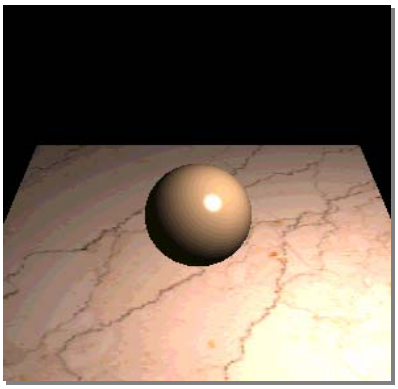
Photorealism: Ray Tracing

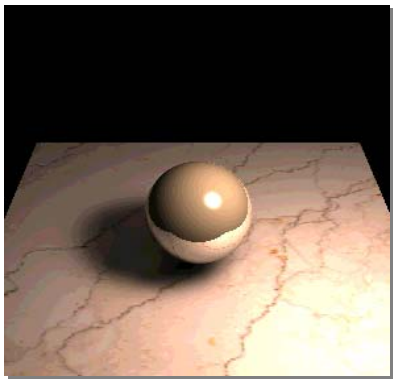
- Reading Assignment: Chapter 13

Local vs. Global Illumination



Local

Illumination depends on
local object and
light sources only



Global

Illumination at a point
can depend
on any other point in the scene

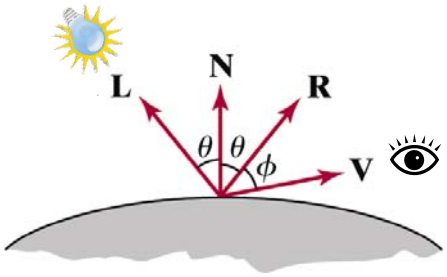
Review: Phong Illumination

- Vertex illumination from a light of intensity I
  = ambient + diffuse + specular

Ambient Term _____

Diffuse Term _____

Specular Term _____

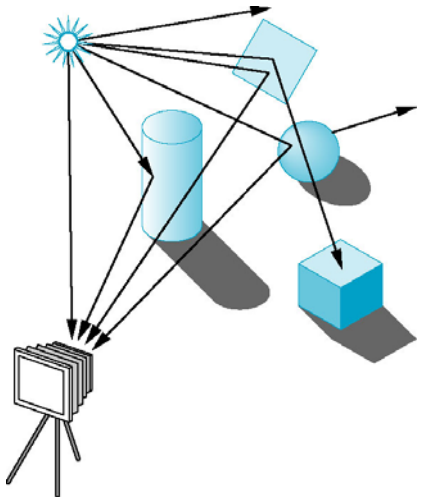


- Used to determine the color of vertices
- Used by OpenGL

- The Phong illumination method cannot generate images like these

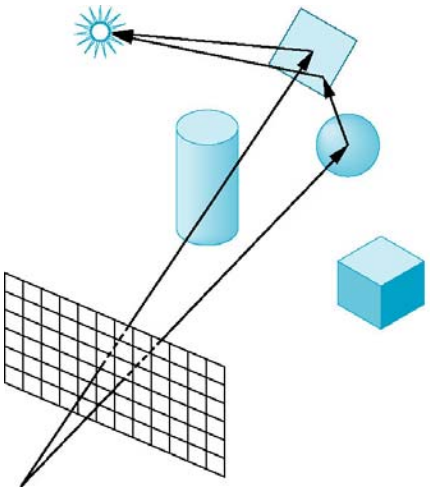
Image Formation

- Only rays that pass through lens contribute to the image.
- Most rays from source do not pass through lens
- Rays that pass through lens may bounce off other surfaces first



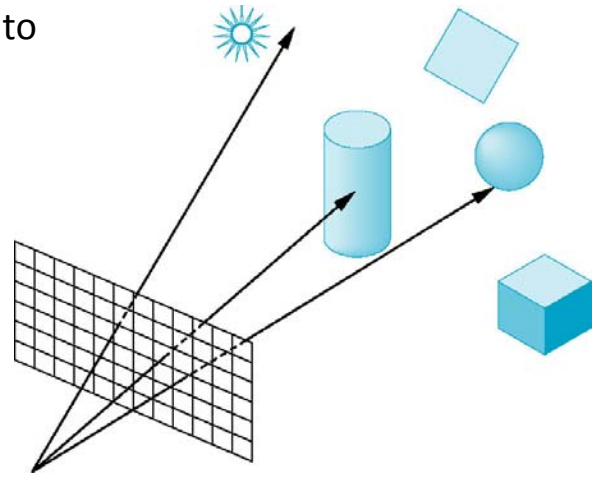
Ray Tracing

- Start at the lens and trace rays *backwards* to the light source.
- Resolve reflections and refractions to get pixel colors
- Recursive by nature
- Produces natural lighting effects
- Used in
  - Entertainment (Movies, Commercials)
  - Games pre-production
  - Simulation



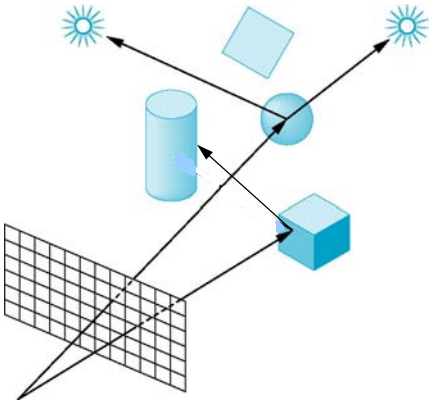
Primary Rays

- Cast a primary ray from camera position through each pixel into the scene.
- Get the first intersection.
- This process is known as ray casting.



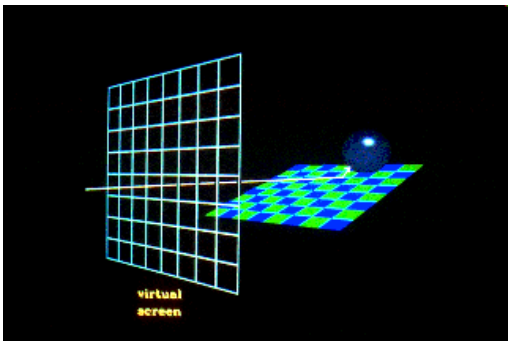
Shadow Rays

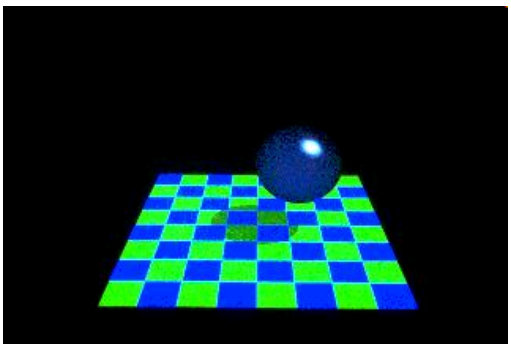
- Is the pixel in shadow, or is it visible to a light source?



- Cast shadow (feeler) rays towards each light source (diffuse term)
- If occluded, this pixel is in shadow from this light source
- Otherwise, solve the Phong model to calculate the contribution of this ray to the color of the pixel
- Each ray contributes to the color of the pixel it originates from

Shadow Ray Example

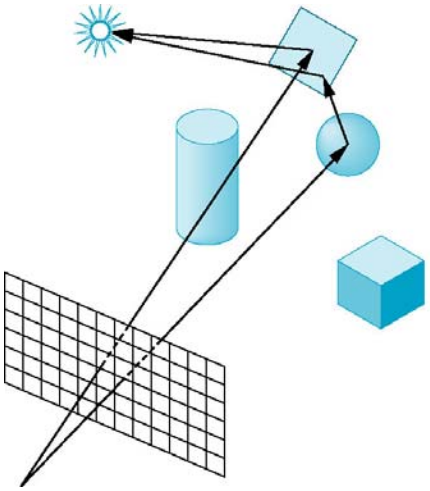




http://www.siggraph.org/education/materials/HyperGraph/raytrace/rtrace1.htm

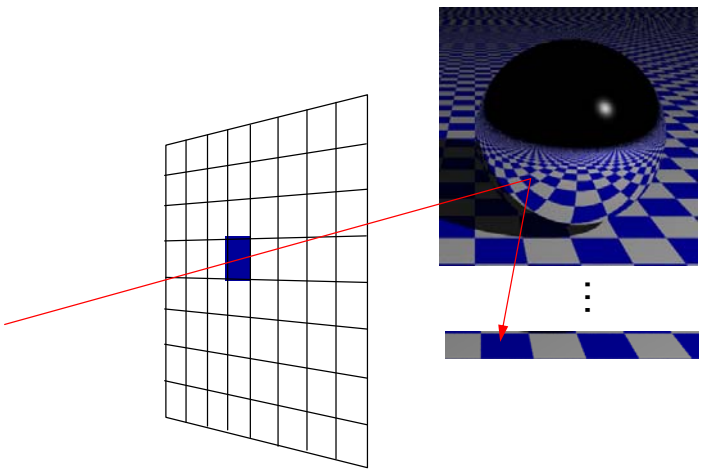
Reflected Rays

- When a ray hits an object, a reflected ray is generated.



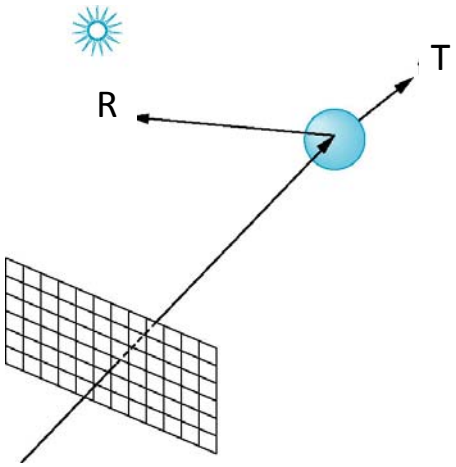
- If the reflected ray hits an object, the Phong illumination model is applied at the point of intersection. The result is carried back to the first intersection point.
- Process is recursive
  - First order reflections
  - Second order reflections, etc.

Reflected Ray Contribution Example



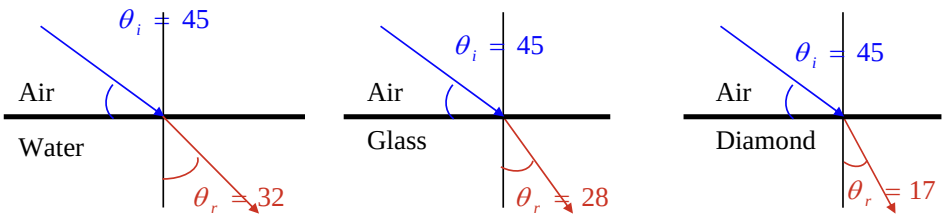
Refracted Rays

- If the object is transparent, then a transmitted (or refracted) ray is generated.
- If the transmitted ray hits an object, the Phong illumination model is applied at the point of intersection. The result is carried back to the first intersection point.



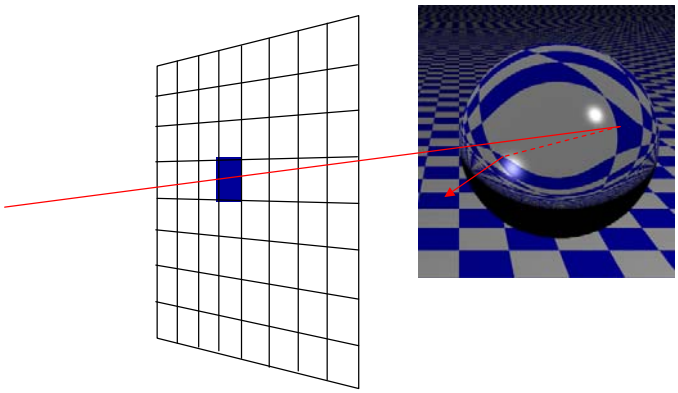
Background: Angle of Refraction

- When light passes from a material of one optical density to another it changes direction
- The amount by which the direction changes is determined by the optical densities of the two media



- Optical density (and thus the amount of bending) is related to a value we call the *refractive index* of the material.

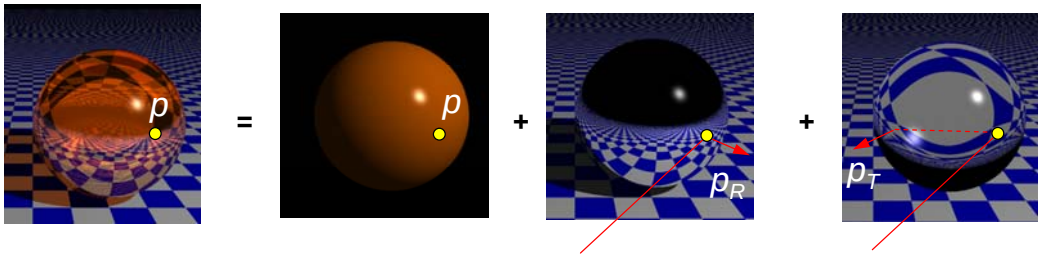
Refracted Ray Contribution Example



Putting It Together …

$$I(p) = I_{local}(p) + k_R I(p_R) + k_T I(p_T)$$

Local term (Phong) — Reflected — Transmitted

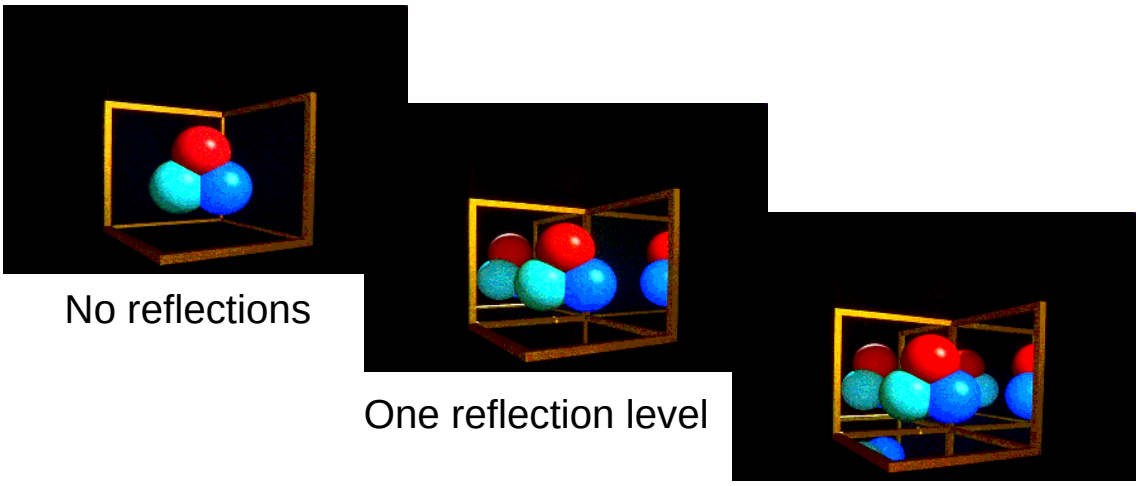


$I(p_R)$ is the intensity of from the reflected ray

$I(p_T)$ is the intensity of light from the transmitted ray

k_R and k_T are the reflection and transmission coefficients

Multiple Reflections



No reflections

One reflection level

Two reflection levels

Ray Tracing Algorithm

For each pixel

- Cast a primary ray and find closest intersection *p*
- If intersect something
  pixelColor = ComputeShade(*p*)

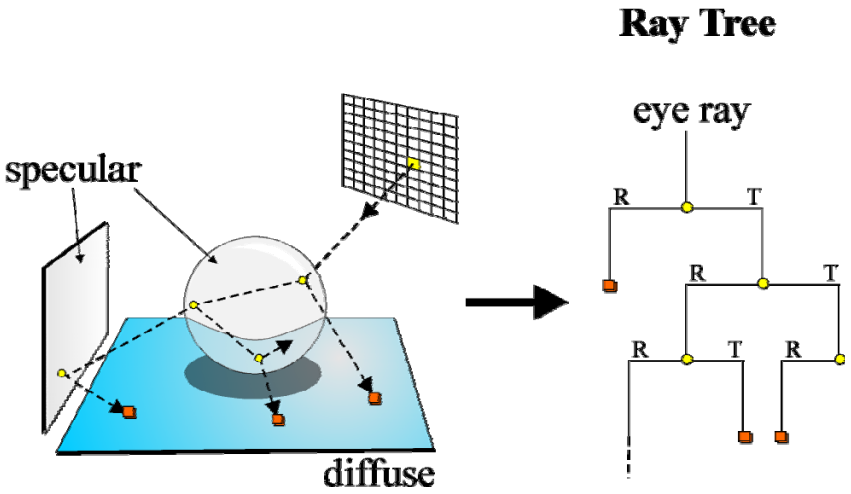
color ComputeShade(*p*)

1. Extend shadow (feeler) ray and find closest intersection
2. Calculate Phong *local intensity I* (ambient term if intersection, diffuse + ambient term if no intersection)
3. If reflective
   Form reflective ray and find closest intersection p_R
   I_R = ComputeShade(p_R)
4. If transparent
   Form transmission ray and find closest intersection p_T
   I_T = ComputeShade(p_T)
5. return $I + k_R I_R + k_T I_T$

Demo @ http://www.siggraph.org/education/materials/HyperGraph/raytrace/rt_java/raytrace.html

Ray Trees

- The spawning of reflected and refracted rays can be represented in *tree* structure.

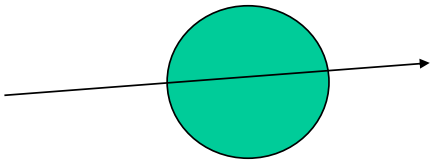


Terminating Recursion

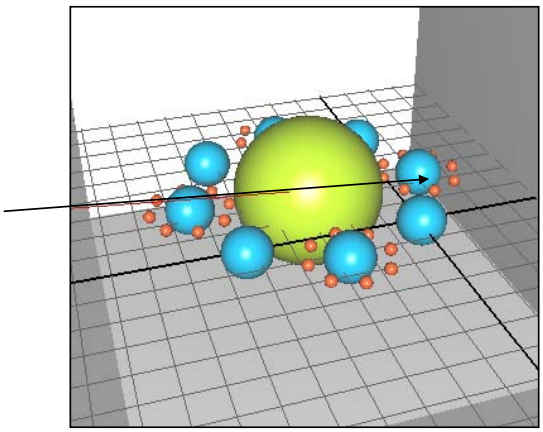
- All of the spawned rays contribute to the pixel that the tree originated from.
- However each new ray contributes less and less to the pixel.
- Unlike in the real-world, we can't keep bouncing around forever, so we stop the recursion at some stage (*recursion clipping*):
  - Stop after a set number of bounces (6, for example).
  - Or stop when contribution becomes less than a certain value.

Ray Tracing

- Most of the time is spent calculating intersections!!!
- Inherently difficult geometric problem.



Where does the ray intersect the object?



Where does the ray intersect the scene?

Ray – Sphere Intersection

- Parametric equation for ray passing through the point $s = (s_x, s_y, s_z)$ in the direction of unit vector $d = (d_x, d_y, d_z)$:

- Equation for unit circle with center $c = (c_x, c_y, c_z)$:

- Compute intersection point:

Shading Calculations (Diffuse Reflections)

- To do shading calculations at a point $p = (p_x, p_y, p_z)$ on a sphere, we need the surface normal vector at p:

- If θ is the angle between surface normal and the direction of the light source, then the Phong model gives the color of the reflected light as

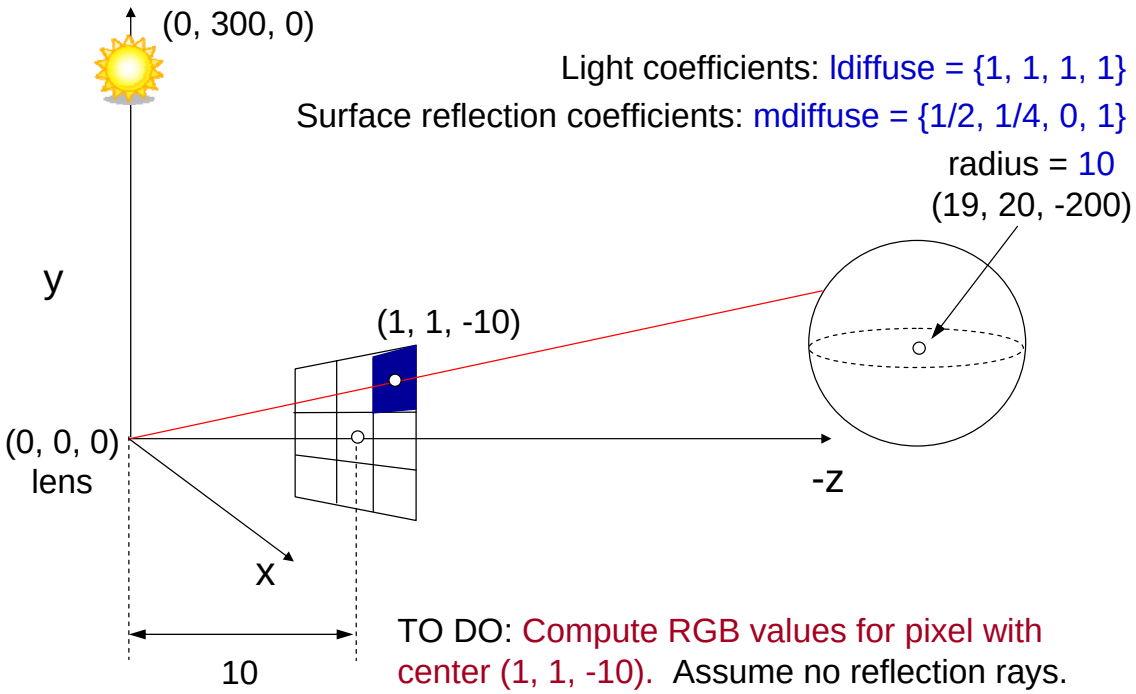
  R =

  B =

  G =

  Use the reflective RGB coefficients k_{rd}, K_{bd}, k_{gd} and light source RGB coefficients L_{rd}, L_{bd}, L_{gd}.

Object Intersections and Lighting Practice



TO DO: Compute RGB values for pixel with center (1, 1, -10). Assume no reflection rays.

Ray – Triangle Intersection

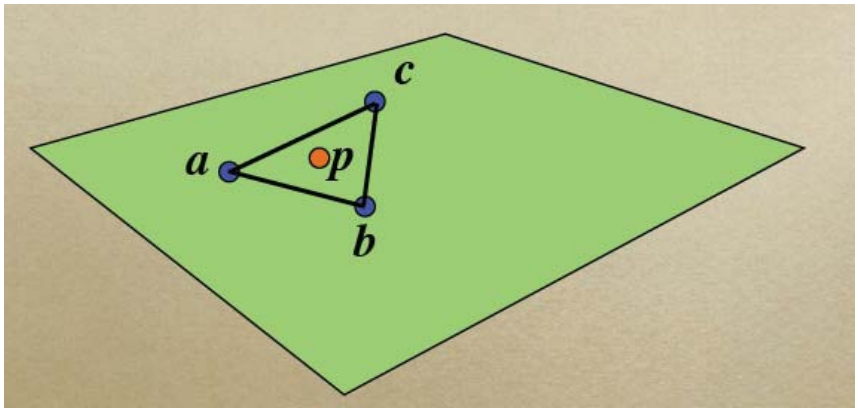
- A triangle can be represented by three vertices a, b, c.
- Using cross products:
  - Recall that the cross product of two vectors in 3D is another vector perpendicular to the original two.
  - The cross product of two parallel vectors is _____
- Then:
  - Normal n to the triangle is n =
  - For any point *p* in the triangle, n.(b-*p*) = _____
- ray passing through the point s = (s_x, s_y, s_z) in the direction of unit vector d = (d_x, d_y, d_z):

Ray – Triangle Intersection

- Compute the point where the ray r = s + td intersects the plane containing Δabc:

- What if n.d = 0? What if b = s?

Have we hit the triangle?



Inside and Outside

(b - a) x (p - a), (c - b) x (p - b), (a - c) x (p - c)
all up or all down means p is inside Δabc:

(b - a) x (p - a) • n,
(c - b) x (p - b) • n,
(a - c) x (p - c) • n

all >0 or all <0

Ray – Triangle Intersection

- Harder than a sphere?

YES, and it takes longer to compute.