Computer Graphics 10 - Ray tracing

Tom Thorne

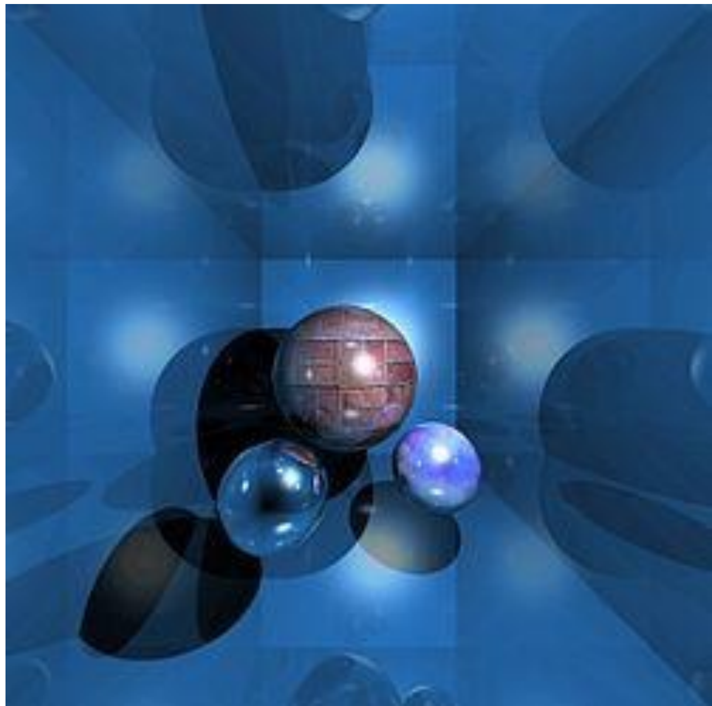
Slides courtesy of Taku Komura
www.inf.ed.ac.uk/teaching/courses/cg

Overview

- **Ray tracing overview**
- Ray trees
- Intersections
  - Spheres
  - Planes
  - Polygons
- Bounding volumes
  - Bounding volume hierarchies

Ray tracing (Appel '68)

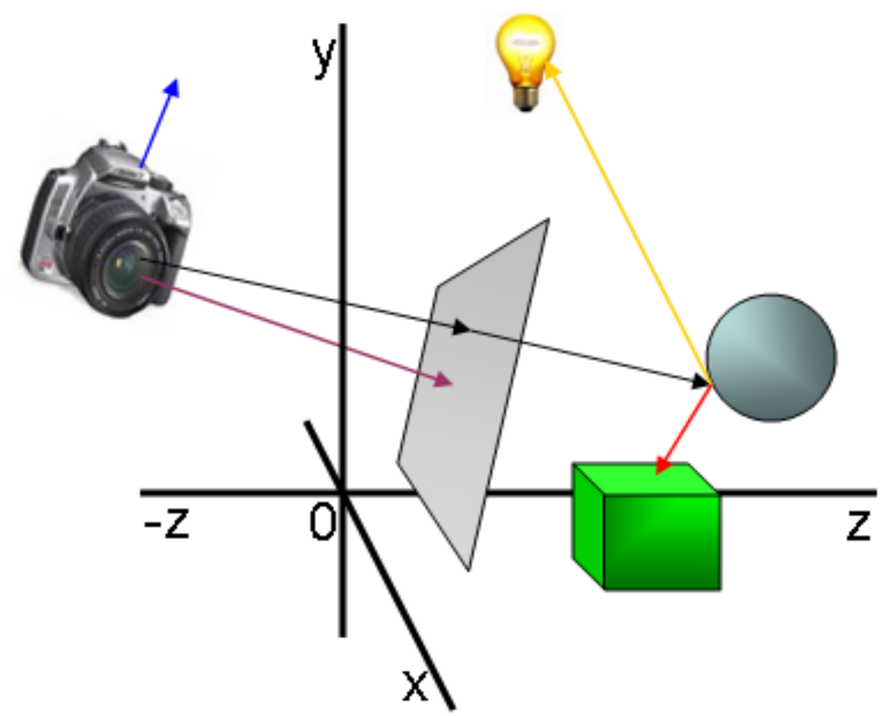
- One of the most popular methods used in 3D computer graphics to render an image
- Different from the rasterisation-based approach
- Good at simulating specular effects, producing shadows
- Also used as a function for other global illumination techniques



Ray tracing

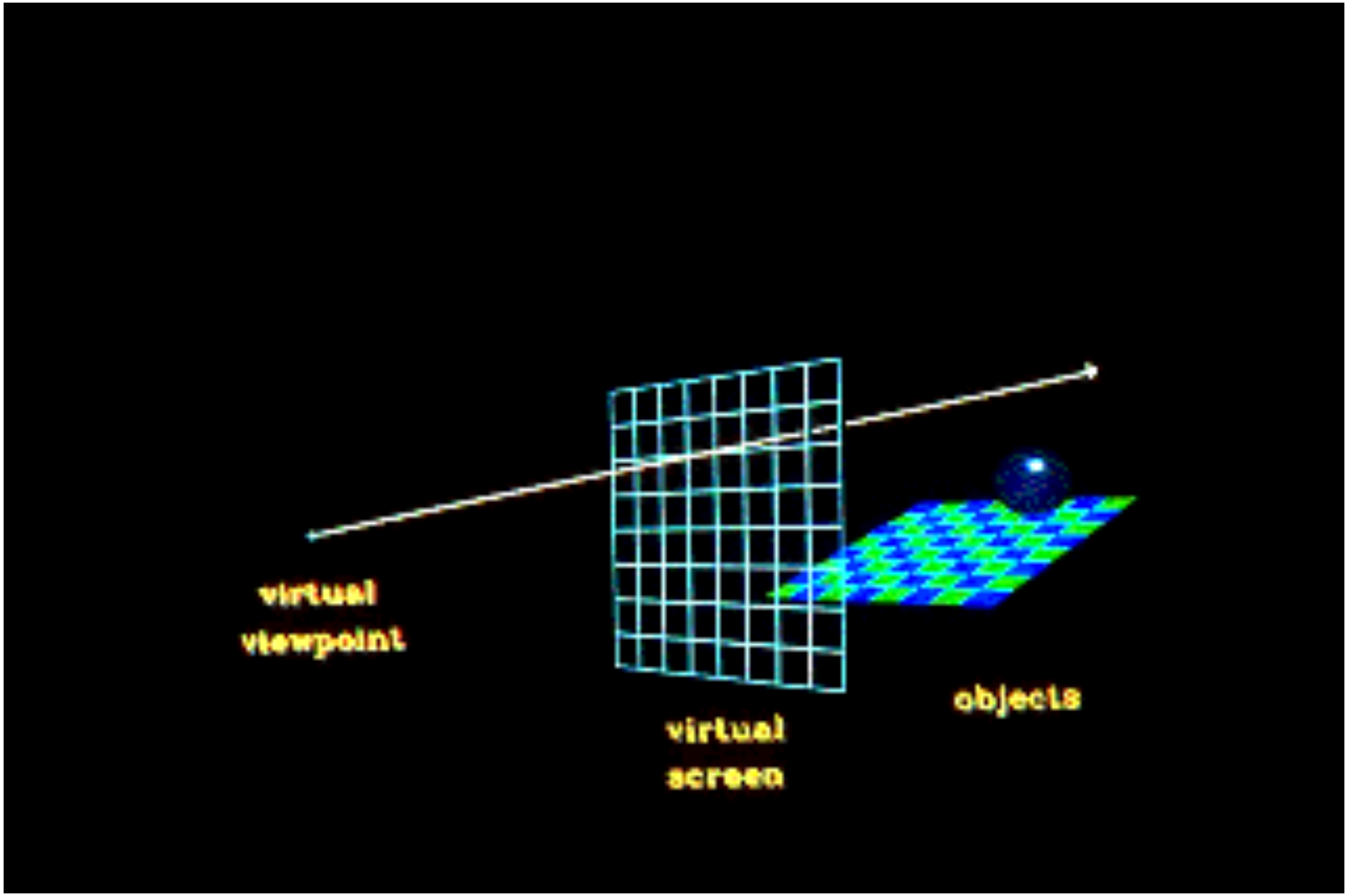
- Tracing the path taken by a ray of light through the scene

- Rays are cast to each pixel. They are reflected, refracted, or absorbed whenever they intersect objects



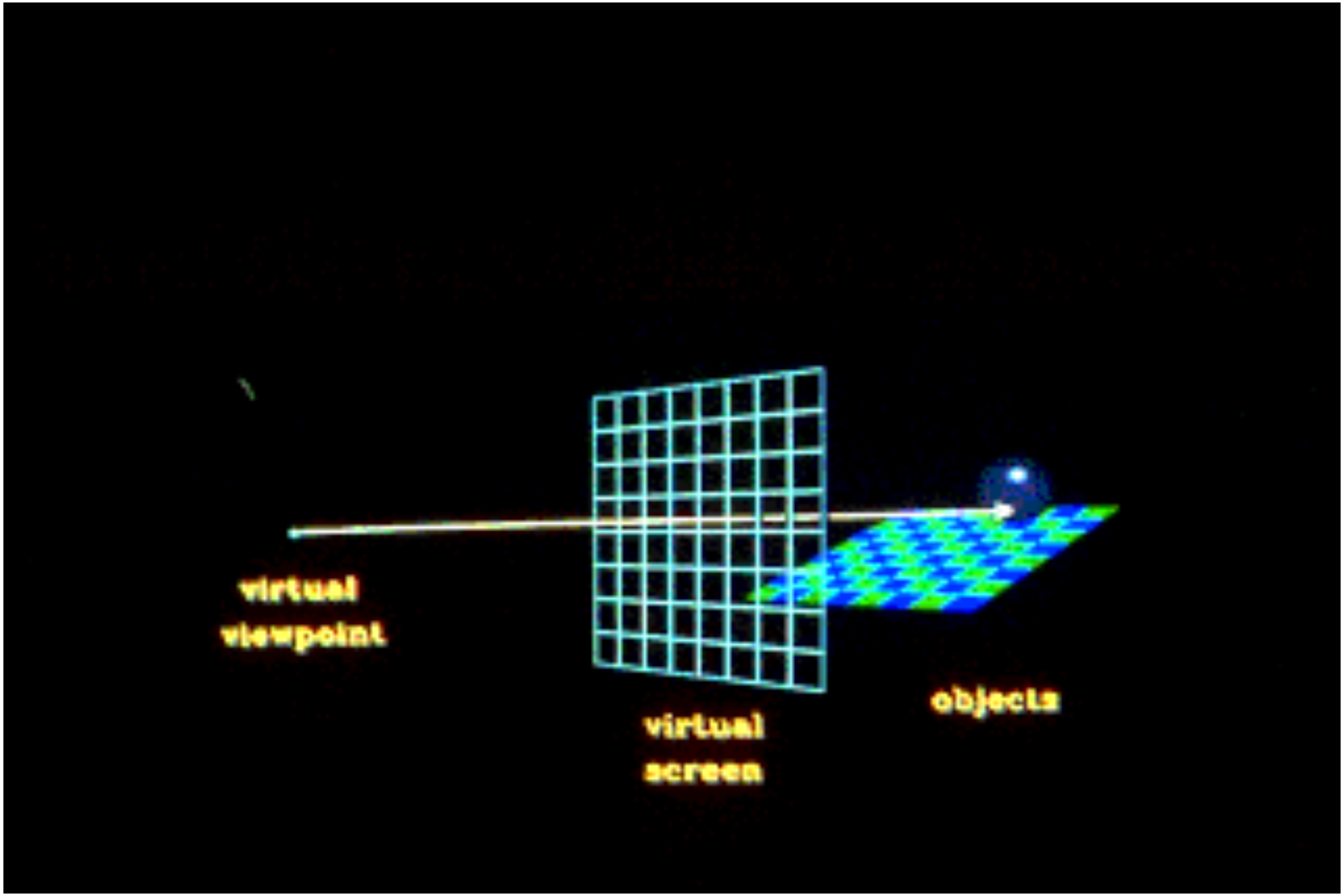
Procedure

- Rays that miss the objects are coloured as the background



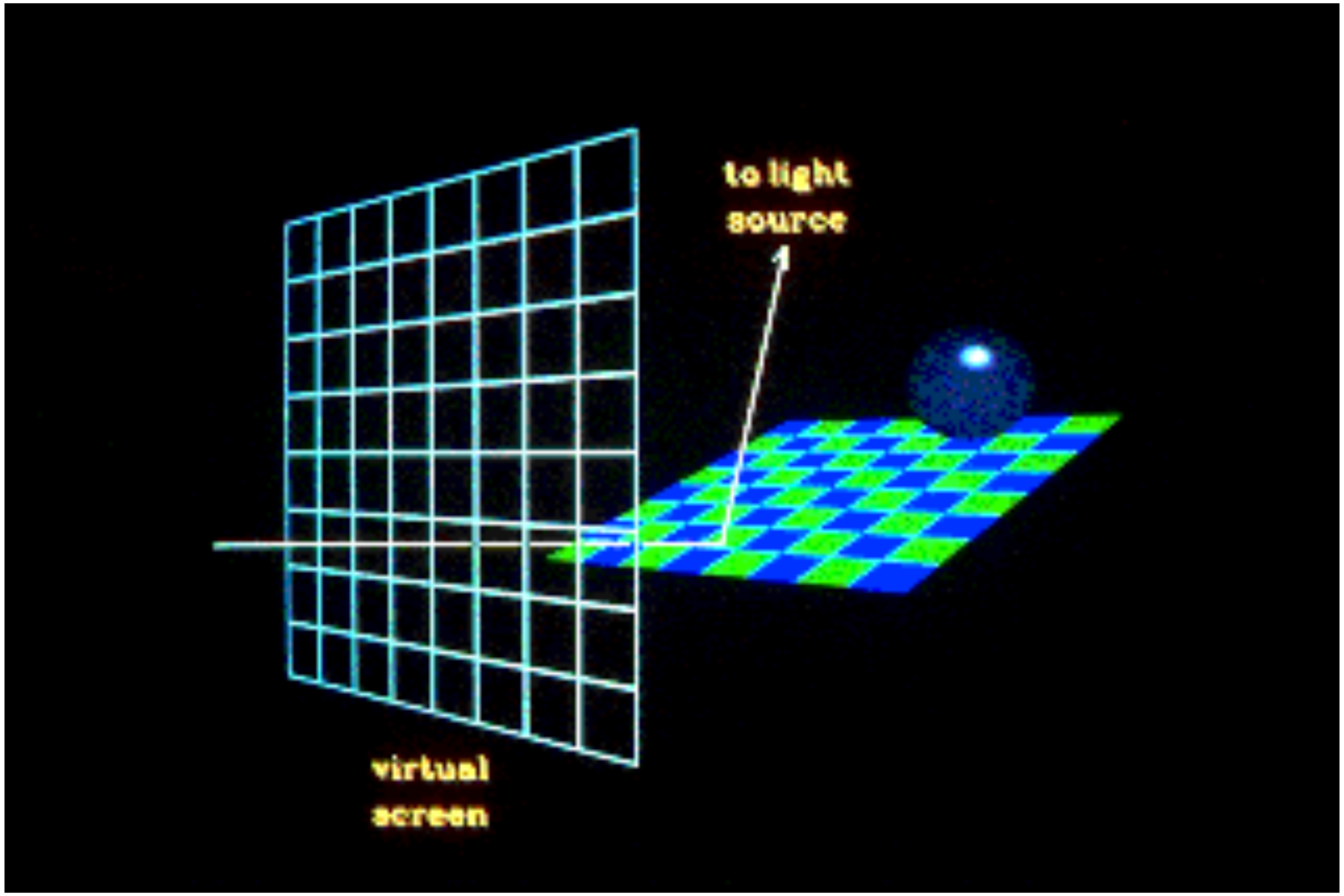
Procedure

- When a ray hits an object...



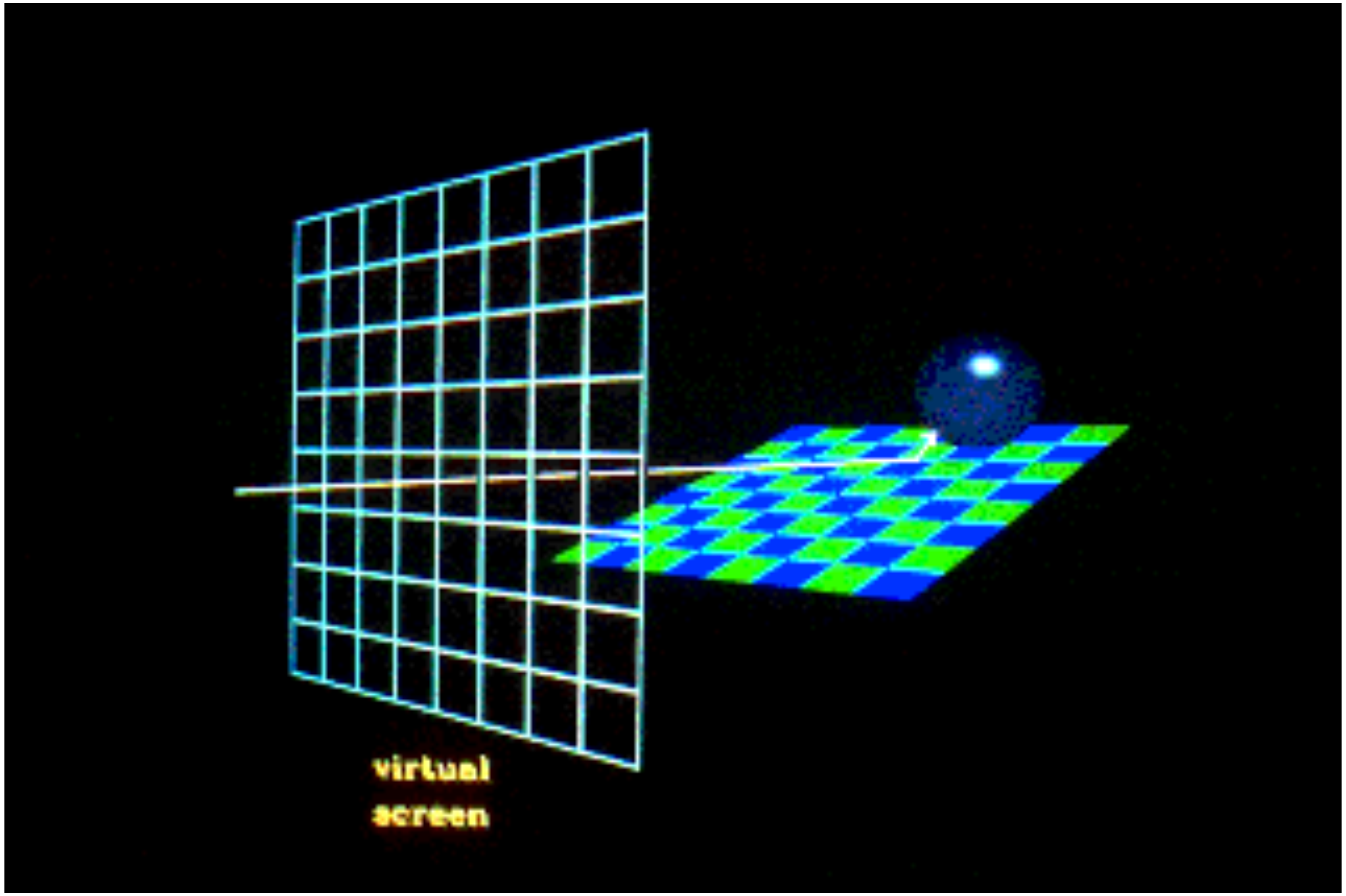
Procedure

- Check for shadowing:
  - Cast a shadow ray towards each light source

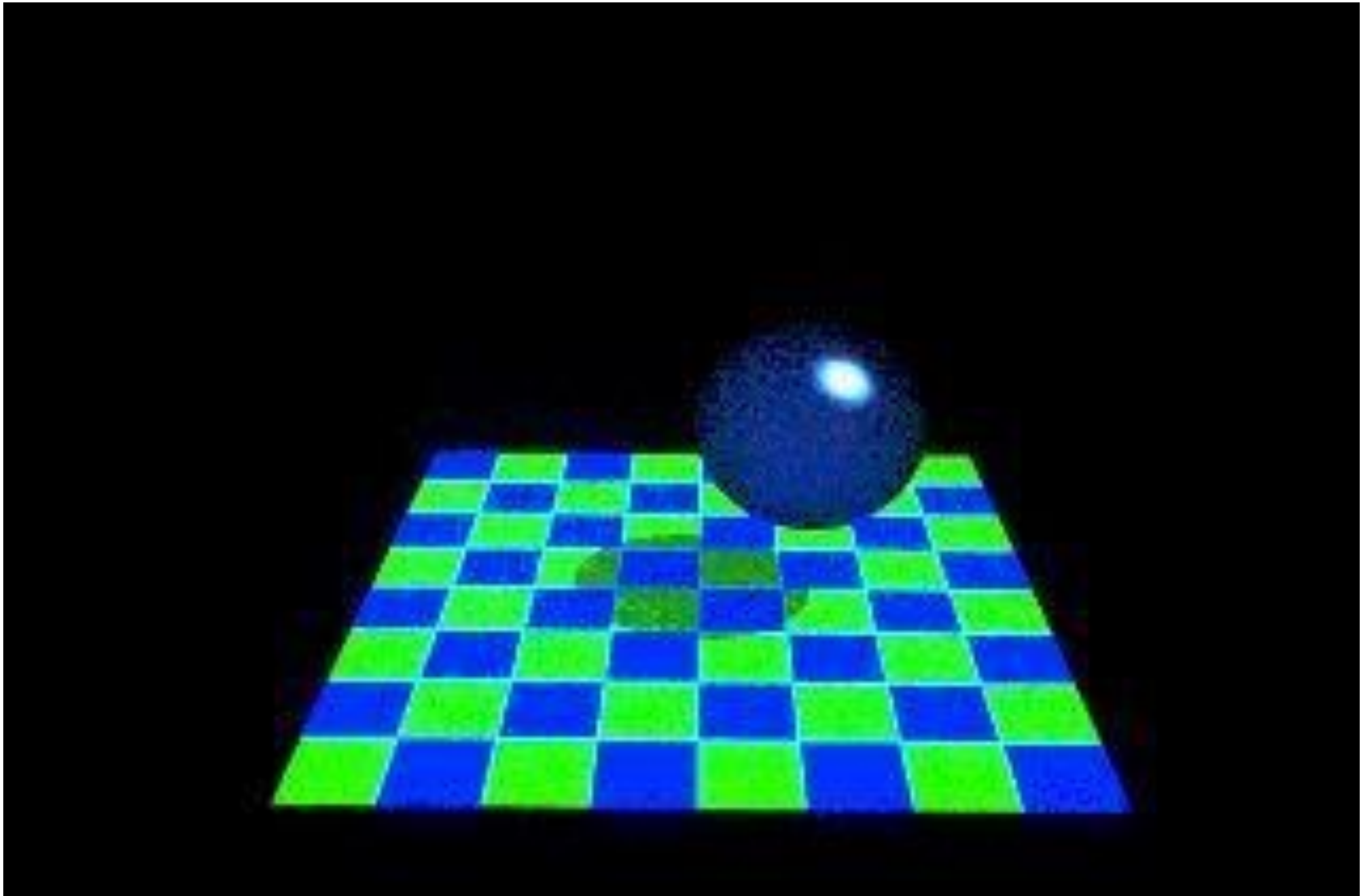


Shadow rays

- If shadow ray hits another object, only apply ambient lighting
- Otherwise perform local Phong illumination

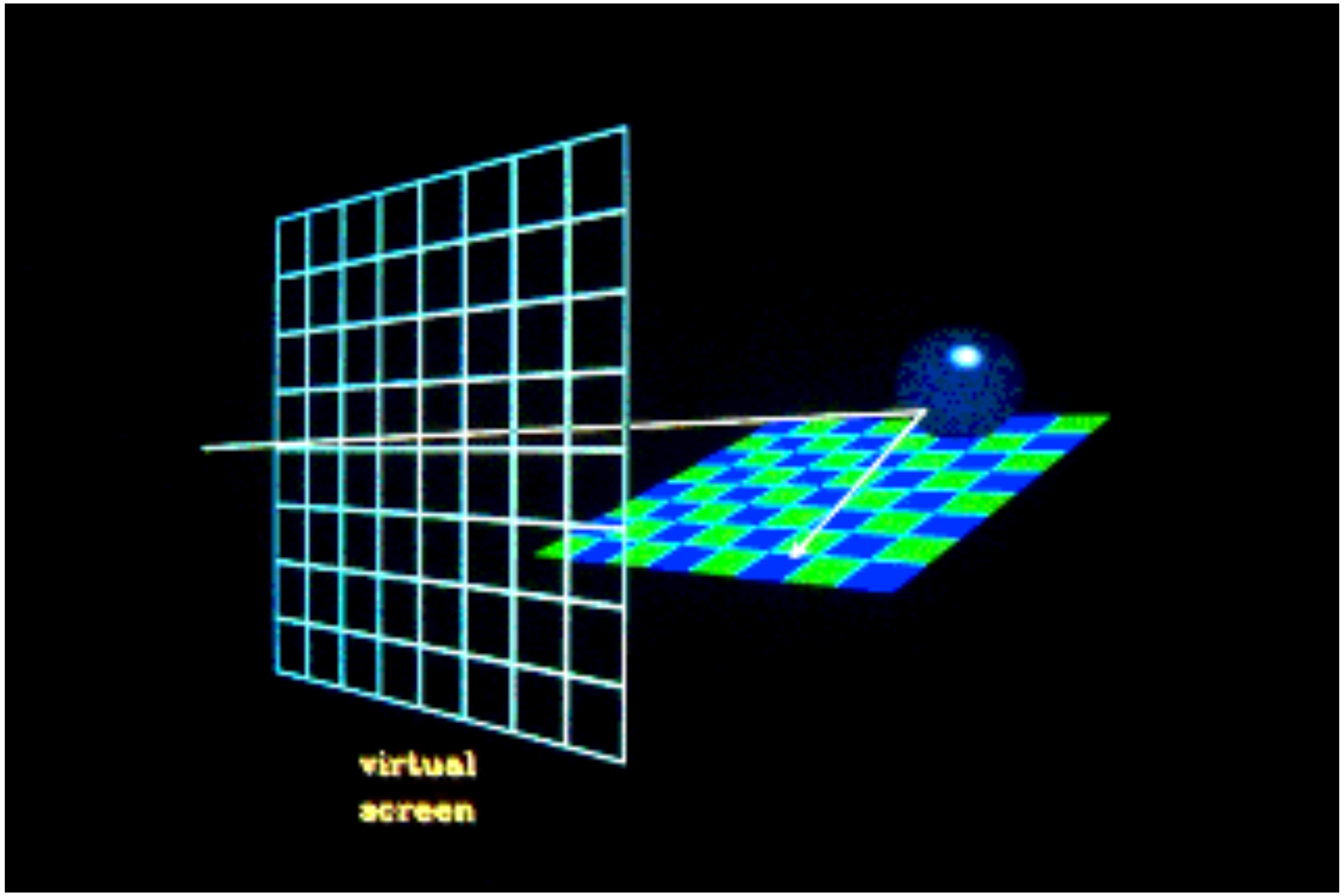


Shadow rays



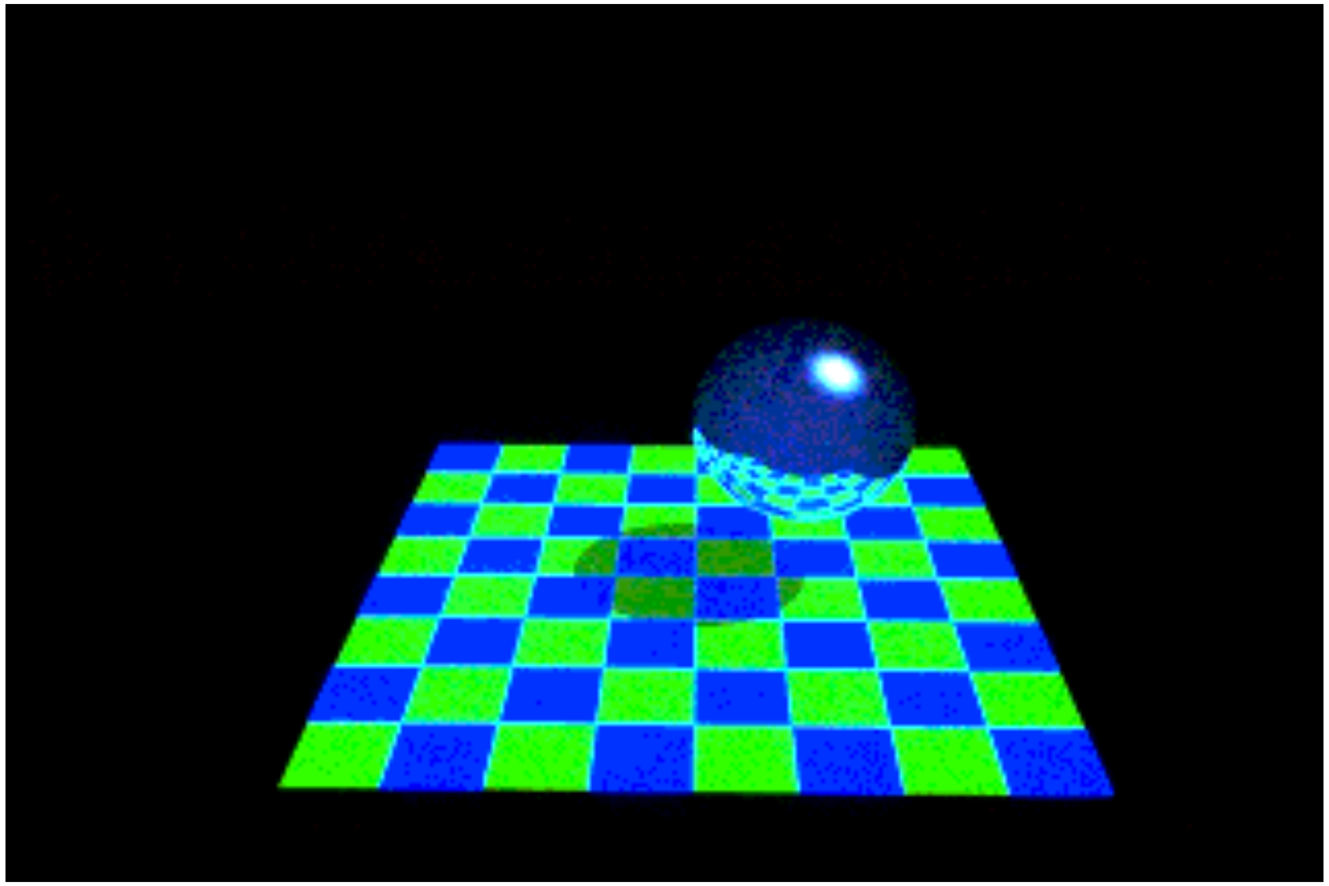
Reflected rays

- Also generate a reflected ray, and test for intersections with the scene



Reflected rays

- If the reflected ray intersects an object, apply local illumination at intersection point, and return result to original intersection point



Refracted rays

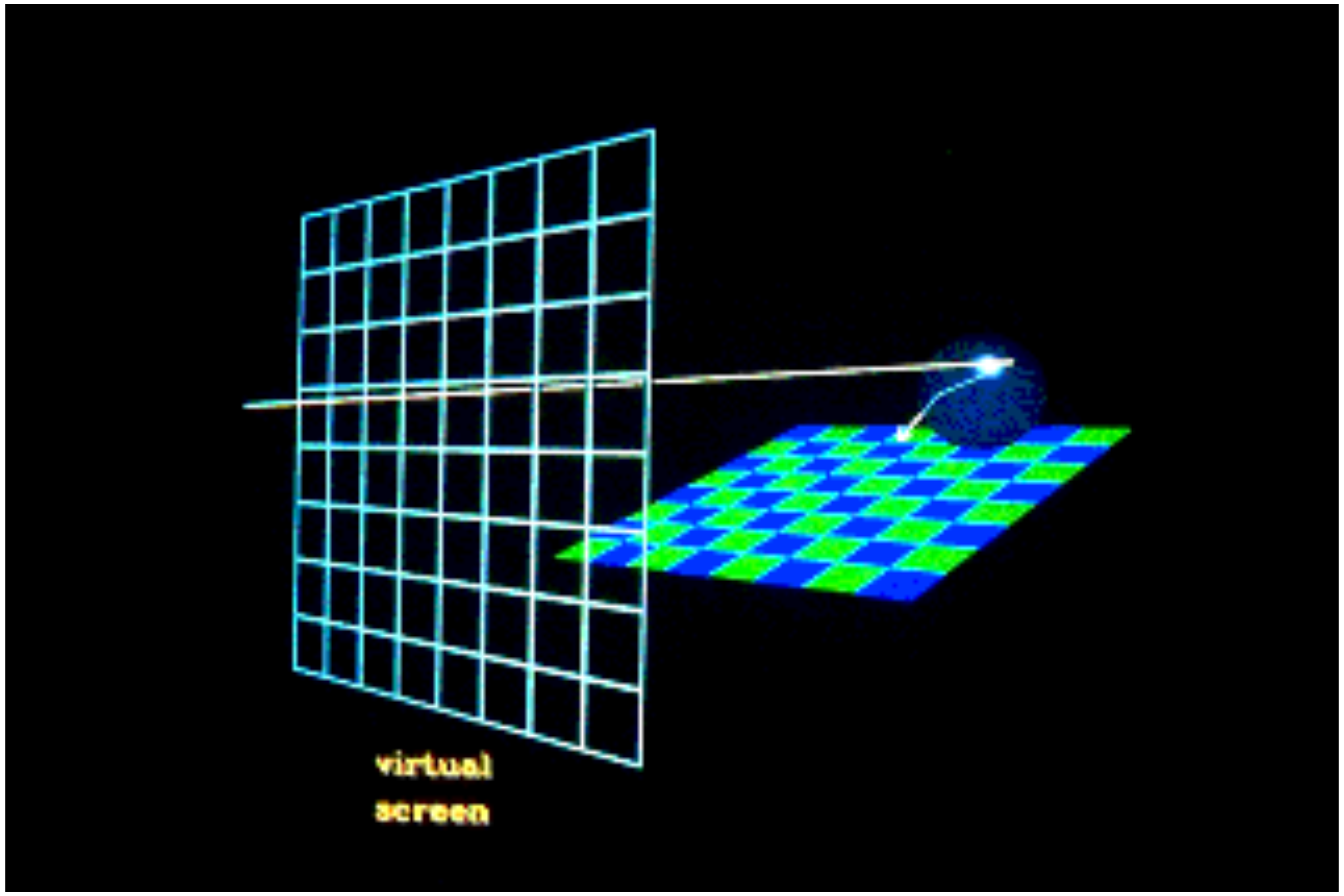
- If the object is transparent, calculate refracted ray based on Snell's law

$$T = rI + (w - k)n$$

$$r = \frac{n_1}{n_2}$$

$$w = -(I \cdot n)r$$

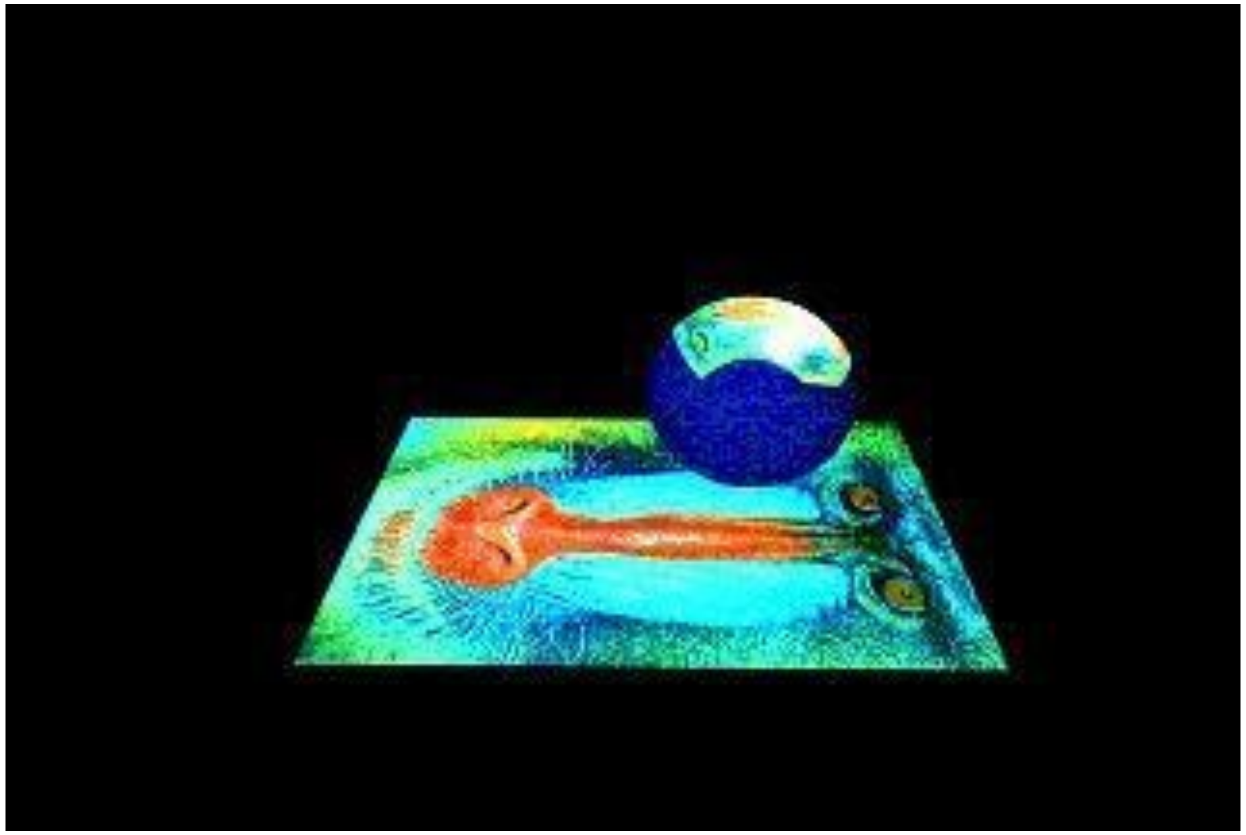
$$k = \sqrt{1 + (w - r)(w + r)}$$



*

Refracted rays

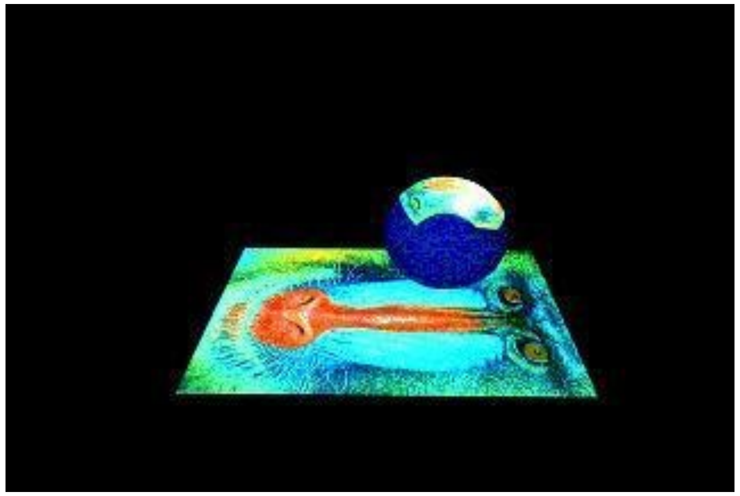
- As with reflection, calculate local illumination of intersection of refracted ray, and return to original intersection



*

Ray tracing outline

- Shadow ray
- Reflection ray
- Refraction ray
- Combine contributions from each ray:

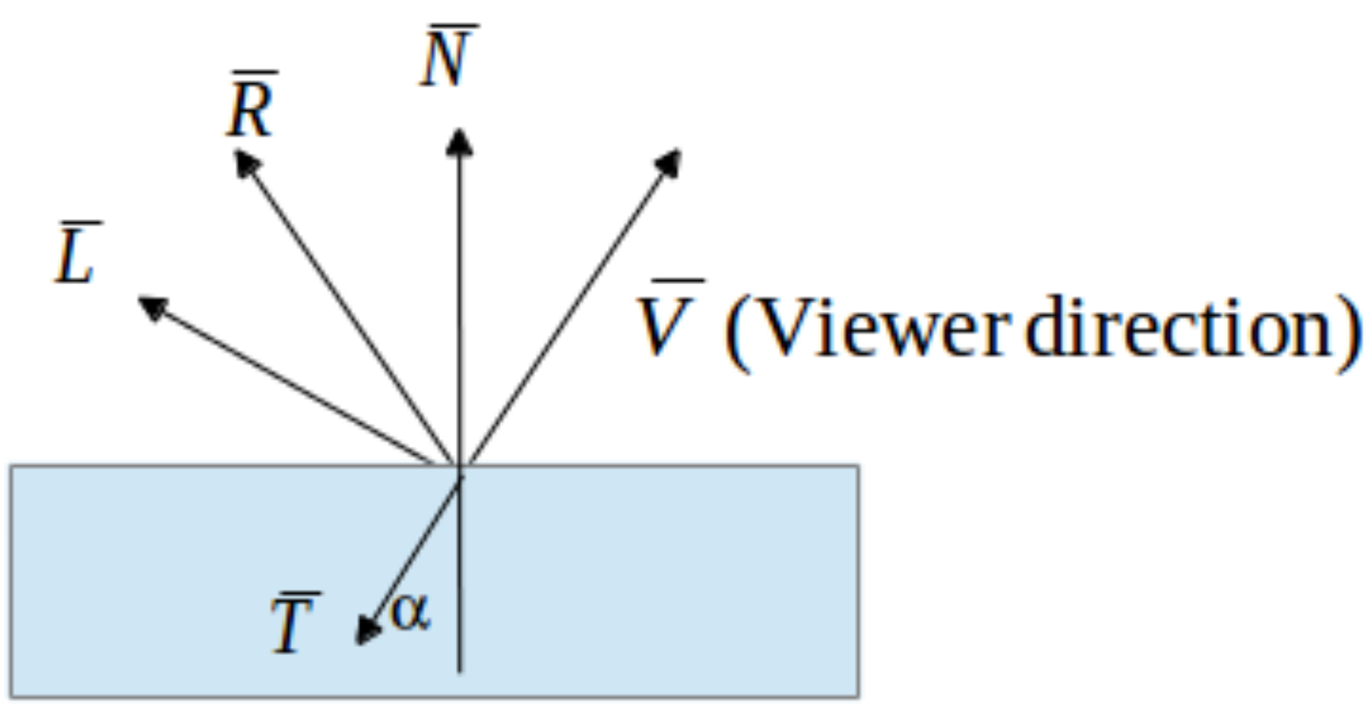


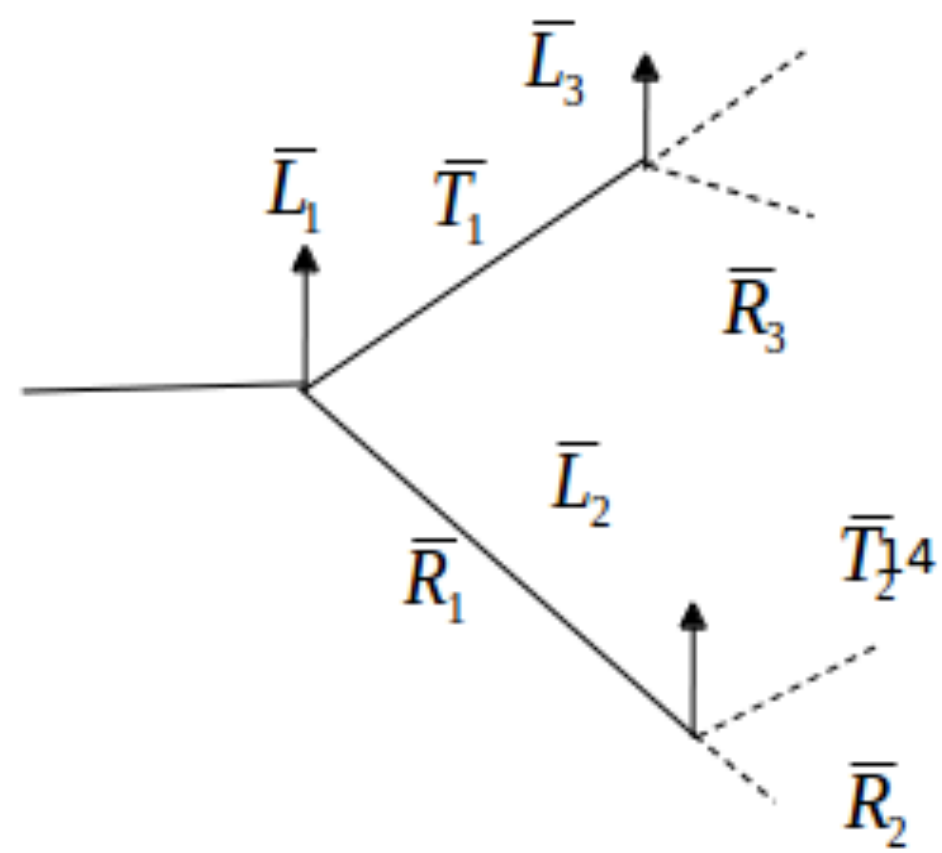
$$I = I_{local} + k_r R + k_t T$$

*

Ray Tree (Whitted '80)

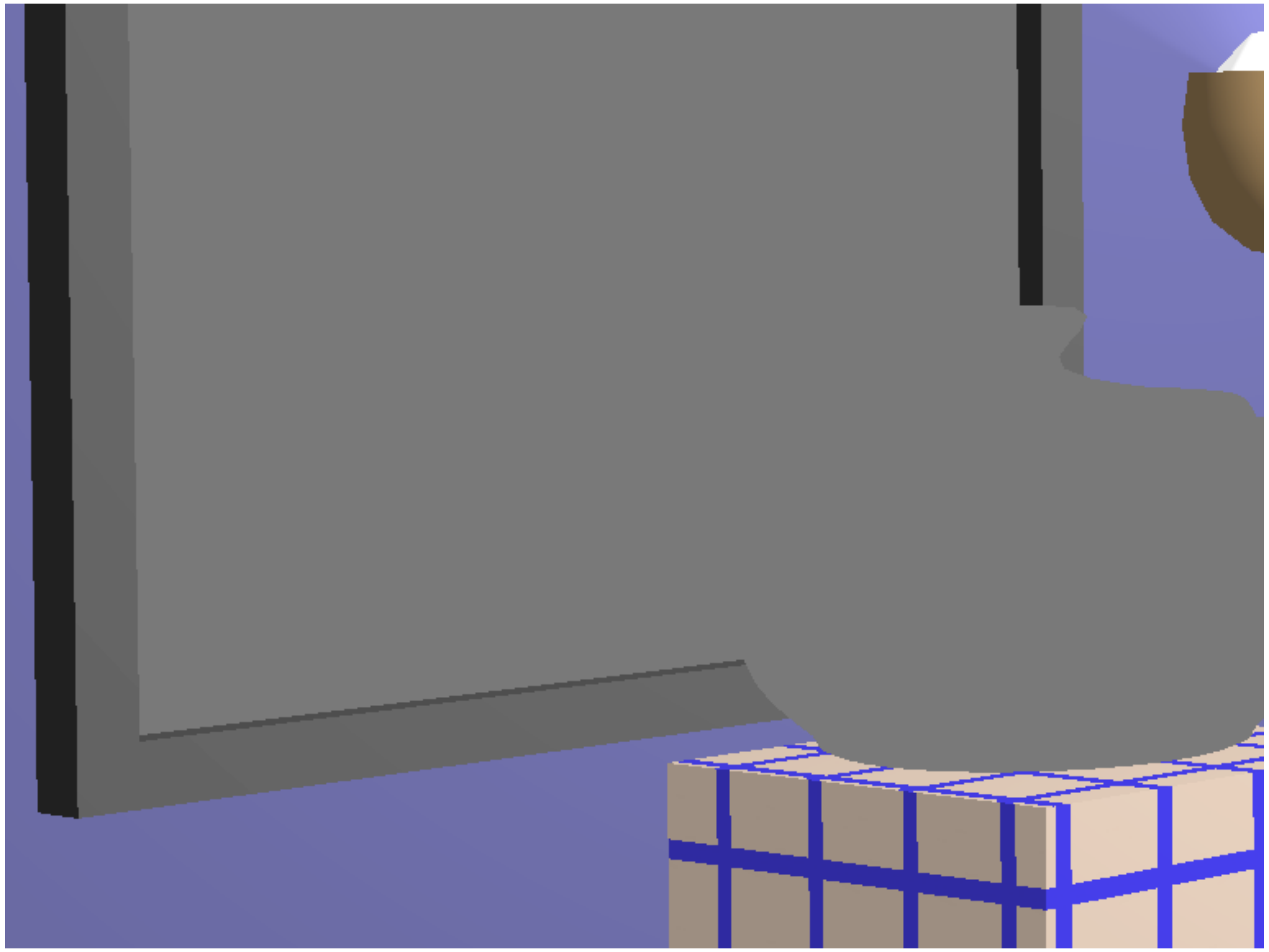
- Reflection and refraction rays are recursively cast on hitting a surface
- Performed to some depth and then returned to the previous hits





Test scene

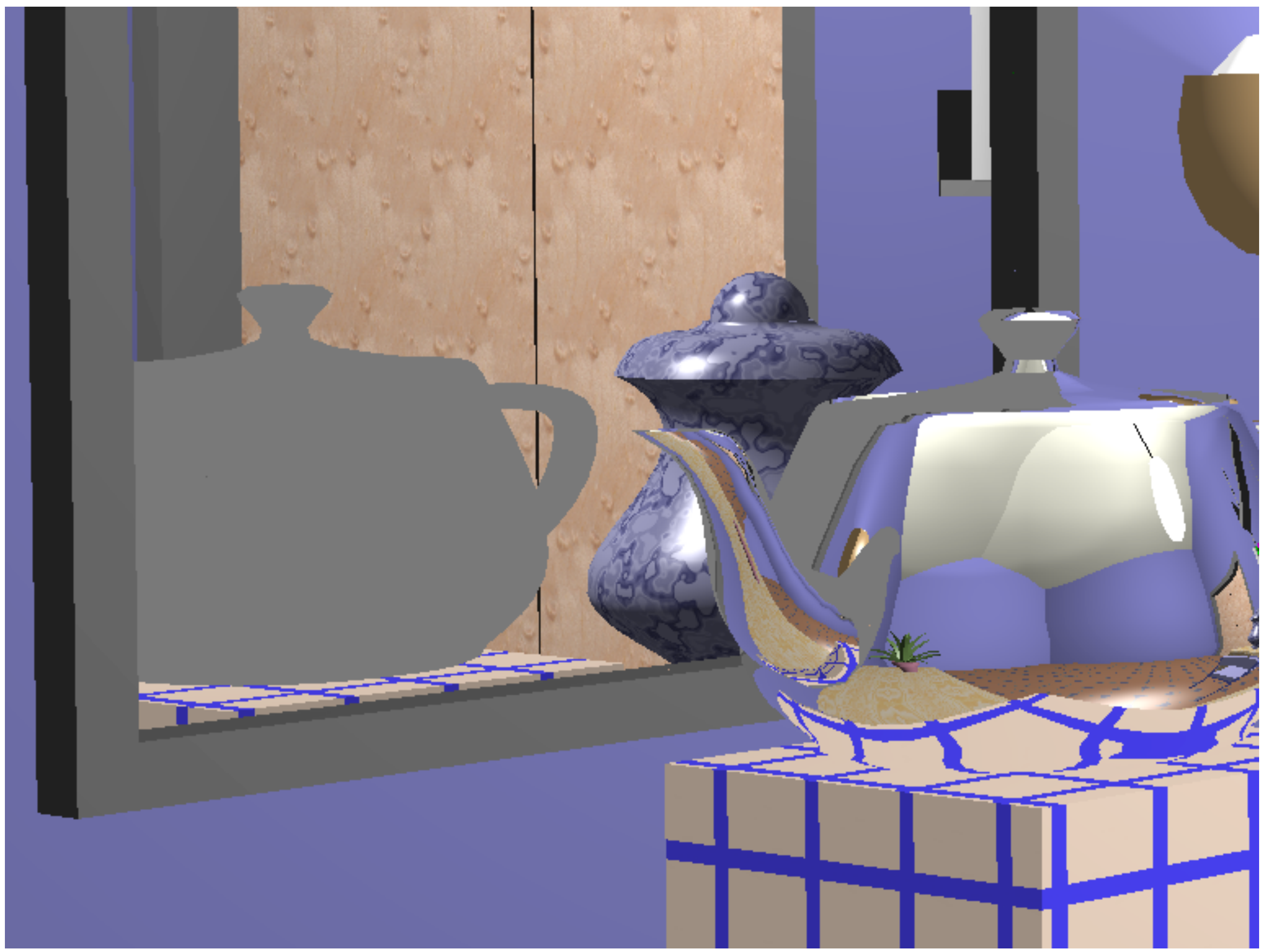
- Ray tree of depth 1. Mirror and teapot are reflective but no reflected ray is cast



*

Test scene

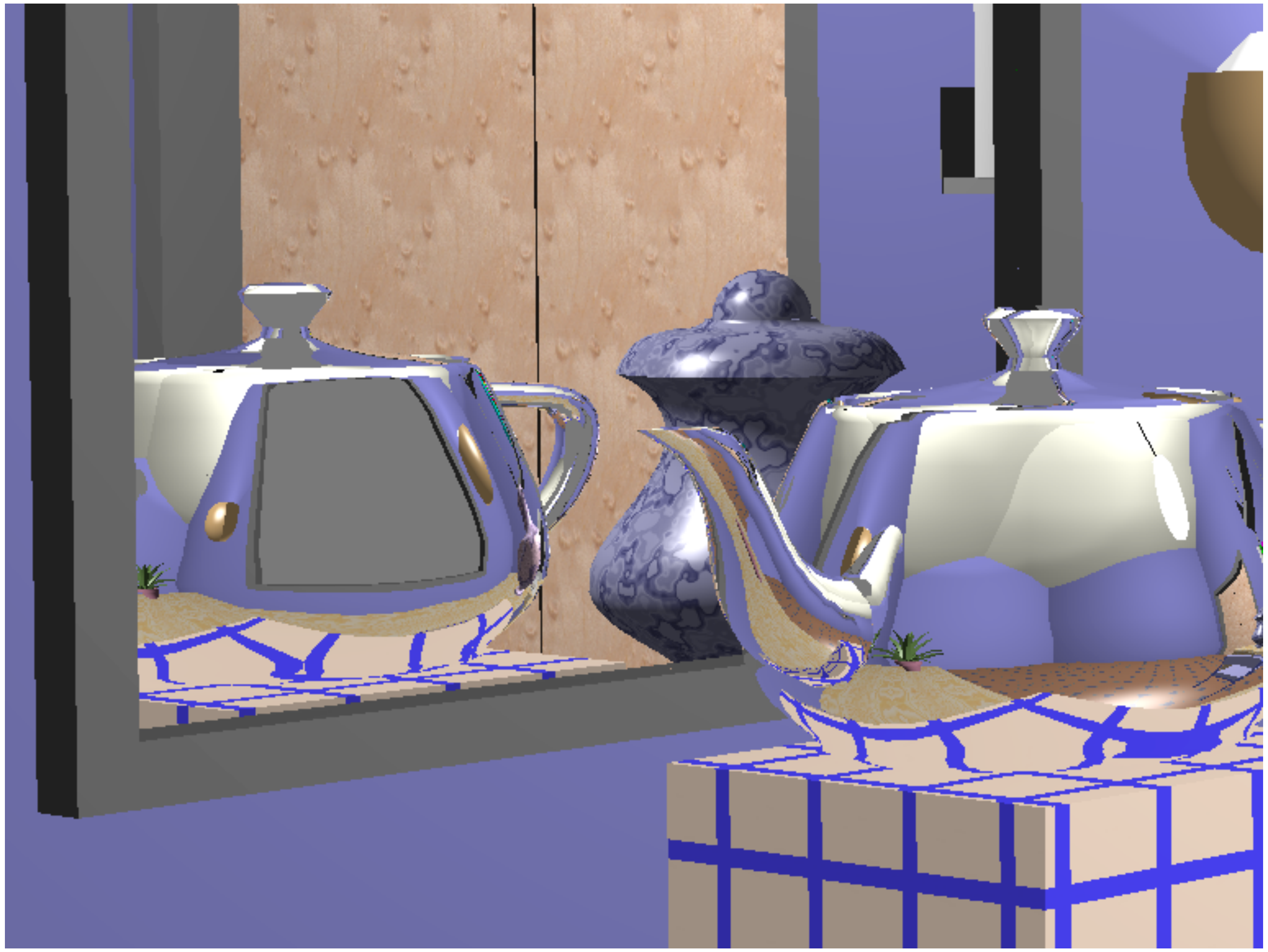
- Ray tree of depth 2. Reflection of mirror and teapot have no reflections on them!



*

Test scene

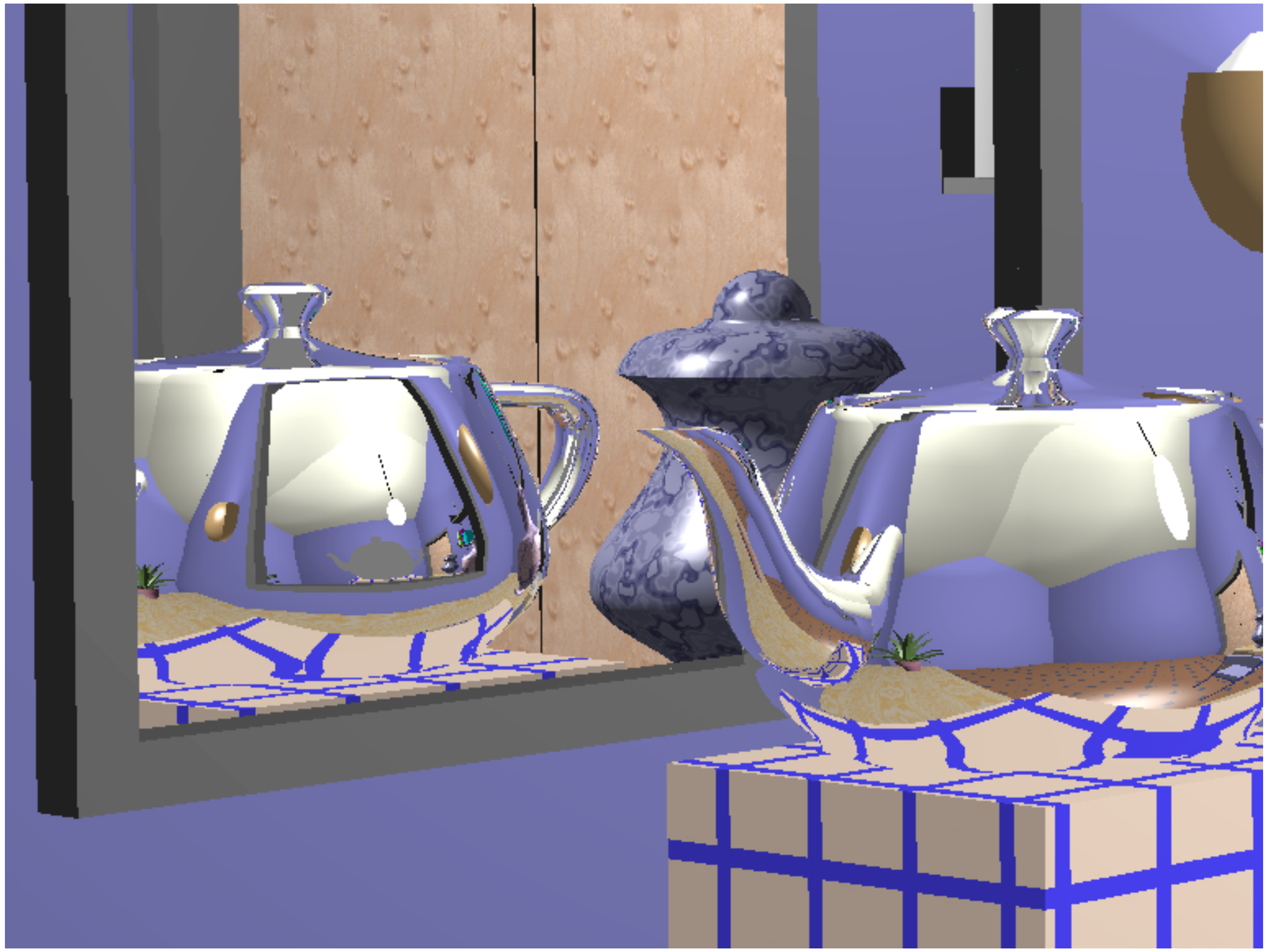
- Ray tree of depth 3. Reflection of mirror on reflected teapot has no reflection.



*

Test scene

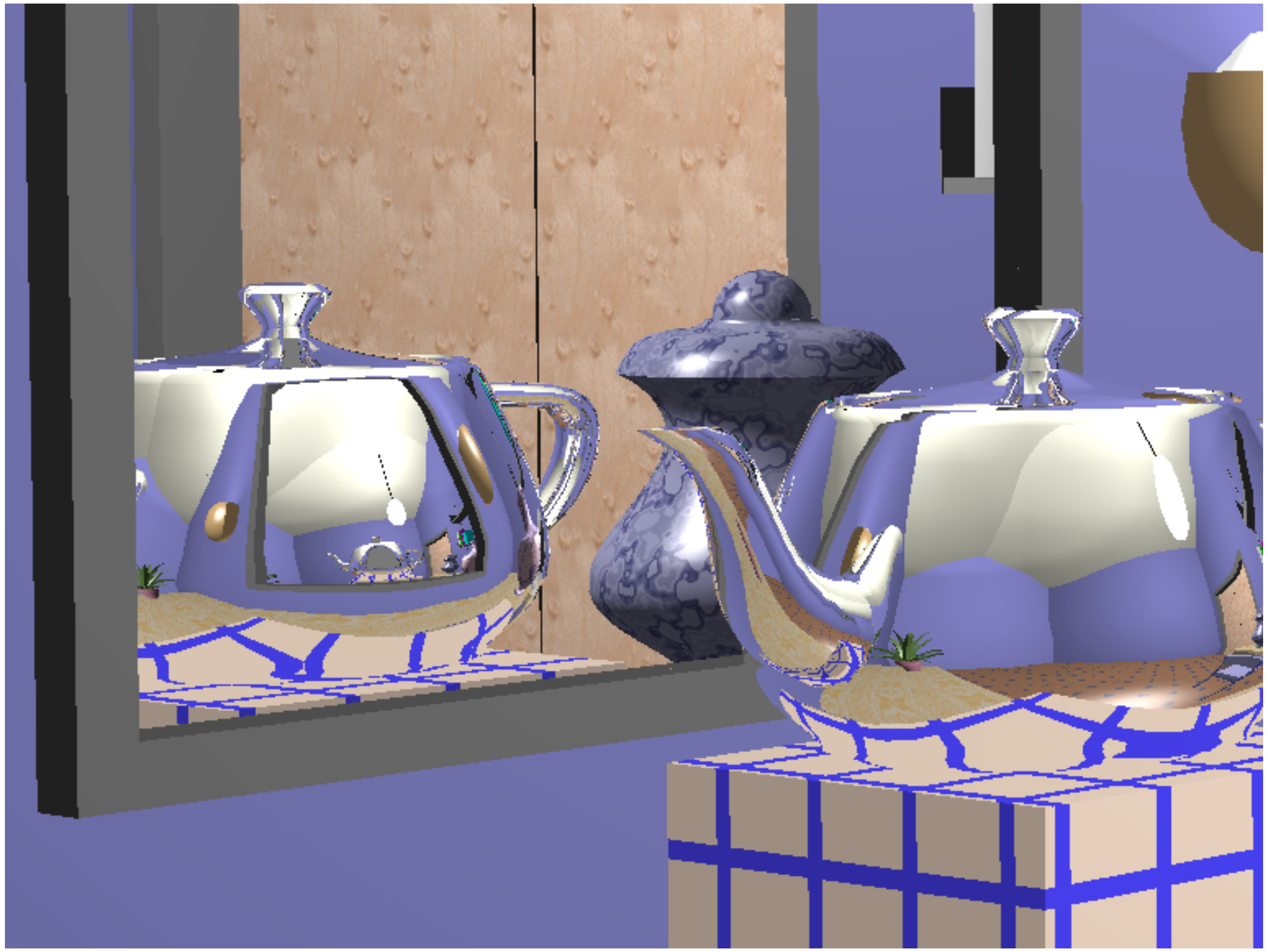
- Ray tree of depth 4. No reflection on teapot in reflection of mirror on teapot in mirror.



*

Test scene

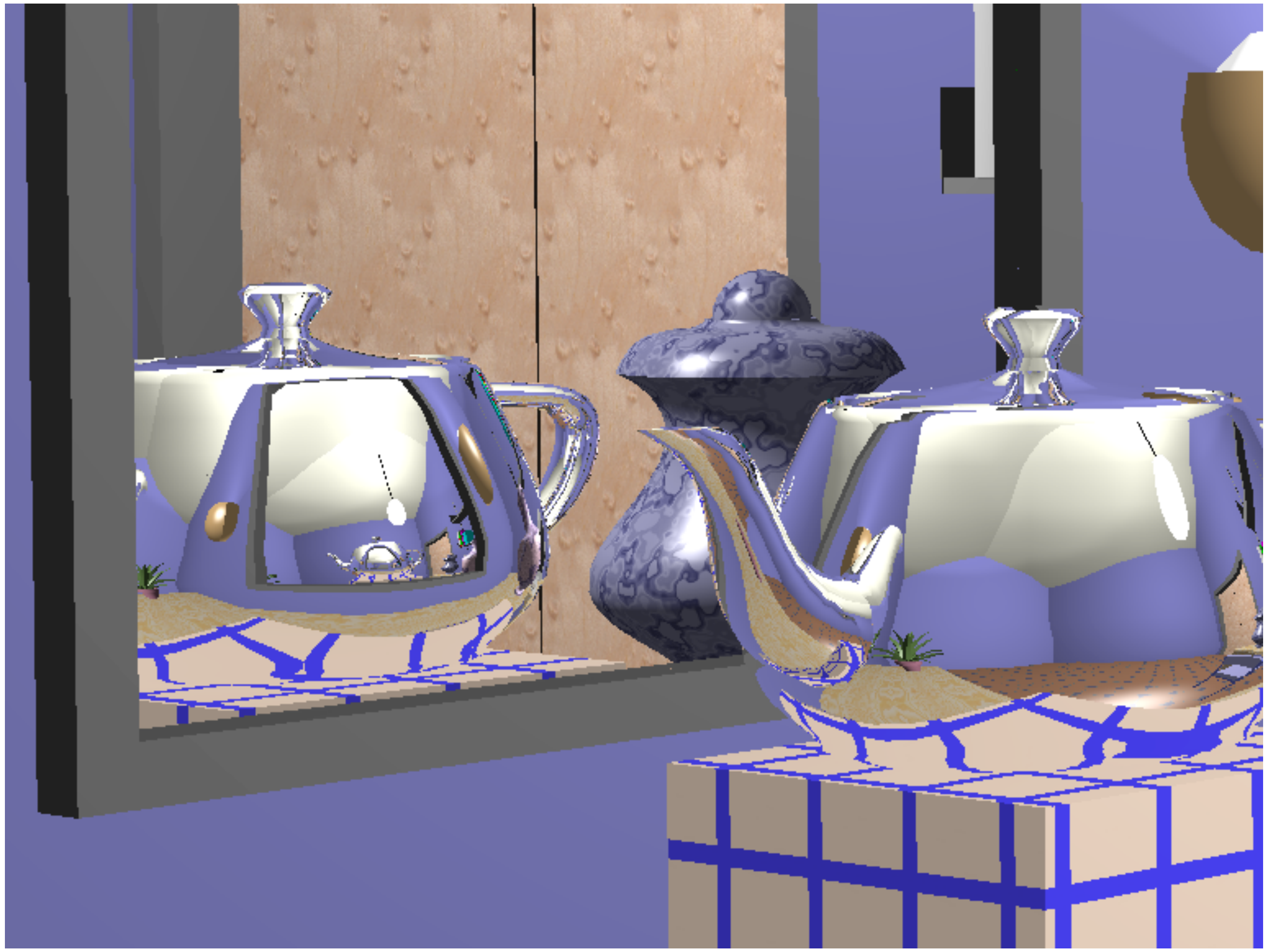
- Ray tree of depth 5...



*

Test scene

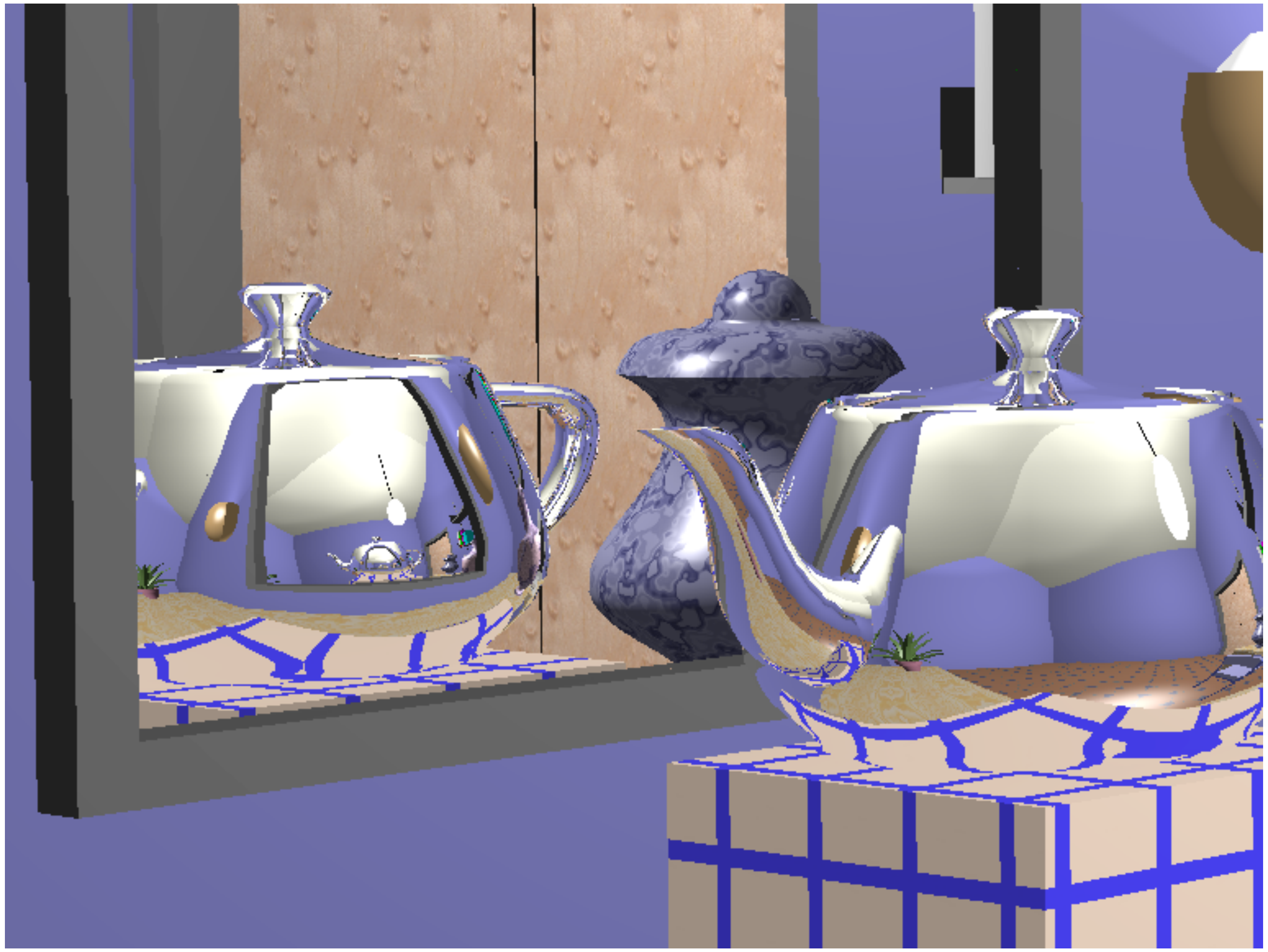
- Ray tree of depth 6...



*

Test scene

- Ray tree of depth 7...



*

Ray trees on a specular surface

- Compute the colour of each ray:

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

- In one single equation:

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

Stopping

- Need to decide when to stop:
  - When we hit a completely diffuse surface
  - On specular surfaces at some fixed depth
  - Once the product of coefficients falls below a 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

Complexity?

- Ray tracing - at a resolution of w by h, and N triangles, O(?)
- Rasterisation - with V vertices and N triangles, O(?)

Overview

- Ray tracing overview
- Ray trees
- **Intersections**
    - Spheres
    - Planes
    - Polygons
- Bounding volumes
    - Bounding volume hierarchies

Parametric representation of rays

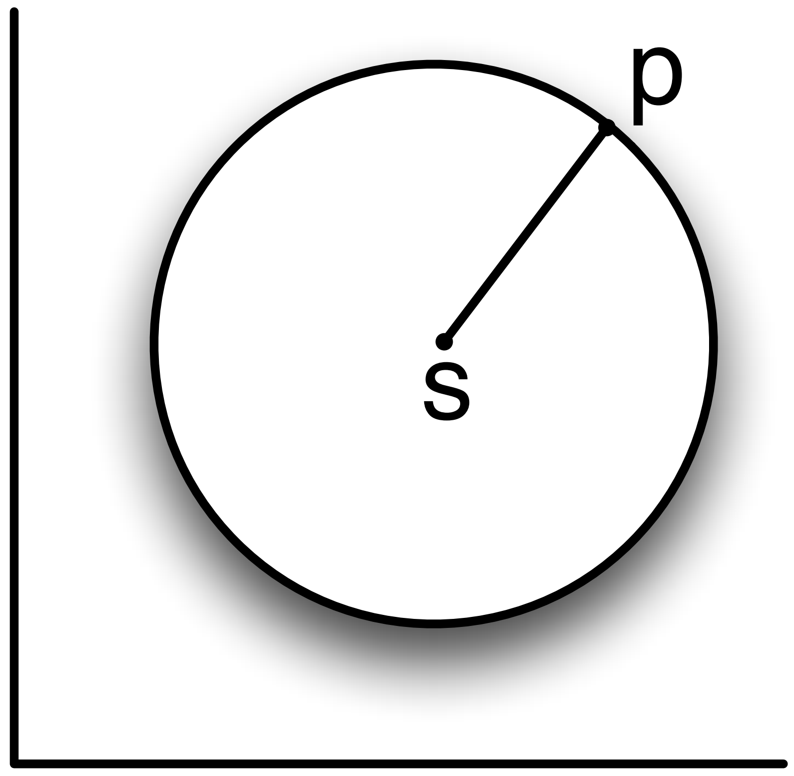
- Ray is a line from some origin $\boldsymbol{e}$ in direction $\boldsymbol{d}$. E.g. starting at the camera in the direction of the pixel, or starting on the surface in the direction of reflection or refraction
- Given an object represented by an implicit surface we can find the value of t at which the ray intersects the object
- Knowing t at the intersection we can calculate the coordinates of the intersection

$$\boldsymbol{r}(t) = \boldsymbol{e} + t\boldsymbol{d}$$

Implicit representation of spheres

We can represent a sphere using an implicit equation of the form $f(\boldsymbol{p}) = 0$.

A sphere is defined by $(x - s_x)^2 + (y - s_y)^2 + (z - s_z)^2 = r^2$, so for a sphere with center at coordinates $\boldsymbol{s}$ and of radius r:



$$(\boldsymbol{p} - \boldsymbol{s}) \cdot (\boldsymbol{p} - \boldsymbol{s}) - r^2 = 0$$

Ray/sphere intersection

To find the intersection of a ray with a sphere, we substitute $\boldsymbol{r}(t) = \boldsymbol{e} + t\boldsymbol{d}$ into the implicit equation for a sphere:

$$\begin{aligned}(\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{s}) \cdot (\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{s}) - r^2 &= 0 \\ (\boldsymbol{d} \cdot \boldsymbol{d})t^2 + 2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{s})t + (\boldsymbol{e} - \boldsymbol{s}) \cdot (\boldsymbol{e} - \boldsymbol{s}) - r^2 &= 0\end{aligned}$$

This is a quadratic equation in t, e.g. $at^2 + bt + c = 0$, and so we can find the solutions for t using:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

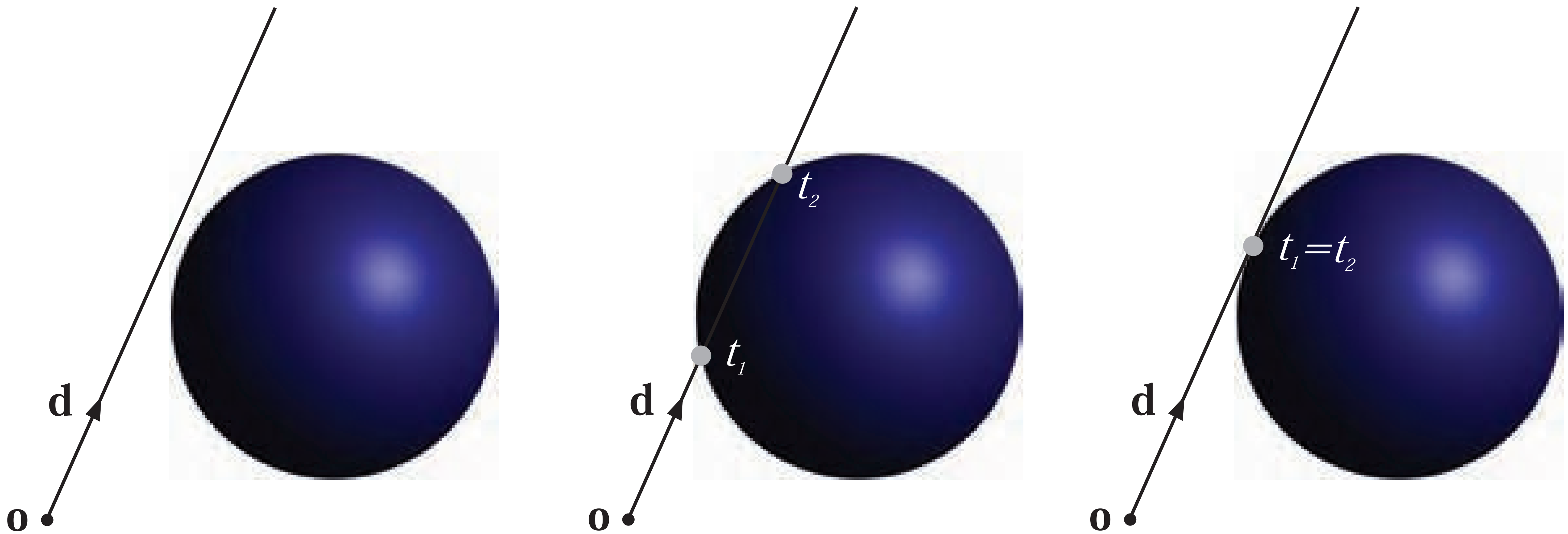
Ray/sphere intersection

This gives us the solution for t as:

$$t = \frac{-2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{s}) \pm \sqrt{(2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{s}))^2 - 4(\boldsymbol{d} \cdot \boldsymbol{d})((\boldsymbol{e} - \boldsymbol{s}) \cdot (\boldsymbol{e} - \boldsymbol{s}) - r^2)}}{2(\boldsymbol{d} \cdot \boldsymbol{d})}$$

With the number of solutions determined by the value in the square root.

- If $b^2 - 4ac > 0$ there are two intersections of the ray with the sphere
- If $b^2 - 4ac = 0$ the ray grazes the sphere and there is a single intersection
- If $b^2 - 4ac < 0$ the ray misses the sphere completely.



$$r(t) = o + dt$$

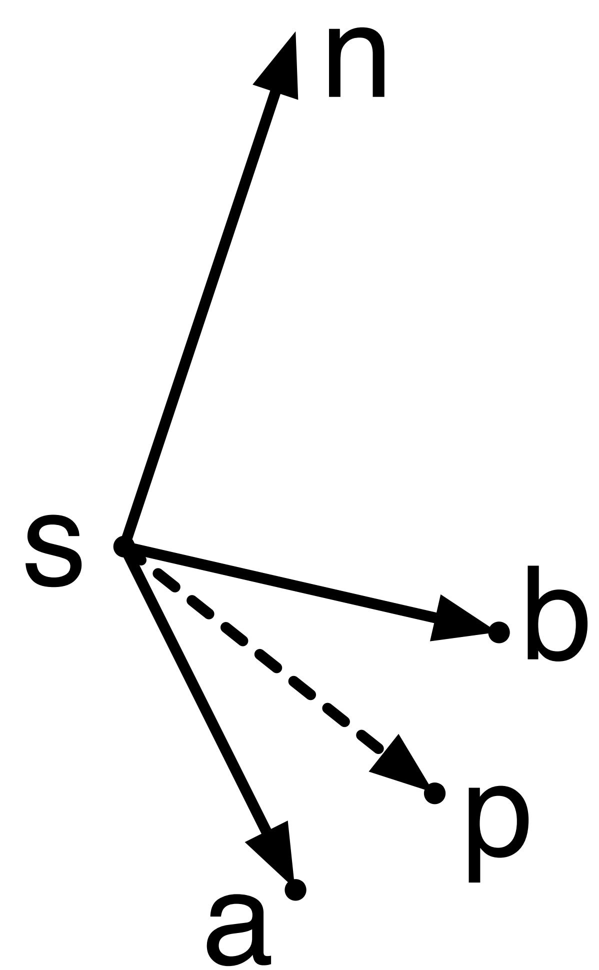
Implicit representation of planes

A plane can be described by the implicit equation

$$(\boldsymbol{p} - \boldsymbol{s}) \cdot \boldsymbol{n} = 0$$

where $\boldsymbol{s}$ is a point on the plane, and $\boldsymbol{n}$ is the normal vector to the plane. Points $\boldsymbol{p}$ satisfying this equation lie on the plane.
For points $\boldsymbol{a}$, $\boldsymbol{b}$ on the plane:

$$\boldsymbol{n} = (\boldsymbol{a} - \boldsymbol{s}) \times (\boldsymbol{b} - \boldsymbol{s})$$
$$(\boldsymbol{p} - \boldsymbol{s}) \cdot ((\boldsymbol{a} - \boldsymbol{s}) \times (\boldsymbol{b} - \boldsymbol{s})) = 0$$



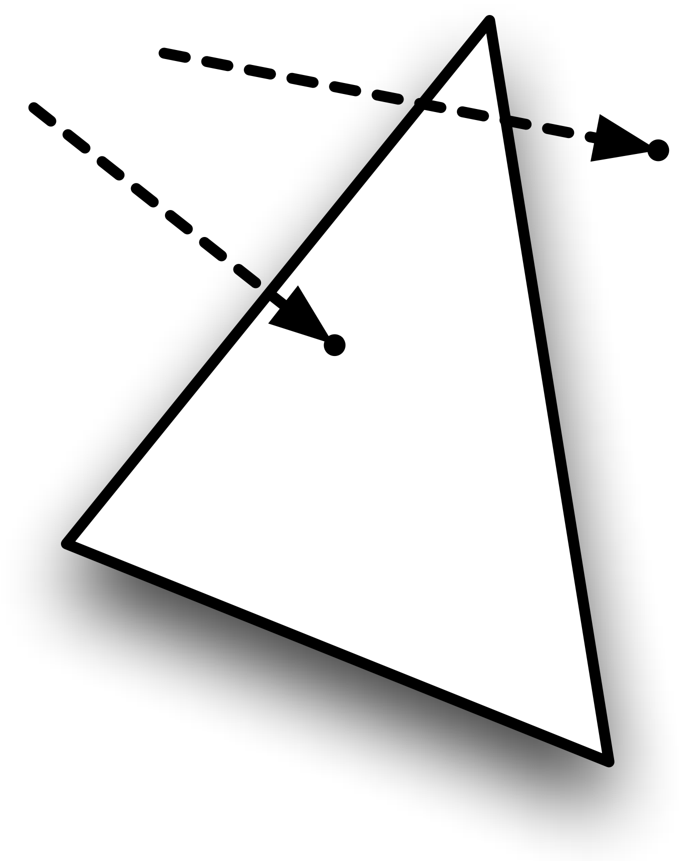
Ray/plane intersections

To calculate the intersection of a ray with a plane we substitute the equation for the points on the ray into the implicit plane equation:

$$
\begin{aligned}
(\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{s}) \cdot \boldsymbol{n} &= 0 \\
(\boldsymbol{e} - \boldsymbol{s}) \cdot \boldsymbol{n} + t\boldsymbol{d} \cdot \boldsymbol{n} &= 0 \\
t &= \frac{(\boldsymbol{s} - \boldsymbol{e}) \cdot \boldsymbol{n}}{\boldsymbol{d} \cdot \boldsymbol{n}}
\end{aligned}
$$

In the case where $\boldsymbol{d} \cdot \boldsymbol{n} = 0$ the ray is parallel to the plane, and so does not intersect it.

Ray/triangle intersection



First perform intersection with the plane:

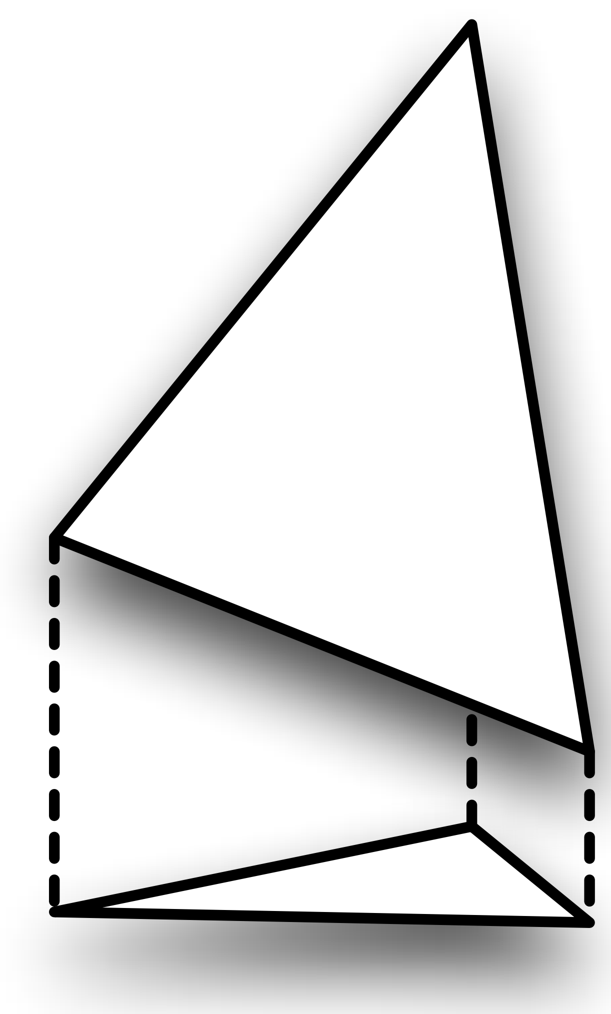
$$t = \frac{(\boldsymbol{s} - \boldsymbol{e}) \cdot \boldsymbol{n}}{\boldsymbol{d} \cdot \boldsymbol{n}}$$

Then test if the point $\boldsymbol{r(t)} = \boldsymbol{e} + t\boldsymbol{d}$ lies within the triangle.

Projection onto primary planes

To make things simpler, we project the triangle onto one of the planes corresponding to a pair of axes (xy, yz or xz).

- We chose the plane on which the triangle has the largest projection, using the normal vector $\boldsymbol{n}$.
- The largest component of $\boldsymbol{n}$ is dropped e.g. if $|n_y|$ is the largest we project onto the xz plane, dropping the y coordinate.



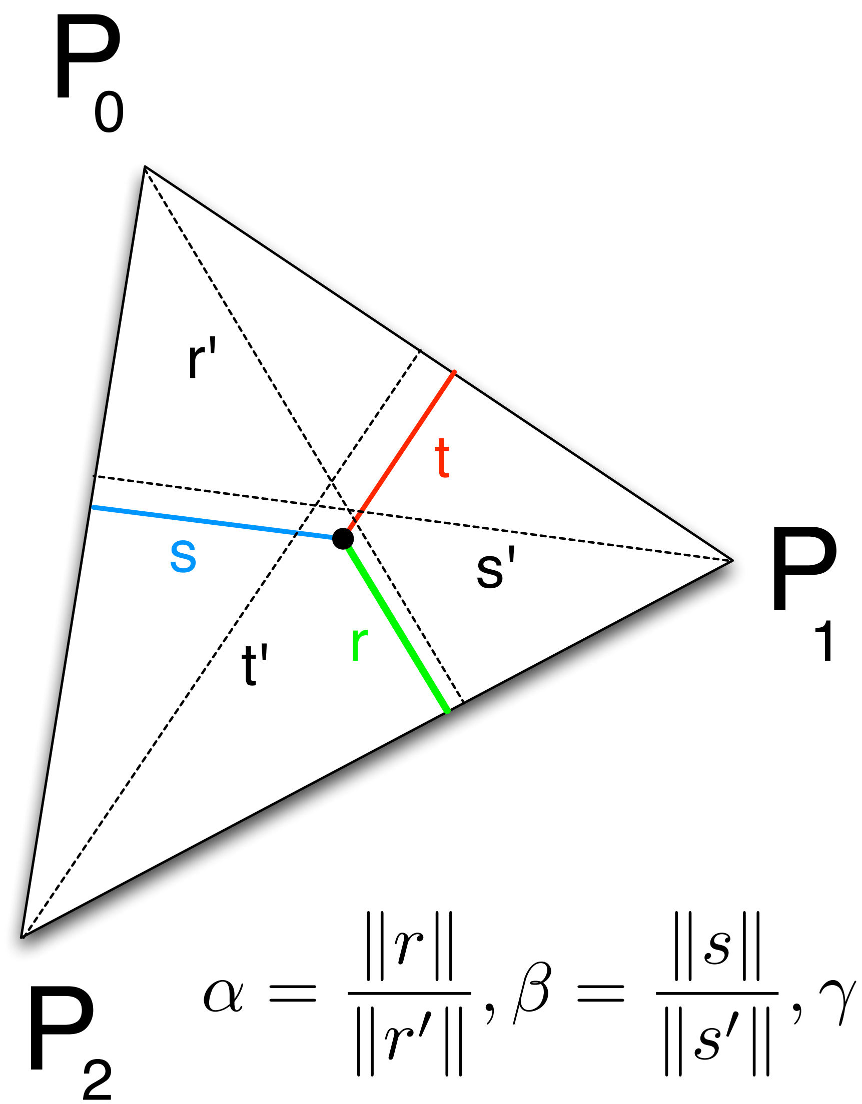
Projection onto primary planes

After projection to a 2D plane we can test for a point being inside the triangle using barycentric coordinates:

$$\alpha = \frac{f_{P_1P_2}(x, y)}{f_{P_1P_2}(x_0, y_0)}$$

$$\beta = \frac{f_{P_2P_0}(x, y)}{f_{P_2P_0}(x_1, y_1)}$$

$$\gamma = \frac{f_{P_0P_1}(x, y)}{f_{P_0P_1}(x_2, y_2)},$$



$$\alpha = \frac{\|r\|}{\|r'\|}, \beta = \frac{\|s\|}{\|s'\|}, \gamma = \frac{\|t\|}{\|t'\|}$$

where

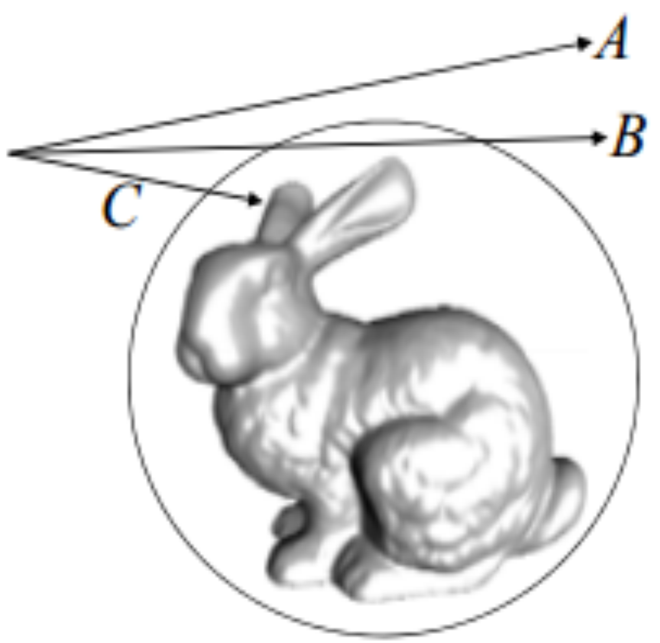
$$f_{pq}(x, y) = (y_q - y_p)x - (x_q - x_p)y + x_q y_p - y_q x_p$$

Overview

- Ray tracing overview
- Ray trees
- Intersections
  - Spheres
  - Planes
  - Polygons
- **Bounding volumes**
  - Bounding volume hierarchies

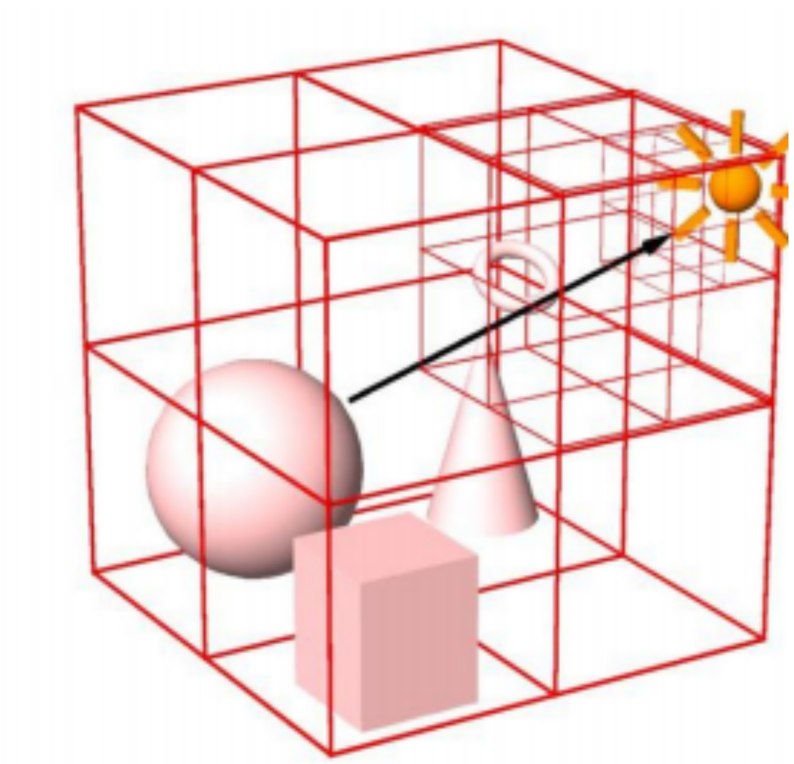
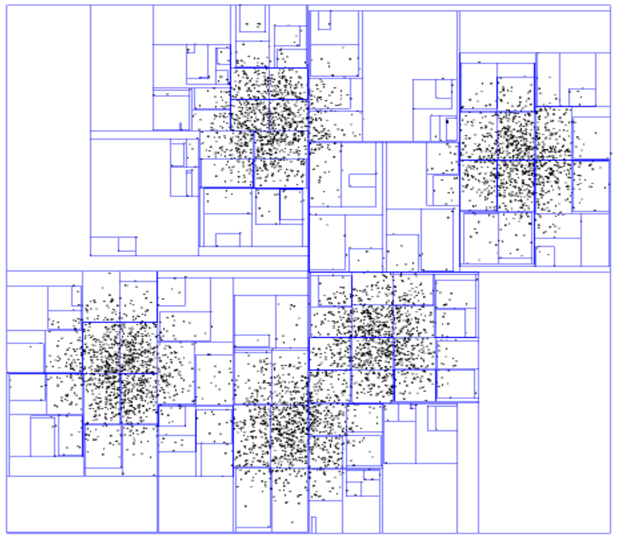
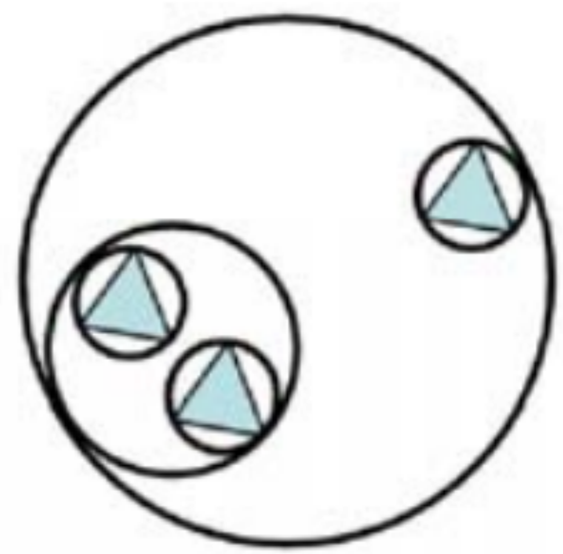
Bounding volumes

- We want to reduce number of ray-object intersections to test
- Use bounding volumes:
  - Test for an intersection with bounding volume
  - Only test intersection with objects inside volume if we intersect the bounding volume
- Boxes, spheres



Hierarchical structures

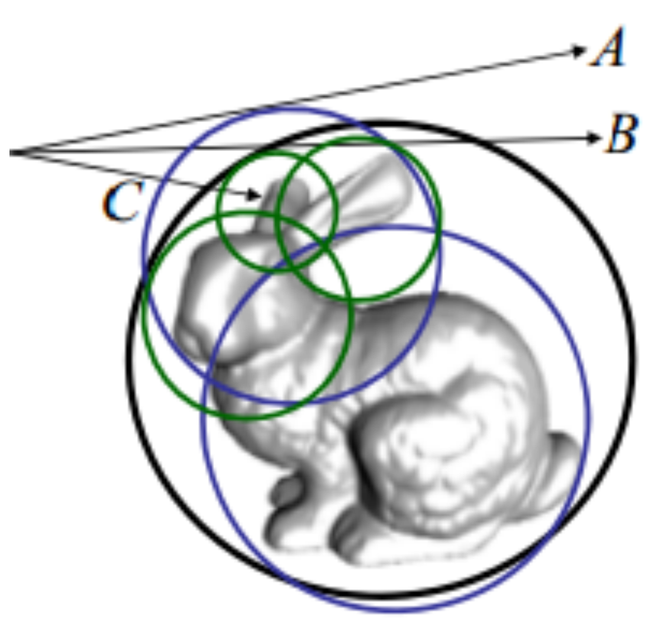
- Enclose objects in hierarchical bounding volumes
- Octrees, KD-trees
- Bounding volume hierarchies

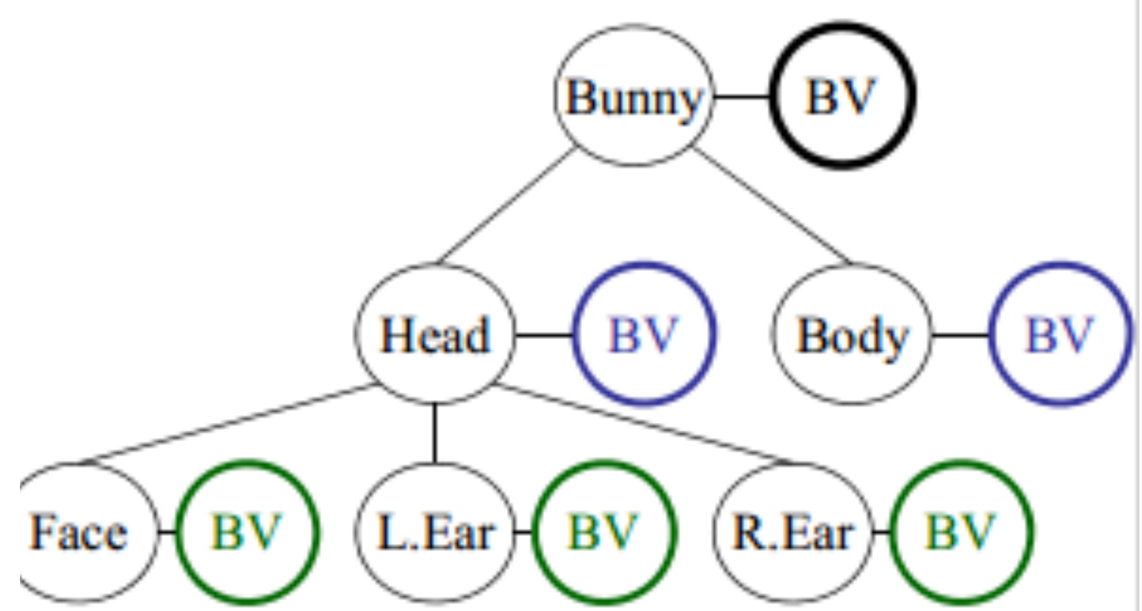




*

Bounding volume hierarchy

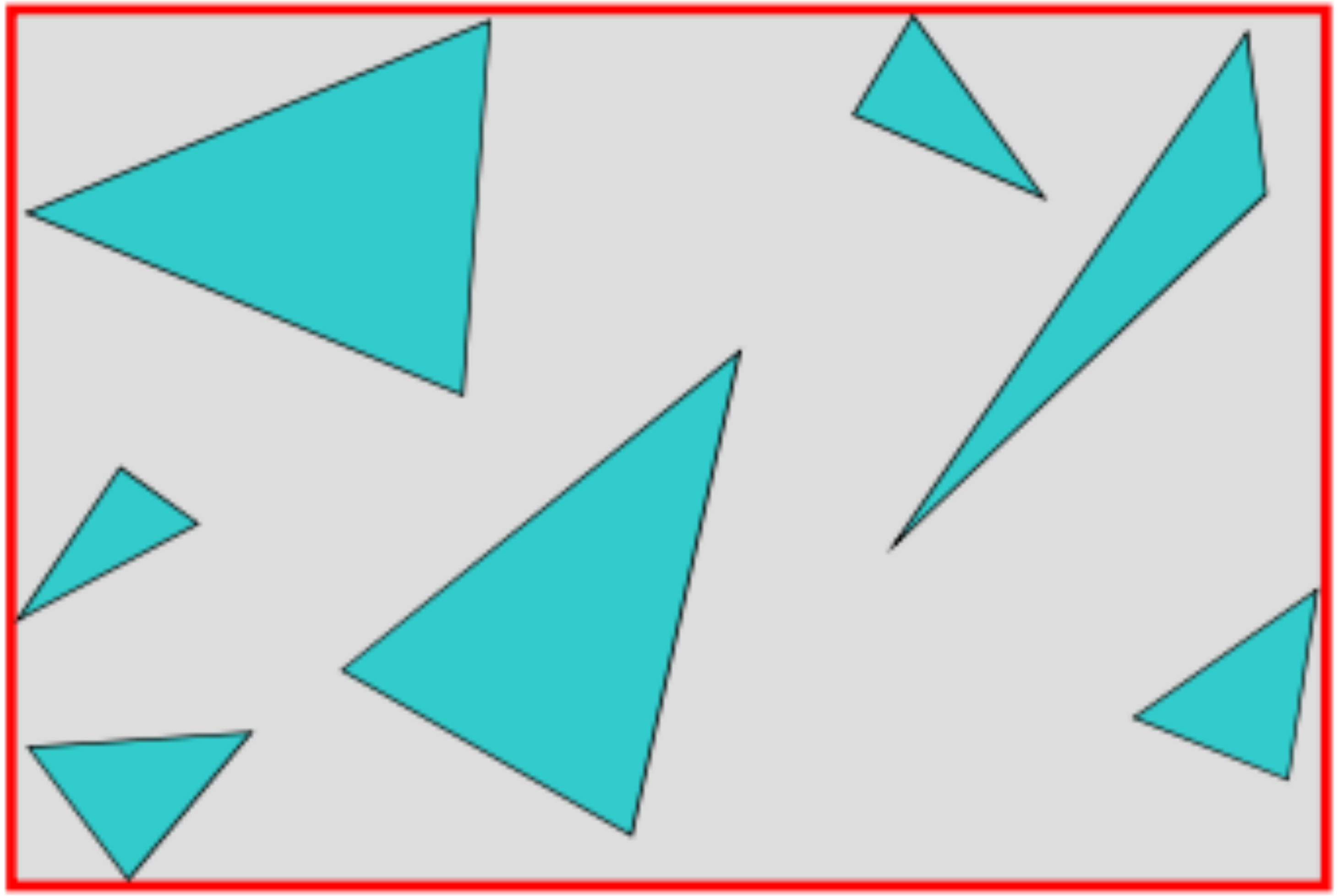
- Give each object a bounding volume
- The bounding volume does not partition
- The bounding volumes can overlap each other
- The volume higher in the hierarchy contains their children
- If a ray misses a bounding volume, no need to check for intersection with children
- If we intersect a bounding volume, check intersection with children





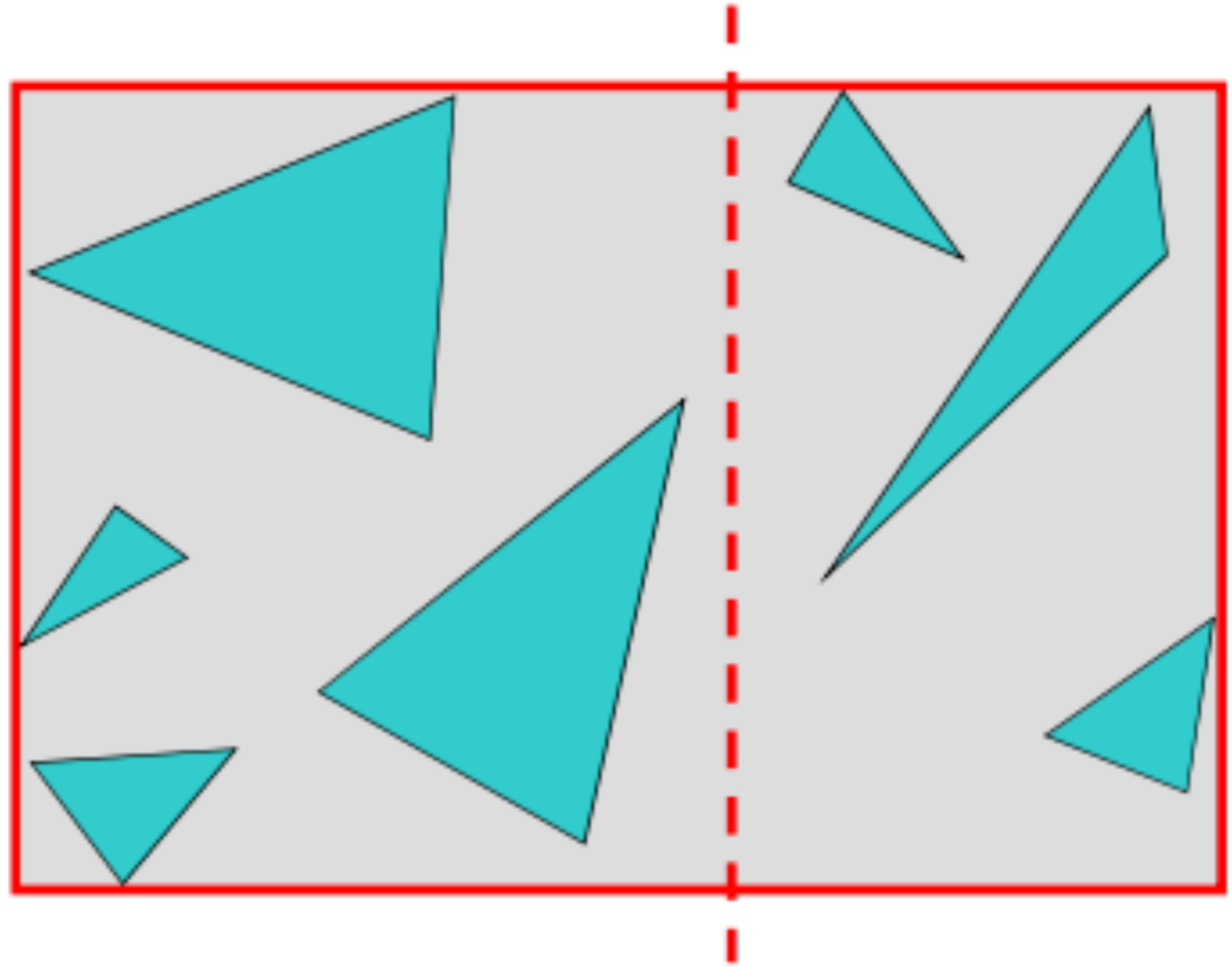
Producing the hierarchy

- Find bounding box of objects



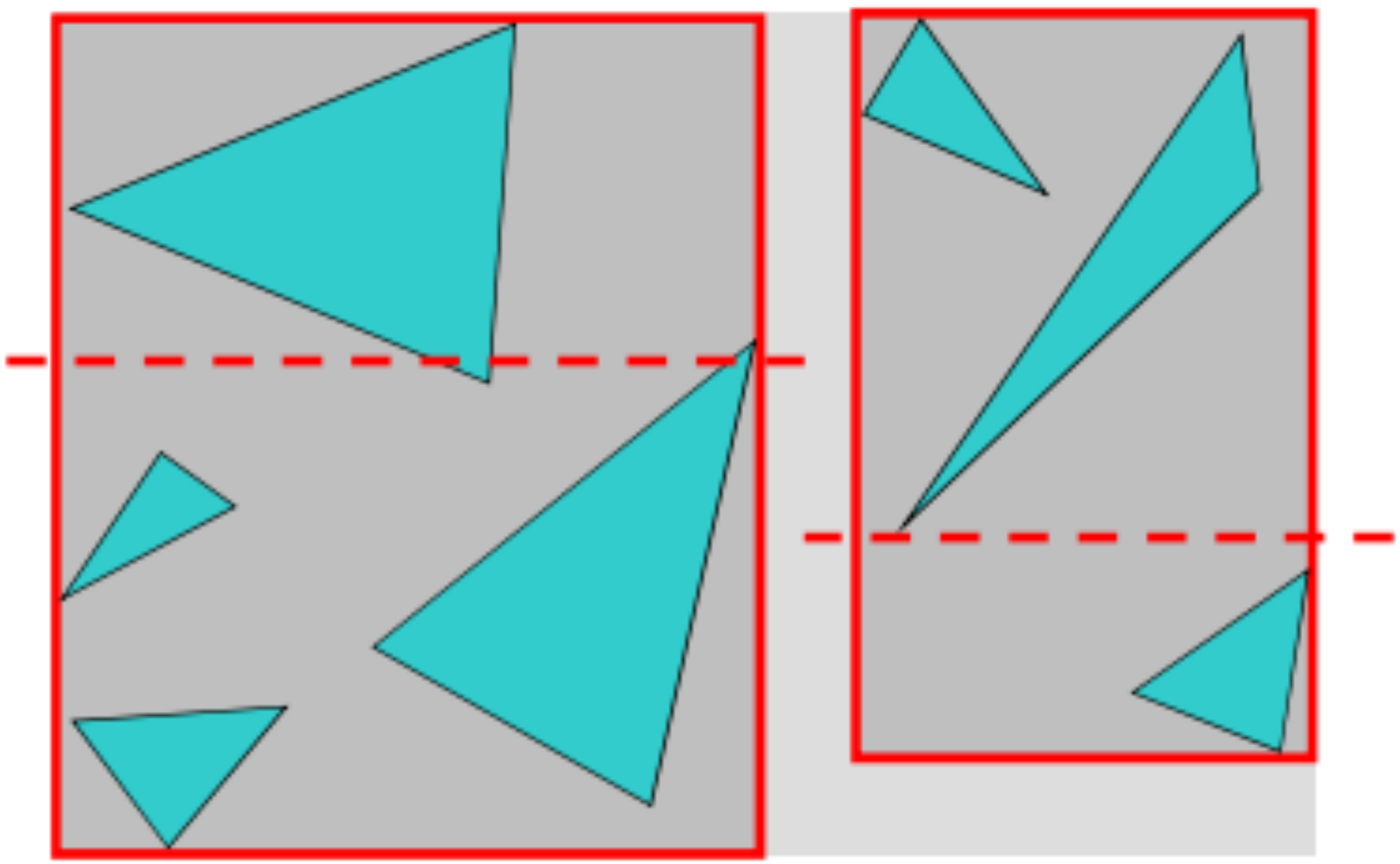
Producing the hierarchy

- Find bounding box of objects
- Split into two groups



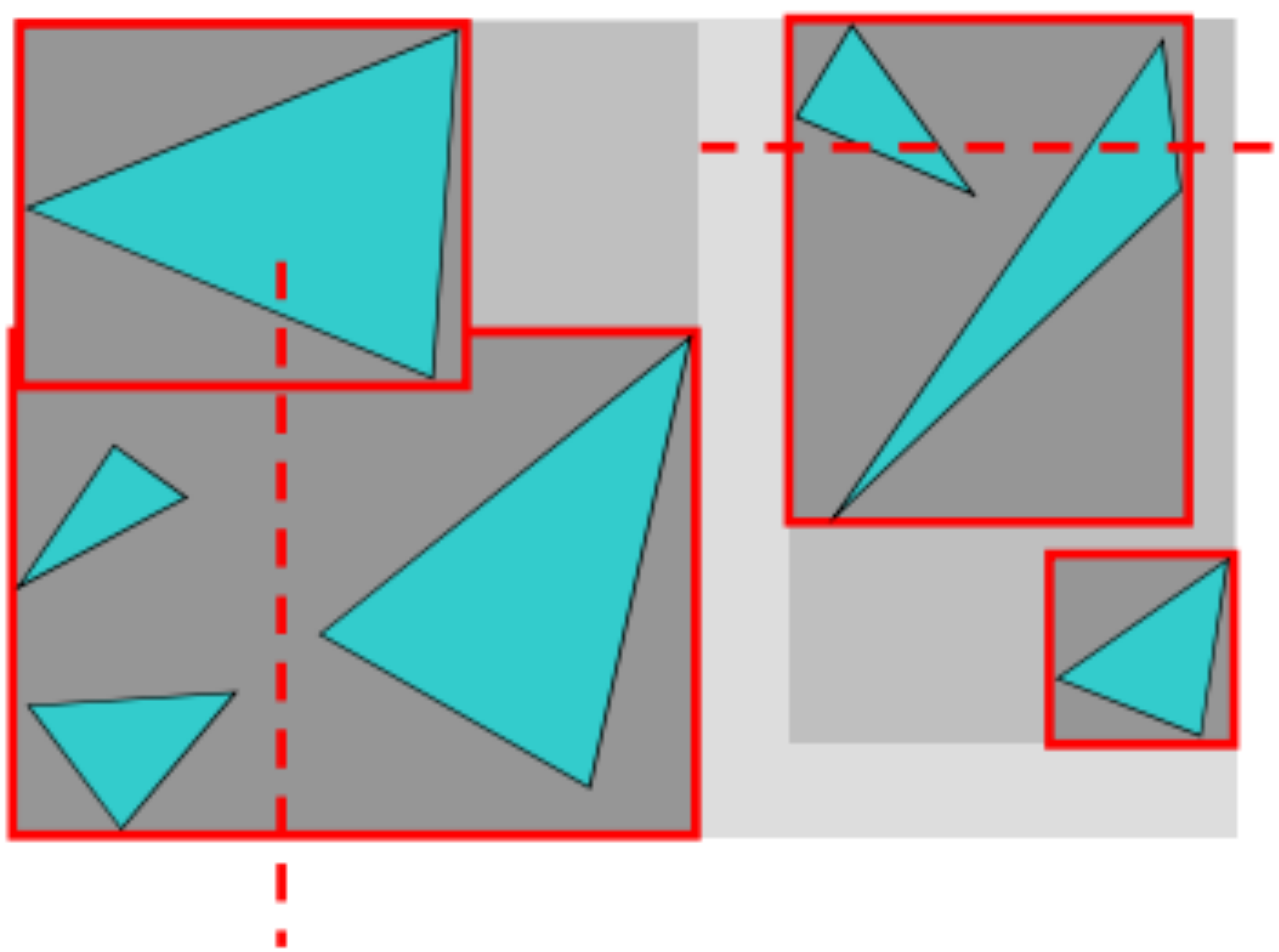
Producing the hierarchy

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



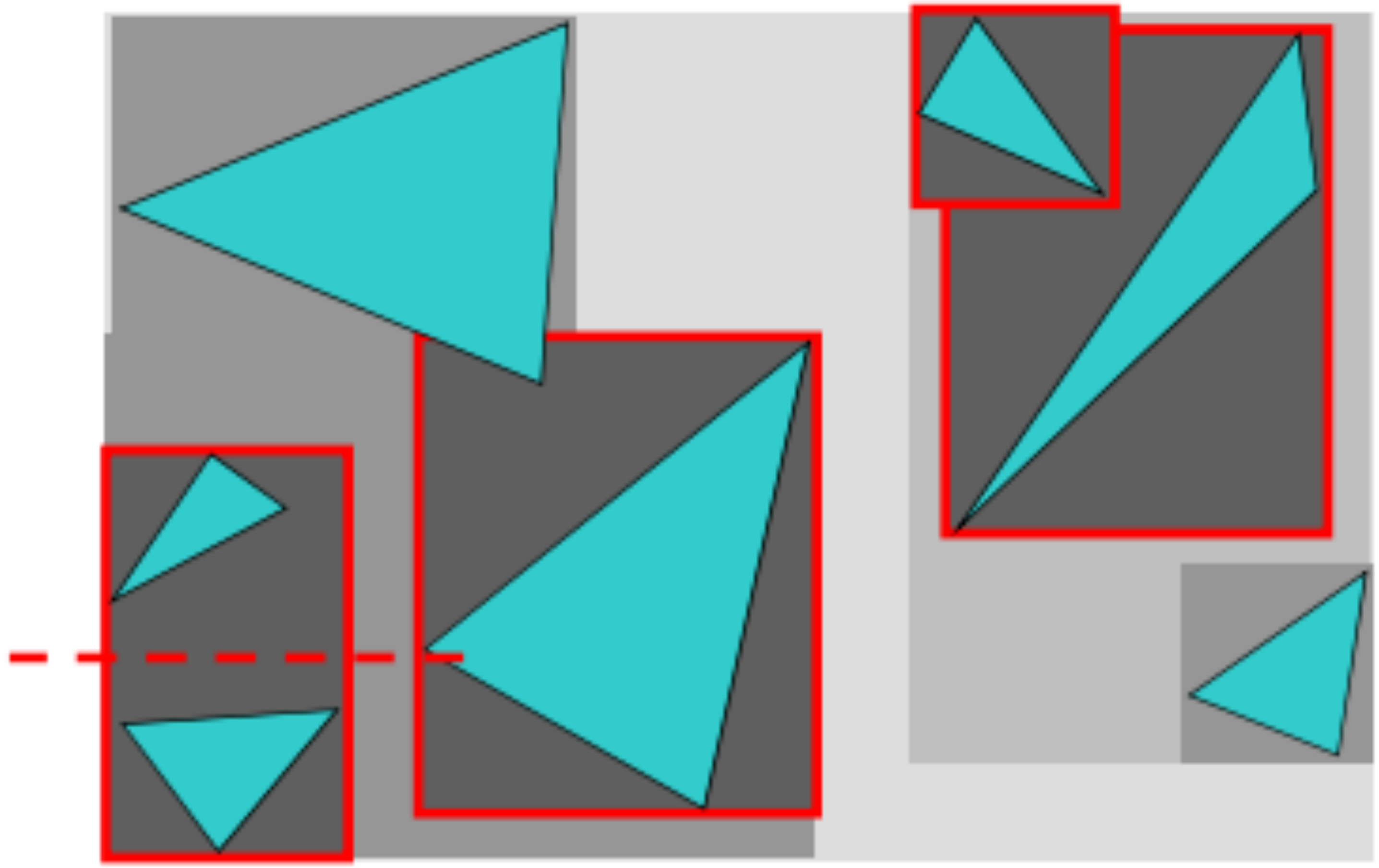
Producing the hierarchy

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



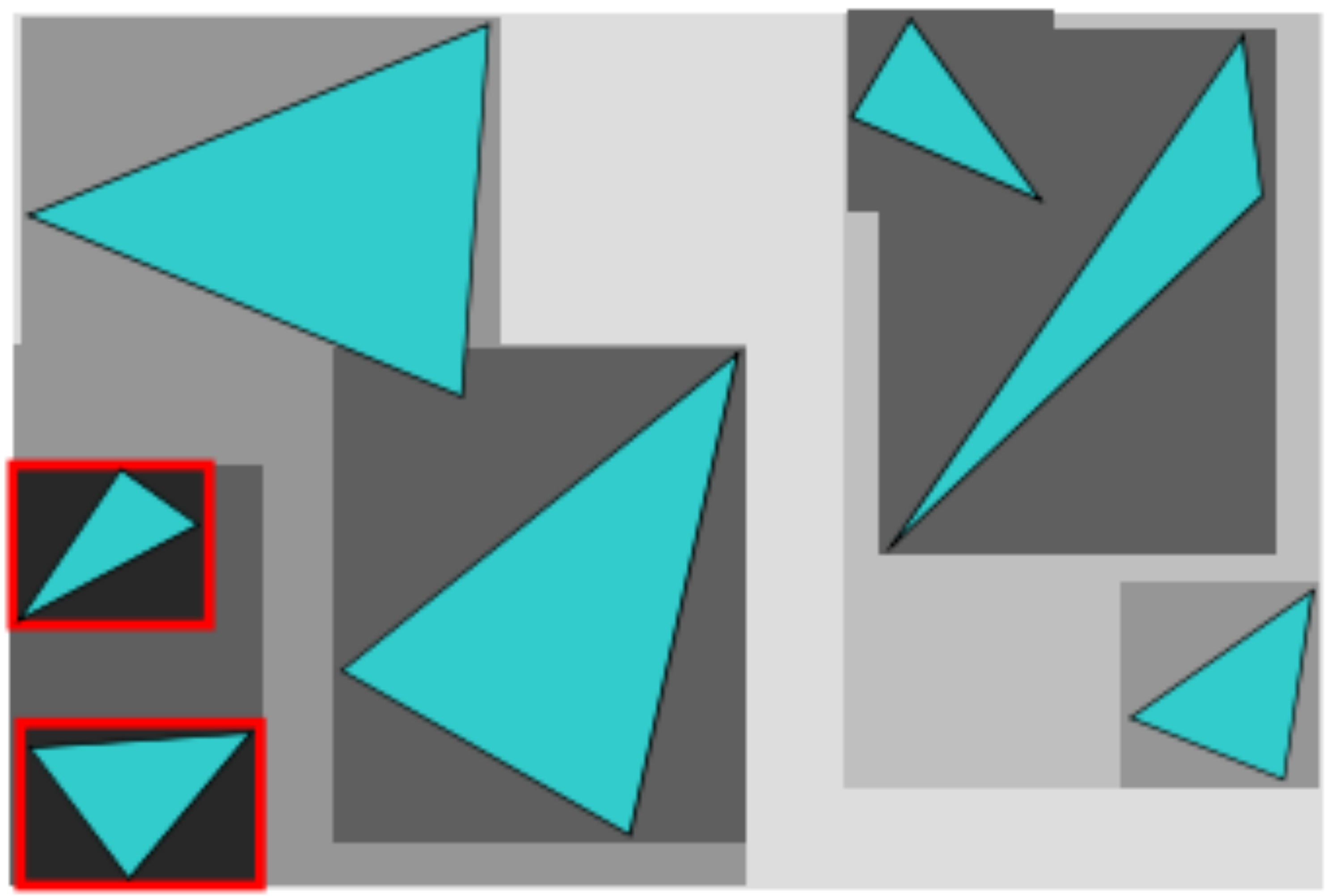
Producing the hierarchy

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



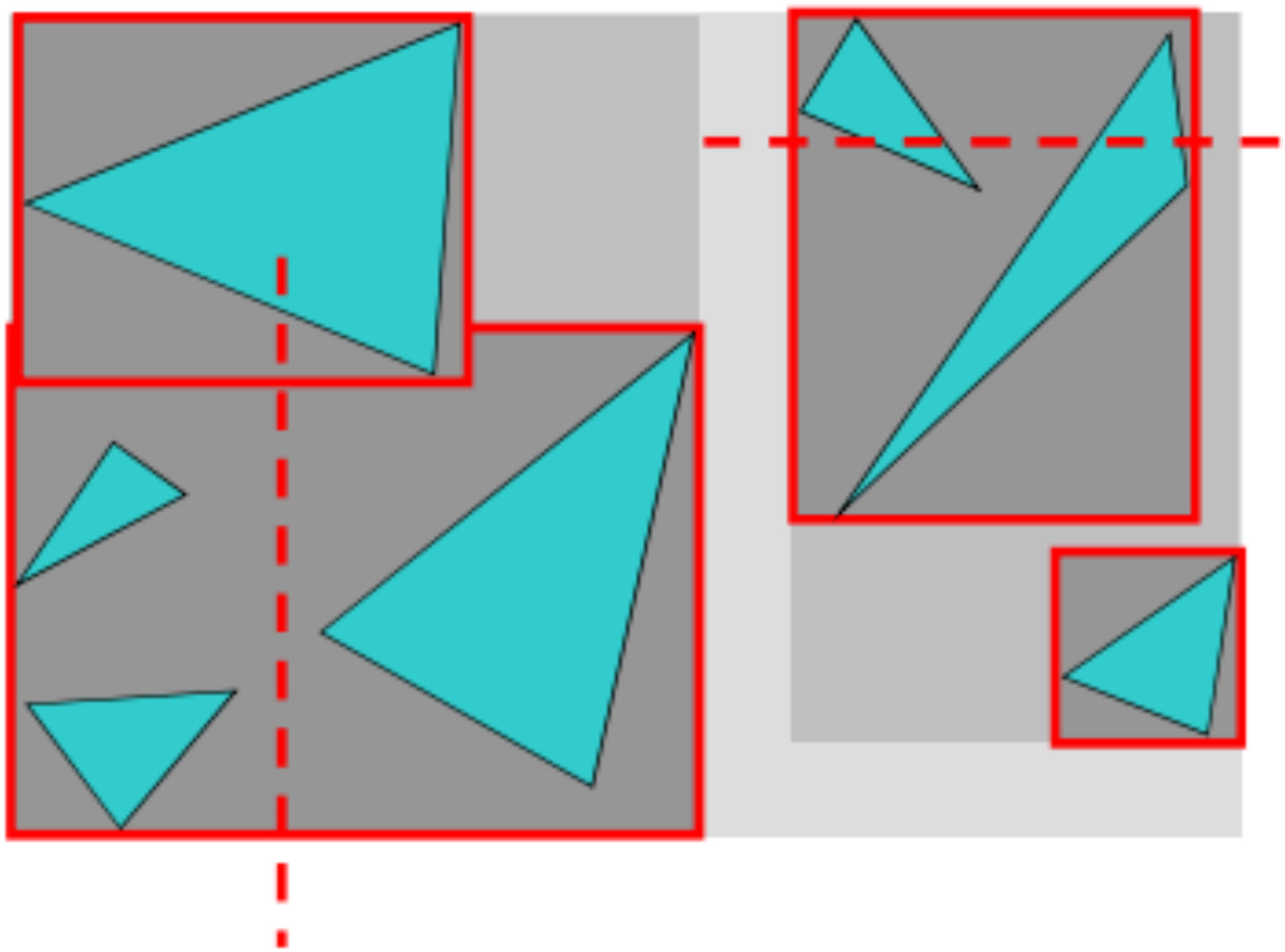
Producing the hierarchy

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



Where to split?

- At midpoint
- Sort and put half on each side



*

Computing intersections

```
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

- Simple but computationally expensive
- Easily includes reflection, refraction and shadows
- Calculating intersections is main bottleneck
- Reduce the number of intersection calculations using a bounding volume hierarchy

References

- Shirley Chapter 4 (Ray tracing)
- Shirley Chapter 12.3 (12.3.1,12.3.2) (Spatial Data Structures)
- Foley Chapter 15.10 (Visible-surface ray tracing), 16.11,16.12 (Global illumination, Recursive ray tracing)
- Akenine-Möller Chapter 16.6, 16.8 (Ray/Sphere intersection, Ray/Triangle intersection)