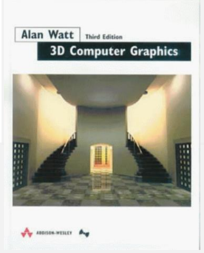
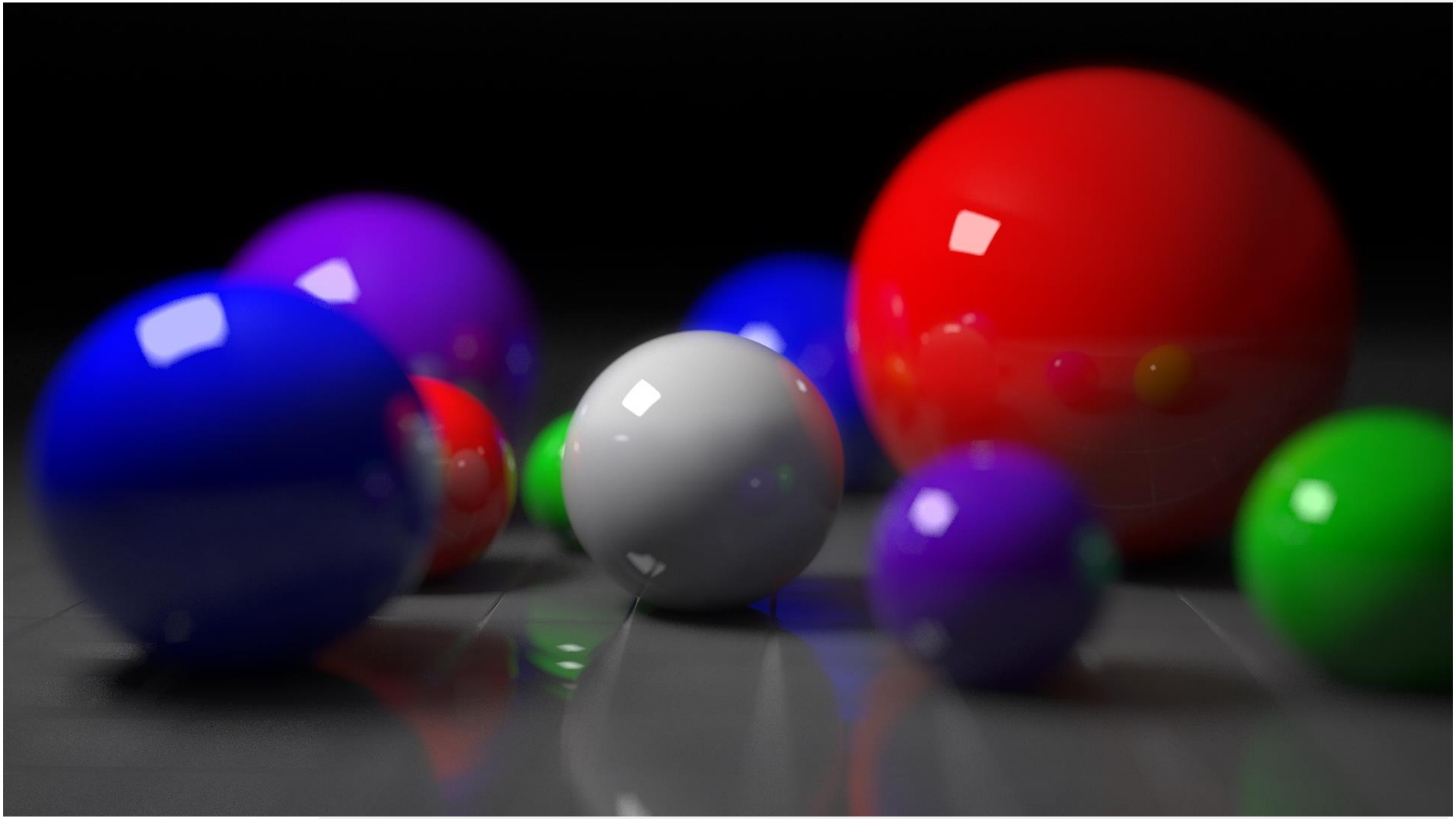
COM3503/4503/6503: 3D Computer Graphics

Lectures 18: Ray tracing: Part 2



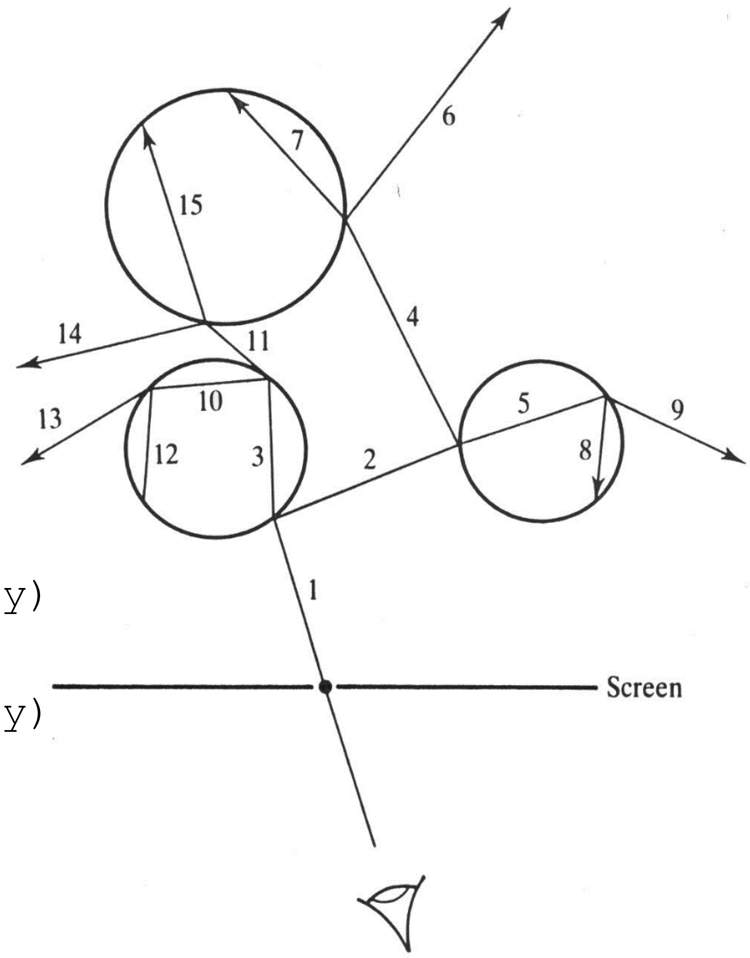
Dr. Steve Maddock, s.maddock@sheffield.ac.uk



0. Summary of part 1

shootRay (ray structure)

```
intersection test for all objects;
if  ray intersects objects {
  get closest object intersection;
  for every light cast shadow ray;
  get normal at intersection point;
  calculate local intensity (I_local);
  if (reflection)
     calculate and shootRay(reflected ray)
  if (refraction)
     calculate and shootRay(refracted ray)
  Intensity at hit point P = local
     + reflected + transmitted
}
```



1. Part 2

Part 1

- Visible surface ray tracing
- Naïve, recursive (Whitted) ray tracing
  - HSR, Shadows, Reflection, Refraction, Recursion

Part 2

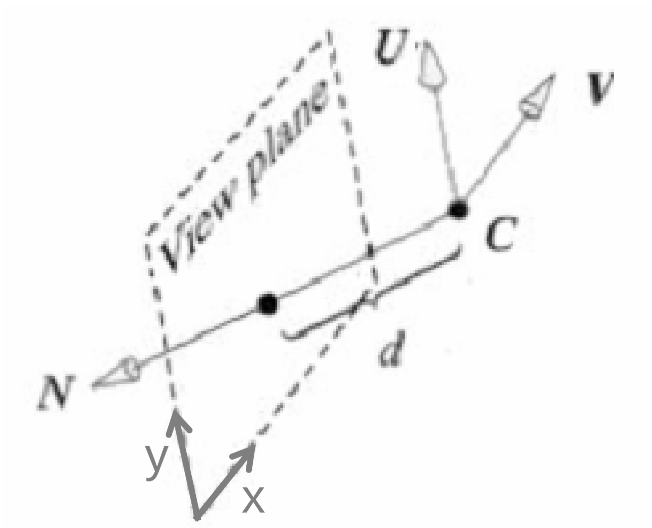
- Intersection calculations
- Speed-up techniques

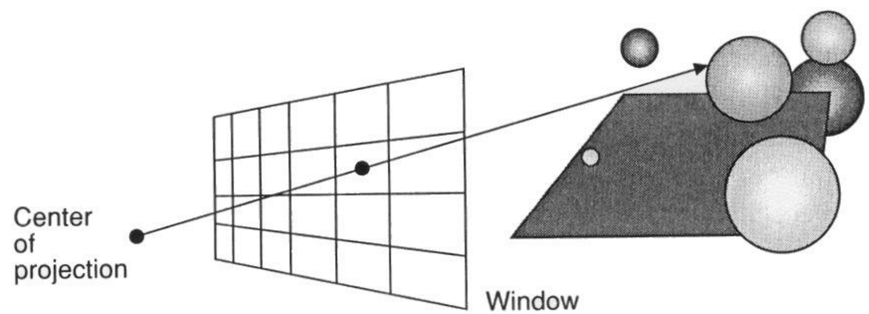
Part 3

- Anti-aliasing
- Advanced techniques

2. Initial ray direction

- Position of camera and view plane are defined
  - Viewplane is at distance `d` from camera along `N`
- A ray is traced for each screen pixel
  - Ray start position and direction calculated using camera and (x,y) screen position



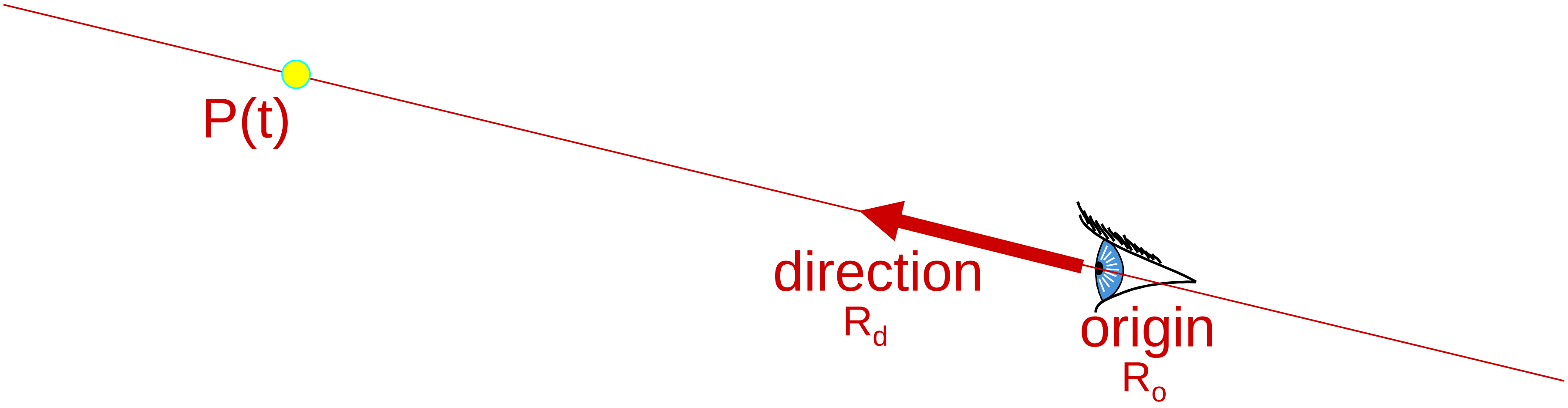


2.1 Ray representation

- $P(t) = R_0 + t * R_d$ = origin + t * direction
- Given two points, (x_1,y_1,z_1) and (x_2,y_2,z_2), which are ray origin and 'some point':

$$x = x_1 + (x_2-x_1)t = x_1 + it$$
$$y = y_1 + (y_2-y_1)t = y_1 + jt$$
$$z = z_1 + (z_2-z_1)t = z_1 + kt$$

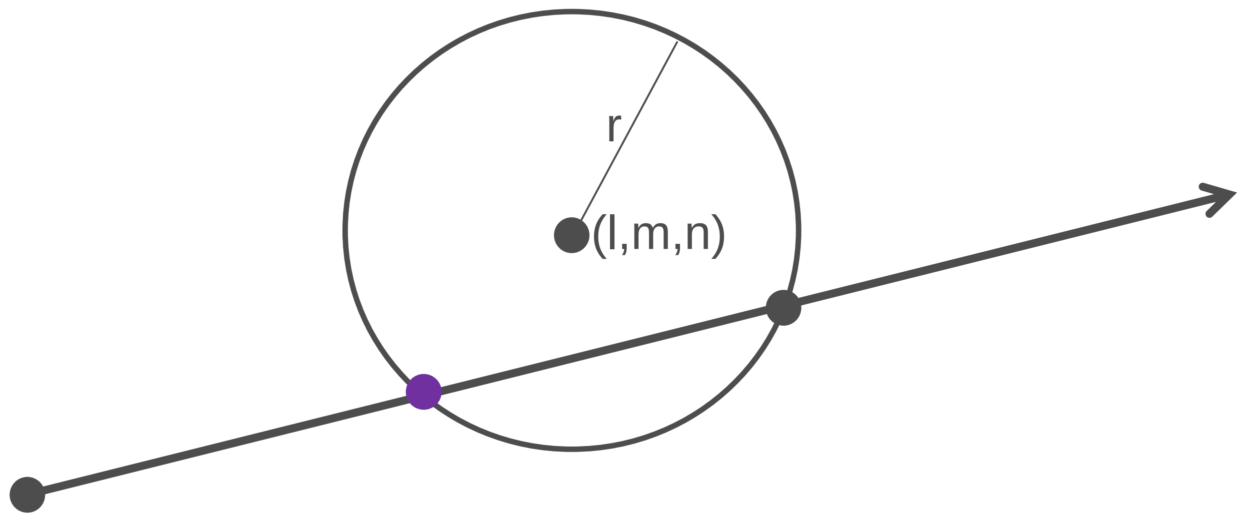
where (i,j,k) is the direction



Durand, 06

3. Ray/sphere intersection

- Given equation of ray (line) using two points (x_1,y_1,z_1) and (x_2,y_2,z_2)
- Sphere with centre (l,m,n), radius r: $(x-l)^2 + (y-m)^2 + (z-n)^2 = r^2$
- Substitute line equation into sphere equation, and solve for the roots, choosing **closest** root



3. Ray/sphere intersection

- Substitution gives a quadratic in t: $at^2 + bt + c = 0$
  where

  $a = i^2 + j^2 + k^2$

  $b = 2i(x_1-l) + 2j(y_1-m) + 2k(z_1-n)$

  $c = l^2 + m^2 + n^2 + x_1^2 + y_1^2 + z_1^2 + 2(-lx_1-my_1-nz_1) - r^2$

  $t = (-b \pm \text{sqrt}(b^2-4ac)) / 2a$

- If b^2-4ac is

  < 0 then no intersection, since no real solutions

  = 0 then line grazes sphere

  > 0 then roots give front and back intersection.

- If $b^2-4ac > 0$ then substitute roots into line equation to give (x,y,z) of intersection points; and use intersection at smallest +ve t;
- Since it is a sphere, normal at intersection point (x_i,y_i,z_i) is given by:

  $\mathbf{N} = ((x_i-l)/r, (y_i-m)/r, (z_i-n)/r)$

3.1 Geometric method

Ray: $P = P_0 + tV$
Sphere: $|P - O|^2 - r^2 = 0$

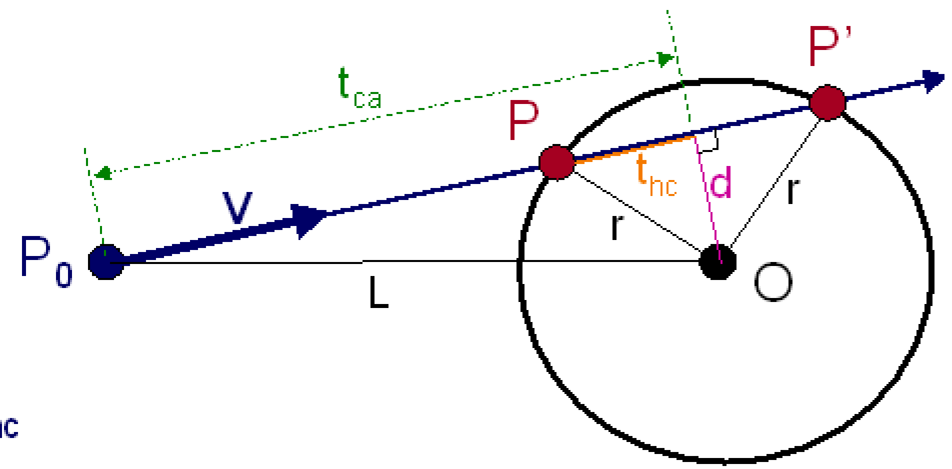
$L = O - P_0$

$t_{ca} = L \bullet V$
if ($t_{ca} < 0$) return 0

$d^2 = L \bullet L - t_{ca}^2$
if ($d^2 > r^2$) return 0

$t_{hc} = \text{sqrt}(r^2 - d^2)$
$t = t_{ca} - t_{hc}$ and $t_{ca} + t_{hc}$

$P = P_0 + tV$

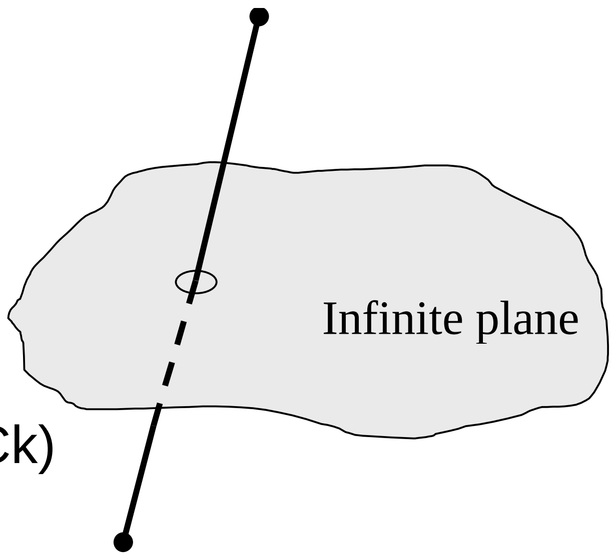


http://www.cs.princeton.edu/courses/archive/fall00/cs426/lectures/raycast/sld013.htm
Also see Glassner, 1990, p.388

4. Ray/plane intersection

- Equation for a plane:

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



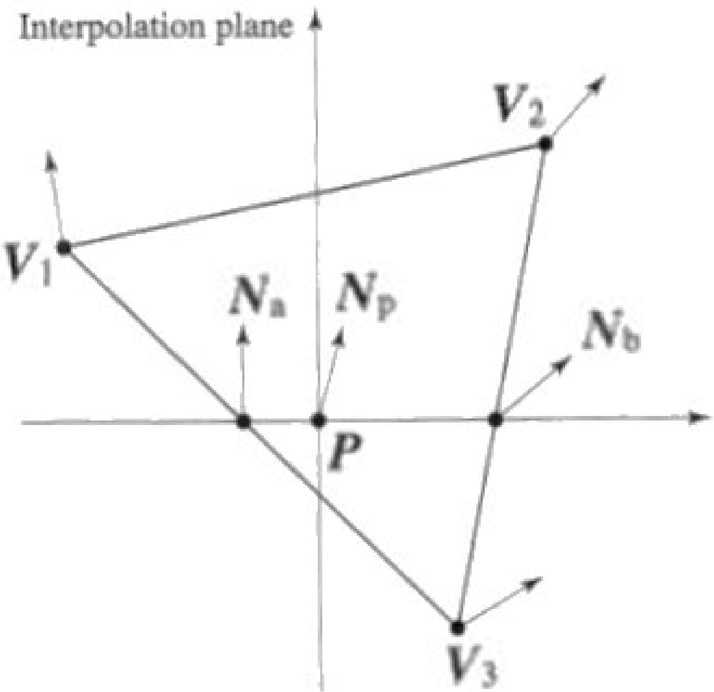
- Substitute the line equation into the plane equation to give the intersection point:

  $$t = -(Ax_1 + By_1 + Cz_1 + D) / (Ai + Bj + Ck)$$

- If $(Ai + Bj + Ck) = 0$, then the line and the plane are parallel and no intersection occurs
- If $(t < 0)$ then behind start point of ray

5. Ray/polygon intersection

- Find the plane equation for the polygon – need 3 non colinear points
  - A, B, C are the components of the polygon's normal vector N_p and D is computed by substituting any vertex into (Ax+By+Cz+D=0)
- Check for ray/plane intersection
- Check if the intersection point is in the polygon
  - use of Barycentric coordinates



5.1 Barycentric coordinates

- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$ with $\alpha + \beta + \gamma = 1$
  and $0 \le \alpha \le 1$ and $0 \le \beta \le 1$ and $0 \le \gamma \le 1$
  where $\alpha = A_a/A$ $\beta = A_b/A$ $\gamma = A_c/A$
- Rewrite:

$$P(\beta, \gamma) = a + \beta(b-a) + \gamma(c-a)$$

- Set ray eqn equal to this

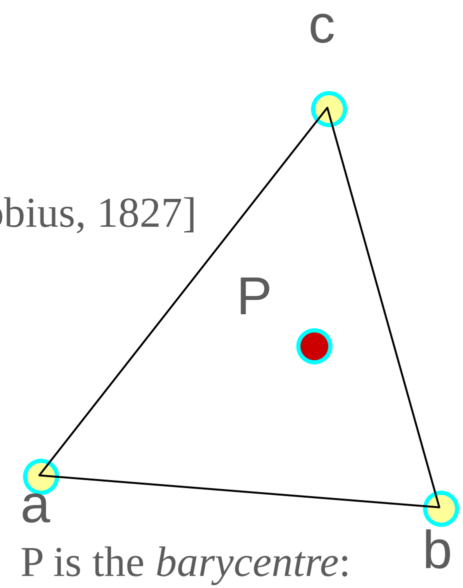
$$P(t) = P(\beta, \gamma)$$

$$R_o + t * R_d = a + \beta(b-a) + \gamma(c-a)$$

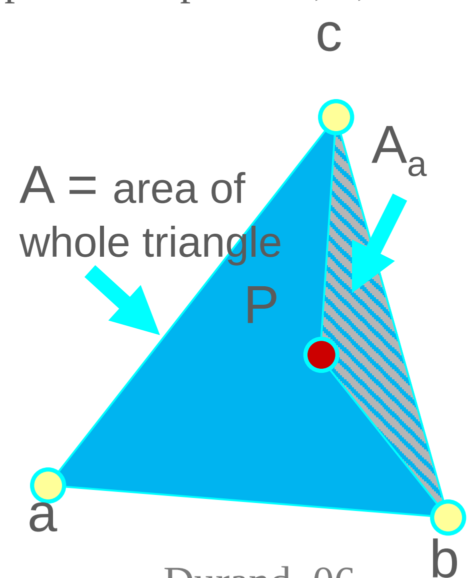
- Solve the 3 equations (in x, y and z) for the 3 unknowns t, β and γ
- Intersection if:

$$\beta + \gamma < 1 \quad \text{and} \quad \beta > 0 \quad \text{and} \quad \gamma > 0$$

[Möbius, 1827]



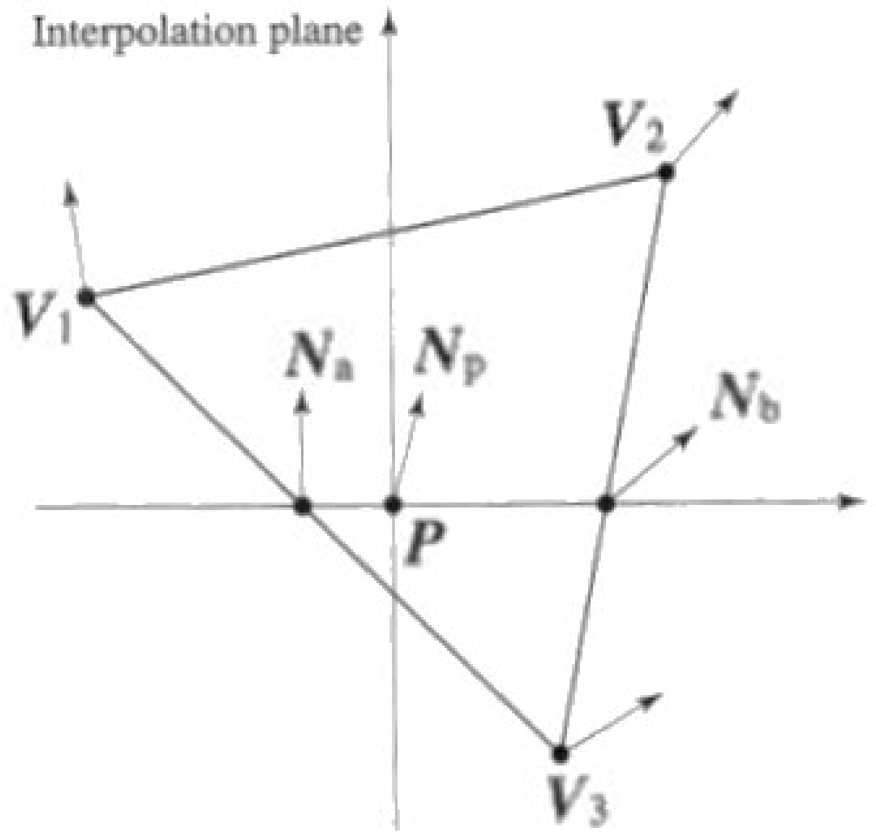
P is the *barycentre*: the single point upon which the plane would balance if weights of size α, β, & γ are placed on points a, b, & c.



Durand, 06

5.2 Calculating the normal

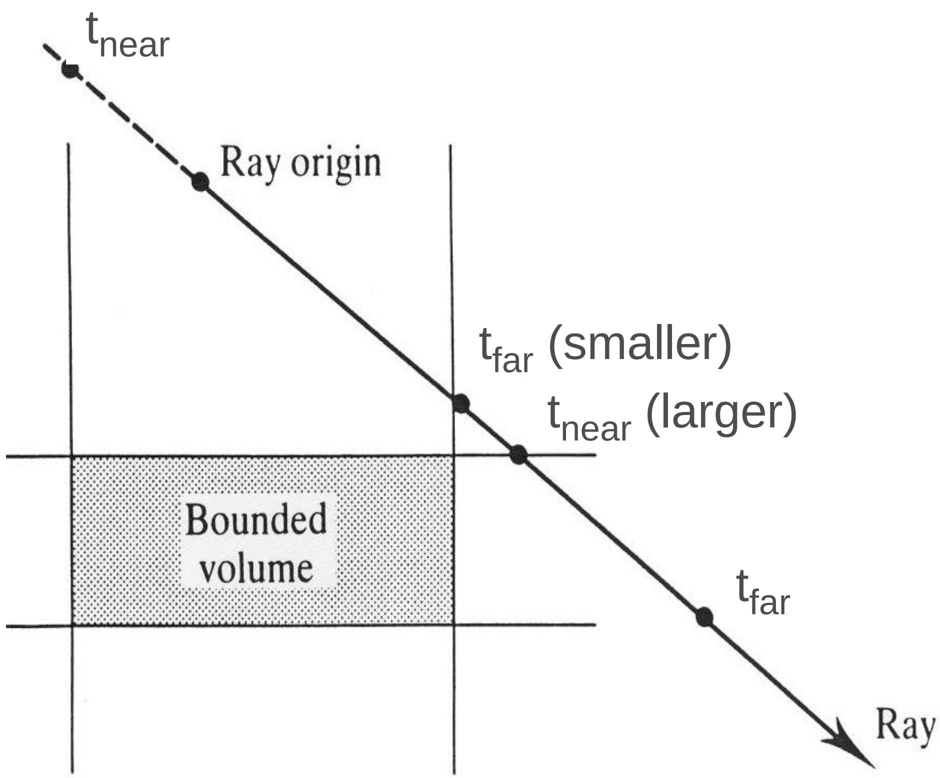
- The normal at the intersection point can be calculated using a linear combination of the polygon's vertex normals

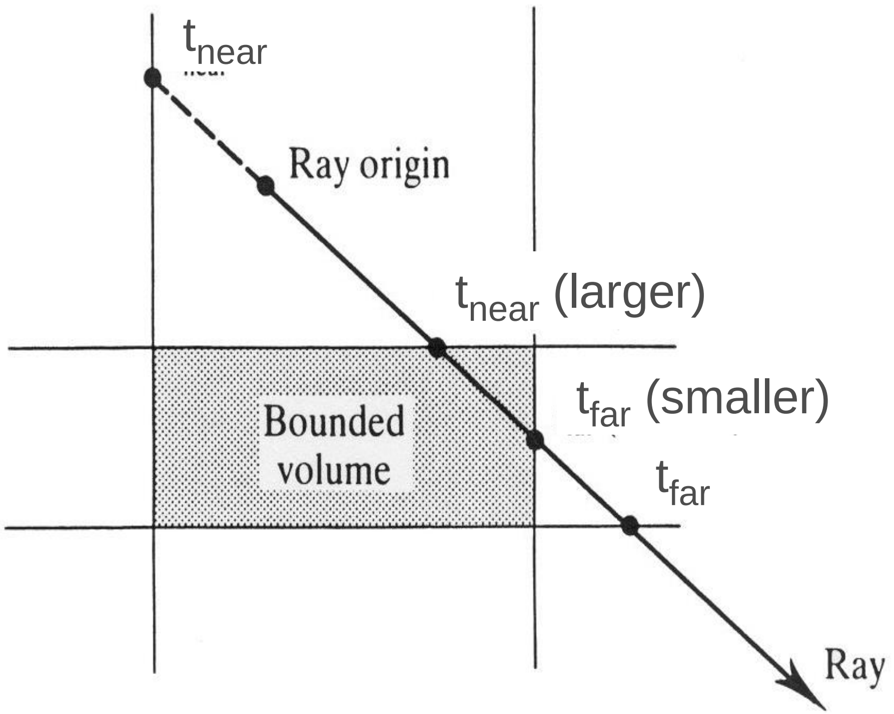


- A comment on ray/polygon test:
  - **Expensive process**, since have to repeat for every polygon in the polygon mesh

6. Ray/box intersection

- For each pair of planes defining the box, find the t_{near} and t_{far} intersection
- *Test*: If larger value of t_{near} > smaller value of t_{far} then no intersection with the box





7. Speed-up techniques

- Intersection testing for every polygon of every object is impractical
- A 1000x1000 image of a scene containing 100 objects each of 1000 polygons would require 100 billion intersection calculations for **first hit** ray tracing
- Reflection rays, refraction rays and shadow rays would increase this exponentially
- (Whitted, 80) estimates 75 to 95 percent of rendering time is intersection calculations
- We cannot cull or clip
  - An out-of-view object part may reflect in a visible object part
- We must use one or more of a range of speed-up techniques

7. Speed-up techniques

Simple, limited techniques include:

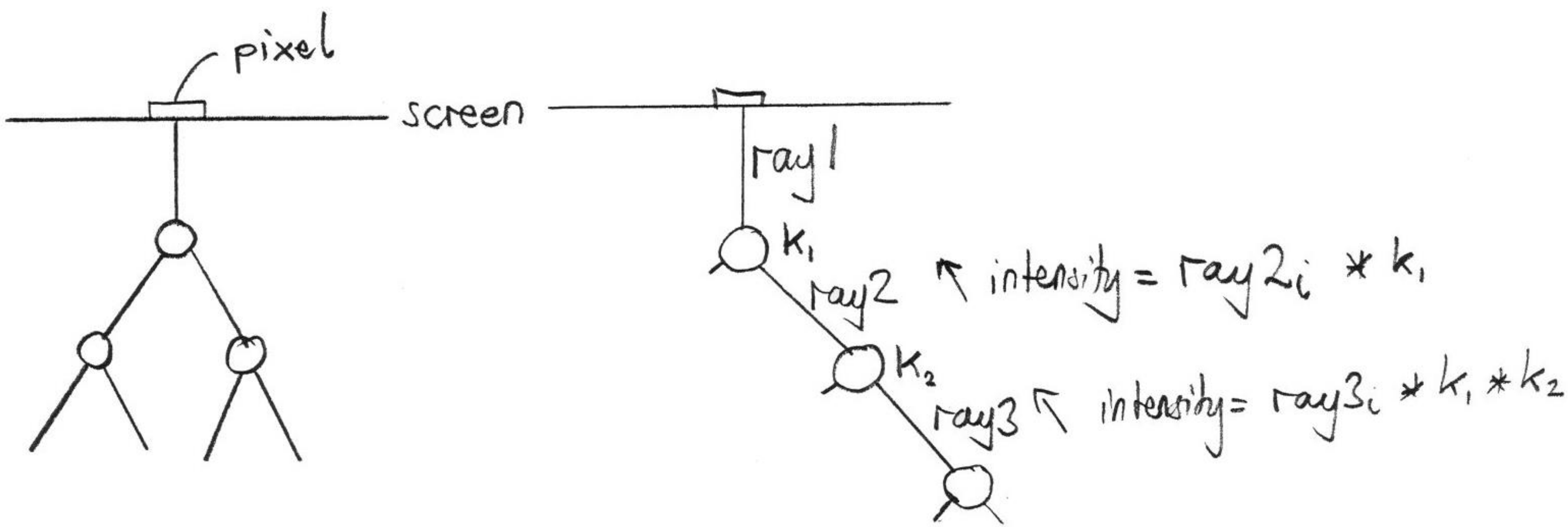
- First hit speed-up
  - Use z-buffer to determine closest object to a pixel
  - Ray trace through each pixel and immediately intersect with known closest object, then continue ray tracing as normal
- Adaptive tree-depth control
- Screen extents

Most used approaches:

- Bounding volumes and hierarchies of bounding volumes
- Spatial partitioning approaches

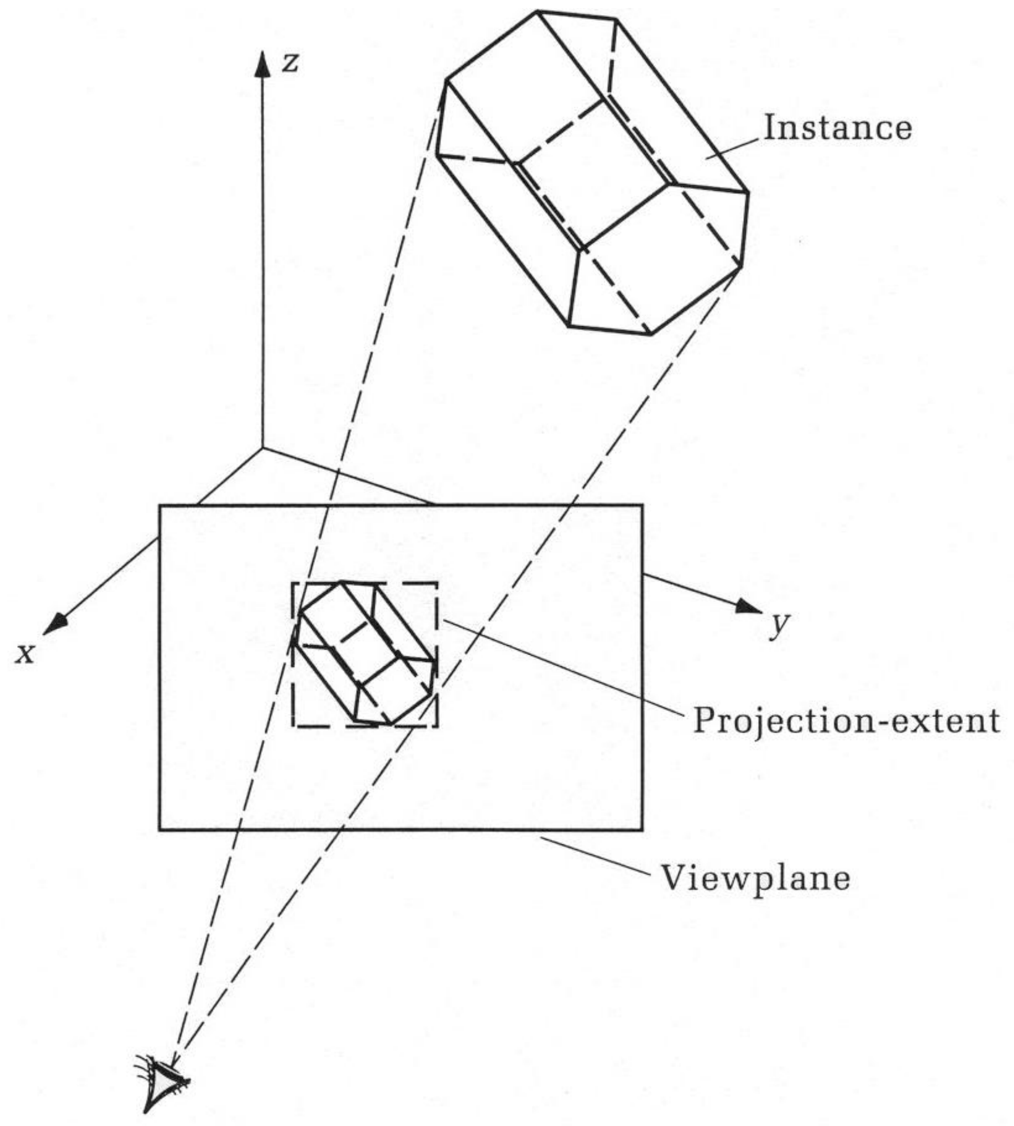
8. Adaptive tree-depth control

- Multiple interactions rapidly become insignificant as we proceed down the ray tree
- Terminate recursion when product of reflection and transmission attenuation factors is less than a threshold



9. Screen extents

- Project each object to screen space and only ray trace areas within projection extent



10. Bounding volume

- Enclose each object in a bounding volume, e.g. sphere
- Only test object if ray intersects bounding volume
  - One sphere test versus, say, test ray against 1000 polygons
- Efficiency depends on how well the object fits into the space of the bounding volume

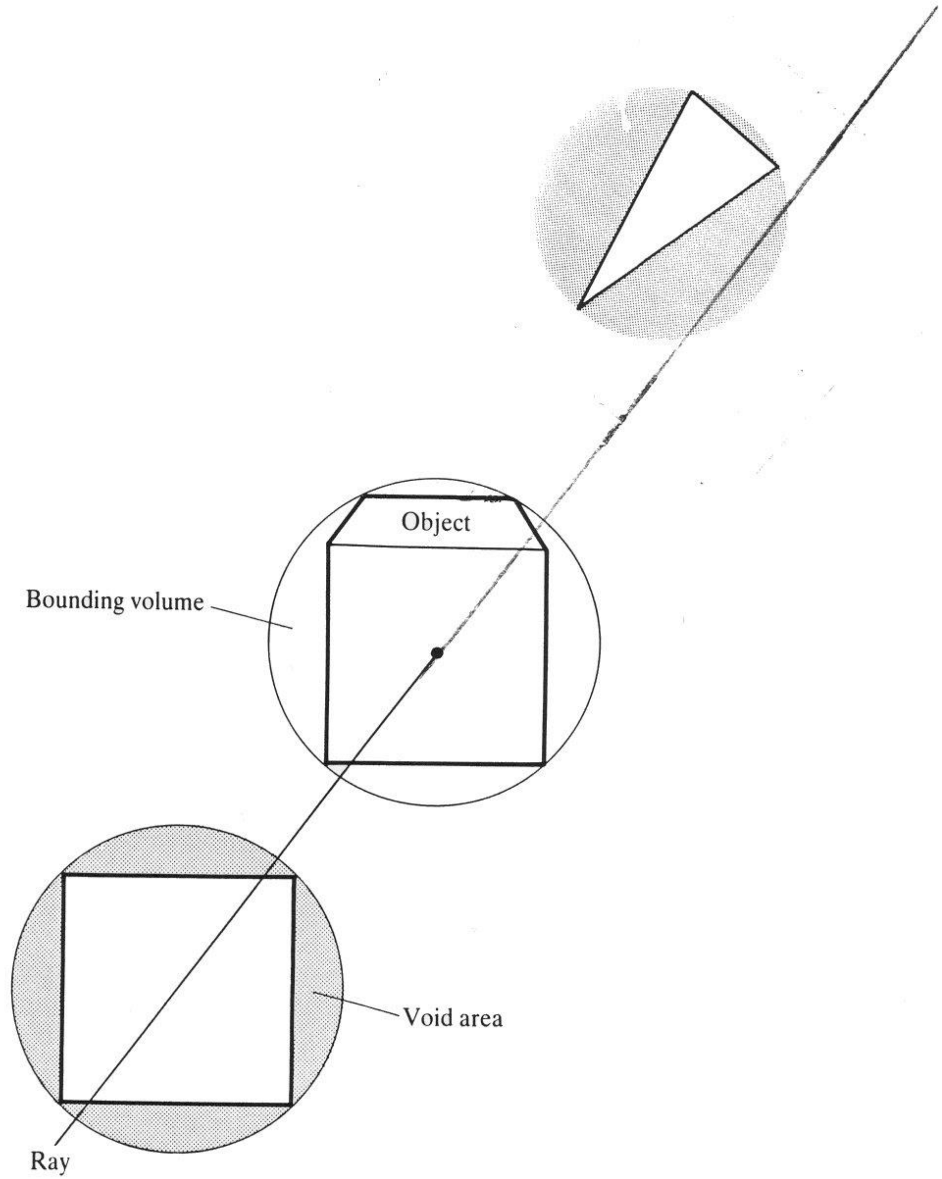
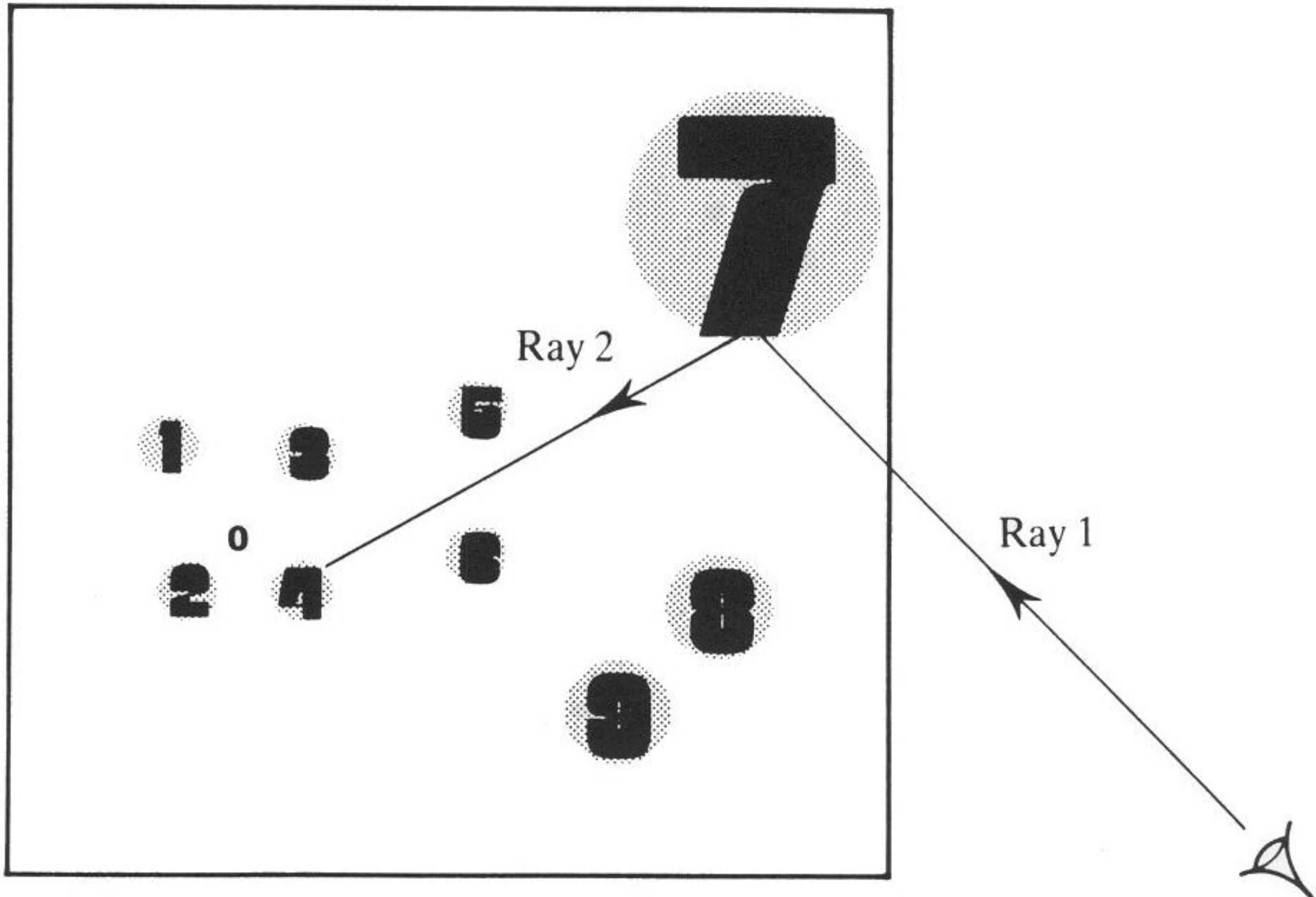
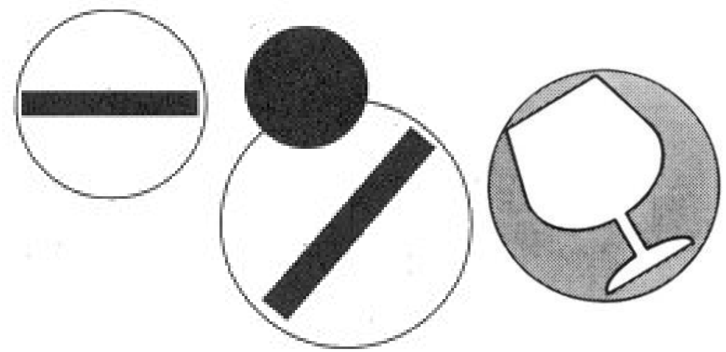


Figure 9.3 The 'efficiency' of a bounding volume is a function of the void area.

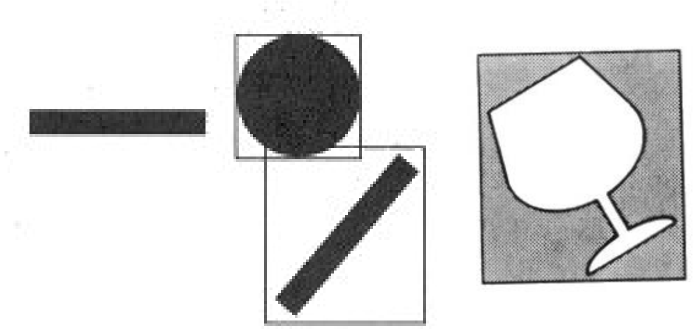

Ray 2
Ray 1
1
3
5
0
2
4
6
7
8
9

10.1 Different bounding volumes

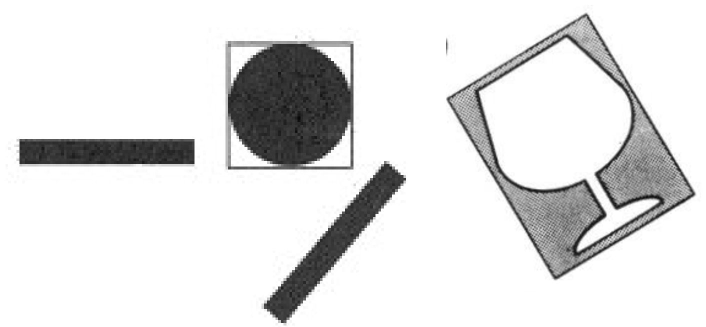
- Sphere
  - Stays same as object translates and rotates



- Axis-aligned bounding box (AABB)
  - Box stays aligned with coordinate system
  - Box alters in size to contain object as it rotates

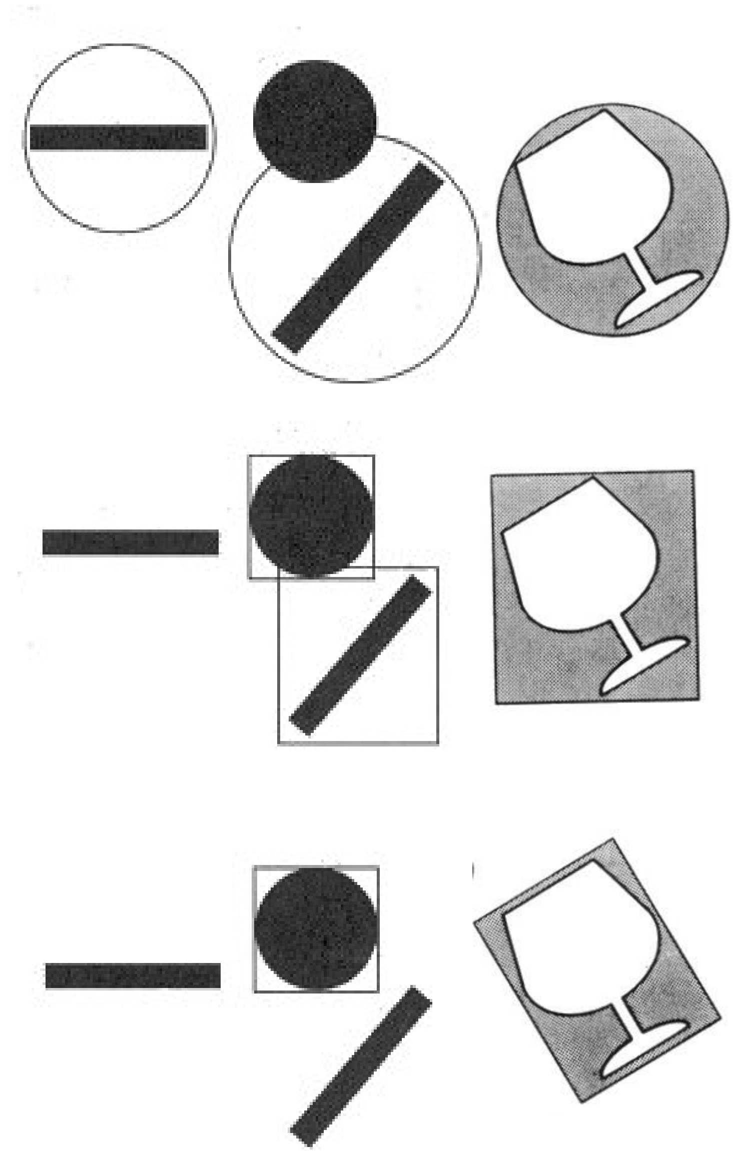


- Oriented bounding box (OBB)
  - Box translates and rotates with object



10.2 Comparing bounding volumes

- Compare tightness of fit vs. intersection speed
- Consider the wine glass and the rod:
  - Increasing tightness of fit: bounding spheres, AABBs, OBBs
  - Increasing intersection speed: OBBs, AABBs, bounding spheres
- Other issues:
  - Updating as object moves



10.3 Mixing bounding volumes

- Different bounding volumes can be combined to create a better fit at the expense of extra complexity

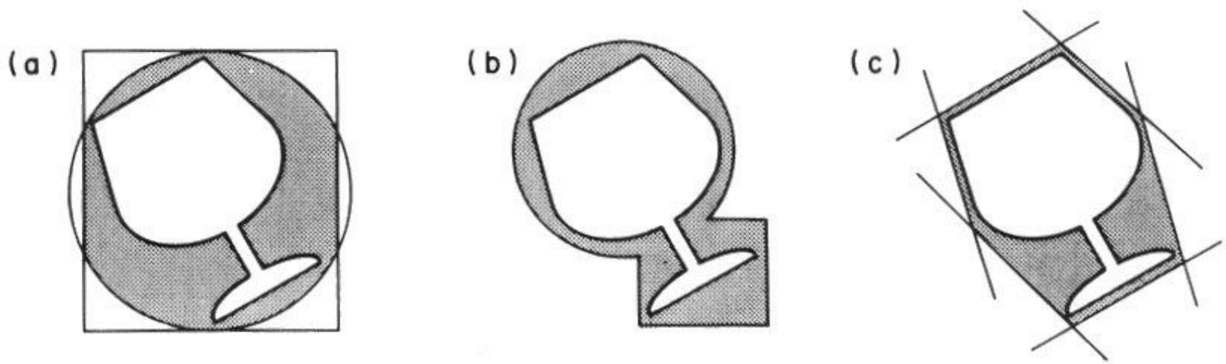
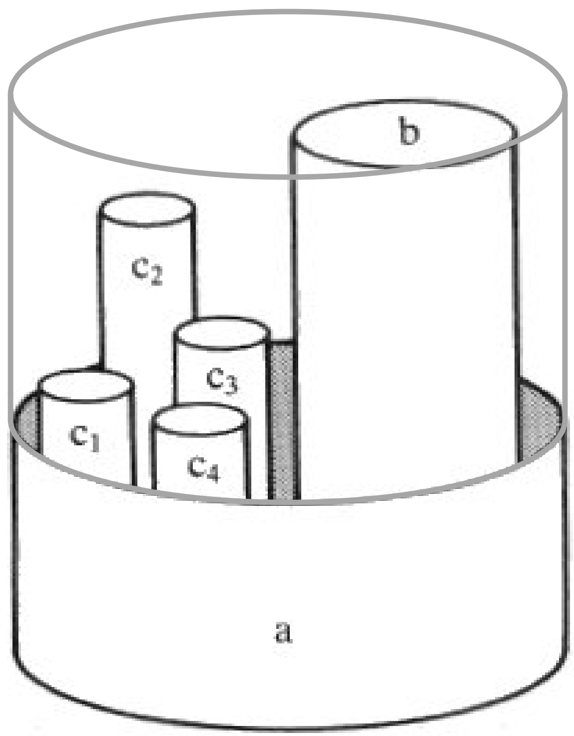
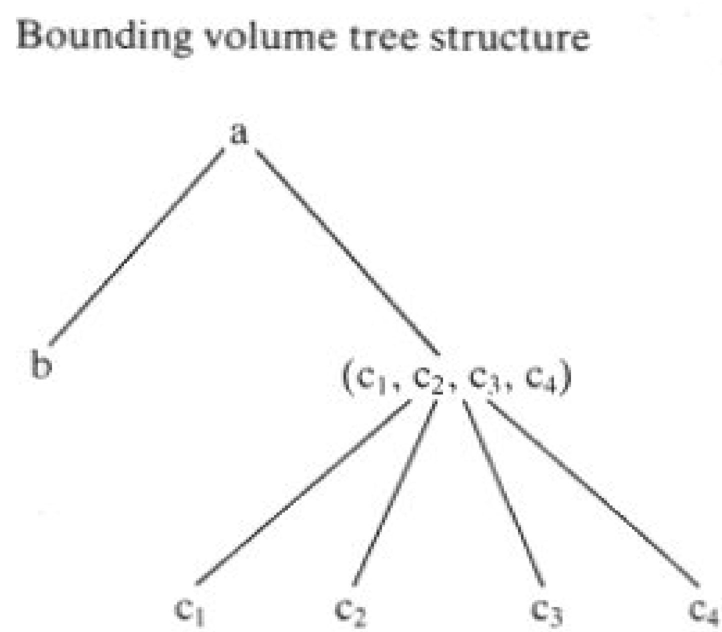


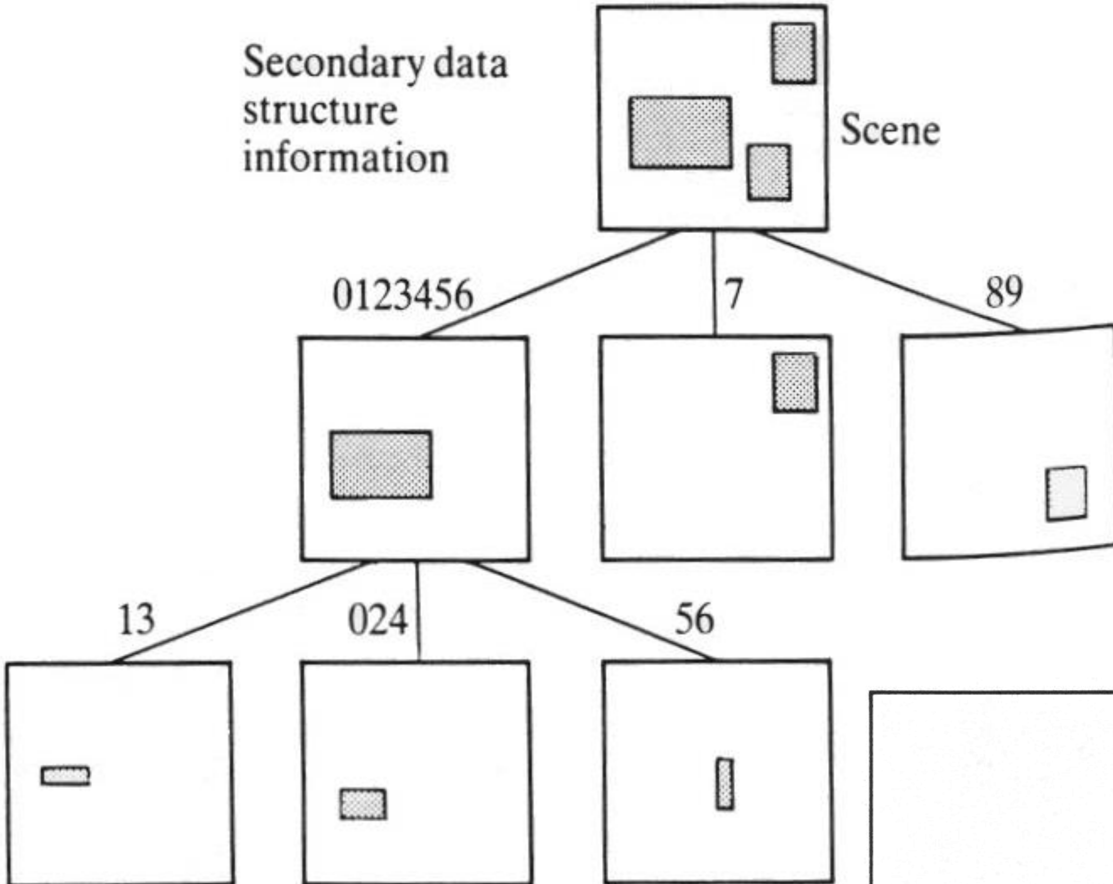
Fig. 6. The intersection and union of multiple bounding volumes can be used to obtain a better fit. Each approach requires a different ray-intersection algorithm for best performance. (a) Intersection of box and sphere. (b) Union of box and sphere. (c) Intersection of slabs.

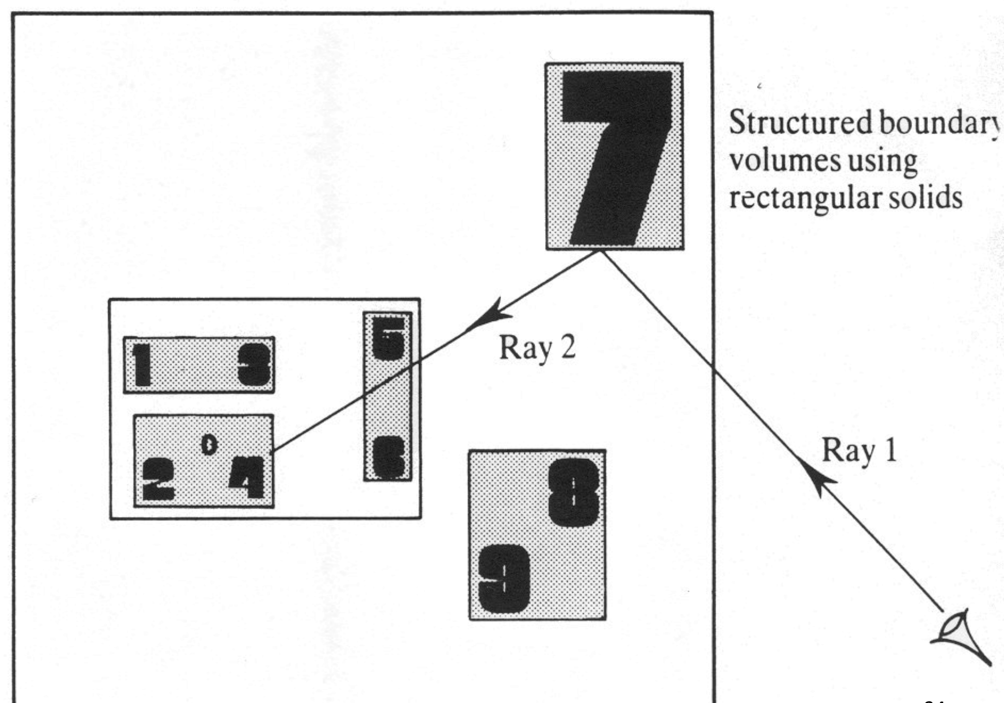
10.4 Hierarchies of bounding volumes

- Bound clusters of bounding volumes to form a hierarchy
  - Possibly use different bounding volumes for different cluster levels
- Test against successive tree levels before testing contained objects
- Can be labour intensive to set up and may need updating during animation, e.g. objects moving apart, or breaking up, etc.





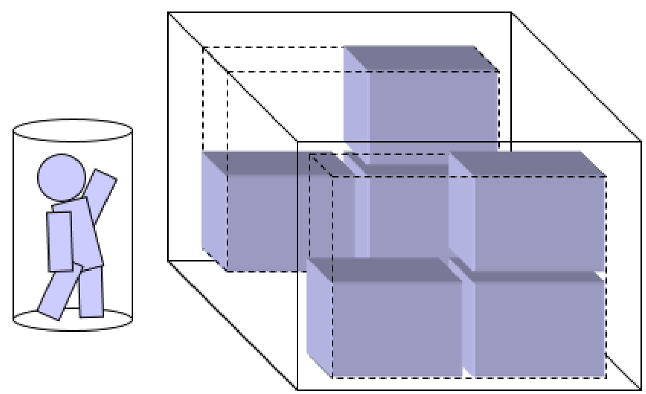

Secondary data structure information
Scene
0123456
7
89
13
024
56


7
Structured boundary volumes using rectangular solids
1
3
5
0
2
4
6
8
9
Ray 2
Ray 1

- Alex Benton, Univ Cambridge, Lecture series on ‘Advanced Graphics’

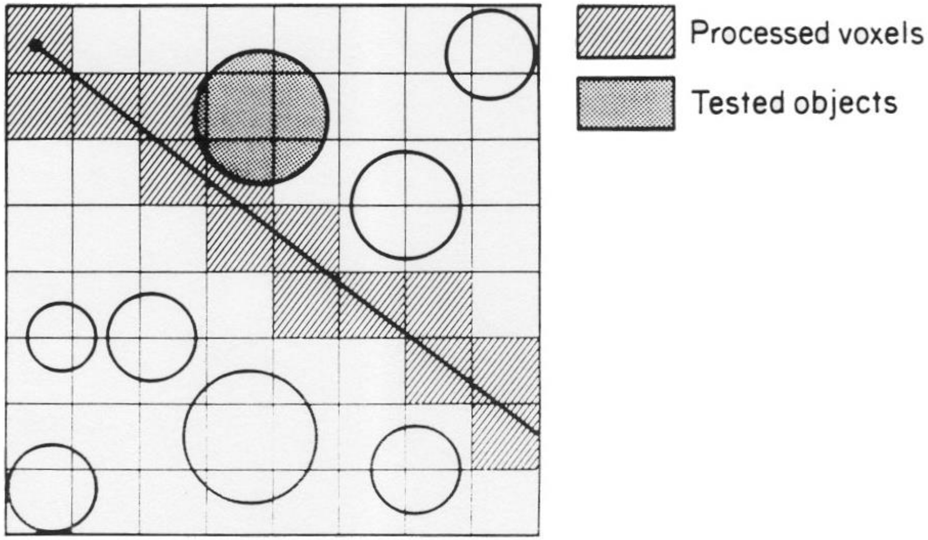
Your scene graph and you

- A common optimization derived from the scene graph is the propagation of *bounding volumes*.
  - These take many forms: bounding spheres, axis-aligned bounding boxes, oriented bounding boxes…
- Nested bounding volumes allow the rapid culling of large portions of geometry
  - Test against the bounding volume of the top of the scene graph and then work down.
- Great for…
  - Collision detection between scene elements
  - Culling before rendering
  - Accelerating ray-tracing



11. Spatial partitioning

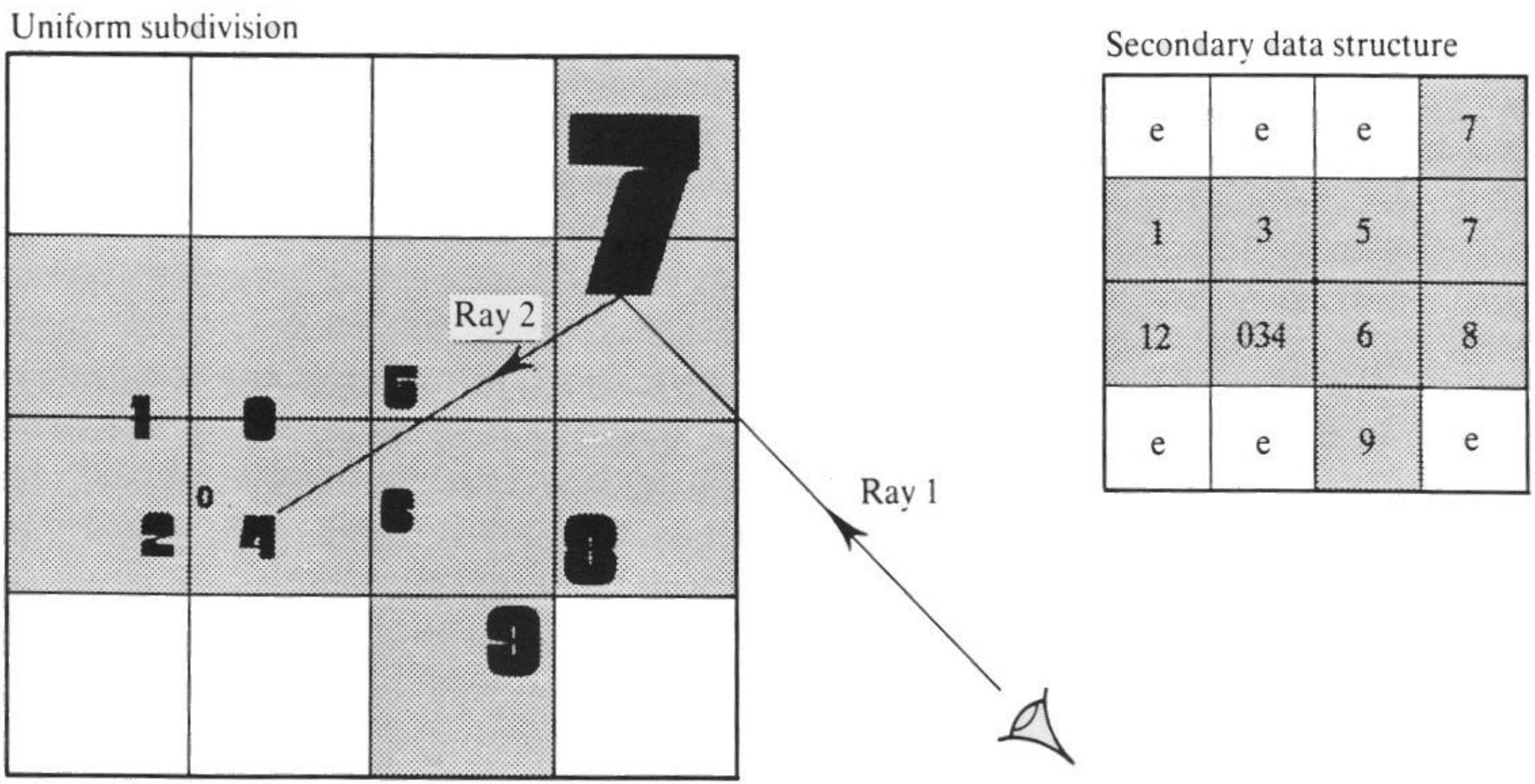
- Label world space by object occupancy
- Instead of testing a ray against all objects, we ask: “Is the region through which a ray is travelling occupied by any object? If so, which object?”



- *Aim*: make rendering time constant, eliminating dependence on scene complexity
- Common alternatives:
  - Regular grids
  - Octree
  - Binary space partitioning (BSP) tree

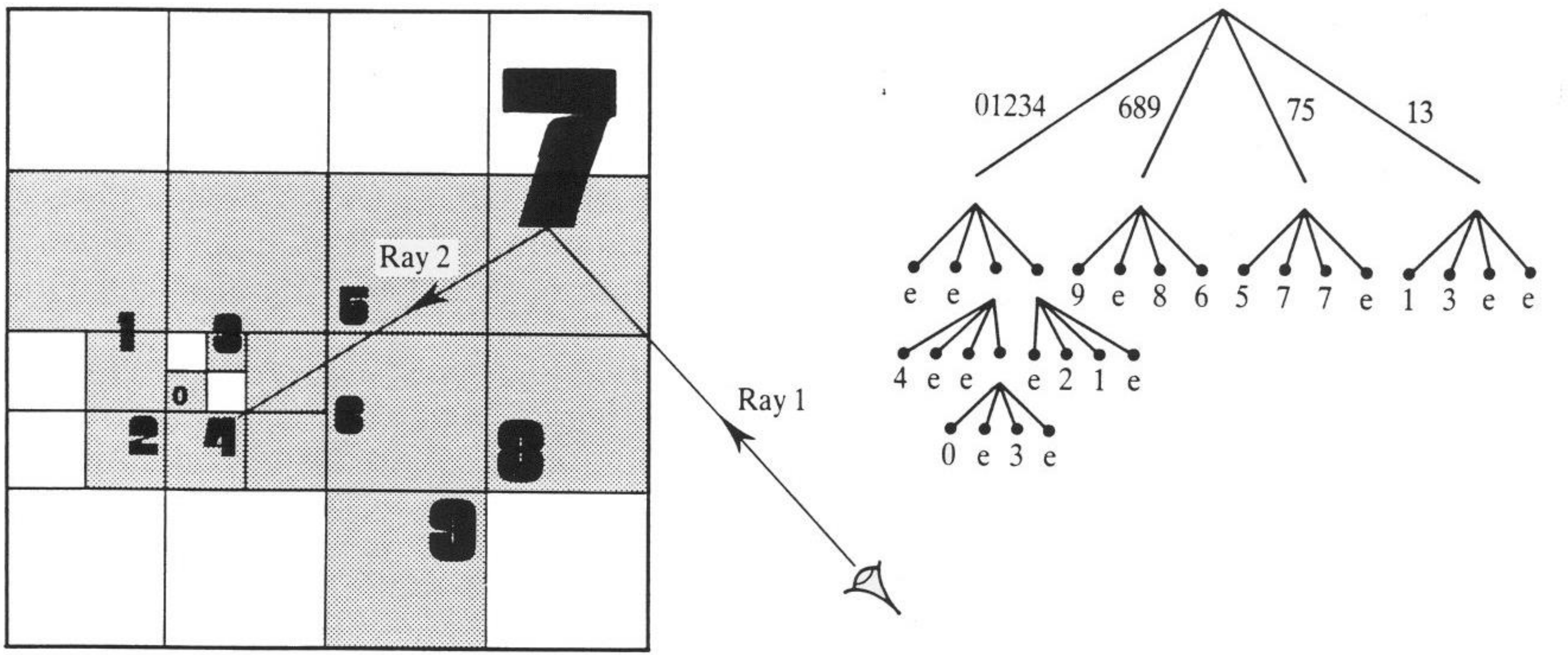
11.1 Uniform subdivision

- 3D space is divided into voxels – need to decide size of a voxel
  - Easy to step from cell to cell



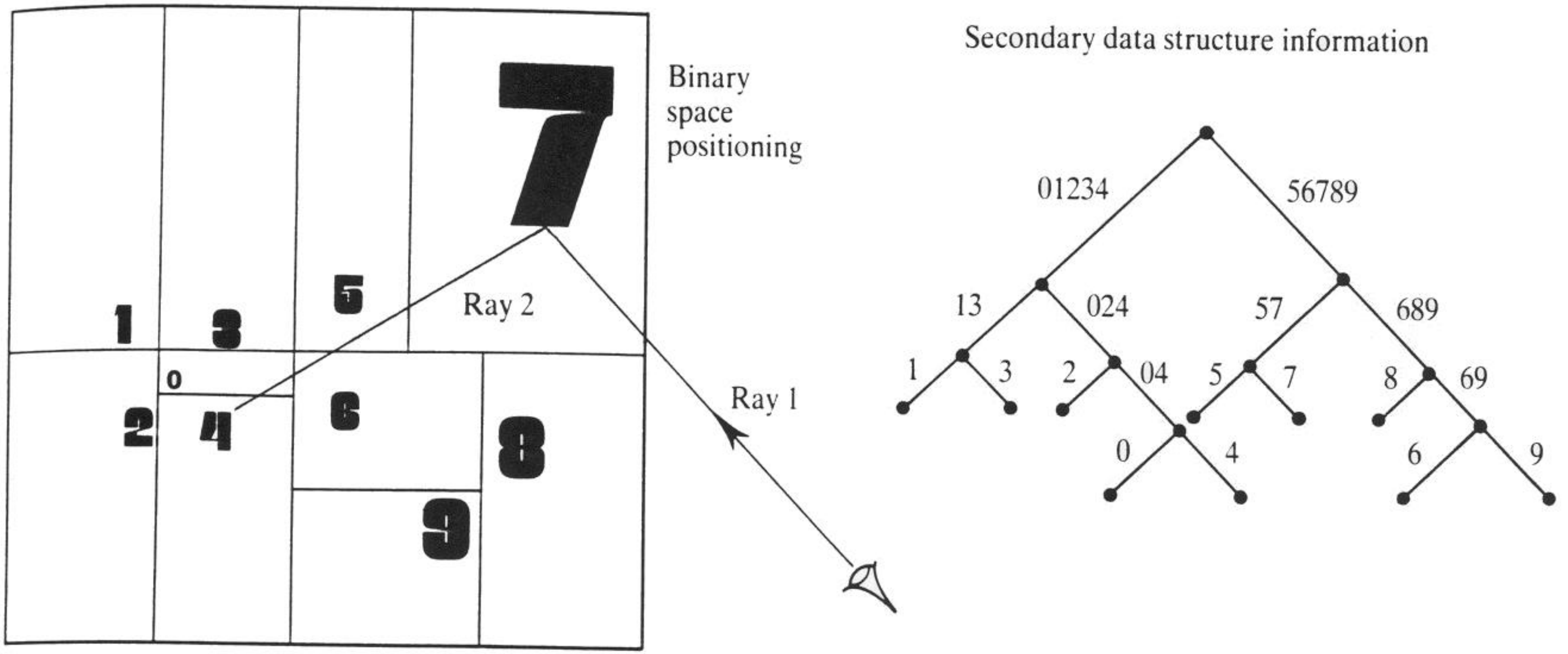
11.2 Octree

- Divide 3D space (which is a cube) into 8 cubes
- If needed, divide a cube into 8 smaller cubes, etc
- More difficult to step from cell to cell



11.3 BSP tree

- Use a plane to partition 3D space into two parts
- If needed, divide a part into two further parts, and so on
- Partition planes can be at any angle in the general scheme
- Issue: Finding good space partitions



12. Bounding volumes (BV) and spatial partitioning (SP)

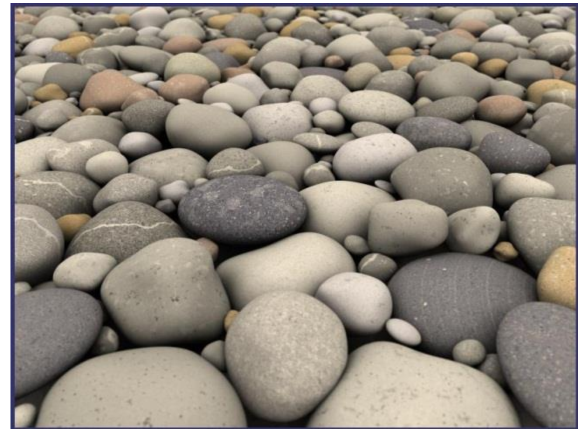
- BV Advantages
  - Saves on intersection tests
  - Easy to update single bounding volumes as objects move
- BV Disadvantages
  - May be difficult to bound objects efficiently
  - Need to update bounding volume hierarchy as objects move
- SP Advantages
  - Saves on intersection tests
  - Eliminate dependence on scene complexity
- SP Disadvantages
  - Potentially large size of secondary data structure
  - Need to update structure as objects move

13. Summary

- We trace rays from the eye into the scene
- The global part of the classic (Whitted) algorithm only deals with pure specular-specular interaction
- Direct diffuse calculation is done using Phong for local illumination, but diffuse-diffuse interaction between objects is not included
- Intersection calculations dominate and we need speed-up techniques for any practical ray tracer

POV-Ray: Persistence of Vision Raytracer: www.povray.org

"Office" by Jaime Vives Piqueres (2004)



"Pebbles" by Jonathan Hunt (2008)

"Christmas Baubles" by Jaime Vives Piqueres (2005)

"Thanks for all the fish" by Robert McGregor (2008)