reflectance distribution function
- 4-dimensional function (angles are parameterized by azimuth and zenith angles)

$$f_r(\omega_i, \omega_o) = \frac{dL_r(\omega_o)}{dE_i(\omega_i)} = \frac{dL_r(\omega_o)}{L_i(\omega_i)\cos\theta_i \, d\omega_i}$$

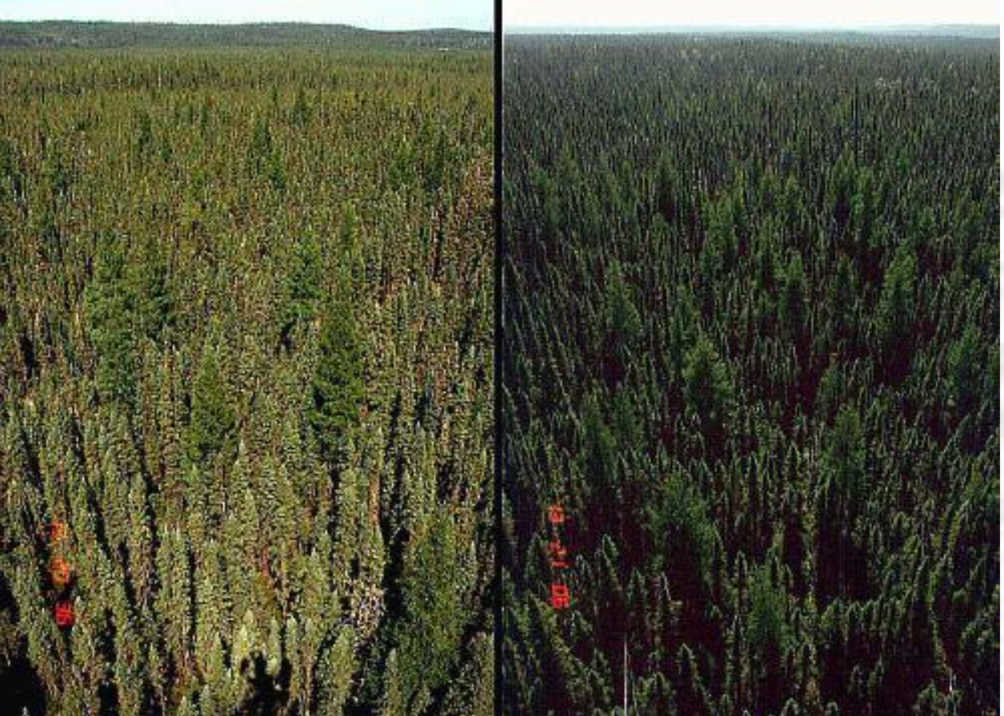




Example

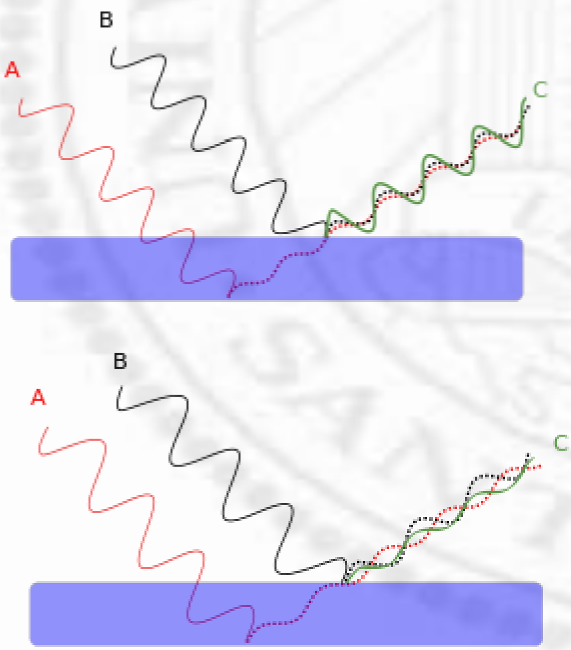


Bidirectional Reflectance
Distribution Functions: Causes
Wolfgang Lucht, 1997
Mirror BRDF:
specular reflectance
Rough water surface BRDF:
sunglint reflectance
Volume scattering BRDF:
leaf/vegetation reflectance
Gap-driven BRDF (Forest):
shadow-driven reflectance

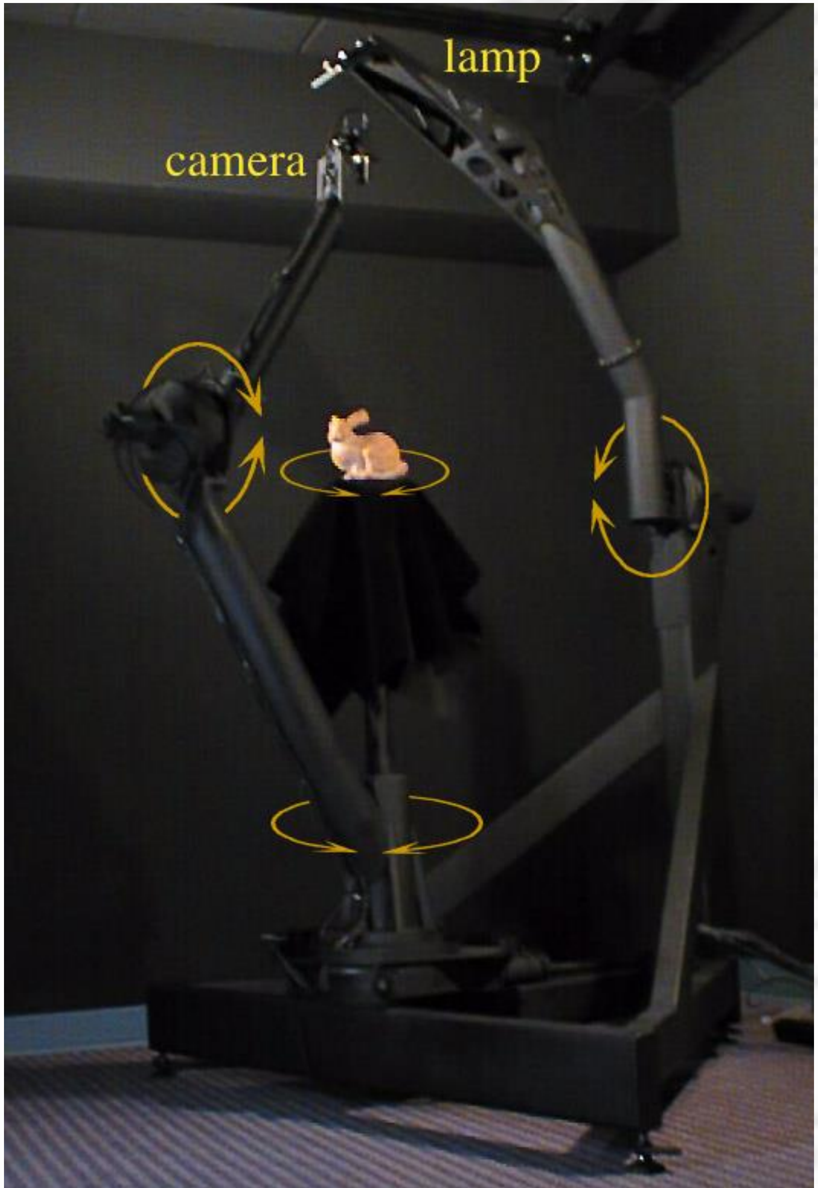


Left: Forward: sun behind observer
Right Backward: sun opposite observer

Left: Forward: sun behind observer
Right Backward: sun opposite observer



- Thin-film interference: thin layer of oil floating on water



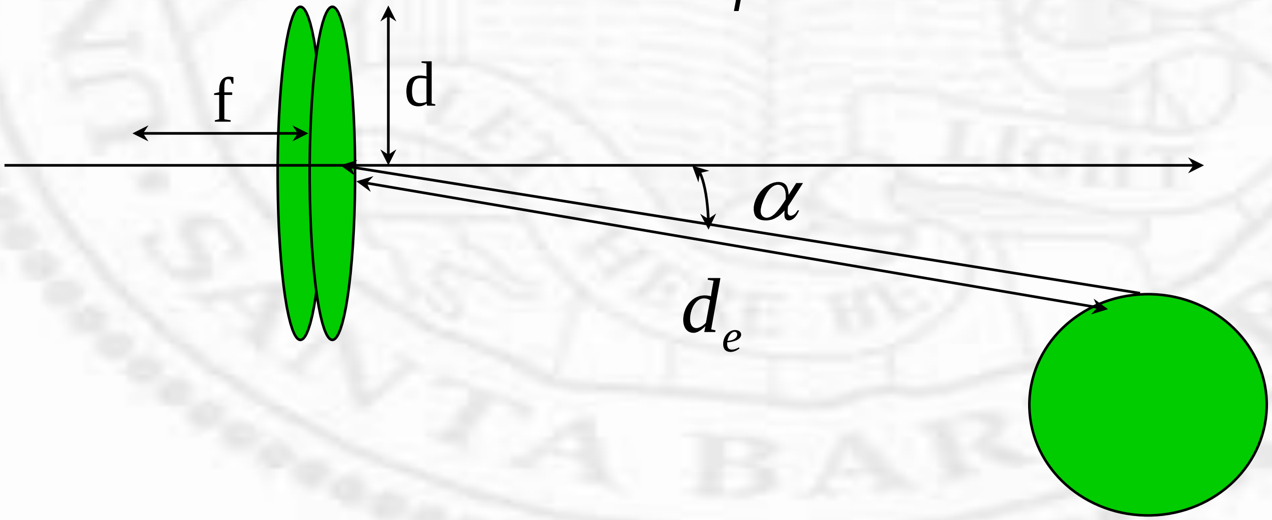
- Stanford gantry: automated setup to study BDRF

Light reaching the viewer

- How much actually being detected?
- Attenuated by distance
- Attenuated by the lens mechanism

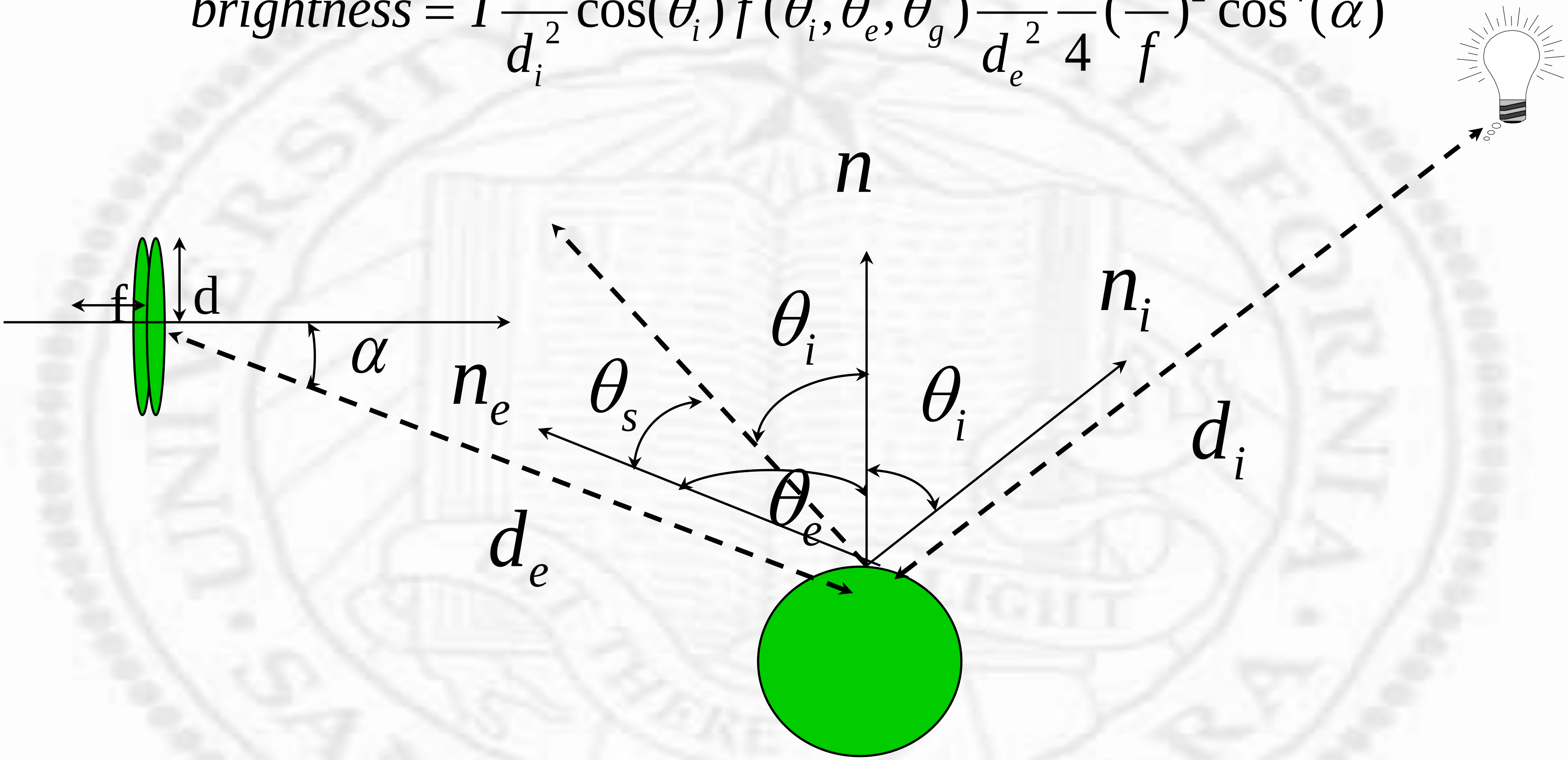
$$\text{scene irradiance} = \frac{1}{d_e^{\,2}} \text{object irradiance}$$

$$\text{image irradiance} = \left(\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha\right) \text{scene irradiance}$$



Putting It All Together

$$brightness = I\frac{1}{d_i^{\,2}}\cos(\theta_i)\,f(\theta_i,\theta_e,\theta_g)\frac{1}{d_e^{\,2}}\frac{\pi}{4}\left(\frac{d}{f}\right)^2\cos^4(\alpha)$$



Caveat: only the *primary* ray is considered here!

Variations

- Distance attenuation
  - square drop-off too drastic
  - adding the ambient light term
  - changing the square drop-off term $\max(\frac{1}{c_1 + c_2 d + c_3 d^2}, 1)$
- Lens model difficult to ascertain
  - use a constant term to absorb it
- Color instead of gray scale
  - three equations instead of one

Popular Models

- Diffuse models

$$Directional:$$

$$I = I_d k_d \cos(\theta_i) = I_d k_d (n_i \cdot n)$$

$$I = I_d k_d \cos(\theta_i) + I_a k_a = I_d k_d (n_i \cdot n) + I_a k_a$$

$$Positional:$$

$$I = \frac{I_d k_d \cos(\theta_i)}{\max(c_1 + c_2 d + c_3 d^2, 1)} + I_a k_a = \frac{I_d k_d (n_i \cdot n)}{\max(c_1 + c_2 d + c_3 d^2, 1)} + I_a k_a$$

- Specular models

$$Directional:$$

$$I = I_s k_s \cos^n(\theta_s) \cos(\theta_i)$$

$$I = I_s k_s \cos^n(\theta_s) \cos(\theta_i) + I_a k_a$$

$$Positional:$$

$$I = \frac{I_s k_s \cos^n(\theta_s) \cos(\theta_i)}{\max(c_1 + c_2 d + c_3 d^2, 1)} + I_a k_a$$

Popular Models (cont.)

- Combined models

$$I = I_d \cos(\theta_i)(\alpha k_d + \beta k_s \cos^n(\theta_s))$$

$$I = I_d \cos(\theta_i)(\alpha k_d + \beta k_s \cos^n(\theta_s)) + I_a k_a$$

$$I = \frac{I_d \cos(\theta_i)(\alpha k_d + \beta k_s \cos^n(\theta_s))}{\max(c_1 + c_2 d + c_3 d^2, 1)} + I_a k_a$$

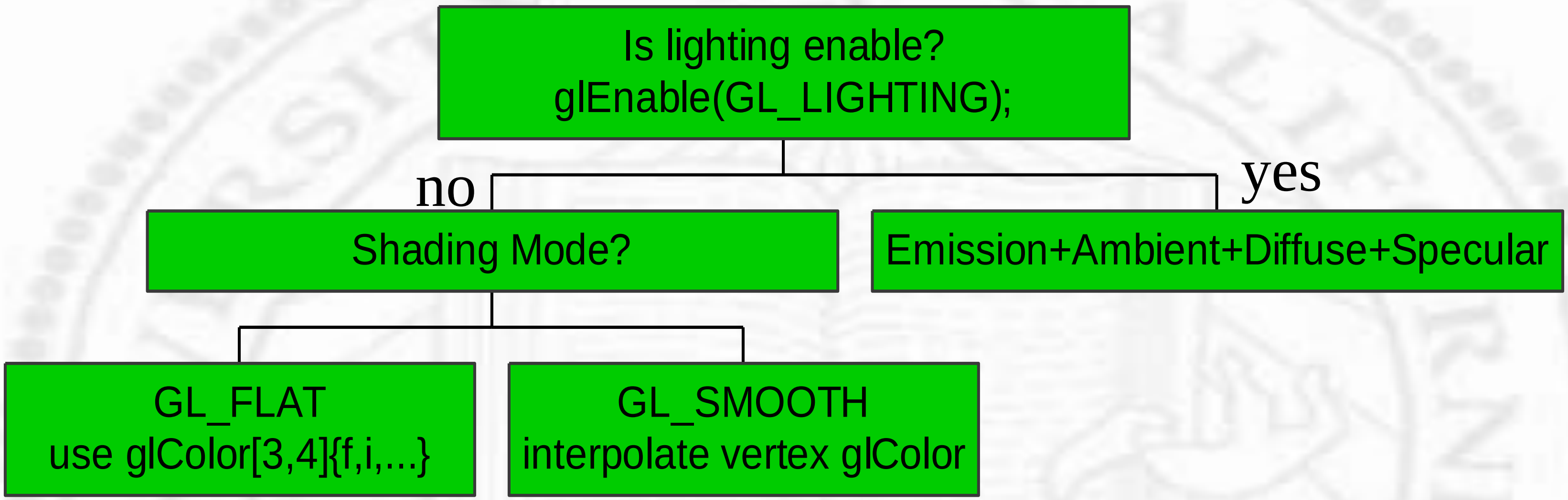
- Color models

$$I_{\{r,g,b\}} = I_{d\{r,g,b\}} \cos(\theta_i)(k_{d\{r,g,b\}} + k_{s\{r,g,b\}} \cos^n(\theta_s))$$

$$I_{\{r,g,b\}} = I_{d\{r,g,b\}} \cos(\theta_i)(k_{d\{r,g,b\}} + k_{s\{r,g,b\}} \cos^n(\theta_s)) + I_{a\{r,g,b\}} k_{a\{r,g,b\}}$$

$$I_{\{r,g,b\}} = \frac{I_{d\{r,g,b\}} \cos(\theta_i)(k_{d\{r,g,b\}} + k_{s\{r,g,b\}} \cos^n(\theta_s))}{\max(c_1 + c_2 d + c_3 d^2, 1)} + I_{a\{r,g,b\}} k_{a\{r,g,b\}}$$

OpenGL Lighting



OpenGL Lighting

- Red, blue, green channels
- Emitted, ambient, diffuse, specular transports
- Ambient, diffuse, specular material properties in red, green and blue channels

$$red = r_{emitted} + r_{ambient} . r_{material\_ambient} + f(r_{diffuse}, r_{material\_diffuse})$$
$$+ g(r_{specular}, r_{material\_specular})$$

Lights

- Void glLight{if}[v](light, pname, param)
  - light: GL_LIGHT0, …, GL_LIGHT7
  - pname: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION, GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION
- Affect later primitives

glLight param

- GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR -> (0.0, 0.0, 0.0, 1.0)
- GL_POSITION -> (0,0,1,0) (directional)
- GL_DIRECTION -> (0,0,-1,0)
- GL_SPOT_EXPONENT -> 0 (uniform)
- GL_SPOT_CUTOFF -> 180 (uniform)
- Etc.

Lights

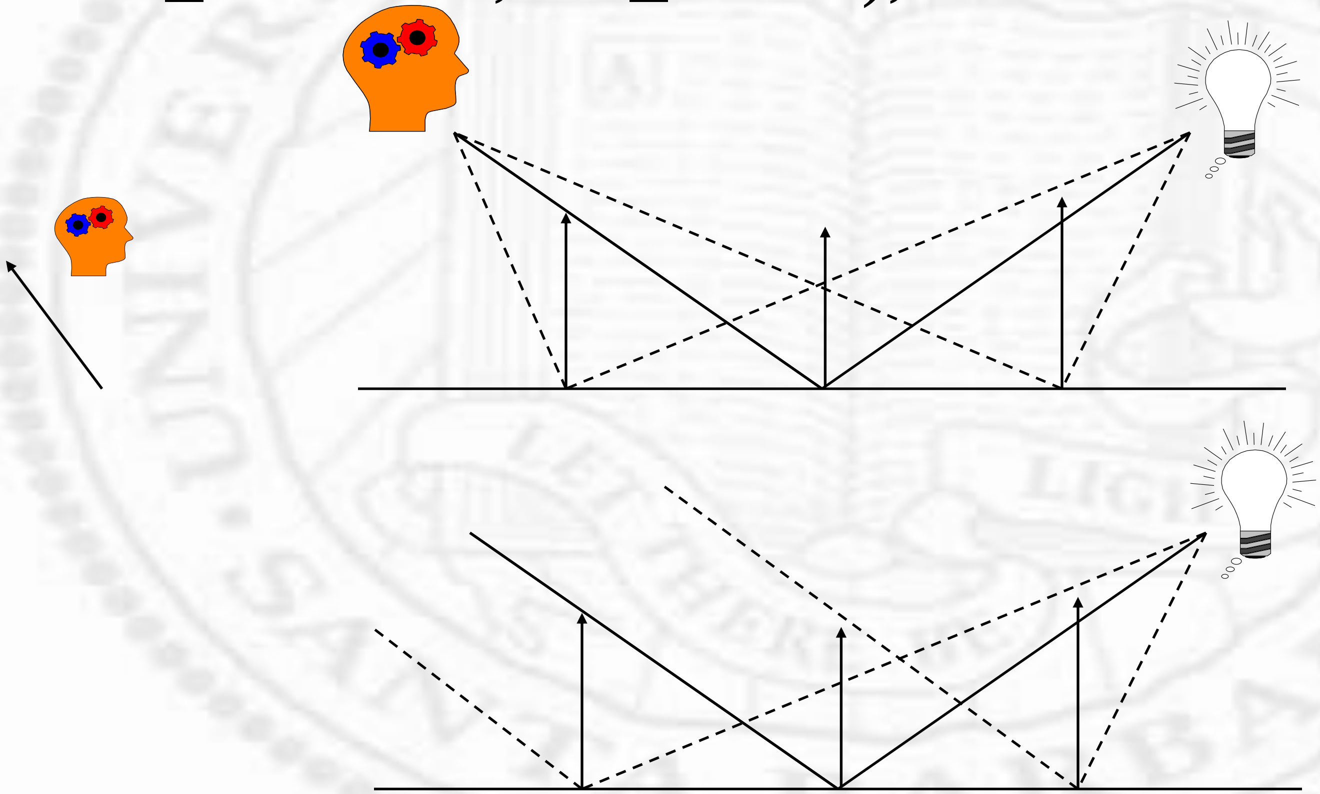
- A light source can add to ambient, diffuse, specular transports in a scene simultaneously
- A directional source (x,y,z,0) at infinity, or
- A positional source (x,y,z,w), radiating energy in all directions
- For positional sources only
  - distance attenuation 1/(kc+kl*d+kq*d^2)
  - spot light effect

Lighting Model

- Global Ambient Light
  - Glfloat global_ambient = {0.2, 0.2, 0.2, 1.0};
  - glLightModelfv(GL_LIGHT_MODEL_AMBIIENT,global_ambient);
- Two-sided Lighting
  - do you need to see back-facing polygon?
  - glLightModeli(LIGHT_MODEL_TWO_SIDE, GL_TRUE);
    - Default is to light only the front
    - If backside is to be lit, the normal is *reverse* and then light

Lighting Model (cont.)

- Local vs. Infinite Viewpoint
  - glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);

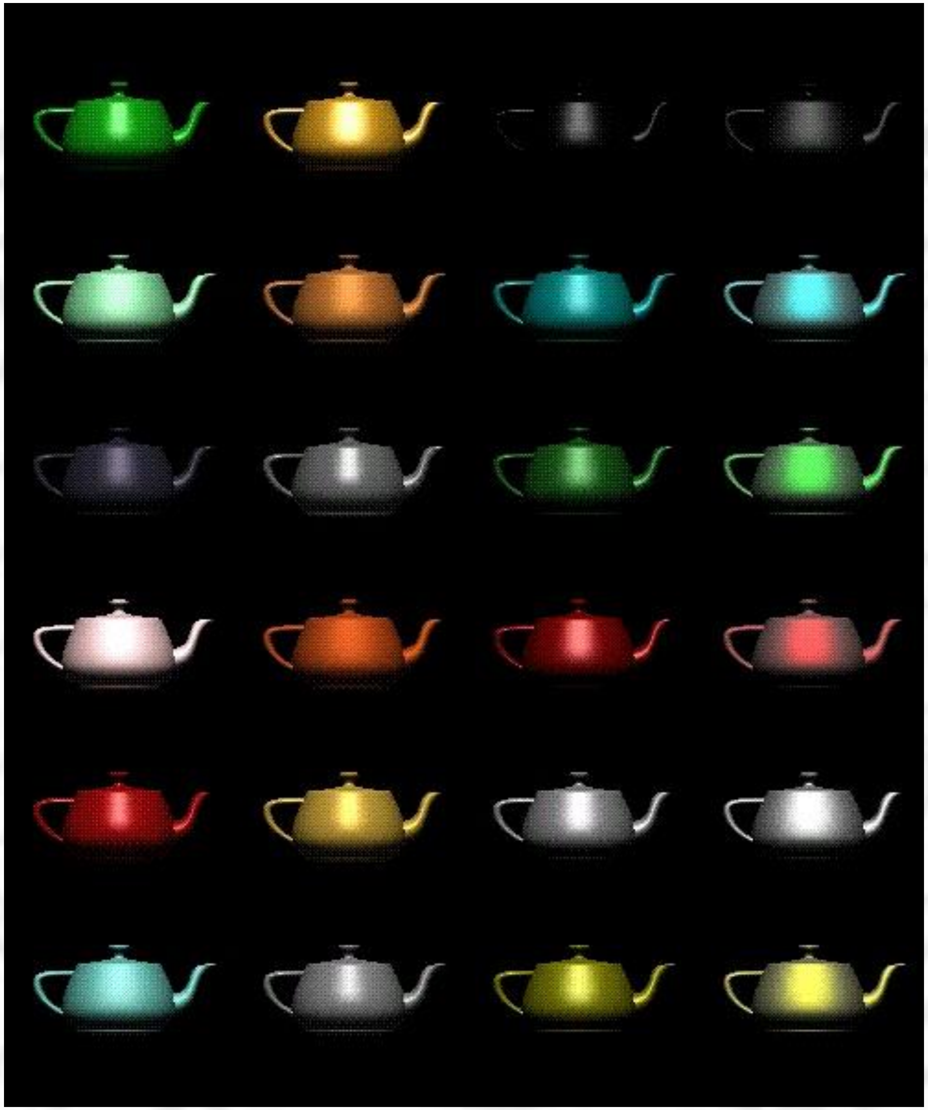


Material

- glMaterial{if}[v](face,pname,param)
  - face: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
  - pname: GL_AMBIENT, GL_DIFFUSE, GL_AMBIENT_AND_DIFFUSE, GL_SPECULAR, GL_SHINESS, GL_EMISSION GL_COLOR_INDEXES
- Affect later primitives

Material param

- GL_AMBIENT -> (0.2, 0.2, 0.2, 1.0)
- GL_DIFFUSE -> (0.8, 0.8, 0.8, 1.0)
- GL_SPECULAR -> (0.0, 0.0, 0.0, 1.0)
- GL_EMISSION -> (0.0, 0.0, 0.0, 1.0)
- GL_SHININESS -> 0



Non-Light-Source part

- Emission component
  - object is a light source
  - Glfloat emision[] = {0.3, 0.2, 0.2, 0.0};
  - glMaterialfv(GL_FRONT, GL_EMISSION, emission)
- Global Ambient Light
  - if present, scaled by the material ambient component

$$ambient_{light\_model} \cdot ambient_{material}$$

Light Source Part

- For each light source
  - contribution = attenuation * spot effect * (ambient + diffuse + specular)
- Attenuation

$$\begin{cases} \dfrac{1}{k_c + k_l d + k_q d^2} & positional \\ 1 & directional \end{cases}$$

Spotlight Effect

$$\begin{cases} 1 & GL\_SPOT\_CUTOFF = 180 \\ 0 & \text{spot light but vertex is out of illumination cone} \\ \max(v \cdot d, 0)^{GL\_SPOT\_EXPONENT} & \begin{array}{l} v: \text{unit vector from spotlight to vertex} \\ \text{d}: \text{spotlight's direction} \end{array} \end{cases}$$

Ambient light

$$ambient_{light} \cdot ambient_{material}$$

Diffuse + Specular

- Diffuse term

$$(\max(l \cdot n, 0)) \cdot diffuse_{light} \cdot diffuse_{material}$$

l : unit vector from vertex to light source

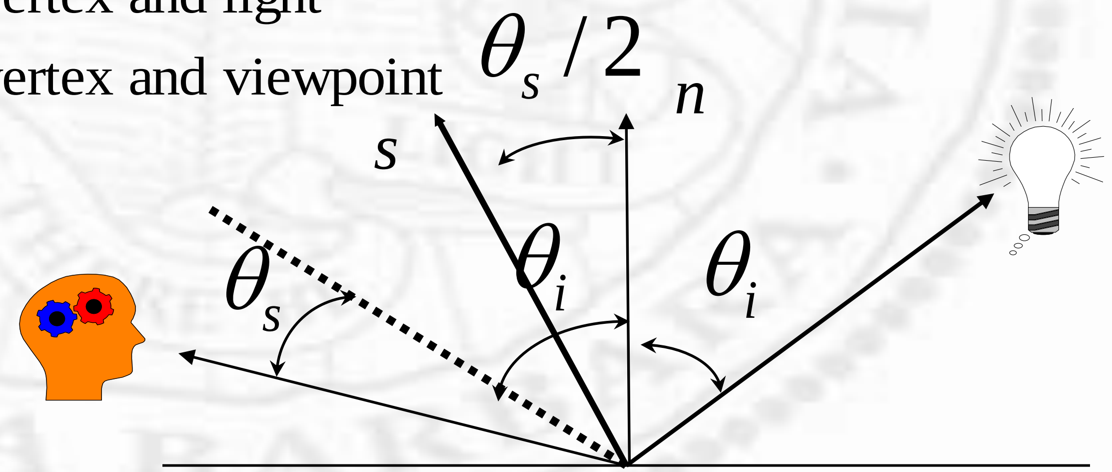
n : surface normal

- Specular term

$$(\max(s \cdot n, 0))^{shininess} \cdot specular_{light} \cdot specular_{material}$$

s : unit vector in between $\begin{pmatrix} \text{vertex and light} \\ \text{vertex and viewpoint} \end{pmatrix}$

n : surface normal



Transformation

- A light source has a number of attributes which are sensitive to transformation
  - position
  - orientation
  - glLightfv(GL_LIGHT0,GL_POSITION,….);
- These attributed are subject to GL_MODELVIEW transform
  - or think them as vertex

Transformation

- Lights should appear close to the top of the transform code

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity
  glLightfv
  …

Transformation

- Globally-Fixed Light Source
  - not affected by object & view transform

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity
glLightfv
…
gluLookAt(…)
glTranslate, glRotate, glScale, etc.
glBegin, glEnd
```

Transformation

- Locally-Fixed Light Source
  - move with the viewer (say, always at the eye location)

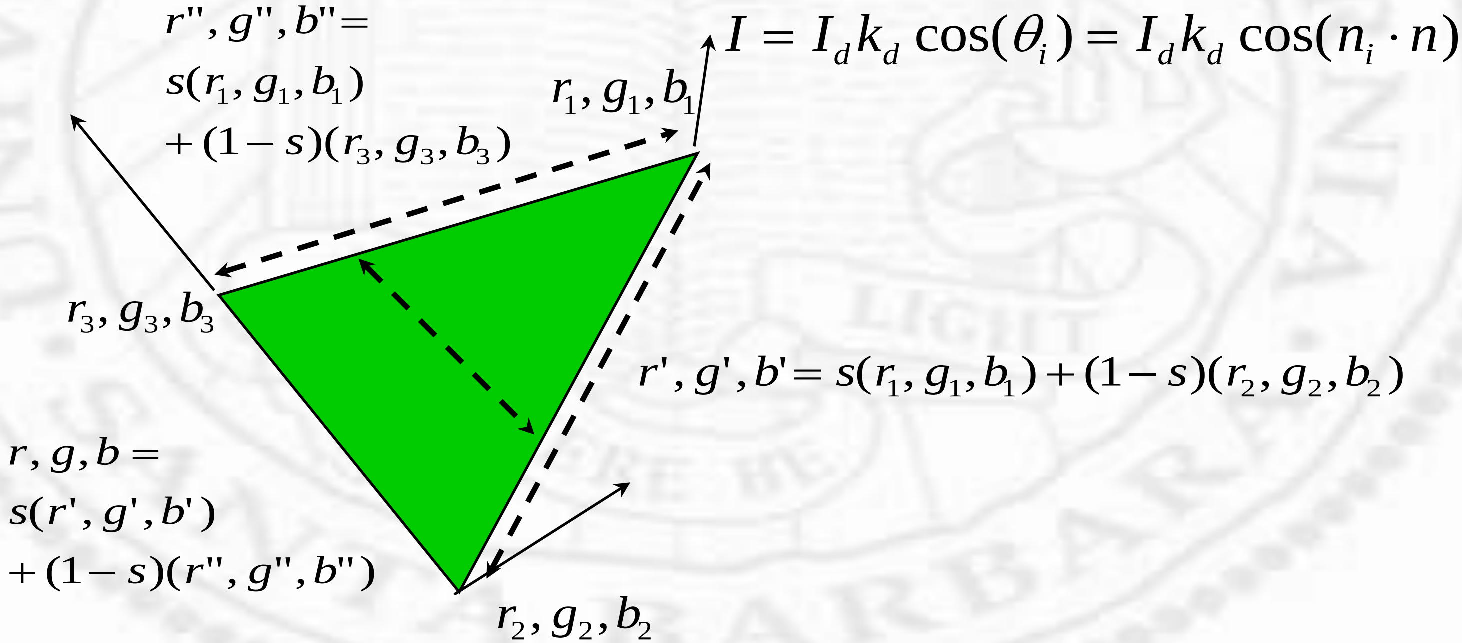
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity
gluLookAt(…)
glPushMatrix()
    glT, glR, glS, glLightfv
glPopMatrix()
glTranslate, glRotate, glScale, etc.
glBegin, glEnd
```

Polygon Shading

- So far, consider only shading *individual points*
- Need to shade a smooth surface
- Often times, a smooth surface is approximated by polygon patches
- Need to shade polygonal approximation without giving out polygon identities

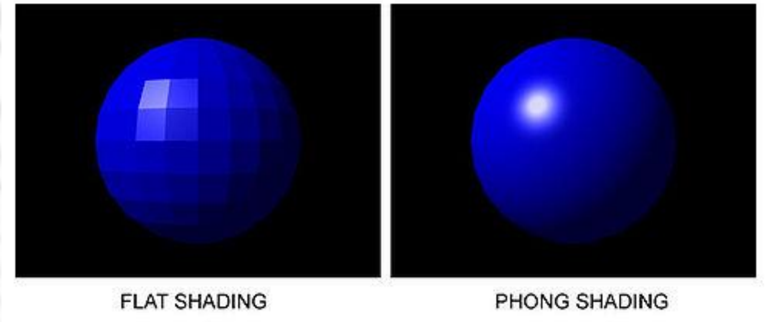
Gouraud Shading

- Interpolative shading
  - Calculate polygon vertex colors
  - Interpolate *colors* for interior points

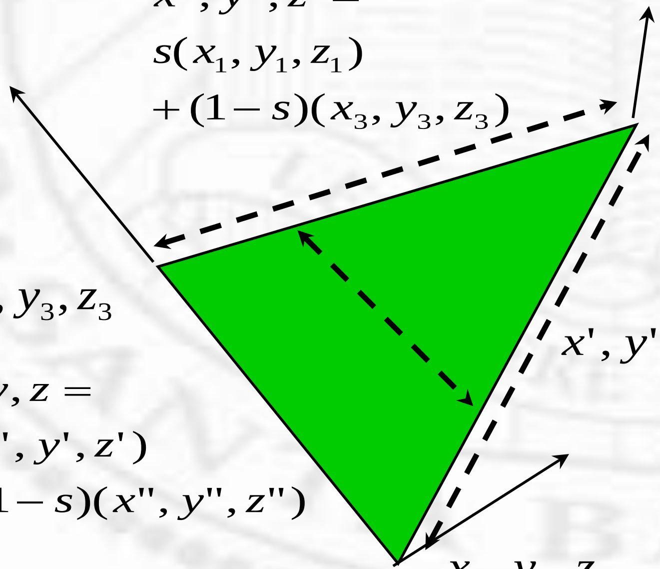


Phong Shading

- Interpolative shading
  - Calculate polygon normals
  - Interpolate *normals* for interior points



$$I = I_d k_d \cos(\theta_i) = I_d k_d \cos(n_i \cdot n)$$



$$x'', y'', z'' = s(x_1, y_1, z_1) + (1-s)(x_3, y_3, z_3)$$

x_1, y_1, z_1

x_3, y_3, z_3

$$x', y', z' = s(x_1, y_1, z_1) + (1-s)(x_2, y_2, z_2)$$

$$x, y, z = s(x', y', z') + (1-s)(x'', y'', z'')$$

x_2, y_2, z_2

Comparison