Ray Tracing

Course web page:
http://goo.gl/EB3aA

Outline

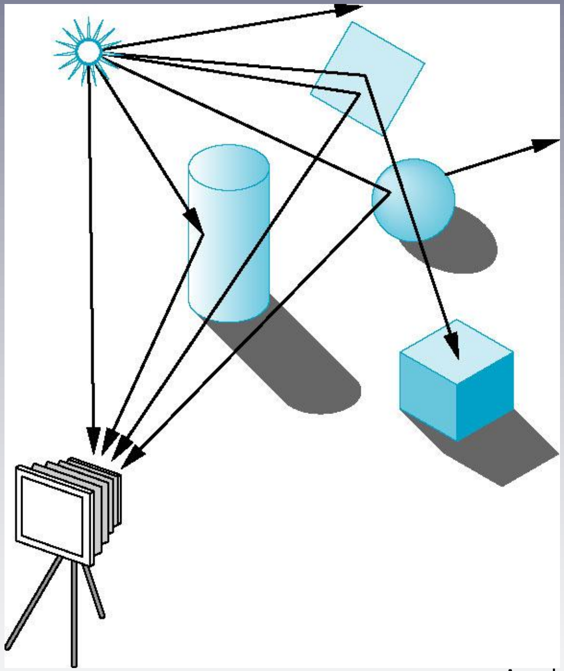
- Global illumination with ray casting & ray tracing
- Ray casting
  - Intersections
  - Shadow rays

Illumination models

- Interaction between light sources and objects in scene that results in perception of intensity and color at eye
- Local vs. global models
  - Local illumination: Perception of a particular primitive only depends on light sources **directly** affecting that one primitive
    - Geometry
    - Material properties
  - Global illumination: Also take into account **indirect** effects on light of other objects in the scene
    - Shadows cast
    - Light reflected/refracted

"Forward" Ray Tracing

- Proper global illumination means simulation of physics of light
  - Rays are emitted from light source, bounce off objects in the scene, and some eventually hit our eye, forming an image
- Problem: Not many rays make it to the image
  - Waste of computation for those that don't

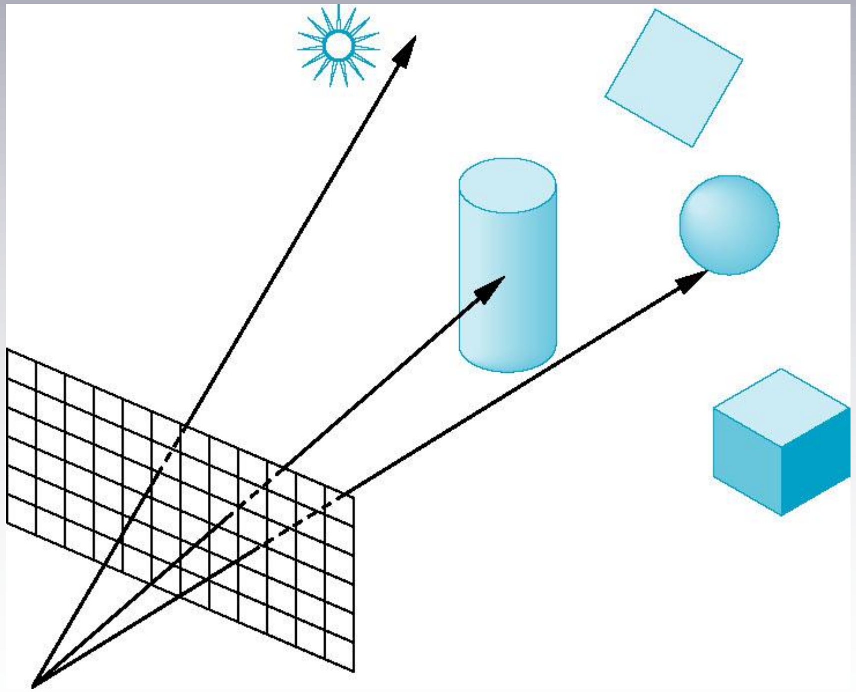


Angel

University of Delaware

“Backward” Ray Tracing

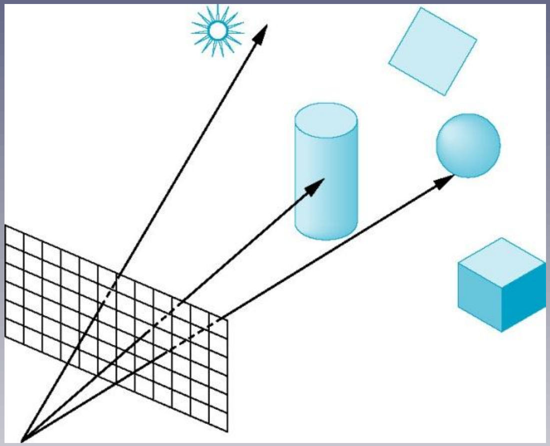
- Idea: Only consider those rays that **do** create the image—where did they come from?

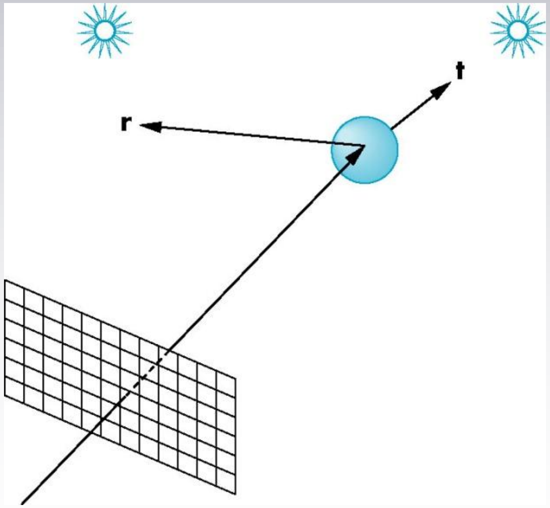


Angel

Backward Ray "Following": Types

- **Ray casting**: Compute illumination at first intersected surface point only
  - Takes care of hidden surface elimination
- **Ray tracing**: Recursively spawn rays at hit points to simulate reflection, refraction, etc.

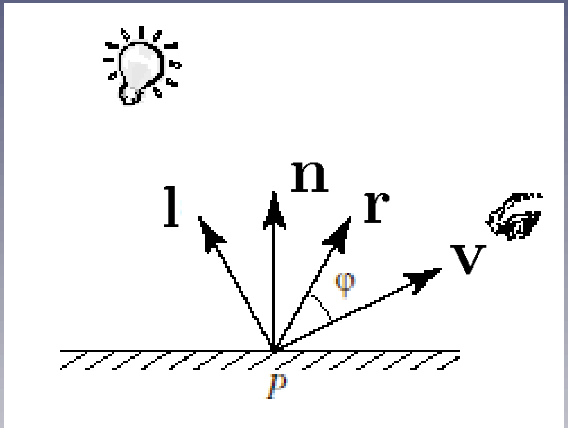




Angel

Lighting a point

- Let **c** = (r, g, b) be **perceived** material color (called **i** on previous slides), **s**(l) be color of light *l*
- Sum over all lights *l* for each color channel (clamp overflow to [0, 1]):



from Hill

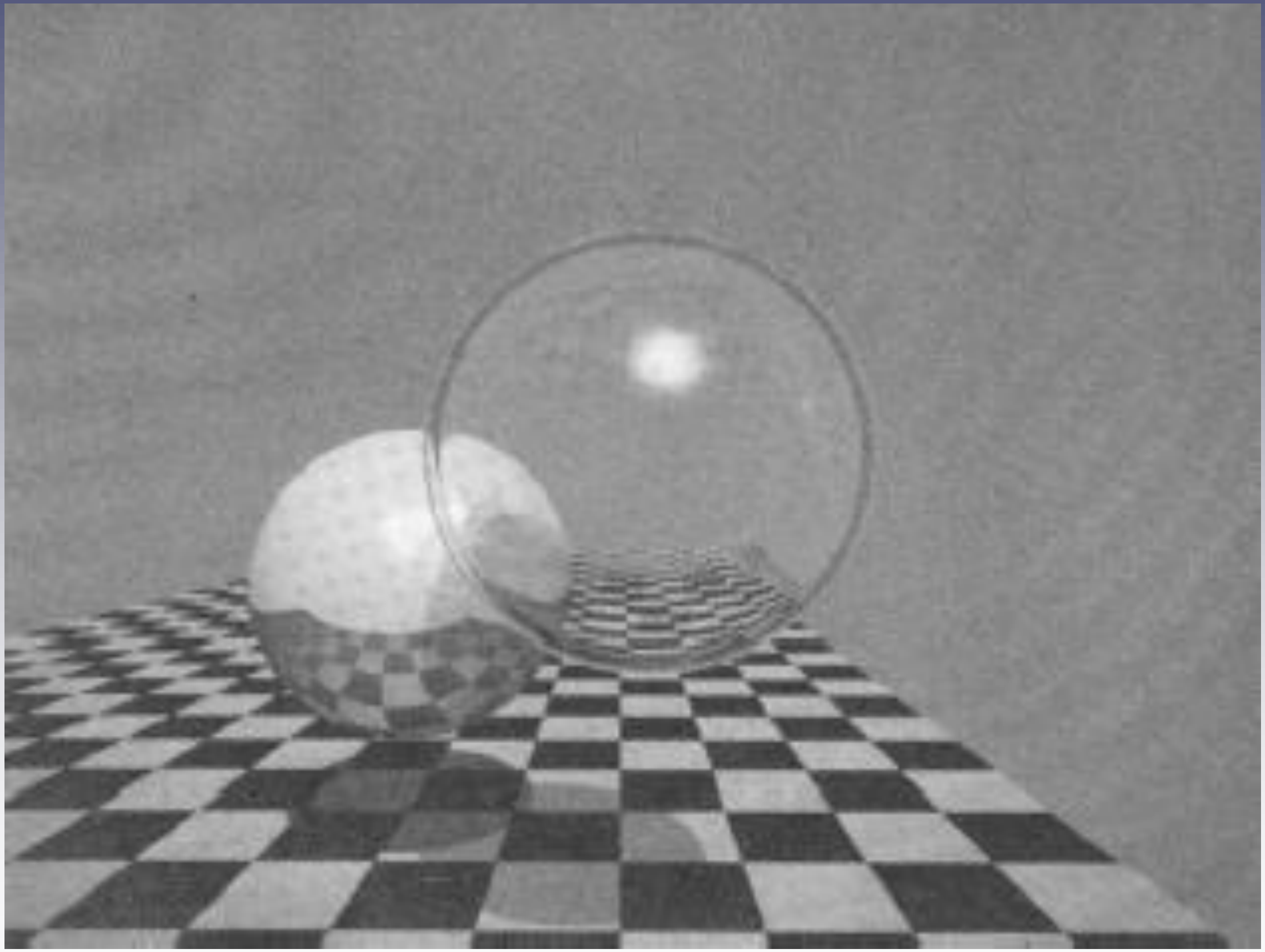
$$\mathbf{c}_{total} = \sum_{l} \mathbf{c}_{amb}(l) + \mathbf{c}_{diff}(l) + \mathbf{c}_{spec}(l)$$

$$\mathbf{c}_{amb}(l) = \mathbf{m}_{amb} \otimes \mathbf{s}_{amb}(l)$$

$$\mathbf{c}_{diff}(l) = \max(0, \mathbf{n} \cdot \mathbf{l}(l))\mathbf{m}_{diff} \otimes \mathbf{s}_{diff}(l)$$

$$\mathbf{c}_{spec}(l) = \max(0, \mathbf{v} \cdot \mathbf{r}(l))^{shine}\mathbf{m}_{spec} \otimes \mathbf{s}_{spec}(l)$$

One of the earliest ray-traced scenes



from T. Whitted's paper

State of the art in 1980
(B & W reproduction of a color image)

Ray Tracing: Example

Ray Tracing: Example from "Cars"

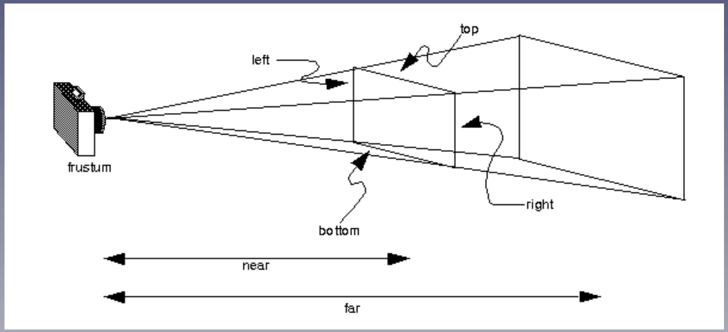
http://www.sci.utah.edu/~wald/RT06/papers/raytracing06per.pdf

Ray Casting

- Simulation of irradiance (incoming light ray) at each pixel
- Send a ray from the focal point through each pixel and out into the scene and see if it **intersects** an object
  - "Background" color if nothing hit
- Local shading model is applied to **first** point hit
  - Easy to apply exact rather than faceted shading model to objects for which we have an analytic description (spheres, cones, cylinders, etc.)

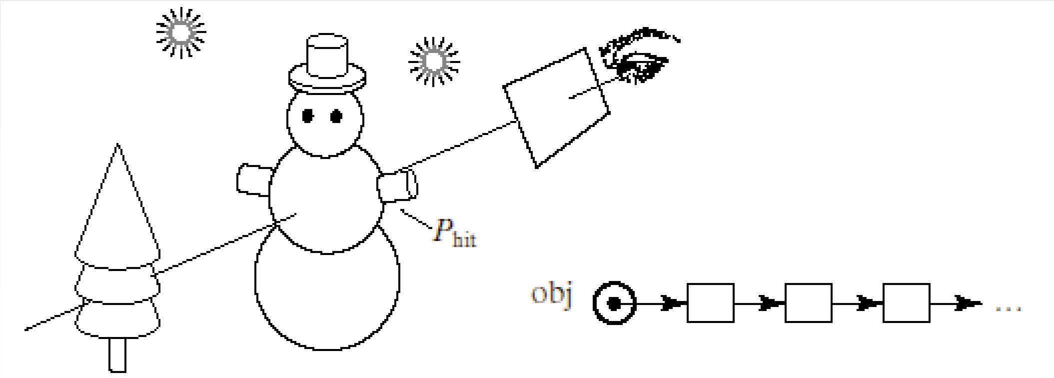
Ray Casting: Details

- Must compute 3-D ray into scene for each 2-D image pixel (Chap. 4.3 of Shirley)



from Woo *et al.*

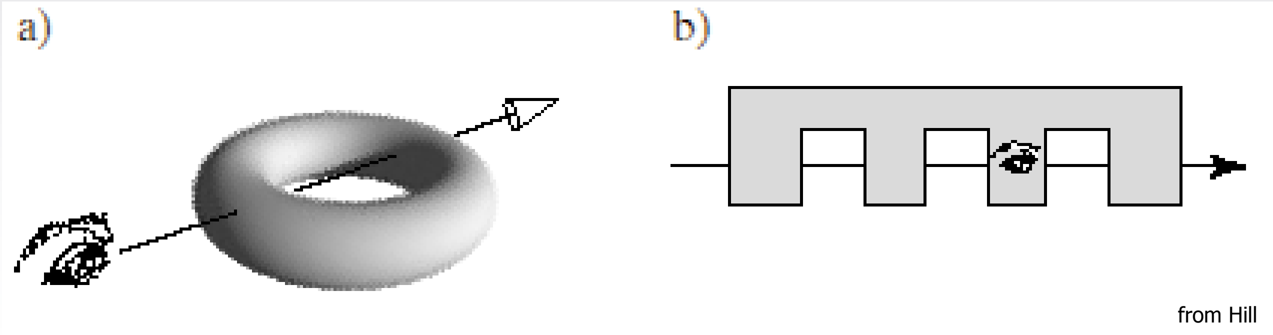
- Compute 3-D **position** of ray's intersection with nearest object and **normal** at that point
- Apply shading model such as Phong to get color at that point and fill in pixel with it



from Hill

Does Ray Intersect any Scene Primitives?

- Test each primitive in scene for intersection individually
- Different methods for different kinds of primitives
  - Polygon
  - Sphere
  - Cylinder, torus
  - Etc.
- Make sure intersection point is **in front of eye** and **nearest one**



from Hill

Ray-Sphere Intersection I

- Combine implicit definition of sphere

$$|\mathbf{p} - \mathbf{p}_c|^2 - r^2 = 0$$

with ray equation

$$\mathbf{p} = \mathbf{o} + t\mathbf{d}$$

(where **d** is a unit vector) to get:

$$|\mathbf{o} + t\mathbf{d} - \mathbf{p}_c|^2 - r^2 = 0$$

Ray-Sphere Intersection II

- Substitute $\Delta\mathbf{p} = \mathbf{p}_c - \mathbf{o}$ and use identity
  $$|\mathbf{a} + \mathbf{b}|^2 = |\mathbf{a}|^2 + 2\mathbf{a}\cdot\mathbf{b} + |\mathbf{b}|^2$$
  to solve for t, resulting in a quadratic equation with roots given by:

$$t = \mathbf{d}\cdot\Delta\mathbf{p} \pm \sqrt{(\mathbf{d}\cdot\Delta\mathbf{p})^2 - (|\Delta\mathbf{p}|^2 - r^2)}$$

- Notes
  – Real solutions mean there actually are 1 or 2 intersections
  – Negative solutions are behind eye

Ray-Polygon Intersection

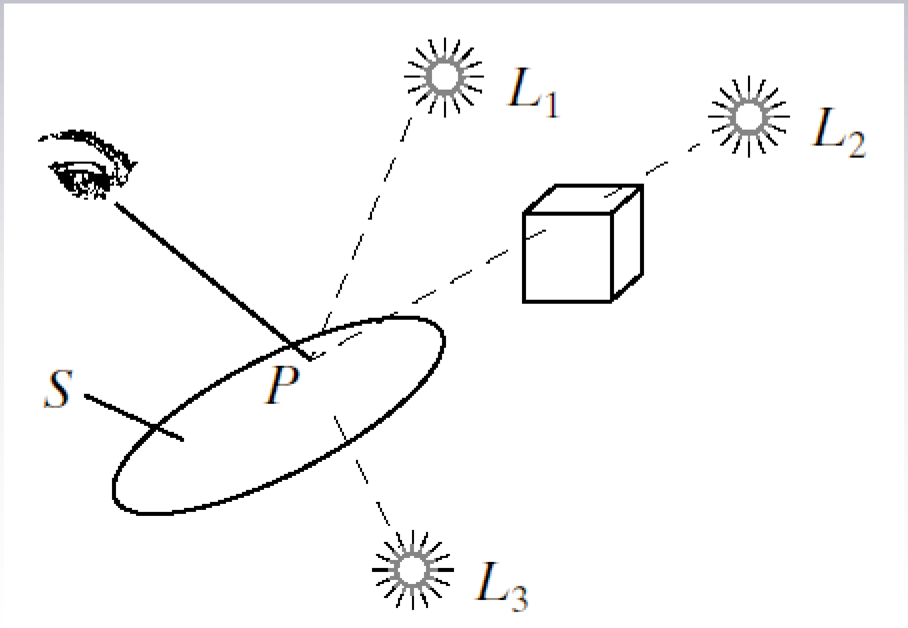
- General polygons
  - Express point **p** on a ray as some distance t along direction **d** from origin **o**: **p** = **o** + t**d**
  - Use plane equation **n** · **x** + d = 0, substitute **o** + t**d** for **x**, and solve for t
  - Only positive t's mean the intersection is in front of the eye
  - Then plug t back into **p** = **o** + t**d** to get **p**
  - Is the 2-D location of **p** on the plane inside the 2-D polygon?
    - For convex polys, Cohen-Sutherland-style outcode test will work
- Triangles
  - Direct **barycentric coordinates** expression (see Shirley, Chaps. 2.7 and 4.4)

$$\mathbf{t}(u, v) = (1 - u - v)\mathbf{v}_0 + u\mathbf{v}_1 + v\mathbf{v}_2$$

  - Set this equal to parametric form of ray **o** + t**d** and solve for intersection point (t, u, v)
  - Only inside triangle if u, v, and $1 - u - v$ are between 0 and 1

Shadow Rays

- For point being locally shaded, spawn new ray in each light direction and check for intersection to make sure light is "visible"

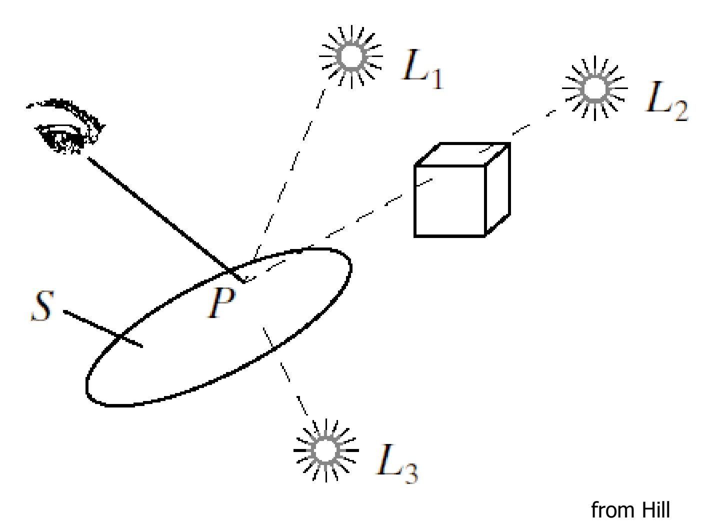


Shadow Rays

- For point **p** being locally shaded, only add diffuse & specular components for light l if light is not occluded (i.e., blocked)
- Test for occlusion of l for **p**:
  - Spawn **shadow ray** for l with origin **p**, direction **l**(l)
  - Check whether shadow ray intersects any scene object
  - Intersection only "counts" if:

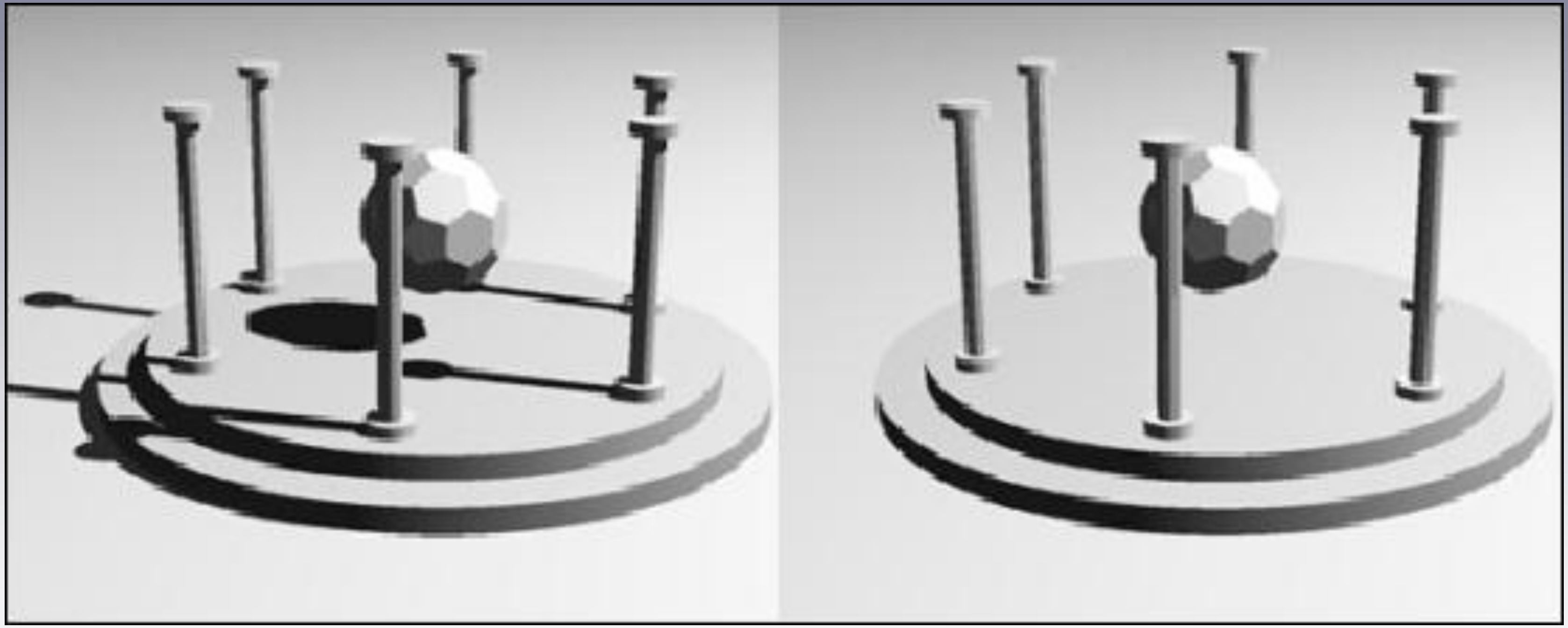
$$0 < t < |\mathbf{p}_l - \mathbf{p}|$$

- More details in Shirley, Chap. 4.7



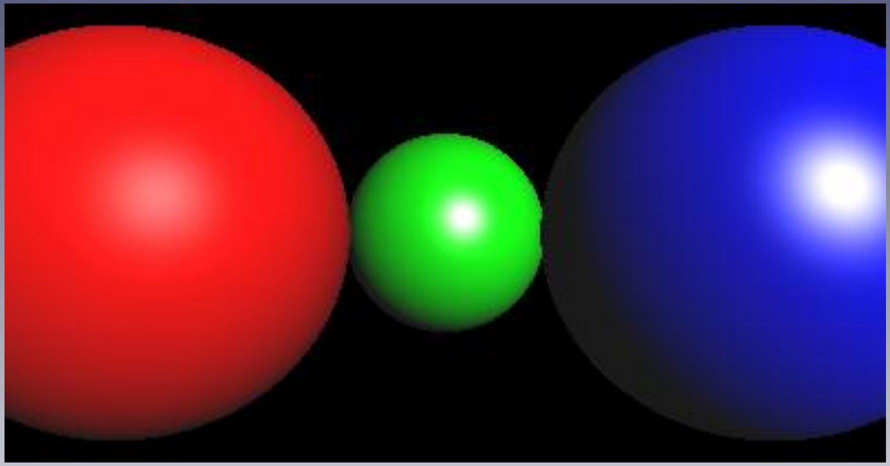
from Hill

Ray-Cast Scene with and without Shadows

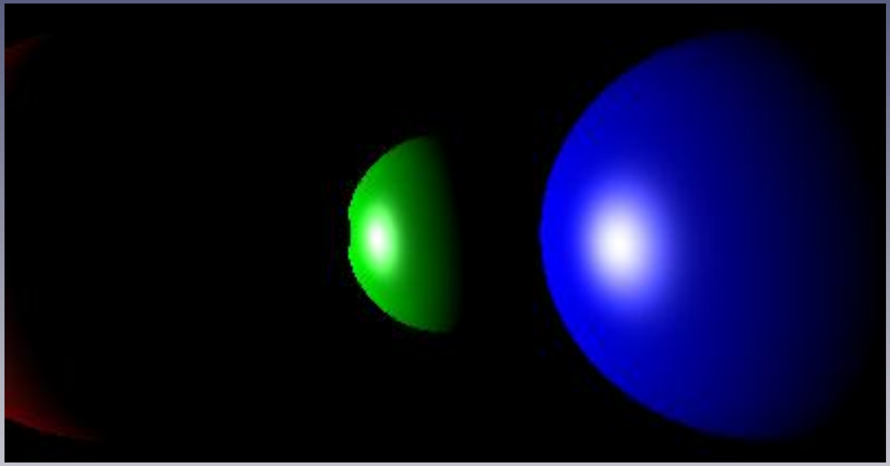


from Hill

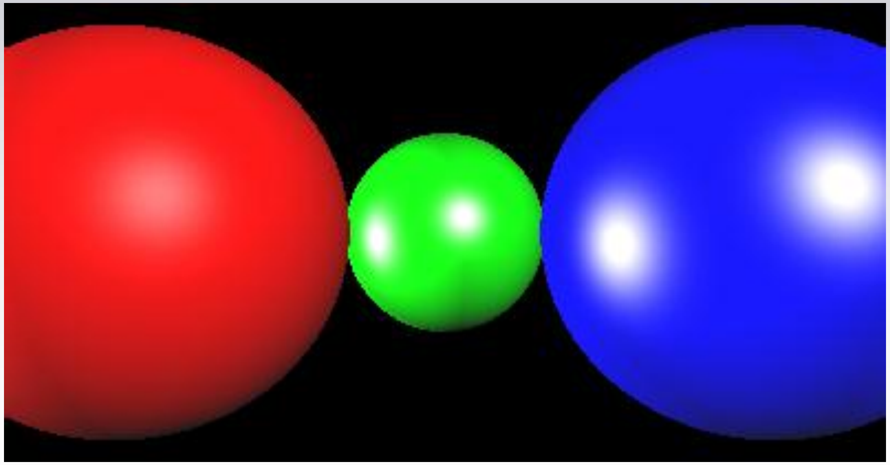
Ray Casting Example: No shadows



Light 1 only

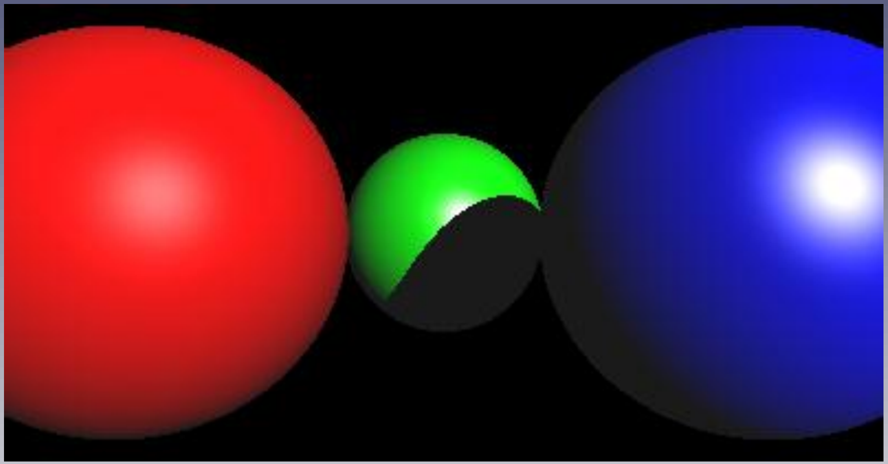


Light 2 only

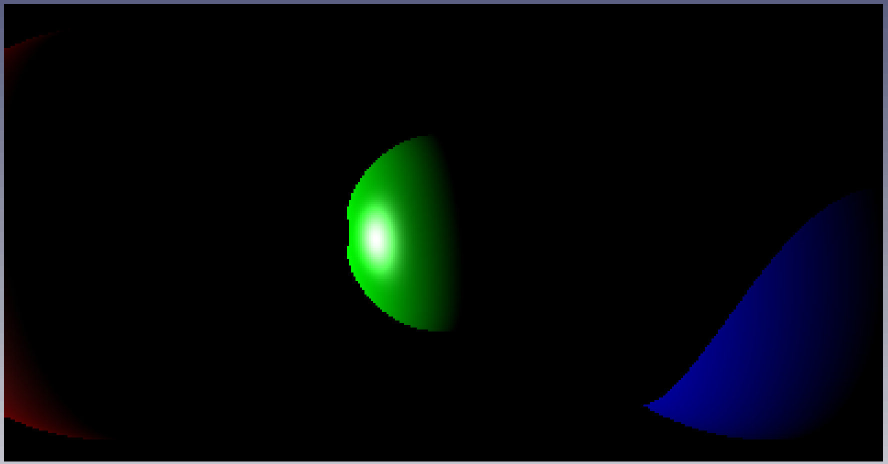


Lights 1 and 2

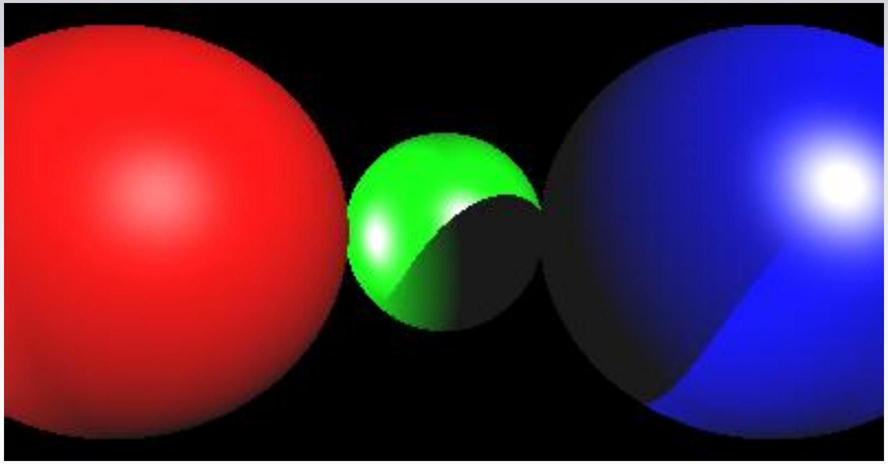
Ray Casting Examples: Shadows



Light 1 only



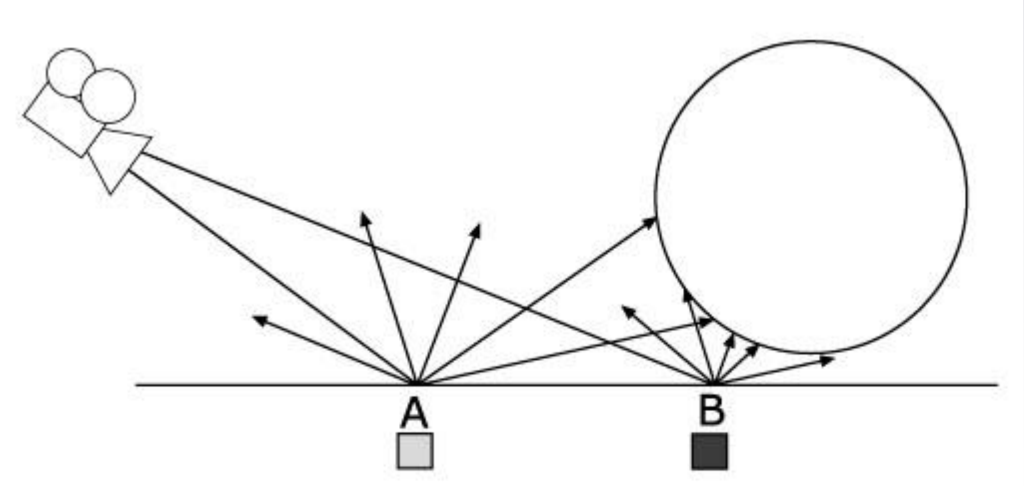
Light 2 only



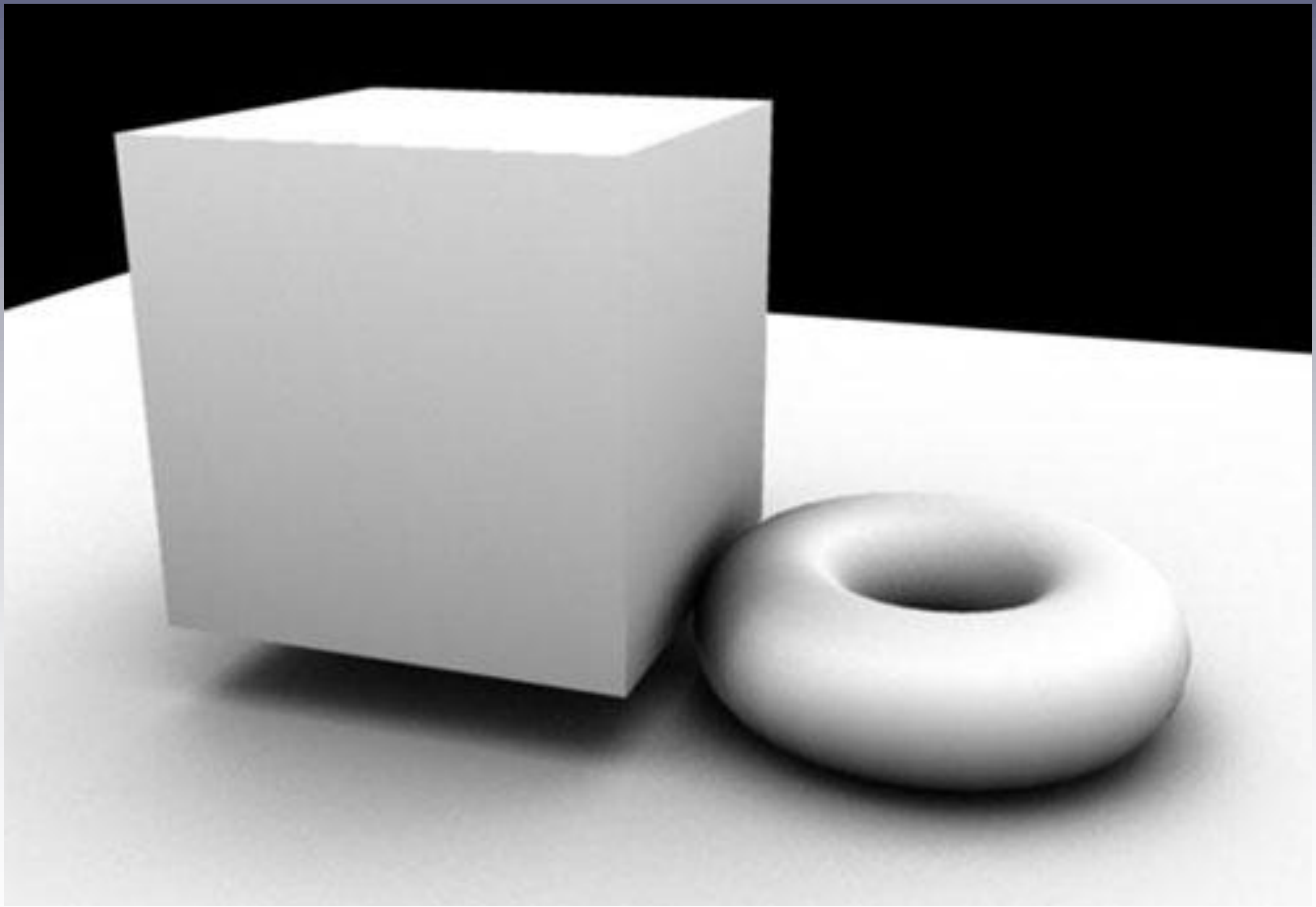
Lights 1 and 2

Ambient Occlusion

- Extension of shadow ray idea—not every point should get full ambient illumination
- Idea: Cast multiple random rays (a "distribution of rays") from each rendered surface point to estimate percent of sky hemisphere that is visible
  - Limit length of rays so distant objects have no effect
  - Cosine weighting/distribution for foreshortening
- Developers of this idea won a technical Oscar in 2010



Ambient Occlusion: Example



Ambient Occlusion: Example

http://www.gavinharrison.co.uk/renderman04.php