M&Ms: Freshmen Experience Ray Tracing

Michael Kazhdan

Outline

- Ray-Tracing
  - Overview
  - Direct Illumination
  - Global Illumination

Geri's Game , *Pixar*

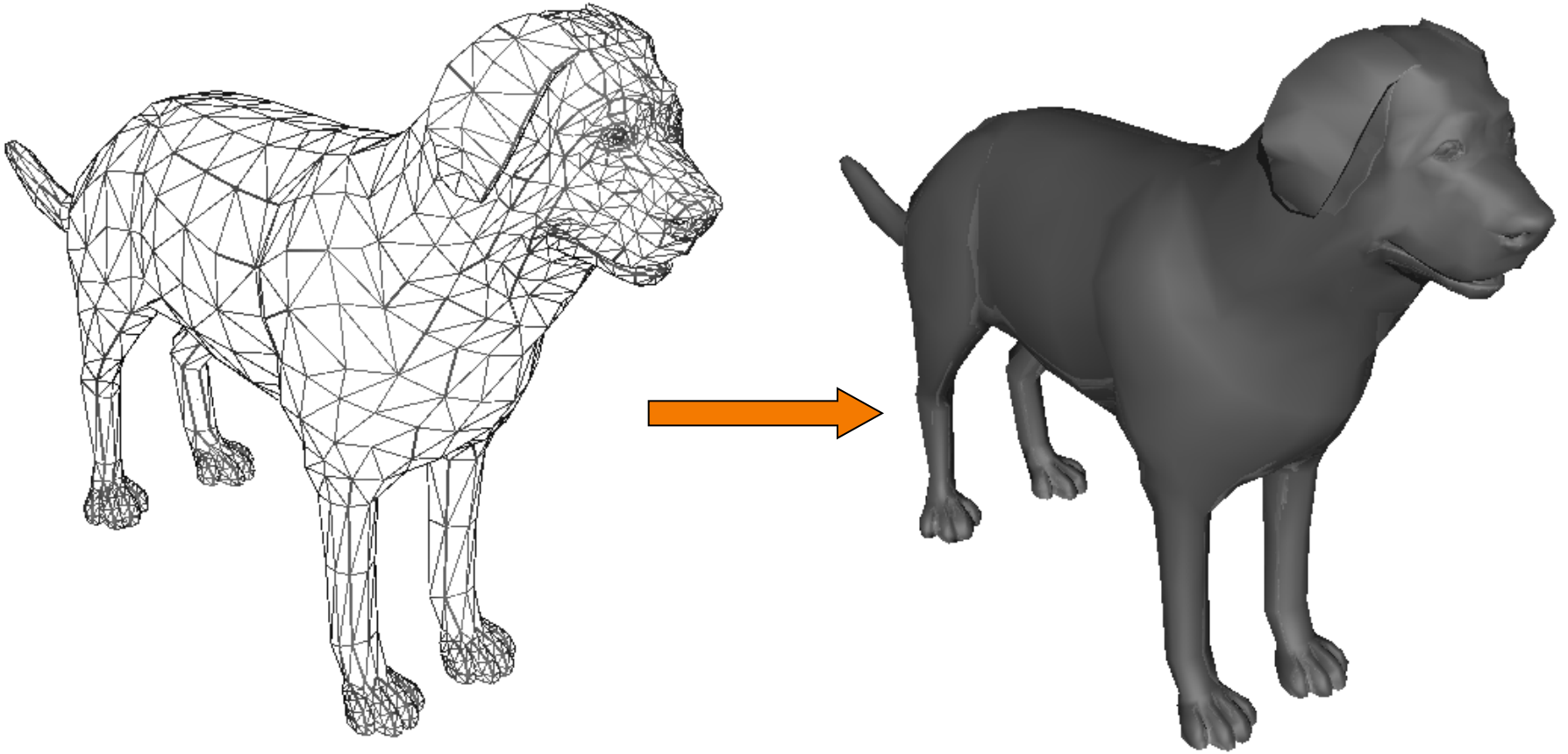
3D Model Representation

In graphics, we often represent the surface of a 3D shape by a set of triangles.

Work by Jim Rygiel for "102 Dalmatians"

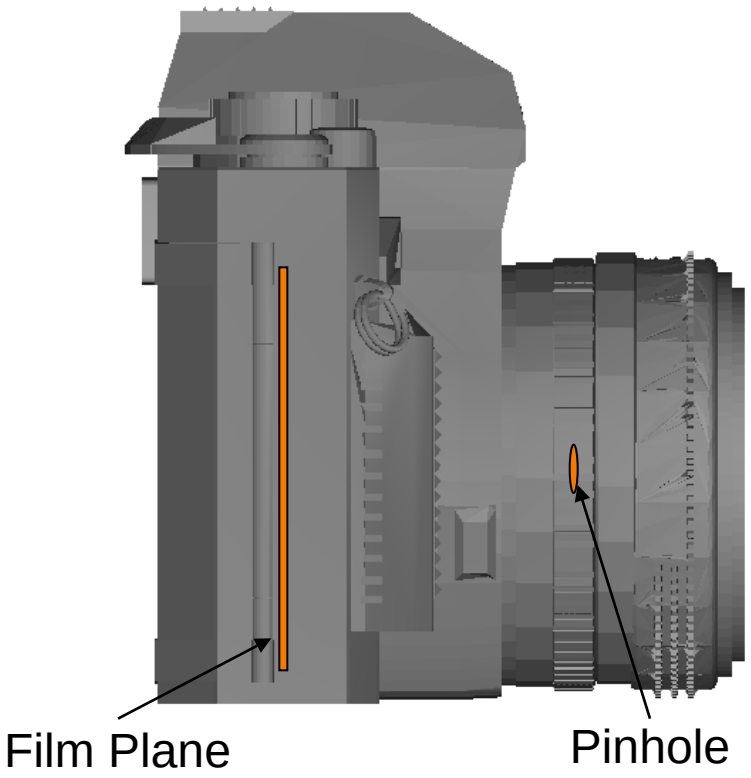
Ray-Tracing

The goal of ray-tracing is to take a collection of geometry representing a 3D scene and render a detailed and credible image.



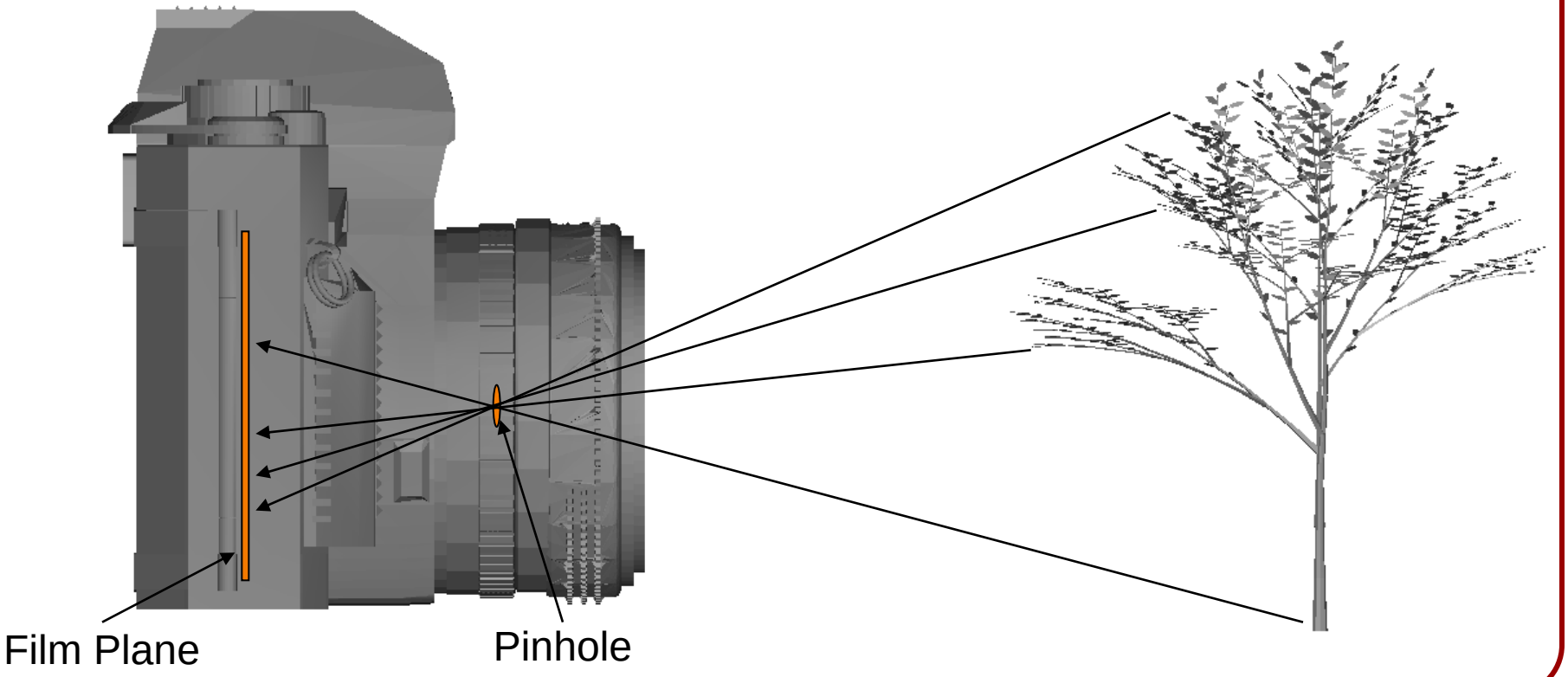
Traditional Pinhole Camera

- The film sits behind the pinhole of the camera.



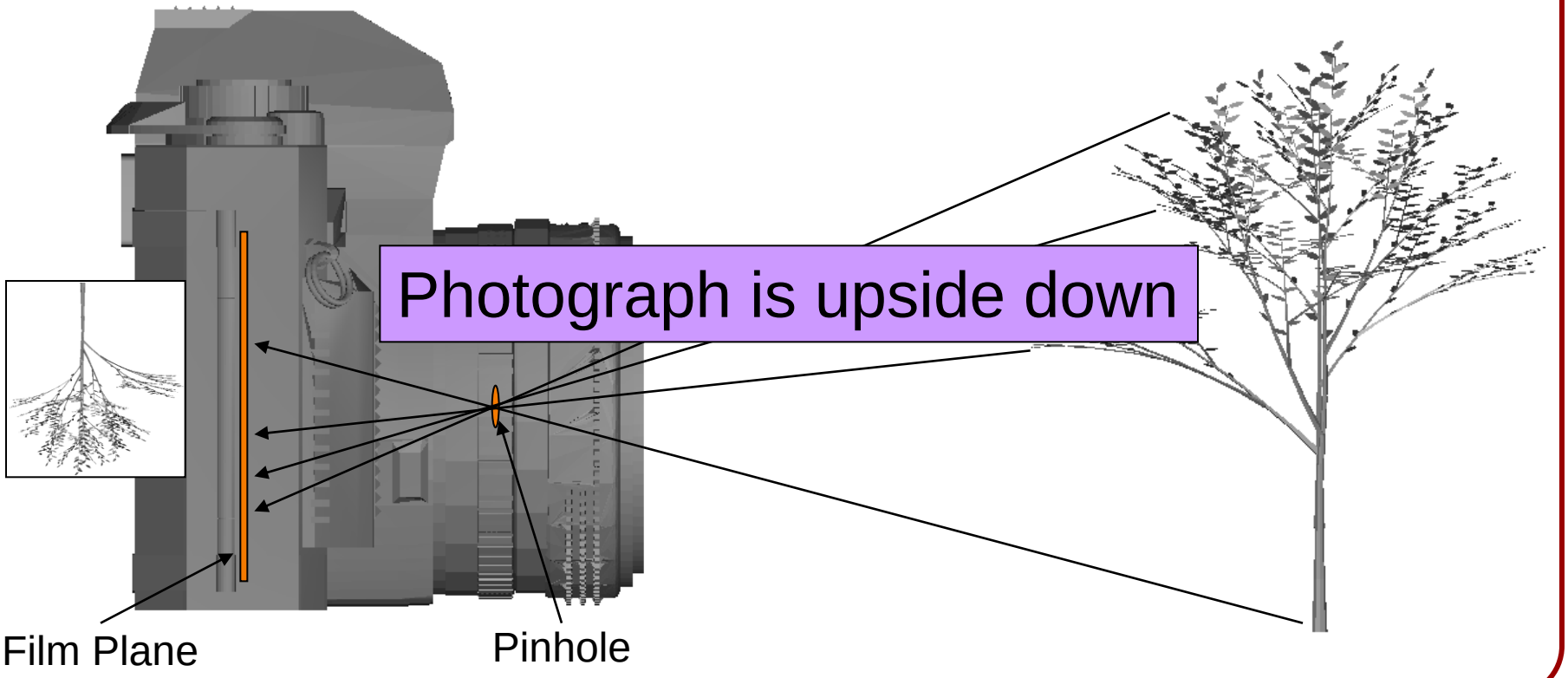
Traditional Pinhole Camera

- The film sits behind the pinhole of the camera.
- Rays come in from the outside, pass through the pinhole, and hit the film plane.



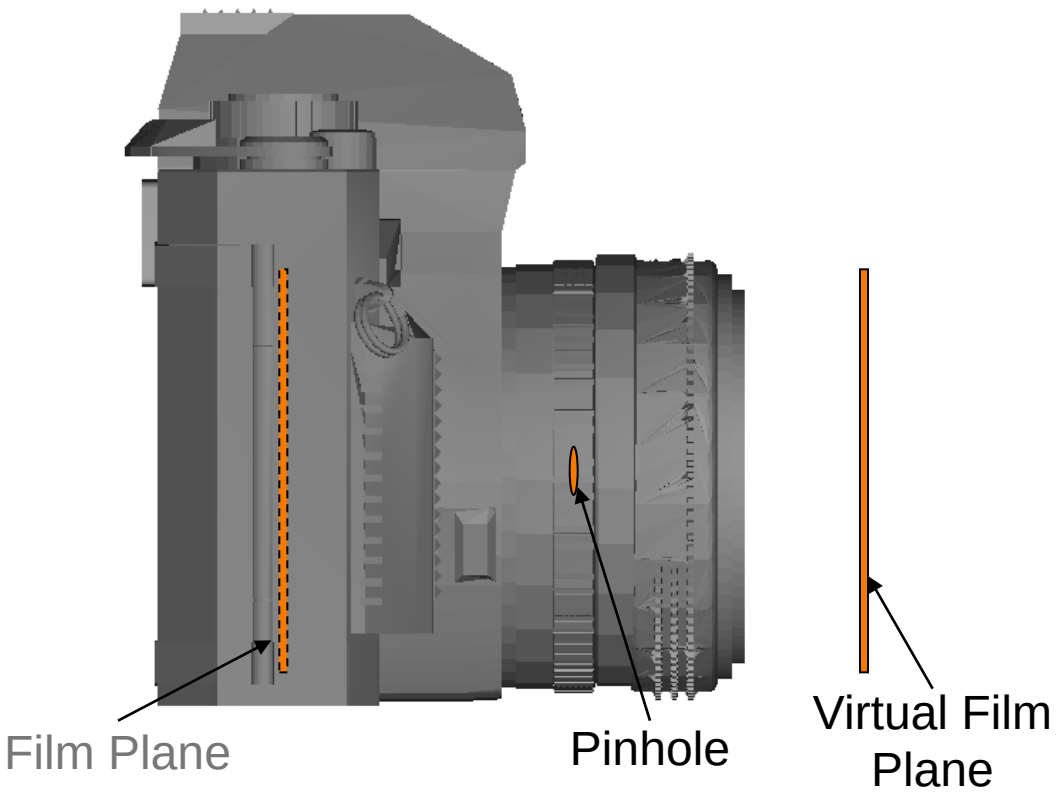
Traditional Pinhole Camera

- The film sits behind the pinhole of the camera.
- Rays come in from the outside, pass through the pinhole, and hit the film plane.



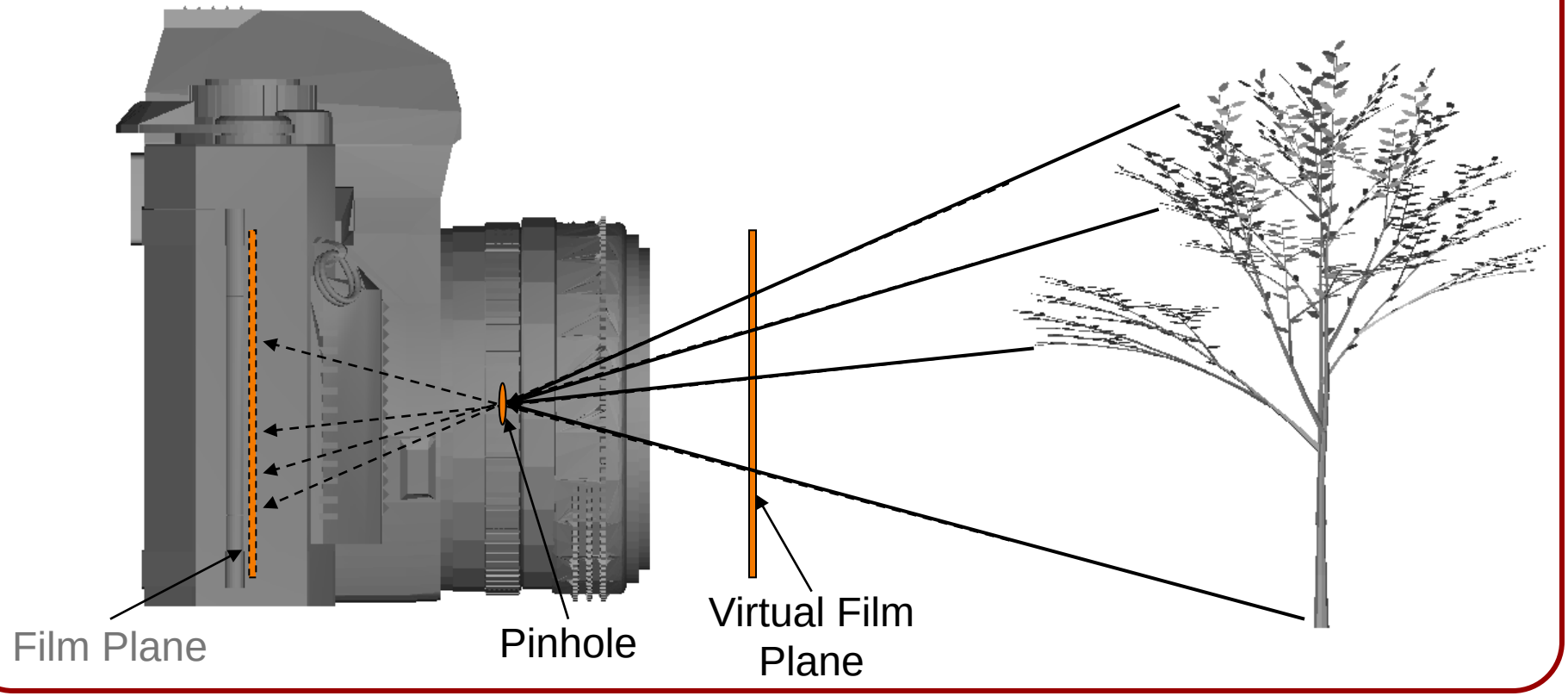
Virtual Camera

- The film sits in front of the pinhole of the camera.



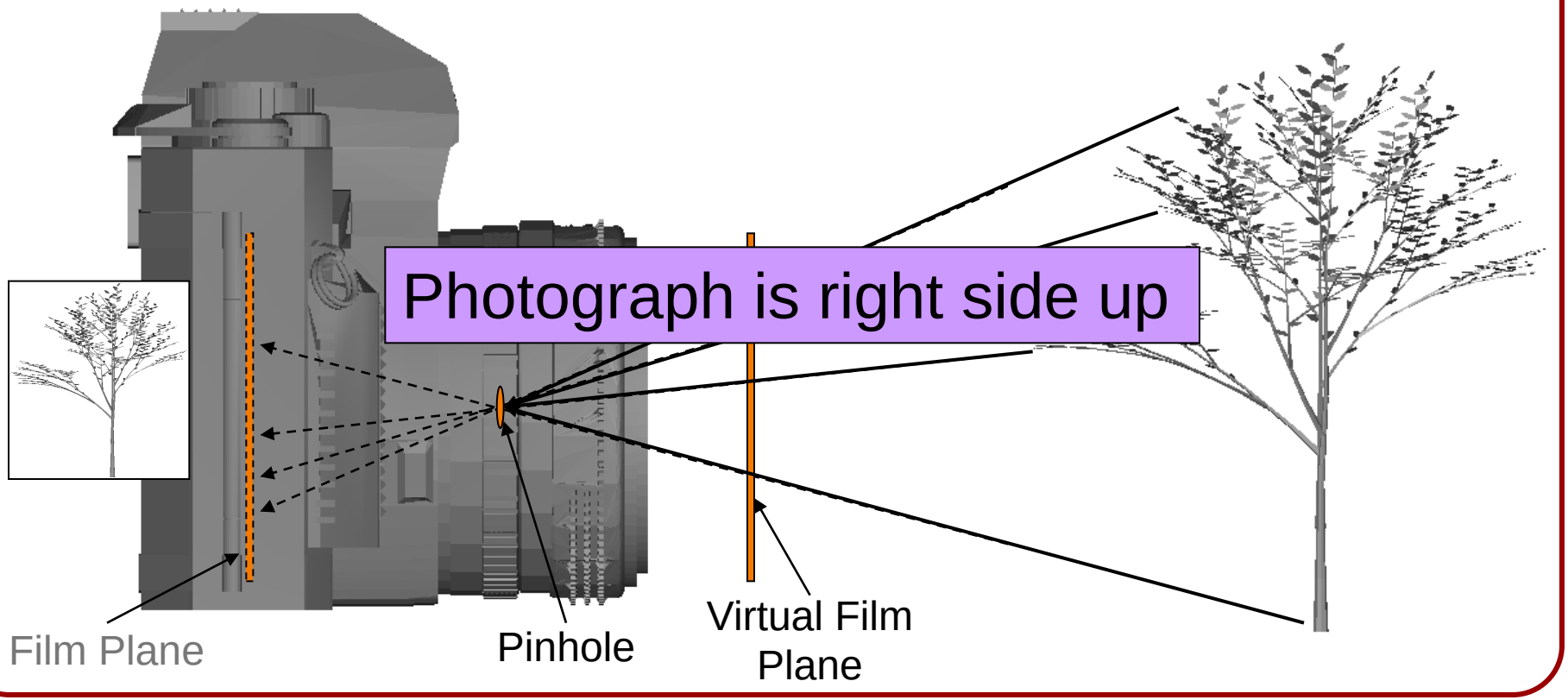
Virtual Camera

- The film sits in front of the pinhole of the camera.
- Rays come in from the outside, pass through the film plane, and hit the pinhole.



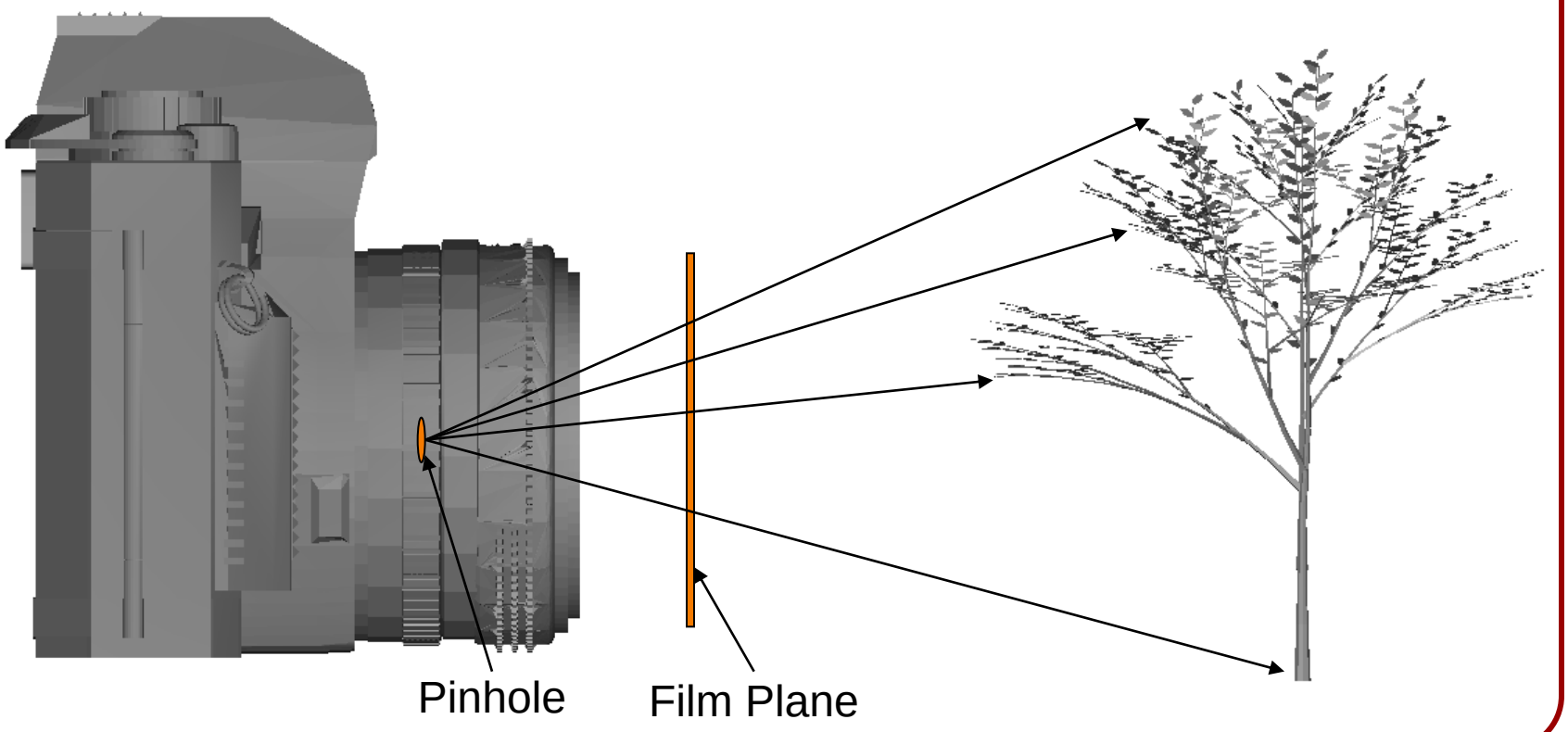
Virtual Camera

- The film sits in front of the pinhole of the camera.
- Rays come in from the outside, pass through the film plane, and hit the pinhole.



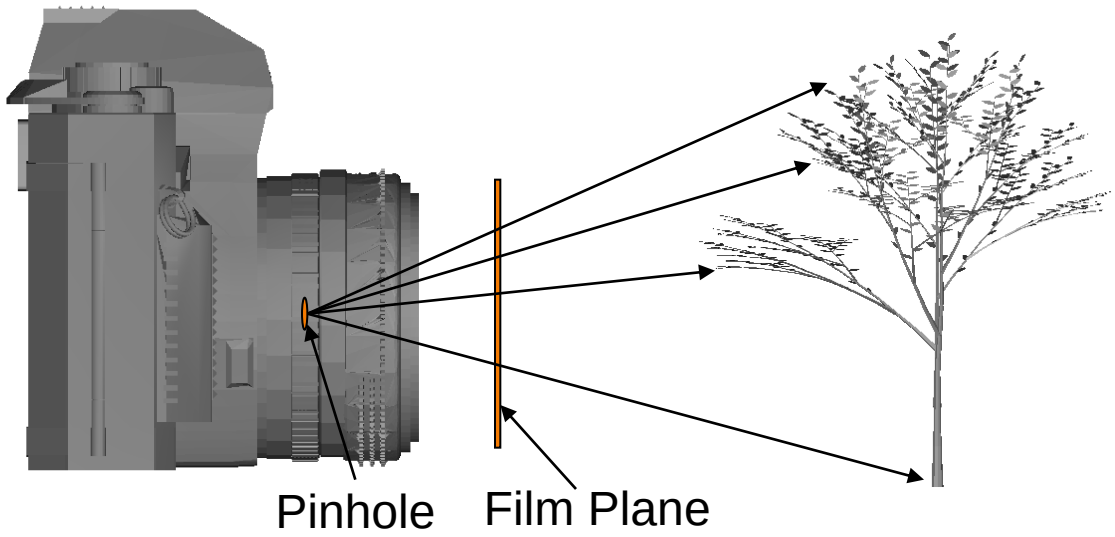
Ray Casting

- We invert the process of image generation by sending rays out from the pinhole.



Ray Casting

- We invert the process of image generation by sending rays out from the pinhole.
- For each pixel in the virtual view screen:
  - Compute the ray from the pinhole, through the pixel
  - Figure out what object in the scene is first hit by the ray
  - Compute the color of the object and assign it to the pixel.



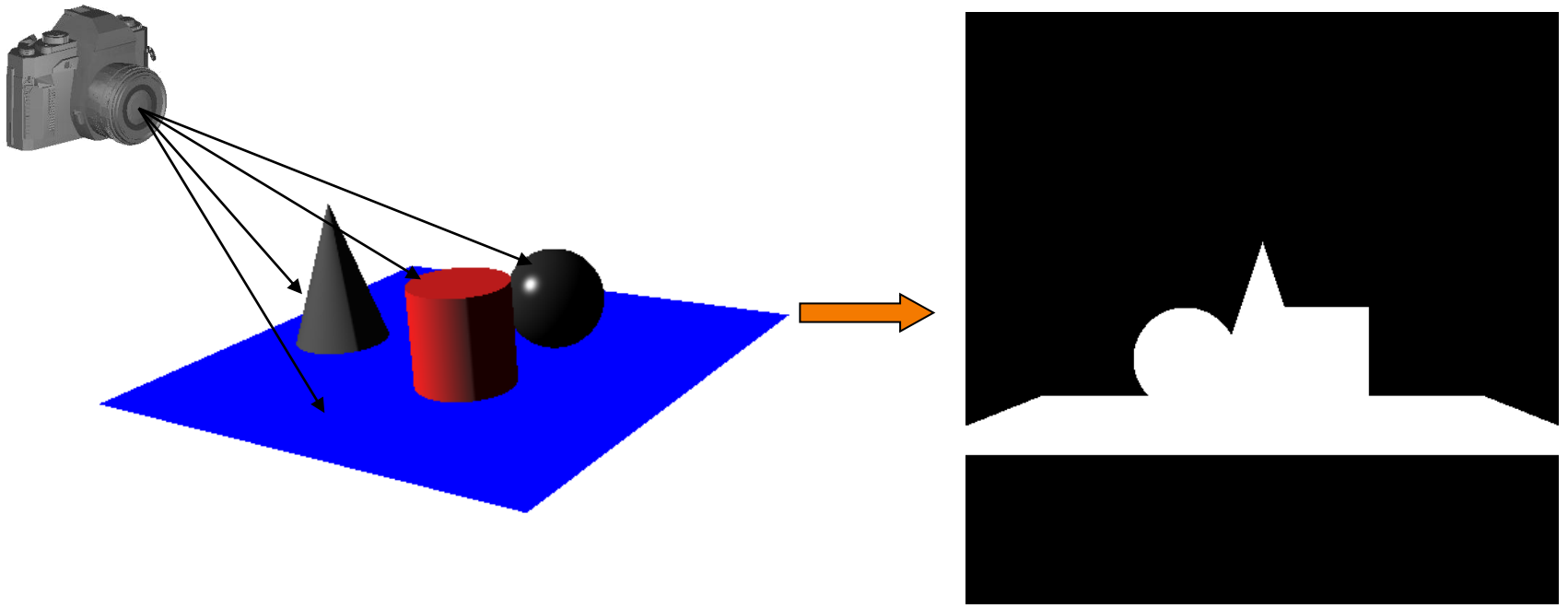
Ray Casting

- Simple implementation:

```
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(hit);
        }
    }
    return image;
}
```

Ray Casting

- If we ignore the color computation, we can quickly determine the silhouettes of the scene:

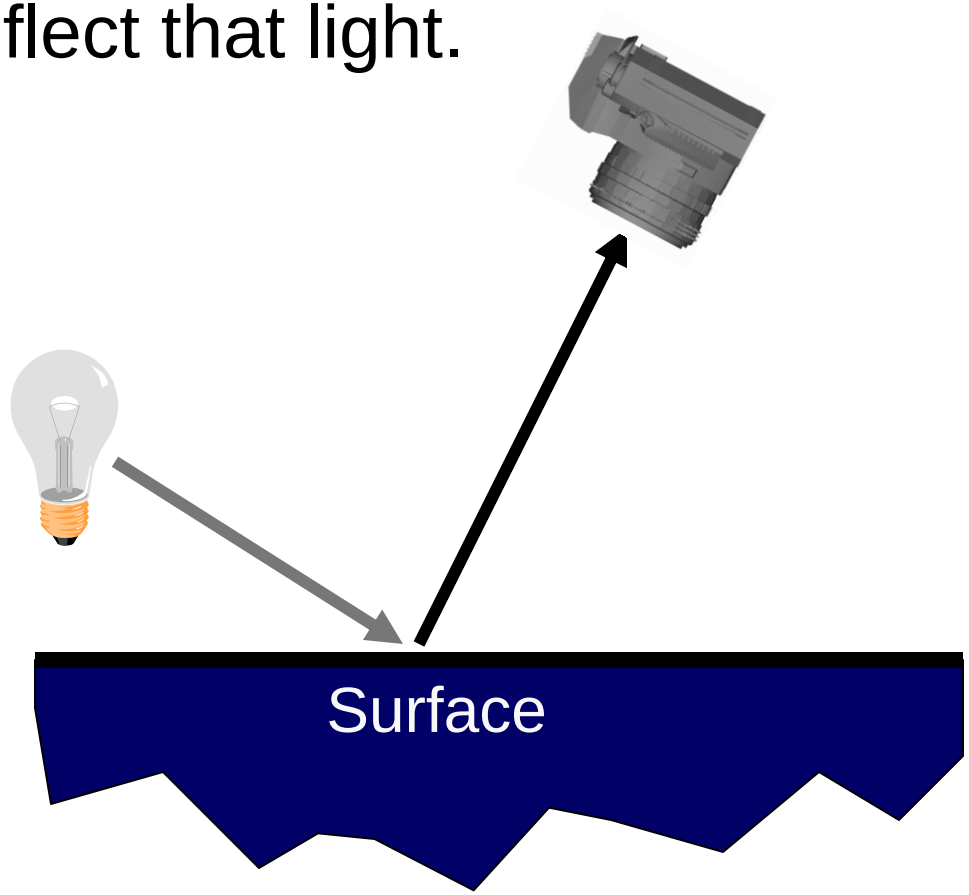


Outline

- Introduction
- Ray-Tracing
  - Overview
  - Direct Illumination
  - Global Illumination

Modeling Surface Reflectance

In practice, the color of the surface is determined by the lights in the scene and the ways in which the different surfaces reflect that light.

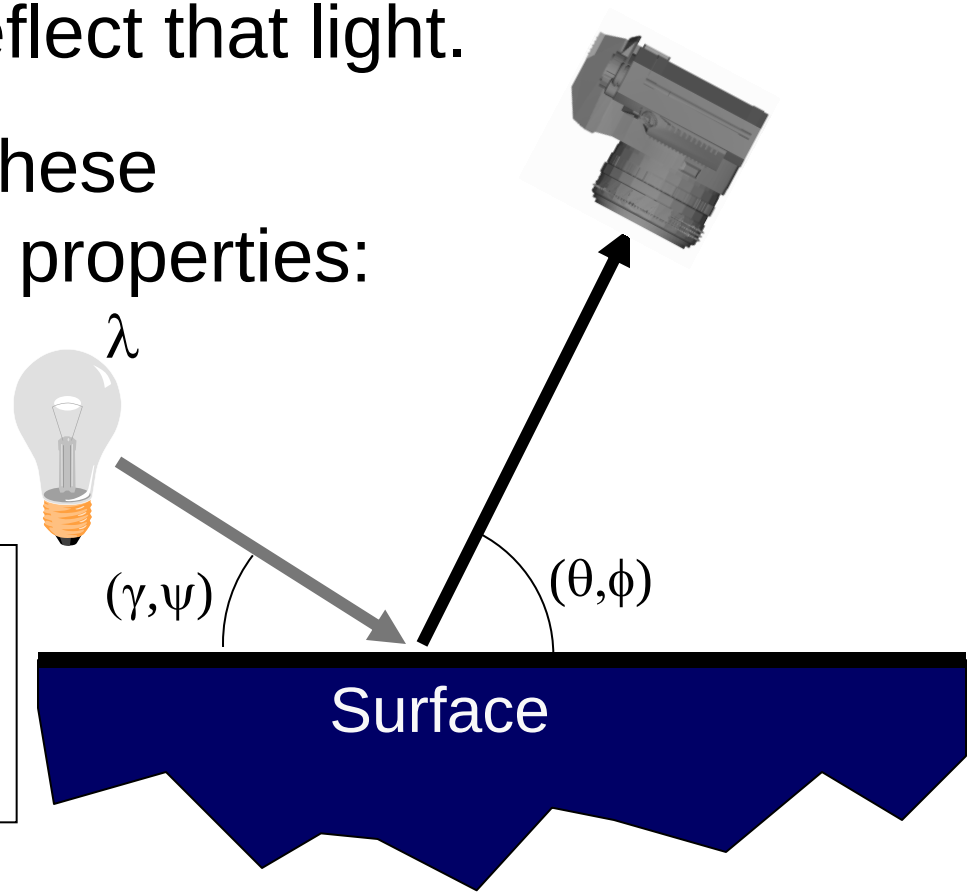


Modeling Surface Reflectance

In practice, the color of the surface is determined by the lights in the scene and the ways in which the different surfaces reflect that light.

We can try modeling these surface reflectance properties:

$$R_s(\theta, \phi, \lambda, \gamma, \psi)$$



R_s is the fraction of incident light:
- arriving from direction (γ,ψ)
- with wavelength λ
- leaving in direction (θ,ϕ)

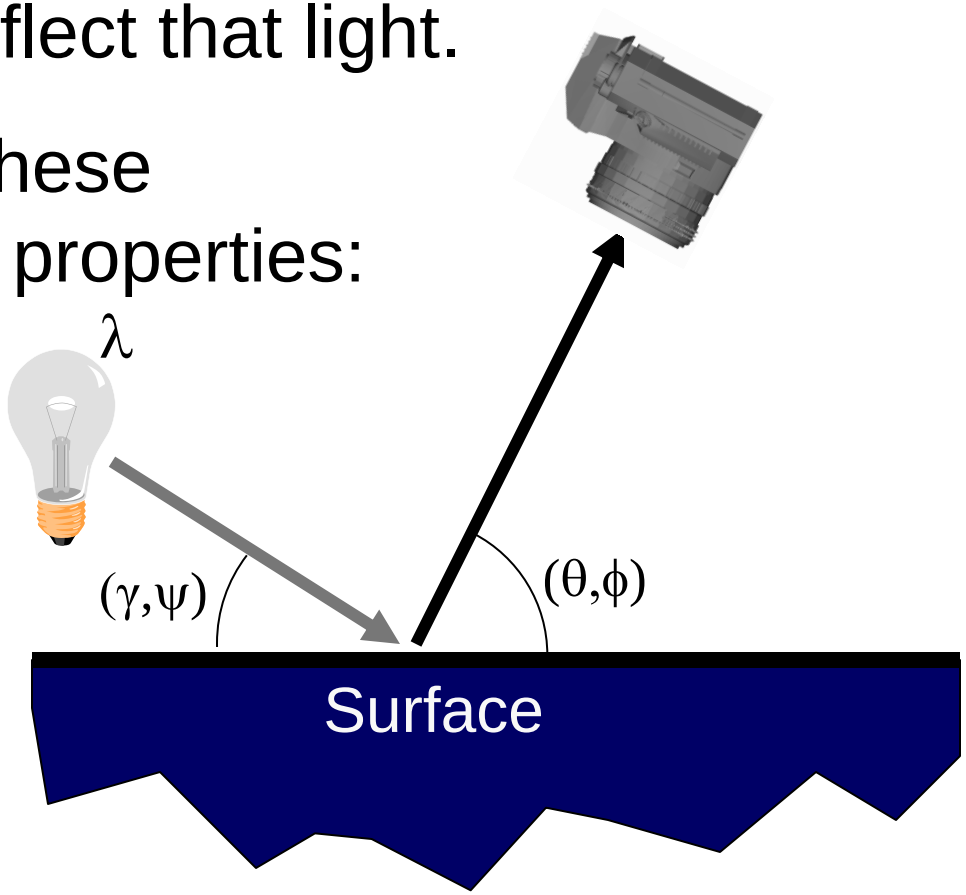
Modeling Surface Reflectance

In practice, the color of the surface is determined by the lights in the scene and the ways in which the different surfaces reflect that light.

We can try modeling these surface reflectance properties:

$$R_s(\theta, \phi, \lambda, \gamma, \psi)$$

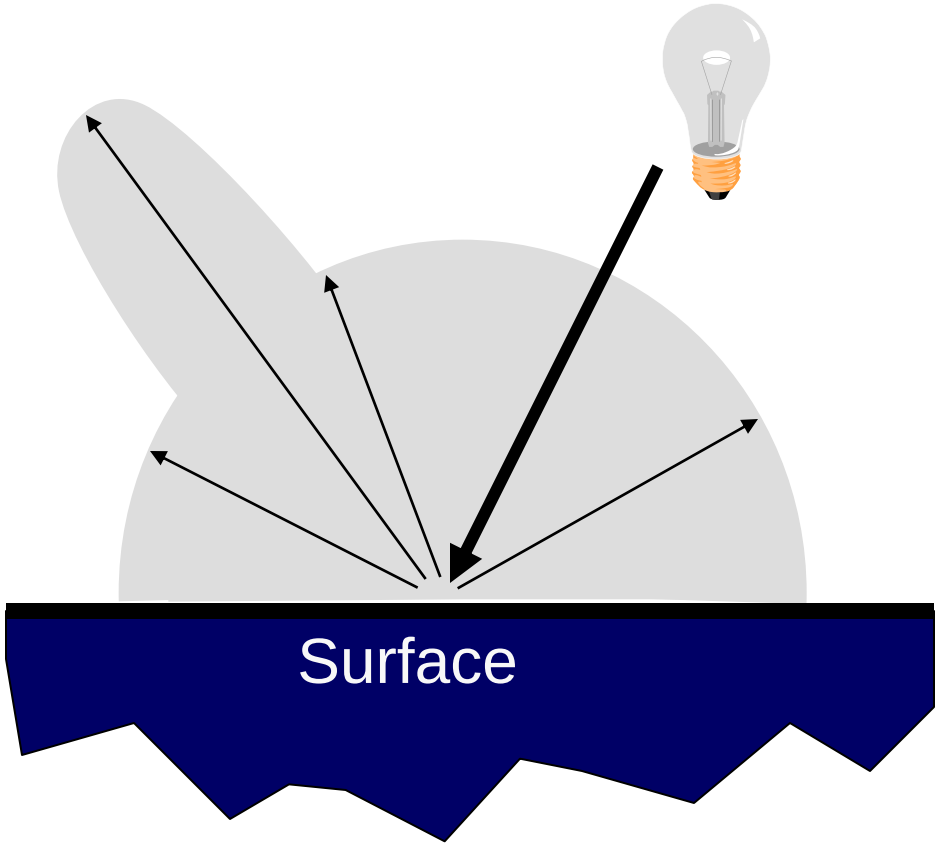
- Too much storage
- Difficult in practice



Simple Reflectance Model

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

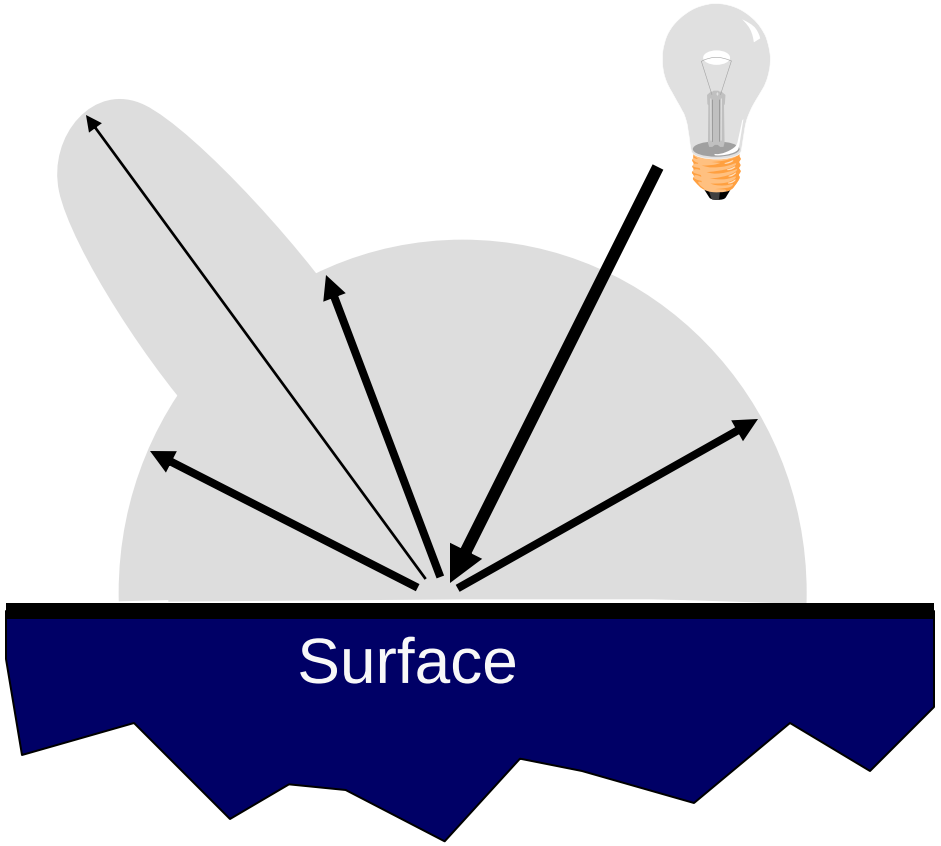
Based on model proposed by Phong



Simple Reflectance Model

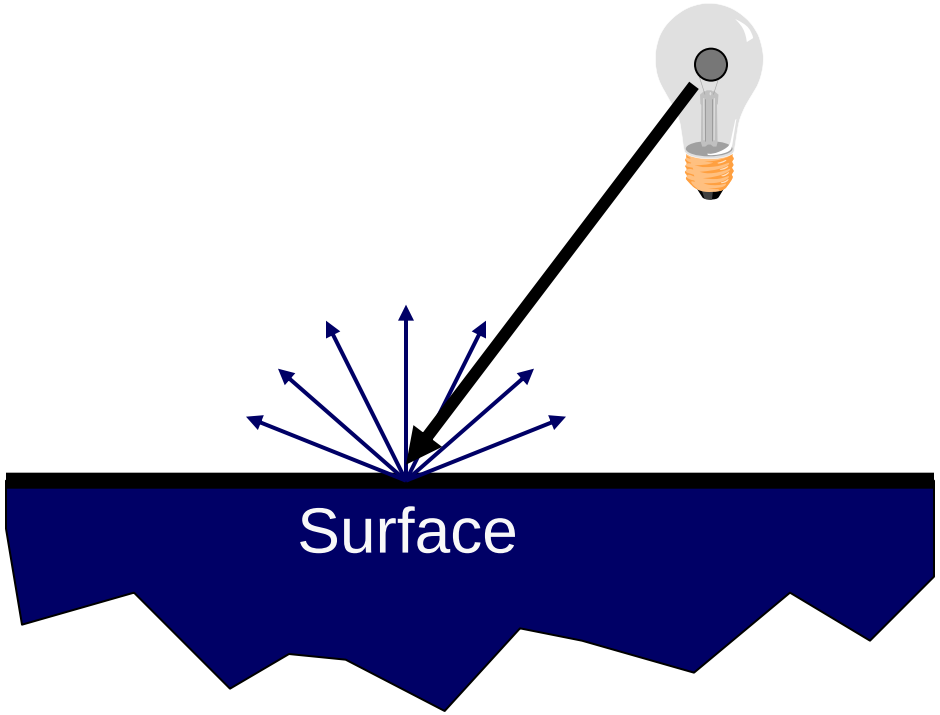
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”

Based on model proposed by Phong



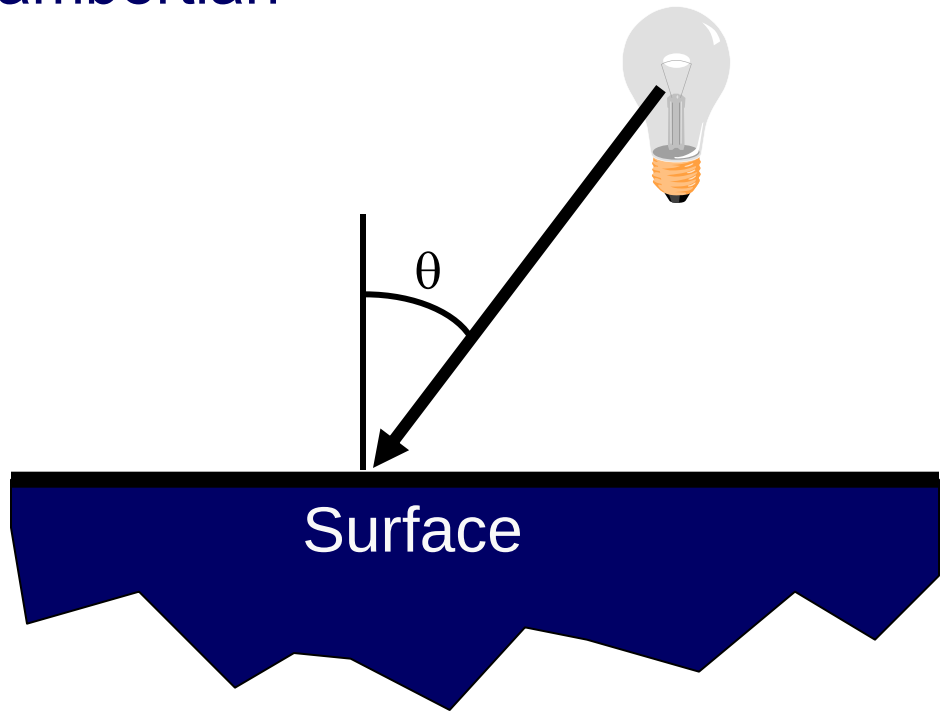
Diffuse Reflection

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



Diffuse Reflection

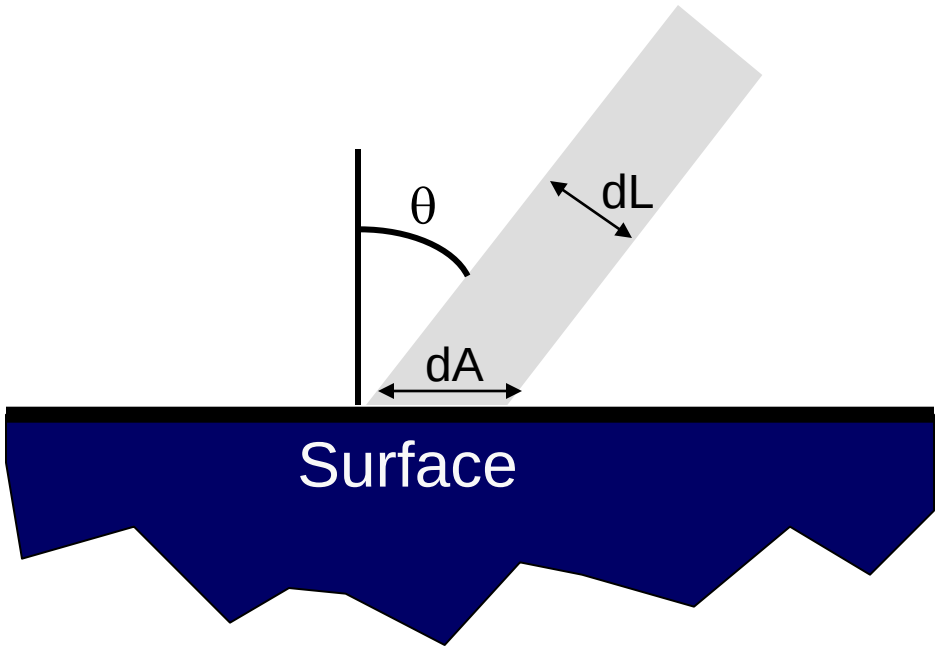
- How much light is reflected?
  - Depends on angle of incident light
  - aka “Lambertian”



Diffuse Reflection

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

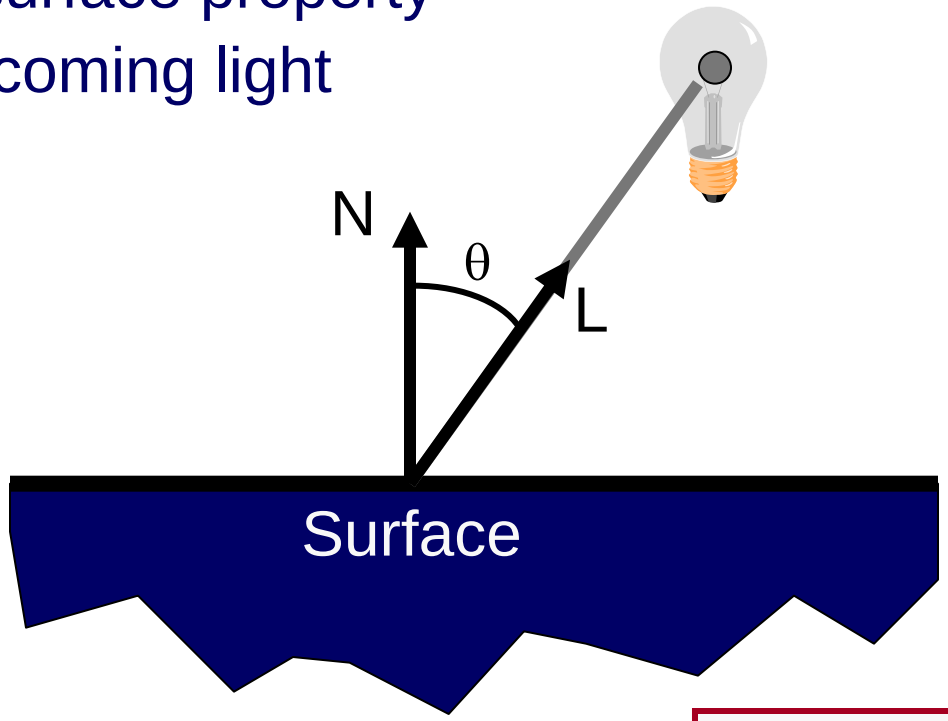
$$dL = dA\cos\Theta$$



Think of a flashlight!

Diffuse Reflection

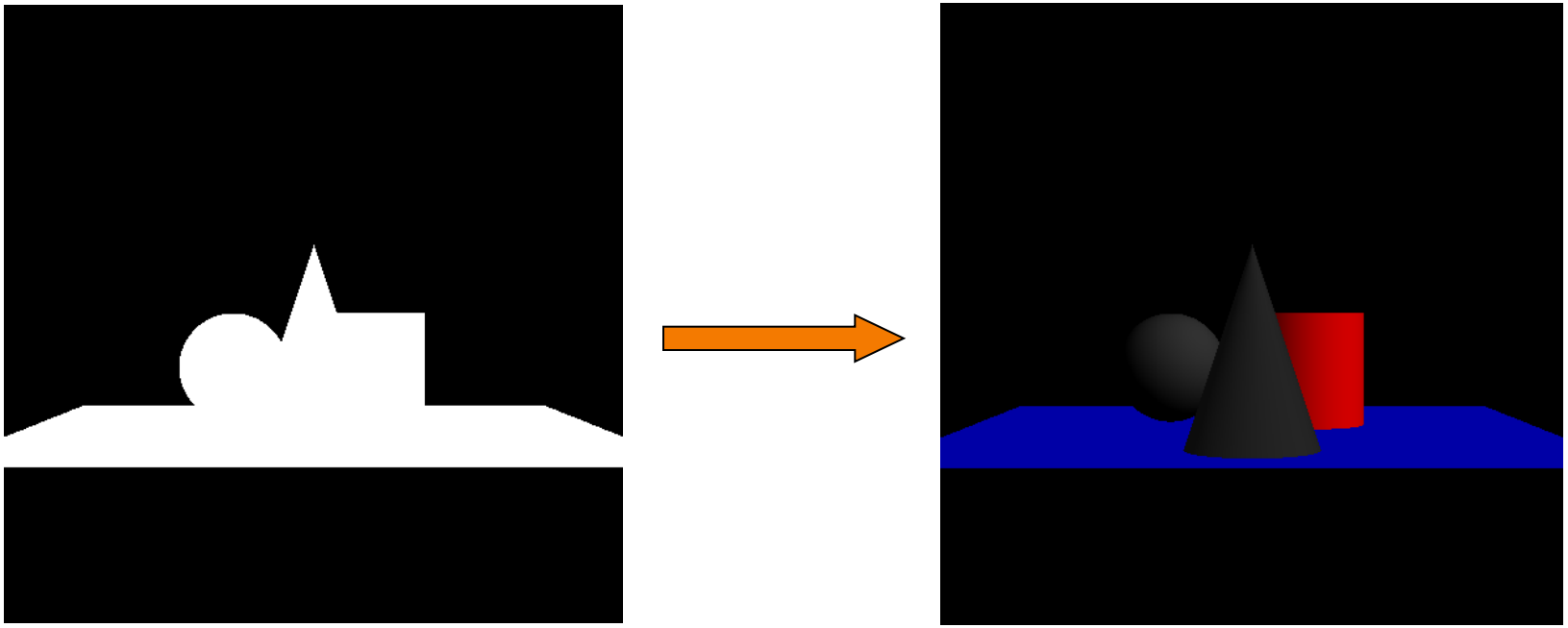
- Lambertian model
  - cosine law (dot product)
  - K_D is surface property
  - I_L is incoming light



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

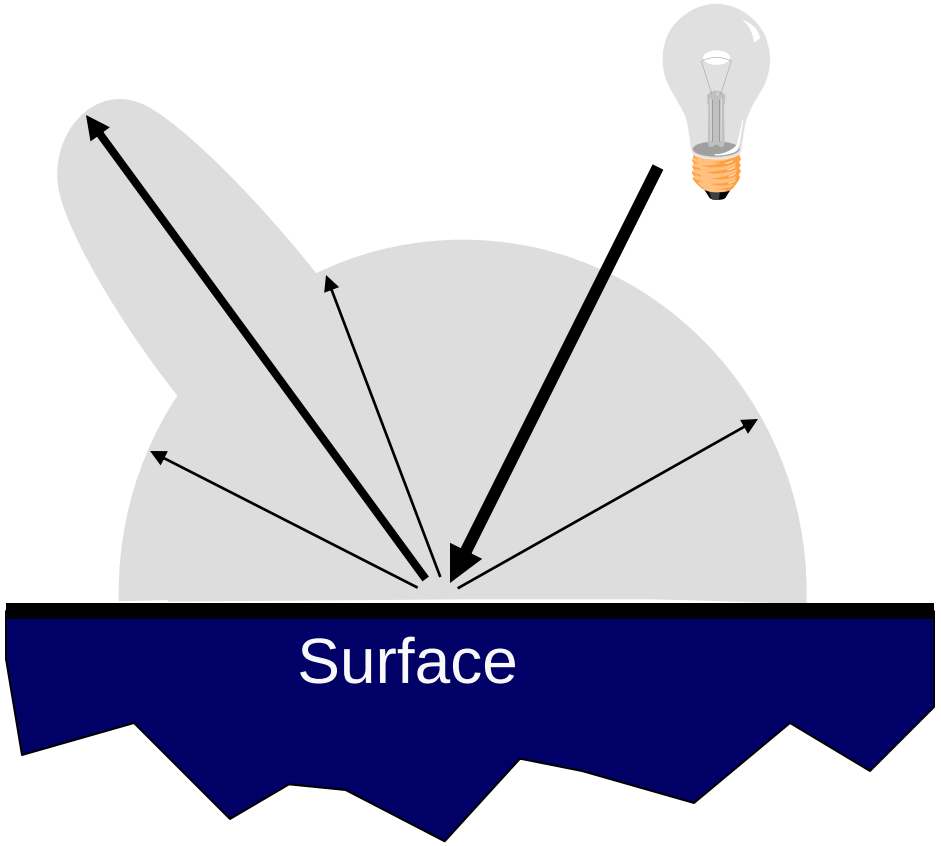
Diffuse Reflection

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



Simple Reflectance Model

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

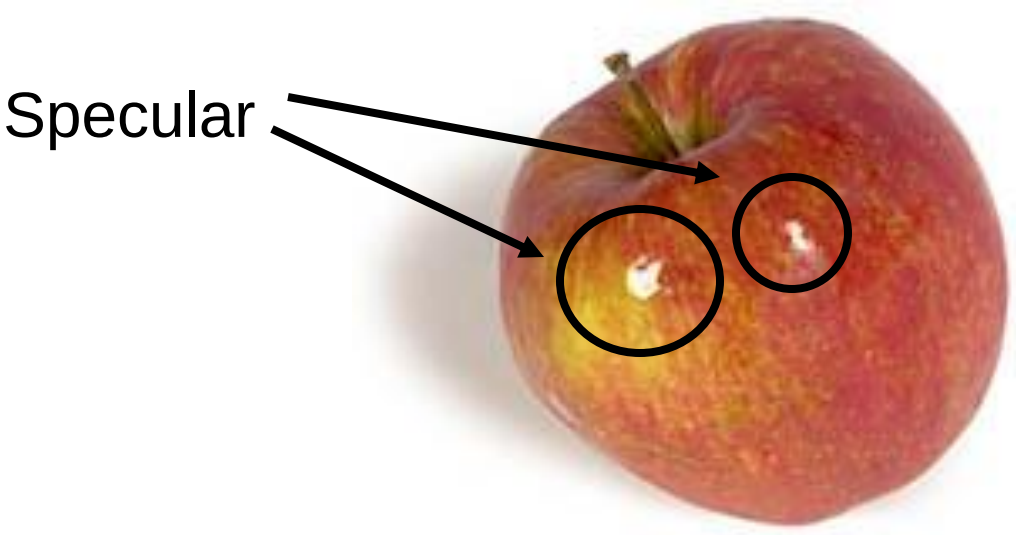


Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: metals, shiny apples

Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: metals, shiny apples

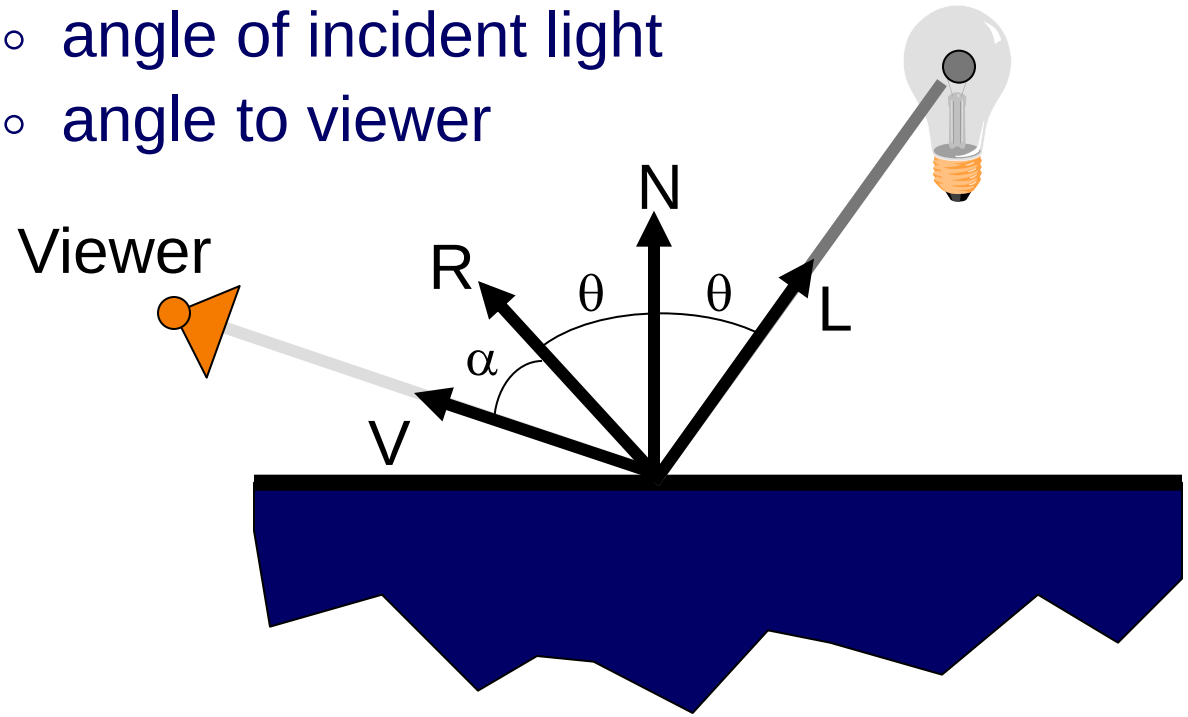


Specular Reflection

How much light is seen?

Depends on:

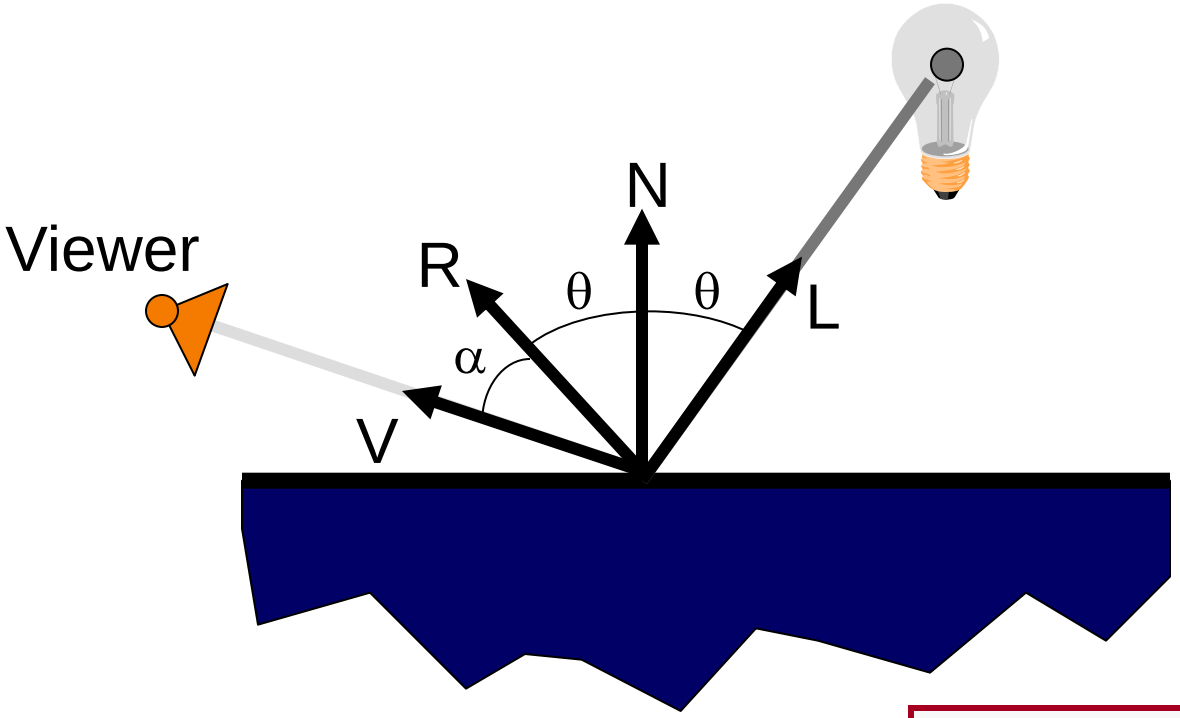
- angle of incident light
- angle to viewer



Specular Reflection

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

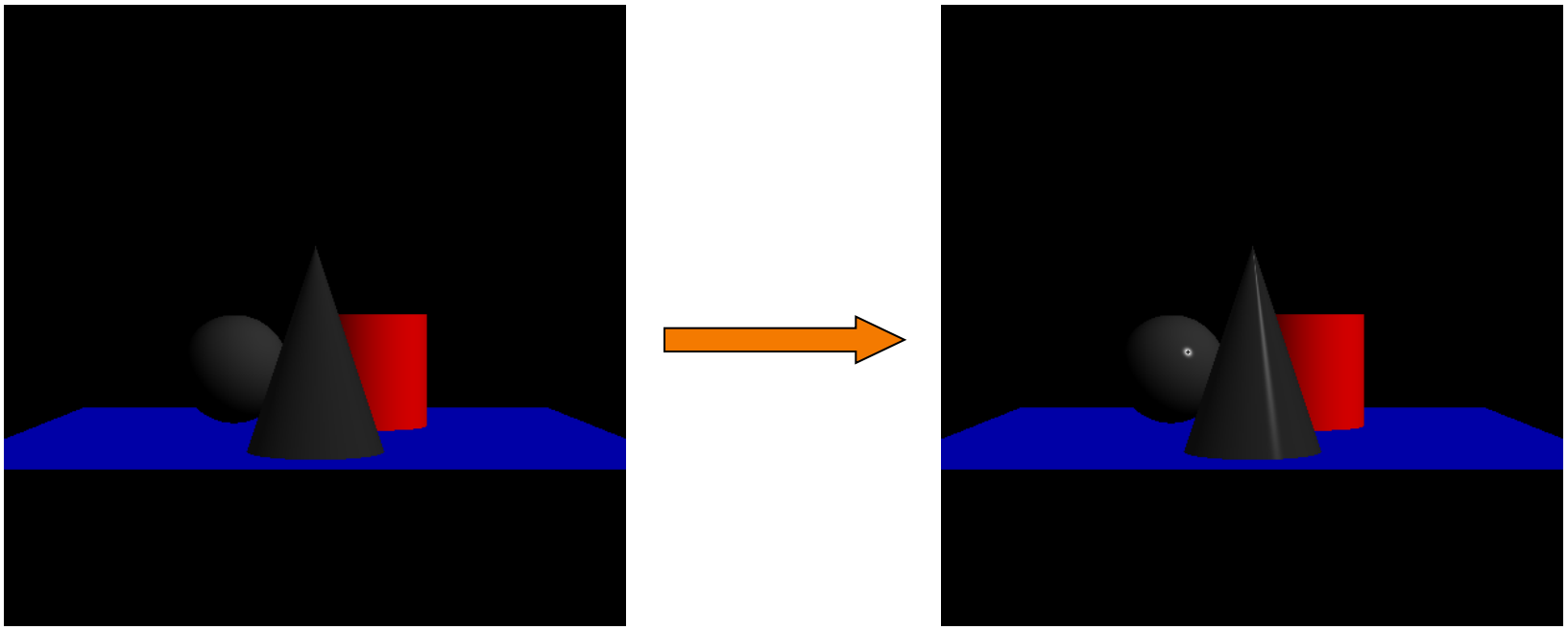
This is a physically-motivated hack!



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

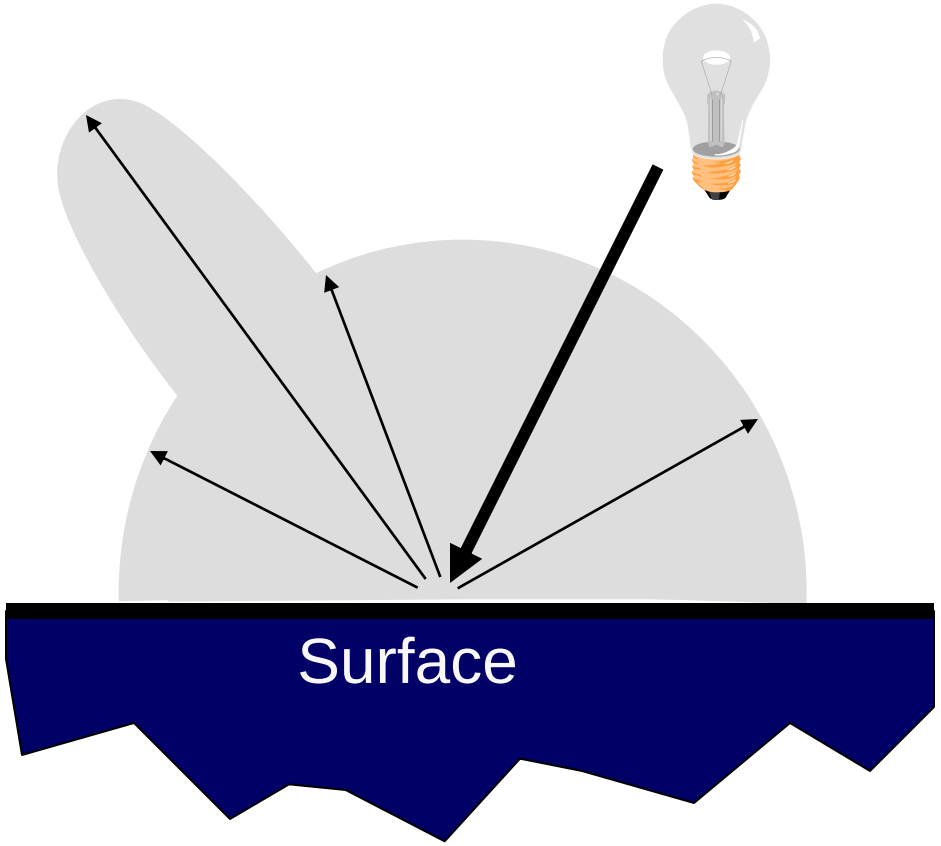
Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: metals, shiny apples



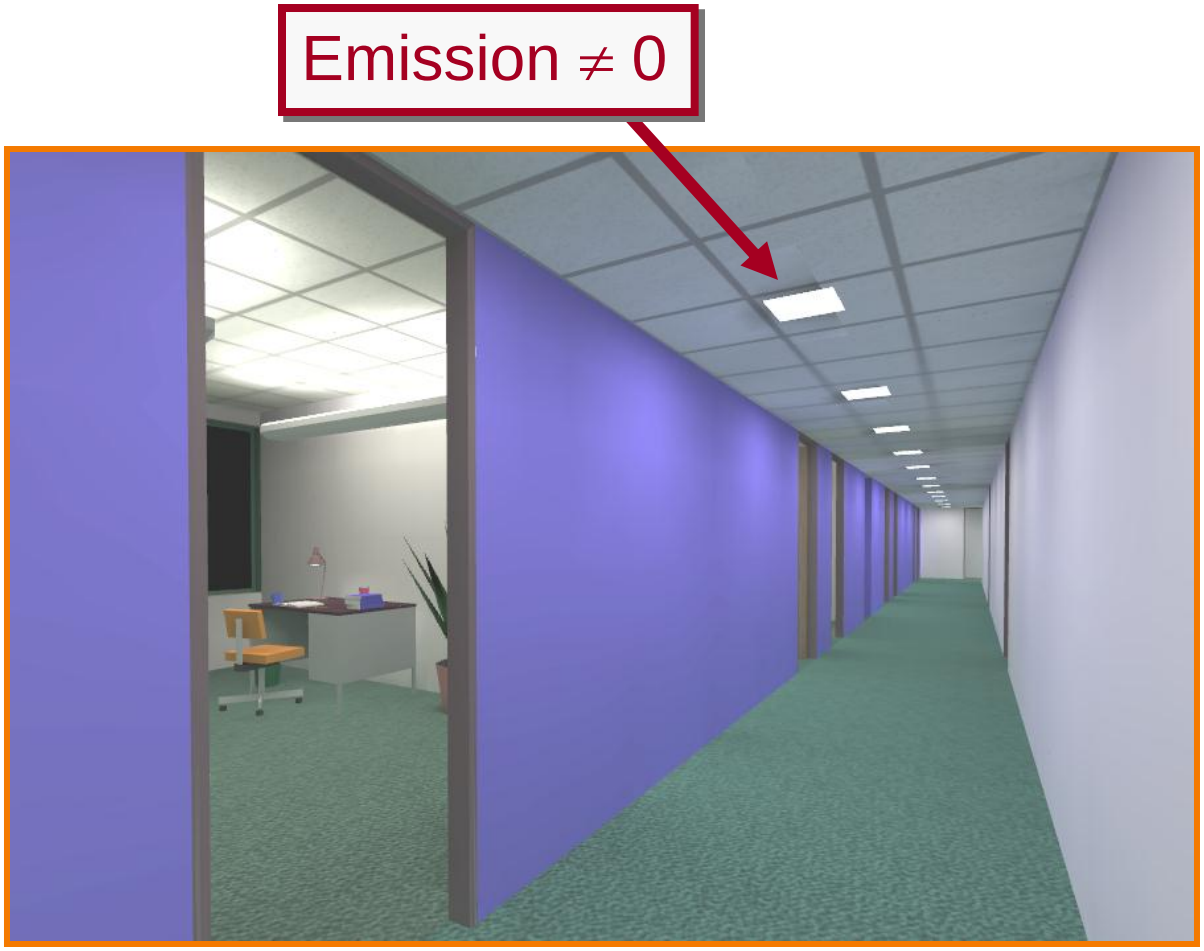
Simple Reflectance Model

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



Emission

- Represents light eminating directly from polygon



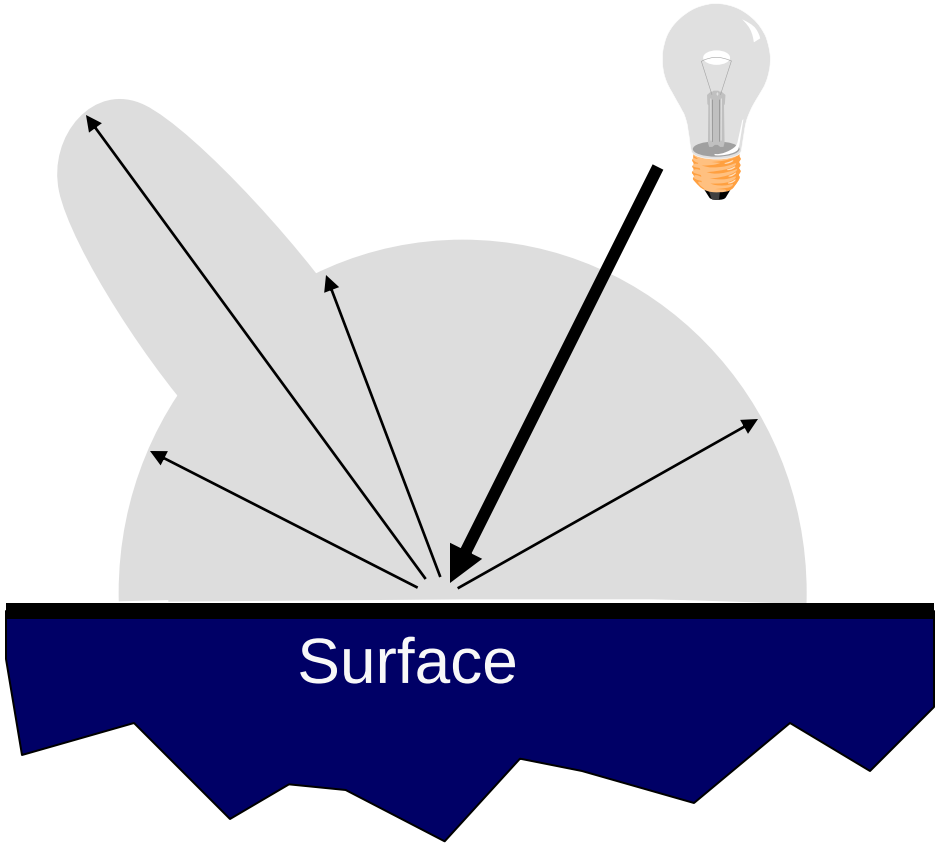
Emission

$$I = I_E$$

Emission ≠ 0

Simple Reflectance Model

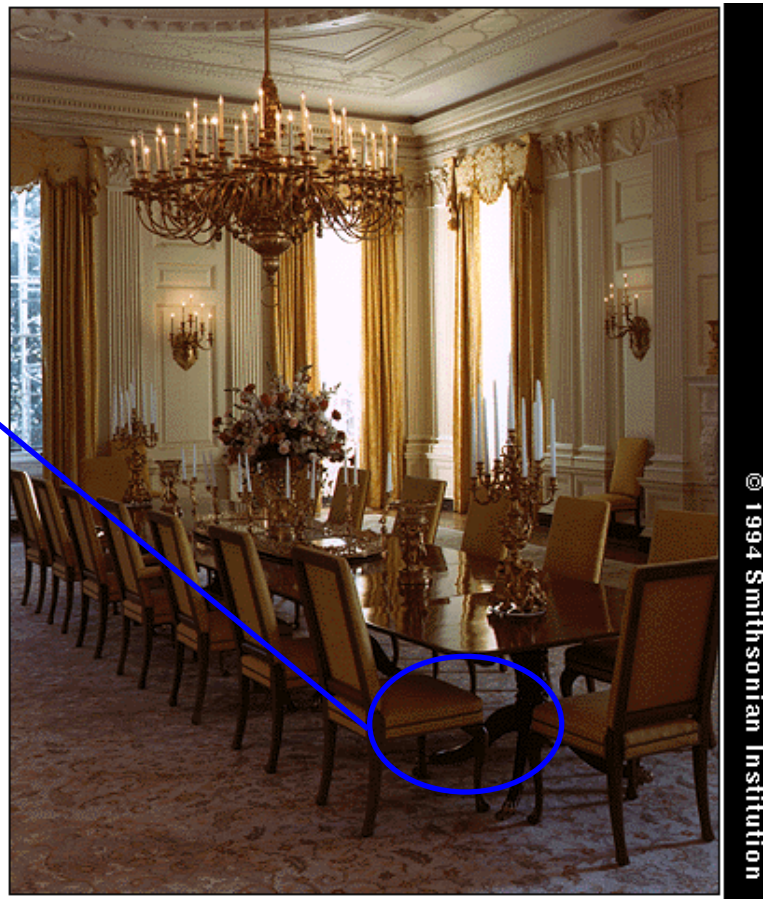
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



Ambient Term

Represents reflection of all indirect illumination

- Locations that are not directly illuminated are still not black because after light rays bounce around the scene, they eventually reach these positions.

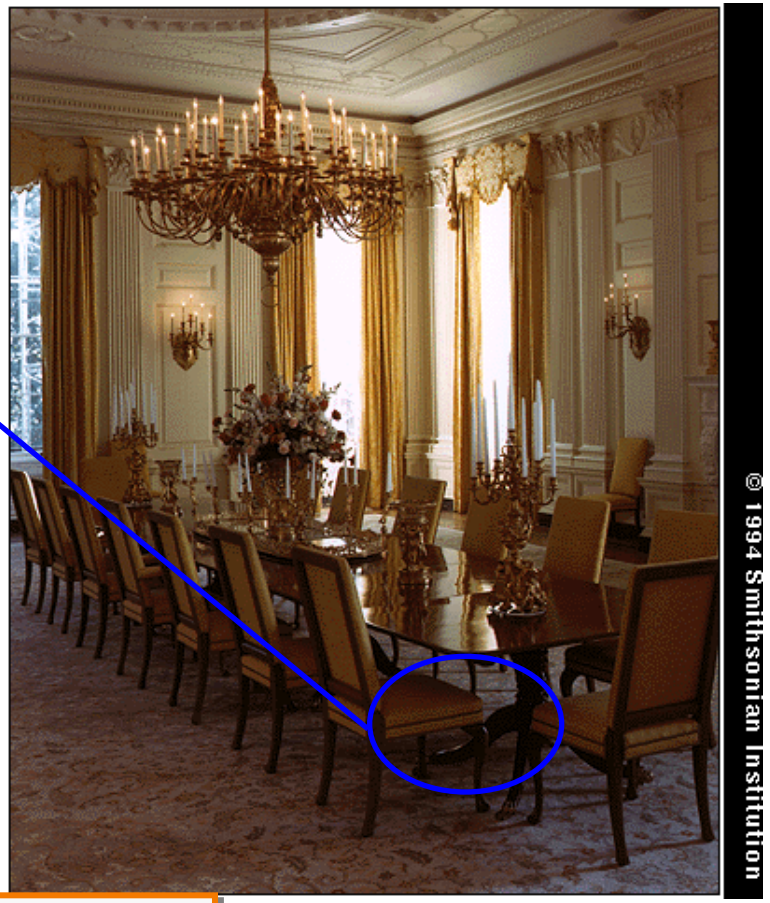


This is a total hack (avoids complexity of global illumination)!

Ambient Term

Represents reflection of all indirect illumination

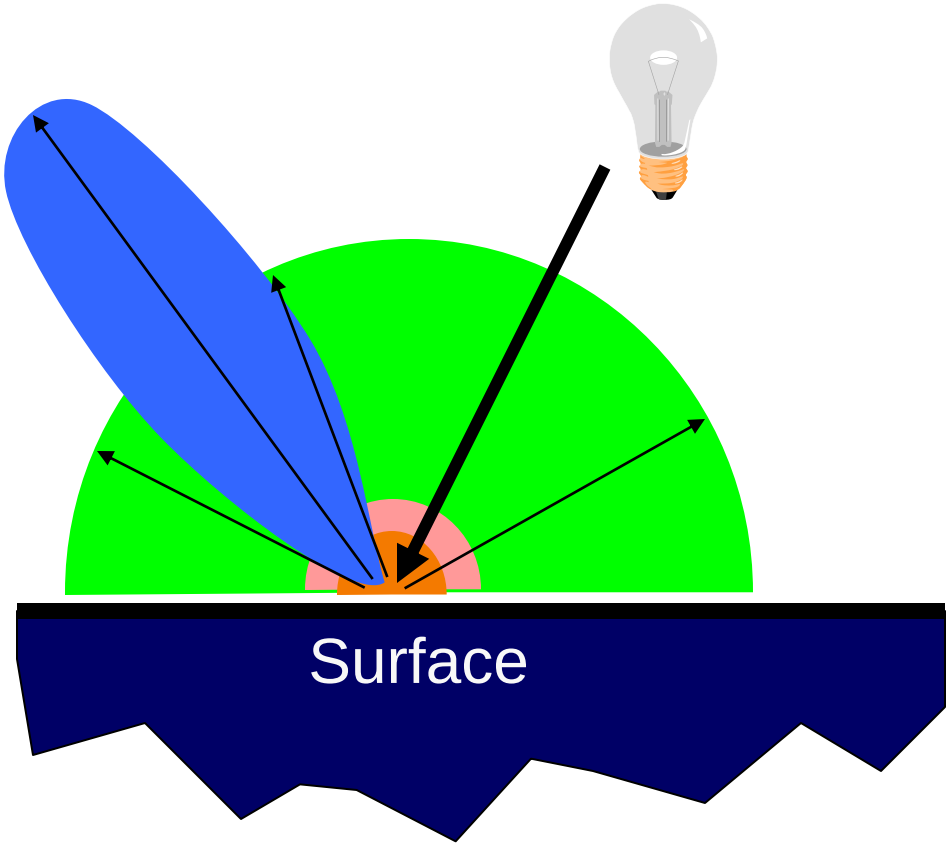
- Locations that are not directly illuminate are still not black because after light rays bounce around the scene, they eventually reach these positions.



$$I_A = K_A I_{AL}$$

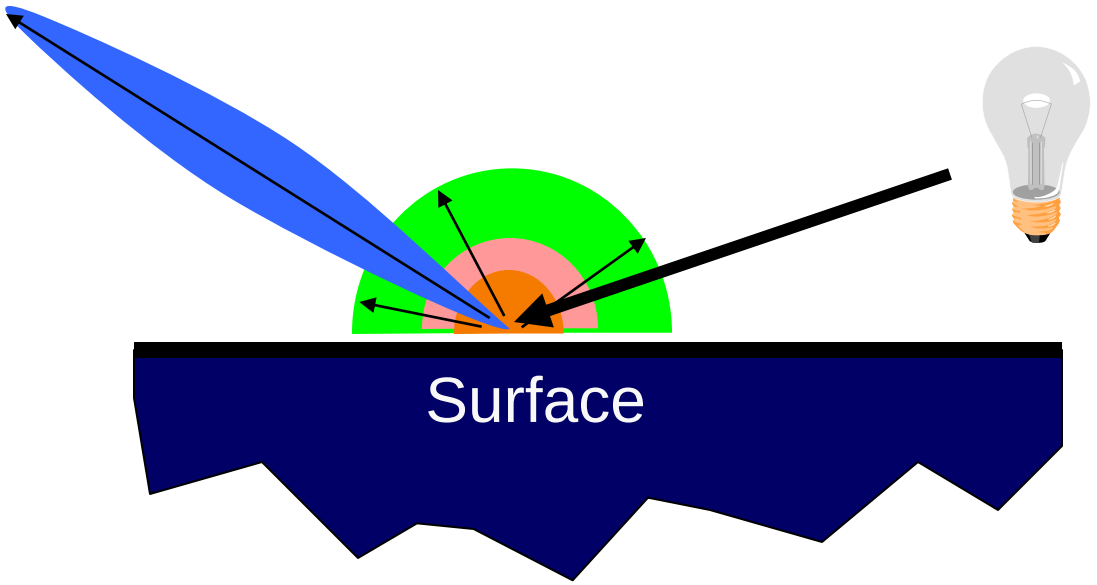
Simple Reflectance Model

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



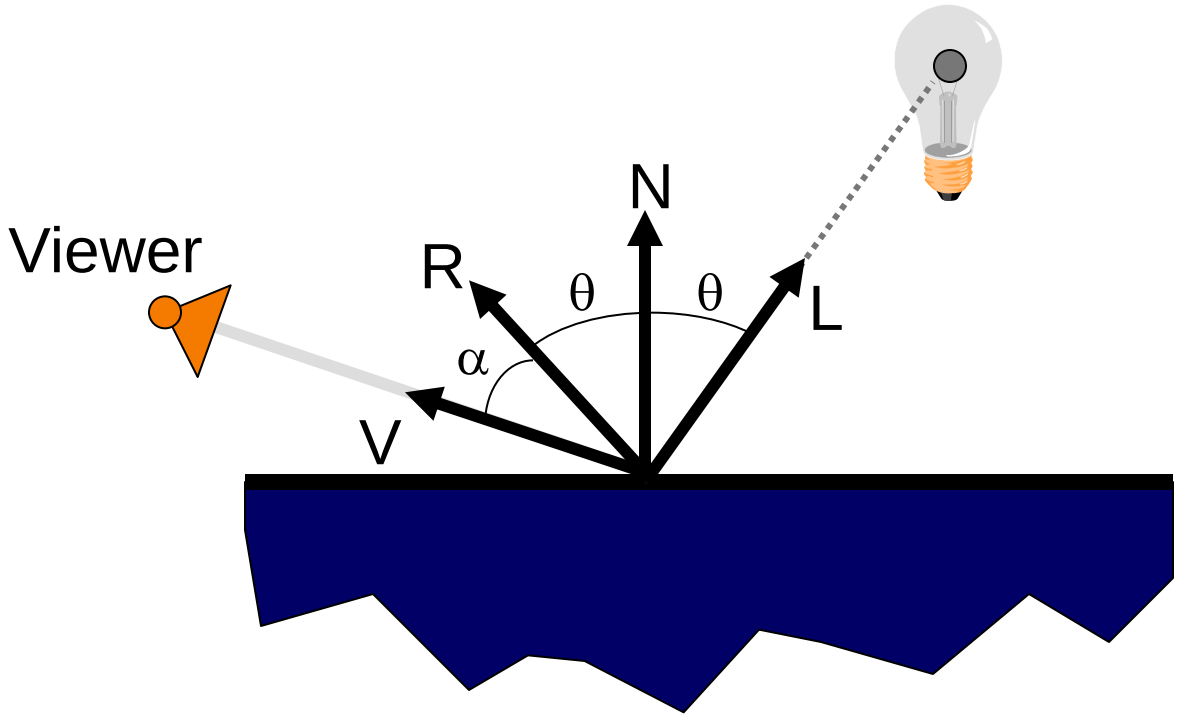
Simple Reflectance Model

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



Surface Illumination Calculation

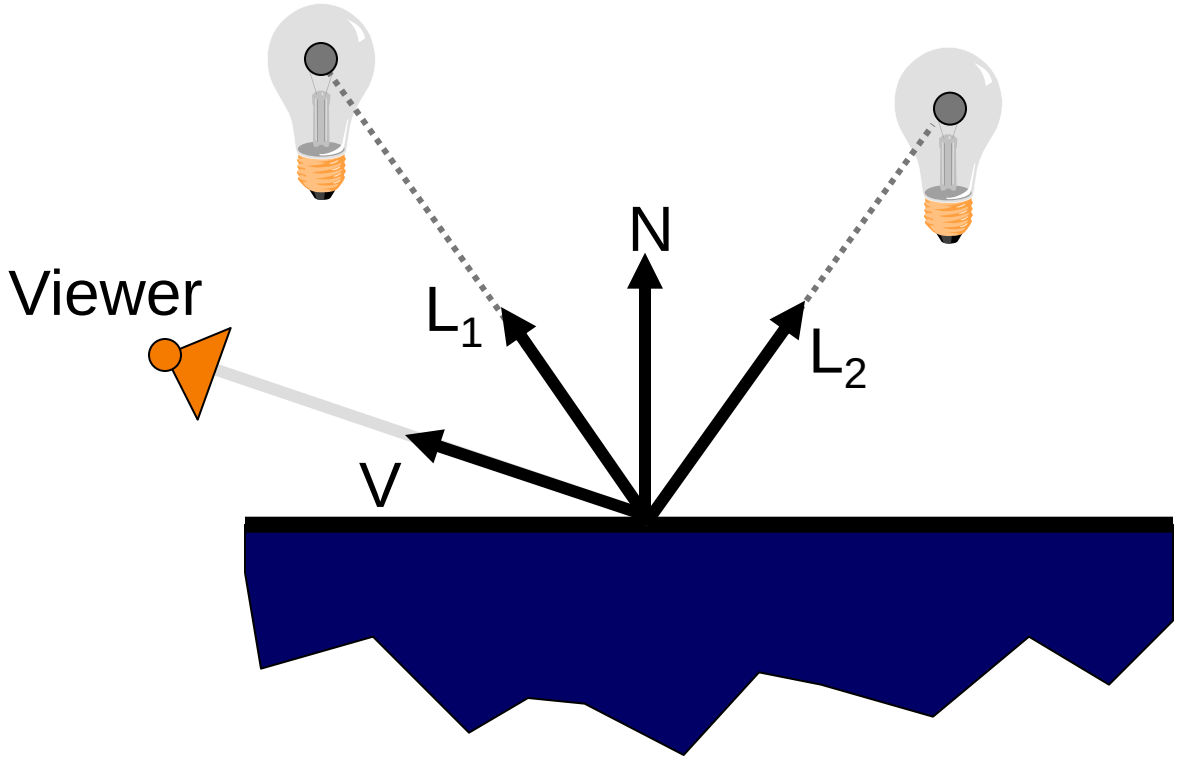
- Single light source :



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

Surface Illumination Calculation

- Multiple light sources:



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

Outline

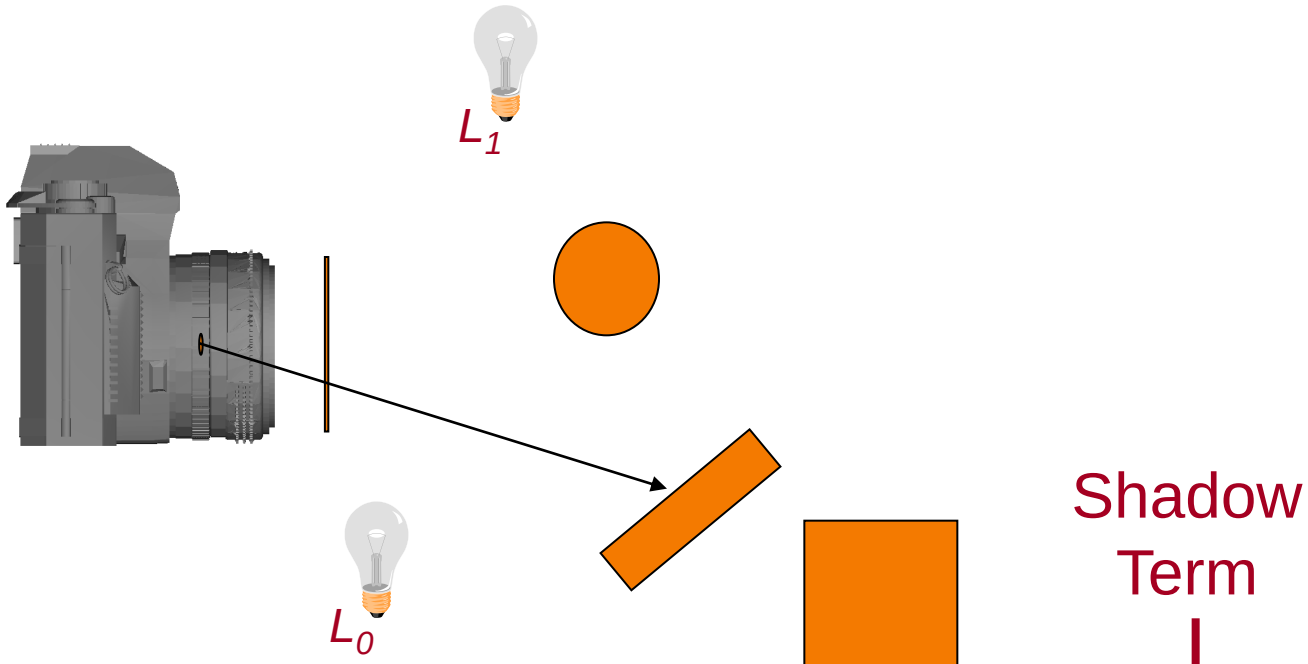
- Introduction
- Ray-Tracing
  - Overview
  - Direct Illumination
  - Global Illumination

Shadows

- Shadow term tells if light sources are blocked
  - Cast ray towards each light source L_i. If the ray is blocked, do not consider the contribution of the light.

Shadows

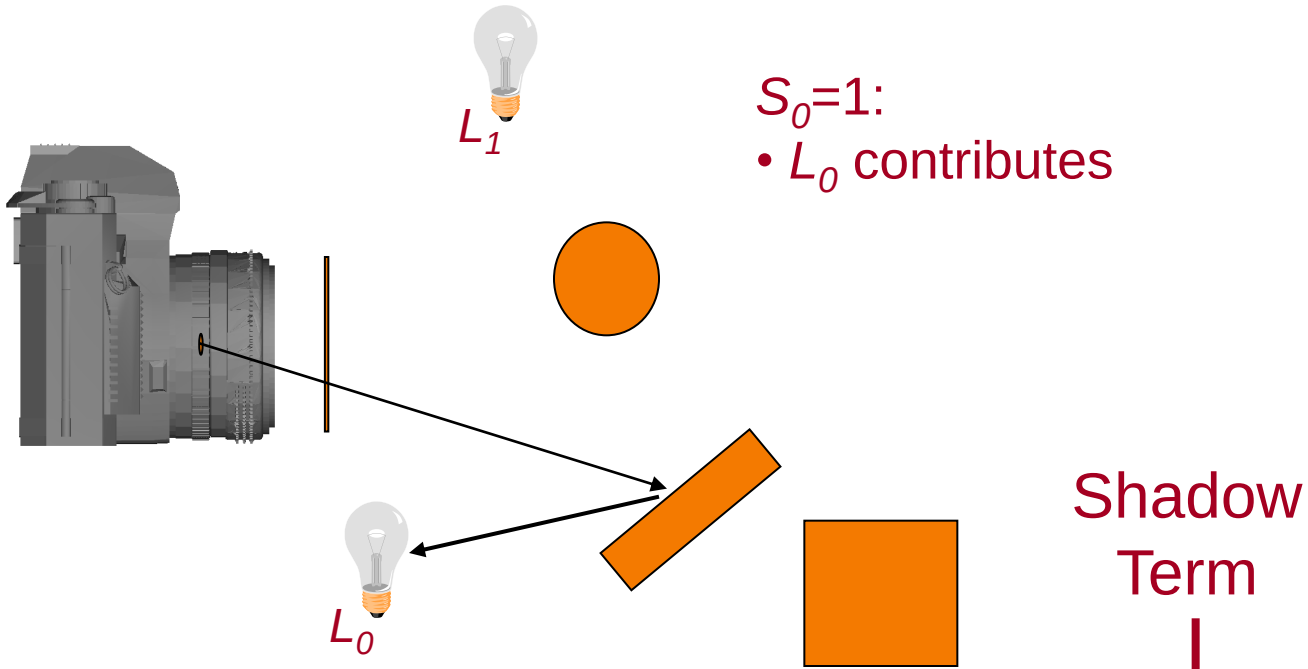
- Shadow term tells if light sources are blocked
  - Cast ray towards each light source L_i
  - $S_i = 0$ if ray is blocked, $S_i = 1$ otherwise



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

Shadows

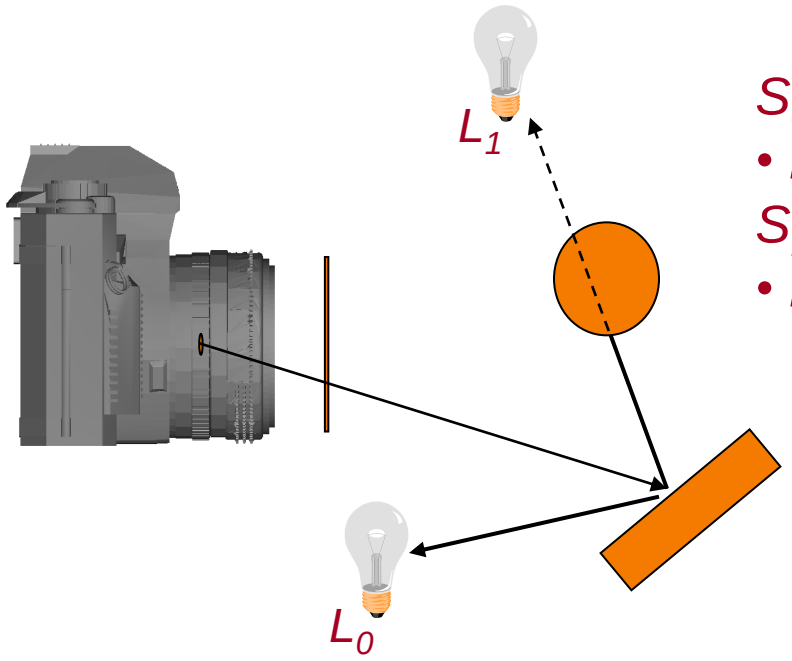
- Shadow term tells if light sources are blocked
  - Cast ray towards each light source L_i
  - $S_i = 0$ if ray is blocked, $S_i = 1$ otherwise



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

Shadows

- Shadow term tells if light sources are blocked
  - Cast ray towards each light source L_i
  - $S_i = 0$ if ray is blocked, $S_i = 1$ otherwise



S_0=1:
- L_0 contributes

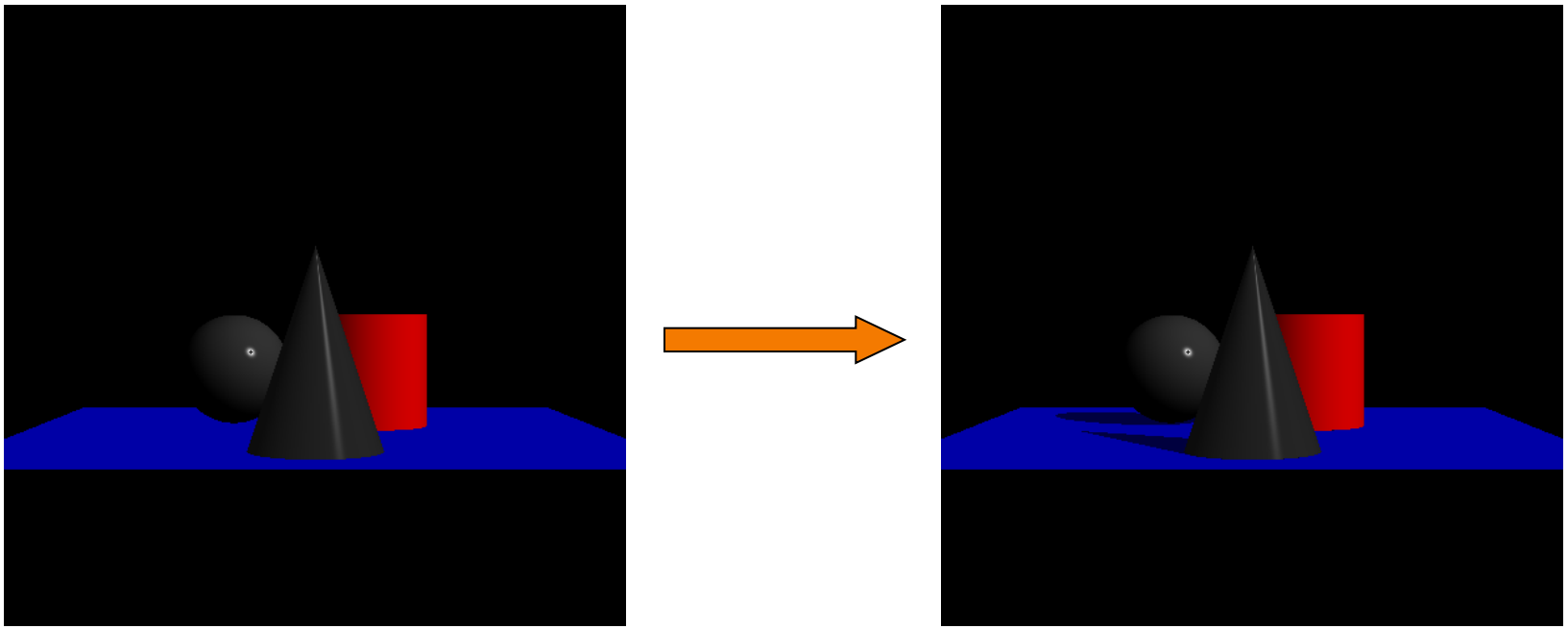
S_1=0:
- L_1 does not contribute

Shadow Term

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

Ray Casting

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

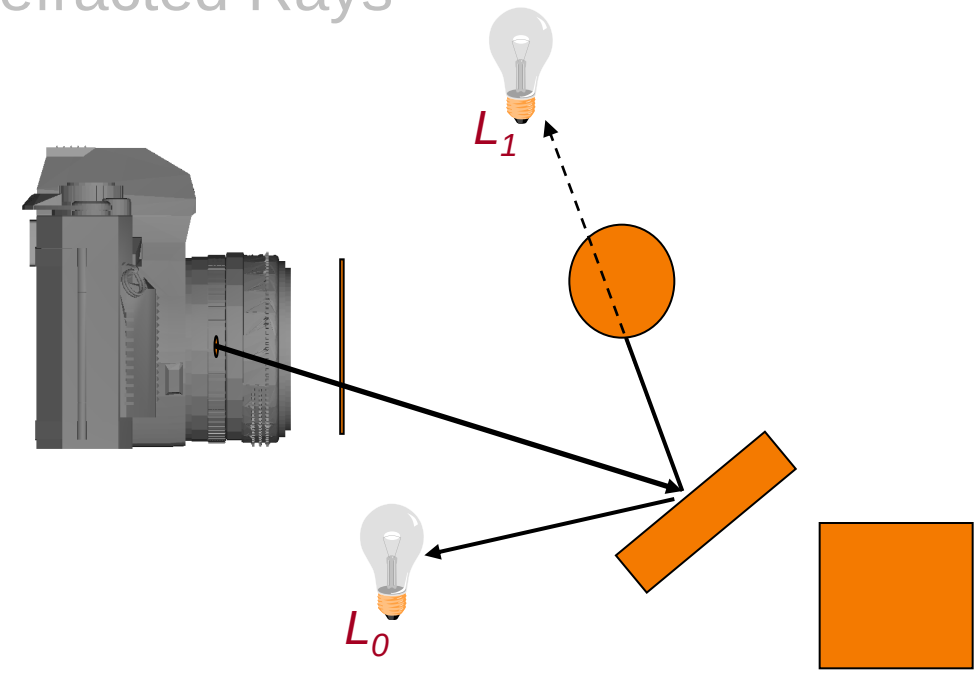


Recursive Ray Tracing

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

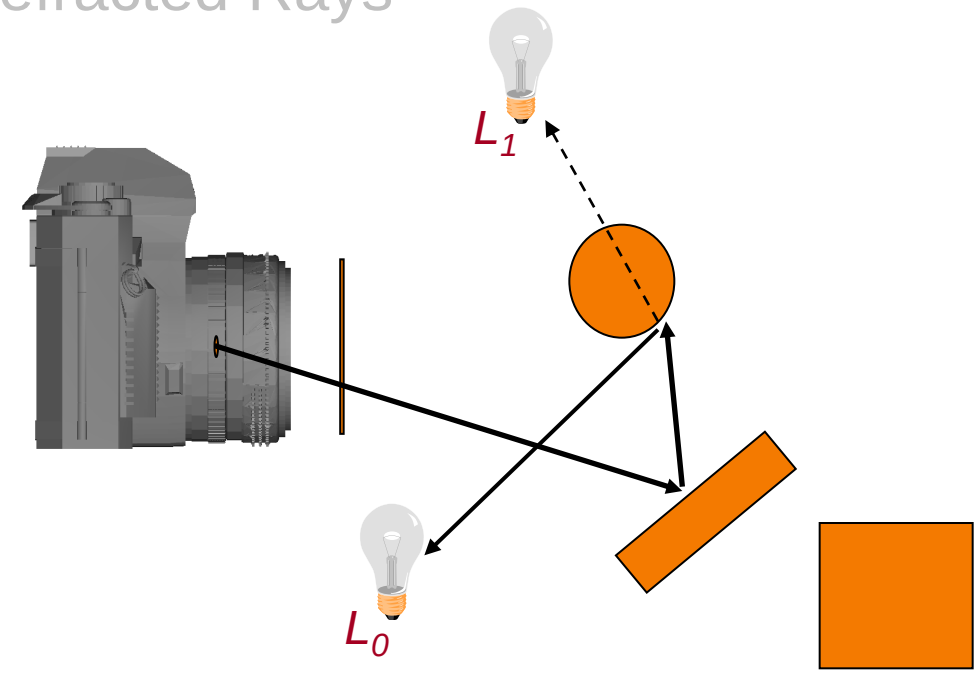


Radiance for mirror reflection ray

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

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

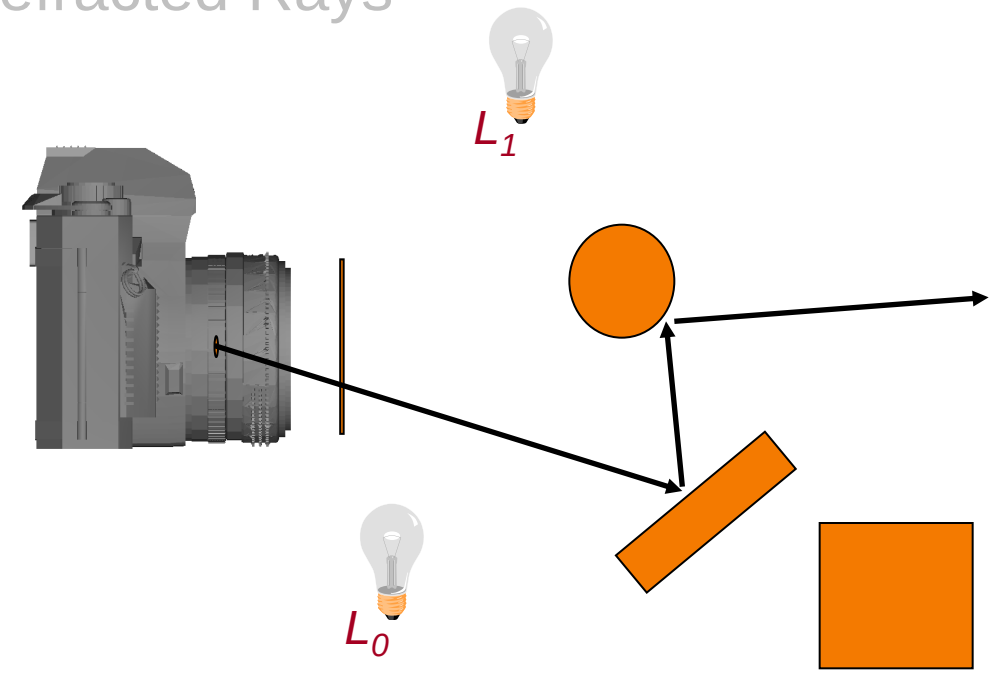


Radiance for mirror reflection ray

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

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

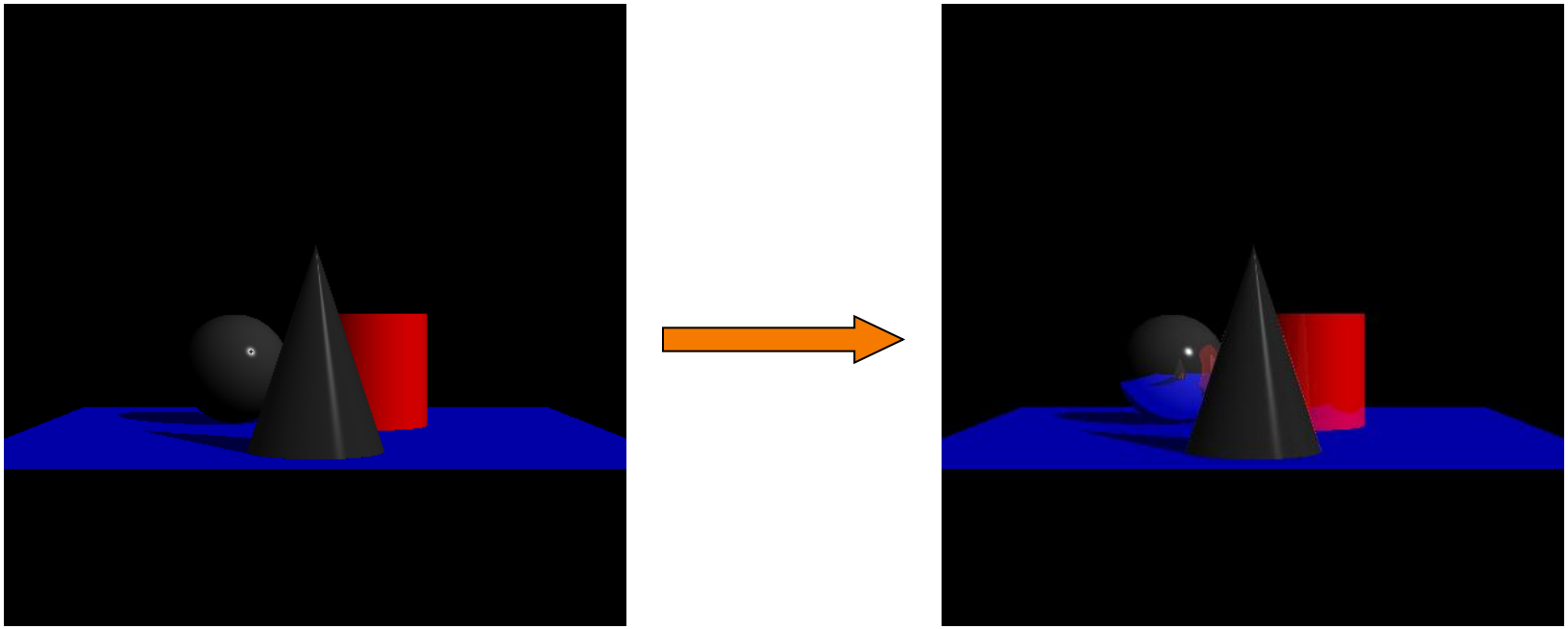


Radiance for mirror reflection ray

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

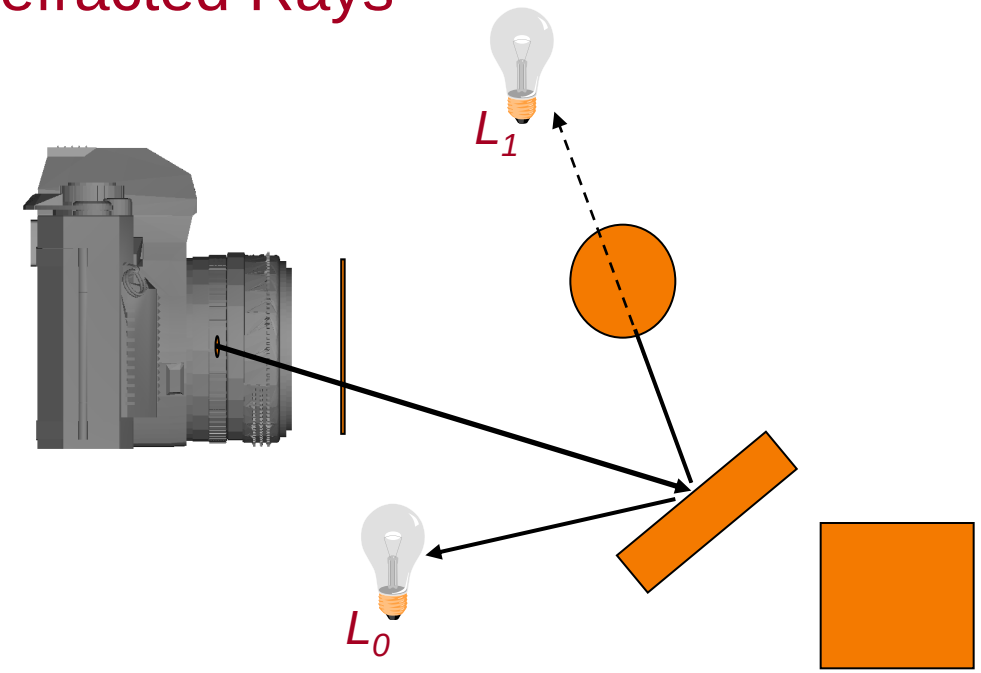
Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays



Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

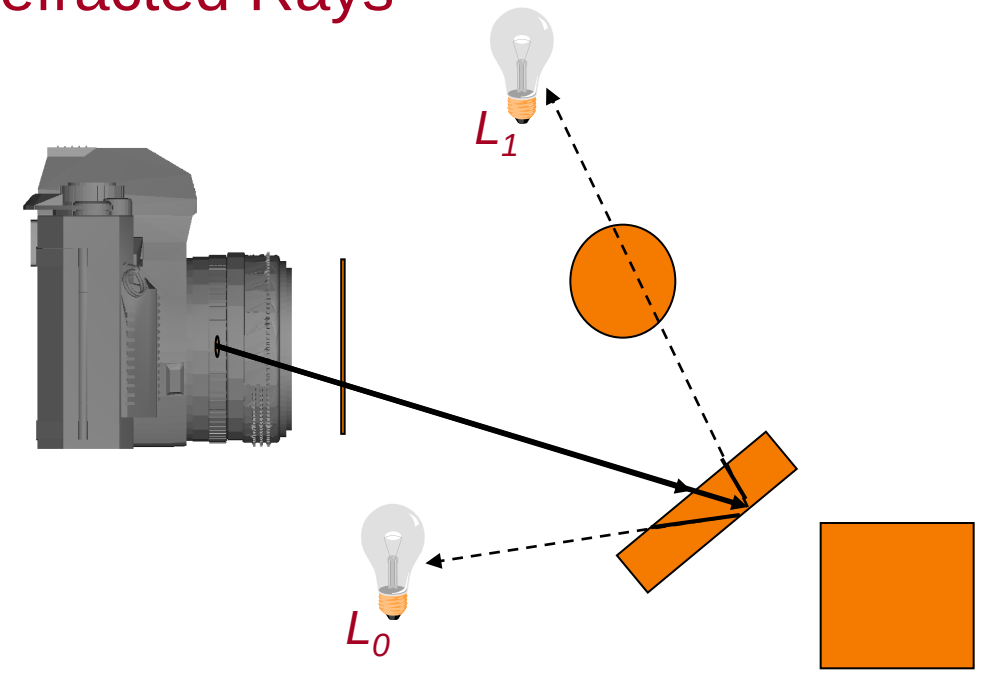


Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

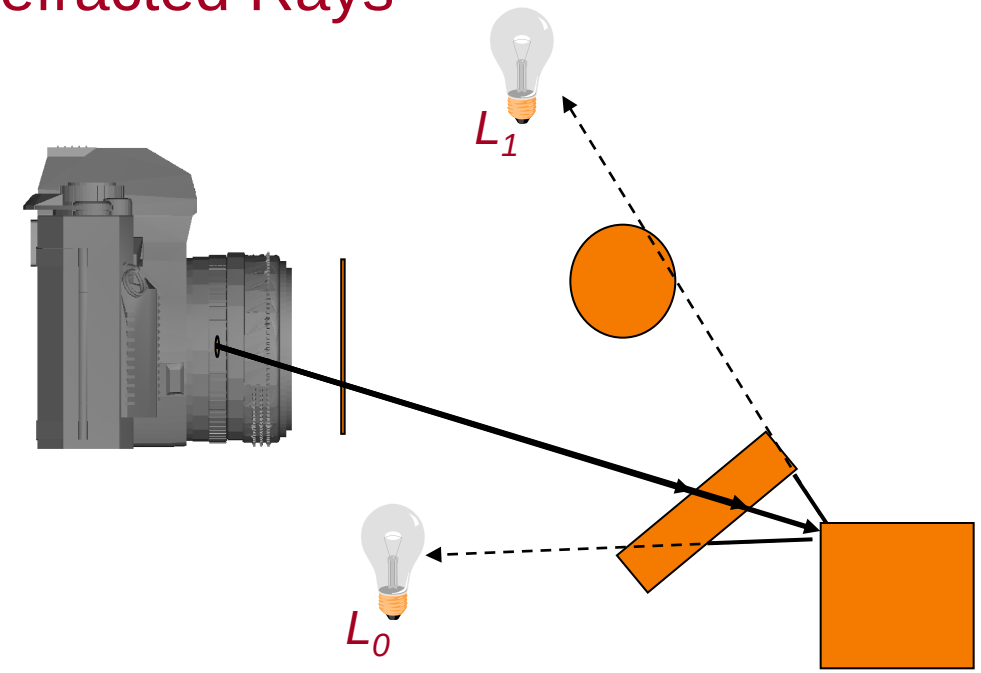


Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

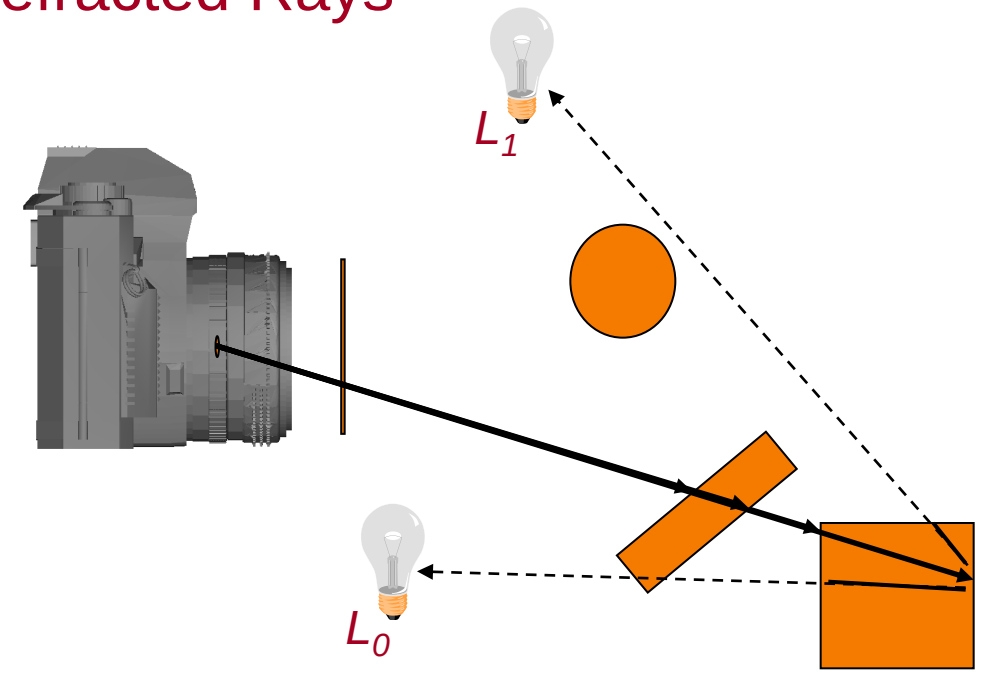


Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

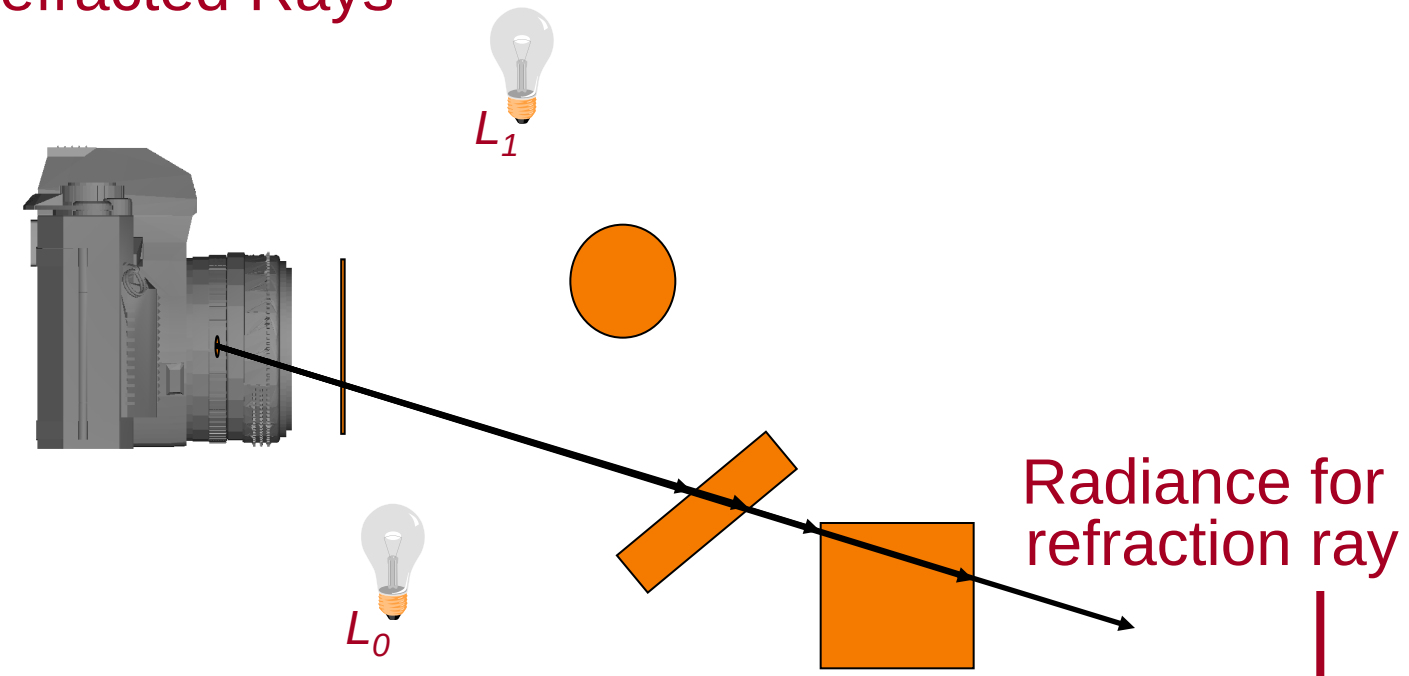


Radiance for refraction ray

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

Transparent Refraction

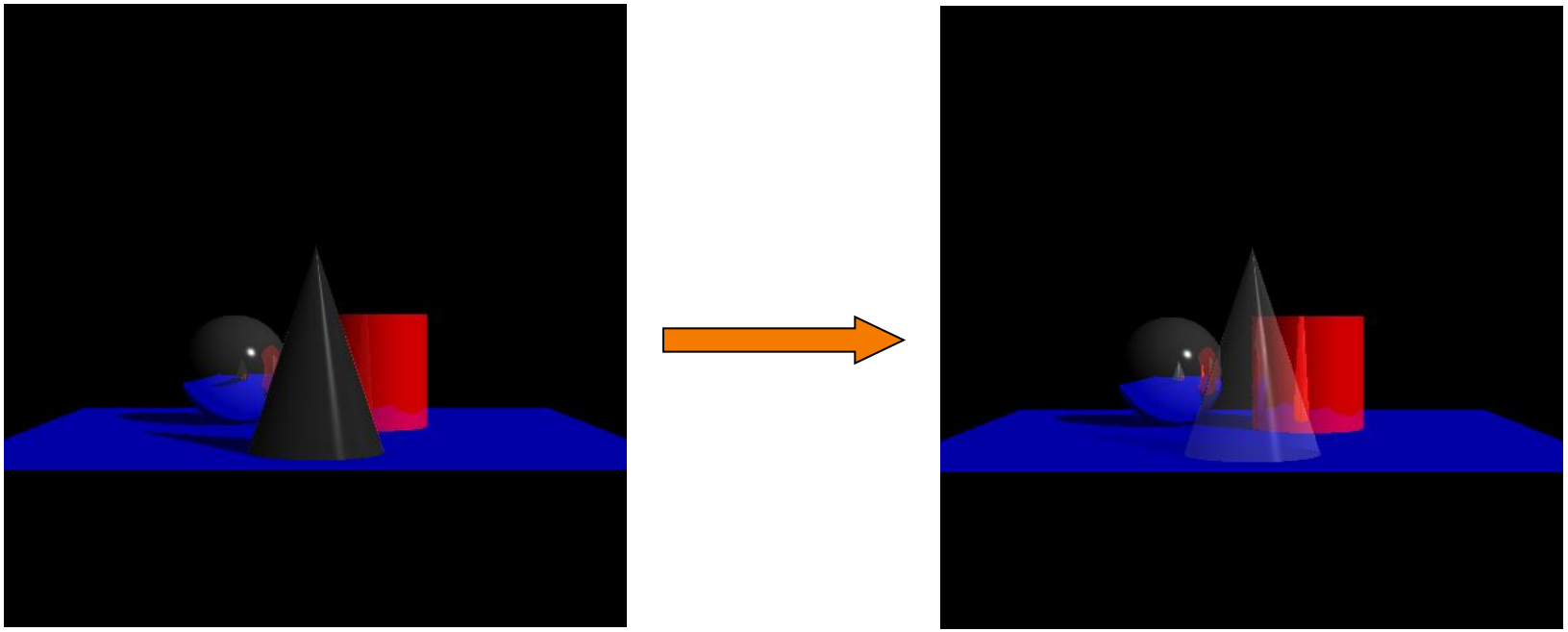
- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays



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

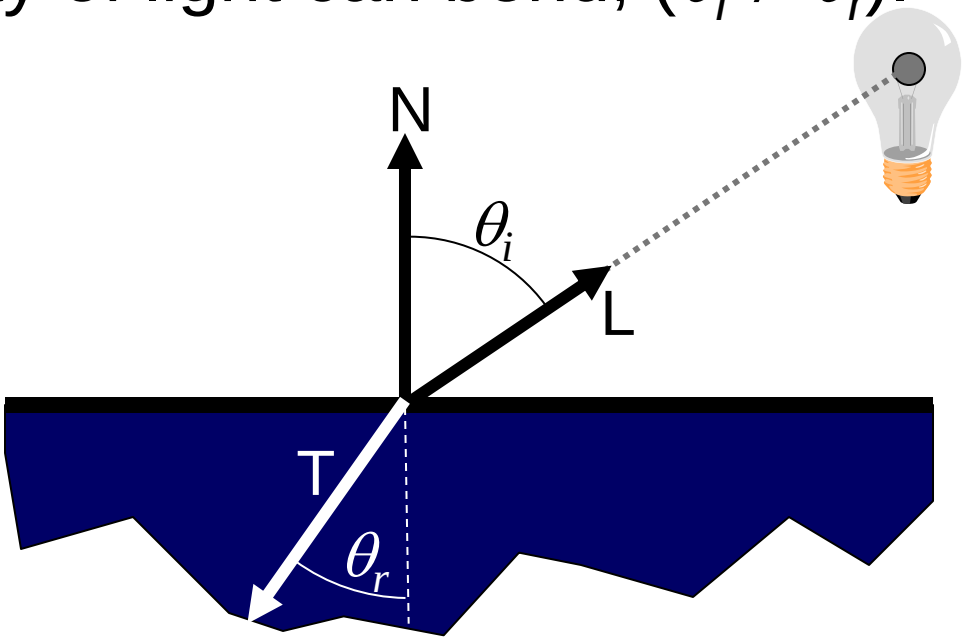
Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays



Transparent Refraction

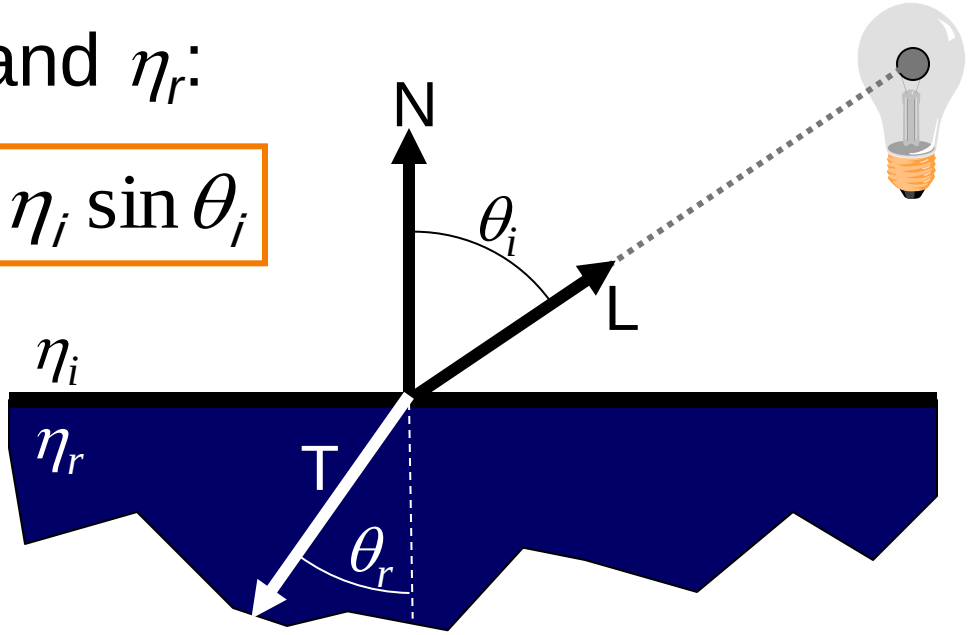
- When a light of light passes through a transparent object, the ray of light can bend, ($\theta_i \neq \theta_r$).



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\eta_r \sin\theta_r = \eta_i \sin\theta_i$$

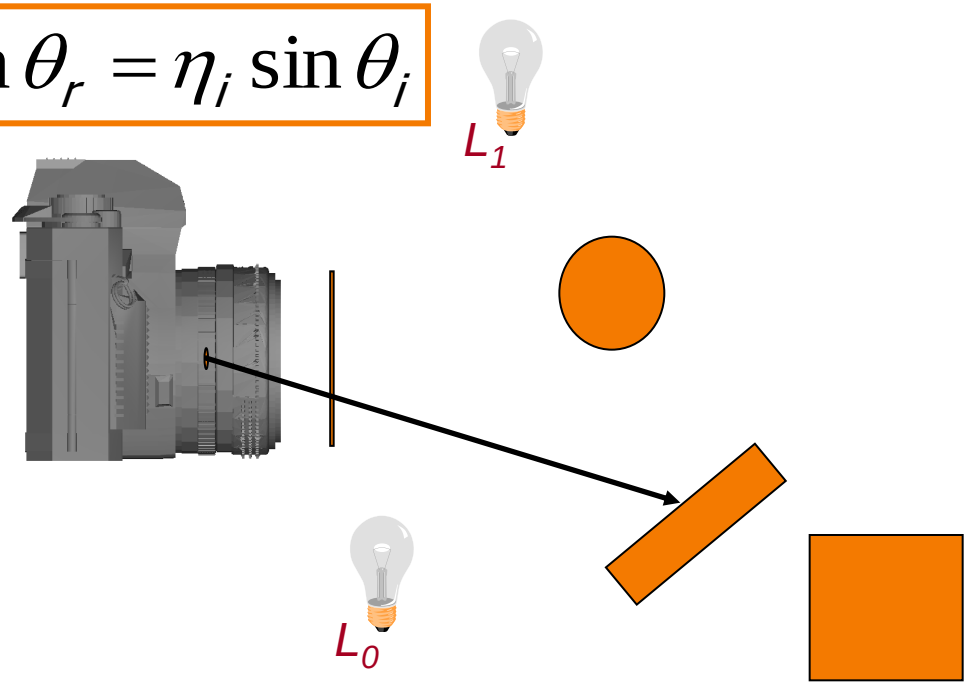


The index of refraction of air is η=1.

Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

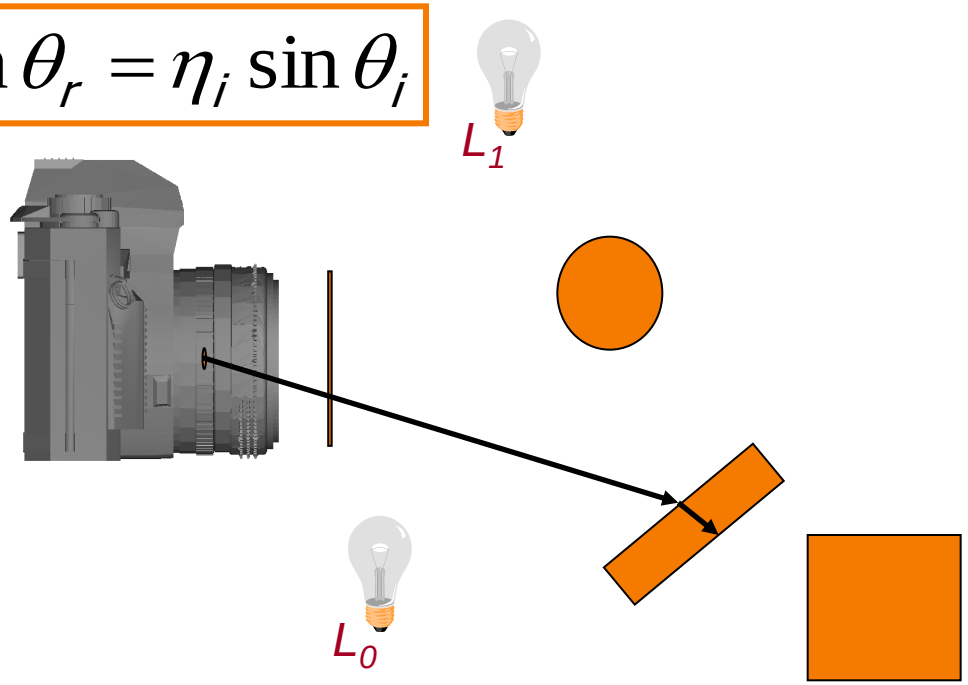
$$\eta_r \sin\theta_r = \eta_i \sin\theta_i$$



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

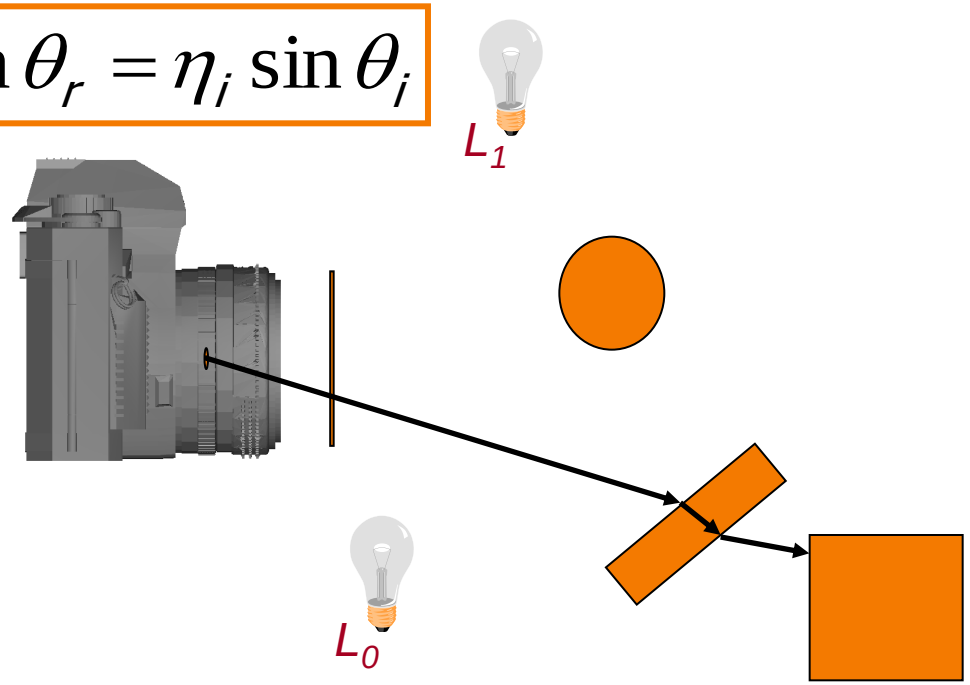
$$\eta_r \sin\theta_r = \eta_i \sin\theta_i$$



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

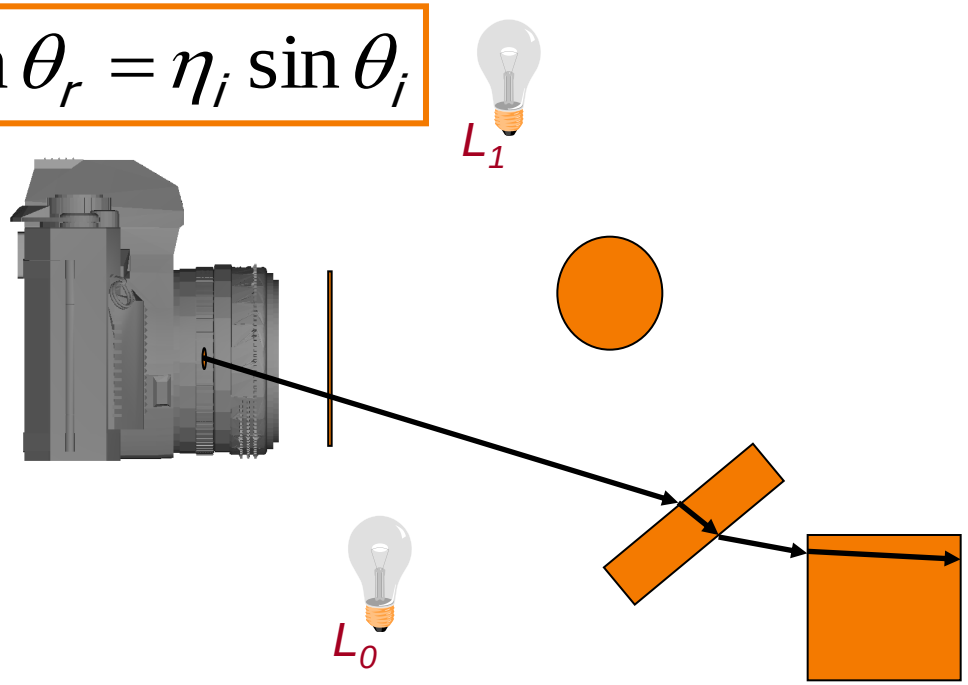
$$\eta_r \sin\theta_r = \eta_i \sin\theta_i$$



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

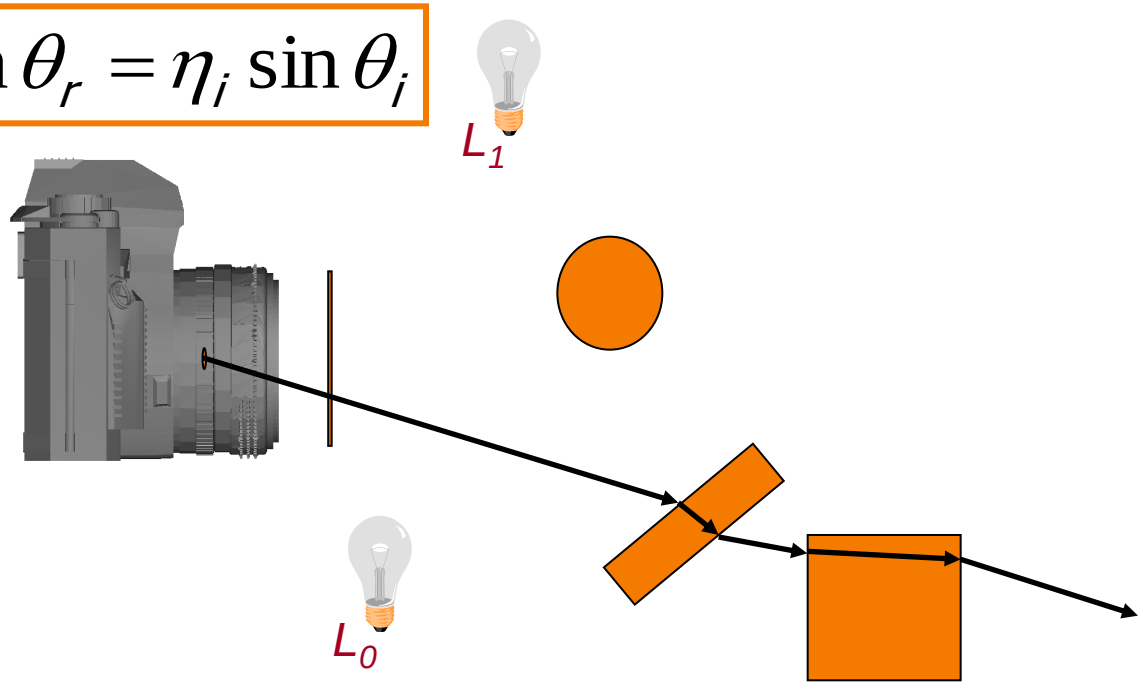
$$\eta_r \sin\theta_r = \eta_i \sin\theta_i$$



Snell's Law

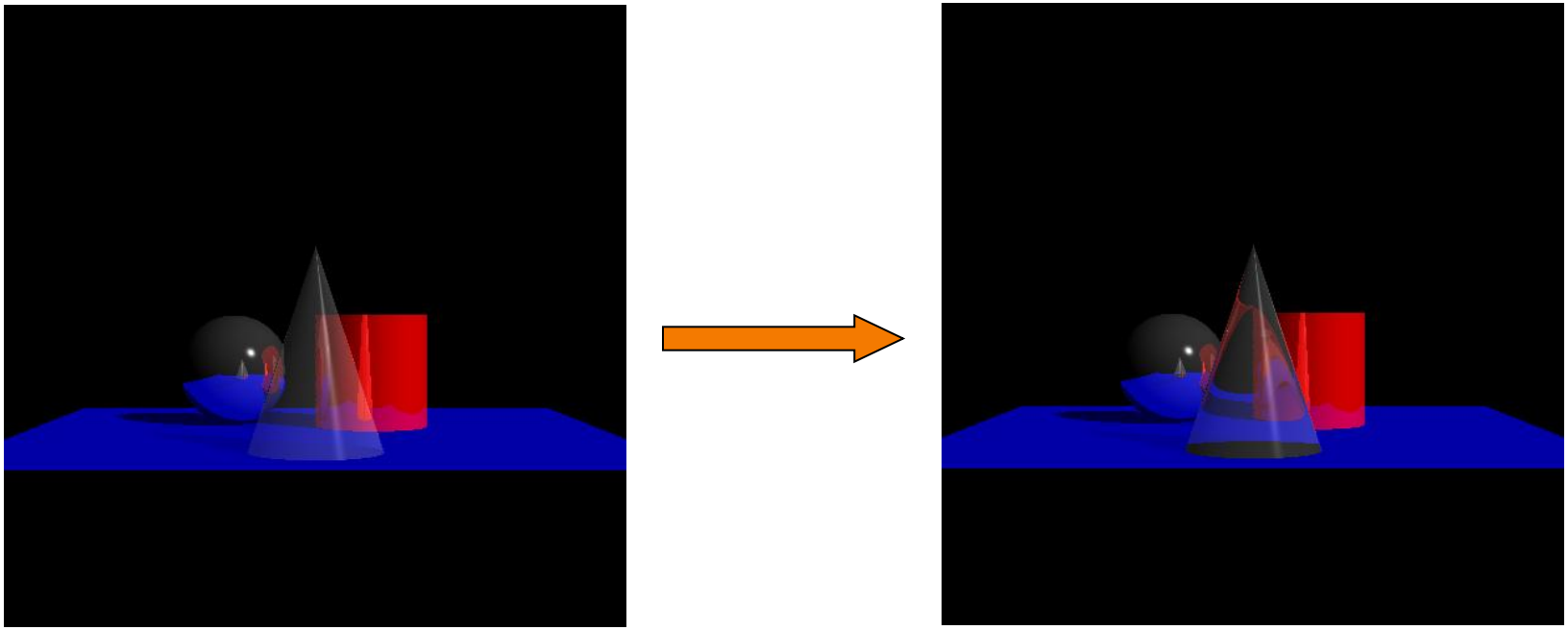
- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\eta_r \sin\theta_r = \eta_i \sin\theta_i$$



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

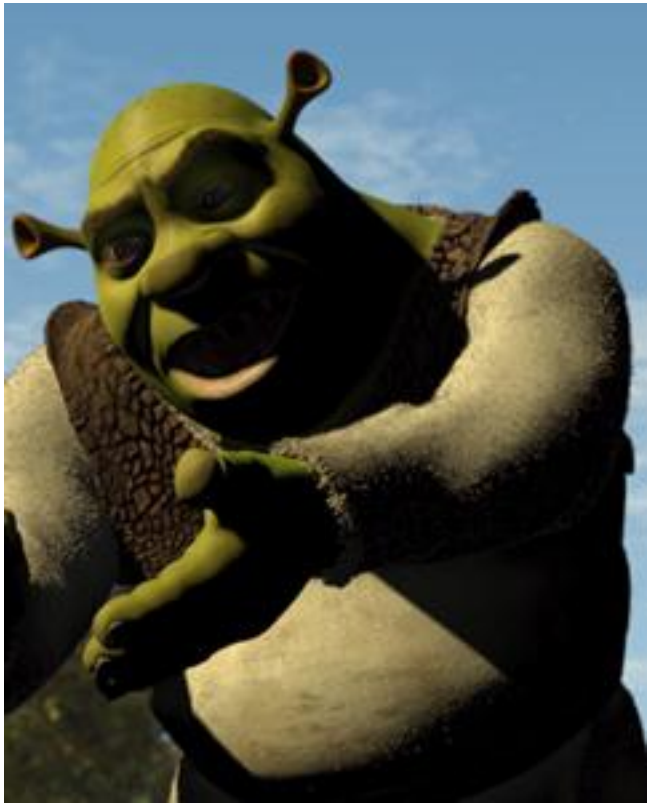


Summary

- 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

Summary



direct illumination

global illumination

Global illumination makes a big difference!

PDI/Dreamworks