תאורה

מבוסס (מאוד) על

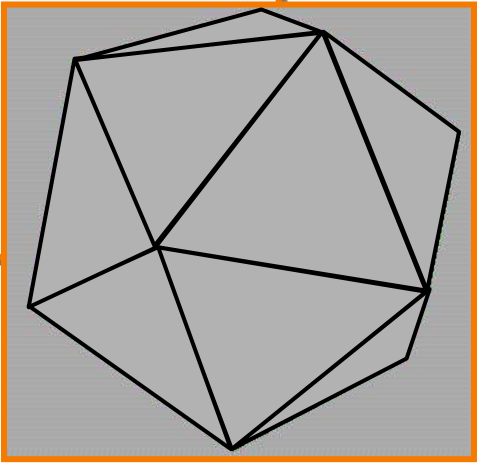
Thomas Funkhouser

Princeton University

C0S 426, Fall 2000

Ray Casting

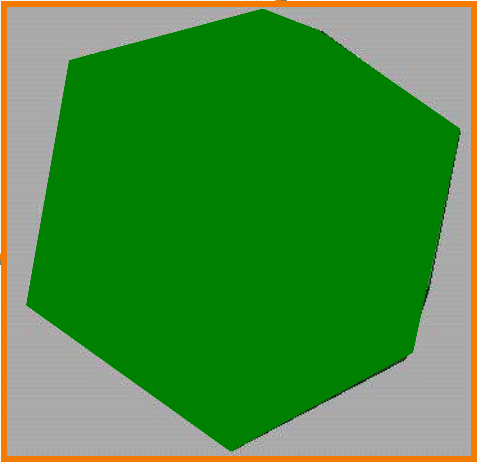
```
Image RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```



Wireframe

Ray Casting

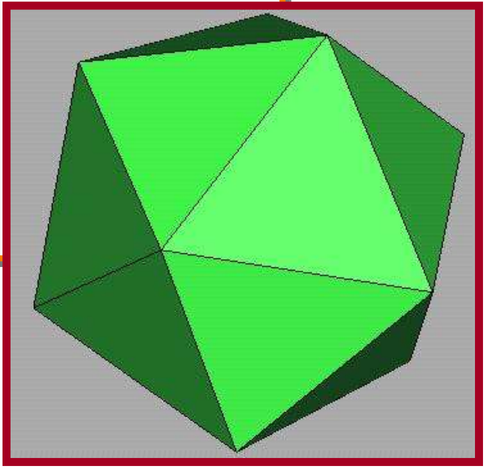
```
Image RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```



Without Illumination

Ray Casting

```
Image RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```

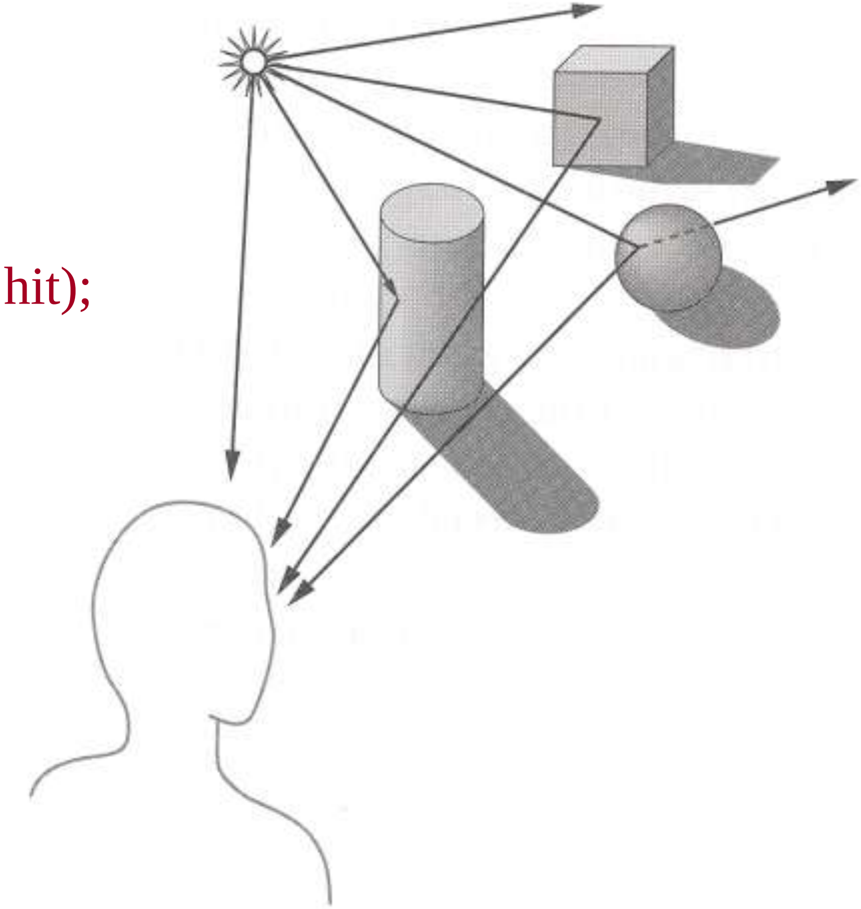


With Illumination

Illumination

- How do we compute radiance for a sample ray?

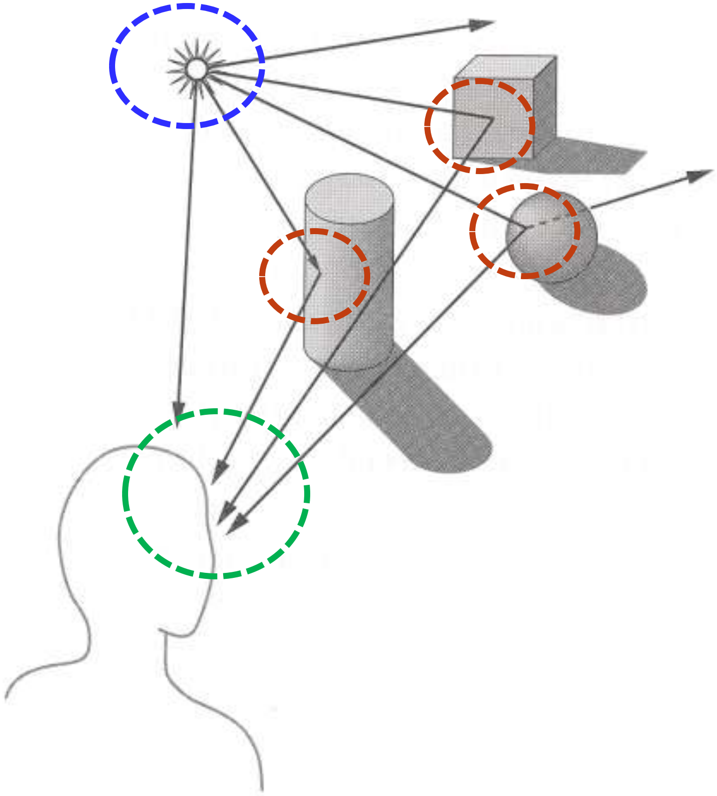
```
image[i][j] = GetColor(scene, ray, hit);
```



Angel Figure 6.2

Goal

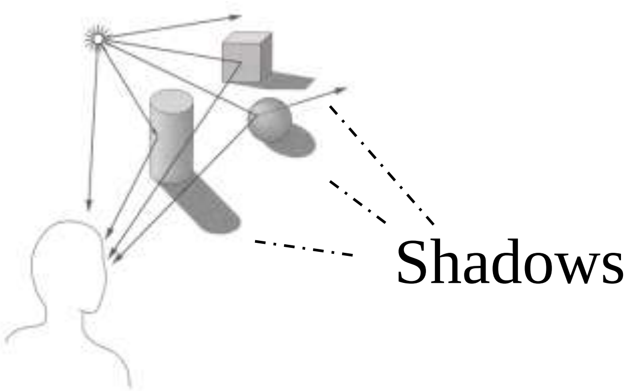
- Must derive computer models for ...
  - Emission at light sources
  - Scattering at surfaces
  - Reception at the camera

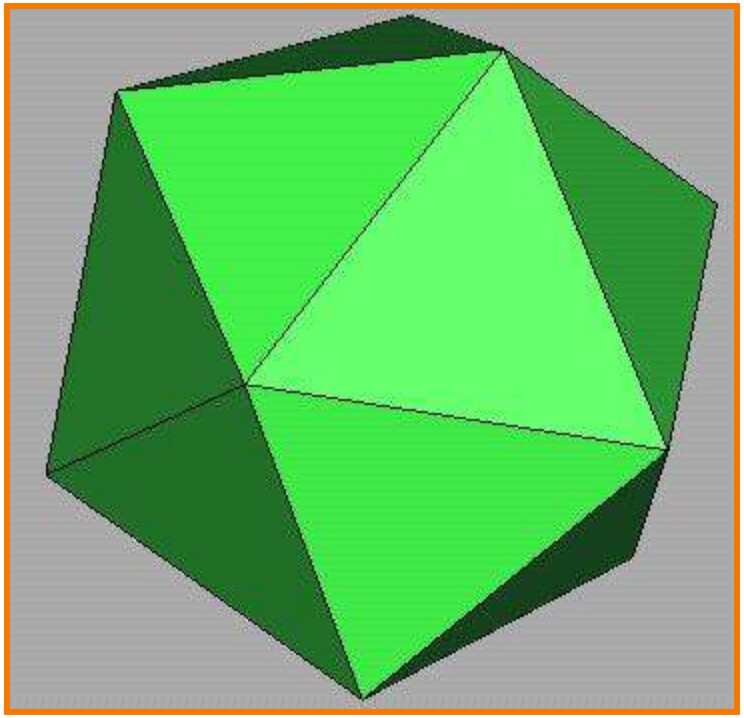


- Desirable features …
  - Concise
  - Efficient to compute
  - “Accurate”

Overview

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

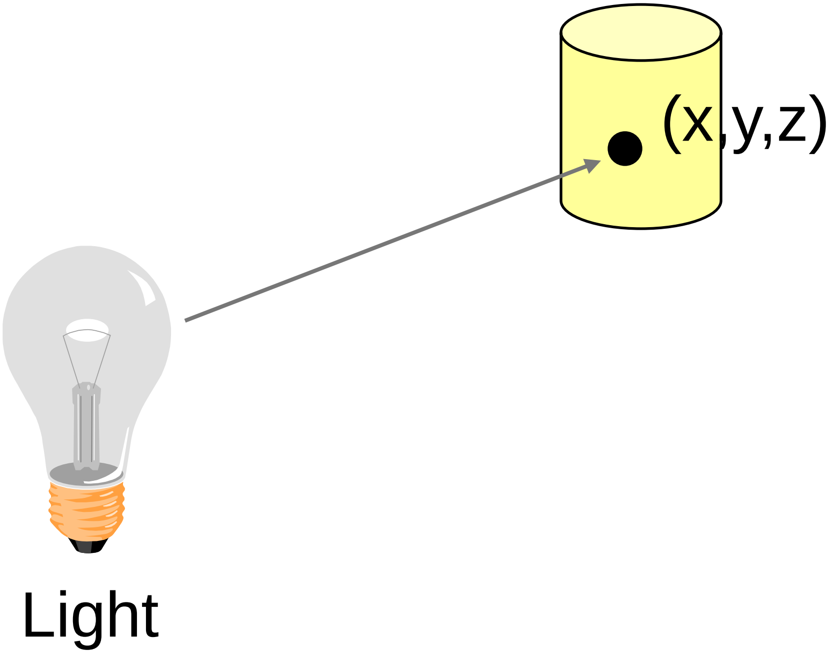




Direct Illumination

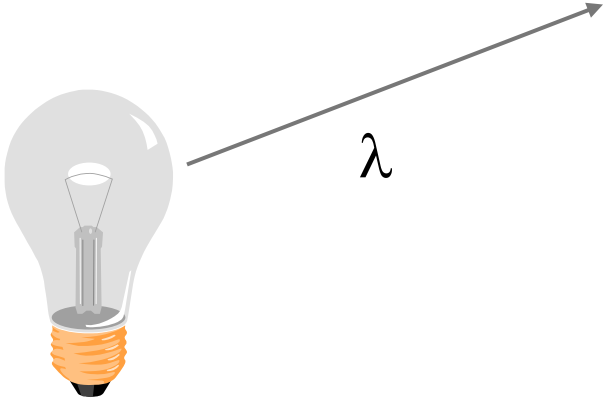
Modeling Light Sources

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



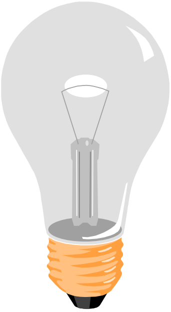
Empirical Models

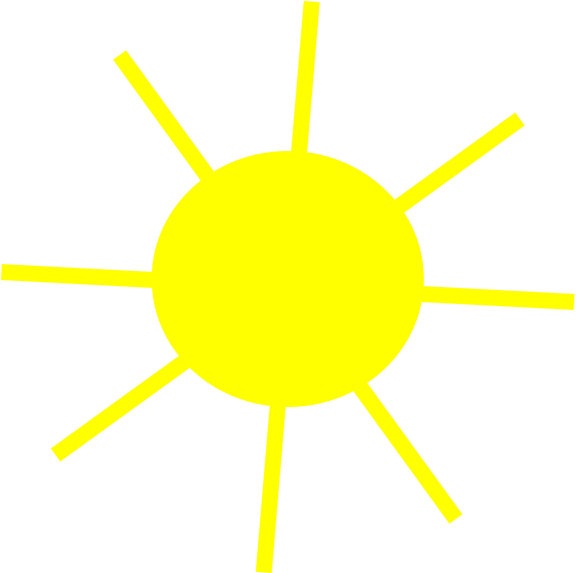
- Ideally measure irradiant energy for “all” situations
  - Too much storage
  - Difficult in practice

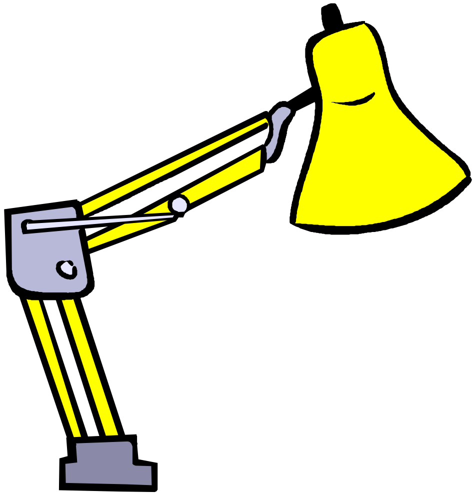


OpenGL Light Source Models

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

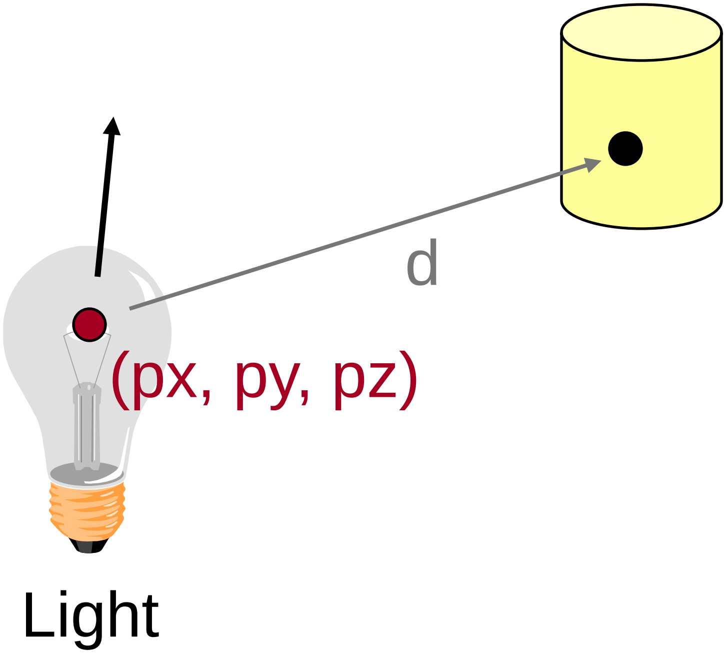






Point Light Source

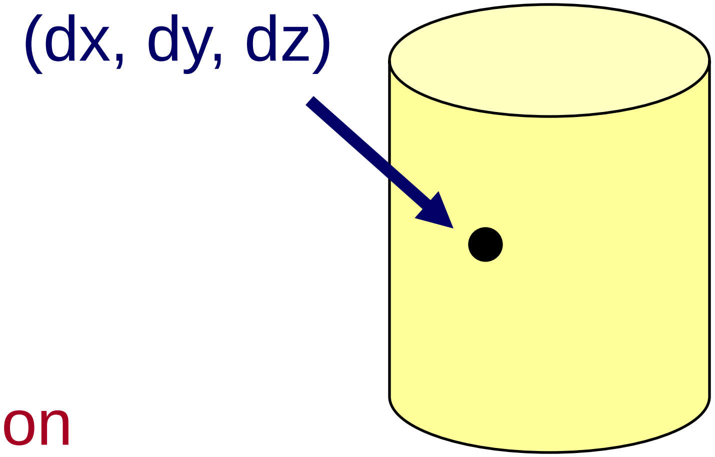
- Models omni-directional point source (e.g., bulb)
  - intensity (I_0),
  - position (px, py, pz),
  - factors (k_c, k_l, k_q) for attenuation with distance (d)



$$I_L = \frac{\mathrm{I}_0}{\mathrm{k}_\mathrm{c} + \mathrm{k}_\mathrm{l} d + \mathrm{k}_\mathrm{q} d^2}$$

Directional Light Source

- Models point light source at infinity (e.g., sun)
  - intensity (I_0),
  - direction (dx,dy,dz)

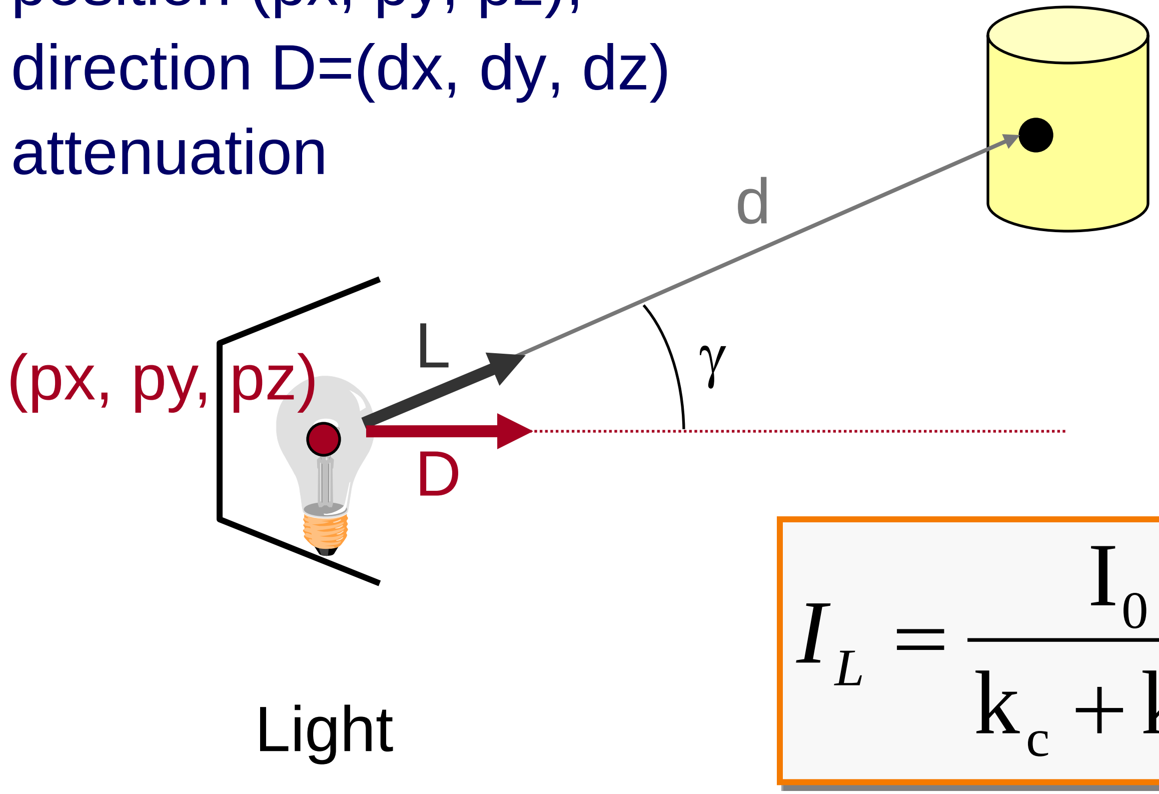


No attenuation
with distance

$$I_L = I_0$$

Spot Light Source

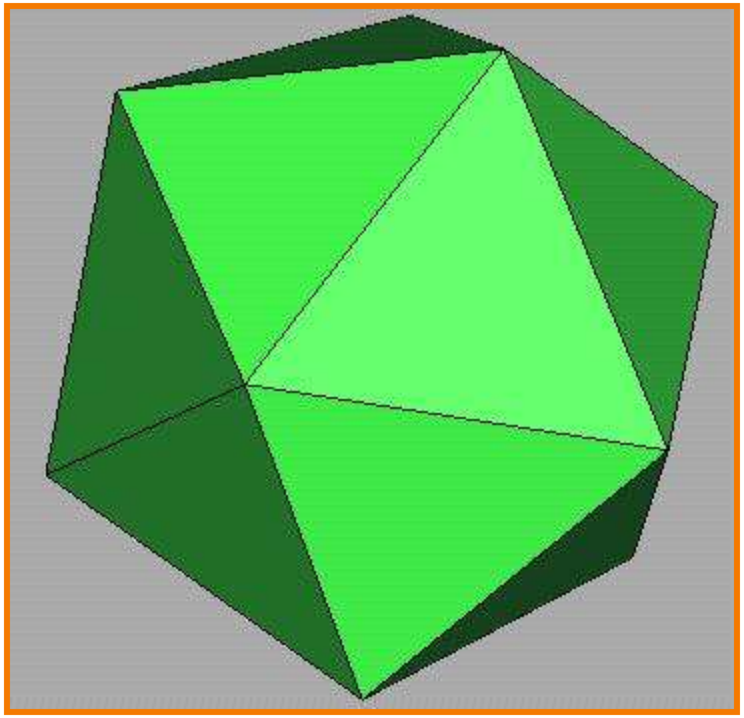
- Models point light source with direction (e.g., Luxo)
  - intensity (I_0),
  - position (px, py, pz),
  - direction D=(dx, dy, dz)
  - attenuation



$$I_L = \frac{I_0 (D \bullet L)}{k_c + k_l d + k_q d^2}$$

Overview

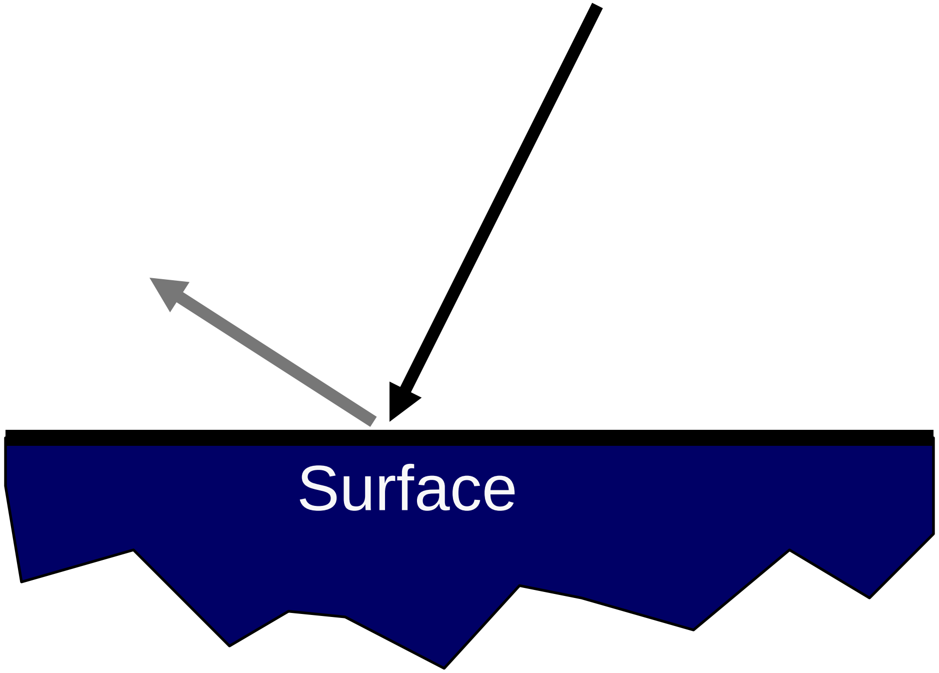
- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Refractions
  - Inter-object reflections



Direct Illumination

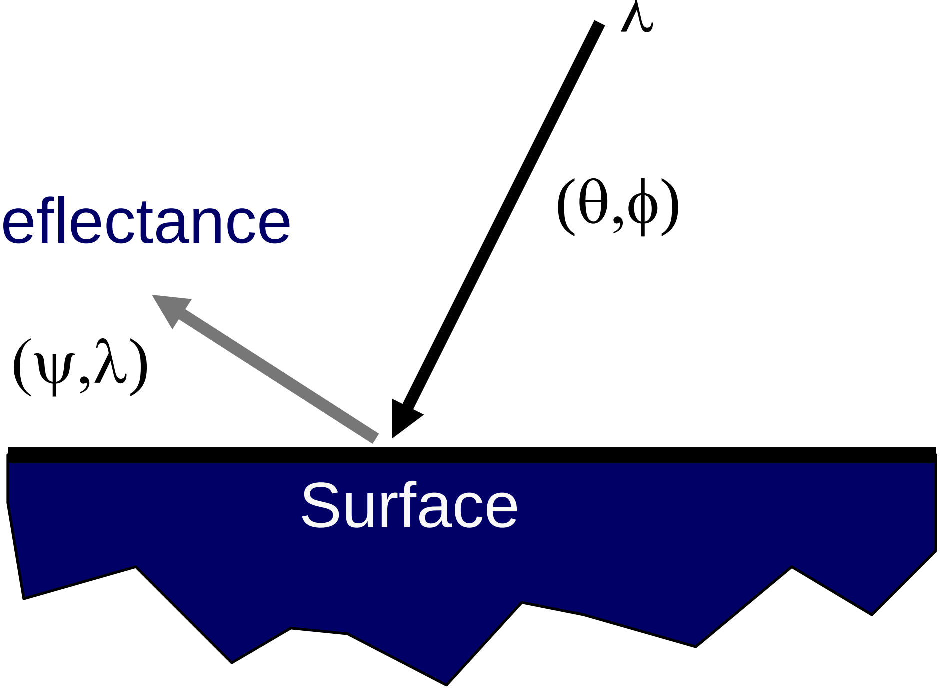
Modeling Surface Reflectance

- $R_s(\theta,\phi,\gamma,\psi,\lambda)$ ...
    - describes the amount of incident energy,
    - arriving from direction (θ,ϕ), ...
    - leaving in direction (γ,ψ), …
    - with wavelength λ



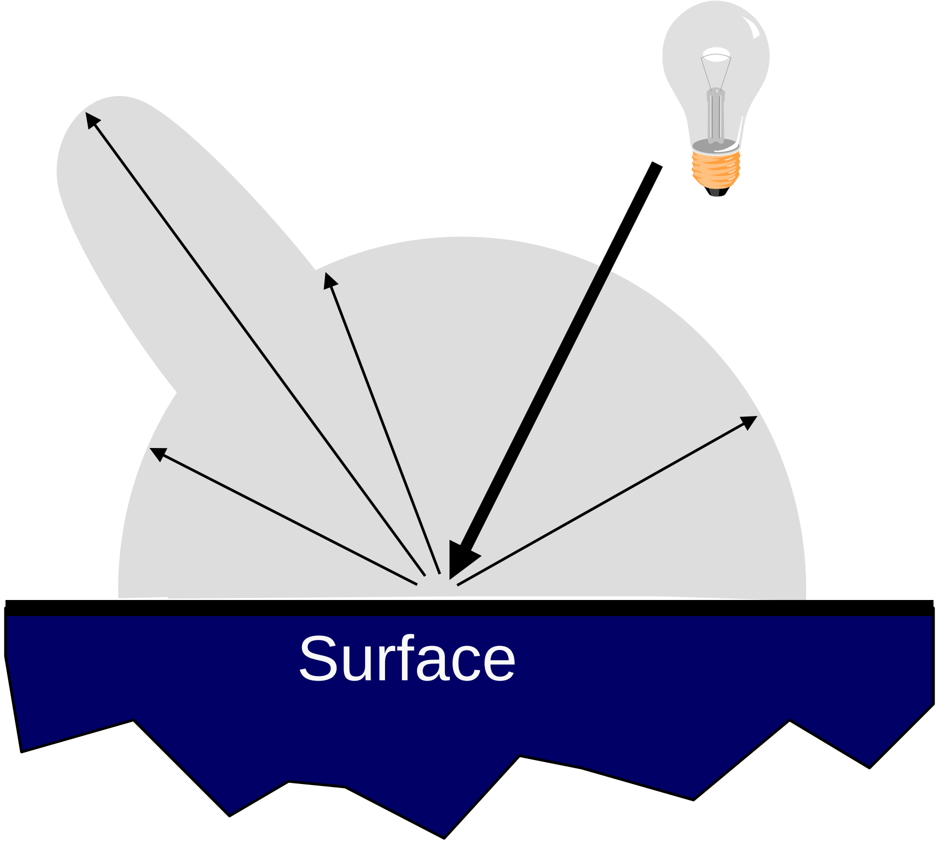
Empirical Models

- Ideally measure radiant energy for "all" combinations of incident angles
  - Too much storage
  - Difficult in practice
- Example
  - BRDF (Bidirectional reflectance distribution function)



OpenGL Reflectance Model

- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - "ambient"

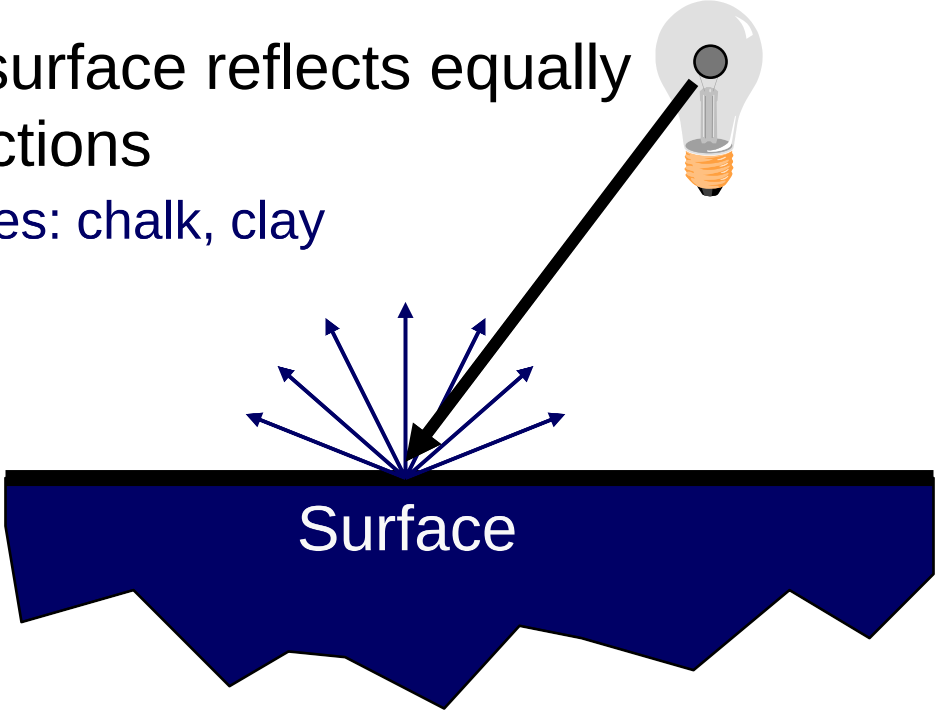


Based on model proposed by Phong in his PhD dissertation 1973

Diffuse Reflection

- Diffuse: Spread Out / To pass by spreading every way / To extend in all directions

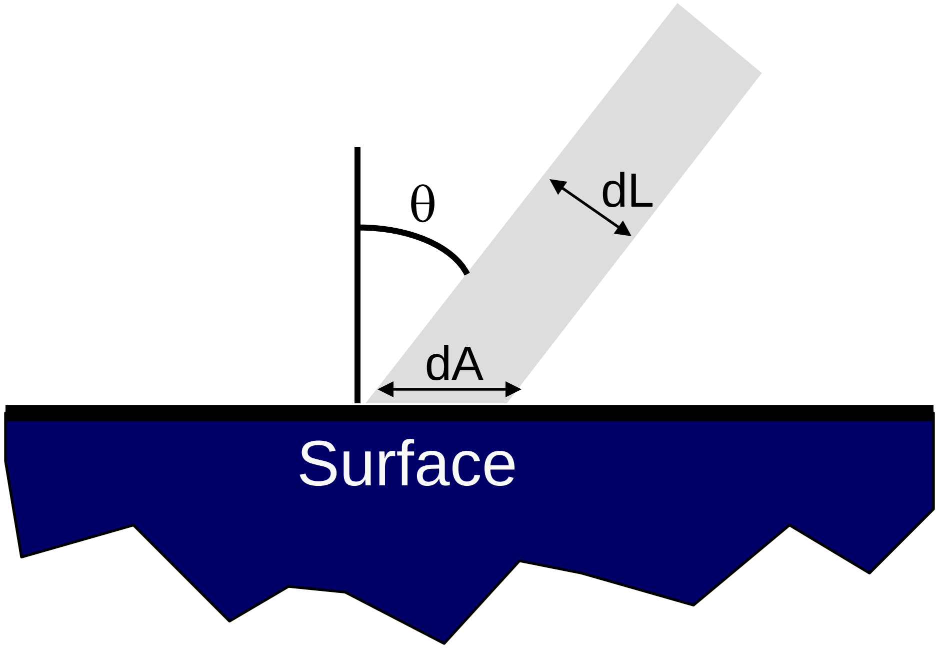
- Assume surface reflects equally in all directions
  - Examples: chalk, clay



Diffuse Reflection

- How much light is reflected?
  - Depends on angle of incident light

$$dL = dA\cos\Theta$$



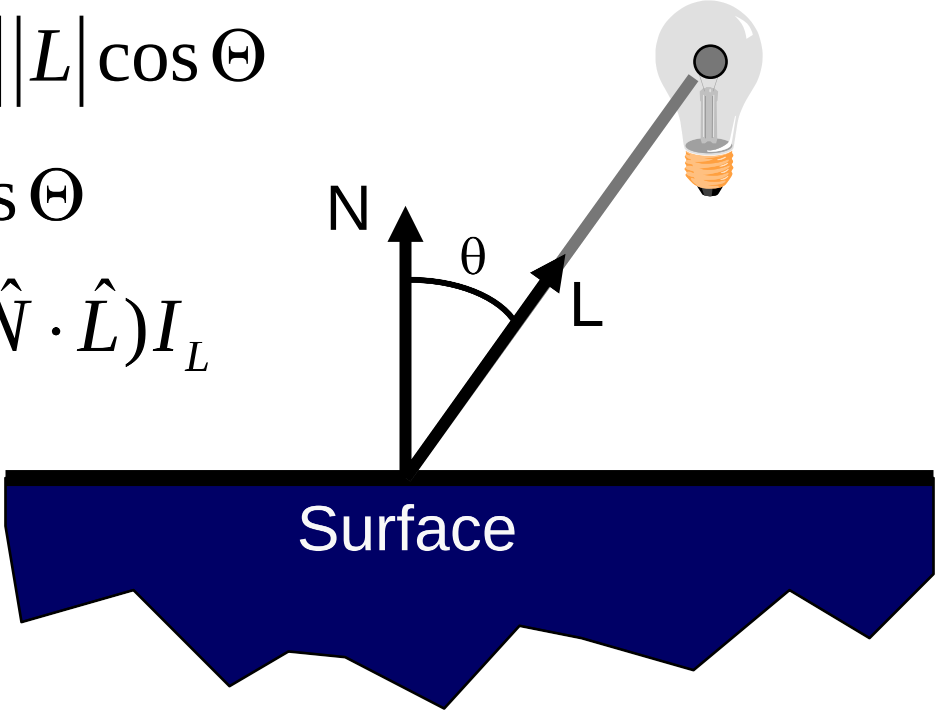
Diffuse Reflection

- Lambertian model
  - cosine law (dot product)

$$N \cdot L = |N||L| \cos \Theta$$

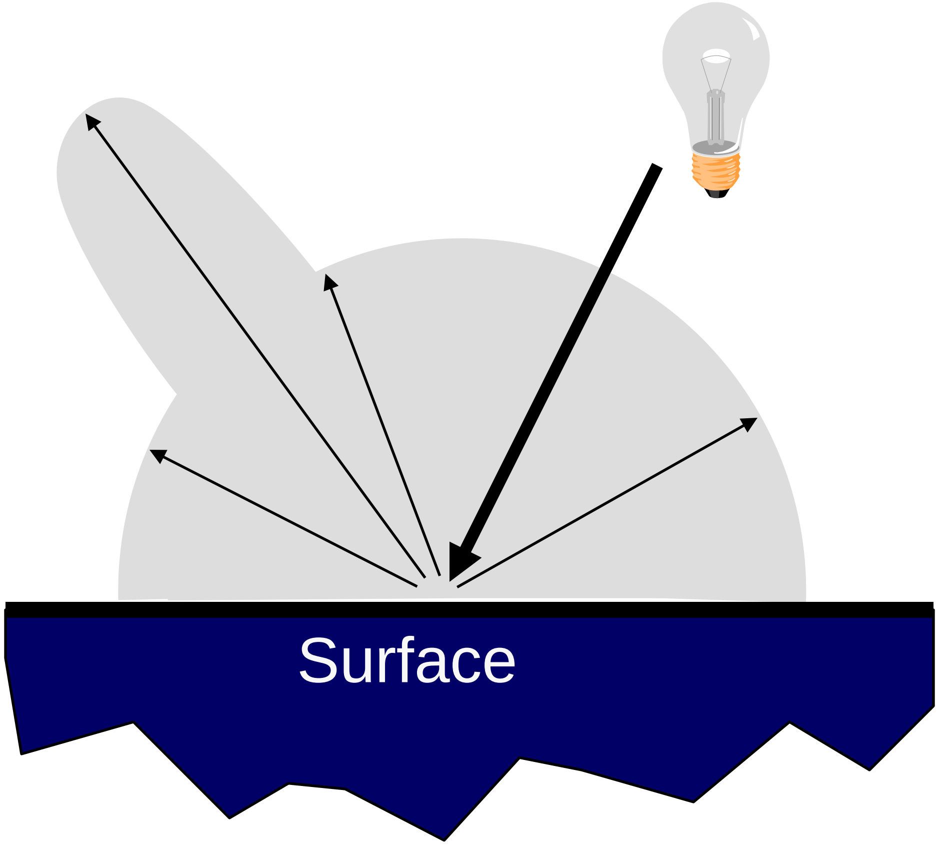
$$\hat{N} \cdot \hat{L} = \cos \Theta$$

$$I_D = K_D(\hat{N} \cdot \hat{L}) I_L$$



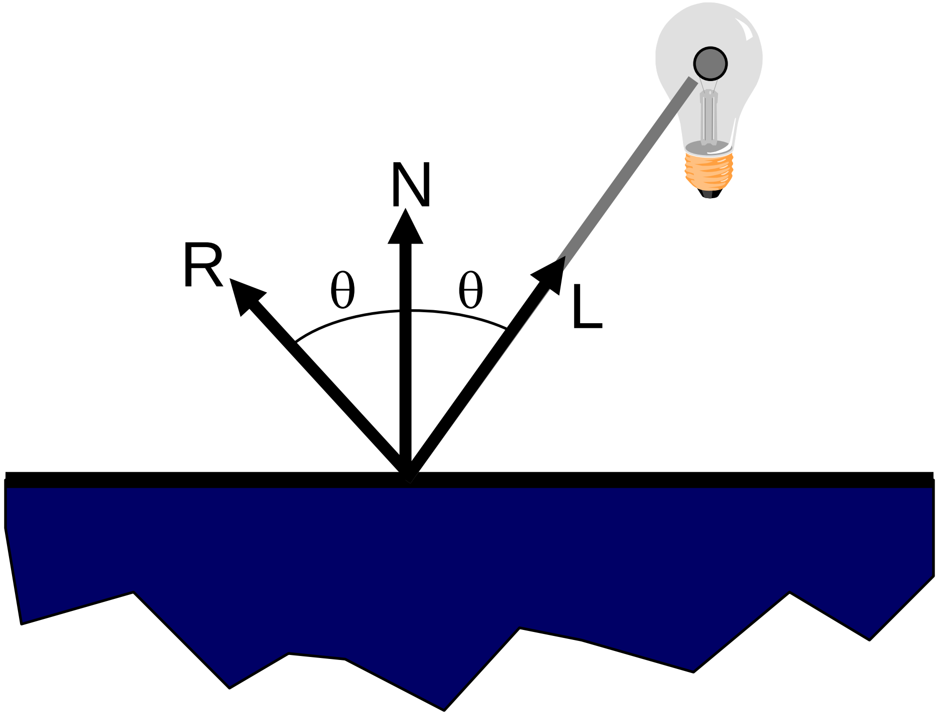
OpenGL Reflectance Model

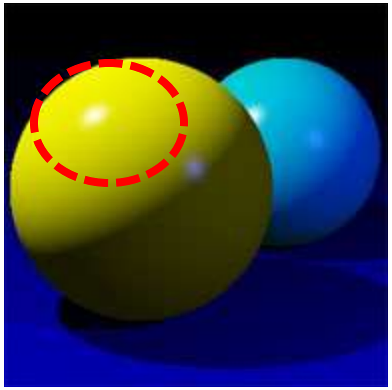
- Simple analytic model:
    - diffuse reflection +
    - specular reflection +
    - emission +
    - “ambient”



Specular Reflection

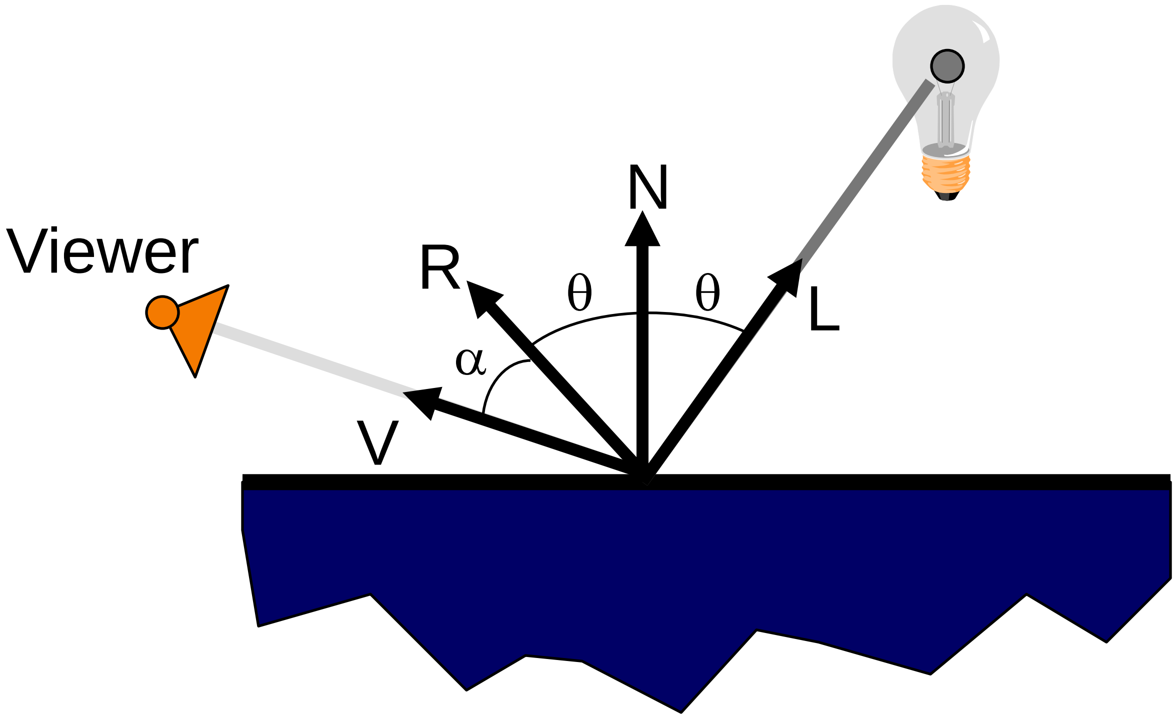
- Reflection is strongest near mirror angle
  - Examples: mirrors, metals





Specular Reflection

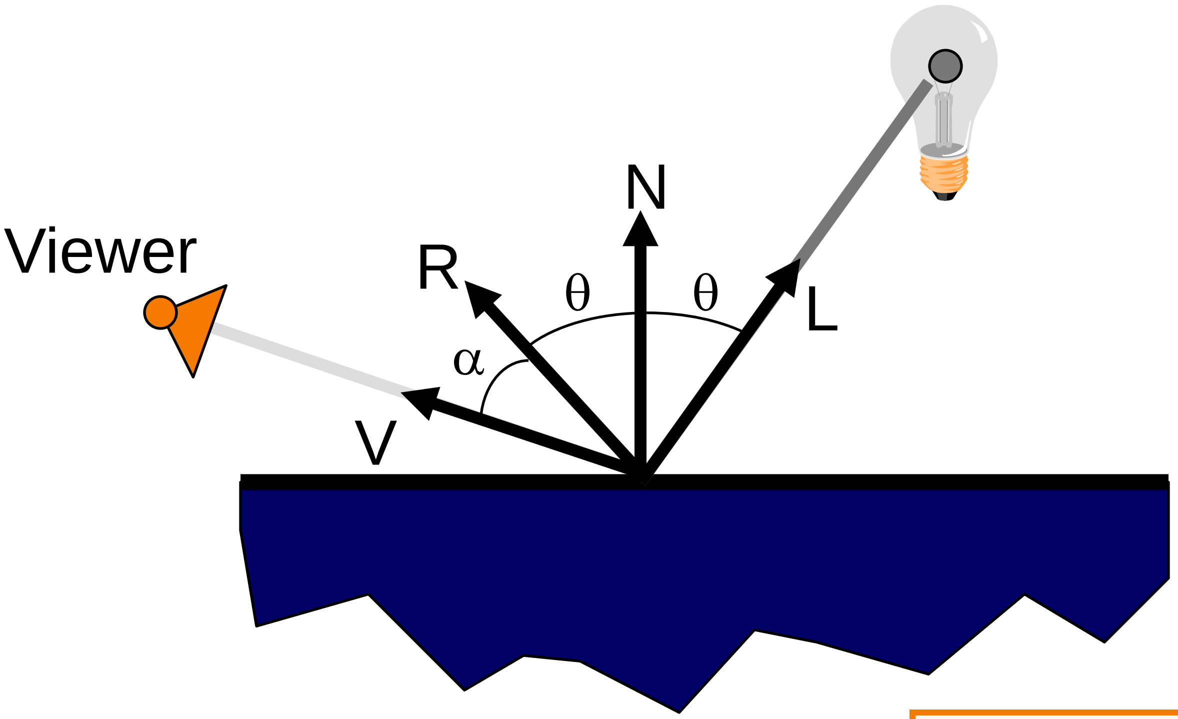
- How much light is seen?
  - Depends on angle of incident light and angle to viewer



Specular Reflection

- Phong Model
  - $\cos(\alpha)^n$

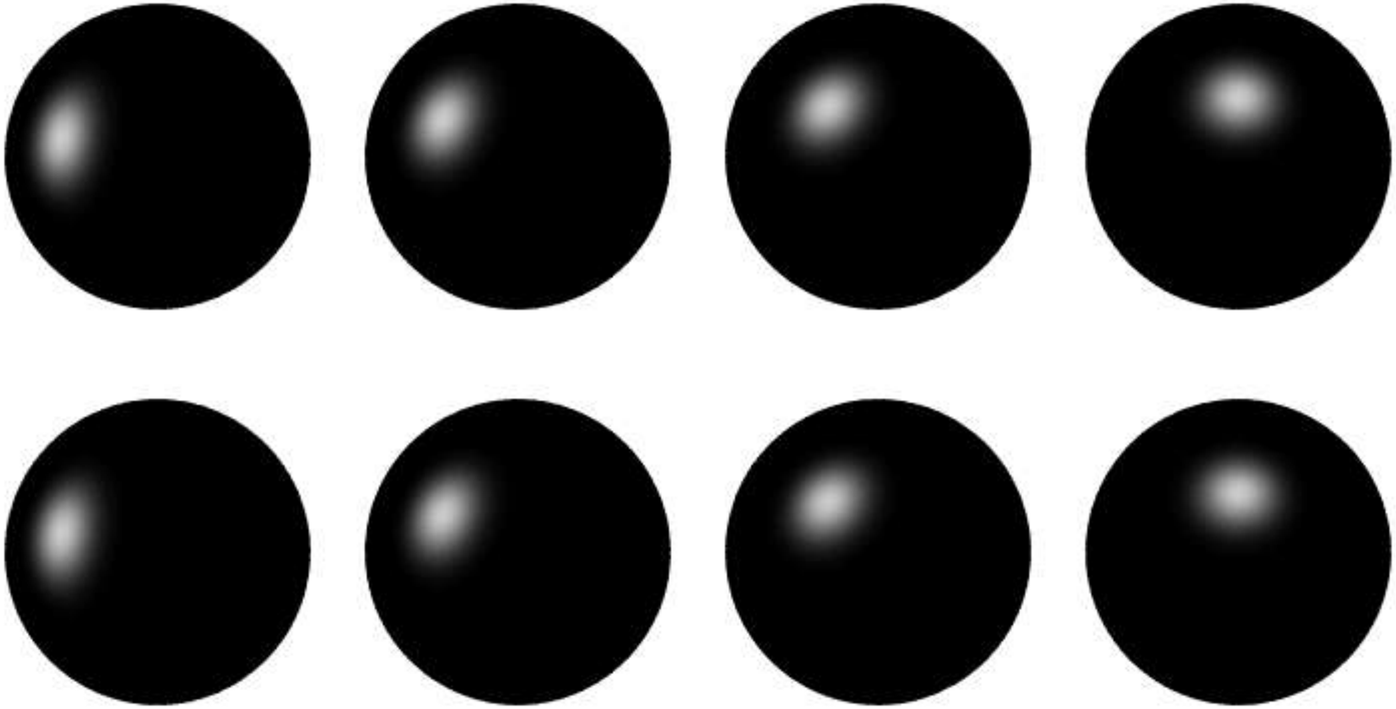
Phong exponent: apparent smoothness of the surface



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

Specular Reflection

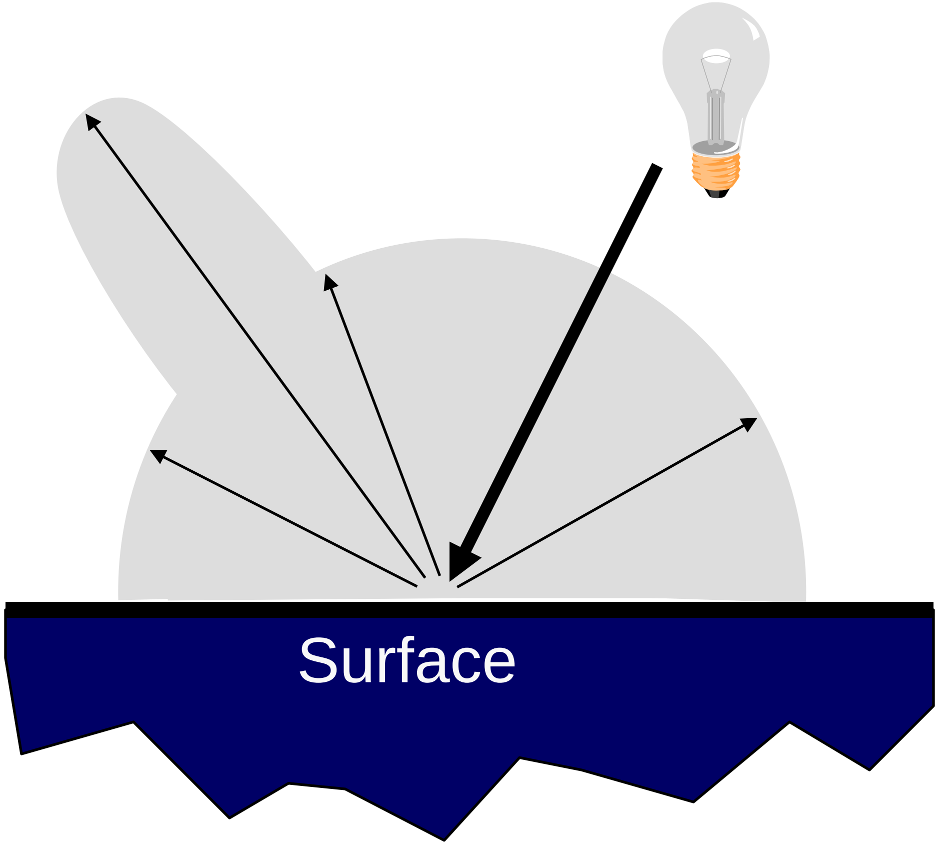
- Phong Examples



Direction of light source and shininess exponent is varied

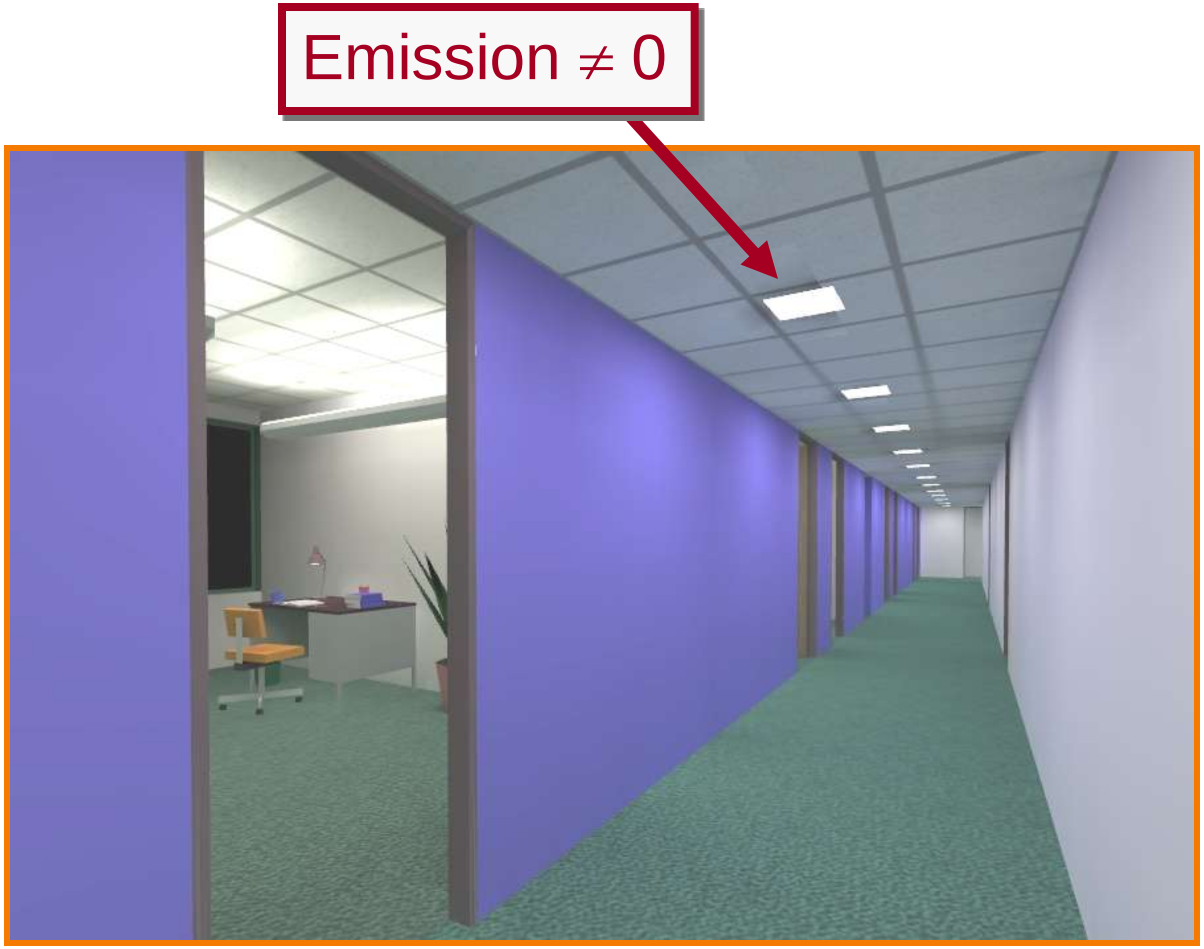
OpenGL Reflectance Model

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



Emission

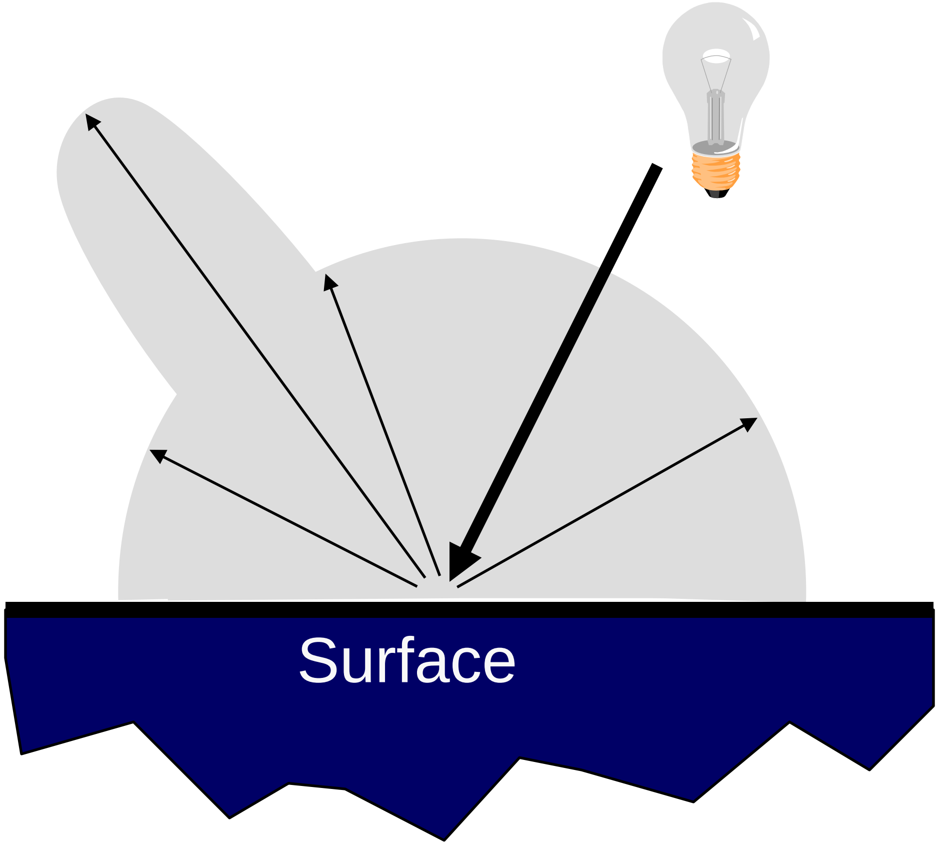
- Represents light eminating directly from polygon



Emission ≠ 0

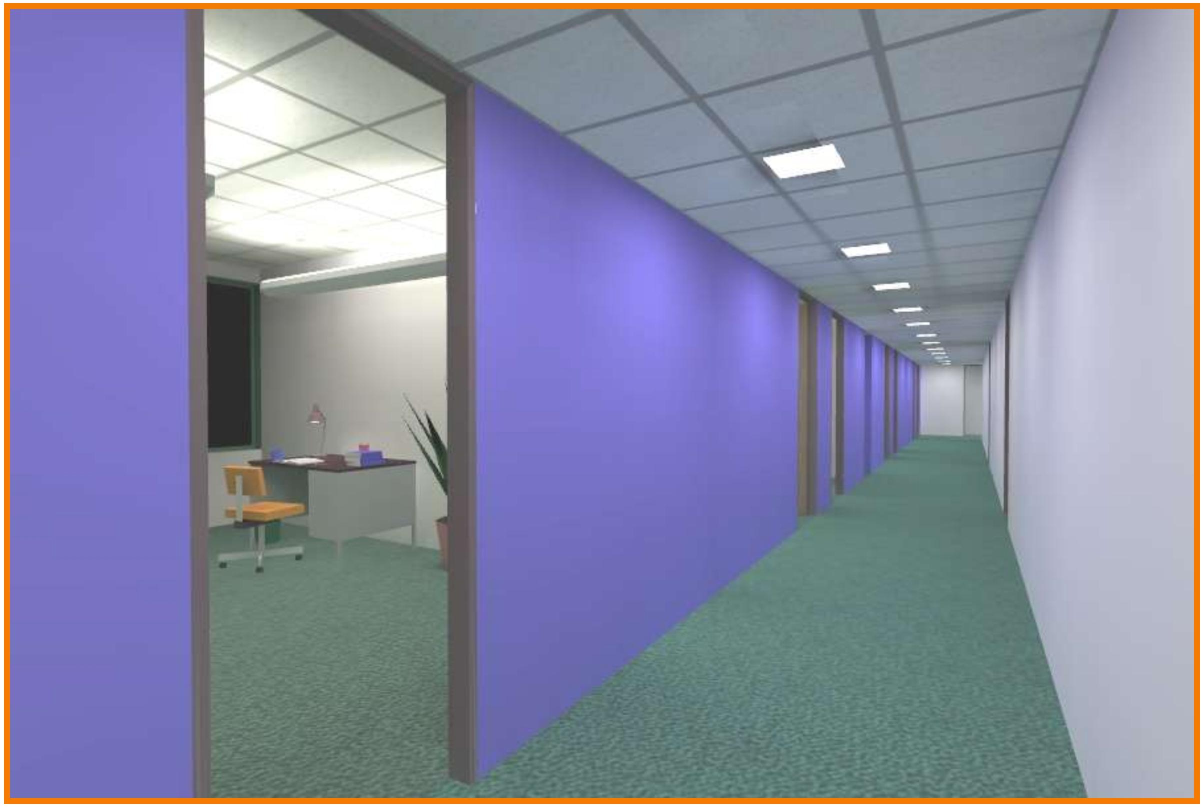
OpenGL Reflectance Model

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



Ambient Term

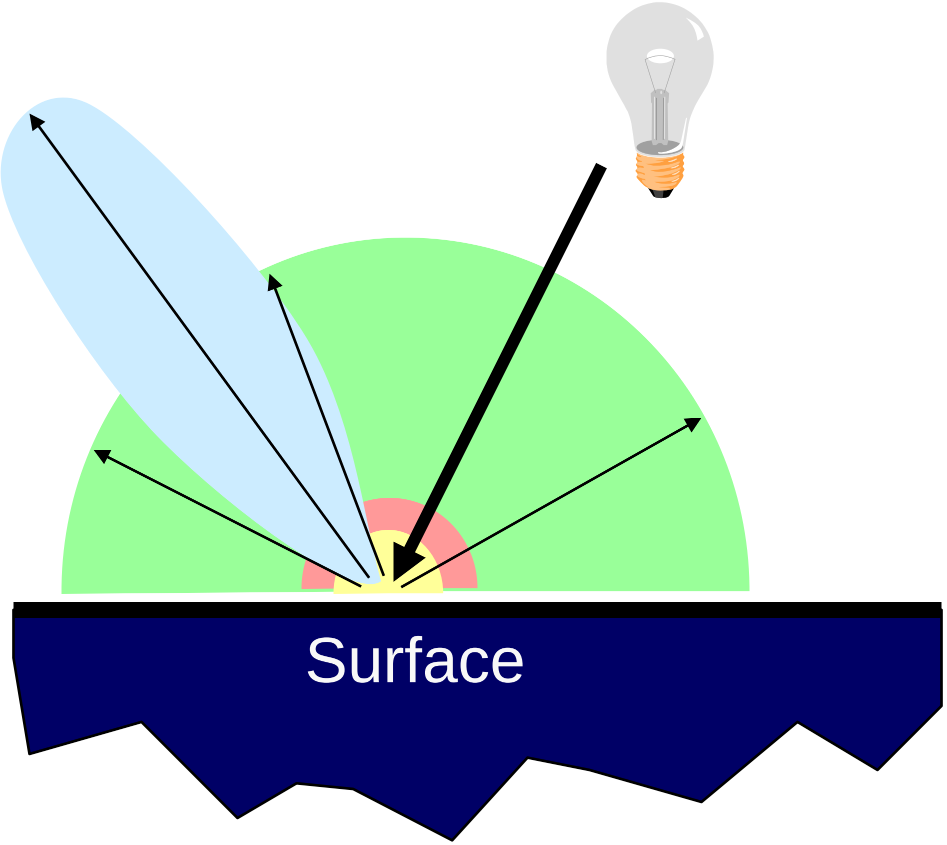
- Represents reflection of all indirect illumination



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

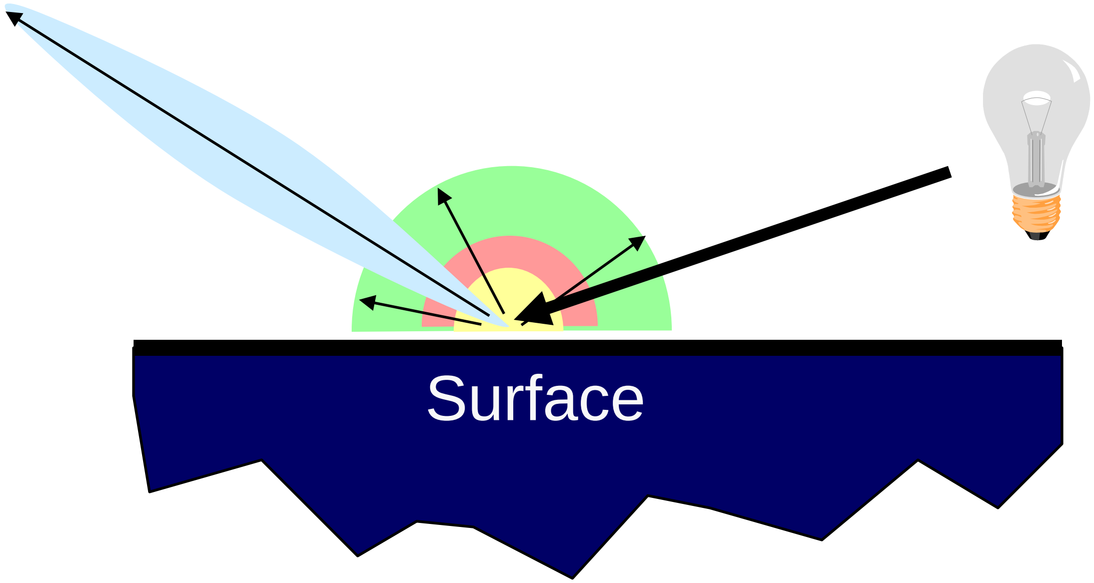
OpenGL Reflectance Model

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



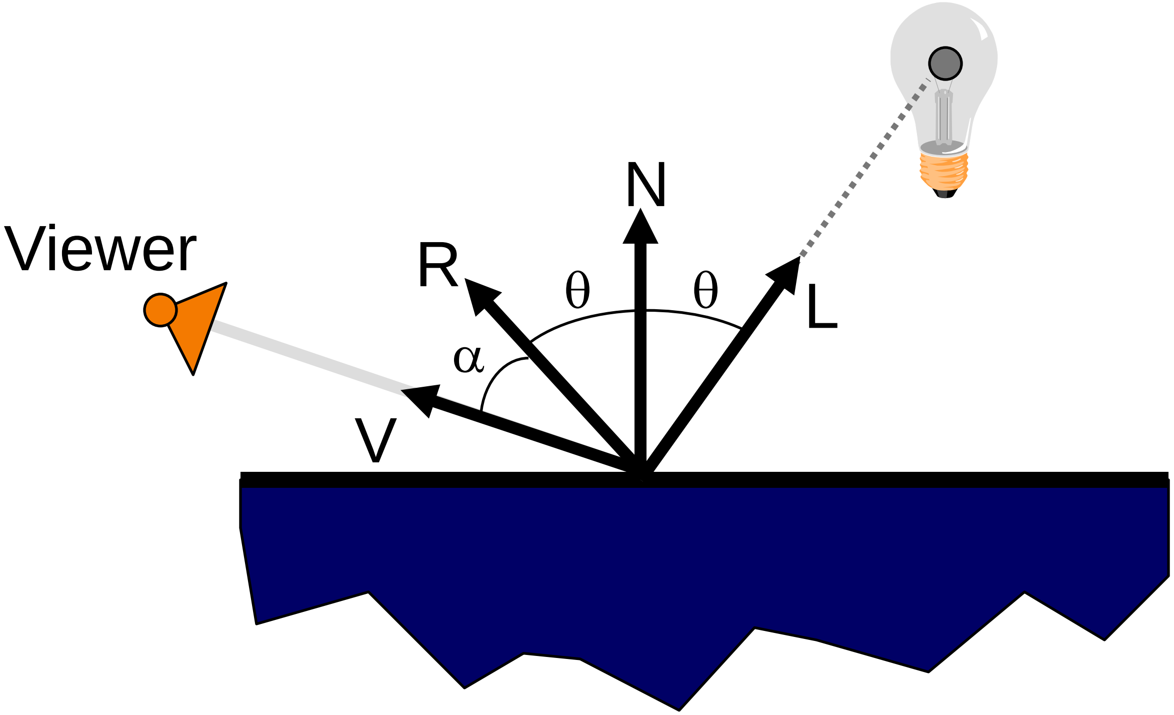
OpenGL Reflectance Model

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



Surface Illumination Calculation

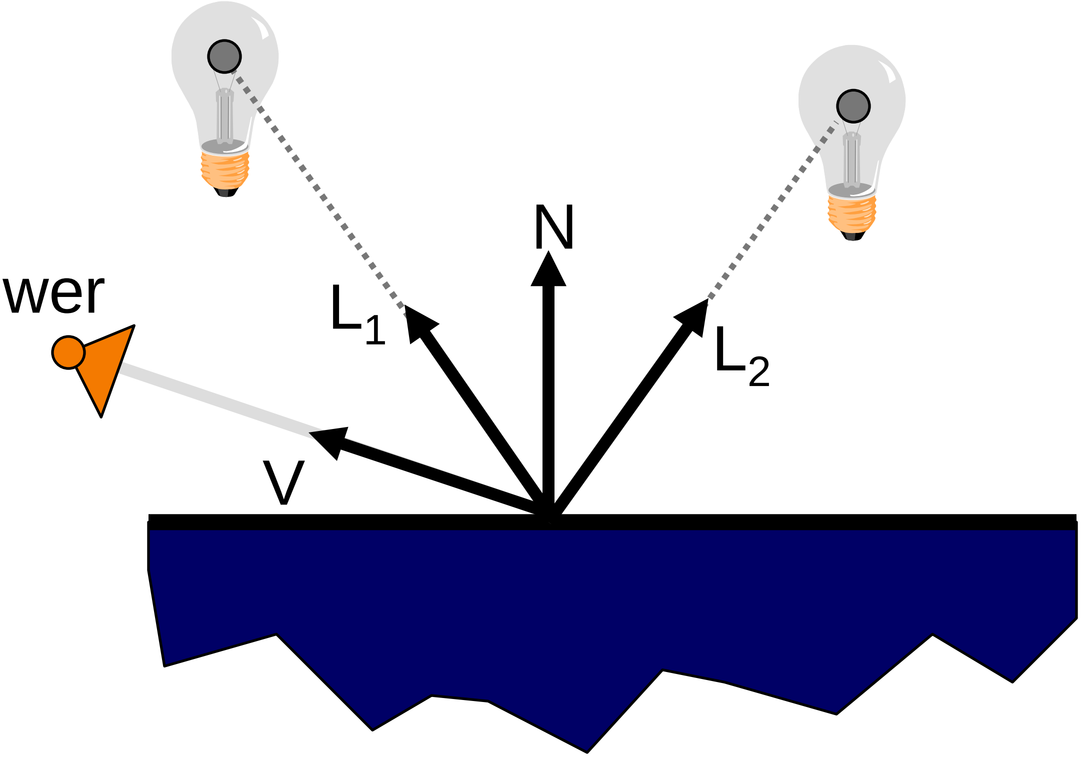
- Single light source:



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

Surface Illumination Calculation

- Multiple light sources:



$$I = I_E + K_A I_{AL} + \sum_i (K_D (N \bullet L_i) I_i + K_S (V \bullet R_i)^n I_i)$$

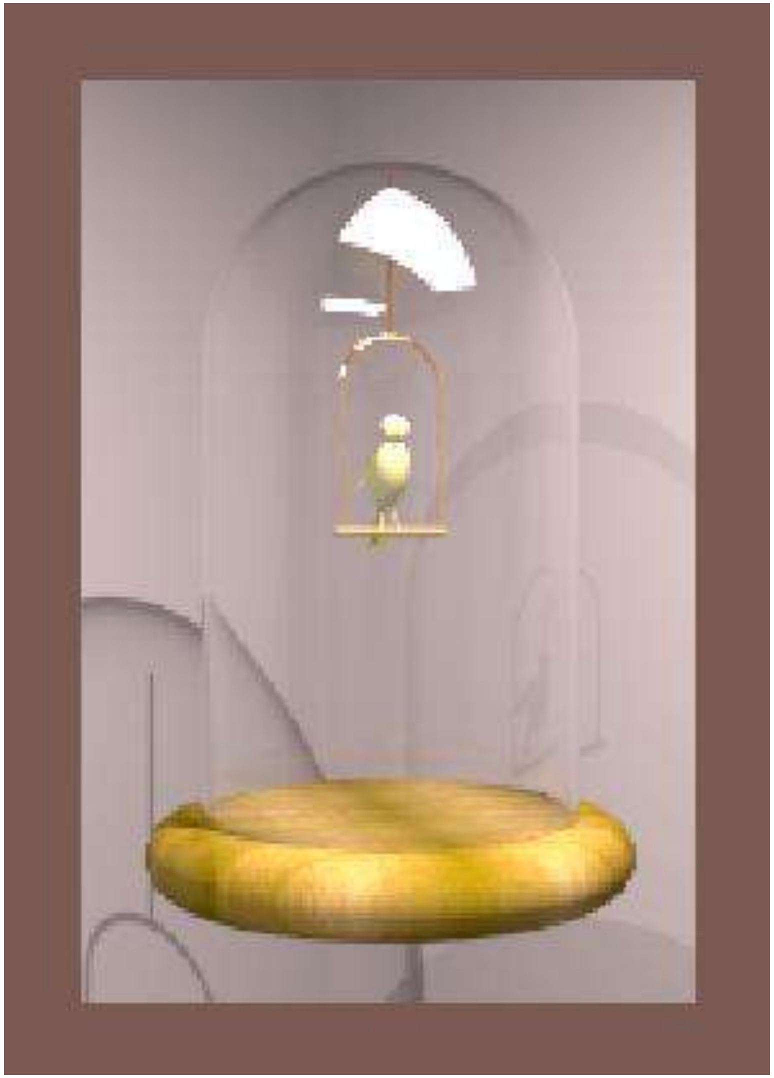
Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces

- Global illumination
  - Shadows
  - Transmissions
  - Inter-object reflections

Global Illumination

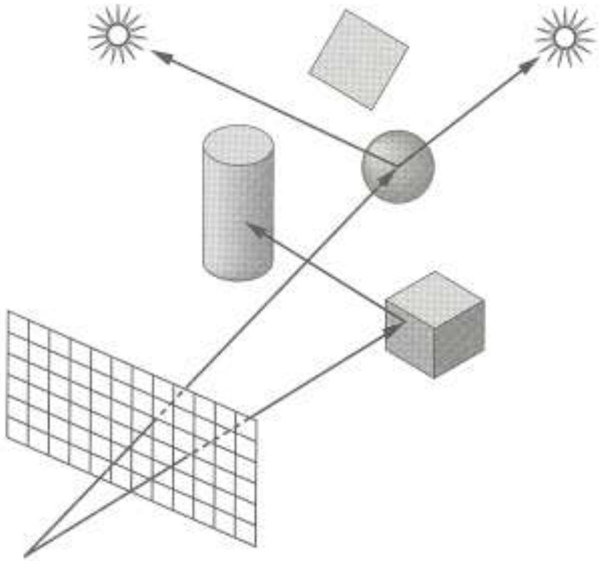
Global Illumination



Greg Larson

Shadows

- Shadow terms tell which light sources are blocked
  - Cast ray towards each light source L_i
  - S_i = 0 if ray is blocked, S_i = 1 otherwise



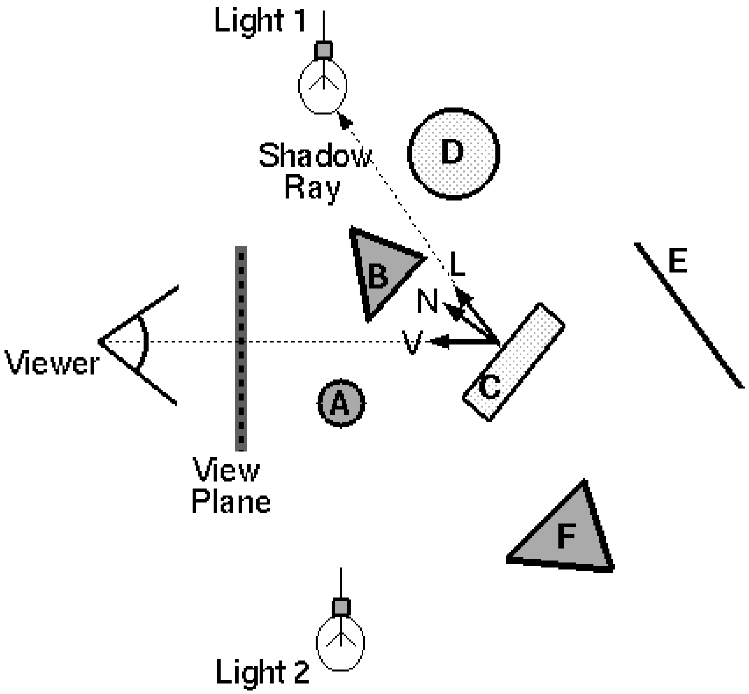
Shadow Term

$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) \underset{\uparrow}{S_L} I_L$$

Angel Figure 6.44

Ray Casting

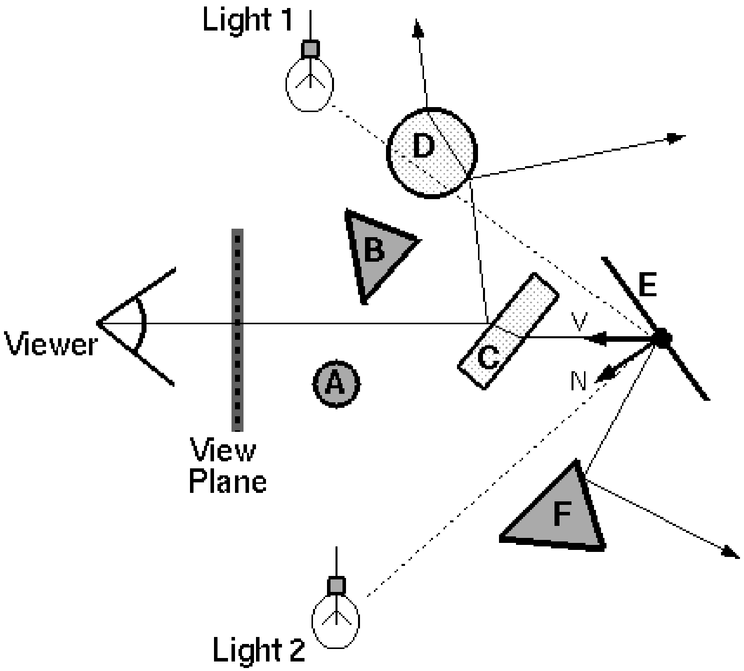
- Trace primary rays from camera
    - Direct illumination from unblocked lights only



$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L$$

Recursive Ray Tracing

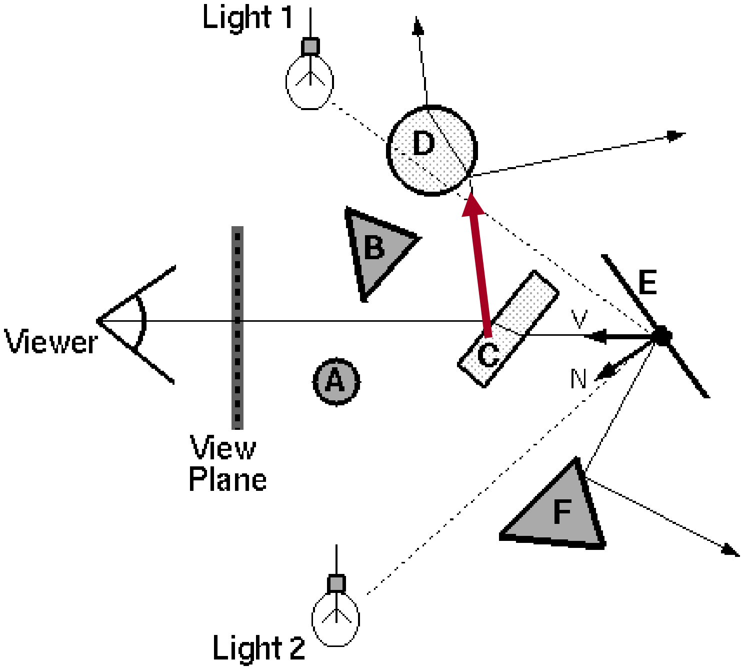
- Also trace secondary rays from hit surfaces
  - Global illumination from mirror reflection and transparency



$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_S I_R + K_T I_T$$

Mirror reflections

- Trace secondary ray in direction of mirror reflection
  - Evaluate radiance along secondary ray and include it into illumination model

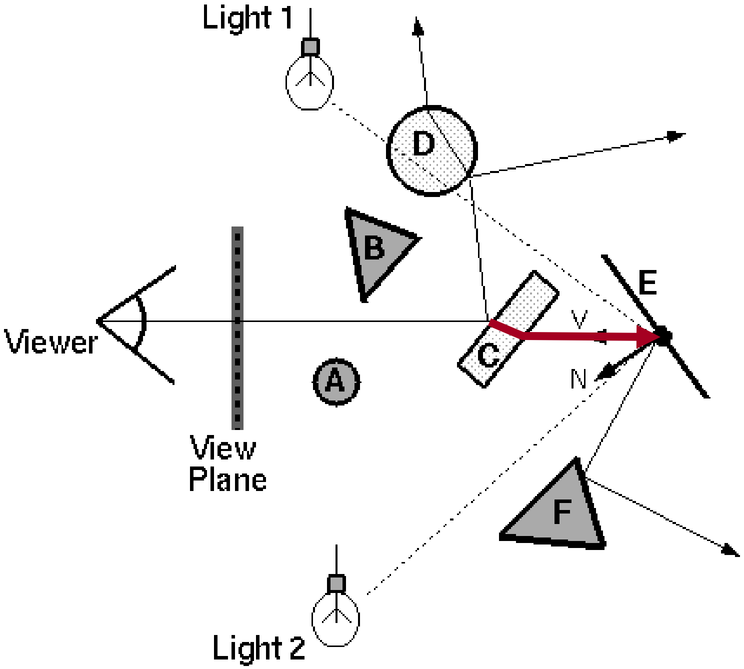


Radiance for mirror reflection ray

$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_S \boldsymbol{I_R} + K_T I_T$$

Transparency

- Trace secondary ray in direction of refraction
  - Evaluate radiance along secondary ray and include it into illumination model

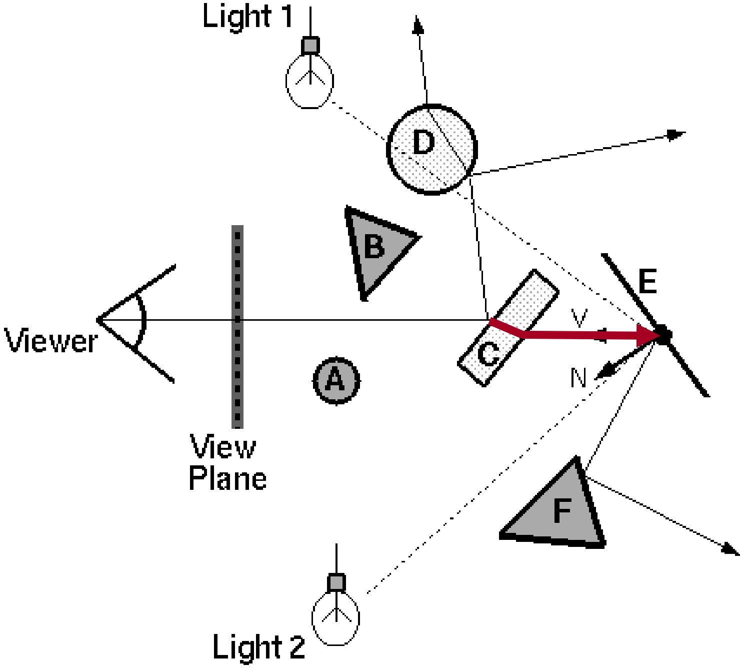


Radiance for refraction ray

$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_S I_R + K_T \boldsymbol{I_T}$$

Transparency

- Transparency coefficient is fraction transmitted
  - K_T = 1 if object is translucent, K_T = 0 if object is opaque
  - 0 < K_T < 1if object is semi-translucent

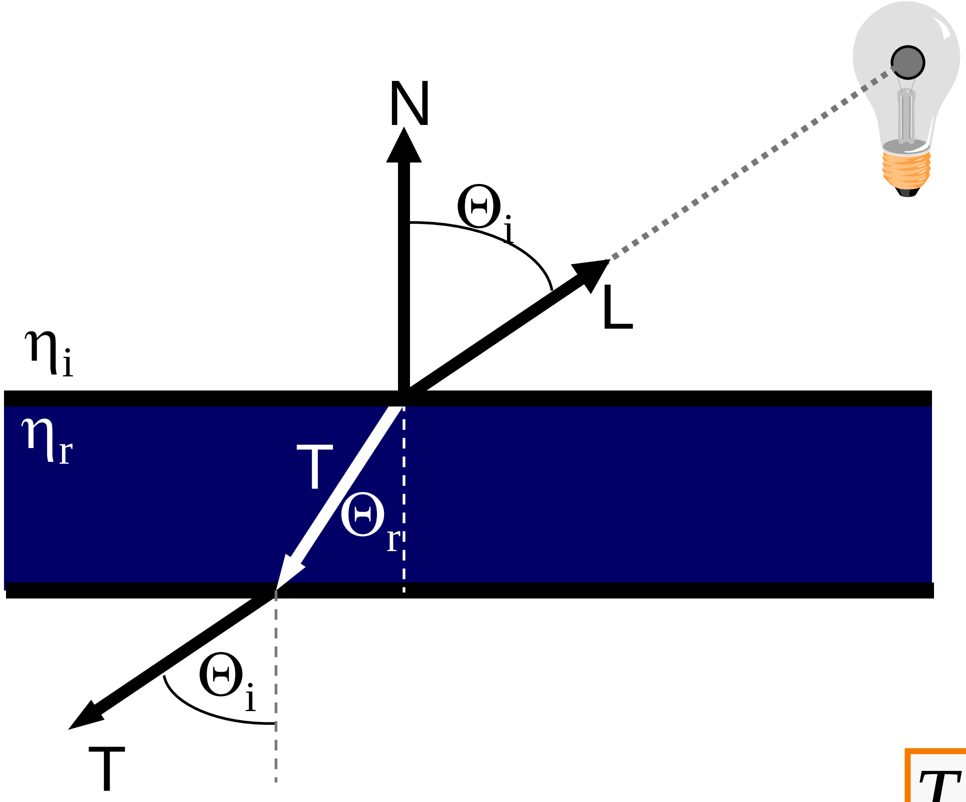


Transparency Coefficient

$$I = I_E + K_A I_A + \sum_L (K_D(N \bullet L) + K_S(V \bullet R)^n)S_L I_L + K_S I_R + \mathbf{K_T} I_T$$

Refractive Transparency

- For thin surfaces, can ignore change in direction
  - Assume light travels straight through surface

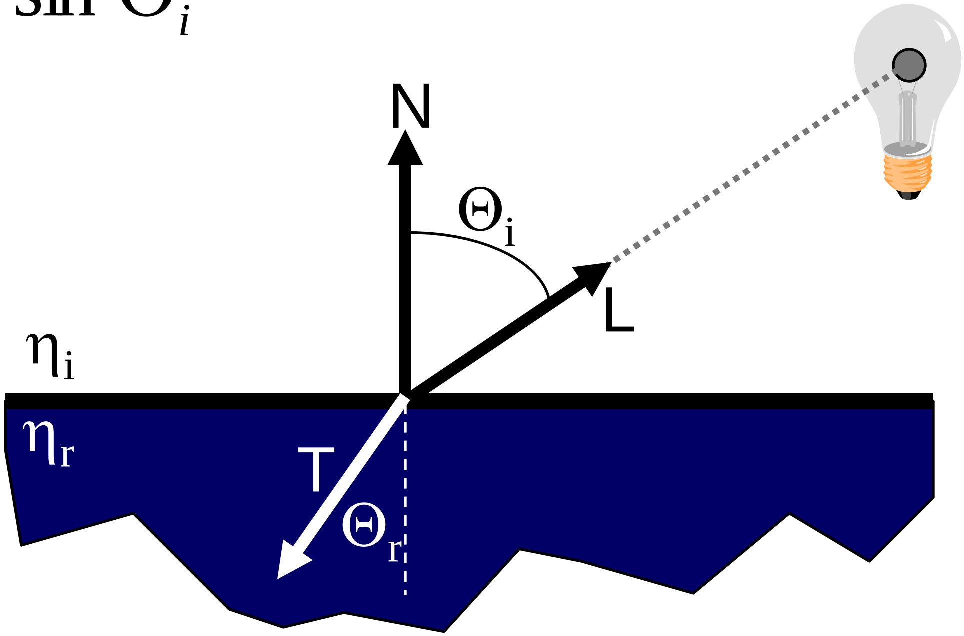


$$\boxed{T \cong -L}$$

Refractive Transparency

For solid objects, apply Snell's law:

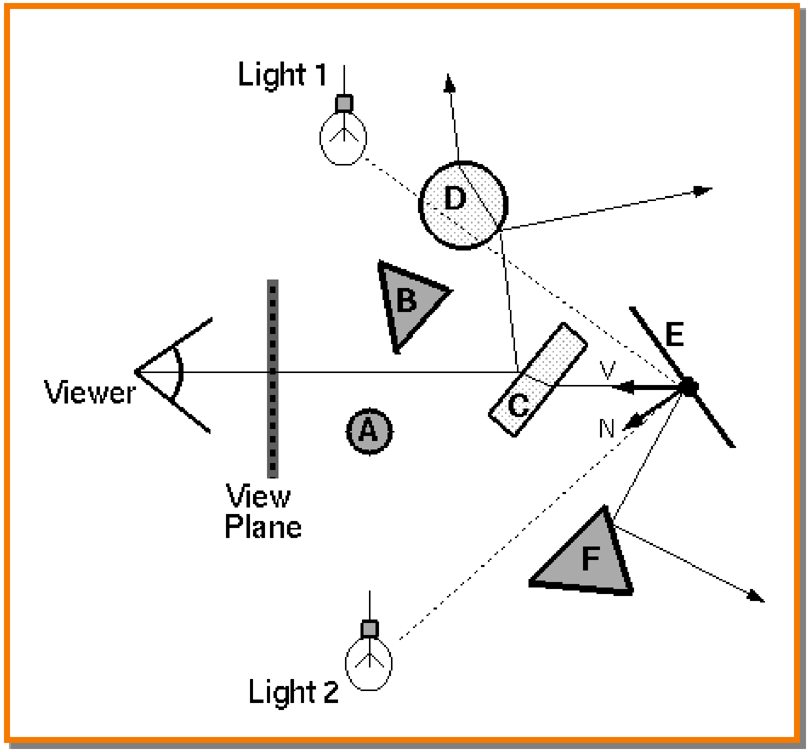
$$\eta_r \sin \Theta_r = \eta_i \sin \Theta_i$$



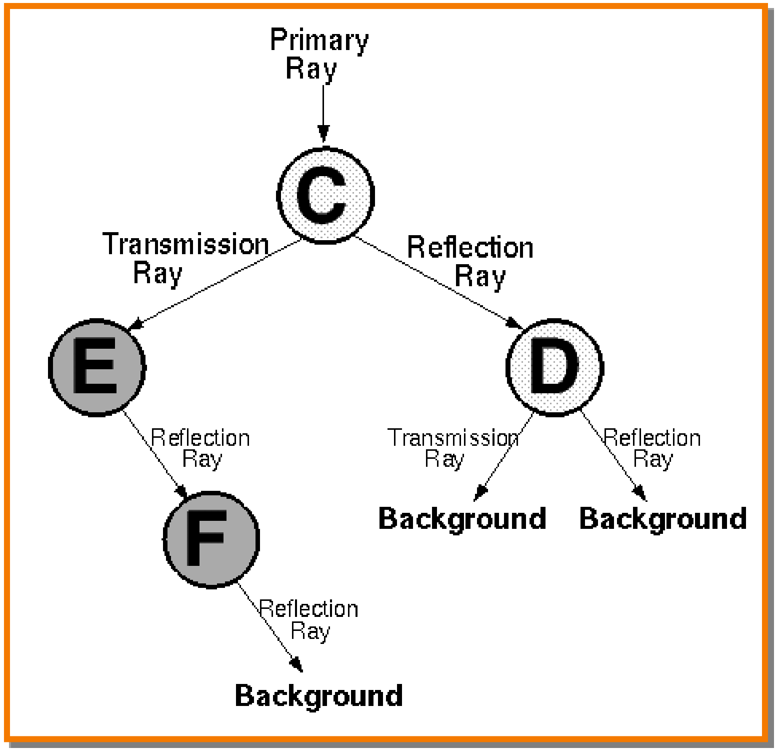
$$T = (\frac{\eta_i}{\eta_r} \cos \Theta_i - \cos \Theta_r) N - \frac{\eta_i}{\eta_r} L$$

Recursive Ray Tracing

- Ray tree represents illumination computation



Ray traced through scene

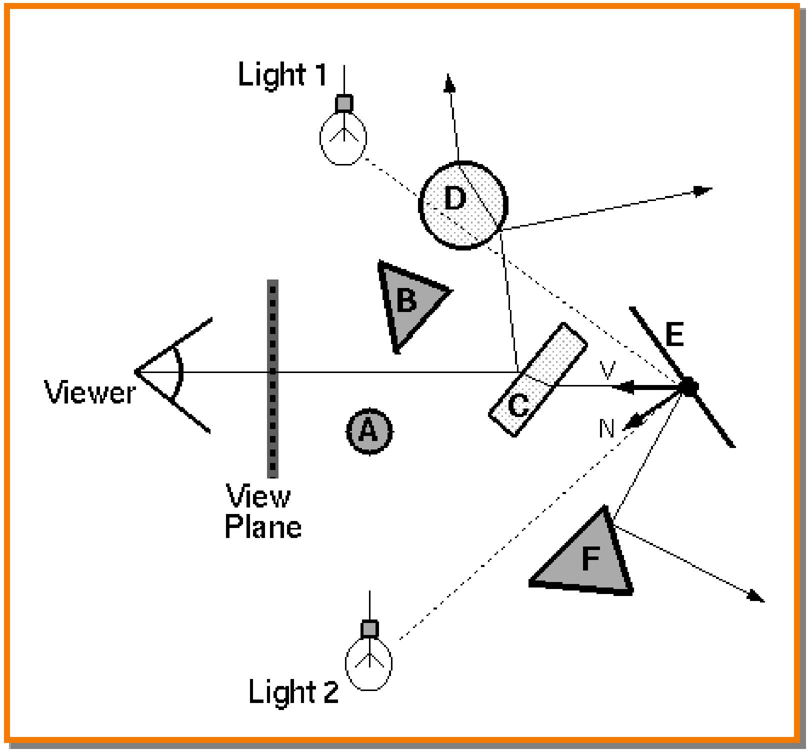


Ray tree

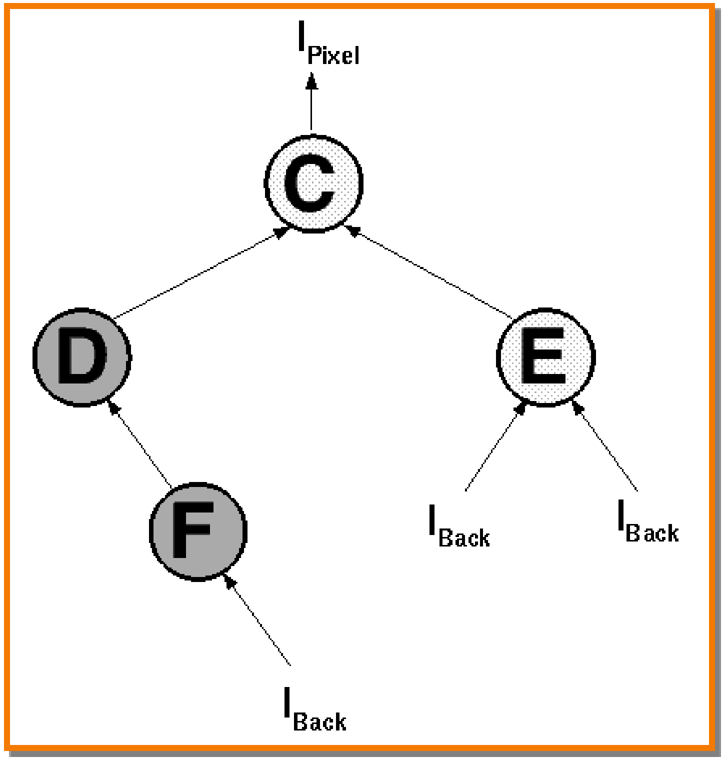
$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_S I_R + K_T I_T$$

Recursive Ray Tracing

- Ray tree represents illumination computation



Ray traced through scene



Ray tree

$$I = I_E + K_A I_A + \sum_L (K_D(N \bullet L) + K_S(V \bullet R)^n) S_L I_L + K_S I_R + K_T I_T$$

Recursive Ray Tracing

- GetColor calls RayTrace recursively

```
Image RayTrace(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```

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 later!

Illumination Terminology

- Radiant power [flux] (Φ)
  - 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^2sr)
    - 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^2)
    - e.g.: light hitting a surface along a
- Radiosity (B)
  - Exitant flux density from a locally planar area (in Watts/ m^2)