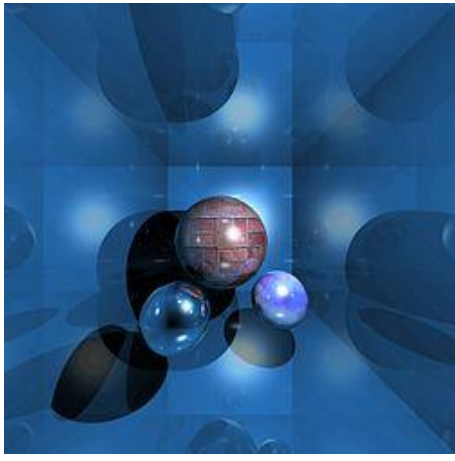
Computer Graphics

Lecture 10

Ray Tracing

Taku Komura

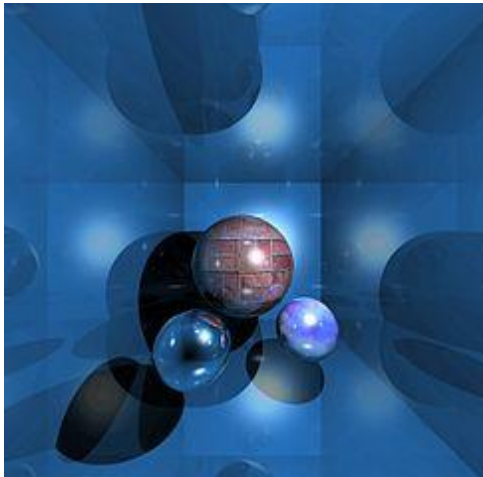


Overview

1. Ray tracing method
   a. method overview
   b. Ray tree
2. Ray-triangle intersection
3. Bounding Volume
   a. Bounding volume overview
   b. Bounding volume hierarchy

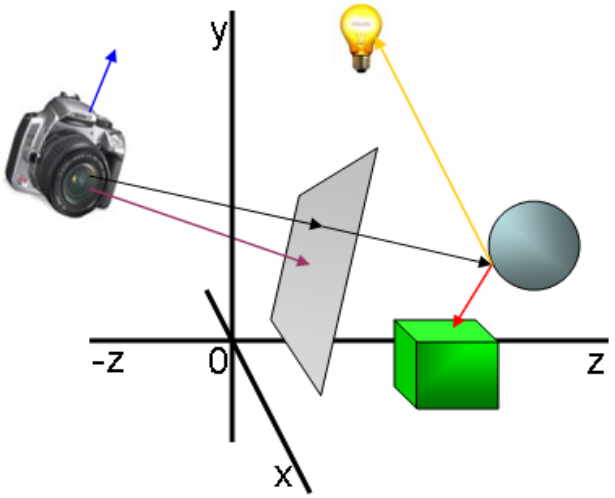
Ray Tracing [Appel ’68]

- One of the most popular methods used in 3D computer graphics to render an image
- Different from the rasterization-based approach
- Good at simulating specular effects, producing shadows
- Also used as a function for other global illumination techniques



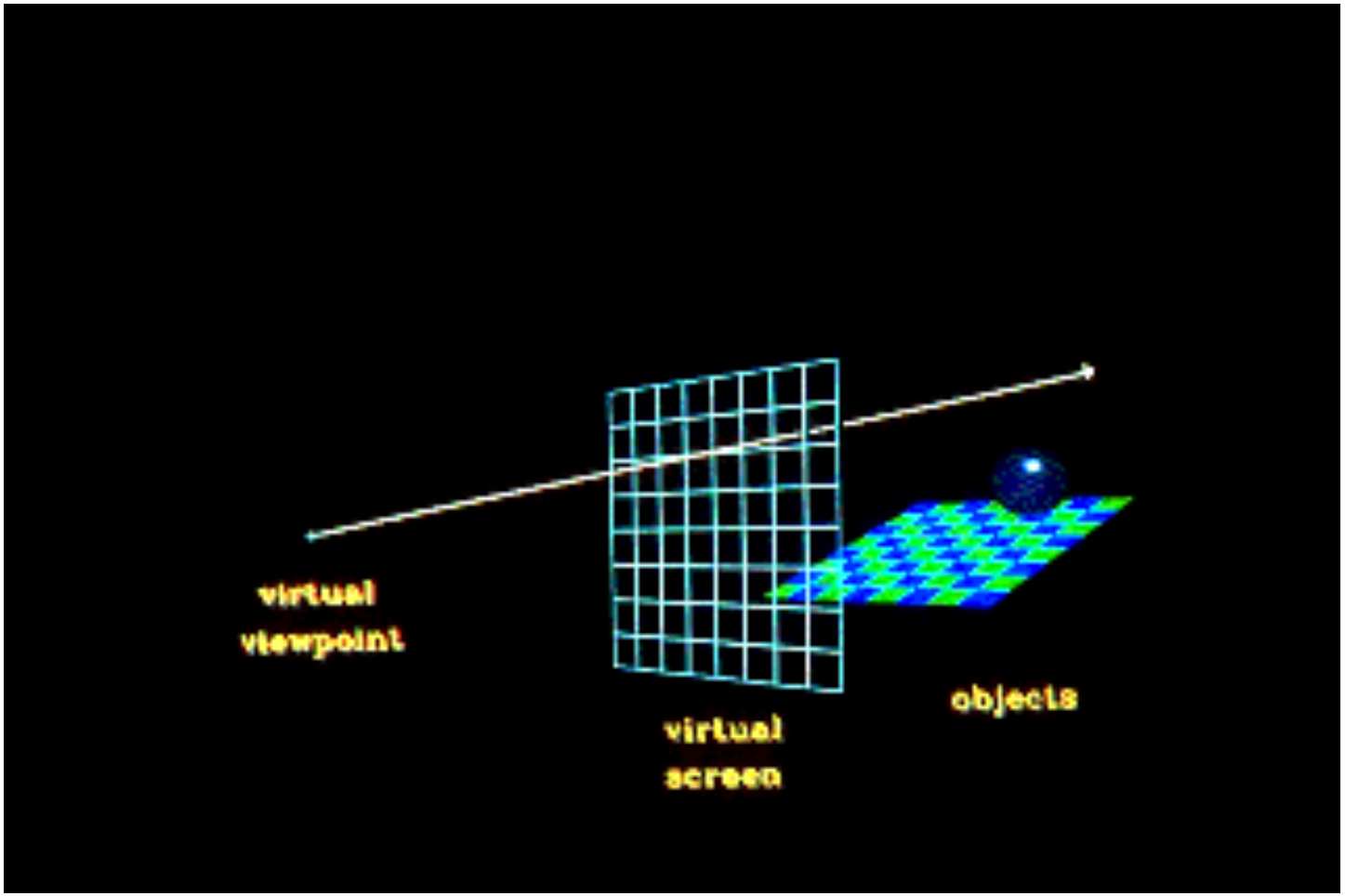
Ray Tracing [Appel ’68]

- Tracing the path taken by a ray of light through the scene
- Rays are cast to each pixel. They are reflected, refracted, or absorbed whenever they intersect objects
- Let’s look into an example



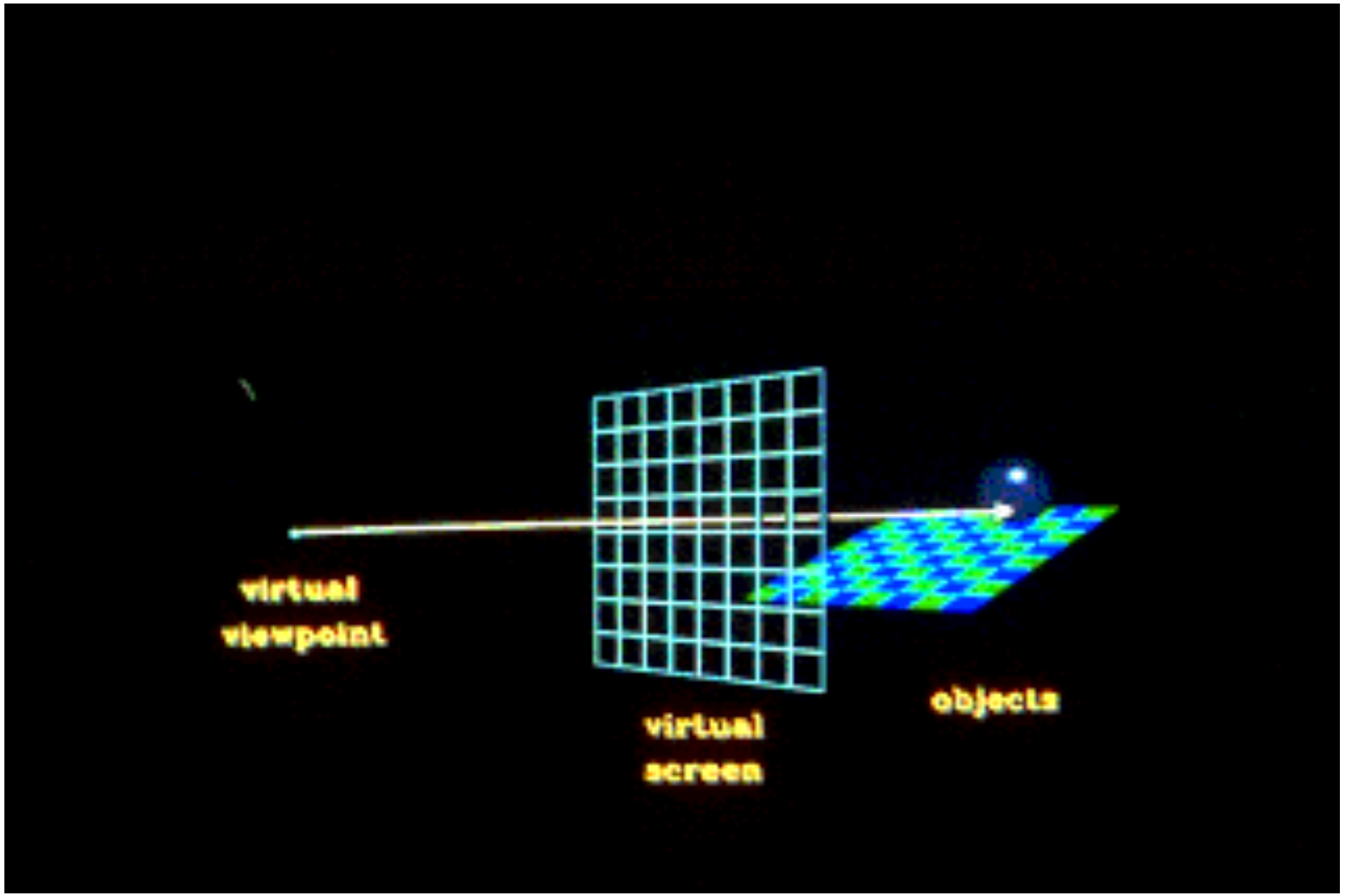
The ray misses objects

→ the pixel is colored by the background color



*

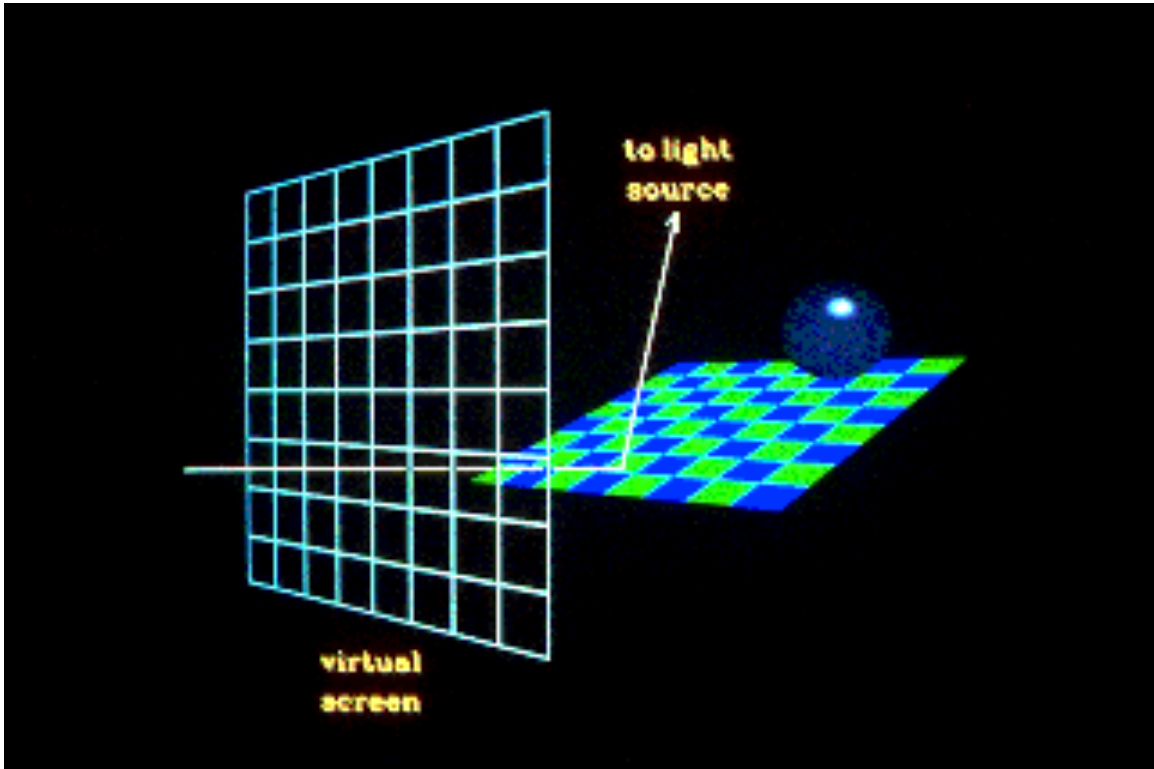
When the ray hits an object:



*

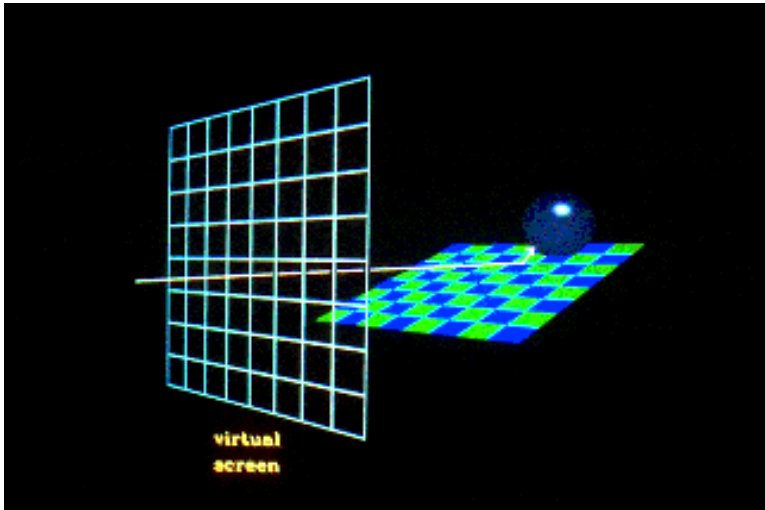
Check if the fragment is shadowed

- A secondary ray, called a "shadow" ray, is shot towards the light sources

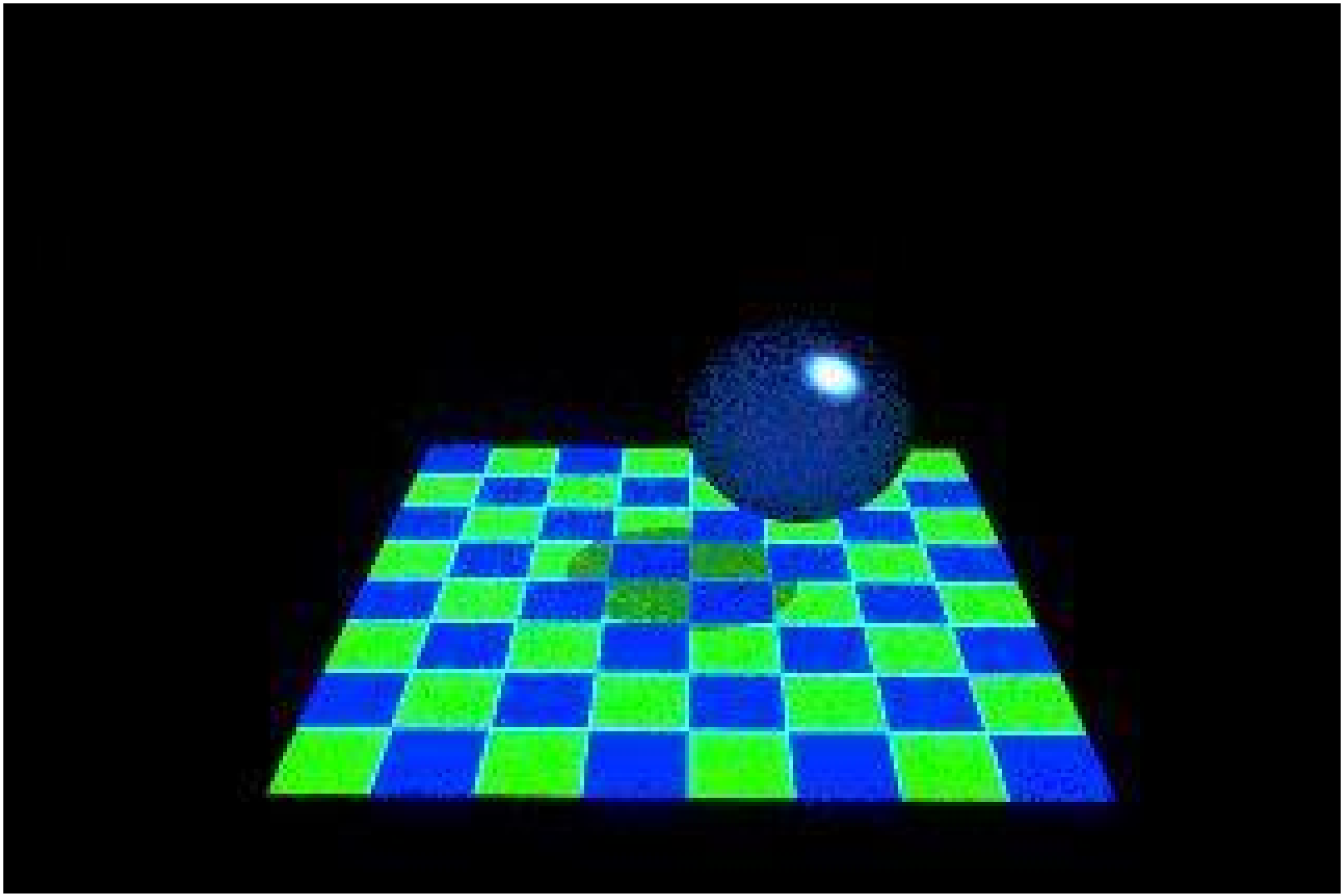


Shadow Ray

- If this shadow ray hits another object before it hits a light source, then the first intersection point is in the shadow of the second object.
- We only apply the ambient term for that light source.
- Otherwise do the local Phong Illumination



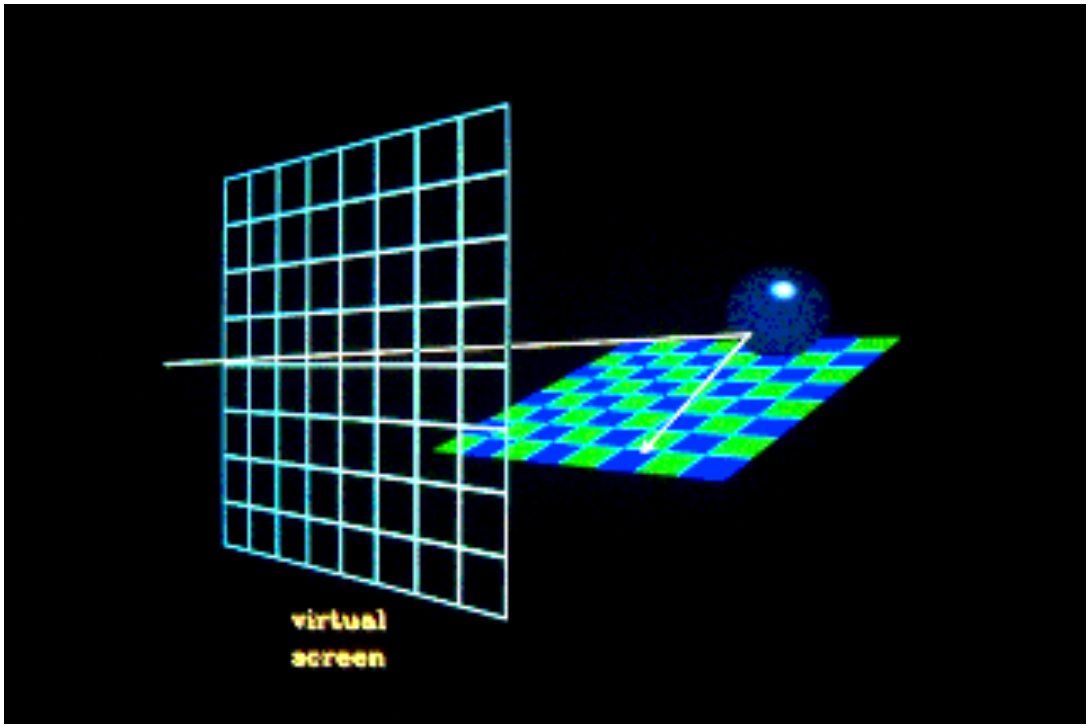
Example shadow



*

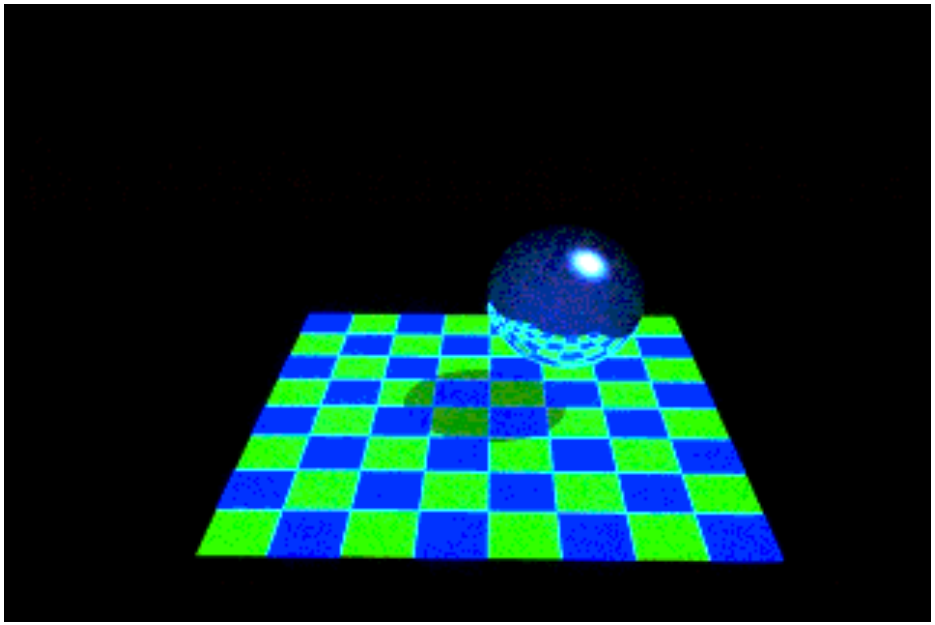
Reflected Ray

- Also, when a ray hits an object, a reflected ray is generated which is tested against all of the objects in the scene.



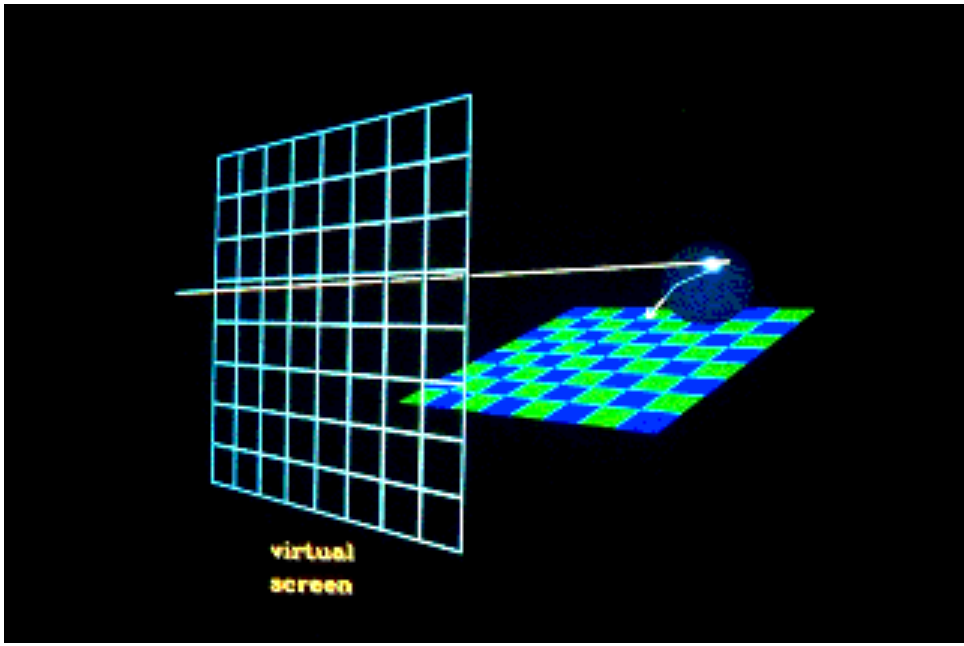
Contribution from the reflected ray

- If the reflected ray hits an object then a local illumination model is applied at the point of intersection and the result is carried back to the first intersection point.



Refraction Ray

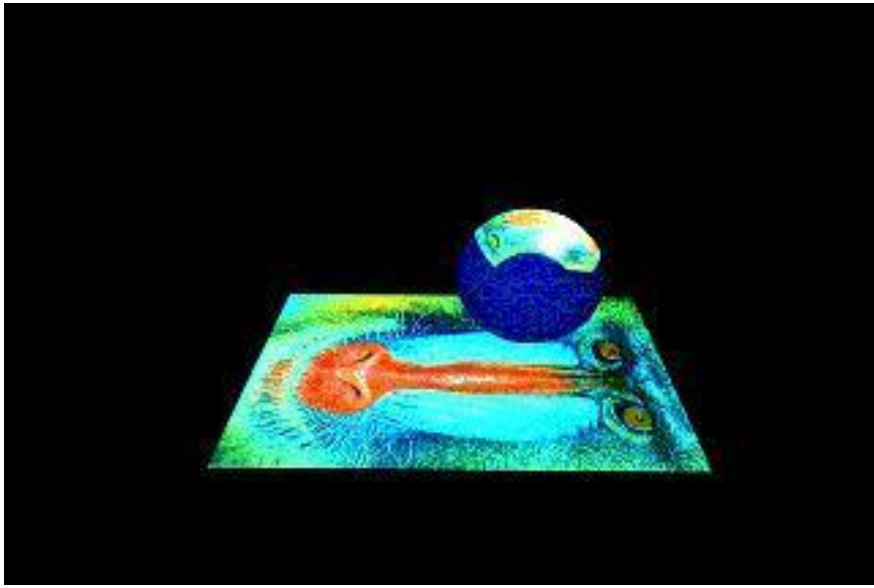
- If the intersected object is transparent, then a refraction ray is generated based on Snell's law



*

Contribution from the refraction ray

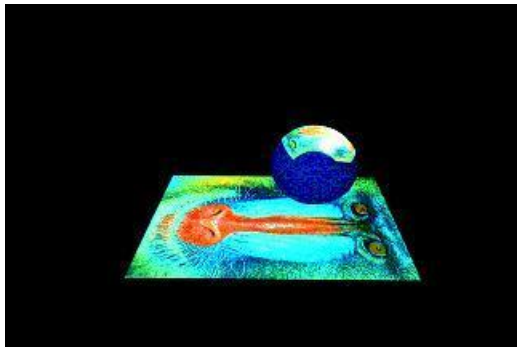
- As with the reflected ray, if the refraction ray hits an object then a local illumination model is applied at the point of intersection and the result is carried back to the first intersection point.



Rays to be considered

- Shadow ray
- Reflection ray
- Refraction ray

Sum the color based on the three rays

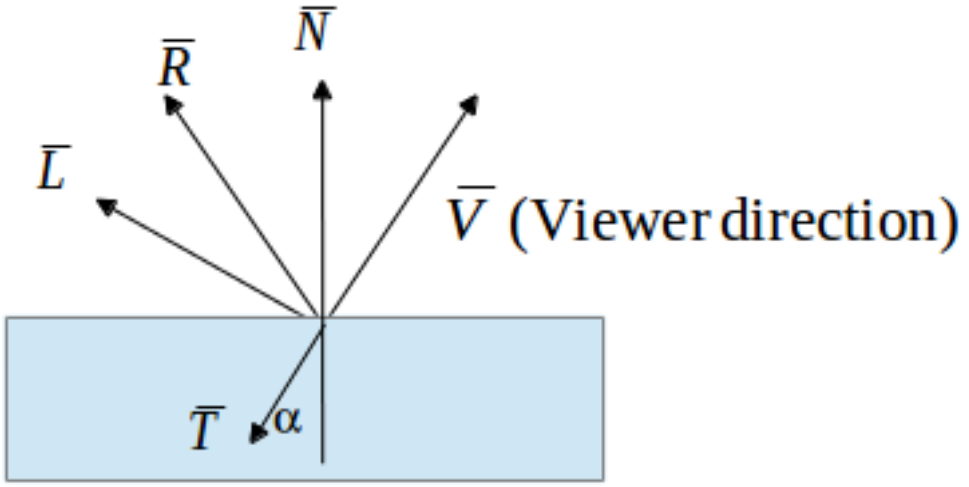


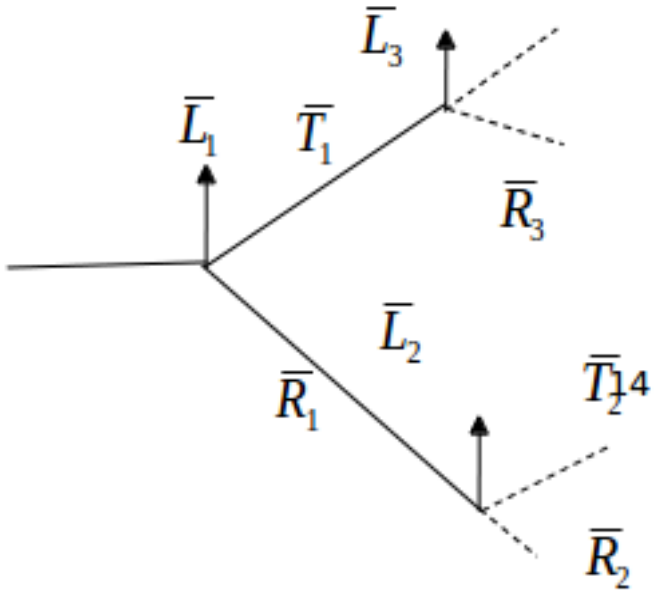
$$I = \mathbf{I}local + Kr * \mathbf{R} + Kt * \mathbf{T}$$

Ilocal : color by local illumination,
R : color by reflection,
T : color by transmission,
and Kr, Kt are coefficients

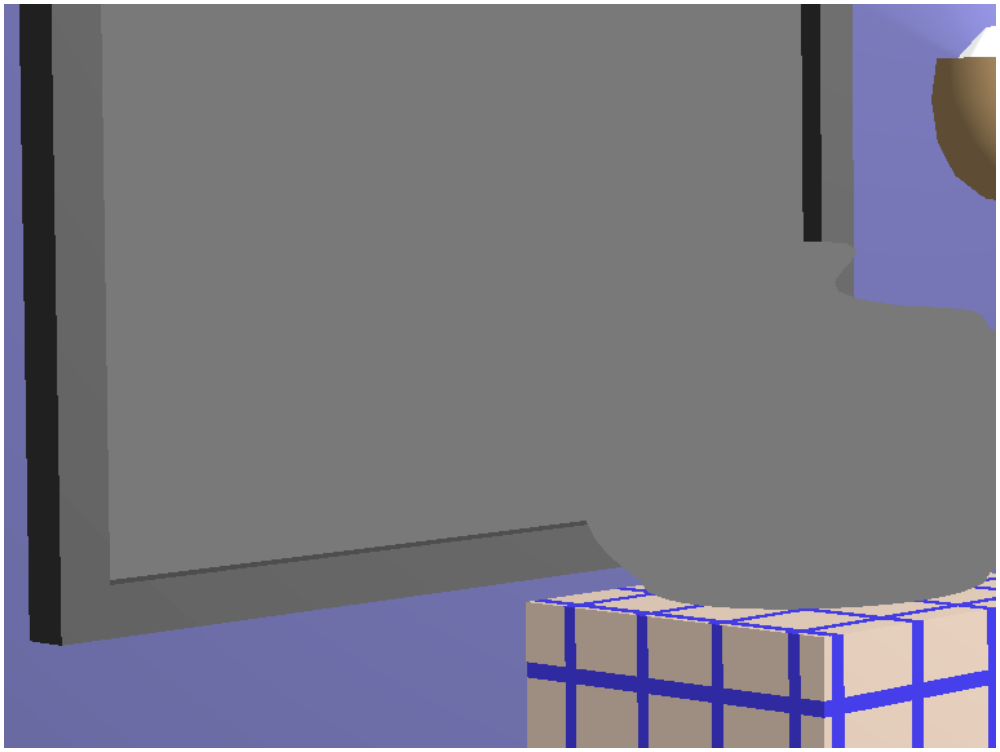
Ray Tree [Whitted '80]

- The reflection rays and refraction rays are recursively launched when they hit other surfaces
- And the light is brought back to the previous hit





Test Scene.

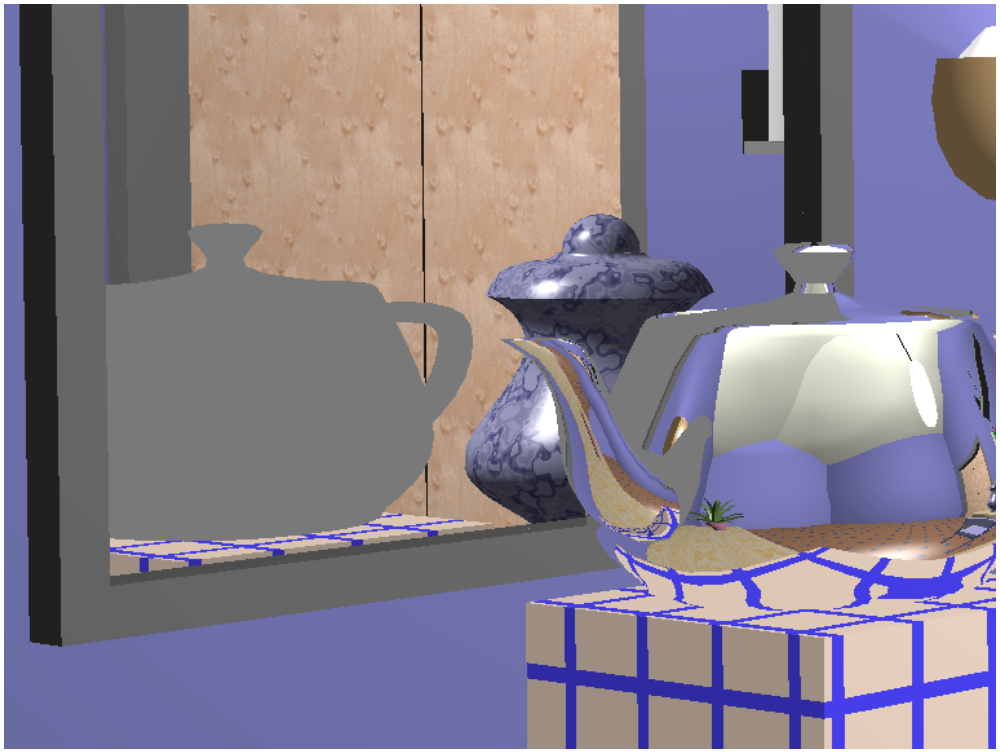


Ray tree depth 1.

Note only ambient shade on mirror and teapot

*

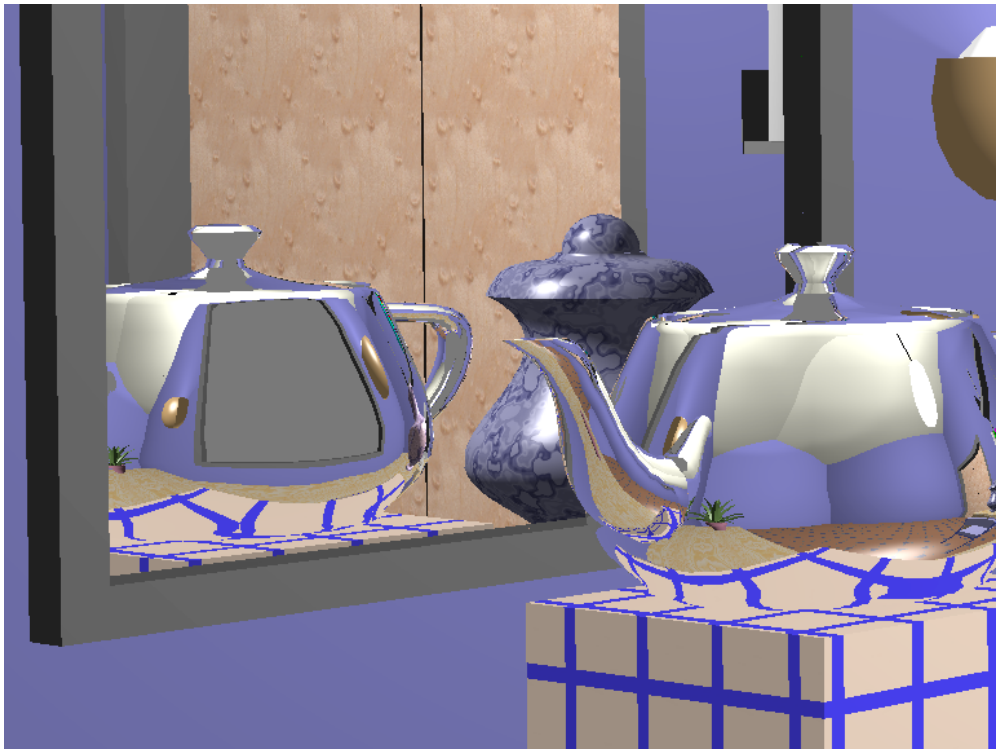
Test Scene.



Ray tree depth 2.

Note only ambient shade on reflection of mirror and teapot.

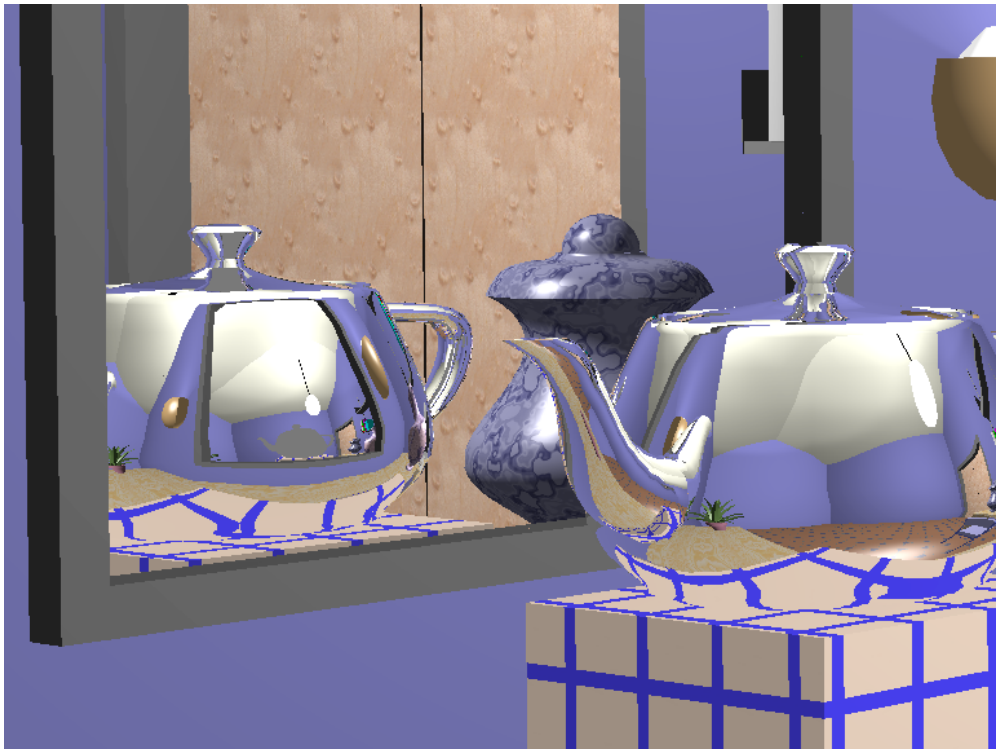
Test Scene.



Ray tree depth 3.

Note only ambient shade on reflection of mirror in teapot.

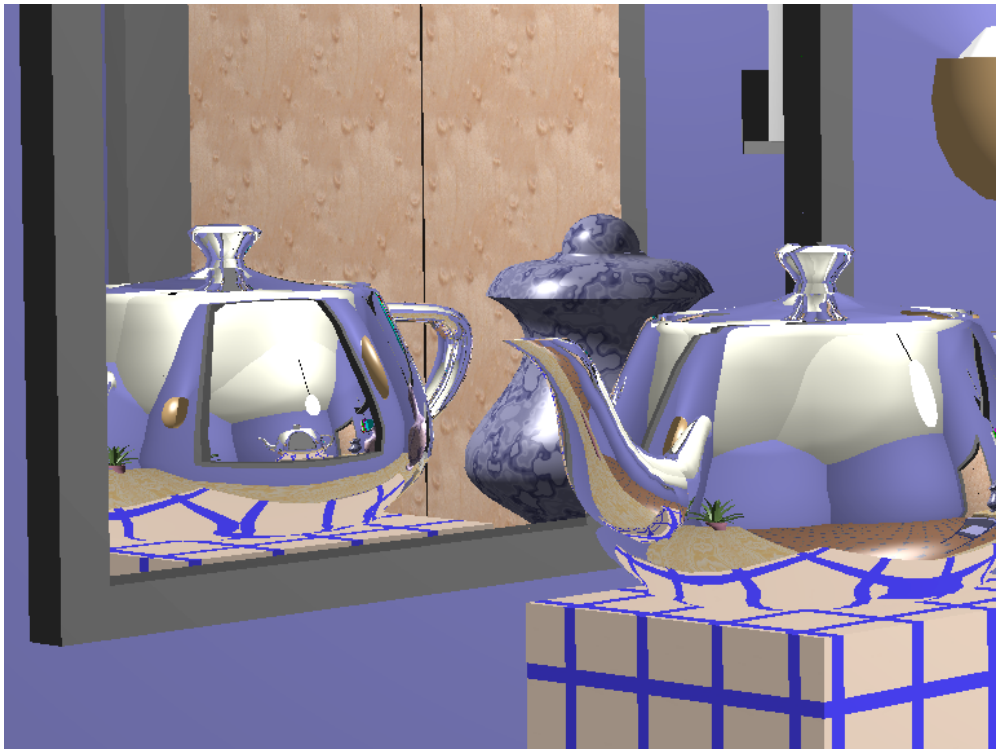
Test Scene.



Ray tree depth 4.

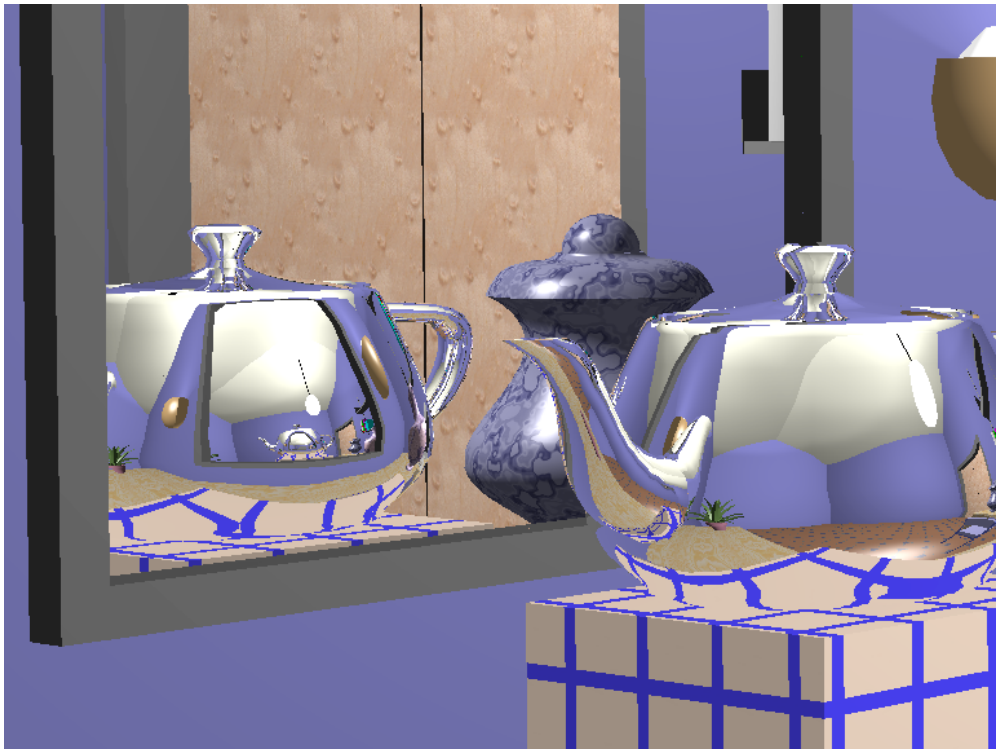
Note ambient shade on reflection of teapot in reflection of mirror in teapot.

Test Scene.



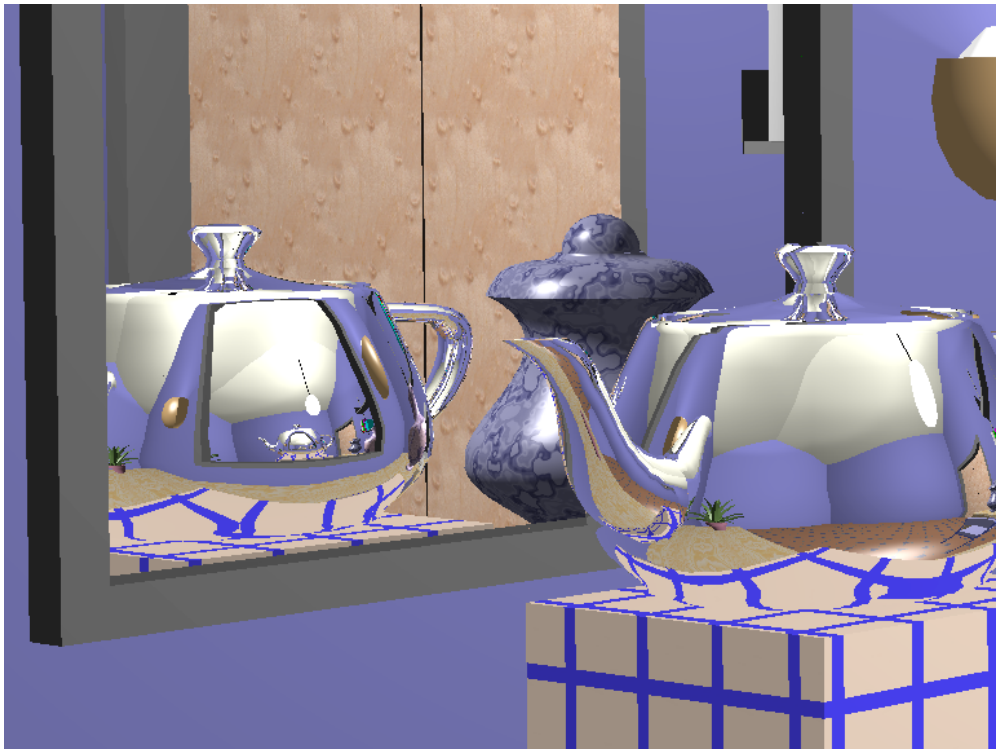
Ray tree depth 5.

Test Scene.



Ray tree depth 6.

Test Scene.



Ray tree depth 7.

Ray Tree (2): Color of Specular Surfaces

- The color is computed per each ray

$$I = I_{local} + K_r R + K_t T$$
$$R = I'_{local} + K'_r R' + K'_t T'$$
$$R' = I''_{local} + K''_r R'' + K''_t T''$$
$$\vdots$$

- Substituting them all into one equation, we get

$$I = I_{local} + K_r (I'_{local} + K'_r (I''_{local} + K''_r (I'''_{local} + K'''_r (\ldots))))$$

When to stop ?

- When the a ray hits a perfectly diffuse surface
- For specular surface, we can define a fixed depth
- Or when the multiplication of the coefficients reach some threshold

$$I = I_{local} + K_r(I'_{local} + K'_r(I''_{local} + K''_r(I'''_{local} + K'''_r(\ldots))))$$

$$K_r K'_r K''_r K'''_r \ldots < threshold$$

Hall, R. A. and Greenberg D.P. , "A Testbed for Realistic Image Synthesis", IEEE Computer Graphics and Applications, 3(8), Nov., 1983

Examples of Ray-traced images.

What is the complexity?

Ray tracing : resolution w x h, number of triangles N

O(?)

Rasterization in the graphics pipeline: number of vertices V,

number of triangles N,

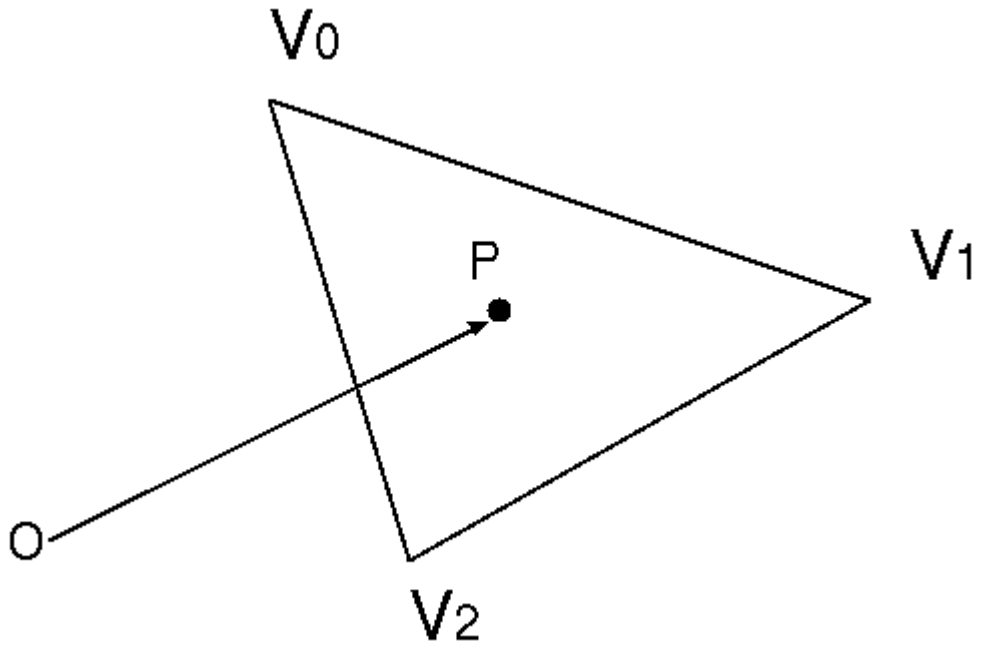
O(?)

Overview

1. Ray tracing method
   a. method overview
   b. Ray tree
2. **Ray-triangle intersection**
3. Bounding Volume
   a. Bounding volume overview
   b. Bounding volume hierarchy

Computing the ray-triangle intersection

- First check the intersection point of the ray and the plane that includes the triangle
- Check if the point is inside the triangle or not



The Plane Equation

- Assuming the triangle is composed of vertices $\boldsymbol{V_0, V_1, V_2}$
- The normal vector of the triangle can be computed by

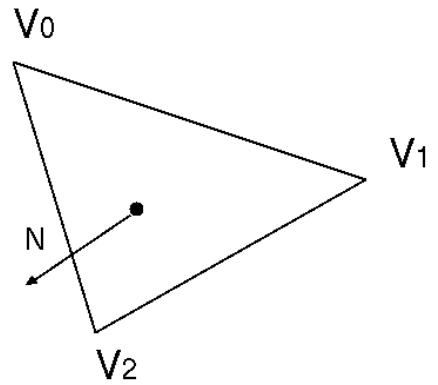
$$\boldsymbol{N = V_0 V_1 \times V_0 V_2}$$

- The equation of the plane can then be written as

$$\boldsymbol{N} \cdot \boldsymbol{P} + d = 0$$

where

$$d = -\boldsymbol{V_0} \cdot \boldsymbol{N}$$



Computing the intersection

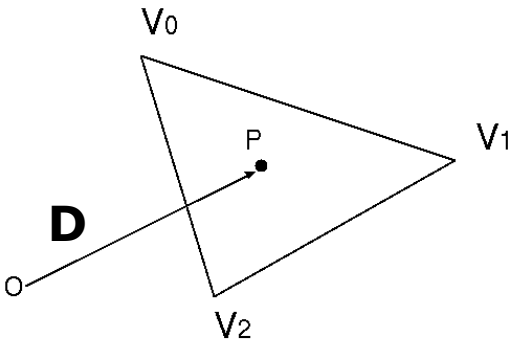
Let the parametric representation of the ray be

$$\mathbf{r}(t) = \mathbf{O} + \mathbf{D}\,t$$

Substituting this into the plane equation

$\boldsymbol{N} \boldsymbol{.} \boldsymbol{P} + d = 0$ we get

$$t = -\frac{d + \mathbf{N} \cdot O}{\mathbf{N} \cdot \mathbf{D}}.$$



The result is rejected if

- The polygon and ray are parallel
- The intersection is behind the origin (t<0)
- A closer intersection has been already found ($t > t_{min}$)

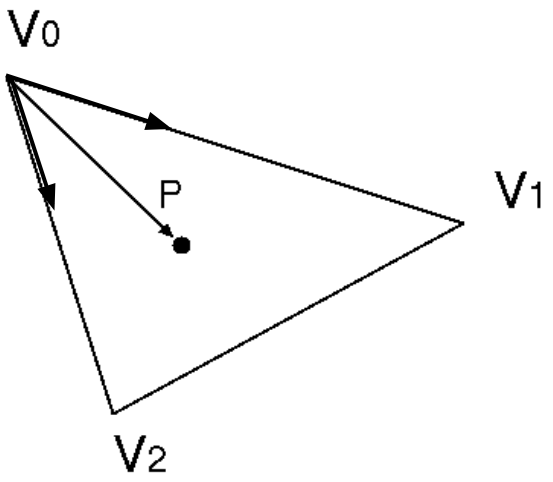
Checking if the point is inside the triangle or not

A vector connecting V_0 and P can be written as

$$\overrightarrow{V_0P} = \alpha\overrightarrow{V_0V_1} + \beta\overrightarrow{V_0V_2}.$$

P will be inside the triangle if

$$\alpha \geq 0, \beta \geq 0, \text{and } \alpha + \beta \leq 1.$$

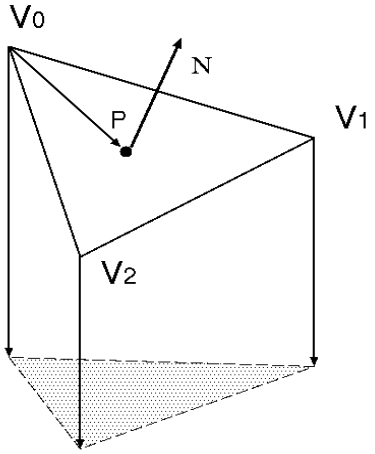


*

Projection onto Primary Planes

- To reduce the system, we project onto one of the primary planes, either xy,yz,zx
- It is better that the projected triangle is bigger
- We examine the absolute value of the normal's attributes and determine the primary plane

$$i_0 = \begin{cases} 0 & \text{if } |\mathbf{N}_x| = \max(|\mathbf{N}_x|, |\mathbf{N}_y|, |\mathbf{N}_z|) \\ 1 & \text{if } |\mathbf{N}_y| = \max(|\mathbf{N}_x|, |\mathbf{N}_y|, |\mathbf{N}_z|) \\ 2 & \text{if } |\mathbf{N}_z| = \max(|\mathbf{N}_x|, |\mathbf{N}_y|, |\mathbf{N}_z|). \end{cases}$$



Projection onto Primary Planes

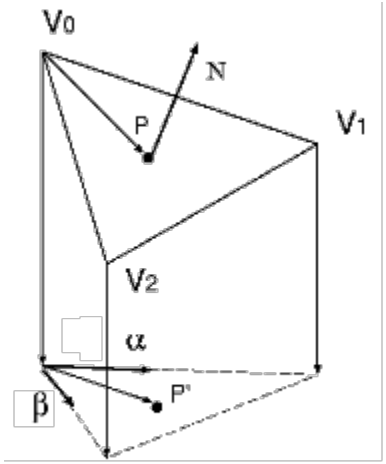
- Consider I1, i2 the indices different from i0
- They represent the primary plane
- Let (u,v) be the 2D coordinates of a vector in this plane; the vectors V_{0P}, V_{01}, V_{02} projected onto the plane are

$$u_0 = P_{i_1} - V_{0_{i_1}} \qquad u_1 = V_{1_{i_1}} - V_{0_{i_1}} \qquad u_2 = V_{2_{i_1}} - V_{0_{i_1}}$$

and

$$v_0 = P_{i_2} - V_{0_{i_2}} \qquad v_1 = V_{1_{i_2}} - V_{0_{i_2}} \qquad v_2 = V_{2_{i_2}} - V_{0_{i_2}}$$

$$\begin{cases} u_0 = \alpha . u_1 + \beta . u_2 \\ v_0 = \alpha . v_1 + \beta . v_2 \end{cases}$$



- The solutions are

$$\alpha = \frac{\det\begin{pmatrix} u_0 & u_2 \\ v_0 & v_2 \end{pmatrix}}{\det\begin{pmatrix} u_1 & u_2 \\ v_1 & v_2 \end{pmatrix}} \quad \text{and} \quad \beta = \frac{\det\begin{pmatrix} u_1 & u_0 \\ v_1 & v_0 \end{pmatrix}}{\det\begin{pmatrix} u_1 & u_2 \\ v_1 & v_2 \end{pmatrix}}.$$

The interpolated normal at P can be computed by

$$\mathbf{N}_P = (1 - (\alpha + \beta))\mathbf{N}_0 + \alpha\mathbf{N}_1 + \beta\mathbf{N}_2.$$

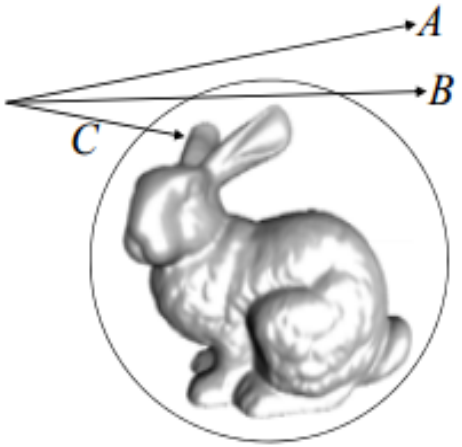
*

Overview

1. Ray tracing method
   a. method overview
   b. Ray tree
2. Ray-triangle intersection
3. **Bounding Volume**
   a. Bounding volume overview
   b. Bounding volume hierarchy

Bounding Volume:
Reducing Ray-Object Intersections

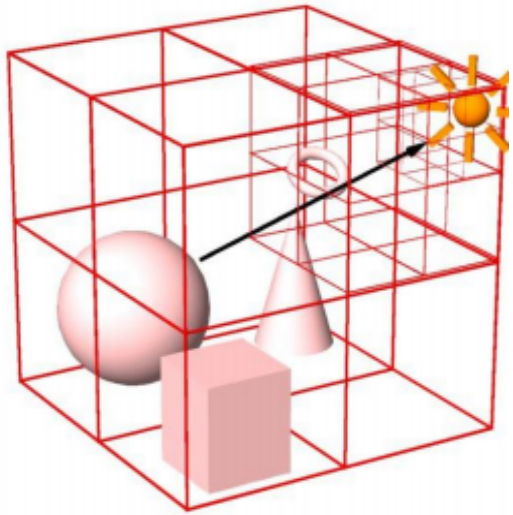
- Bounding Volumes:
- First test for intersection with the bounding volume
  - Only if there is an intersection, test against the objects enclosed by the volume.
  - Can save a lot of computation when the ray does not hit the bunny
  - Can use boxes, spheres

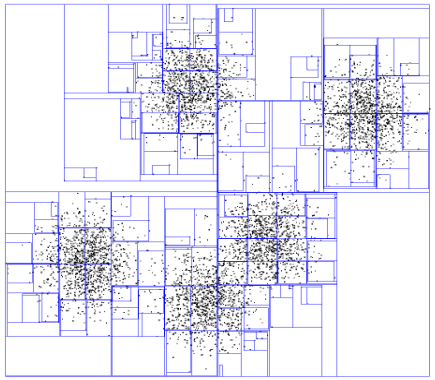


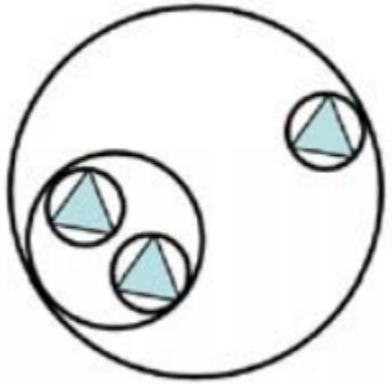
*

Using Hierarchical Structures

- Enclose groups of objects in sets of hierarchical bounding volumes
  - Oct-tree, KD-tree (axis aligned boxes)
  - Bounding Volume Hierarchy (boxes, spheres)

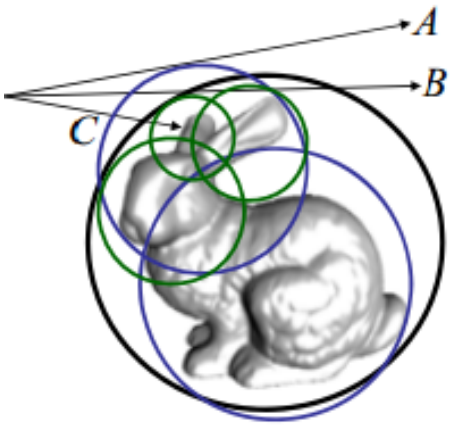


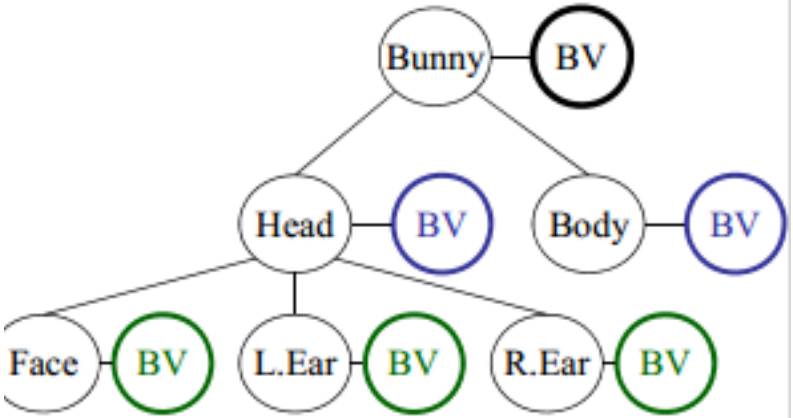




Bounding Volume Hierarchy

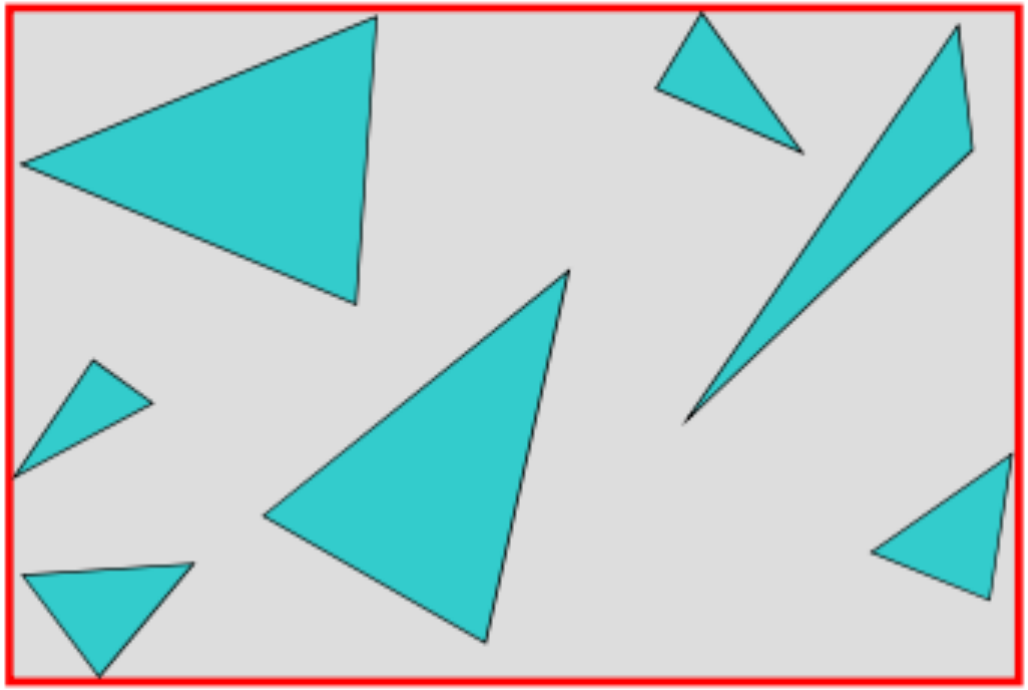
–Associate bounding volume with each object
–The bounding volume does not partition
  –the bounding volumes can overlap each other.
–The volume higher in the hierarchy contains their children
–If ray misses a node's bounding volume, then no need to check any node beneath it
–If ray hits a node's BV, then check with its children





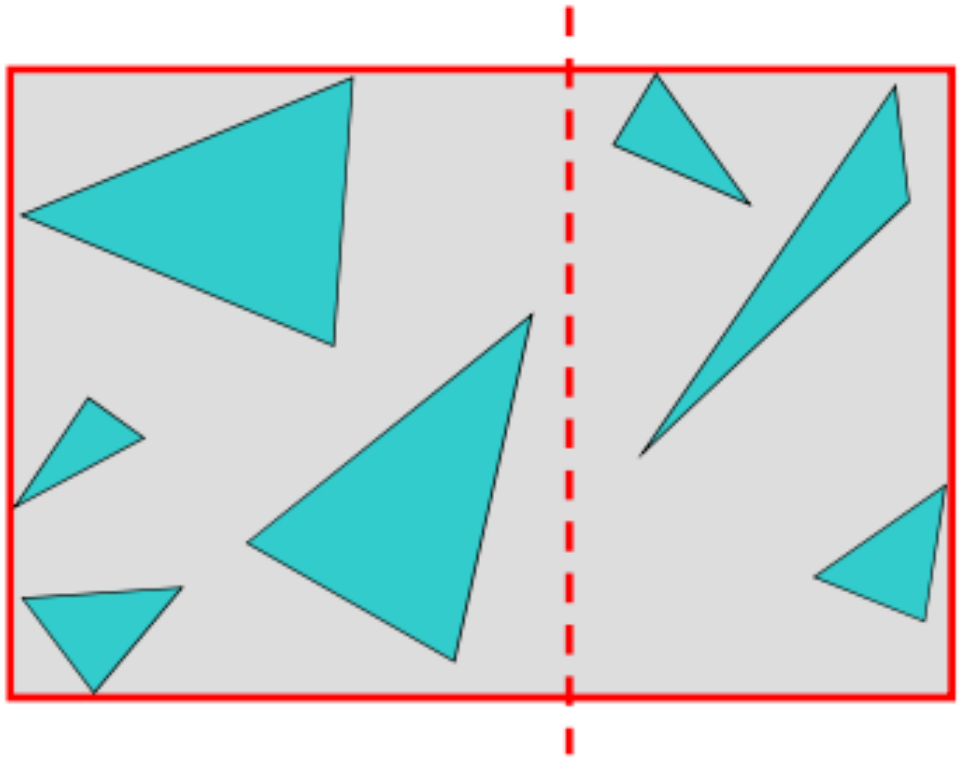
Producing the Bounding Volume Hierarchy

Find bounding box of objects



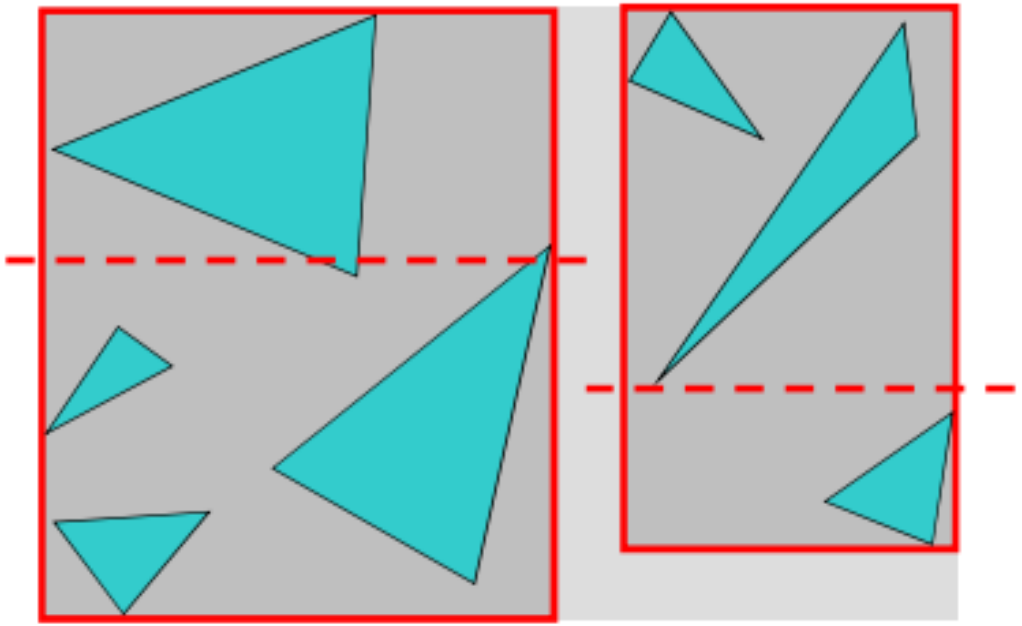
Producing the Bounding Volume Hierarchy

- Find bounding box of objects
- Split objects into two groups



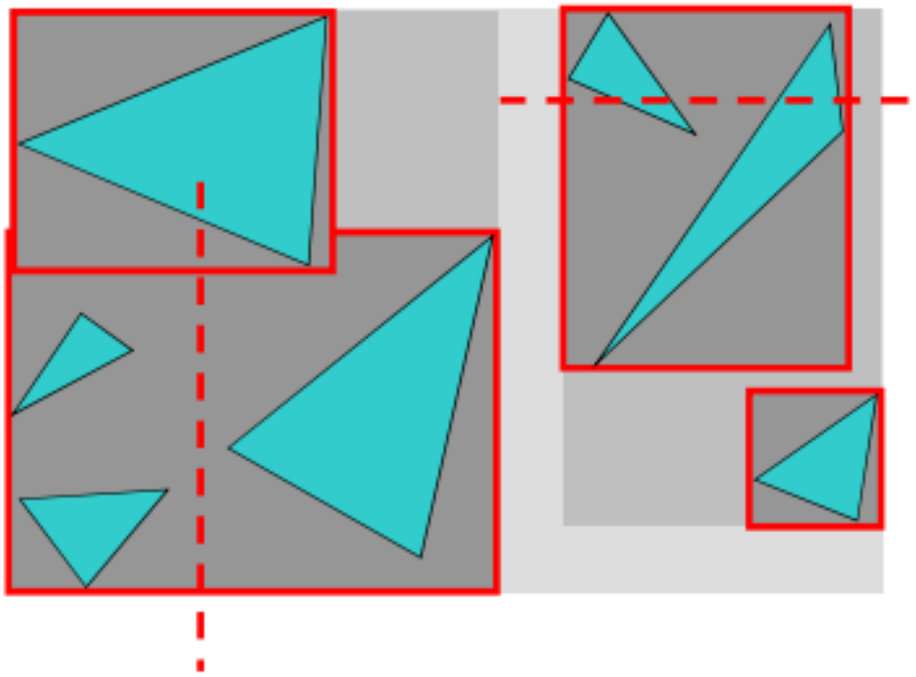
Producing the Bounding Volume Hierarchy

- Find bounding box of objects
- Split objects into two groups
- Recurse



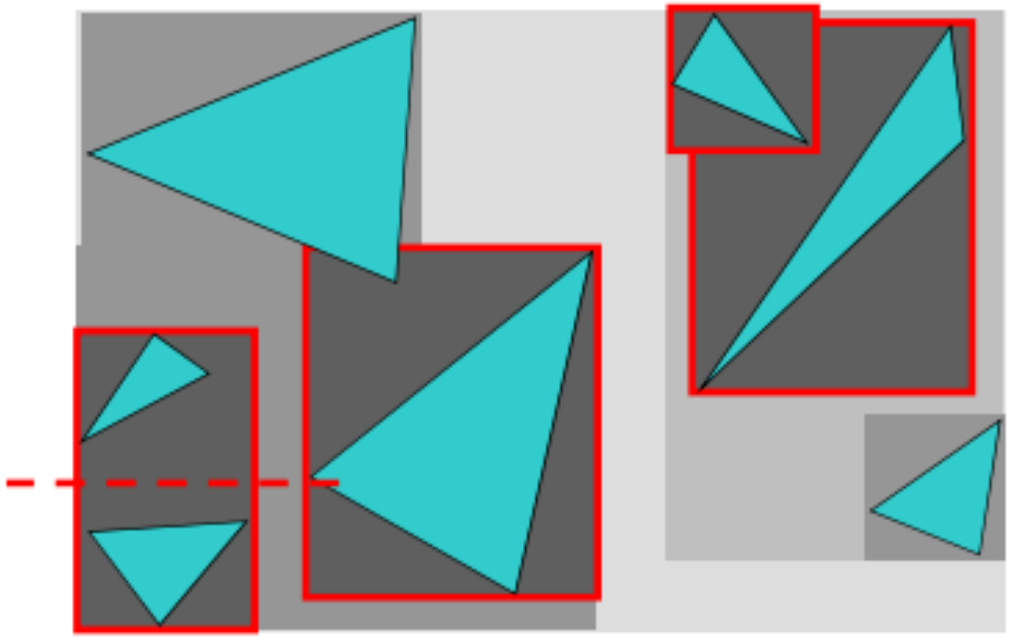
Producing the Bounding Volume Hierarchy

- Find bounding box of objects
- Split objects into two groups
- Recurse



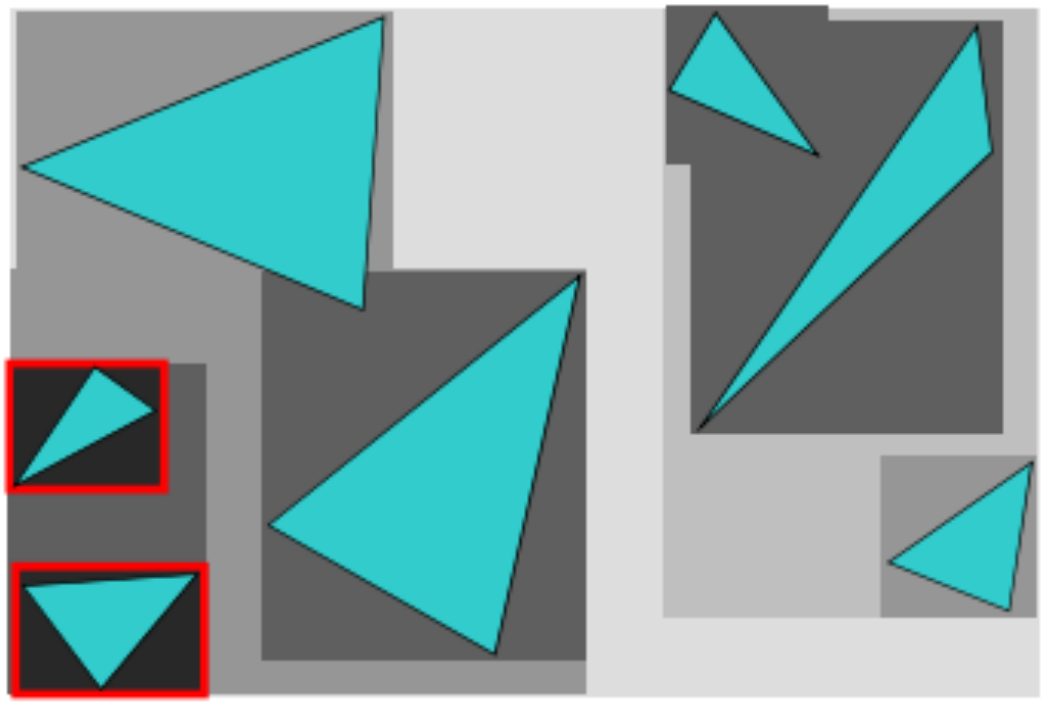
Producing the Bounding Volume Hierarchy

- Find bounding box of objects
- Split objects into two groups
- Recurse



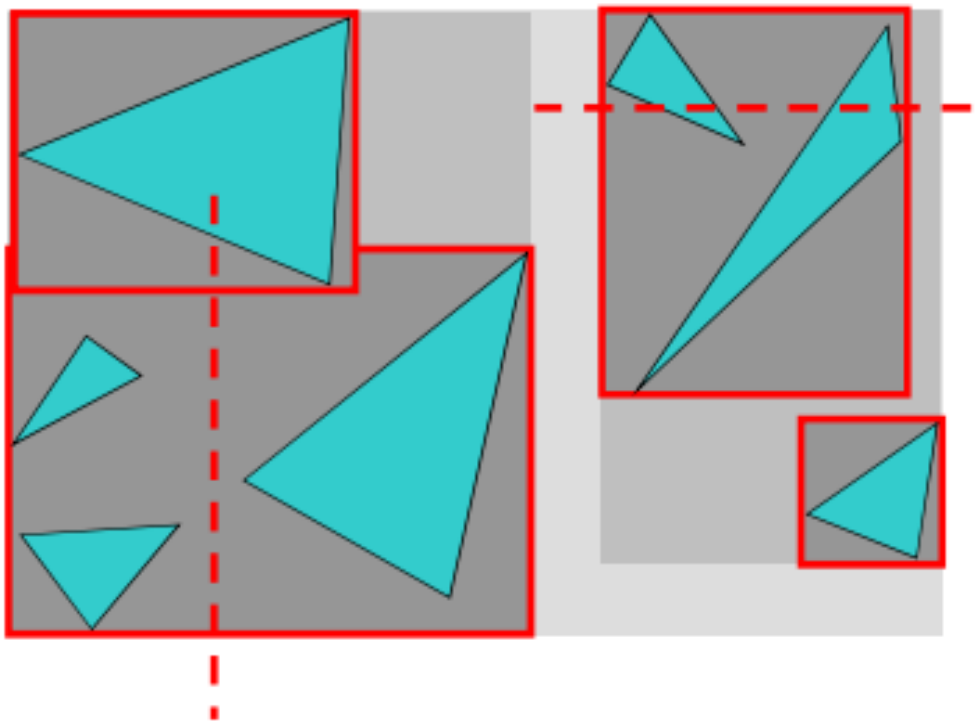
Producing the Bounding Volume Hierarchy

- Find bounding box of objects
- Split objects into two groups
- Recurse



Where to split objects?

- At midpoint OR
- Sort, and put half of the objects on each side



Computing the Intersection

```
intersect(node,ray,hits) {
  if( intersectp(node->bound,ray)
        if( leaf(node) )
              intersect(node->prims,ray,hits)
        else
              for each child
                    inter sect(child,ray,hits)
}
```

Summary

- Raytracing is a simple but expensive method for simulating scenes
- Effects of reflection, refraction and shadows can be easily produced
- The high costs come from the triangle-ray intersection tests
- The costs can be reduced by using the bounding volume hierarchy

Recommended Reading

- Foley at al. Chapter 16, sections 16.11, 16.12 and 16.12.5.
- Introductory text Chapter 14 sections 14.6 and 14.7.
- An Efficient Ray-Polygon Intersection
- by Didier Badouel from Graphics Gems I
- Most graphics texts cover recursive ray tracing.