



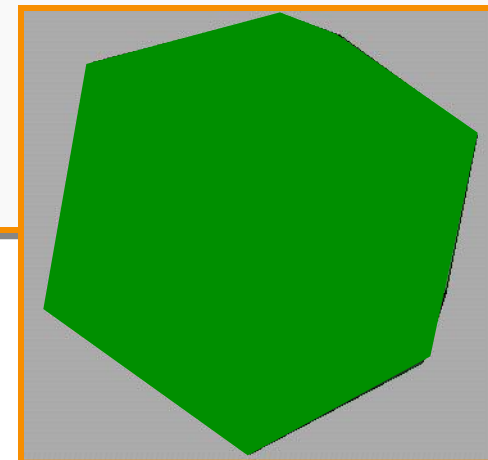
# Lighting and Reflectance

COS 426, Spring 2015  
Princeton University



# Ray Casting

```
R2Image *RayCast(R3Scene *scene, int width, int height)
{
    R2Image *image = new R2Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            R3Ray ray = ConstructRayThroughPixel(scene->camera, i, j);
            R3Rgb radiance = ComputeRadiance(scene, &ray);
            image->SetPixel(i, j, radiance);
        }
    }
    return image;
}
```

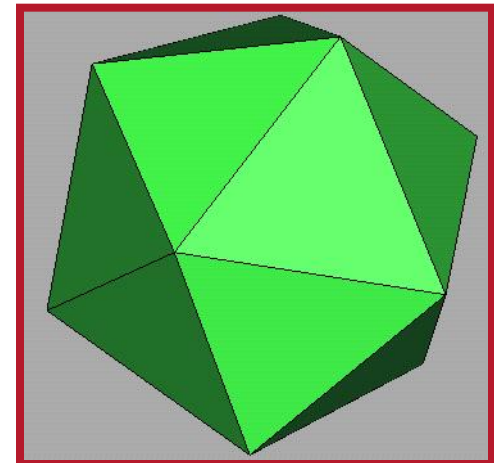


Without Illumination

# Ray Casting



```
R3Rgb ComputeRadiance(R3Scene *scene, R3Ray *ray)
{
    R3Intersection intersection = ComputeIntersection(scene, ray);
    return ComputeRadiance(scene, ray, intersection);
}
```

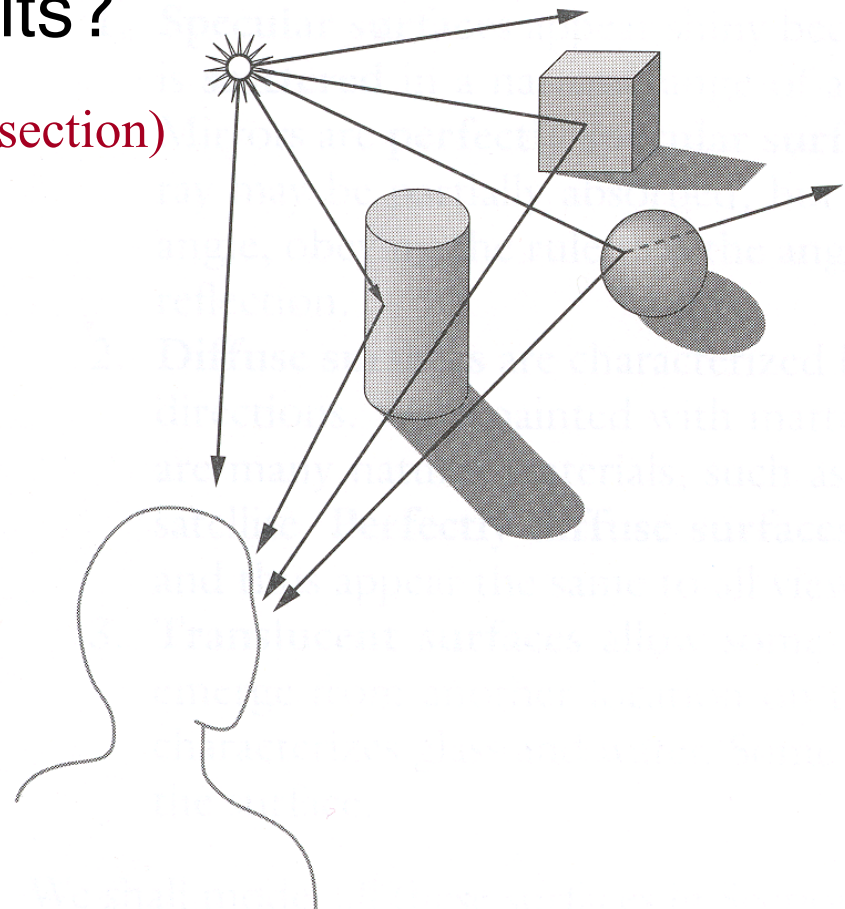


With Illumination

# Illumination

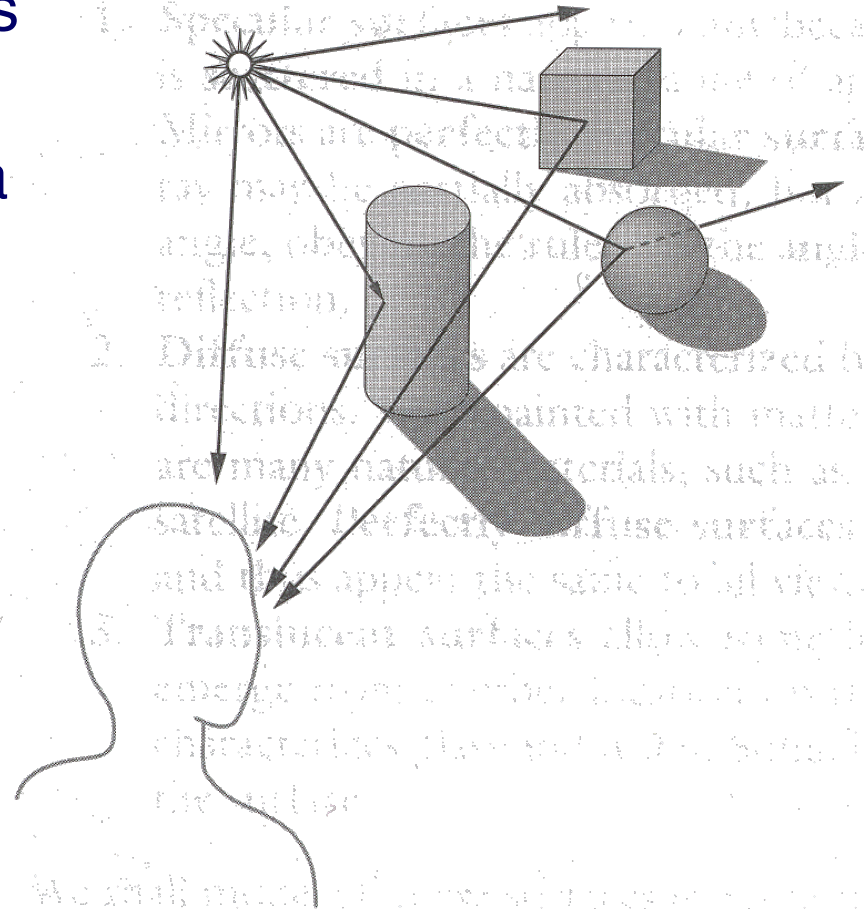
- How do we compute radiance for a sample ray once we know what it hits?

ComputeRadiance(scene, ray, intersection)



# Goal

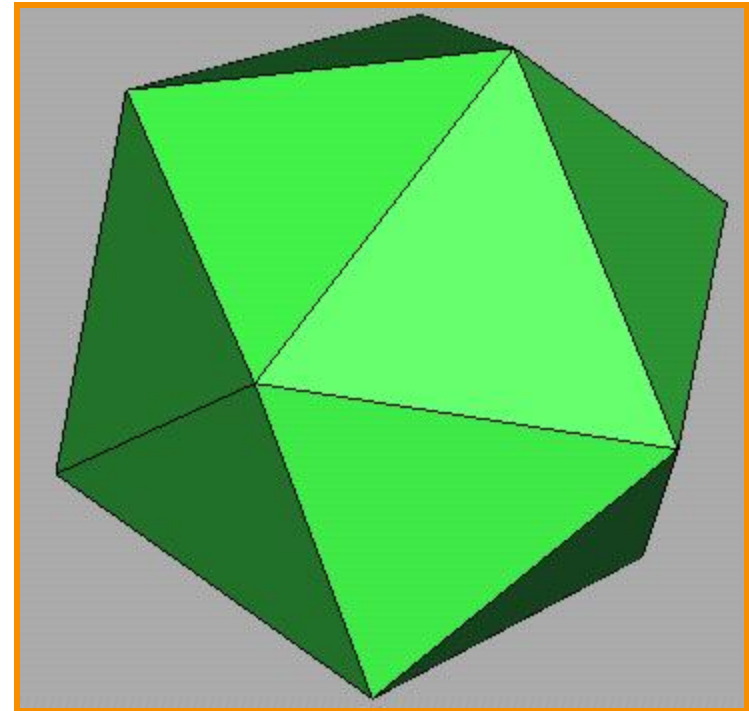
- Must derive computer models for ...
  - Emission at light sources
  - Scattering at surfaces
  - Reception at the camera
- Desirable features ...
  - Concise
  - Efficient to compute
  - “Accurate”



# Overview



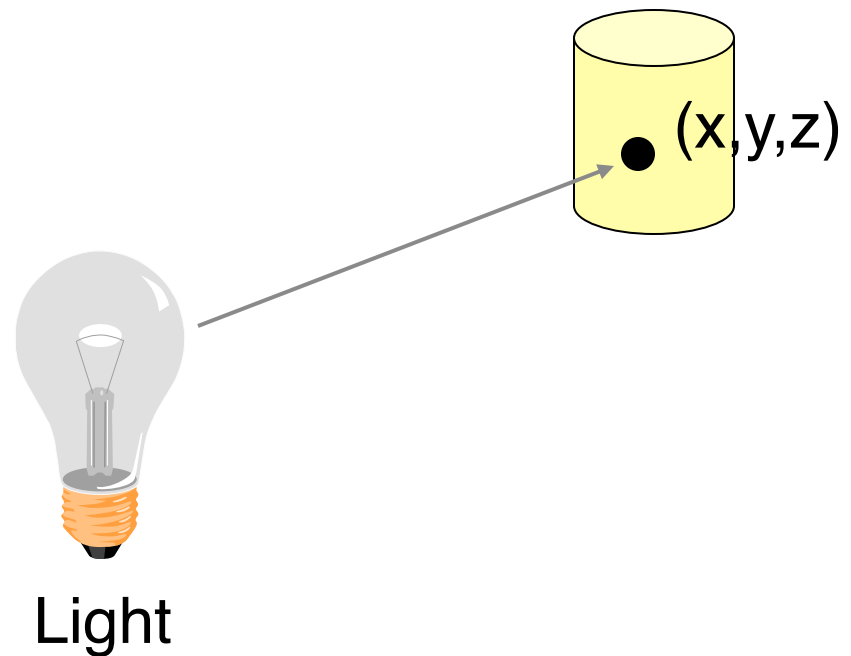
- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Refractions
  - Inter-object reflections



Direct Illumination

# Emission at Light Sources

- $I_L(x, y, z, \theta, \phi, \lambda) \dots$ 
  - describes the intensity of energy,
  - leaving a light source, ...
  - arriving at location  $(x, y, z)$ , ...
  - from direction  $(\theta, \phi)$ , ...
  - with wavelength  $\lambda$



# Empirical Models

- Ideally measure irradiant energy for “all” situations
  - Too much storage
  - Difficult in practice

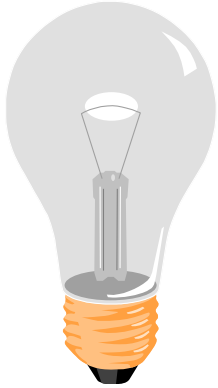




# OpenGL Light Source Models



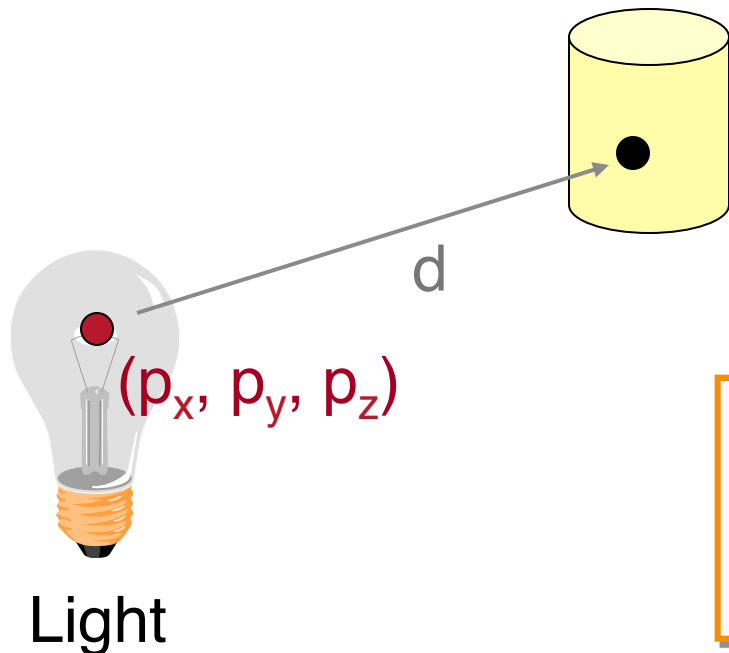
- Simple mathematical models:
  - Point light
  - Directional light
  - Spot light



# Point Light Source



- Models omni-directional point source
  - intensity ( $I_0$ ),
  - position ( $p_x, p_y, p_z$ ),
  - coefficients ( $c_a, l_a, q_a$ ) for attenuation with distance ( $d$ )



$$I_L = \frac{I_0}{c_a + l_a d + q_a d^2}$$

# Point Light Source

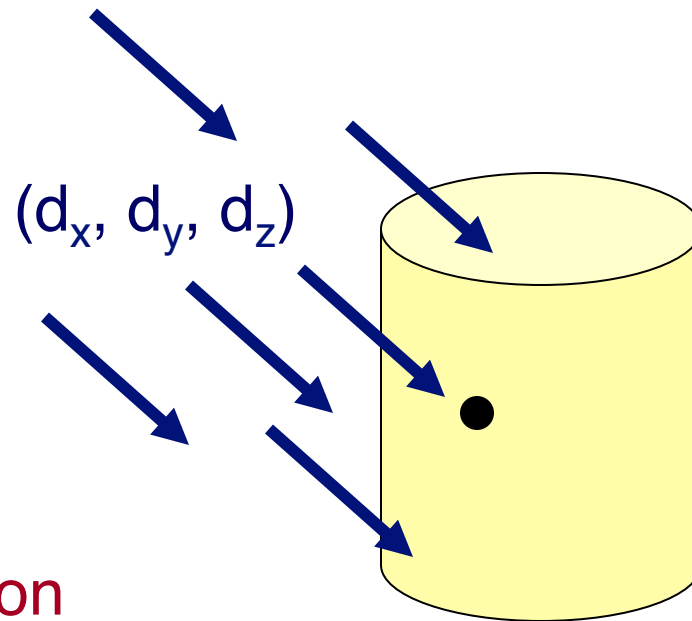


$$I_L = \frac{I_0}{c_a + l_a d + q_a d^2}$$

- Physically-based: “inverse square law”
  - $c_a = l_a = 0$
- Use  $c_a$  and  $l_a \neq 0$  for non-physical effects
  - Better control of the look (artistic)

# Directional Light Source

- Models point light source at infinity
  - intensity ( $I_0$ ),
  - direction ( $d_x, d_y, d_z$ )



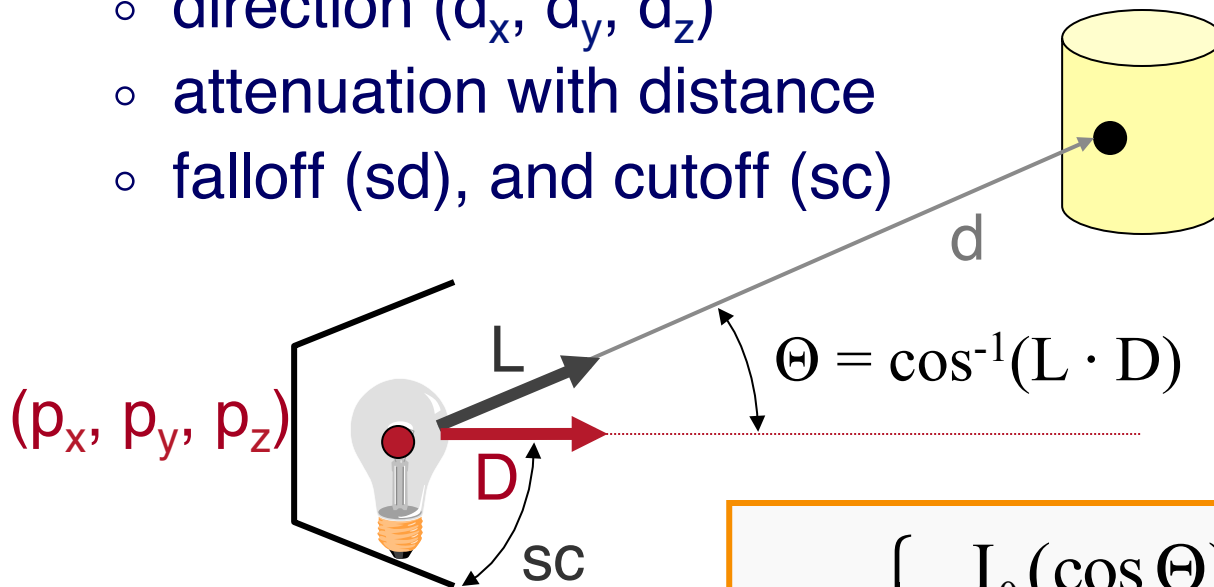
No attenuation  
with distance



$$I_L = I_0$$

# Spot Light Source

- Models point light source with direction
  - intensity ( $I_0$ ),
  - position ( $p_x, p_y, p_z$ ),
  - direction ( $d_x, d_y, d_z$ )
  - attenuation with distance
  - falloff ( $sd$ ), and cutoff ( $sc$ )

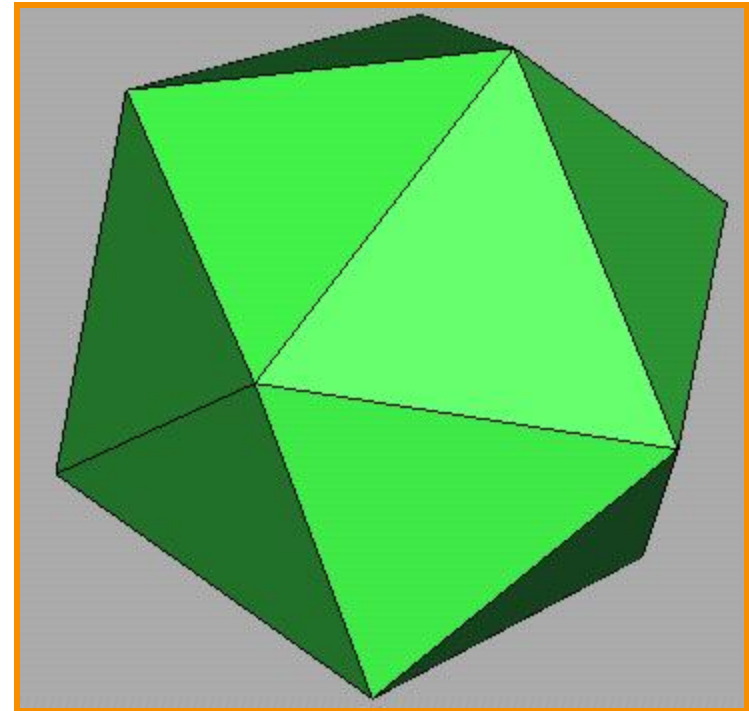


$$I_L = \begin{cases} \frac{I_0 (\cos \Theta)^{sd}}{c_a + l_a d + q_a d^2} & \text{if } \Theta \leq sc, \\ 0 & \text{otherwise} \end{cases}$$

# Overview



- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Refractions
  - Inter-object reflections



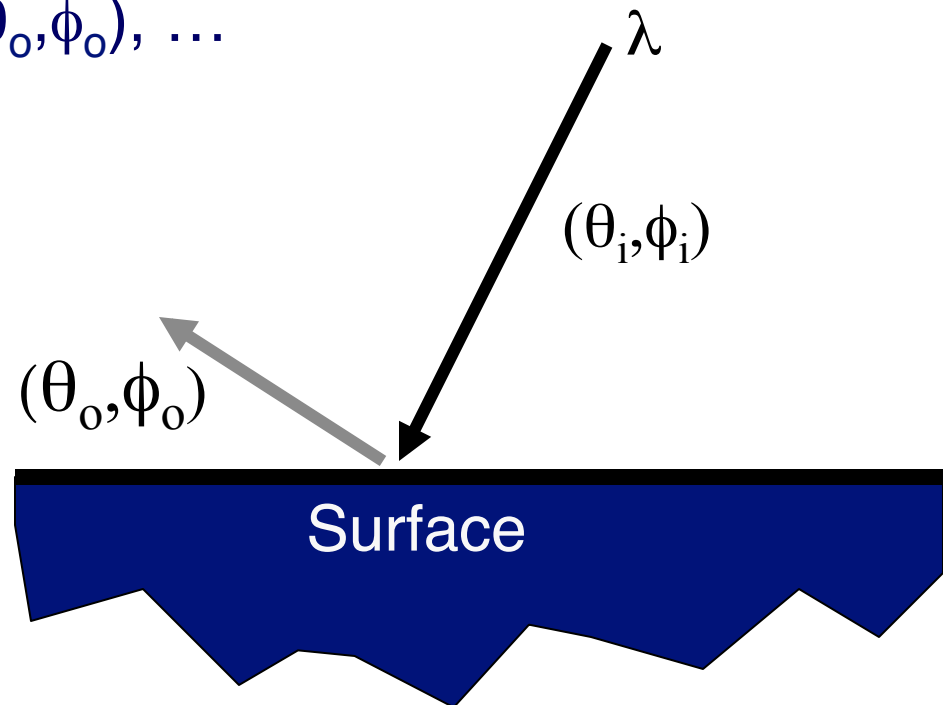
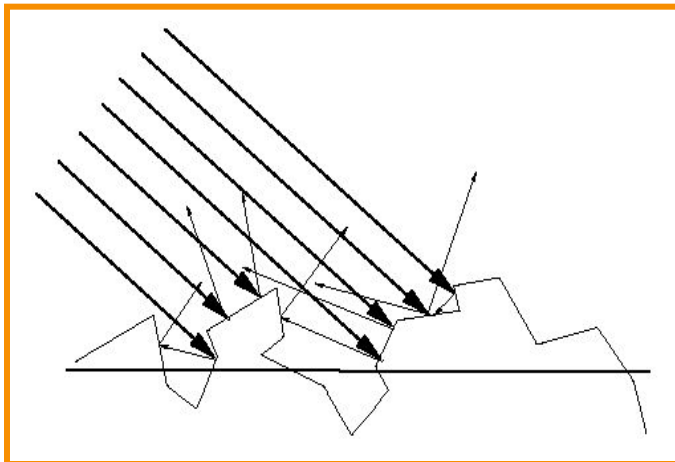
Direct Illumination

# Scattering at Surfaces

## Bidirectional Reflectance Distribution Function

$f_r(\theta_i, \phi_i, \theta_o, \phi_o, \lambda) \dots$

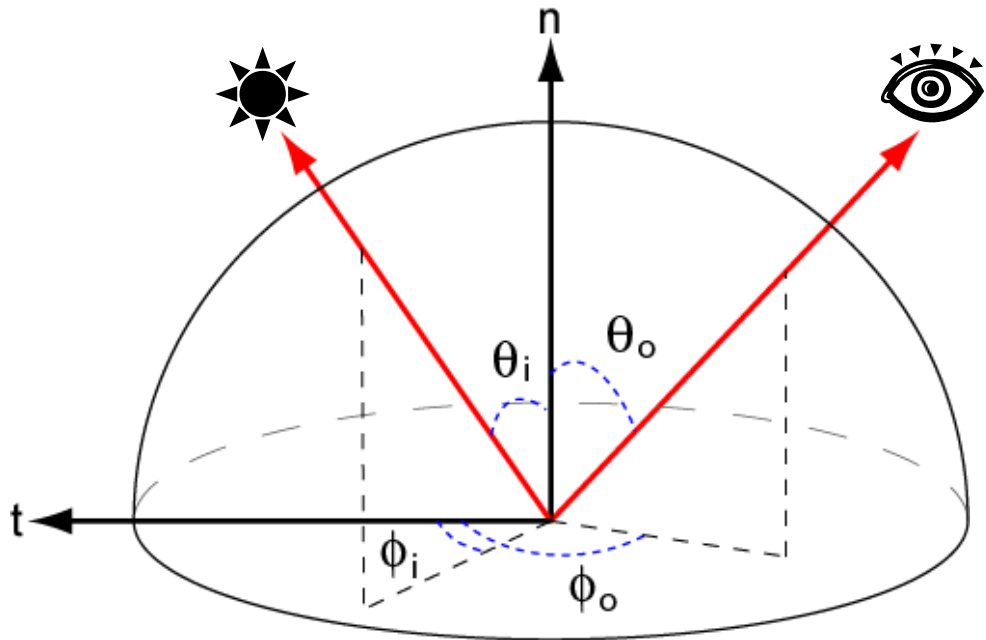
- describes the aggregate fraction of incident energy,
- arriving from direction  $(\theta_i, \phi_i)$ , ...
- leaving in direction  $(\theta_o, \phi_o)$ , ...
- with wavelength  $\lambda$



# Empirical Models

Ideally measure BRDF for “all” combinations of angles:  $\theta_i, \phi_i, \theta_o, \phi_o$

- Difficult in practice
- Too much storage



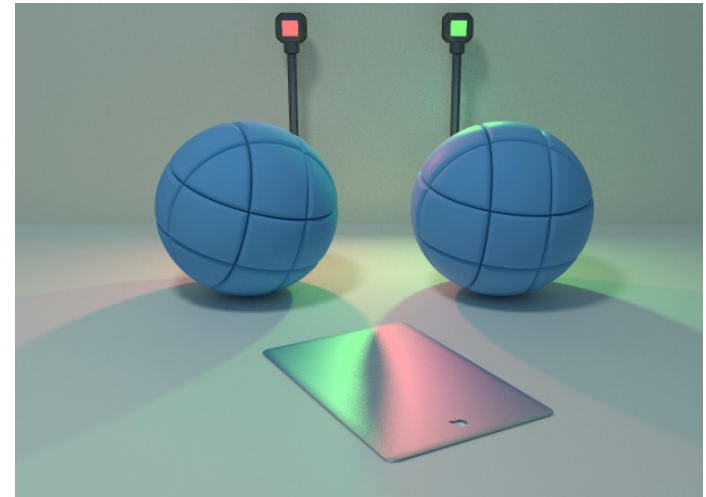


# Parametric Models



Approximate BRDF with simple parametric function that is fast to compute.

- Phong [75]
- Blinn-Phong [77]
- Cook-Torrance [81]
- He et al. [91]
- Ward [92]
- Lafortune et al. [97]
- Ashikhmin et al. [00]
- etc.



Lafortune [97]



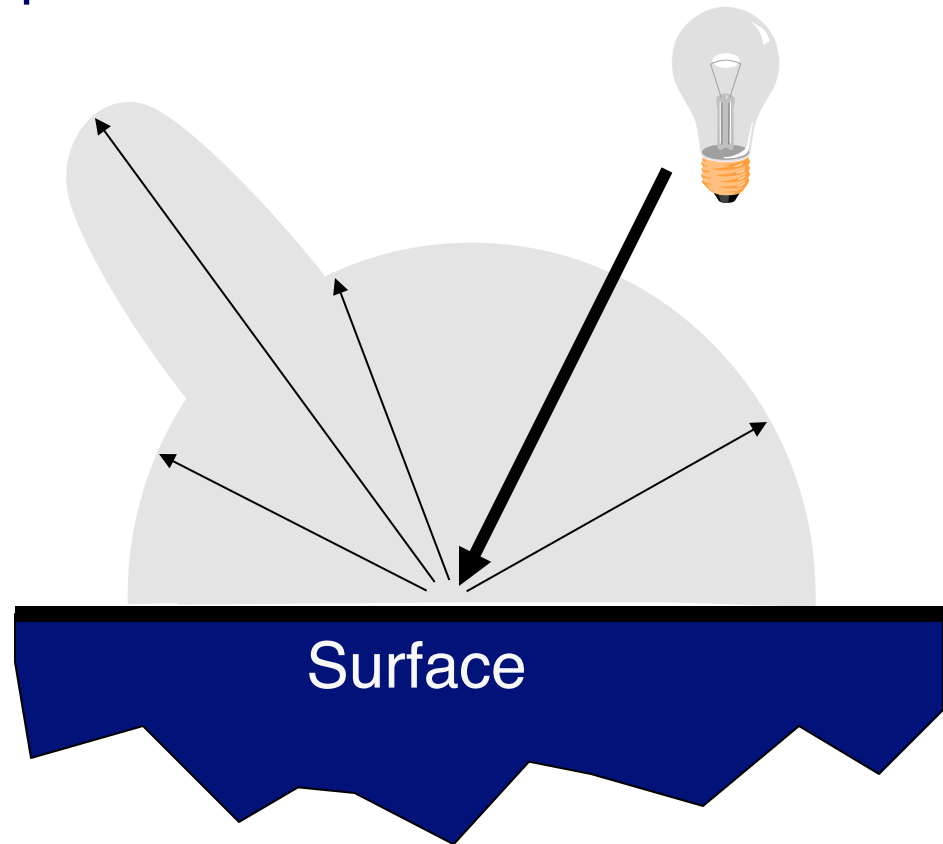
Cook-Torrance [81]

# OpenGL Reflectance Model



- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”

Based on model  
proposed by Phong

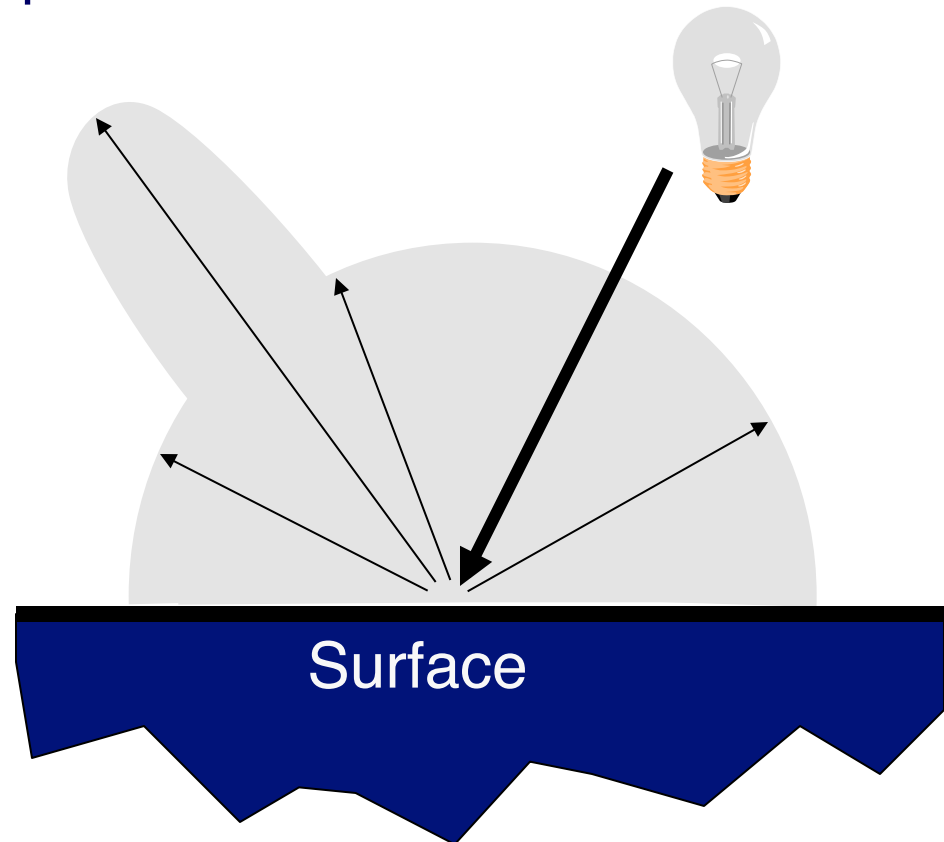


# OpenGL Reflectance Model



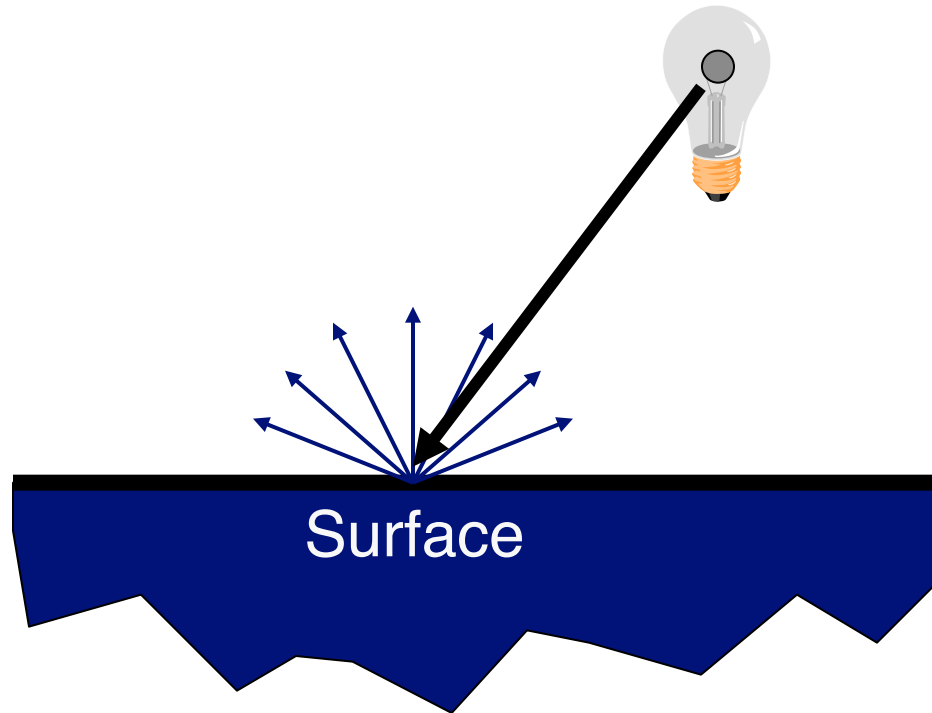
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”

Based on model  
proposed by Phong



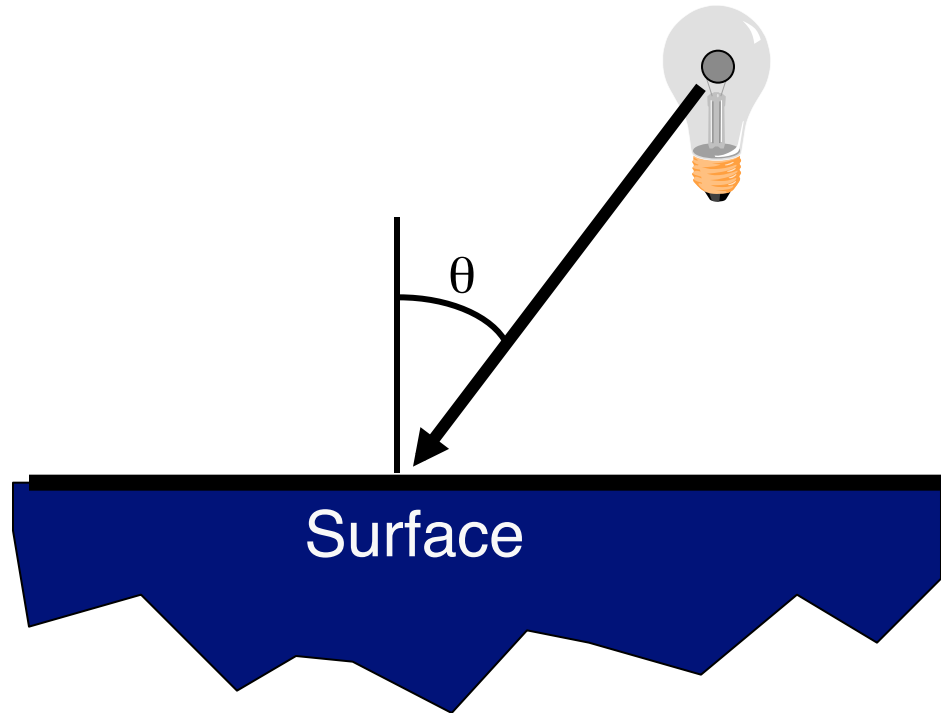
# Diffuse Reflection

- Assume surface reflects equally in all directions
  - Examples: chalk, clay



# Diffuse Reflection

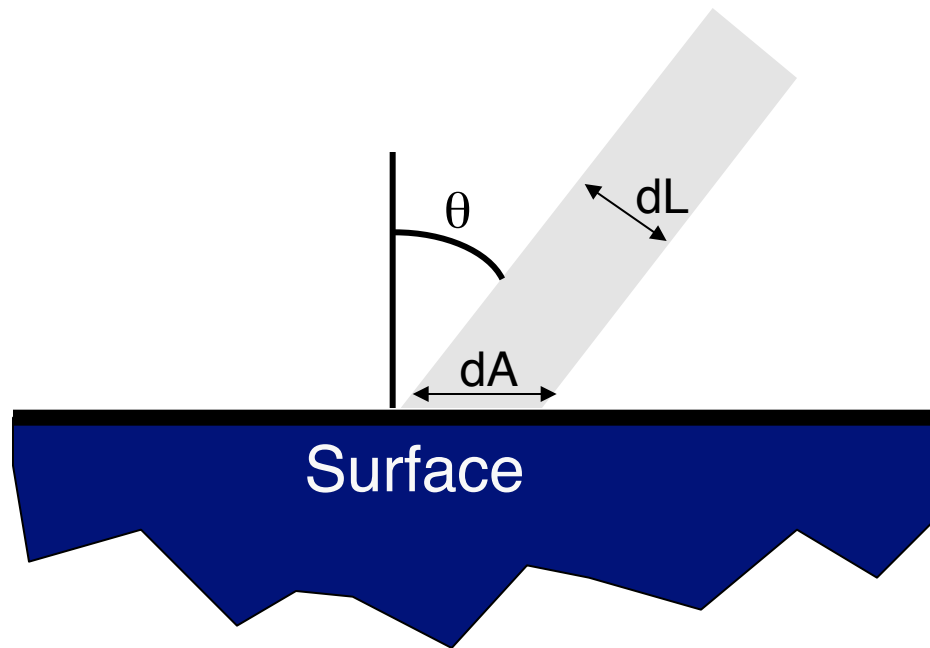
- What is brightness of surface?
  - Depends on angle of incident light



# Diffuse Reflection

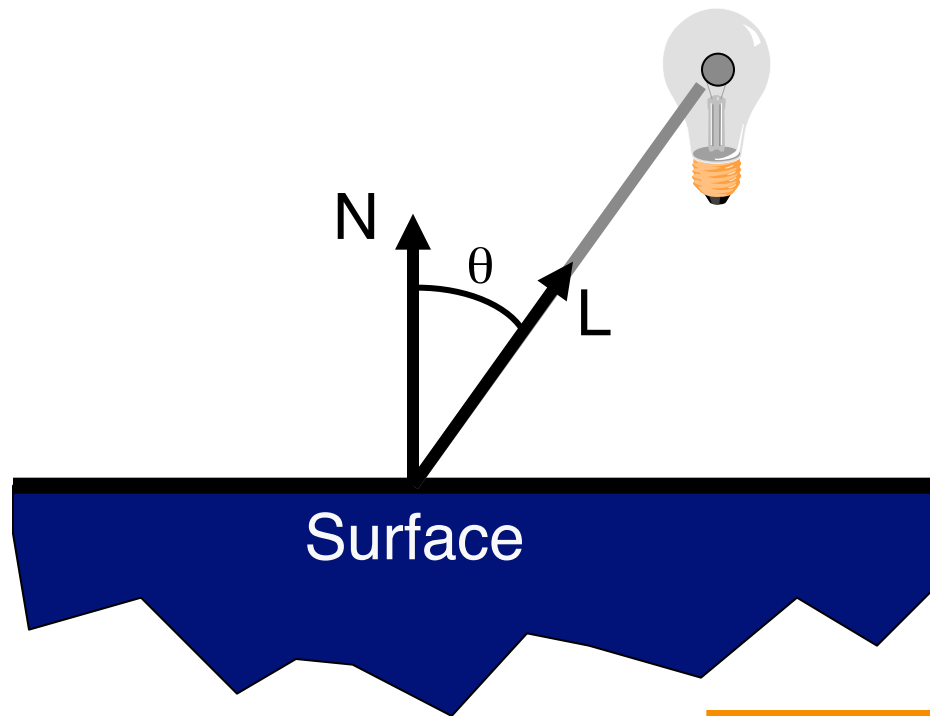
- What is brightness of surface?
  - Depends on angle of incident light

$$dL = dA \cos \Theta$$



# Diffuse Reflection

- Lambertian model
  - cosine law (dot product)

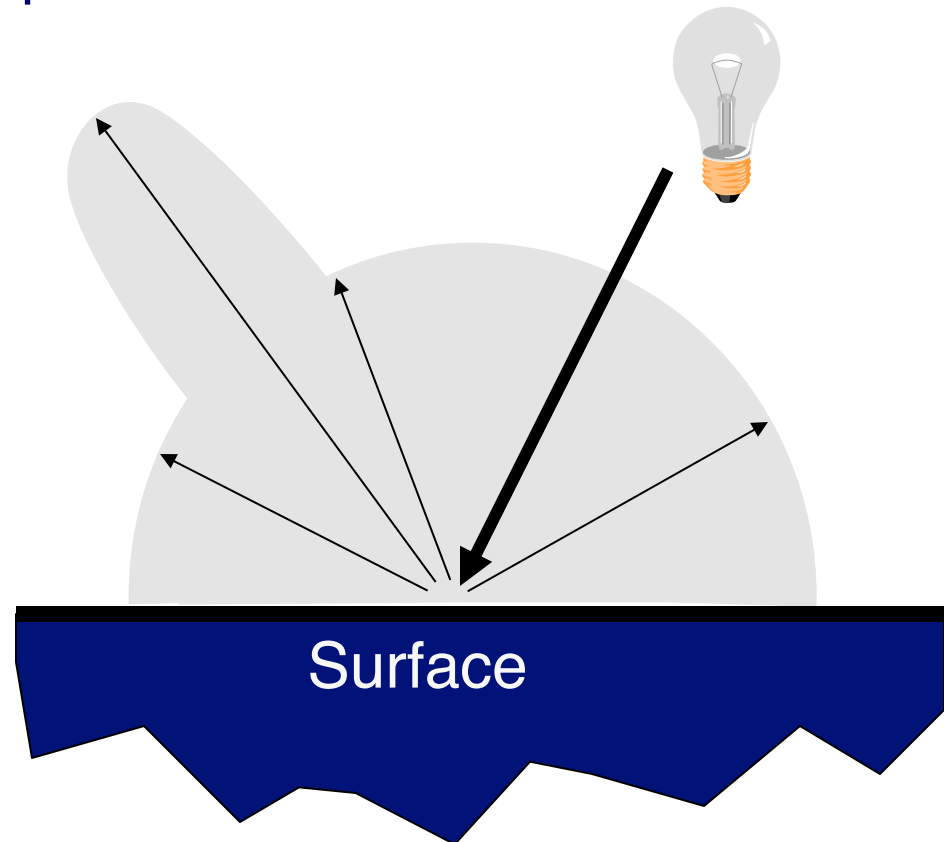


$$I_D = K_D (N \cdot L) I_L$$

# OpenGL Reflectance Model



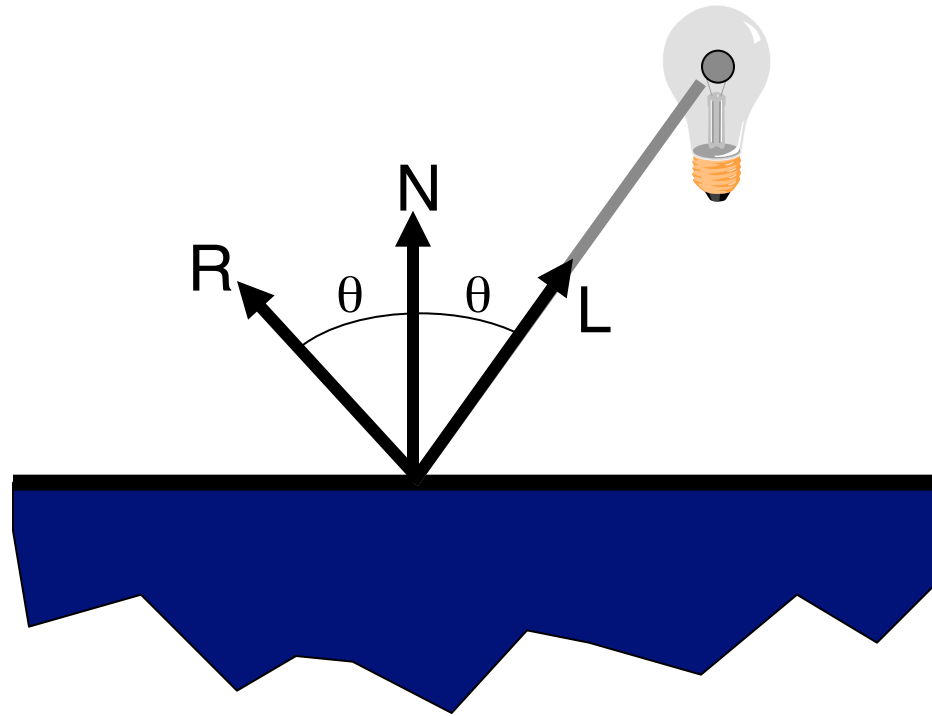
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”





# Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: mirrors, metals

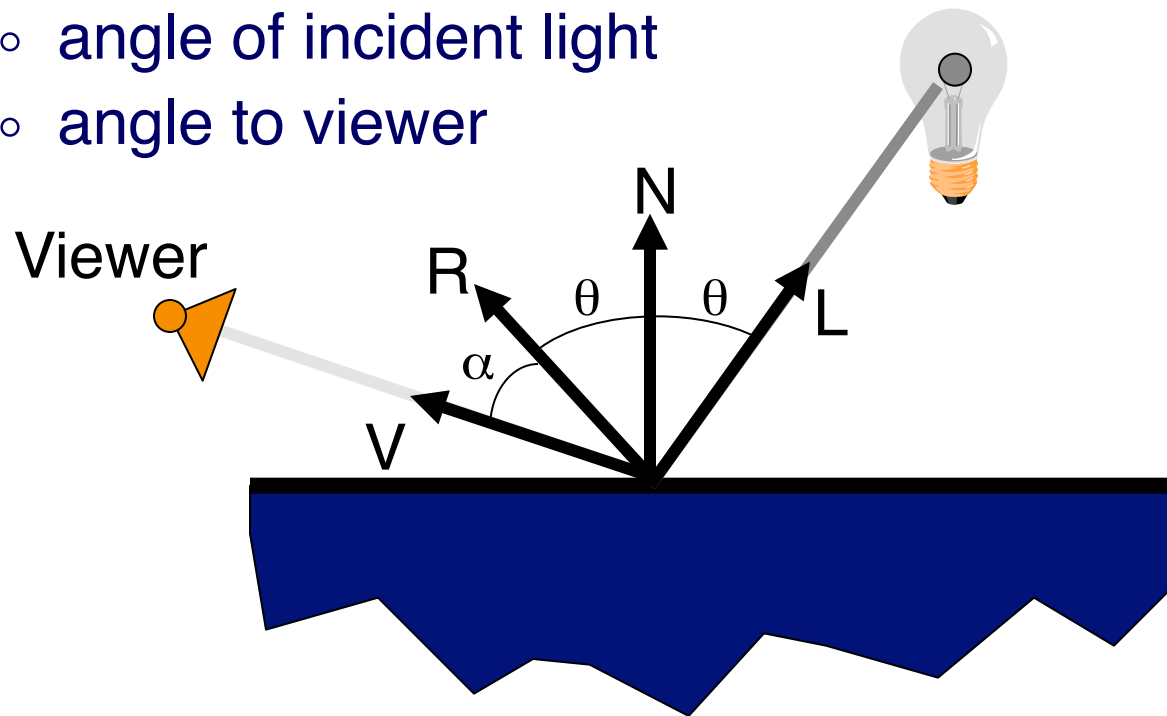


# Specular Reflection

How much light is seen?

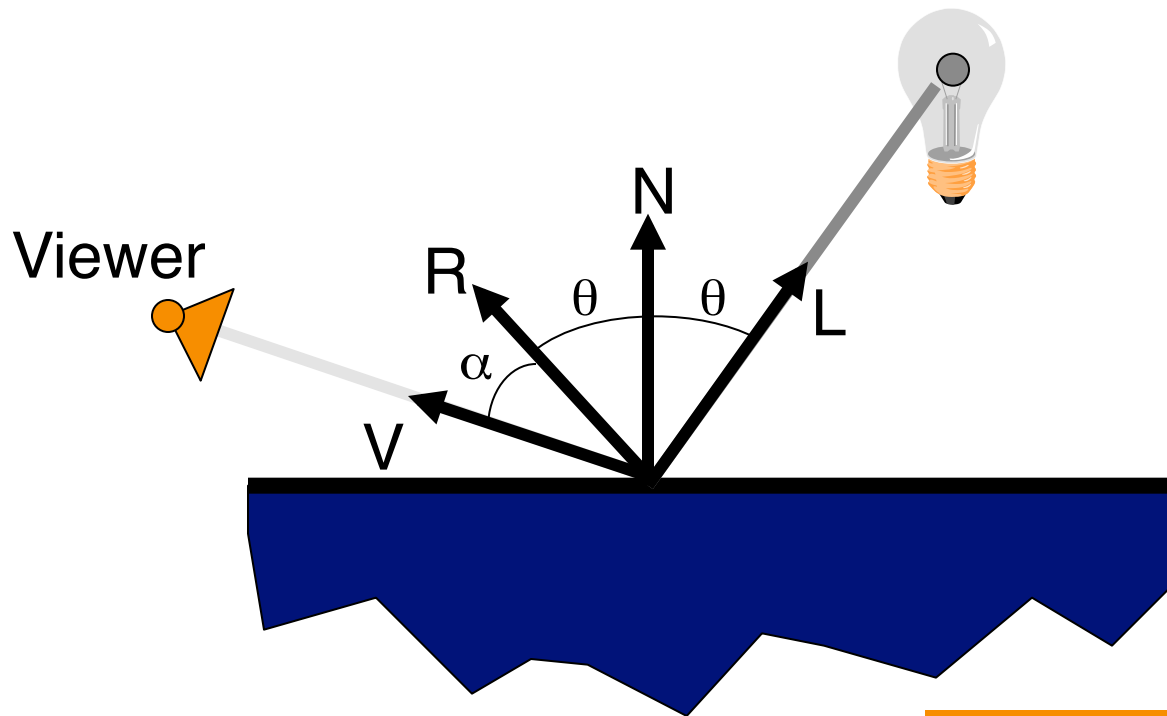
Depends on:

- angle of incident light
- angle to viewer



# Specular Reflection

- Phong Model
  - $(\cos \alpha)^n$  This is a (vaguely physically-motivated) hack!

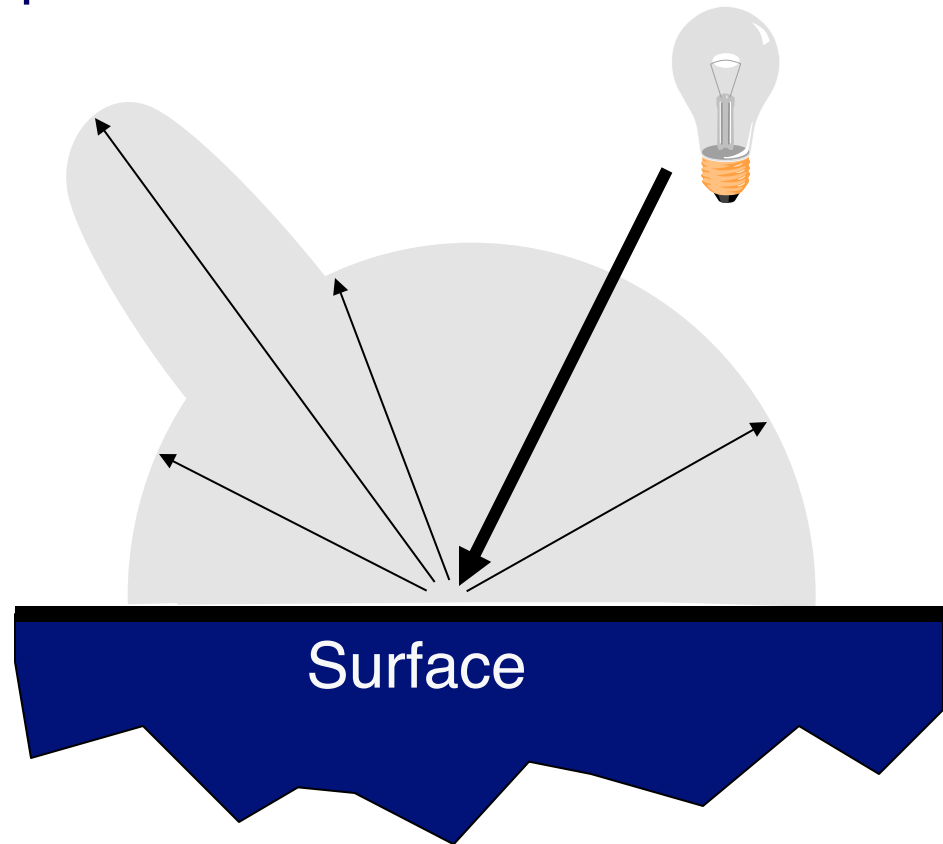


$$I_S = K_S (V \cdot R)^n I_L$$

# OpenGL Reflectance Model



- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - **emission** +
  - “ambient”



# Emission



Represents light emanating directly from surface

- Note: does not automatically act as light source!  
Does not affect other surfaces in scene!

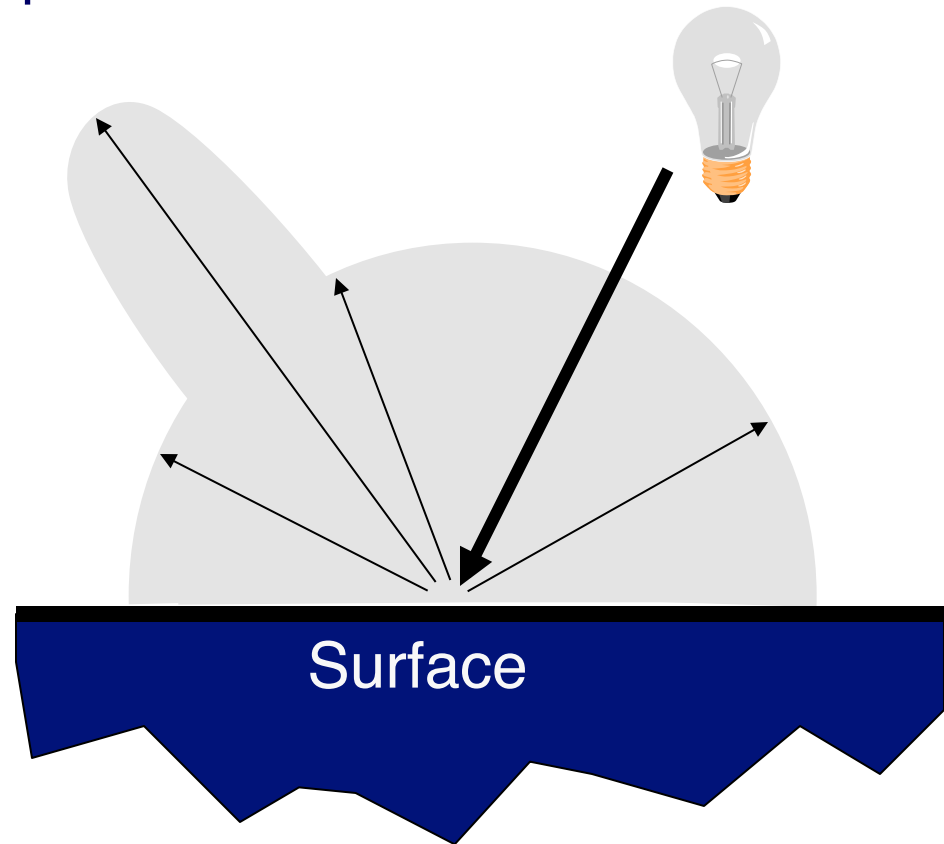


Emission  $\neq$  0

# OpenGL Reflectance Model



- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



# Ambient Term



Represents reflection of all indirect illumination

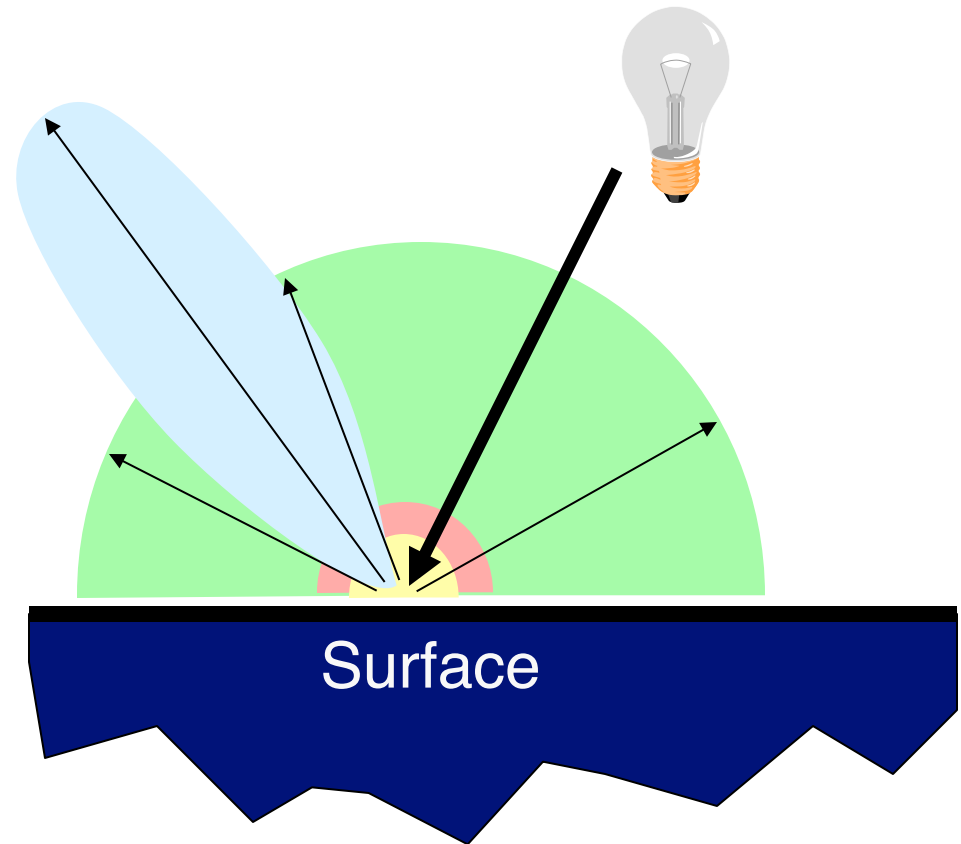


This is a hack (avoids complexity of global illumination)!

# OpenGL Reflectance Model



- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”

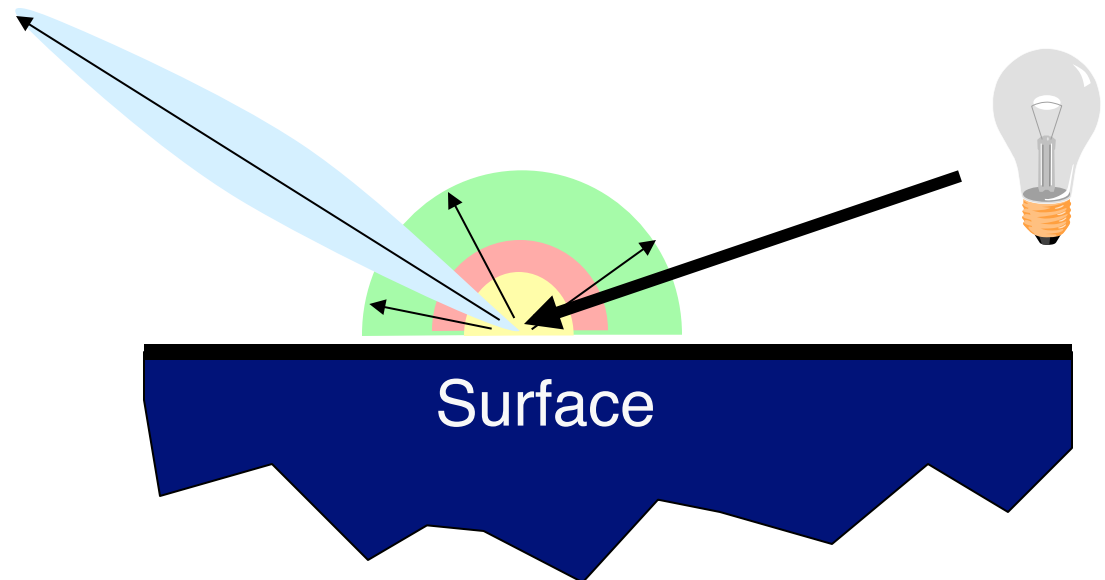




# OpenGL Reflectance Model



- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



# OpenGL Reflectance Model



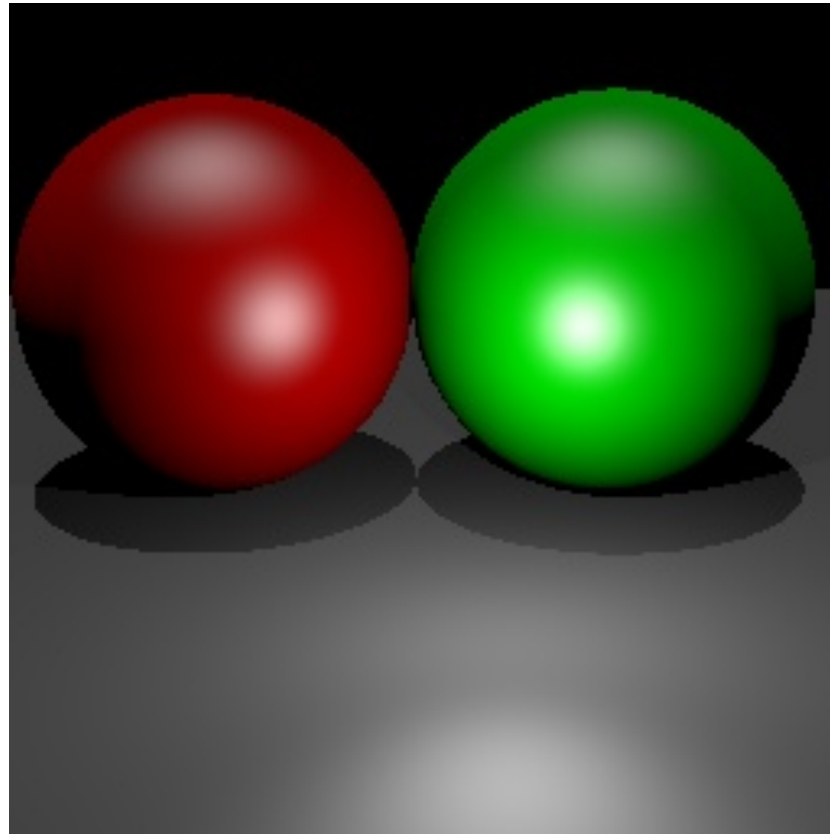
Sum diffuse, specular, emission, and ambient

Phong	$\rho_{\text{ambient}}$	$\rho_{\text{diffuse}}$	$\rho_{\text{specular}}$	$\rho_{\text{total}}$
$\phi_i = 60^\circ$				
$\phi_i = 25^\circ$				
$\phi_i = 0^\circ$				

# OpenGL Reflectance Model

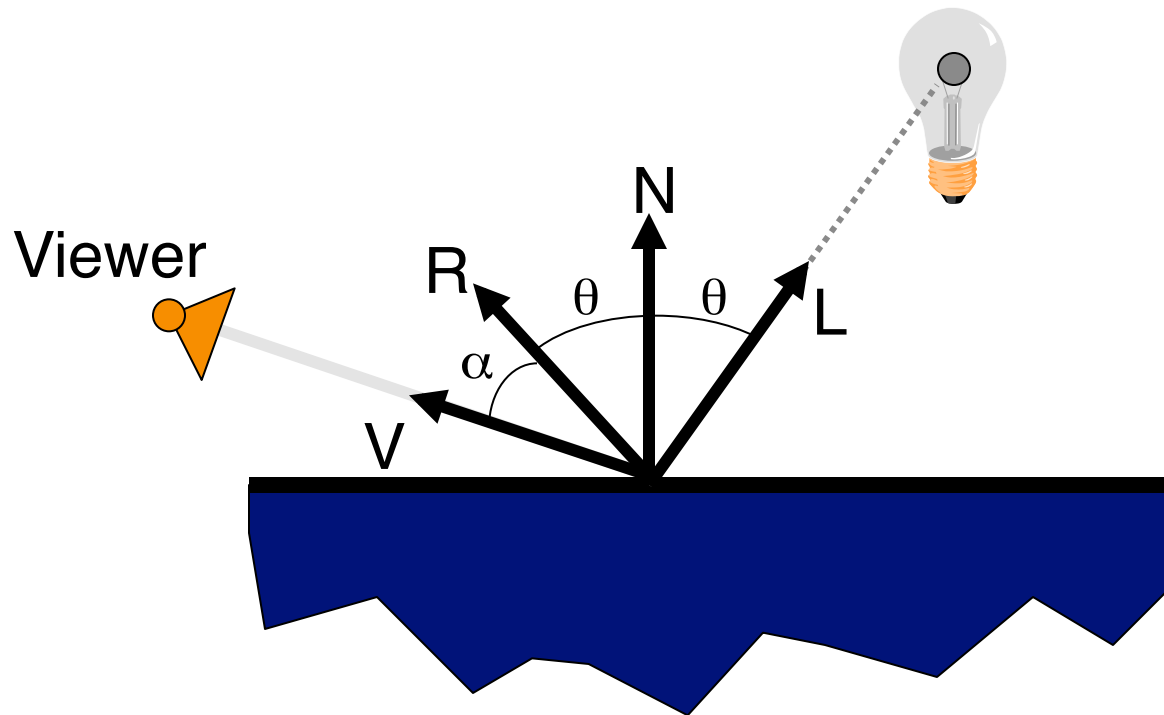


Good model for plastic surfaces, ...



# Direct Illumination Calculation

Single light source:

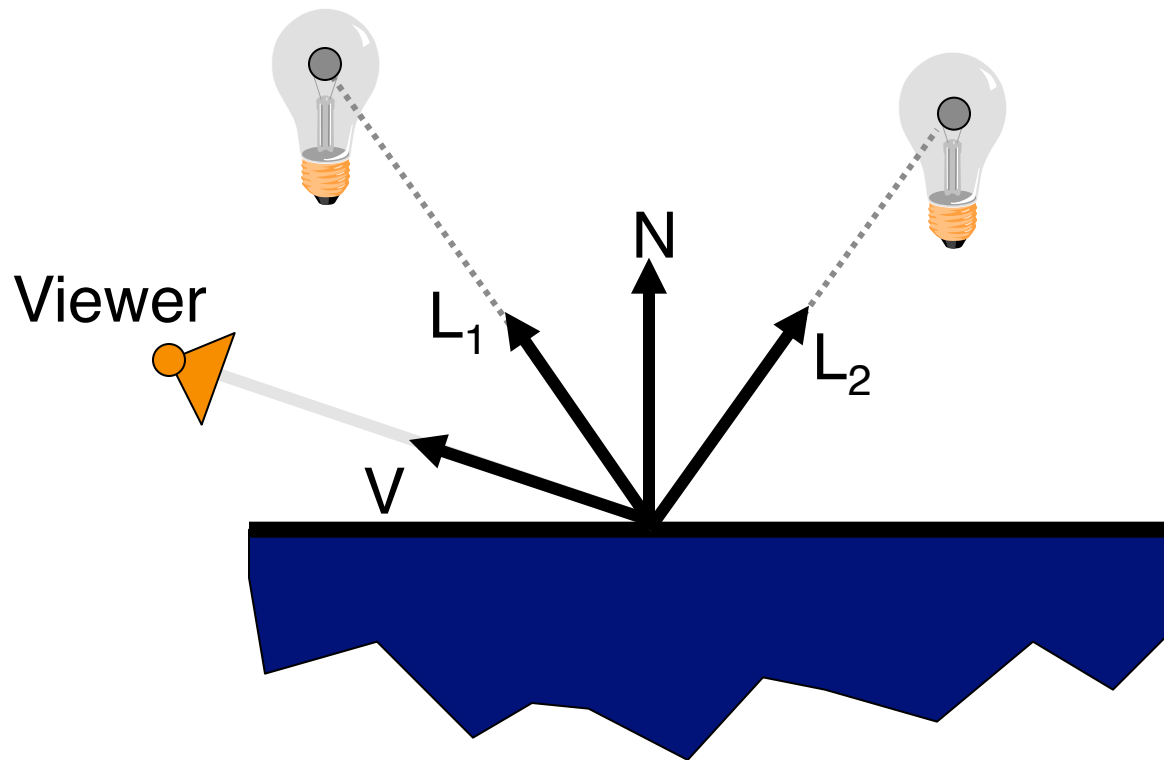


$$I = I_E + K_A I_{AL} + K_D (N \cdot L) I_L + K_S (V \cdot R)^n I_L$$



# Direct Illumination Calculation

Multiple light sources:



**Note:**  
all of the  
 $K$  and  $I$   
are RGB  
colors



$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) I_L$$

# Example from production



This scene had 400 virtual lights (~100 params)



# Overview



- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Transmissions
  - Inter-object reflections



Global Illumination

# Global Illumination



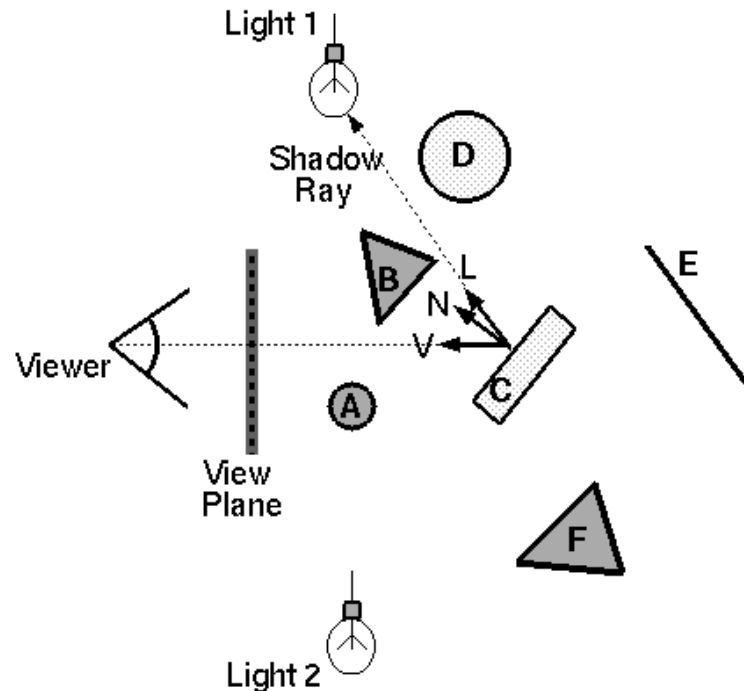
Greg Ward



# Ray Casting (last lecture)

Trace primary rays from camera

- Direct illumination from unblocked lights only

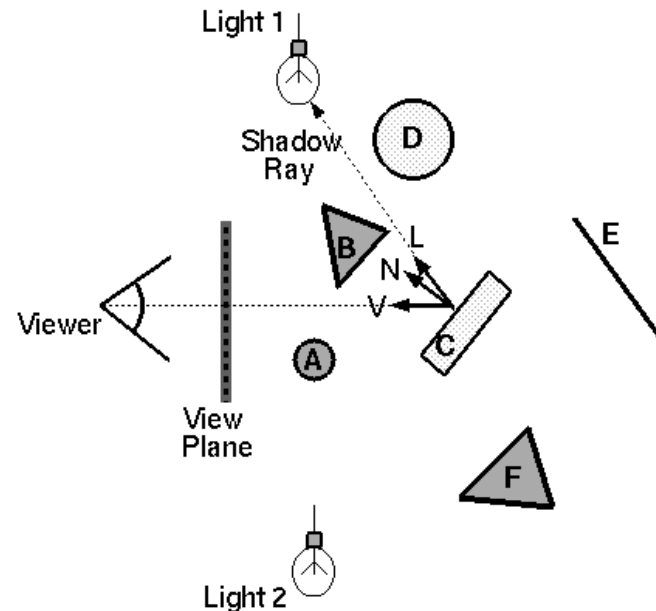


$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) I_L$$

# Shadows

Shadow term tells if light sources are blocked

- Cast ray towards each light source
- $S_L = 0$  if ray is blocked,  $S_L = 1$  otherwise



Shadow  
Term

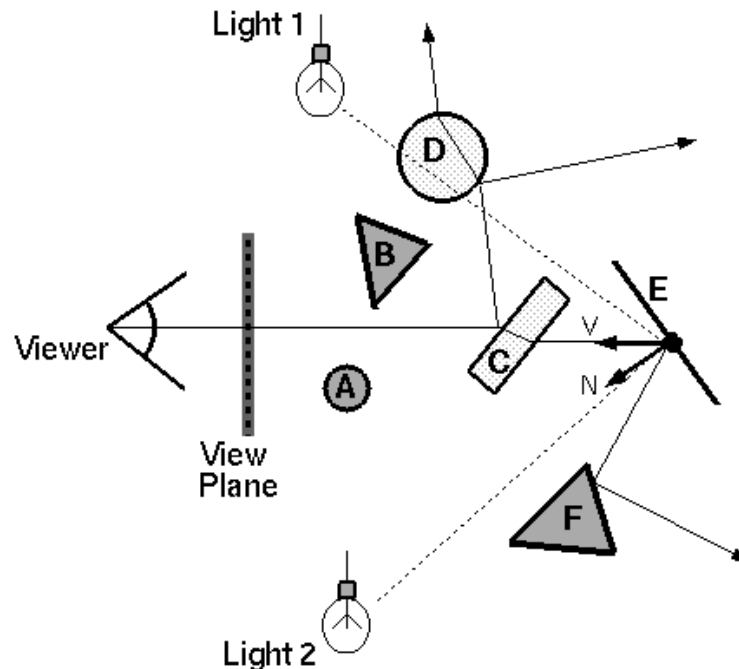


$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L$$

# Recursive Ray Tracing

Also trace secondary rays from hit surfaces

- Mirror reflection and transparency

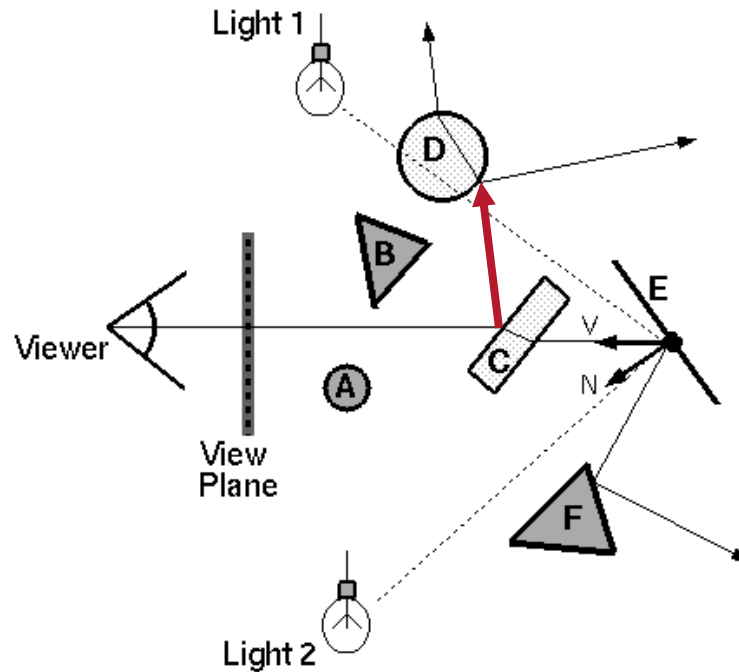


$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

# Mirror reflections

Trace secondary ray in mirror direction

- Evaluate radiance along secondary ray and include it into illumination model



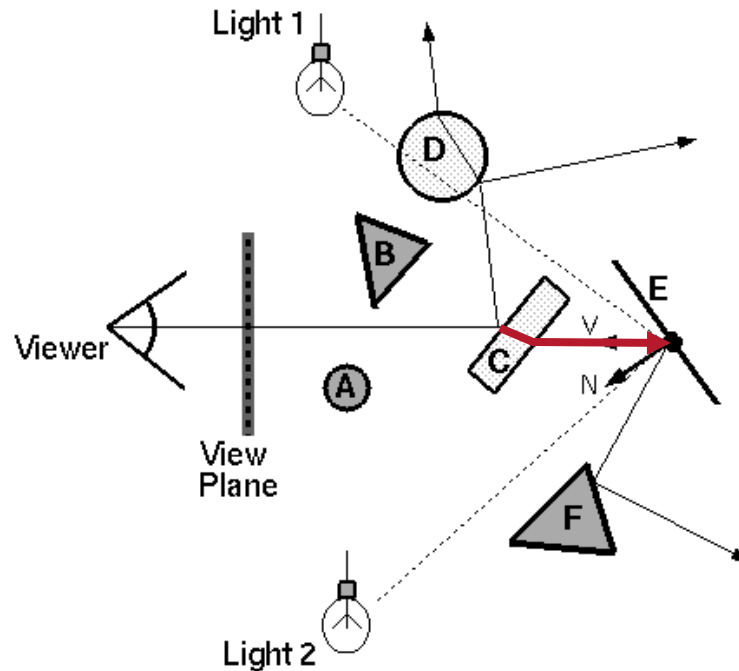
Radiance  
for mirror  
reflection ray

$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

# Transparency

Trace secondary ray in direction of refraction

- Evaluate radiance along secondary ray and include it into illumination model



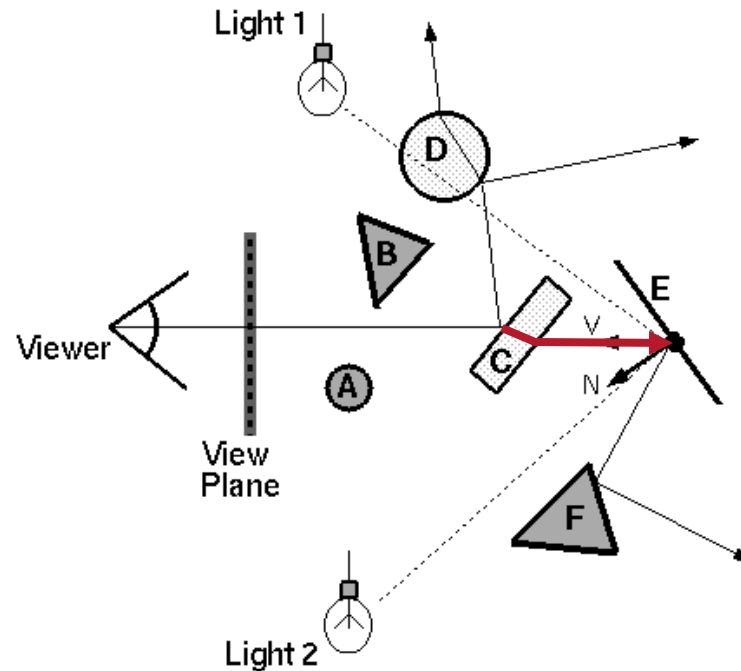
Radiance for refraction ray

$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

# Transparency

Transparency coefficient is fraction transmitted

- $K_T = 1$  for translucent object,  $K_T = 0$  for opaque
- $0 < K_T < 1$  for object that is semi-translucent



Transparency  
Coefficient

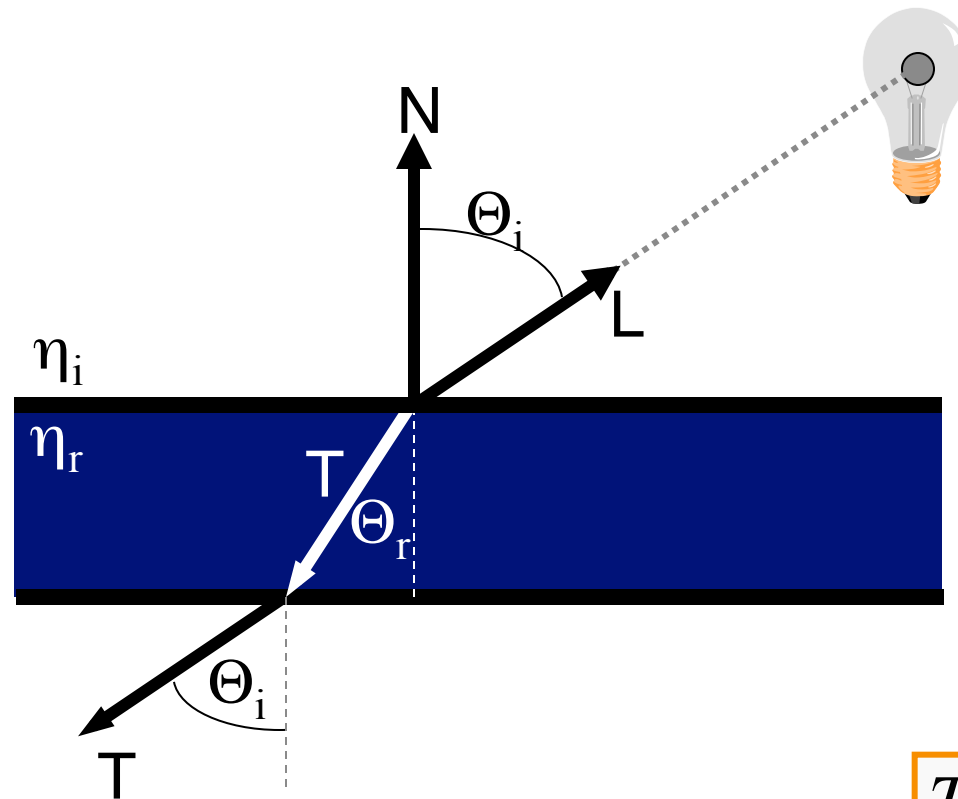
$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$



# Refractive Transparency

For thin surfaces, can ignore change in direction

- Assume light travels straight through surface

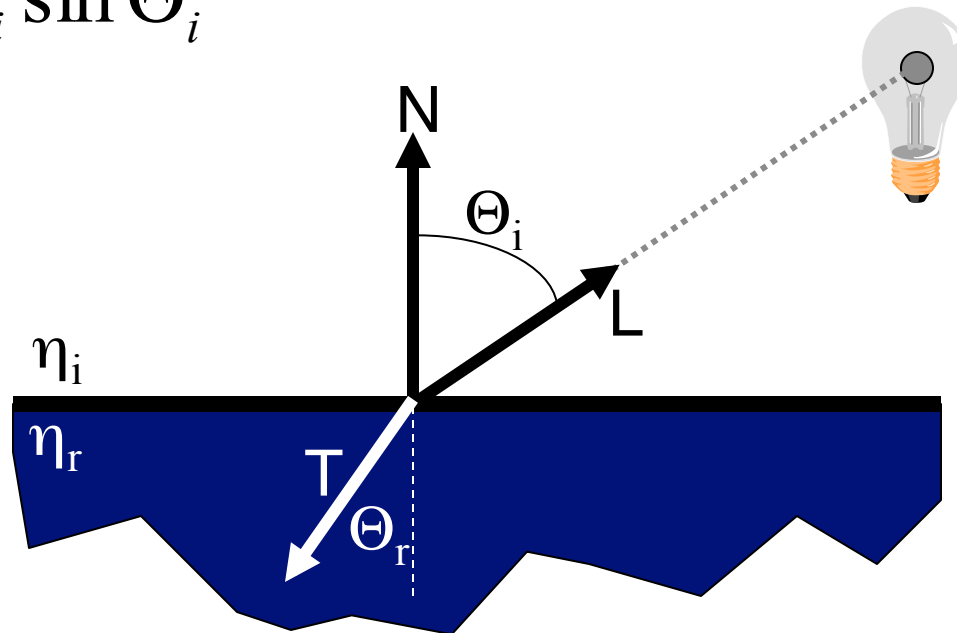


$$T \cong -L$$

# Refractive Transparency

For solid objects, apply Snell's law:

$$\eta_r \sin \Theta_r = \eta_i \sin \Theta_i$$

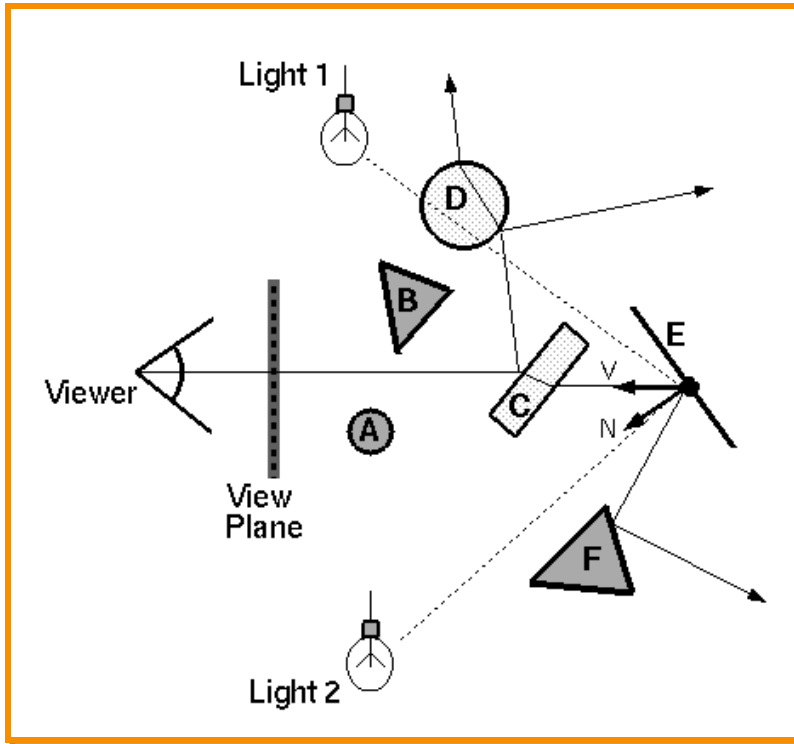


$$T = \left( \frac{\eta_i}{\eta_r} \cos \Theta_i - \cos \Theta_r \right) N - \frac{\eta_i}{\eta_r} L$$

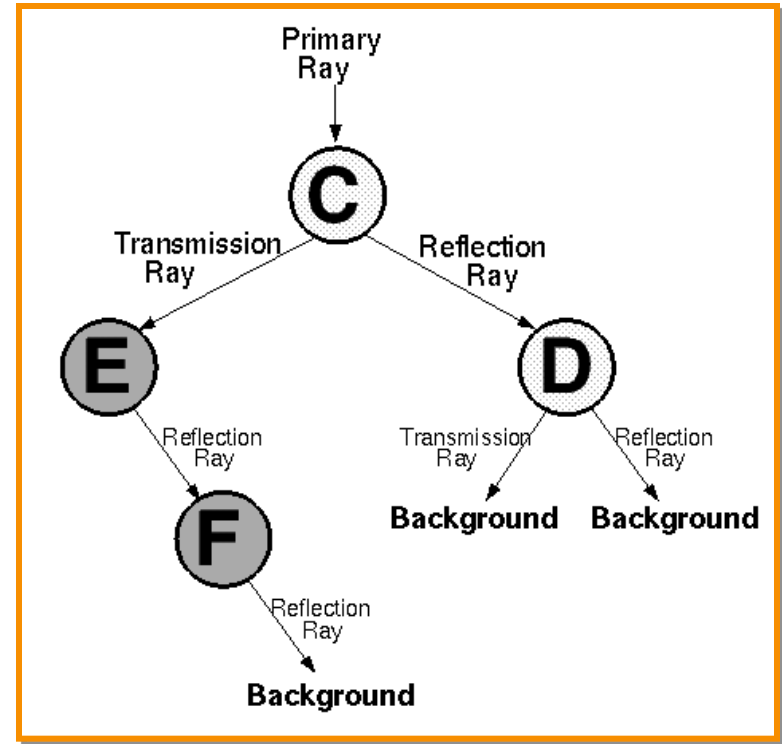


# Recursive Ray Tracing

Ray tree represents illumination computation



Ray traced through scene

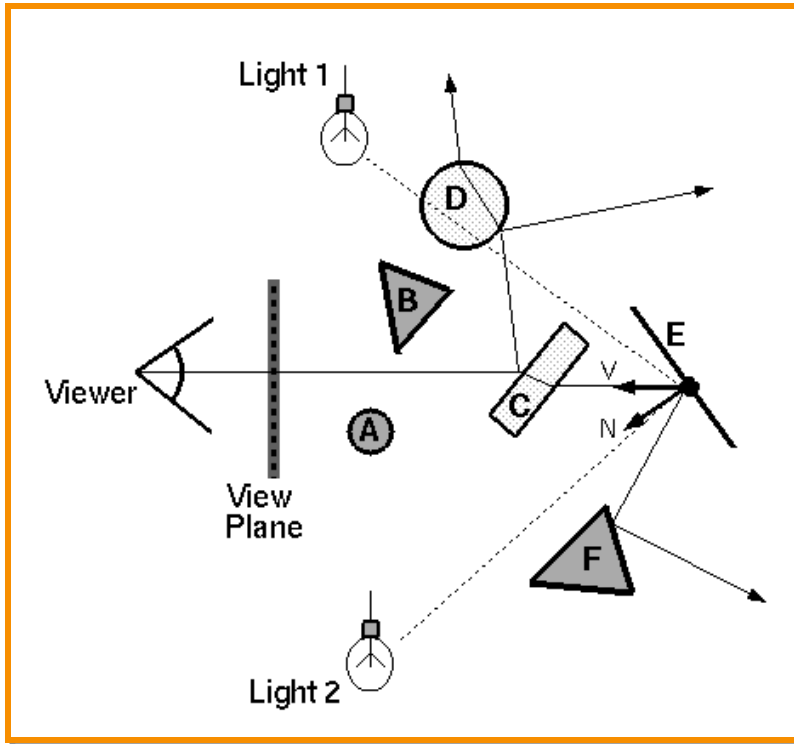


Ray tree

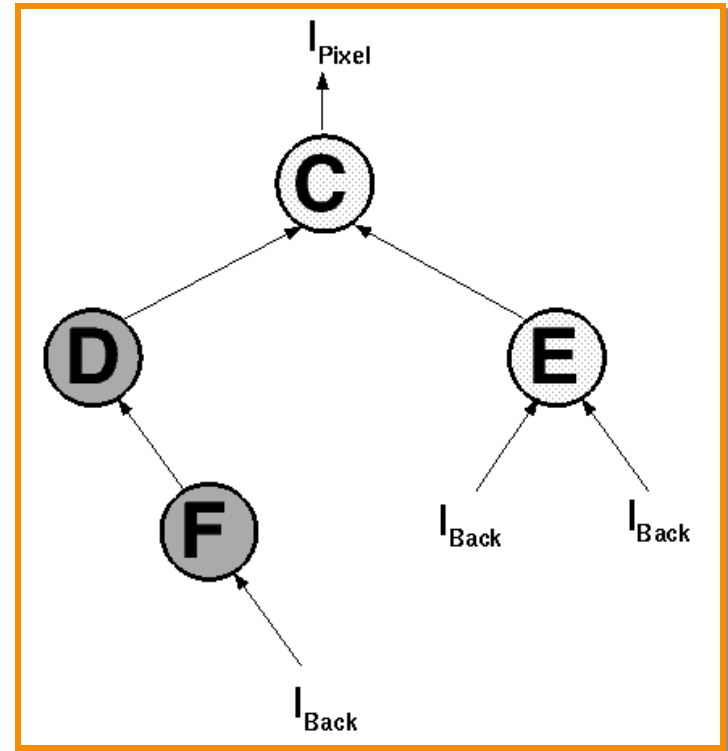
$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

# Recursive Ray Tracing

Ray tree represents illumination computation



Ray traced through scene



Ray tree

$$I = I_E + K_A I_{AL} + \sum_L \left( K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

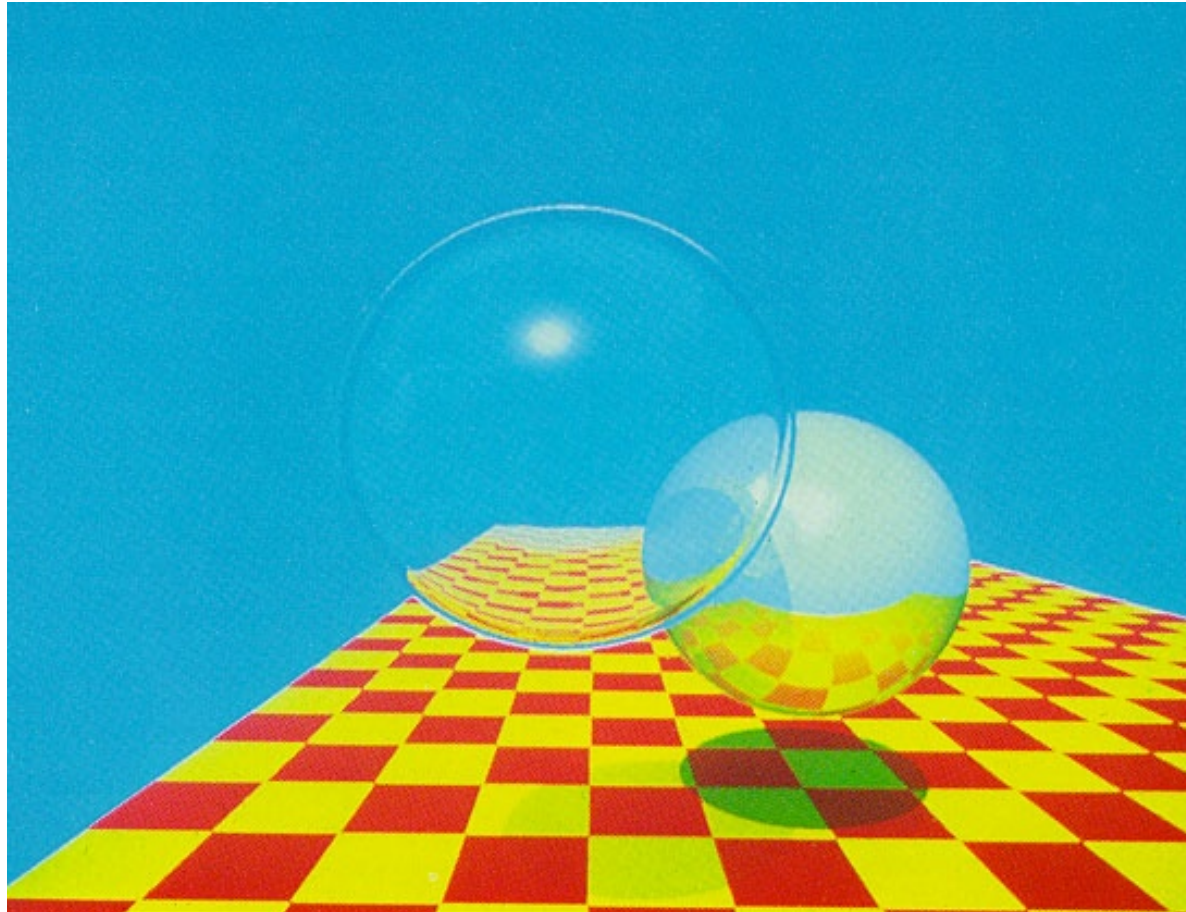
# Recursive Ray Tracing



ComputeRadiance is called recursively

```
R3Rgb ComputeRadiance(R3Scene *scene, R3Ray *ray, R3Intersection& hit)
{
    R3Ray specular_ray = SpecularRay(ray, hit);
    R3Ray refractive_ray = RefractiveRay(ray, hit);
    R3Rgb radiance = Phong(scene, ray, hit) +
                    Ks * ComputeRadiance(scene, specular_ray) +
                    Kt * ComputeRadiance(scene, refractive_ray);
    return radiance;
}
```

# Example



Turner Whitted, 1980

# Summary



- Ray casting (direct Illumination)
  - Usually use simple analytic approximations for light source emission and surface reflectance
- Recursive ray tracing (global illumination)
  - Incorporate shadows, mirror reflections, and pure refractions

All of this is an approximation  
so that it is practical to compute

More on global illumination after next week!



# Illumination Terminology

- Radiant power [flux] ( $\Phi$ )
  - Rate at which light energy is transmitted (in Watts).
- Radiant Intensity (I)
  - Power radiated onto a unit solid angle in direction (in Watts/sr)
    - » e.g.: energy distribution of a light source (inverse square law)
- Radiance (L)
  - Radiant intensity per unit projected surface area (in Watts/m<sup>2</sup>sr)
    - » e.g.: light carried by a single ray (no inverse square law)
- Irradiance (E)
  - Incident flux density on a locally planar area (in Watts/m<sup>2</sup>)
    - » e.g.: light hitting a surface at a point
- Radiosity (B)
  - Exitant flux density from a locally planar area (in Watts/m<sup>2</sup>)