Ray Tracing

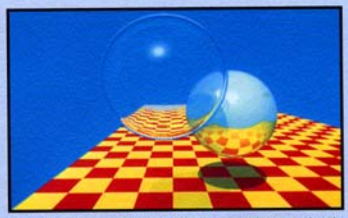


Plate III.10 Spheres and checkerboard. An early image produced with recursive ray tracing (Section 16.12). (Courtesy of Turner Whitted, Bell Laboratories.)

Ray Tracing

- Ray Tracing kills two birds with one stone:
  - Solves the Hidden Surface Removal problem
  - Evaluates an improved global illumination model
    - shadows
    - ideal specular reflections
    - ideal specular refractions
  - Enables direct rendering of a large variety of geometric primitives
- Book: A. Glassner, An Introduction to Ray Tracing

Backward Tracing

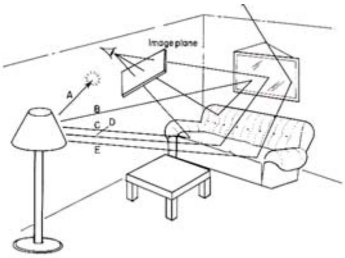


Fig. 5. Some light rays (like A and E) never reach the image plane at all. Others follow simple or complicated routes.

Reflected, Transmitted and Shadow rays

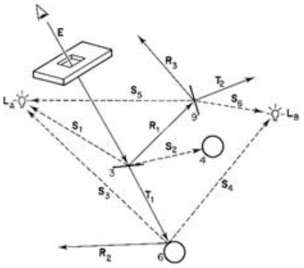


Fig. 11. An eye ray **E** propagated through a scene. Many of the intersections spawn reflected, transmitted, and shadow rays.

The Illumination Model

- Remember the local illumination model we saw earlier?

$$I_r = I_a k_a + \sum_{i=1}^{\ell} f_{att_i} I_{p_i} \left[k_d (N \cdot L_i) + k_s (R_i \cdot V)^n \right]$$

- First, let's add shadows into the model:

$$I_r = I_a k_a + \sum_{i=1}^{\ell} S_i f_{att_i} I_{p_i} \left[k_d (N \cdot L_i) + k_s (R_i \cdot V)^n \right]$$

Illumination Model (cont'd)

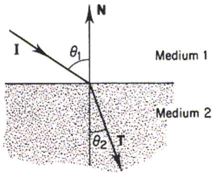
- Add in light arriving from the mirror-reflected direction $k_s I_s$
- Add in light arriving from the ideal refracted direction (Snell's Law) $k_t I_t$

$$I_r = I_a k_a + \sum_{i=1}^{\ell} S_i f_{att_i} I_{p_i} \left[k_d (N \cdot L_i) + k_s (R_i \cdot V)^n \right] + k_s I_s + k_t I_t$$

Refraction

Fig. 9. Refraction causes the ruler to appear bent in a glass of water.

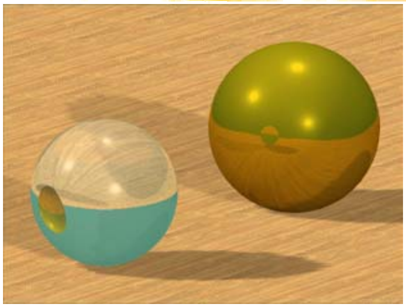
Refraction Geometry



$$\frac{\sin\theta_1}{\sin\theta_2} = \eta_{21} = \frac{\eta_2}{\eta_1} \quad , \mathbf{T} = \alpha \mathbf{I} + \beta \mathbf{N}$$

Fig. 10. The geometry of transmission.

And the result is…



Mirror morphine This scene is composed entirely of spheres: the 40-sphere morphine molecule is tucked in the corner between two large mirrored balls and the yellow ground ball. The image was calculated at a resolution of 2048 × 2048 with 10 levels of reflections, 3 × 3 supersampling, and analytic penumbra calculations (not probabilistic methods), in 8 days of VAX 11/780 time. For a discussion of shadows and penumbrae, see Section 5.1. (Copyright © Paul Heckbert, NYIT, 1983)

The RT Algorithm

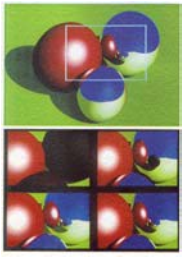
- For each pixel (x,y) in the image, generate the corresponding ray in 3D.
- Image(x,y) := TraceRay(ray)
- TraceRay(ray)
  - compute nearest ray-surface intersection
  - if none found, return background color
  - compute direct illumination
  - compute illumination arriving from reflected direction
  - compute illumination arriving from refracted direction
  - combine illumination components using the shading model
  - return resulting color

The RT Algorithm

- Direct illumination: test the visibility of each source by shooting a shadow ray towards it. Only sources which are found visible are summed in the shading model.

- Reflected/refracted illumination: a recursive call to TraceRay with the reflected/refracted ray as argument.

The depth of reflection



Ray-Surface Intersection

- Implicit surfaces: $f(x,y,z)=0$
  - Use a parametric representation for the ray:

$$R(t)=O+tD$$
$$R_x(t)=O_x+tD_x$$
$$R_y(t)=O_y+tD_y$$
$$R_z(t)=O_z+tD_z$$

  - Substitute into the implicit equation:

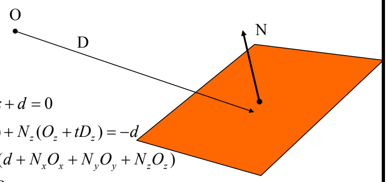
$$f(O_x+tD_x,O_y+tD_y,O_z+tD_z)=0$$

  - Solve the resulting equation
  - Examples: plane, sphere

Ray Plane intersection
Implicit Formulation

- Find ‘t’ such that f(x,y,z) = 0

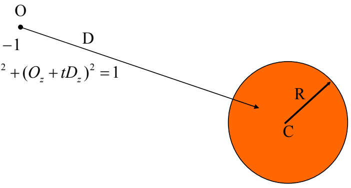
$$R(t)=O+tD$$
$$R_x(t)=O_x+tD_x$$
$$R_y(t)=O_y+tD_y$$
$$R_z(t)=O_z+tD_z$$
$$f(x,y,z)=N_xx+N_yy+N_zz+d=0$$
$$N_x(O_x+tD_x)+N_y(O_y+tD_y)+N_z(O_z+tD_z)=-d$$
$$(N_xD_x+N_yD_y+N_zD_z)t=-(d+N_xO_x+N_yO_y+N_zO_z)$$
$$t=-\frac{d+N_xO_x+N_yO_y+N_zO_z}{N_xD_x+N_yD_y+N_zD_z}$$



Ray Sphere intersection

- Find ‘t’ such that f(x,y,z) = 0

$$R(t)=O+tD$$
$$f(x,y,z)=x^2+y^2+z^2-1$$
$$(O_x+tD_x)^2+(O_y+tD_y)^2+(O_z+tD_z)^2=1$$
$$\dots$$



Ray-Surface Intersection

- Parametric surfaces: $$S(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$$
- Several approaches:
  - Tessellation
  - Subdivision
  - Implicitization
  - Other numerical methods (involve solving a system of two or three nonlinear equations)

Ray-Plane Intersection
Explicit formulation

- Find t, u,v such that:

$$\begin{bmatrix} O_x + tD_x \\ O_y + tD_y \\ O_z + tD_z \end{bmatrix} = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = u \begin{bmatrix} x_u(u,v) \\ y_u(u,v) \\ z_u(u,v) \end{bmatrix} + v \begin{bmatrix} x_v(u,v) \\ y_v(u,v) \\ z_v(u,v) \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}$$

- Linear system 3 equations, 3 unknowns

Advantages of Ray Tracing Algorithm

- Computes global illuminations effects:
  - Shadows
  - Reflections
  - Refractions
- Computes visibility and shading at once
- Consistent and easy implementation
- Can be extended easily
- Can be parallelized

Disadvantages of Ray Tracing

- Slow
- Memory bound – all objects must be kept in memory
- Does not compute all global illuminations effects:
  - Caustics
  - Color Bleeding
  - More…

Accelerating Ray Tracing

- Four main groups of acceleration techniques:
  - Parallelization, specialized hardware
  - Reducing the total number of rays that are traced
    - Adaptive recursion depth control
  - Reducing the average cost of intersecting a ray with a scene:
    - Faster intersection calculations
    - Fewer intersection calculations
  - Using generalized rays
    - beams
    - cones
    - pencils

Parallel/Distributed RT

- Two main approaches:
  - Each processor is in charge of tracing a subset of the rays. Requires a shared memory architecture, replication of the scene database, or transmission of objects between processors on demand.
  - Each processor is in charge of a subset of the scene (either in terms of space, or in terms of objects). Requires processors to transmit rays among themselves.

The Ray Tree

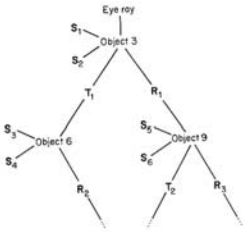


Fig. 12. The ray tree in schematic form.

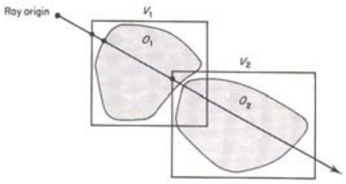
Accelerating Ray Tracing

- Faster intersection calculations:
  - Object-dependent optimizations
  - Bounding volumes
- Fewer intersection calculations:
  - Bounding volume hierarchy
  - Spatial subdivisions:
    - Uniform grids
    - Octrees
    - BSP-trees
    - Hybrids
  - Directional techniques
    - The light buffer
    - Ray classification

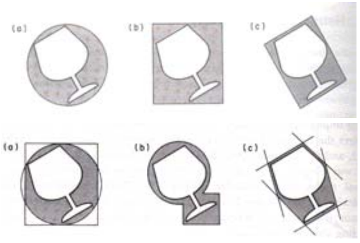
Bounding Volumes

- Idea: associate with each object a simple bounding volume. If a ray misses the bounding volume, it also misses the object contained therein.
- Common bounding volumes:
  - spheres
  - bounding boxes
  - bounding slabs
- Effective for additional applications:
  - Clipping acceleration
  - Collision detection
- Note: bounding volumes offer no asymptotic improvement!

Bounding Boxes



Bounding Volumes



Bounding Volume Hierarchy

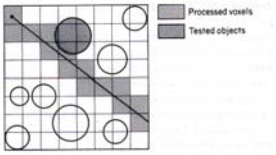
- Introduced by James Clark (SGI, Netscape) in 1976 for efficient view frustum culling.

```
Procedure IntersectBVH(ray, node)
begin
   if IsLeaf(node) then
      Intersect(ray, node.object)
   else if IntersectBV(ray,node.boundingVolume)
   then
      foreach child of node do
         IntersectBVH(ray, child)
      endfor
   endif
end
```

Spatial Subdivision

- Uniform spatial subdivision:
  - The space containing the scene is subdivided into a uniform grid of cubes "voxels".
  - Each voxel stores a list of all objects at least partially contained in it.in
  - Given a ray, voxels are traversed using a 3D variant of the 2D line drawing algorithms.
  - At each voxel the ray is tested for intersection with the primitives stored therein
  - Once an intersection has been found, there is no need to continue to other voxels.

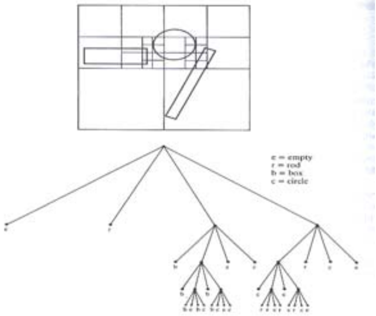
Uniform Subdivision



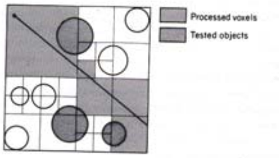
Adaptive Spatial Subdivision

- Disadvantages of uniform subdivision:
  - requires a lot of space
  - traversal of empty regions of space can be slow
  - not suitable for "teapot in a stadium" scenes
- Solution: use a hierarchical adaptive spatial subdivision data structure
  - octrees
  - BSP trees
- Given a ray, perform a depth-first traversal of the tree. Again, can stop once

Octrees



Octree traversal



Directional Techniques

- Light buffer: accelerates shadow rays.
  - Discretize the space of directions around each light source using the *direction cube*
  - In each cell of the cube store a sorted list of objects visible from the light source through that cell
  - Given a shadow ray locate the appropriate cell of the direction cube and test the ray with the objects on its list

Directional Techniques

- Ray classification (Arvo and Kirk 87):
  - Rays in 3D have 5 degrees of freedom: (x,y,z,θ,ϕ)
  - Rays coherence: rays belonging to the same small 5D neighborhood are likely to intersect the same set of objects.
  - Partition the 5D space of rays into a collection of 5D hypercubes, each containing a list of objects.
  - Given a ray, find the smallest containing 5D hypercube, and test the ray against the objects on the list.
  - For efficiency, the hypercubes are arranged in a hierarchy: a 5D analog of the 3D octree. This data structure is constructed in a lazy fashion.