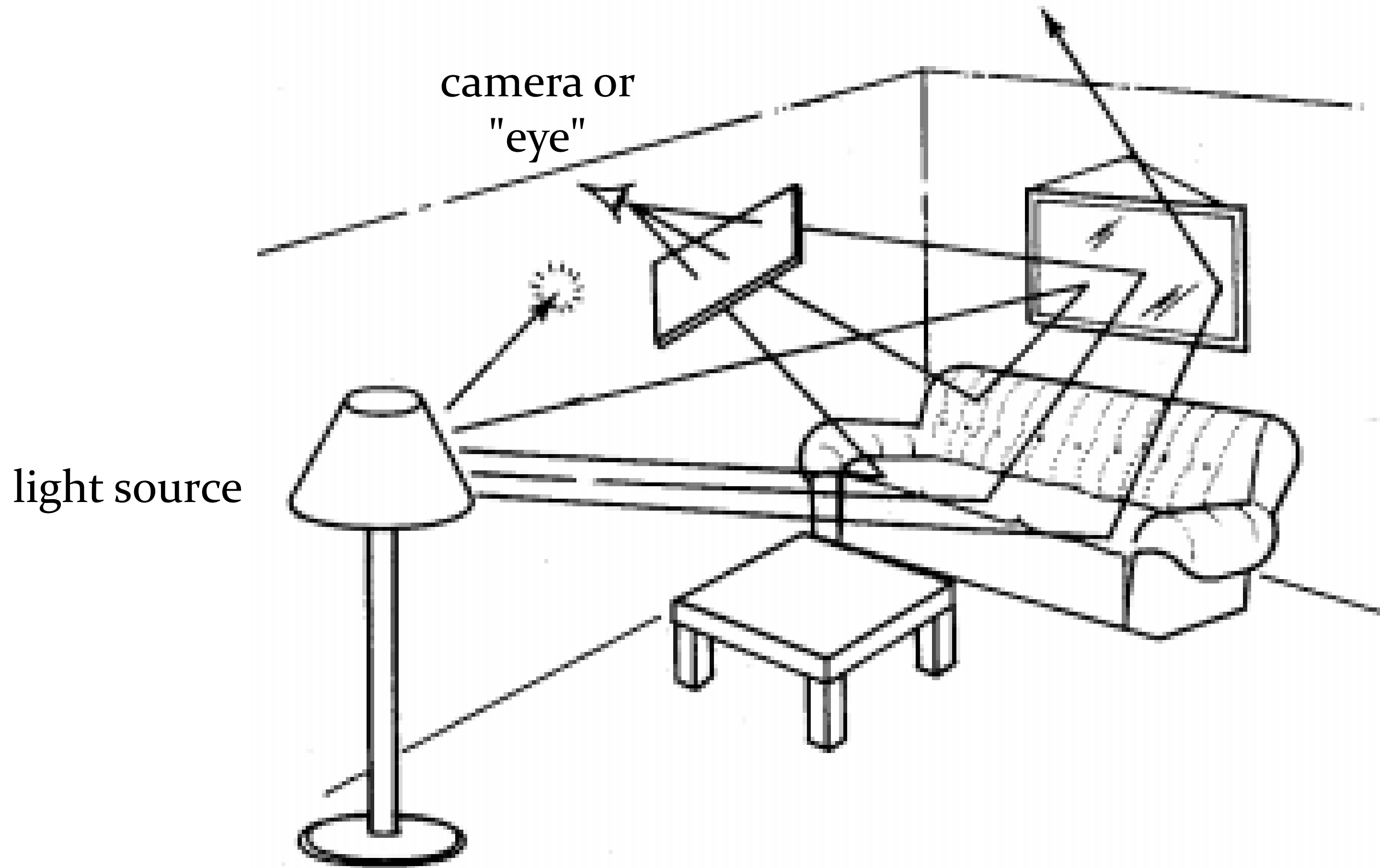
Recursion and Data Structures in Computer Graphics

Ray Tracing

Forward Ray Tracing

- imagine that you take a picture of a room using a camera
- exactly what is the camera sensing?
  - light reflected from the surfaces of objects into the camera lens

Forward Ray Tracing



Forward Ray Tracing

- forward ray tracing traces the paths of light from the light source to the camera to produce an image
- computationally infeasible because almost all of the possible paths of light miss the camera

Backward Ray Tracing

- backward ray tracing traces the paths of light from the camera out into the environment to produce an image
- computationally feasible because the process starts with a single* ray per screen pixel

Why Ray Tracing: Shadows

shadows with ~1000 light sources

Ray Tracing for the Movie 'Cars', P. Christensen et al

Ray Tracing: Reflections

Ray Tracing for the Movie 'Cars', P. Christensen et al

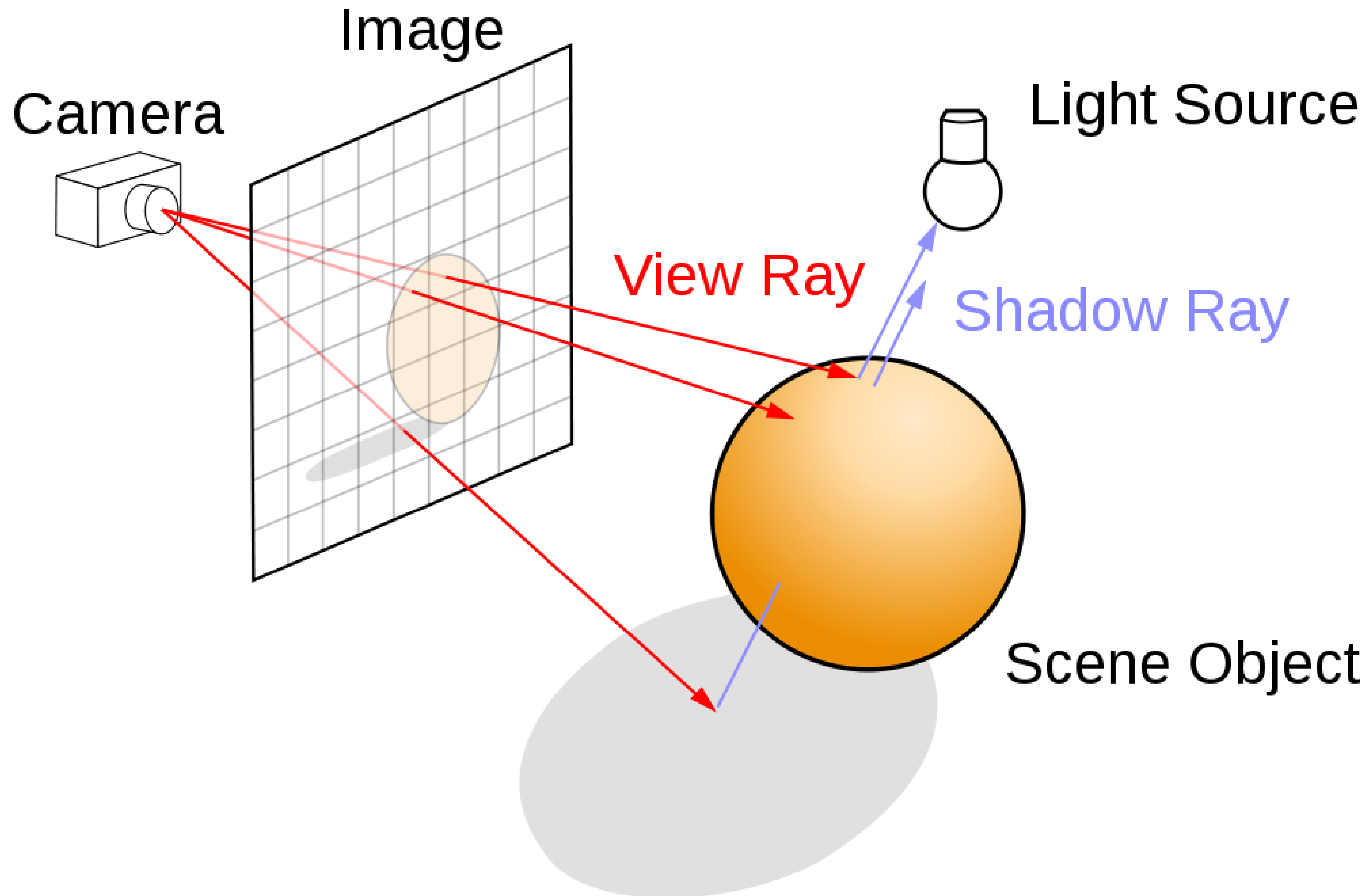
Ray Tracing: Reflections

Ray Tracing for the Movie 'Cars', P. Christensen et al

Comment on Previous Images

- most of the rendering in the previous images was not done using ray tracing
- ray tracing was only used on those parts of the image that would produce a noticeable difference

Backward Ray Tracing



http://en.wikipedia.org/wiki/File:Ray_trace_diagram.svg

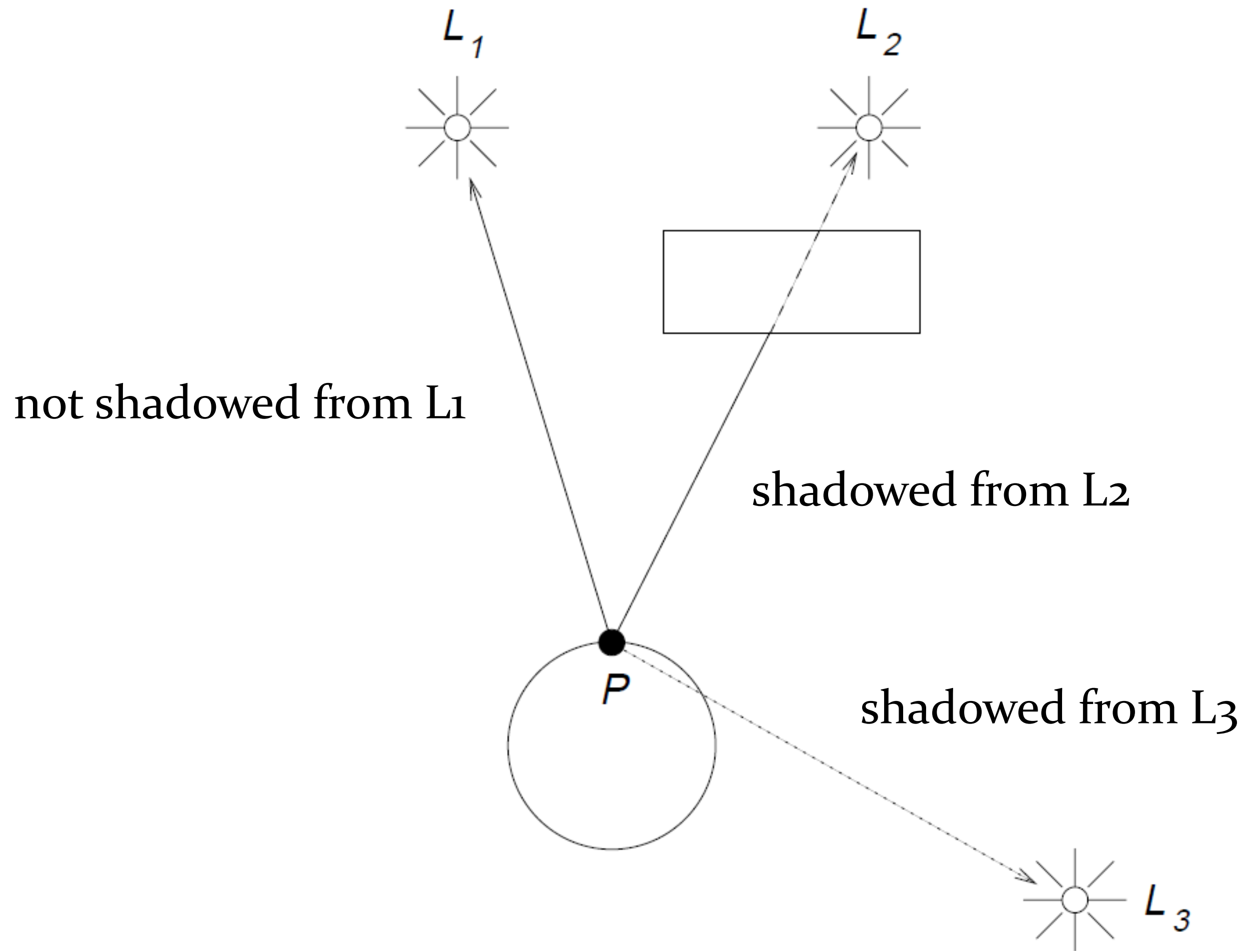
Pseudocode for recursive raytracing (missing base cases)

```
Color raytrace(ray) {
  if ray hits an object {
    color0 = object_color
    if object is shiny {
      color1 = raytrace(reflected_ray)
    }
    if object is transparent {
      color2 = raytrace(transmitted_ray)
    }
    return color0 + color1 + color2;
  }
  return background_color
}
```

Shadows

- we can determine if a point is in shadow by tracing rays from the point to each light source
  - called shadow rays
- if a shadow ray hits an object before it reaches the light source then the point is in shadow

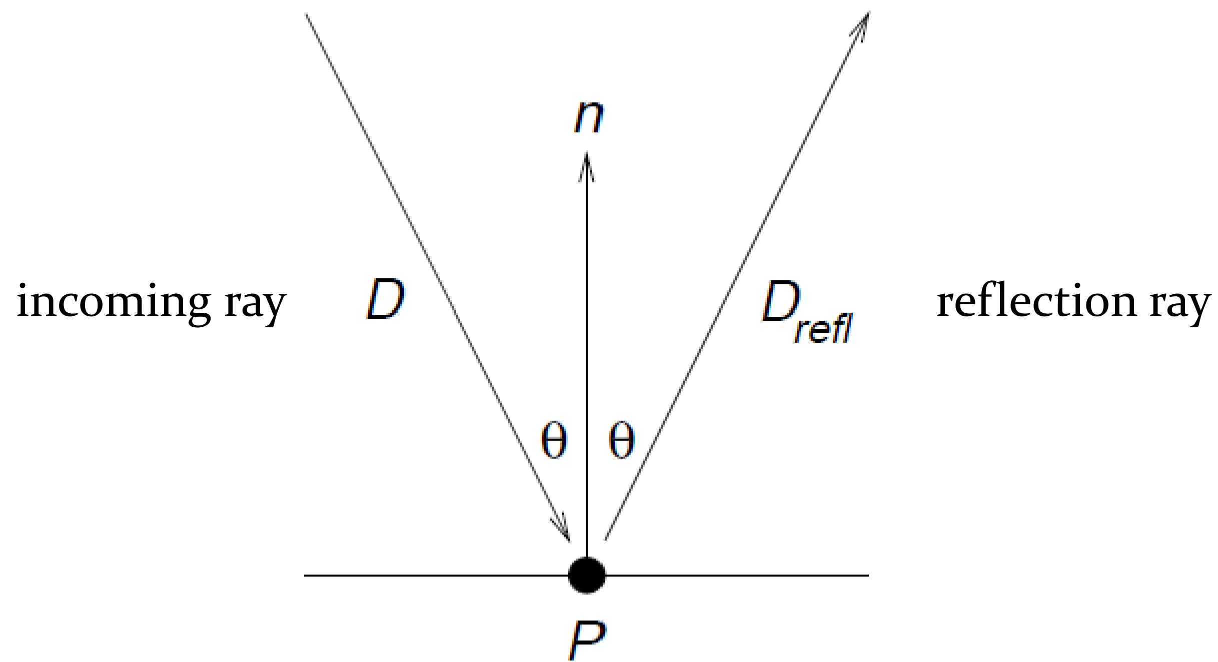
Shadows



Reflections

- if the ray hits a shiny object then we would like to know what reflection is seen at the hit point
- we can cast a new ray in the mirror reflection direction to determine the reflection

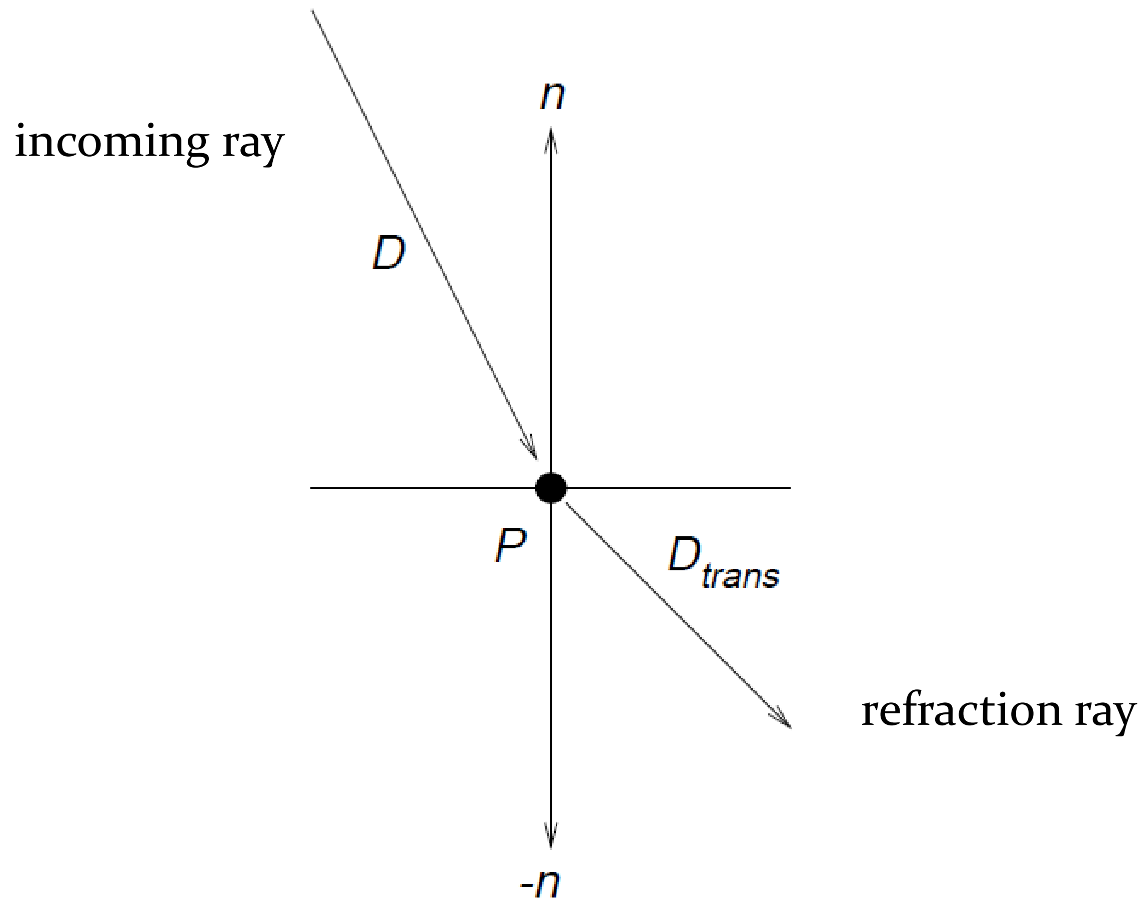
Reflections



Transparent Objects

- if the ray hits a transparent object then we would like to know what can be seen through the object
- we can cast a new ray in the refraction direction to determine what can be seen through the object

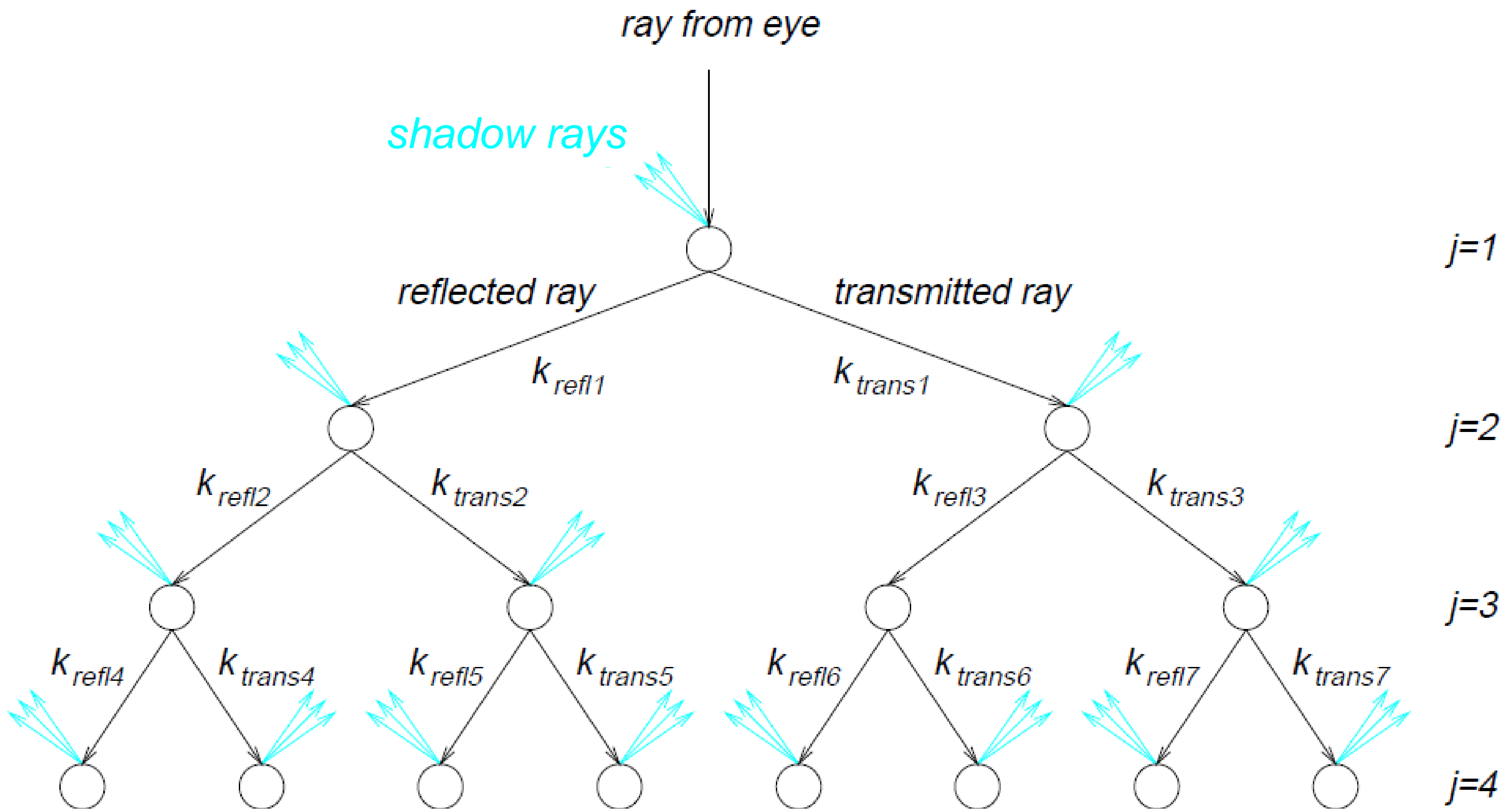
Transparent Objects



Recursion

- each reflected and refracted ray can be treated as a new view ray emanating from a hit point
  - i.e., we recursively trace the reflected and refracted rays

Ray Tracing as a Binary Tree



Stopping the Recursion

- what are the base cases?
  - ray misses all objects
  - level of recursion exceeds a fixed value
  - other cases outside the scope of EECS1030

How Fast is Ray Tracing

- approaching real time for non-cinematic quality, e.g.,
  - Brigade 2 game engine
  - NVIDA OptiX
    - demos here if you have a high-end NVIDIA graphics card

- cinematic quality is much slower

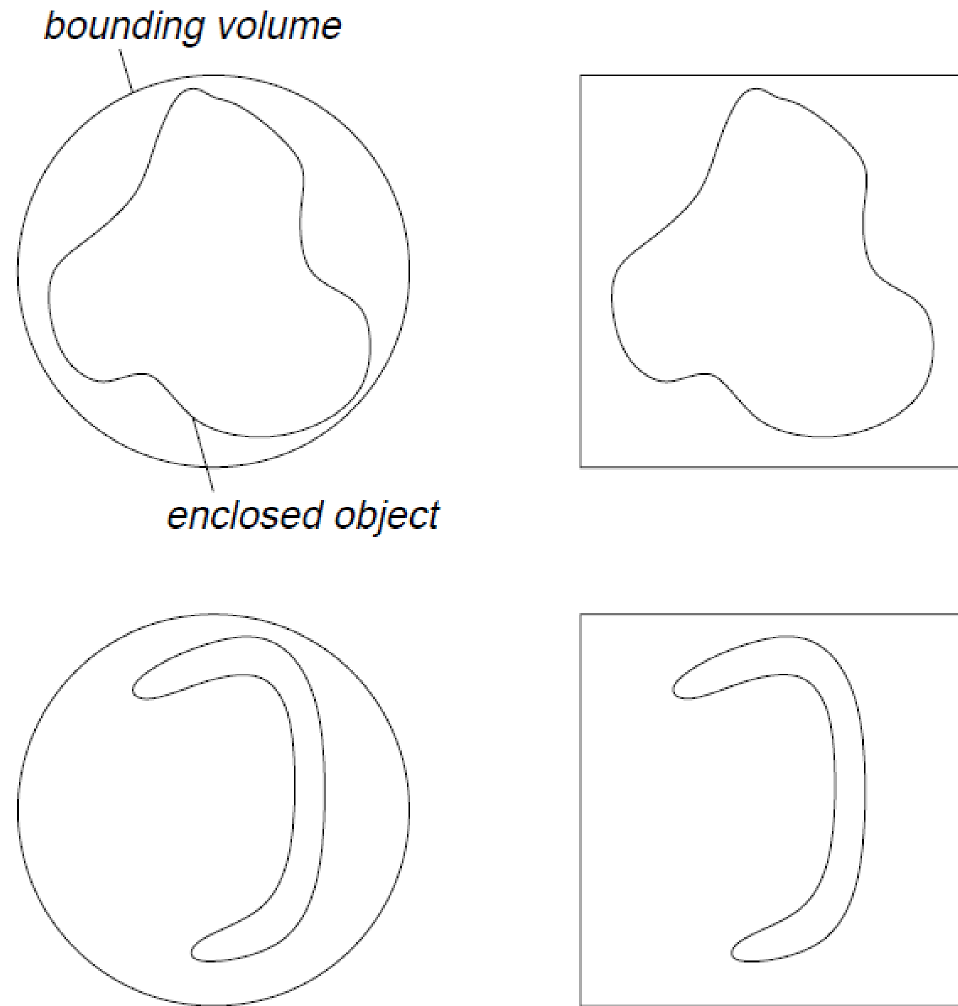
How Fast is Ray Tracing

- 678 million triangles
- rays
  - 111 million diffuse
  - 37 million specular
  - 26 million shadow
- 1.2 billion ray-triangle intersections
- 106 minutes on 2006 hardware

Bounding Volumes

- it is easy to compute the intersection of a ray with certain shapes, e.g.,
  - spheres and cubes
- it is hard or expensive to compute the intersection of a ray with arbitrary shapes
- idea
  - put complex shapes inside simple ones

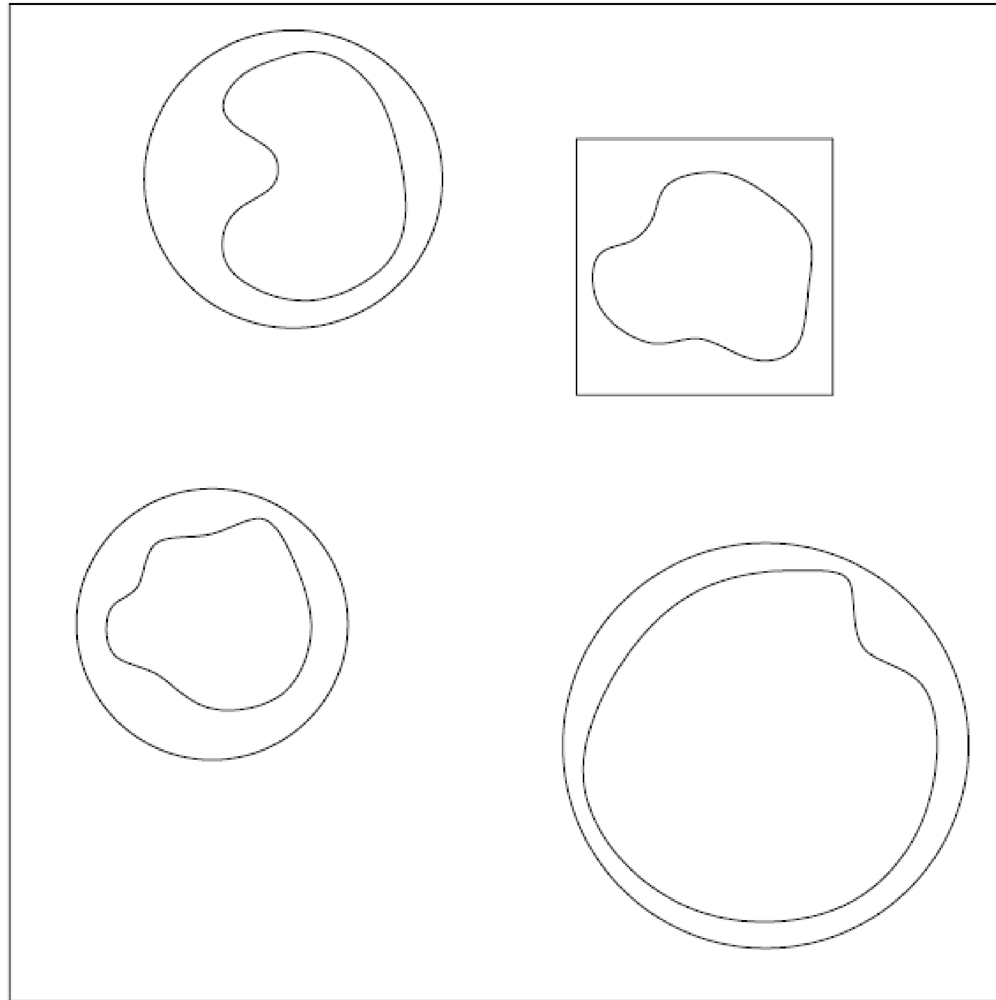
Bounding Volumes



Hierarchy of Bounding Volumes

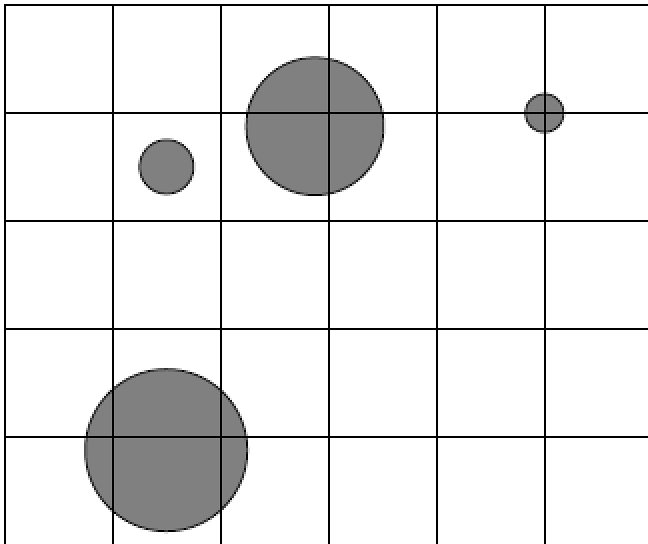
- why stop at putting complex shapes into bounding volumes?
- why not put bounding volumes inside bounding volumes?

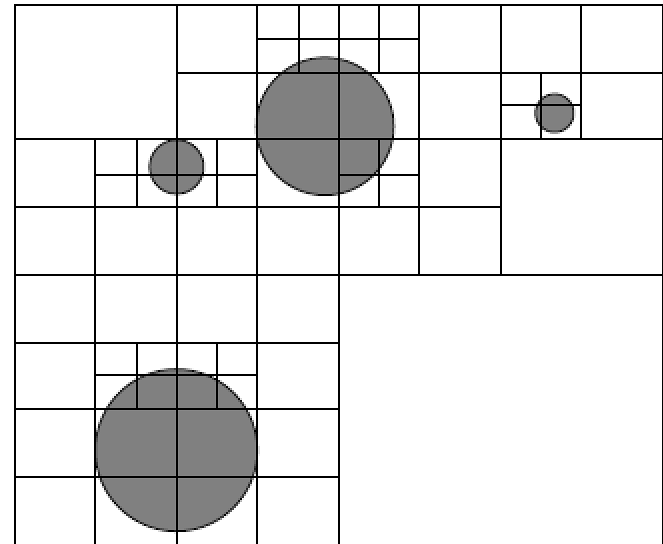
Hierarchy of Bounding Volumes



Spatial Subdivision

- instead of putting objects inside volumes we can subdivide space





Quadtree Decomposition

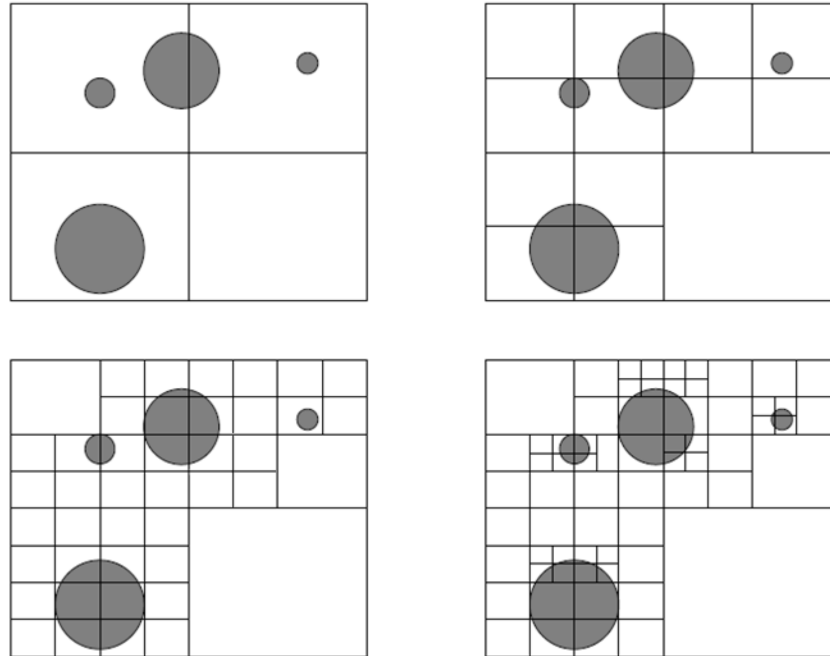
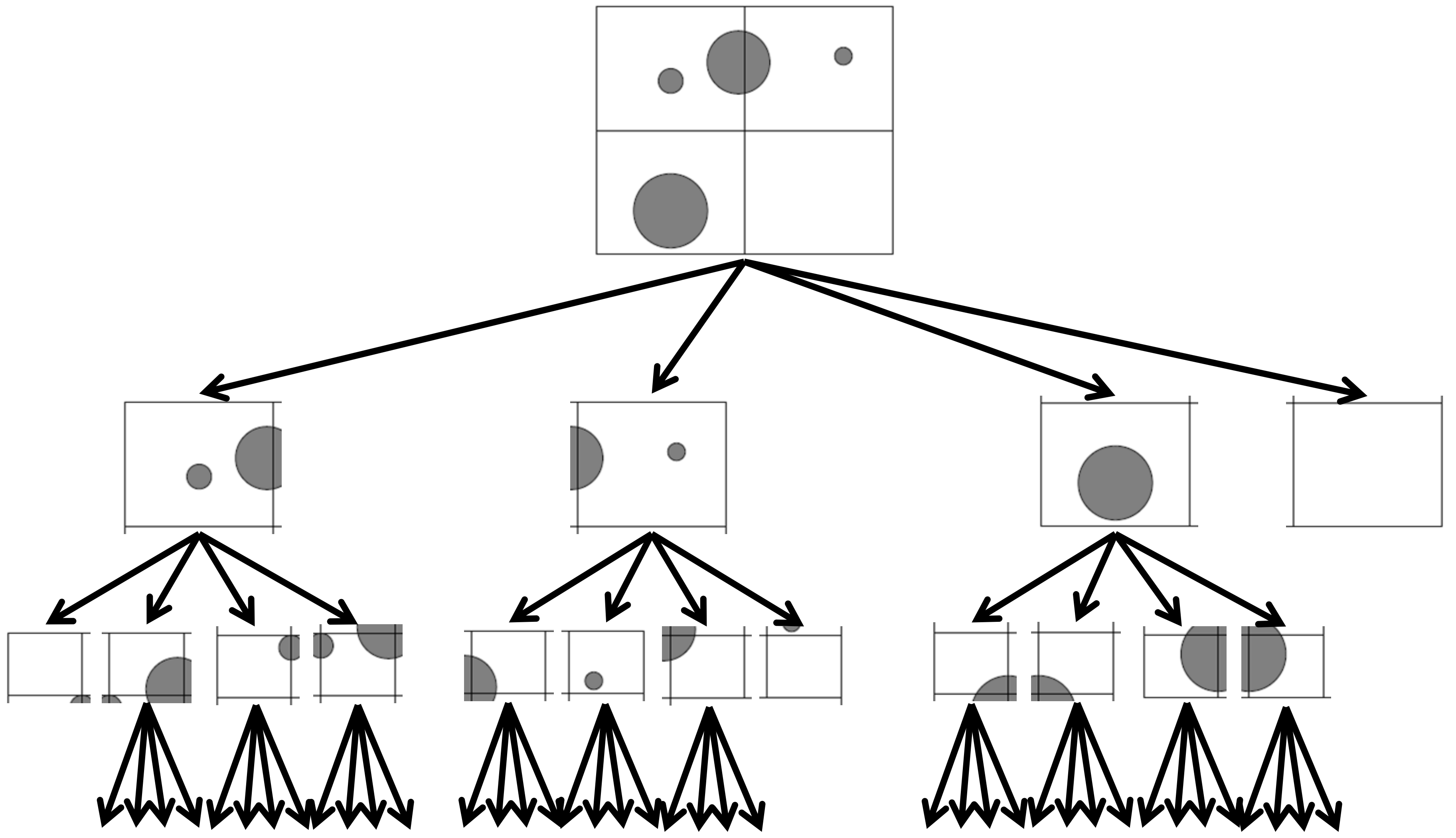


Figure 7.16: Quadtree decomposition of 2D space. Each quadrant of the world is recursively subdivided into subquadrants until the subquadrant meets some measure of simplicity. In this example, the recursive subdivision stops when then i) the subquadrant is occupied by zero objects or; ii) the object occupying the subquadrant takes up more than one-half the area of the subquadrant or; iii) the depth of recursion is four. A more common heuristic is to subdivide any quadrant that that is occupied by *n* or more objects.



and so on …

Using a Quadtree in Ray Tracing

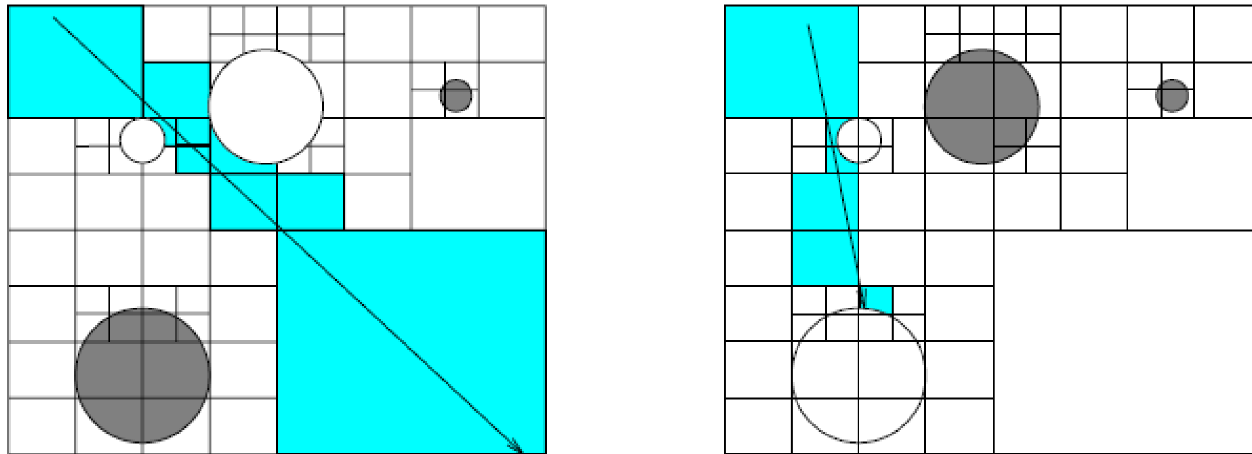


Figure 7.17: Using an octree to reduce the number of intersection calculations. On the left, nine voxels are examined and two objects are tested for an intersection with the ray. On the right, six voxels are examined and two objects are tested for an intersection with the ray. If an intersection is found, no more voxels need to be examined. This is because voxels are examined in the order that the ray passes through the world—the first intersection found must be the nearest hit point to the starting point of the ray.

Open Source Ray Tracers

- Art of Illusion
- POV-Ray
- YafaRay
- Manta
- several others