CS380: Computer Graphics
Ray Tracing

Sung-Eui Yoon
(윤성의)

Course URL:
http://sgvr.kaist.ac.kr/~sungeui/CG/

KAIST

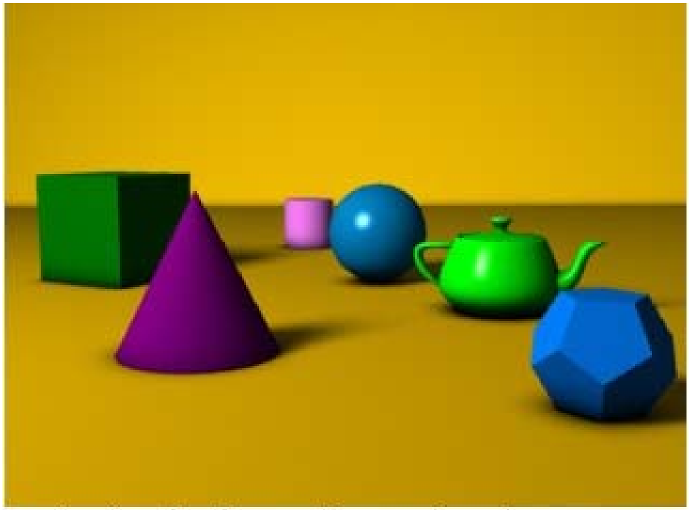
Class Objectives

- **Understand overall algorithm of recursive ray tracing**
  - **Ray generations**
  - **Intersection tests**

- **Related chapter**
  - **Part II, Ch. 10: Ray Tracing**

Various Visibility Algorithm

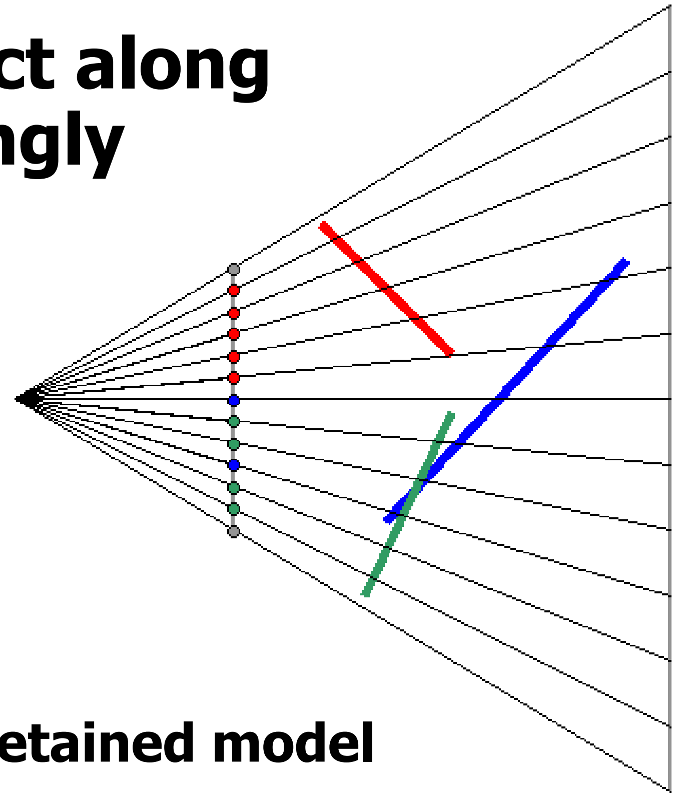
- Z-buffer
- Ray casting, etc.



A simple three dimensional scene

Z-buffer representation

Ray Casting

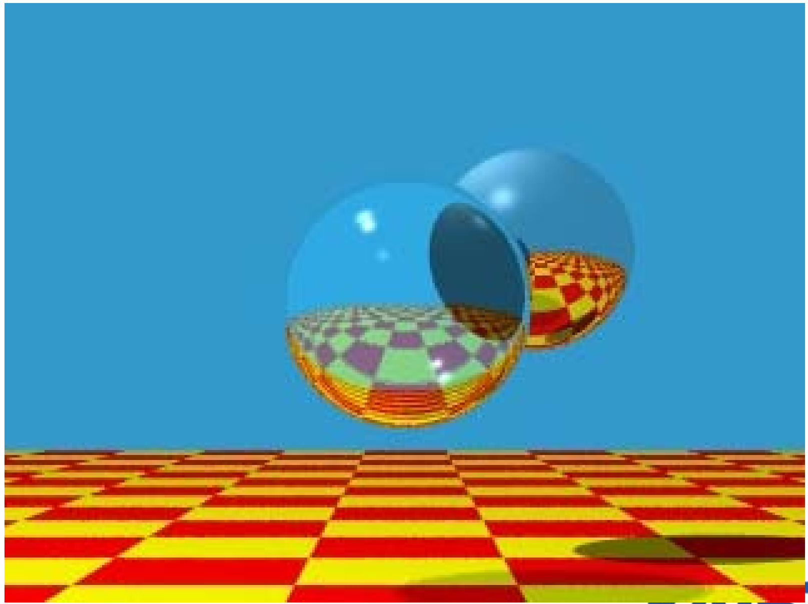


- For each pixel, find closest object along the ray and shade pixel accordingly
- Advantages
  - Conceptually simple
  - Can be extended to handle global illumination effects
- Disadvantages
  - Renderer must have access to entire retained model
  - Hard to map to special-purpose hardware
  - Less efficient than rasterization in terms of utilizing spatial coherence

Recursive Ray Casting

- **Ray casting generally dismissed early on because of aforementioned problems**

- **Gained popularity in when Turner Whitted (1980) showed this image**
  - ***Show recursive* ray casting could be used for global illumination effects**

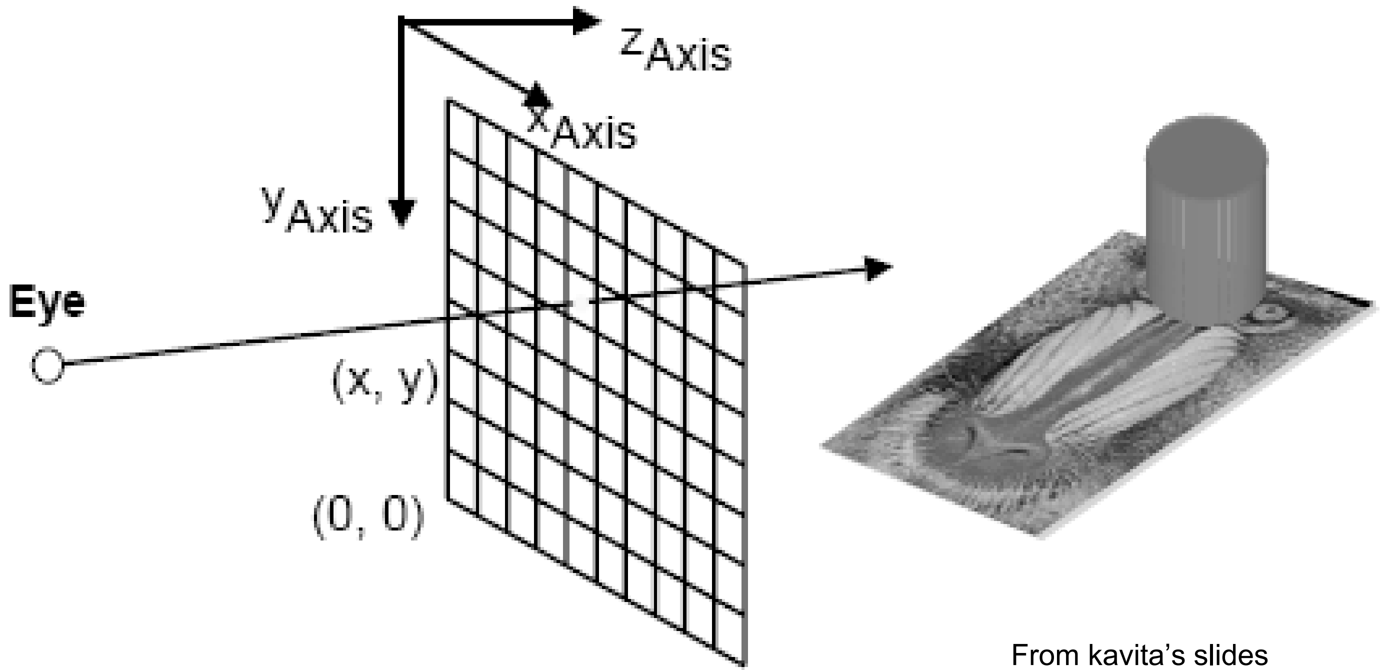


Ray Casting and Ray Tracing

- **Trace rays from eye into scene**
  - **Backward ray tracing**
- **Ray casting used to compute visibility at the eye**
- **Perform (recursive) ray tracing for arbitrary rays needed for shading**
  - **Reflections**
  - **Refraction and transparency**
  - **Shadows**

Basic Algorithms of Ray Tracing

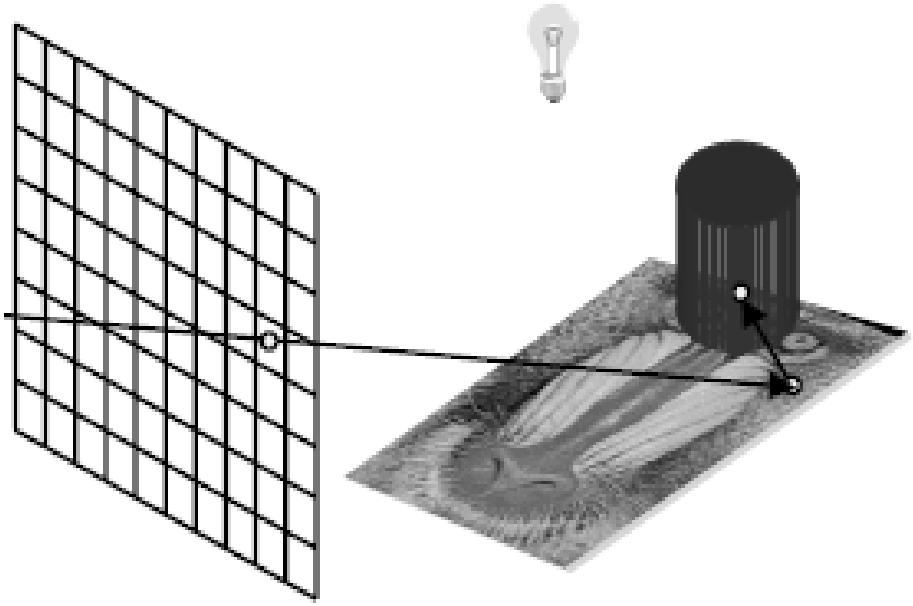
- **Rays are cast from the eye point through each pixel in the image**

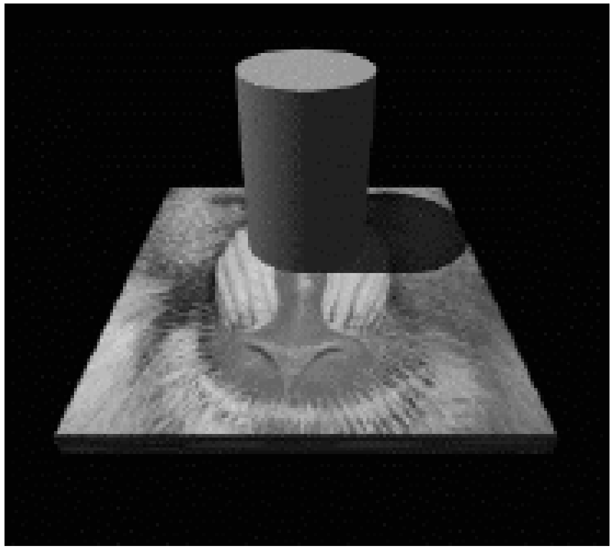


From kavita's slides

Shadows

- **Cast ray from the intersection point to each light source**
  - **Shadow rays**

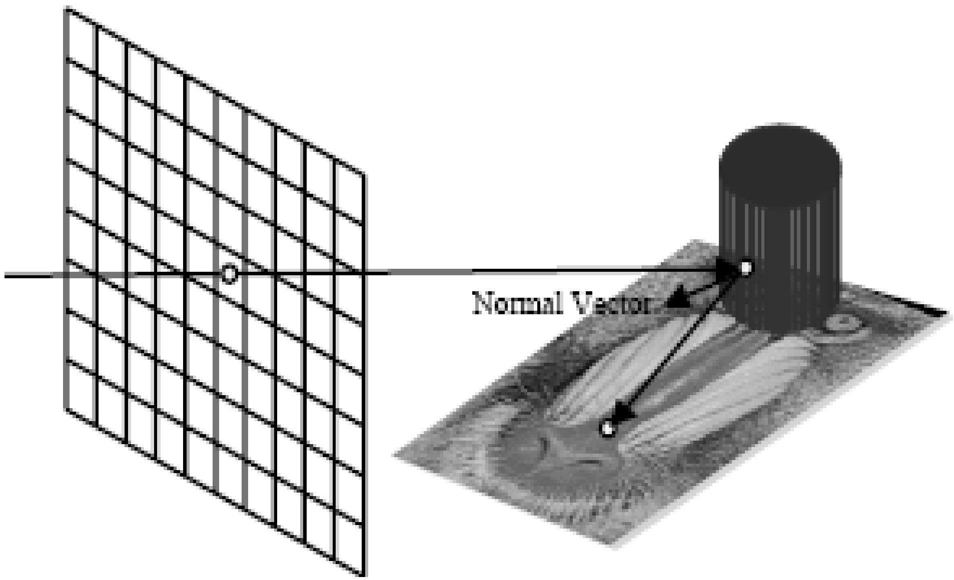


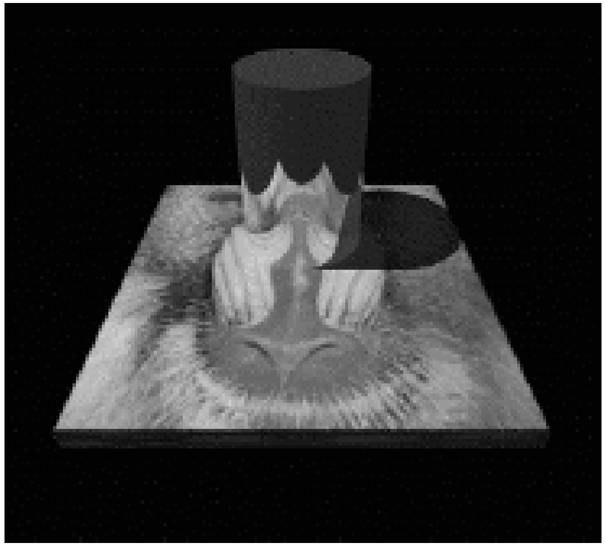


From kavita's slides

Reflections

- **If object specular, cast secondary reflected rays**

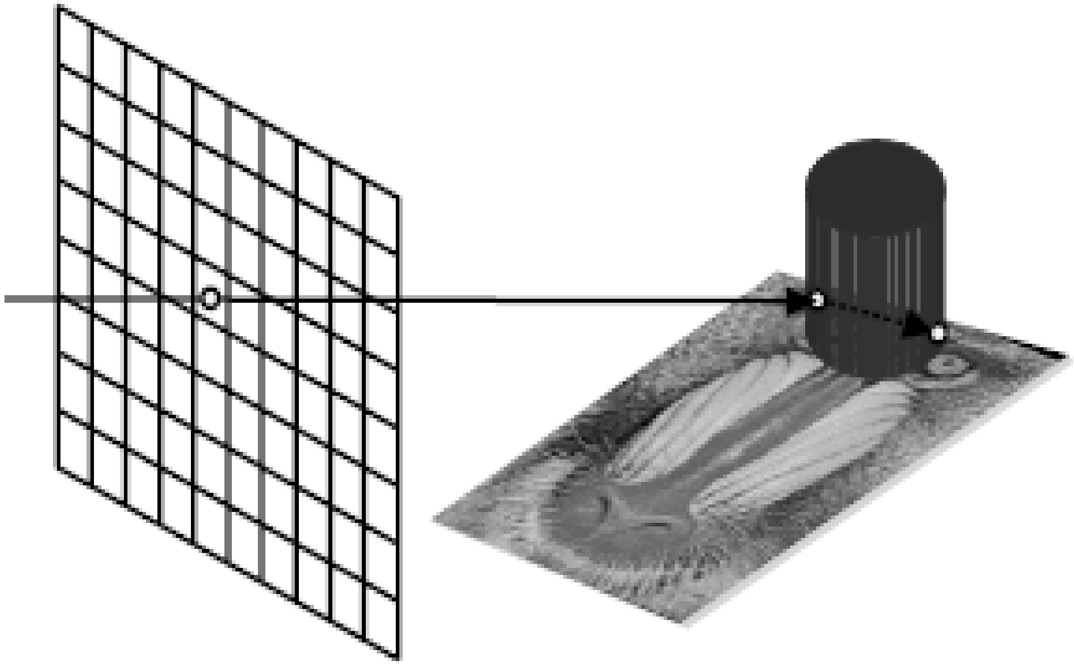


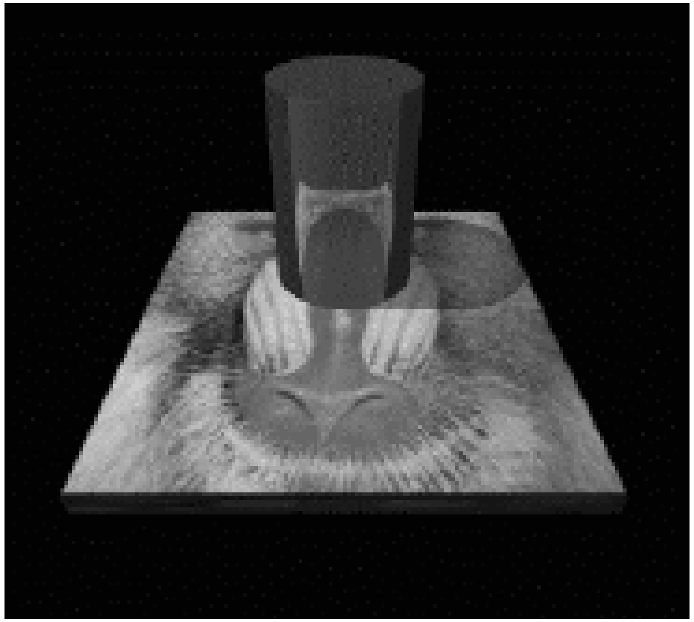


From kavita's slides

Refractions

- **If object tranparent, cast secondary refracted rays**





From kavita's slides

Generate rays for supporting effects

An Improved Illumination Model [Whitted 80]

- **Phong model**

$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j (\hat{N} \bullet \hat{L}_j) + k_s^j I_s^j (\hat{V} \bullet \hat{R})^{n_s})$$

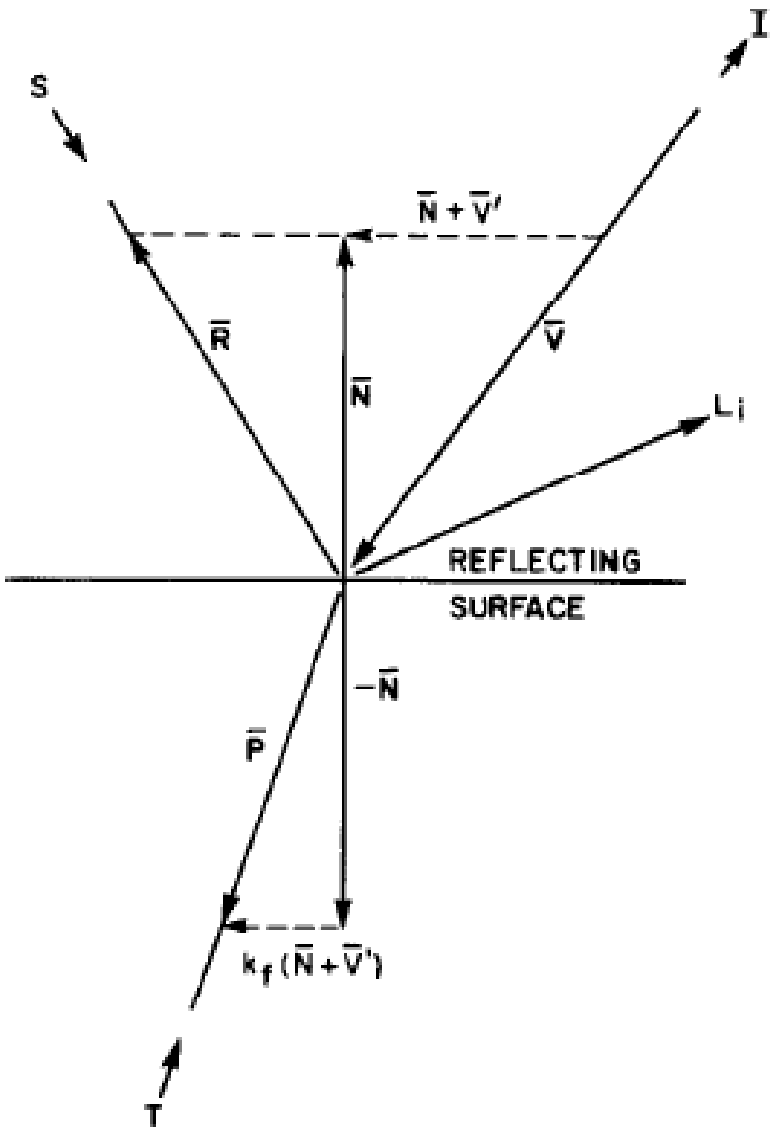
- **Whitted model**

$$I_r = \sum_{j=1}^{num\_Visible\_Lights} (k_a^j I_a^j + k_d^j I_d^j (\hat{N} \bullet \hat{L}_j)) + k_s S + k_t T$$

  - **S and T are intensity of light from reflection and transmission rays**
  - K_s, K_t **are specular and transmission coefficient**

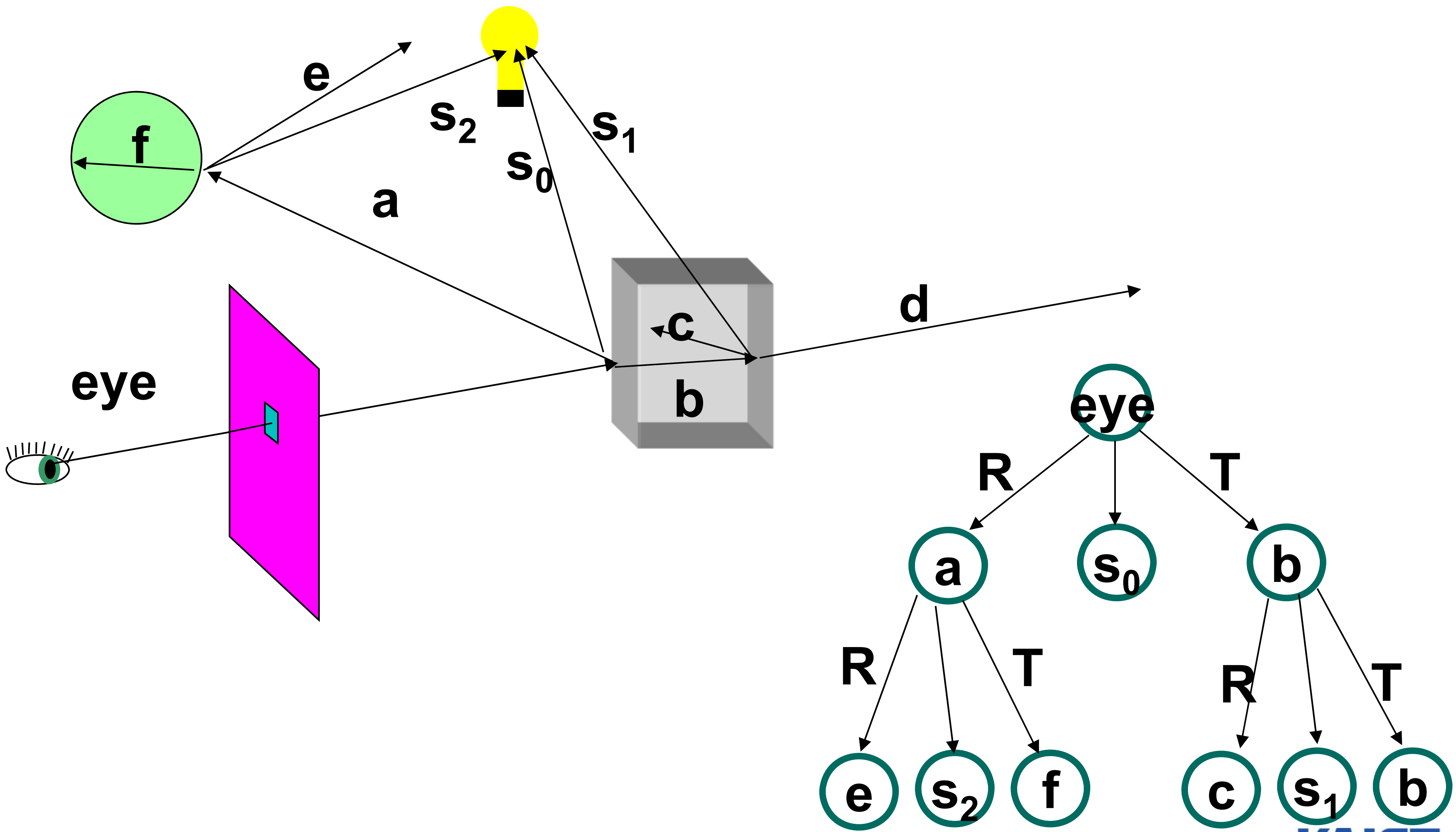
An Improved Illumination Model [Whitted 80]

$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j (\hat{N} \bullet \hat{L}_j)) + k_s S + k_t T$$



Computing reflection and transmitted/refracted rays is based on Snell's law

Ray Tree



Overall Algorithm of Ray Tracing

- Per each pixel, compute a ray, R

Def function RayTracing (R)

- Compute an intersection against objects
- If no hit,
  - Return the background color
- Otherwise,
  - Compute shading, c
  - General secondary ray, R', if necessary
  - Perform c' = RayTracing (R')
  - Return c+c'

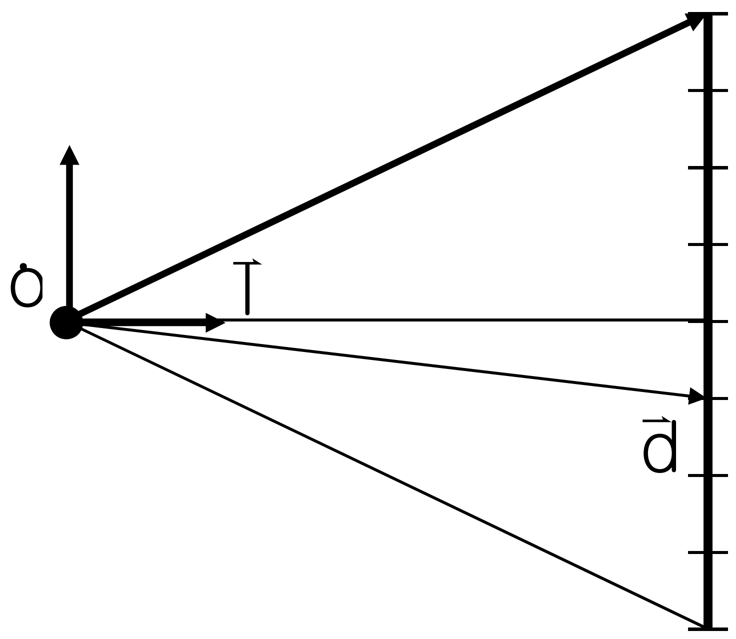
KAIST

Ray Representation

- **We need to compute the first surface hit along a ray**
  - **Represent ray with origin and direction**
  - **Compute intersections of objects with ray**
  - **Return the closest object**

$$\dot{p}(t) = \dot{o} + t\vec{d}$$

Generating Primary Rays

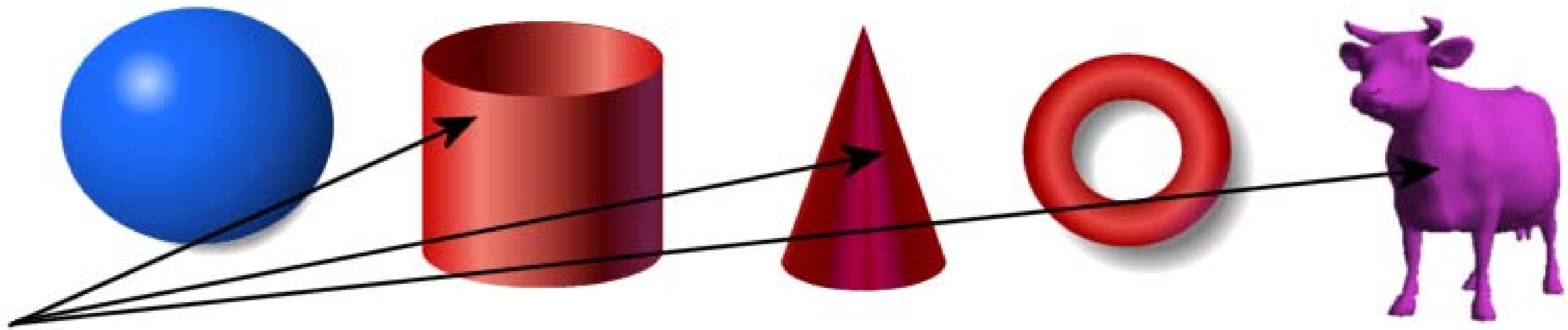


Generating Secondary Rays

- The origin is the intersection point p_0
- Direction depends on the type of ray
  - Shadow rays – use direction to the light source
  - Reflection rays – use incoming direction and normal to compute reflection direction
  - Transparency/refraction – use snell’s law

Intersection Tests

- Go through all of the objects in the scene to determine the one closest to the origin of the ray (the eye)



- Strategy
  - Solve of the intersection of the ray with a mathematical description of the object

Simple Strategy

- **Parametric ray equation**
  - **Gives all points along the ray as a function of the parameter**

$$\dot{p}(t) = \dot{o} + t\vec{d}$$

- **Implicit surface equation**
  - **Describes all points on the surface as the zero set of a function**

$$f(\dot{p}) = 0$$

- **Substitute ray equation into surface function and solve for t**

$$f(\dot{o} + t\vec{d}) = 0$$

Ray-Plane Intersection

- **Implicit equation of a plane:**

$$\vec{n} \cdot \dot{p} - d = 0$$

- **Substitute ray equation:**

$$\vec{n} \cdot (\dot{o} + t\vec{d}) - d = 0$$

- **Solve for t:**

$$t(\vec{n} \cdot \vec{d}) = d - \vec{n} \cdot \dot{o}$$

$$t = \frac{d - \vec{n} \cdot \dot{o}}{\vec{n} \cdot \vec{d}}$$

Class Objectives were:

- Understand overall algorithm of recursive ray tracing
  - Ray generations
  - Intersection tests

- Related chapter
  - Part II, Ch. 10: Ray Tracing

Next Time

- **Acceleration structures and extensions of ray tracing**

Homework

- Go over the next lecture slides before the class
- Submit questions two times during the whole semester