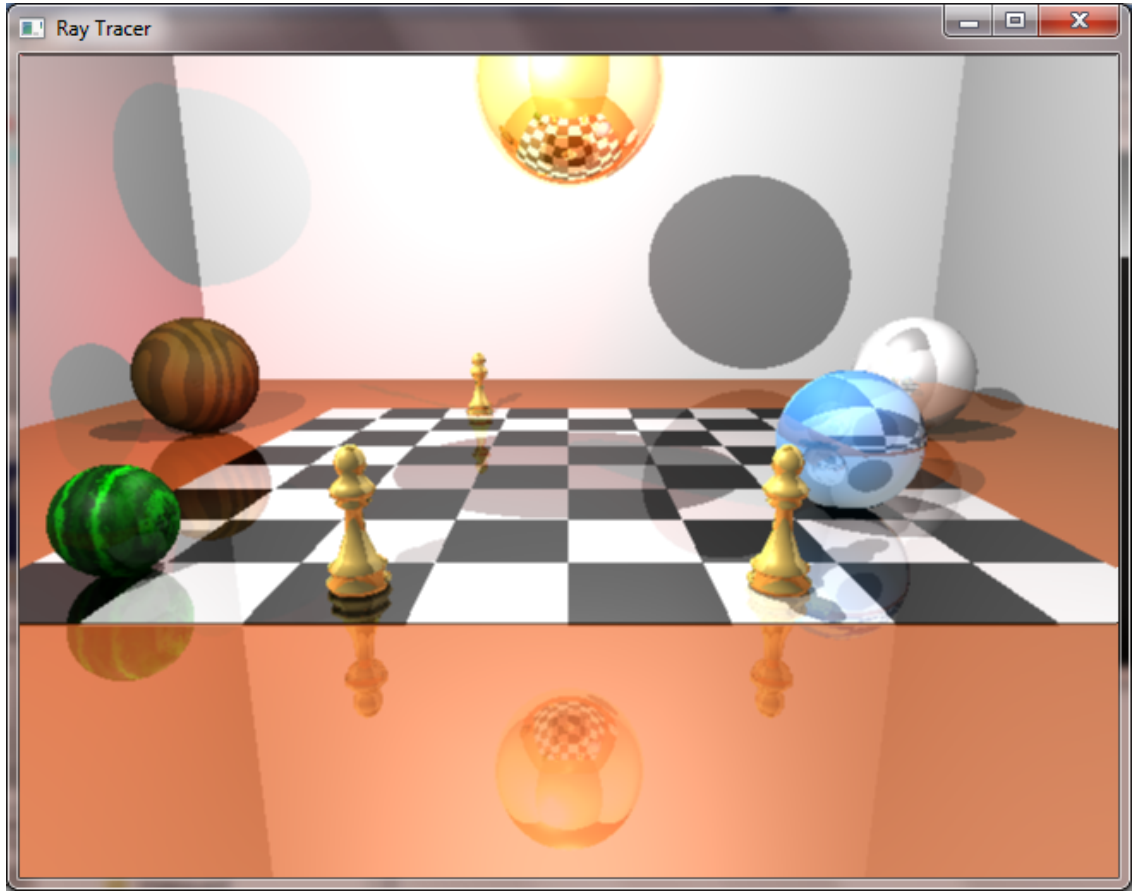
Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

Module 2C
Recursive Ray Tracing

Ray Tracing



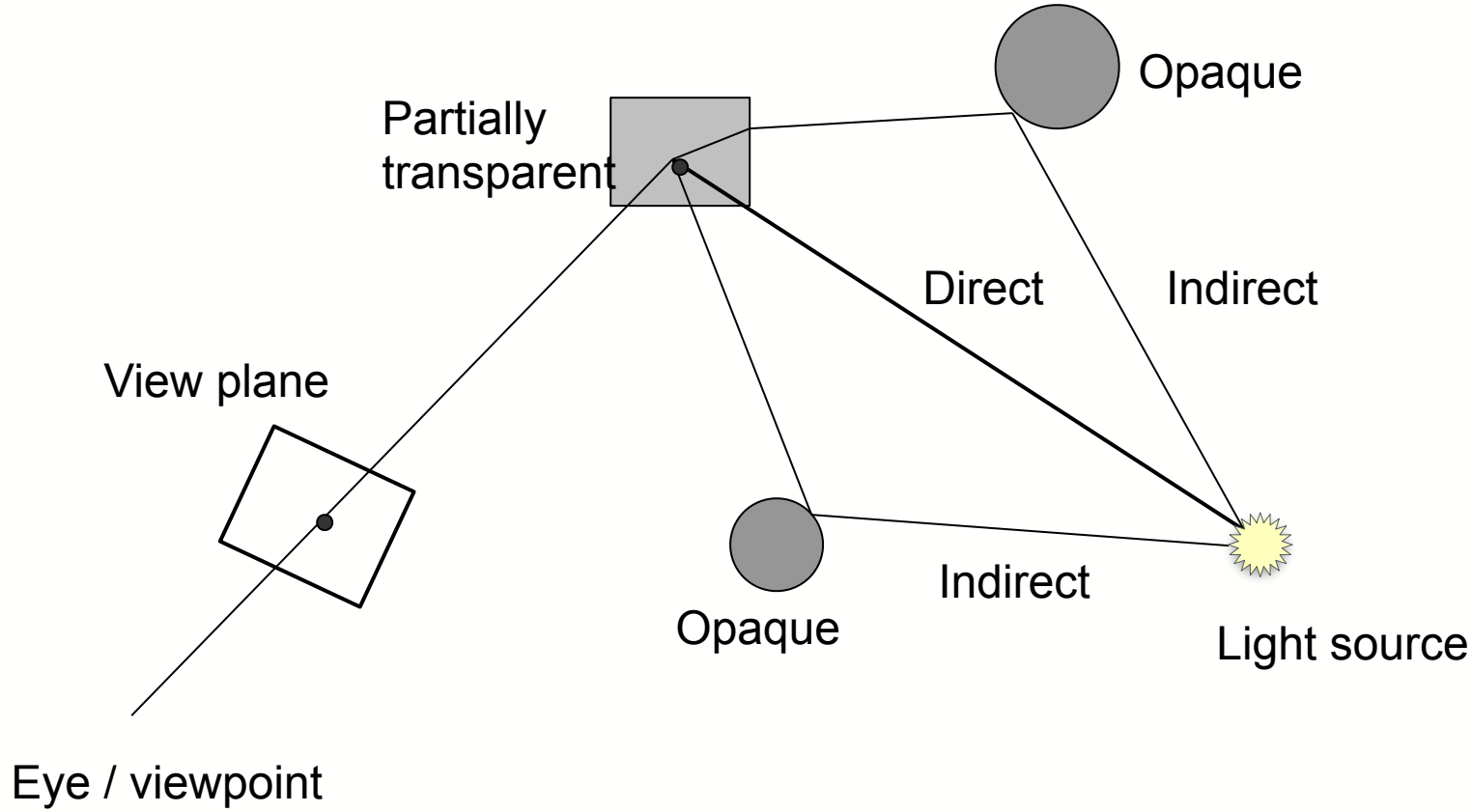
Sample Lab - 2008

Recursive Ray-Tracing

- Recursive ray tracing conditionally spawns **reflection rays**, **refraction rays**, and **shadow rays** from the point of intersection
  - Shadow, reflection, and refraction rays are called secondary rays
- Each reflection and refraction ray may in turn recursively spawn additional reflection, refraction, and shadow rays
  - As the spawned rays intersect other objects in the scene
  - Shadow rays do not spawn additional rays
  - Form a **ray-tree**
- Due to the recursive nature, ray tracing is often implemented as a recursive algorithm
  - Recursive procedures have own private copy of parameters and local variables

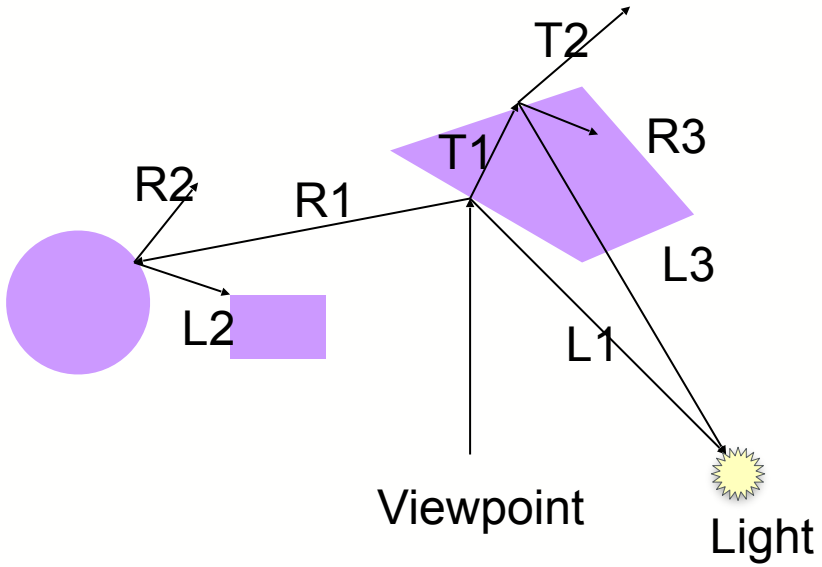
Ray Tracing Diagram

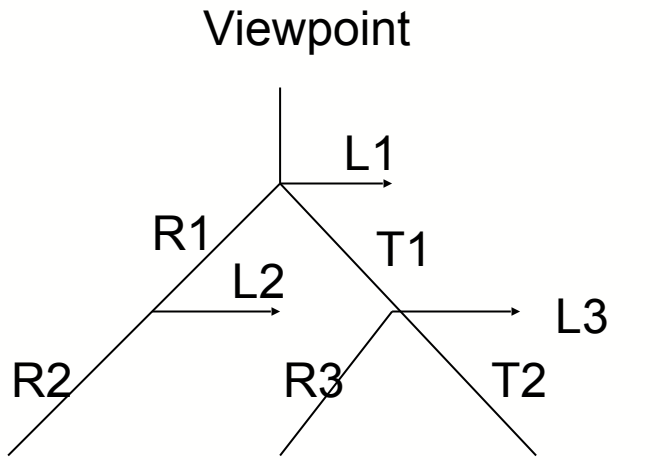
- Illustration of simple ray tracing



Tree Structure of Ray-Tracing

- Ray-tracing recursions can be illustrated as a tree
  - Nodes are intersect points
  - R - reflected ray
  - L - shadow ray
  - T - transmitted (refracted) ray
- Halt procedure when a specified depth is reached
  - Or when the ray intersects no objects in the scene





Halt processing at this depth

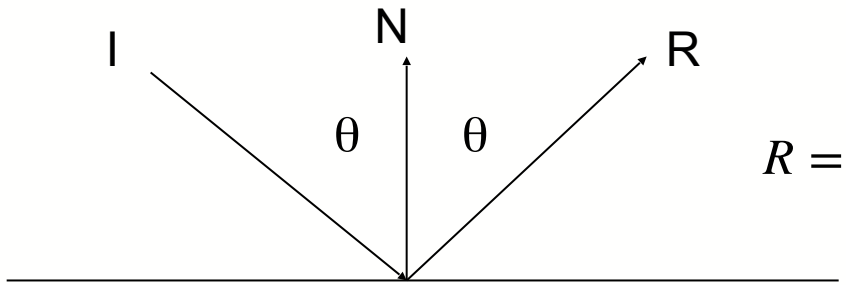
Ray Tracer Pseudo Code

```
Color TraceRay(Ray& ray)  {
   Get the nearest positive intersection - populate hitInfo
   Get the material definition for hit
   Add emissive and ambient component to color
   for each light source
      if not in shadow: (generate shadow ray to determine if in shadow)
         add ambient (due to light source), reflected and specular local illumination to color
   if (recursionLevel == maxRecusion)
      return color
   if (material is reflective)             // Spawn reflection ray
   {
      reflectedRay = build reflected ray (increment recursion level)
      color += (material reflective coefficient * TraceRay(reflectedRay)
   }
   if (material is transparent)
   {
      if (not total internal reflection)   // Spawn refraction ray
      {
         refractedRay = build refracted ray (increment recursion level)
         color += (material transparency coefficient * TraceRay(reflectedRay)
      }
   }
   return color
}
```

Reflection

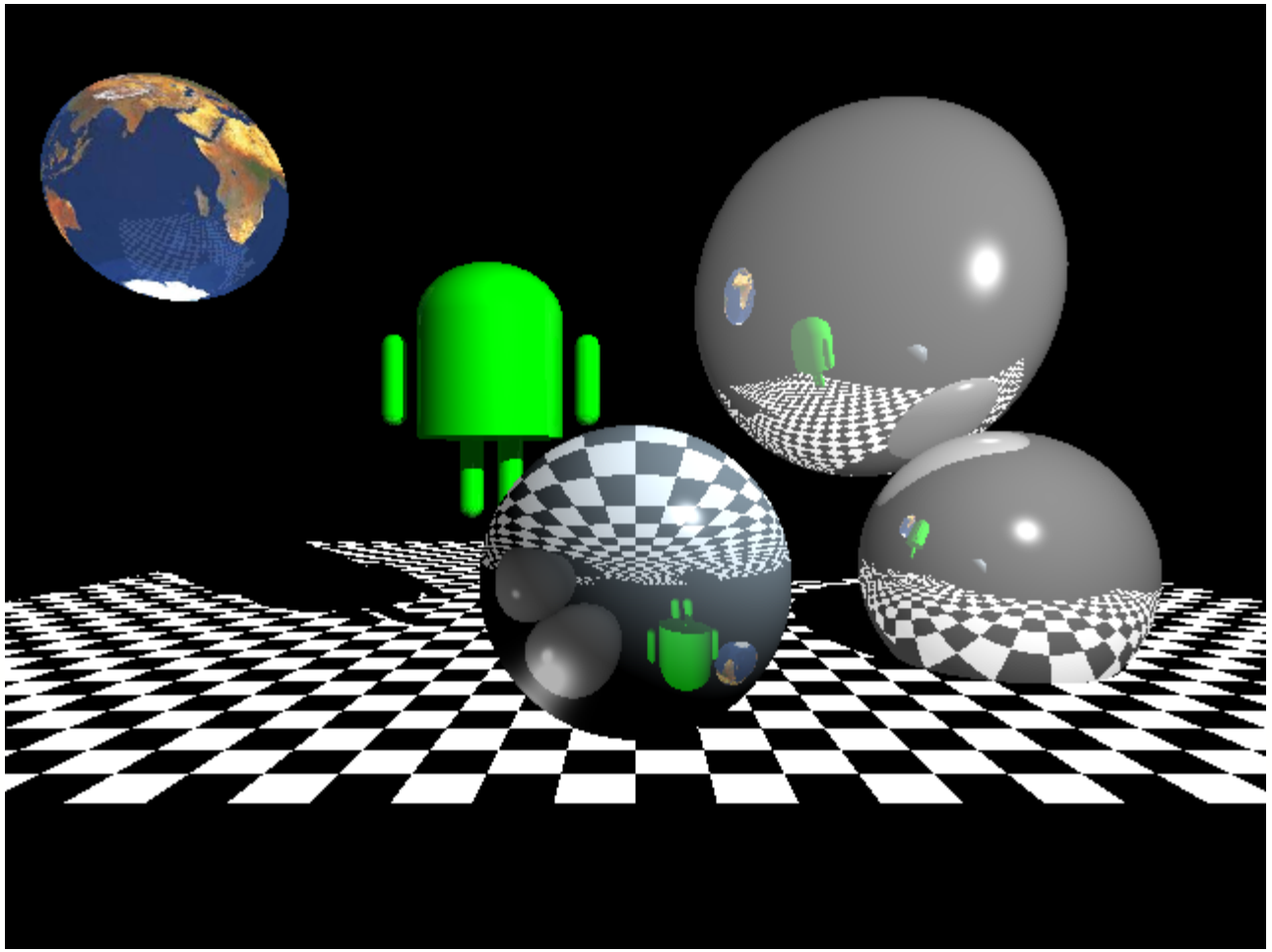
- When a ray strikes a surface it produces a reflected ray
  - If material is reflective
- Reflection vector
  - Angle of incidence = angle of reflection

    $R = I + 2N\cos\theta = 2(N \cdot L)N - L$
    - I is a unit vector representing the incident ray direction
    - I = -L where L is the light vector from the point to the light
    - N is the unit surface normal
  - R, N, and I are coplanar
- Vector3::Reflect – already coded from last semester



$$R = I + 2N\cos\theta = I - 2(I \cdot N)N$$

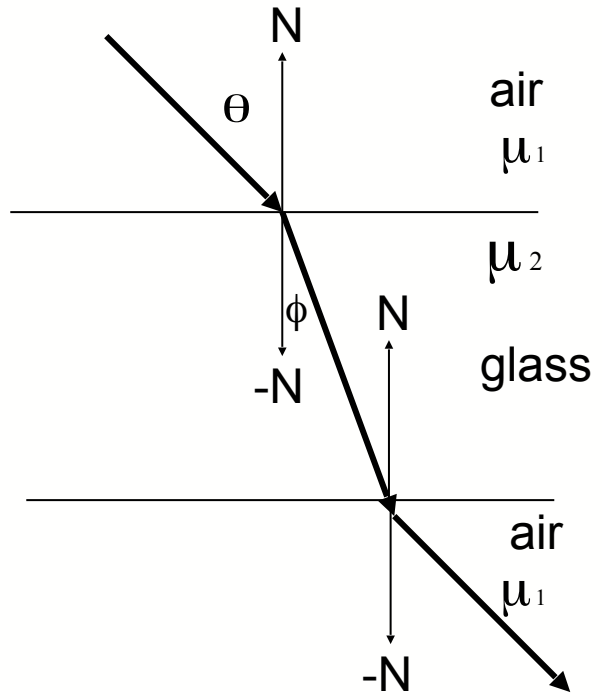
Refraction



https://stackoverflow.com/questions/15846867/glossy-reflection-in-ray-tracing

Refraction

- A ray striking a transparent surface is refracted or bent as it is transmitted through the material
  - Change in velocity of light in different media
- Angles of incidence and refraction are given by Snell's Law
  - $\mu_1 \sin\theta = \mu_2 \sin\varphi$
  - μ are the indices of refraction
    - Comparing speed of light in vacuum to speed of light in the material
- Moving from medium where light travels faster than the medium it is entering
  - Light bends (refracts) towards the normal
  - Example: from air into water
- Refracts away from the normal if moving into medium where light travels faster



Refraction Vector

- Transmitted ray T is given by:

$$T = \mu I - \left(\mu\left(I \cdot N\right) + \cos\varphi\right)N$$

$$\mu = \frac{\mu_1}{\mu_2} \qquad \cos\varphi = \sqrt{1 - \mu^2\left(1 - (I \cdot N)^2\right)}$$

I and *N* are both unit vectors

Derivation:

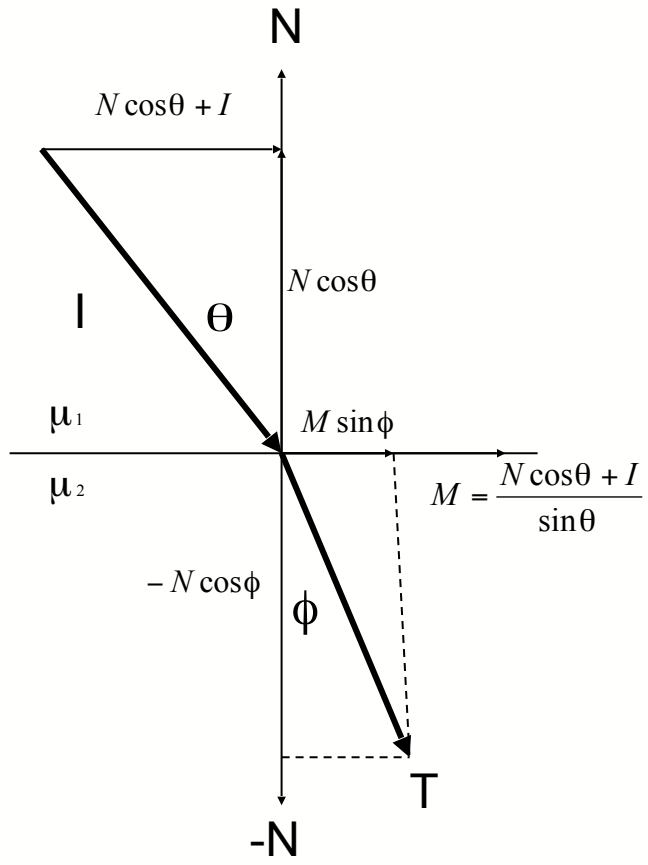
$$T = M\sin\varphi - N\cos\varphi$$

M is unit vector perpendicular to N in the plane of *I* and *T*

$$T = \frac{\sin\varphi}{\sin\theta}\left(N\cos\theta + I\right) - N\cos\varphi$$

Since $\mu = \frac{\mu_1}{\mu_2} = \frac{\sin\varphi}{\sin\theta}$ we get $T = \left(\mu\cos\theta - \cos\varphi\right)N + \mu I$

$$\cos\theta = -(I \cdot N) \qquad \cos\varphi = \sqrt{1 - \sin^2\varphi} = \sqrt{1 - \mu^2\sin^2\theta} = \sqrt{1 - \mu^2\left(1 - (I \cdot N)^2\right)}$$



Note: if you follow derivation in Foley and van Dam (and some on line sites) the vector *I* is reversed.
I prefer working with the incoming ray direction.

Index of Refraction

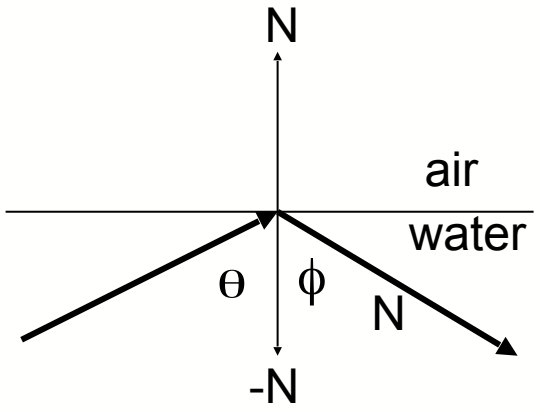
- Vacuum has refractive index = 1.0
  - Other media are higher
- Absolute refractive index
  - Ratio of speed of light in vacuum over speed of light in material 2

Vacuum	1
Air	1.0003
Water – ice	1.31
Water – liquid	1.333
Ethyl alcohol	1.36
Glass	1.51
Quartz	1.54
Polycarbonate	1.59
Dense glass	1.66
Ruby	1.76
Garnet	1.80
Diamond	2.417

From various sources including Wikipedia

Total Internal Reflection

- Light passing from one medium to another whose index of refraction is lower
  - If θ becomes sufficiently large then φ exceeds 90 degrees
    - Ray becomes reflected back into the media and is not transmitted
    - Known as total internal reflection
- θ_c is called the critical angle
  - Above this angle total internal reflection occurs
  - Critical angle is value θ at which sin φ = 1
  - $\theta_c = \sin^{-1}\left(\frac{\mu_2}{\mu_1}\right)$
- To detect Total Internal Reflection
  - $\cos\varphi = \sqrt{1-\mu^2\left(1-(I\cdot N)^2\right)}$ is imaginary
    - Value under sqrt is less than 0
  - Do not spawn a refractive ray



Handling Refraction in a Recursive Ray Tracer

- When tracing scenes that include translucent objects we must keep track of the medium through which the ray is passing
  - So we can determine the transmitted ray direction: depends on μ
- If translucent objects cannot interpenetrate
  - Ray is either in air alone or inside a single object
    - e.g., cannot have water inside a drinking glass
  - Alternates between inside and outside a translucent object
    - Store a flag indicating inside an object, and store its refractive index
    - When inside, the next intersection will generate a ray into air
  - When the ray is inside the object
    - Need to reverse the sign of the normal components when generating reflected and transmitted rays
    - Normally do not add any local color contribution at the intersect point
    - If the angle of incidence is less than the critical angle spawn a refracted ray
- Interpenetrating translucent objects
  - Track how many objects the ray is inside and keep an array of objects/materials
    - See F.S. Hill or Watt

Shadows

Example using POV-Ray: http://members.tripod.com/~azzazzel/page_P1.htm

Shadows

- Discussion until now: I_{local} is computed at each intersection assuming that light vectors L do not intersect other objects in the scene
- Shadow determination: find if objects lie in the path of L by creating additional rays
  - Called shadow feelers
  - From the intersect point to each light source
- I_{local} is reduced to ambient value if a totally opaque object lies in the path of a shadow feeler
- Shadow feelers do not spawn other rays as they intersect objects
- Attenuation of I_{local} may be calculated if partially transparent object is intersected by the shadow feeler

Shadows (cont.)

- As number of light sources increase, shadow testing becomes more computationally intensive
  - Each main ray spawns n shadow feelers
    - n is the number of light sources
  - Standard ray tracer: each ray may spawn n+2 rays
    - Reflected, refracted and n shadow feelers
- Slightly less object intersection costs compared to the main ray
  - Do not need to find the closest intersecting object
    - Only if an intersection occurs