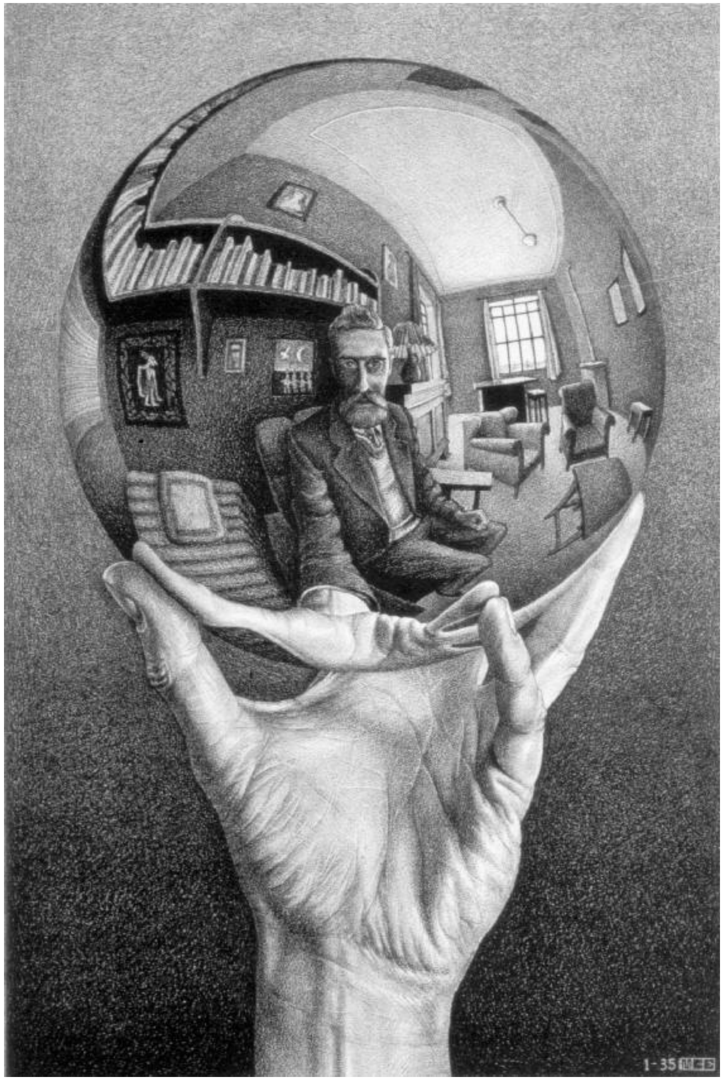
Ray Tracing

Foley & Van Dam, Chapters 15 and 16

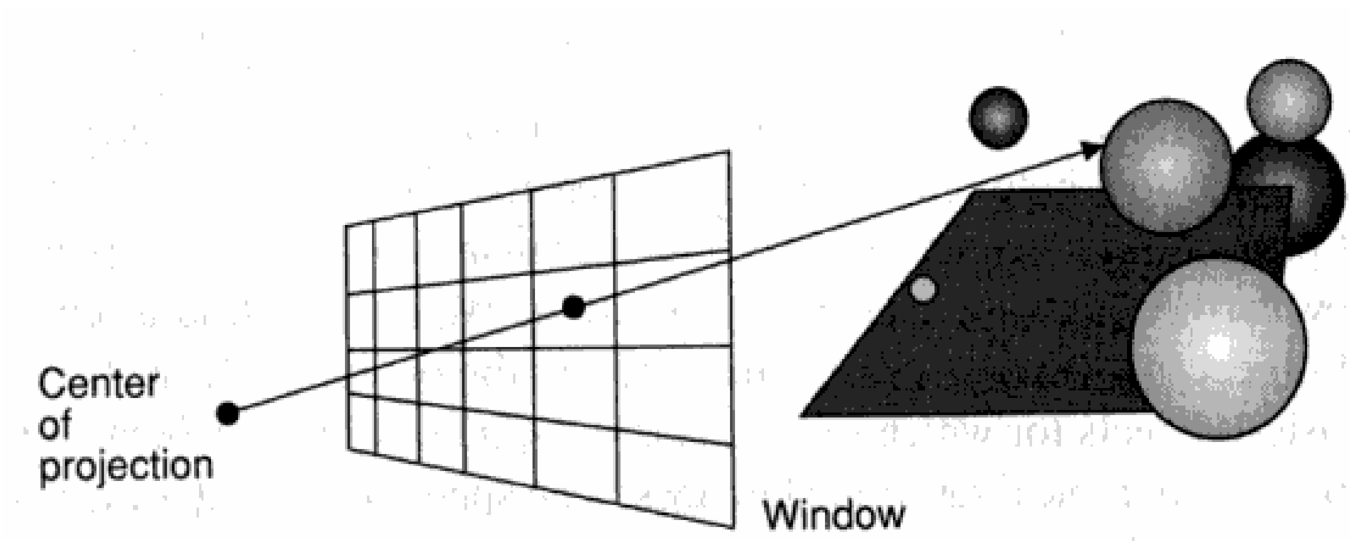


Ray Tracing

- Visible Surface Ray Tracing (Ray Casting)
- Examples
- Efficiency Issues
- Computing Boolean Set Operations
- Recursive Ray Tracing

Visible Surface Ray Tracing

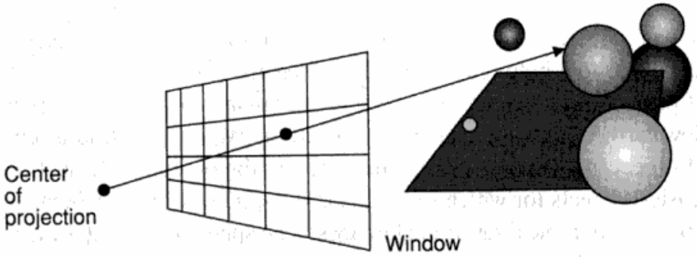
- Determine visibility of a surface by tracing rays of light from the viewer's eye to the objects in the scene
- Image precision algorithm
- VSRT is sometimes called "Ray Casting"



Visible Surface Ray Tracing

- Simple Ray Tracer:

```
select center of projection and window on viewplane;
for (each scan line in image) {
    for (each pixel in scan line) {
        determine ray from center of projection through pixel;
        for (each object in scene) {
            if (object is intersected and is closest considered thus far)
                record intersection and object name;
        }
        set pixel's color to that at closest object intersection;
    }
}
```



Visible Surface Ray Tracing

- **Problem**: computing intersections between the ray and the objects

- Each point (x,y,x) of the ray from (x_0,y_0,z_0) to (x_1,y_1,z_1) is defined by:

$$x = x_0 + t\Delta x, \quad y = y_0 + t\Delta y, \quad z = z_0 + t\Delta z$$

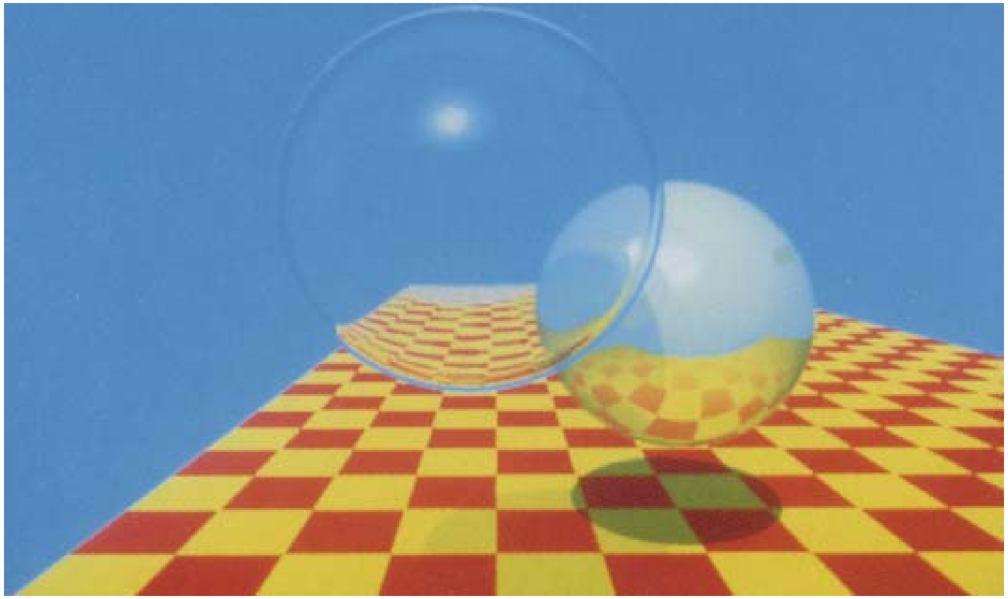
where:

$$\Delta x = x_1 - x_0, \quad \Delta y = y_1 - y_0, \quad \Delta z = z_1 - z_0$$

Visible Surface Ray Tracing

- **Simplest case:**
  Intersection with a sphere of center (a,b,c) and radius r

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$$



Visible Surface Ray Tracing

- **Example:**

Ray: $x = x_0 + t\Delta x, \quad y = y_0 + t\Delta y, \quad z = z_0 + t\Delta z$

Sphere: $(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$

$$x^2 - 2ax + a^2 + y^2 - 2by + b^2 + z^2 - 2cz + c^2 = r^2,$$

$$(x_0 + t\Delta x)^2 - 2a(x_0 + t\Delta x) + a^2 + (y_0 + t\Delta y)^2 - 2b(y_0 + t\Delta y) + b^2$$
$$+ (z_0 + t\Delta z)^2 - 2c(z_0 + t\Delta z) + c^2 = r^2,$$

$$x_0^2 + 2x_0\Delta xt + \Delta x^2t^2 - 2ax_0 - 2a\Delta xt + a^2$$
$$+ y_0^2 + 2y_0\,\Delta yt + \Delta y^2t^2 - 2by_0 - 2b\Delta yt + b^2$$
$$+ z_0^2 + 2z_0\,\Delta zt + \Delta z^2t^2 - 2cz_0 - 2c\Delta zt + c^2 = r^2.$$

Visible Surface Ray Tracing

- **Example:**

  Collecting terms:

$$(\Delta x^2 + \Delta y^2 + \Delta z^2)t^2 + 2t[\Delta x(x_0 - a) + \Delta y(y_0 - b) + \Delta z(z_0 - c)]$$
$$+ (x_0^2 - 2ax_0 + a^2 + y_0^2 - 2by_0 + b^2 + z_0^2 - 2cz_0 + c^2) - r^2 = 0,$$
$$(\Delta x^2 + \Delta y^2 + \Delta z^2)t^2 + 2t[\Delta x\,(x_0 - a) + \Delta y(y_0 - b) + \Delta z(z_0 - c)]$$
$$+ (x_0 - a)^2 + (y_0 - b)^2 + (z_0 - c)^2 - r^2 = 0.$$

  Quadratic equation in t $\quad At^2 + Bt + C = 0$

  Has zero, one or two real roots

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Visible Surface Ray Tracing

- **Example:**
  Zero real roots: no intersections
  One root: ray grazes the sphere
  Two roots: smallest positive t is the closest

At the intersection point, the surface has normal:

$$\frac{x-a}{r}, \quad \frac{y-b}{r}, \quad \frac{z-c}{r}$$

Visible Surface Ray Tracing

- Similar approach with other quadratic surfaces

- Intersection between a ray and a polygon is harder to find:
  - Find the intersection between the ray and the polygon's plane;
  - Check whether the intersection lies within the polygon

Visible Surface Ray Tracing

- Similar approach for other quadratic surfaces:
  - Sphere: $x2 + y2 + z2 - r2$
  - Cylinder: $x2 + y2 - r2$
  - Cone: $x2 + y2 - z2$
  - Paraboloid: $x2 + y2 - z$
  - Hyperboloid: $x2 + y2 - z2 \pm r2$
- Roots of equations of degree higher than 2 can be found with an iterative method like Newton

Visible Surface Ray Tracing

- Intersection between a ray and a polygon is harder to find:

  - Find the intersection between the ray and the polygon's plane;

  - Check whether the intersection lies within the polygon

Visible Surface Ray Tracing

- **Example:**

Ray: $x = x_0 + t\Delta x, \quad y = y_0 + t\Delta y, \quad z = z_0 + t\Delta z$

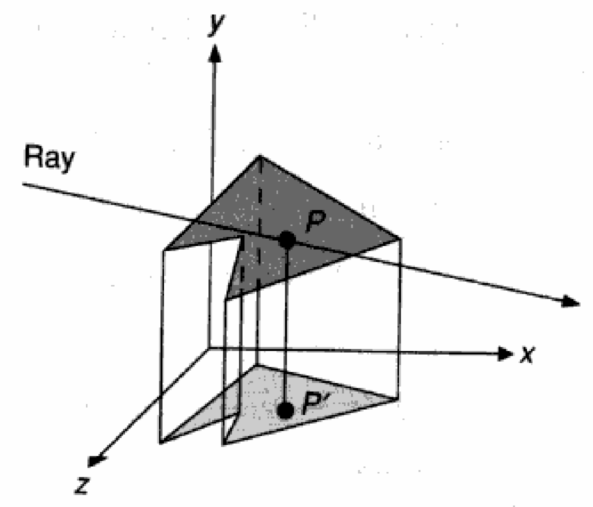
Plane: $Ax + By + Cz + D = 0$

$$A(x_0 + t\Delta x) + B(y_0 + t\Delta y) + C(z_0 + t\Delta z) + D = 0,$$

$$t(A\Delta x + B\Delta y + C\Delta z) + (Ax_0 + By_0 + Cz_0 + D) = 0,$$

$$t = -\frac{(Ax_0 + By_0 + Cz_0 + D)}{(A\Delta x + B\Delta y + C\Delta z)}.$$

If the denominator is 0, the ray and the plane are parallel



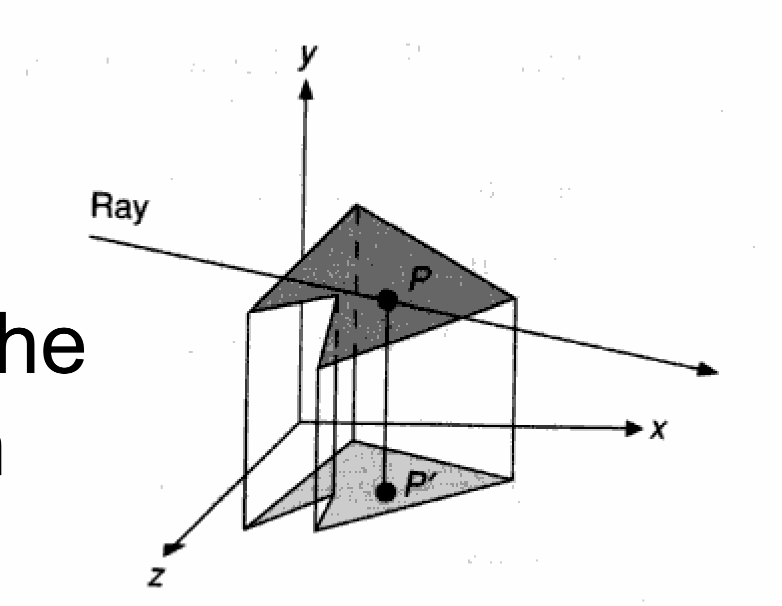
Visible Surface Ray Tracing

- **Example:**

$$t = -\frac{Ax_0 + By_0 + Cz_0 + D}{A\Delta x + B\Delta y + C\Delta z}$$

If the denominator is 0, the ray and the plane are parallel

Testing whether the intersection lies within the polygon is done with an orthographic projection and 2D tests



Visible Surface Ray Tracing

Efficiency considerations:

- Ray tracing is slow because it intersects every ray with every object
- To make ray tracing faster we can use coherence:

Image coherence - neighboring pixel,

same object,

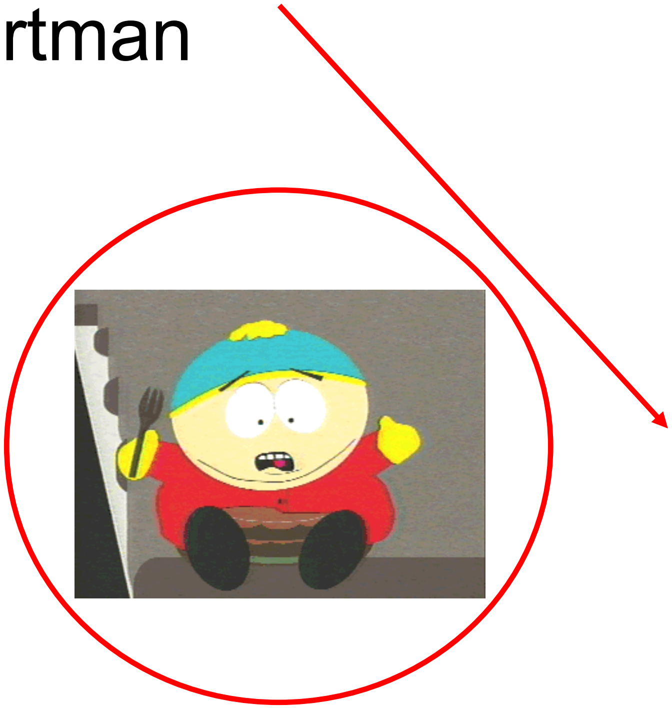
...

Spatial coherence - neighboring points,

Visible Surface Ray Tracing

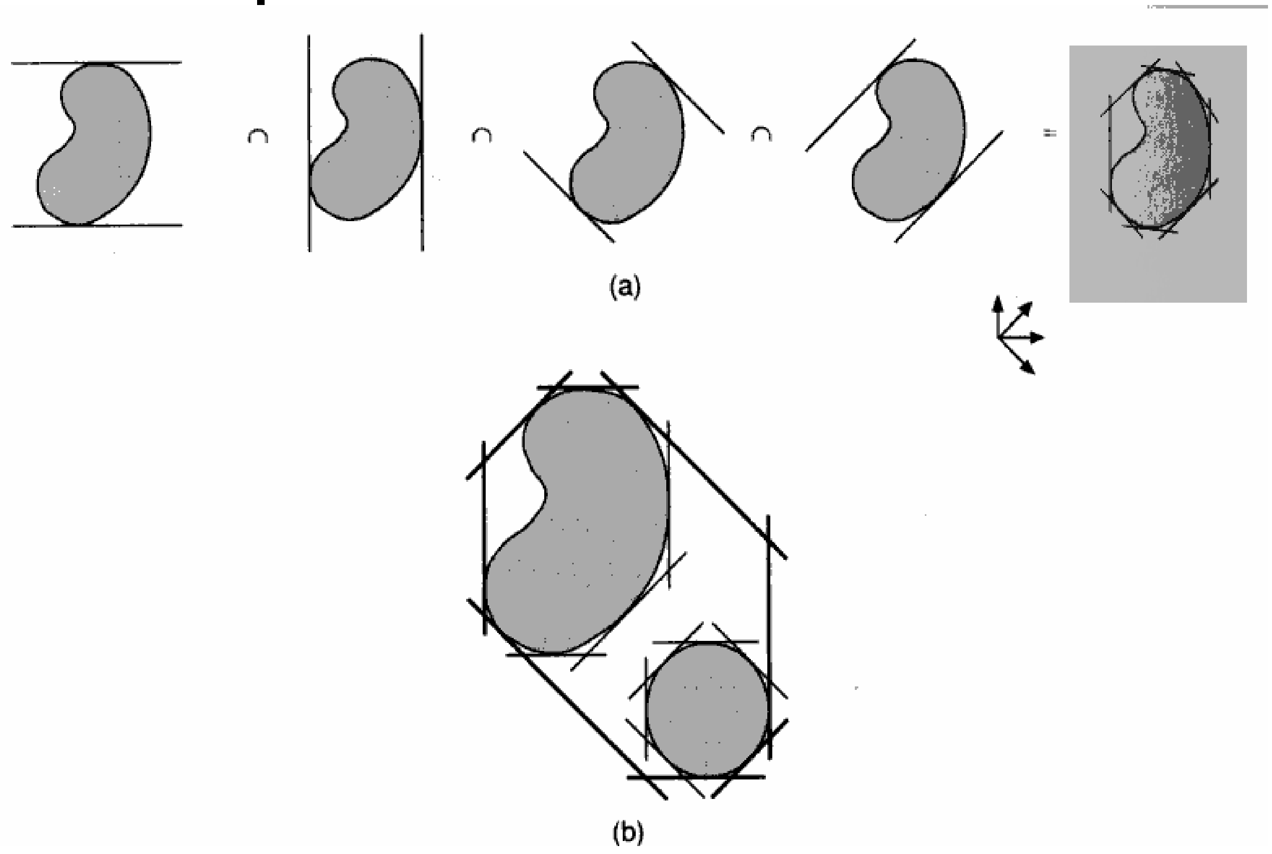
- **Problem:**
  - Cartman is composed by 100000 polygons
  - Ray tracing computes 100000 ray-polygon intersections
  - Even when the ray misses Cartman

- **Solution:**
  - Place a sphere around Cartman
  - If ray misses sphere then the ray misses Cartman



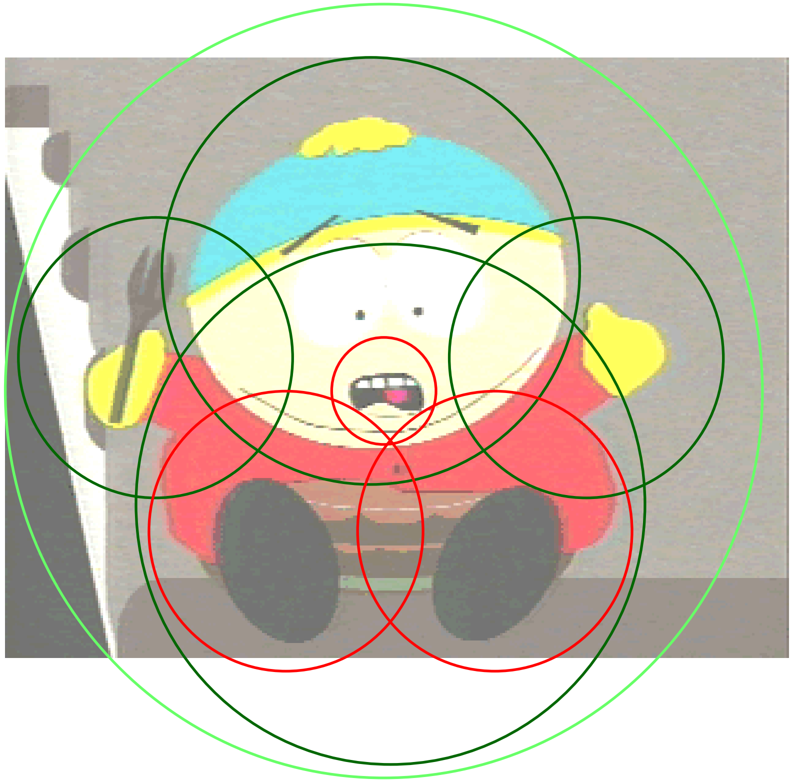
Visible Surface Ray Tracing

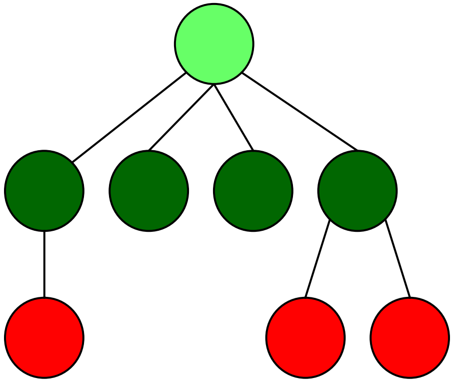
- **Efficiency considerations**: optimizing intersection calculations by bounding objects with parametrized slabs



Visible Surface Ray Tracing

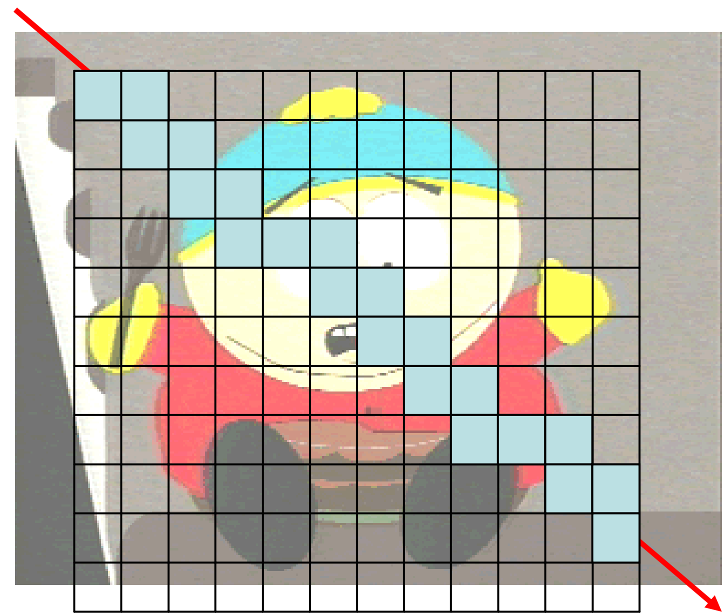
- **Efficiency considerations**: avoiding intersection calculations with bounding volumes hierarchies



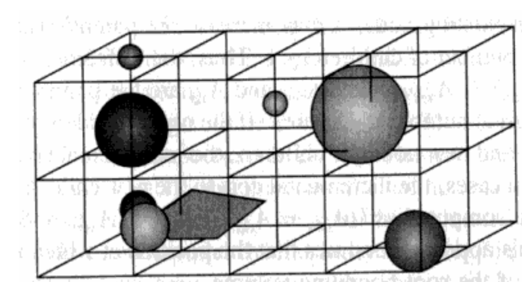


Visible Surface Ray Tracing

- **Efficiency considerations**: avoiding intersection calculation with spatial partitioning

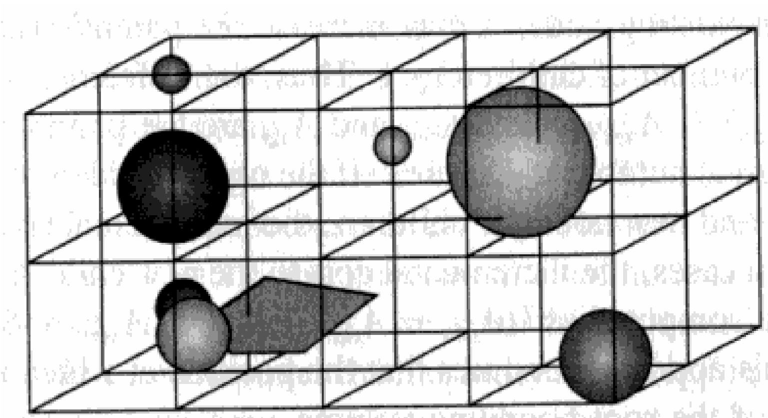


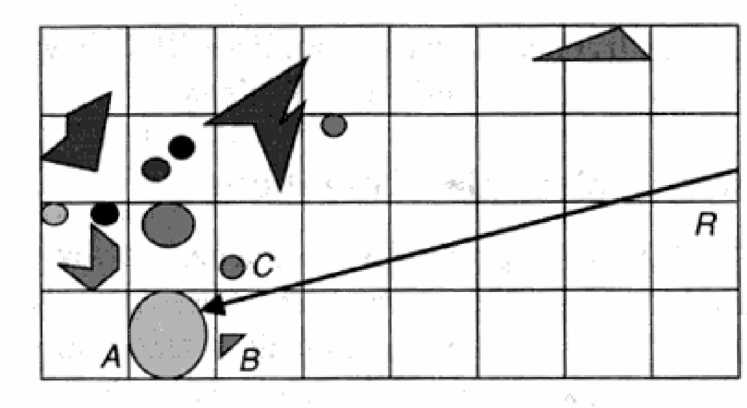
- 3-D array of cells
- Each cell contains list of all objects it intersects
- Ray intersected with all objects in a given cell's list
- Cells visited in Bresenham order



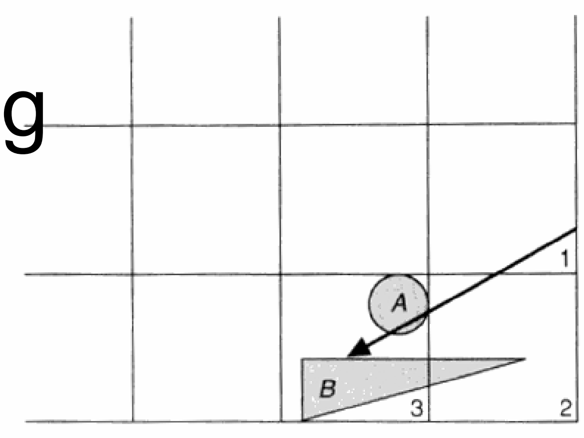
Visible Surface Ray Tracing

- **Efficiency considerations**: avoiding intersection calculation with spatial partitioning



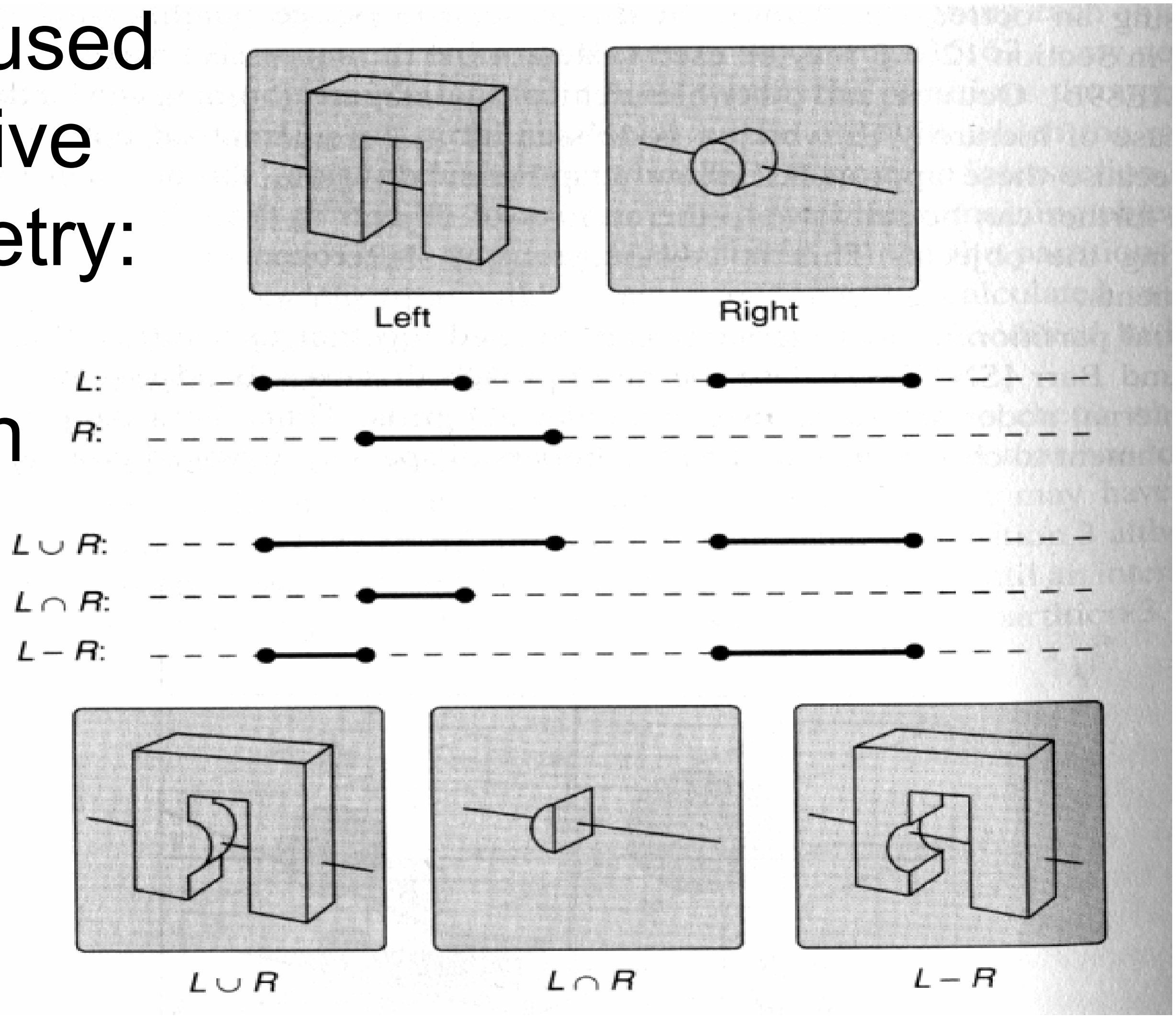


- Must be careful when excluding objects, since an object may intersect in a different voxel than the current one



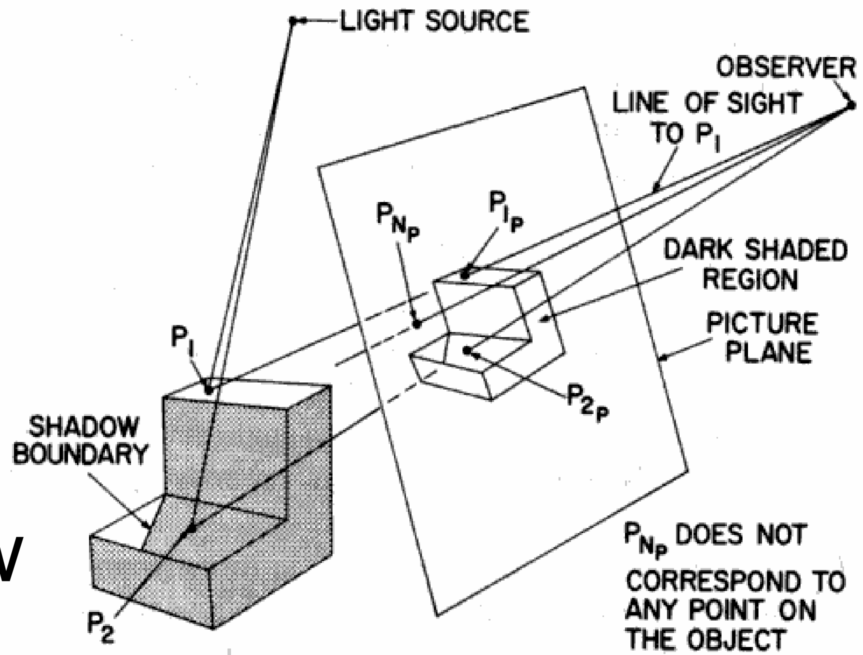
Visible Surface Ray Tracing

- Useful to compute Boolean Set Operations that can be used in Constructive Solid Geometry:
  - Union
  - Intersection
  - Difference



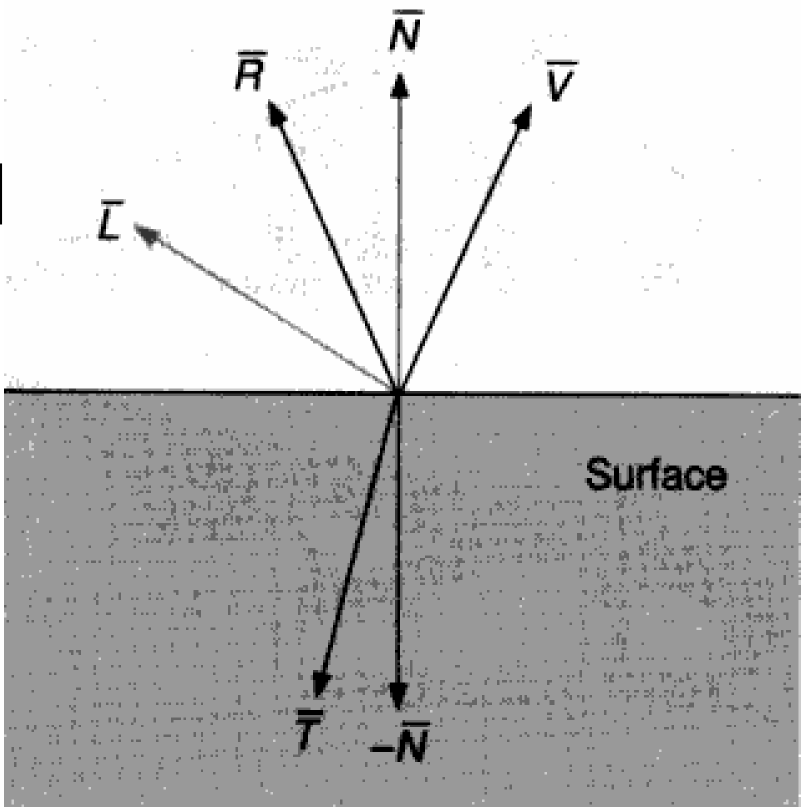
Recursive Ray Tracing

- Extends the basic Ray Tracing algorithm to handle shadows, reflection and refraction
- **Shadows:** fire an additional ray from the intersection to the light source.
  If this *shadow ray* intersects any object along the way, then the point is in shadow



Ray Tracing

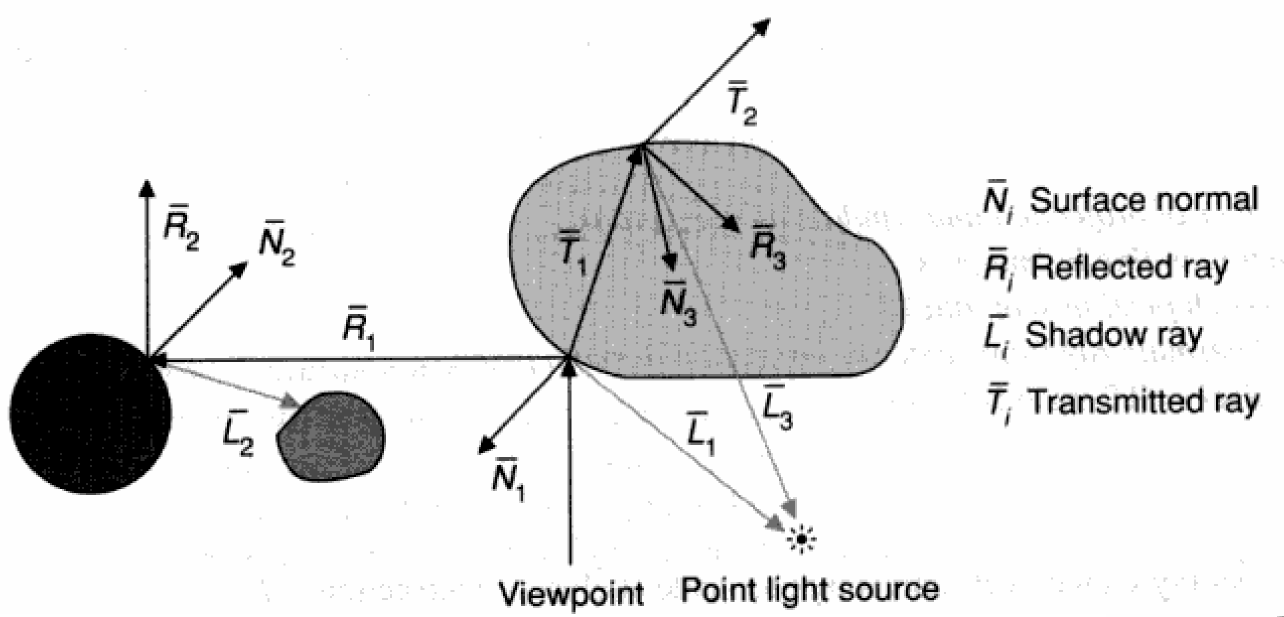
- **Reflection and Refraction:** secondary reflection and refraction rays are fired at intersections. In turn, these rays may spawn shadow, reflection and refraction rays

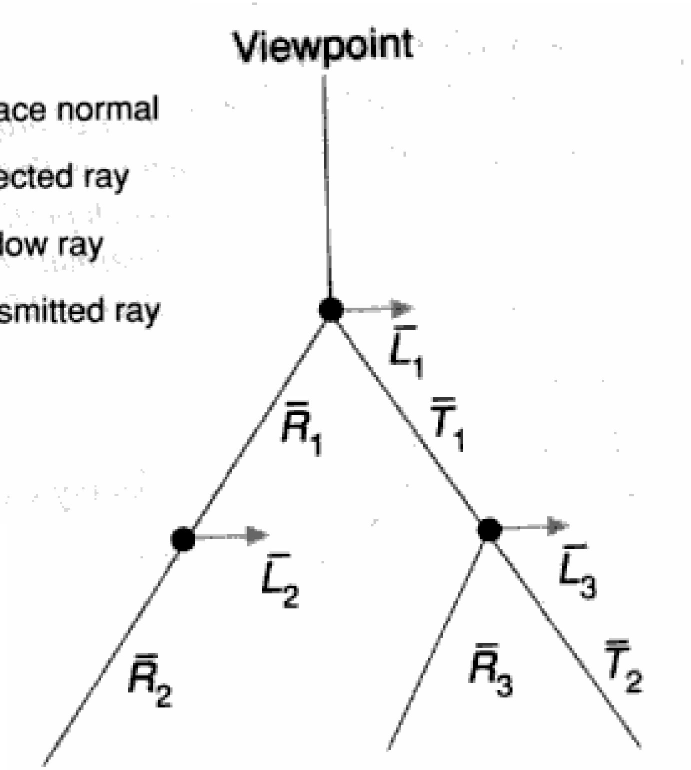


- **N:** Surface normal
- **L:** Shadow ray
- **R:** Reflected ray
- **T:** Transmitted ray

Ray Tracing

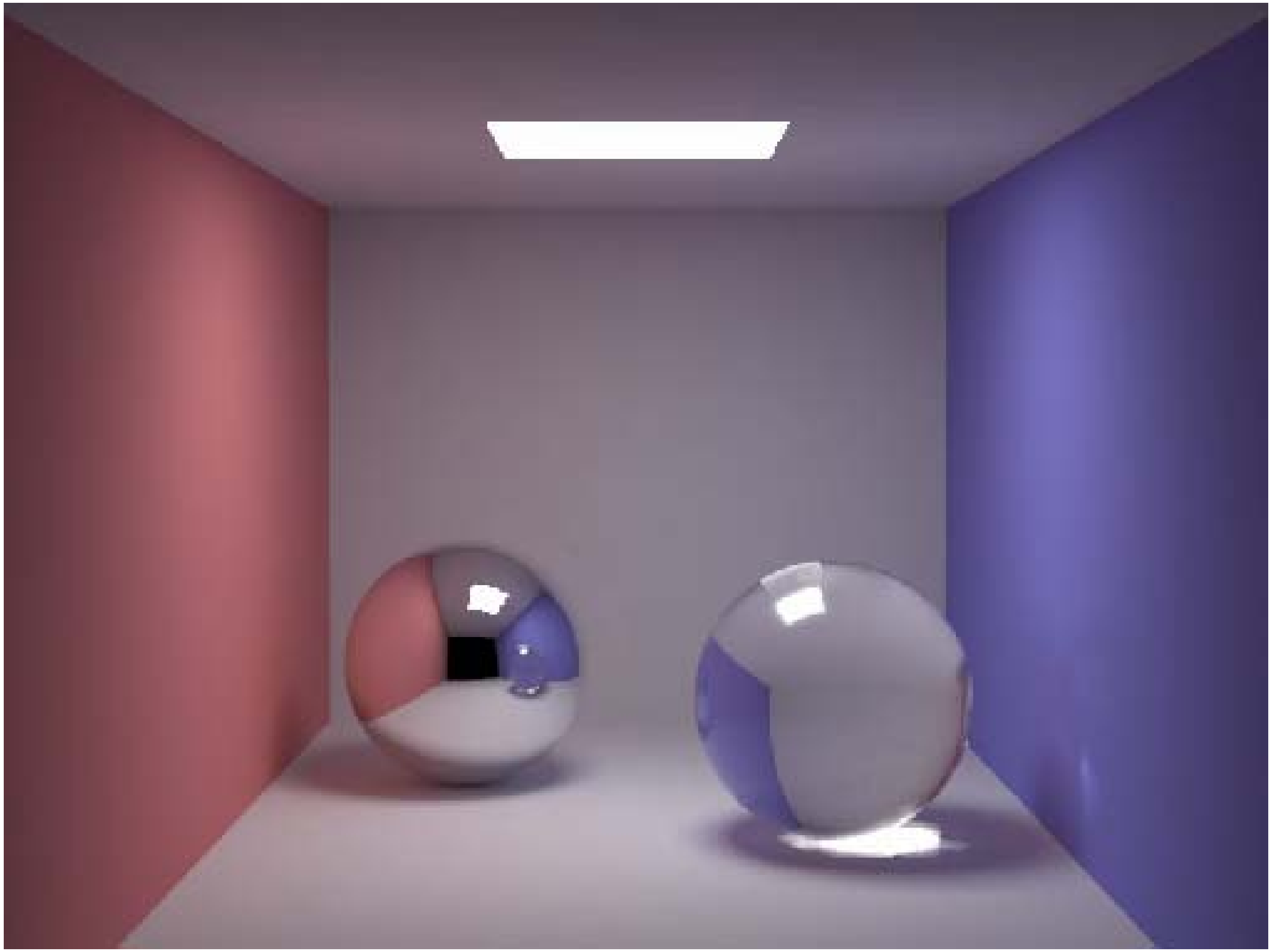
- The rays form a **ray tree**





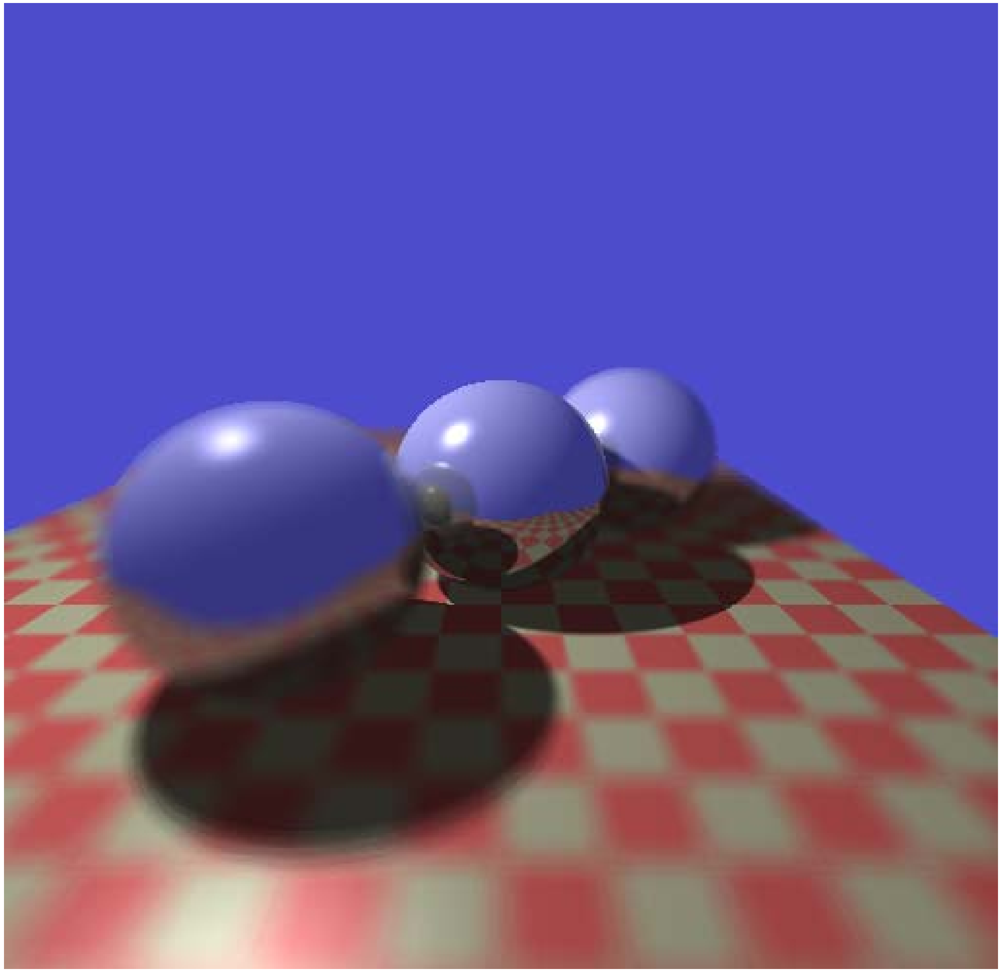
Recursive Ray Tracing

- **Example:**



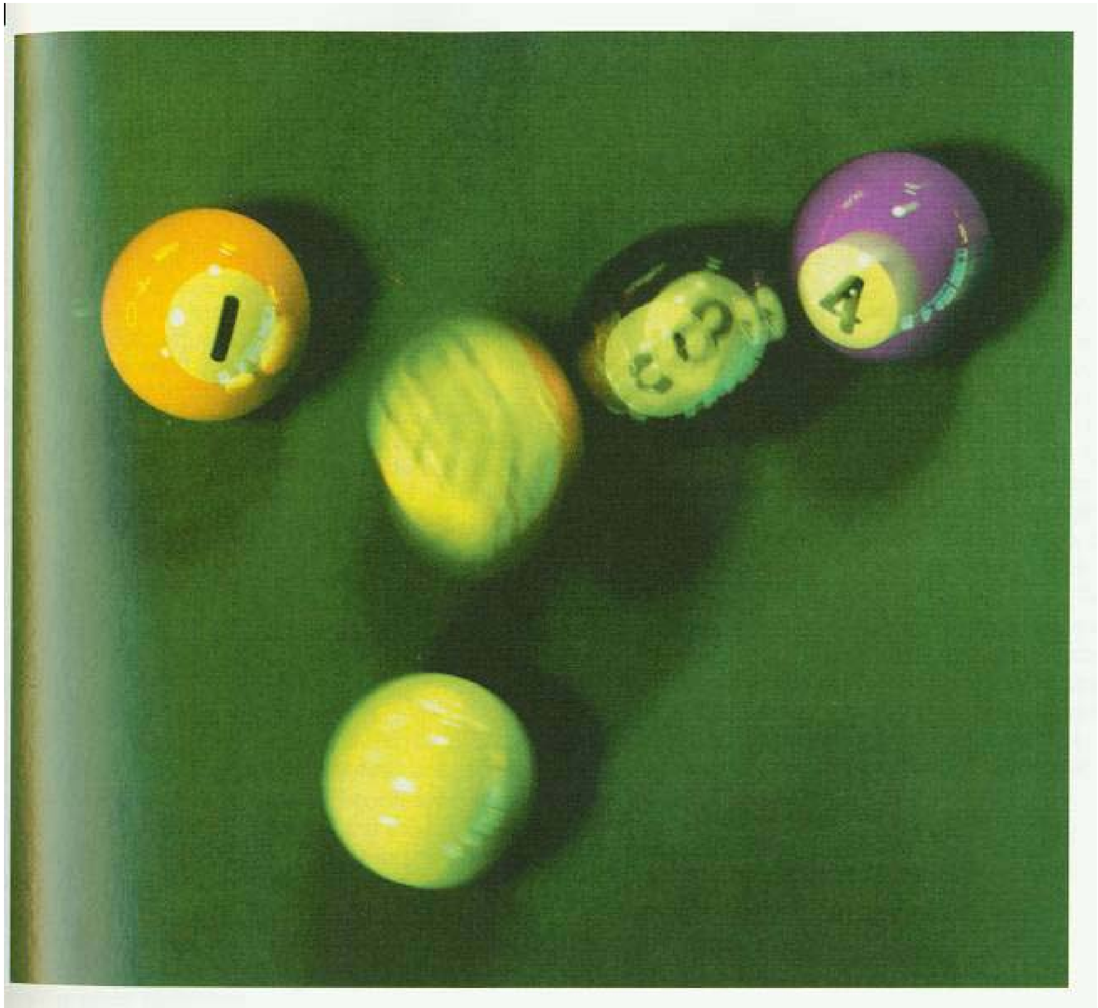
Recursive Ray Tracing

- **Example:** Depth of Field with a camera model



Recursive Ray Tracing

- **Example:** Motion Blur



Ray Tracing

- **Software**

Ray tracer:
Persistence Of Vision POV-Ray
http://www.povray.org/

Wireframe modeller for POV-Ray:
MoRay
http://www.stmuc.com/moray/