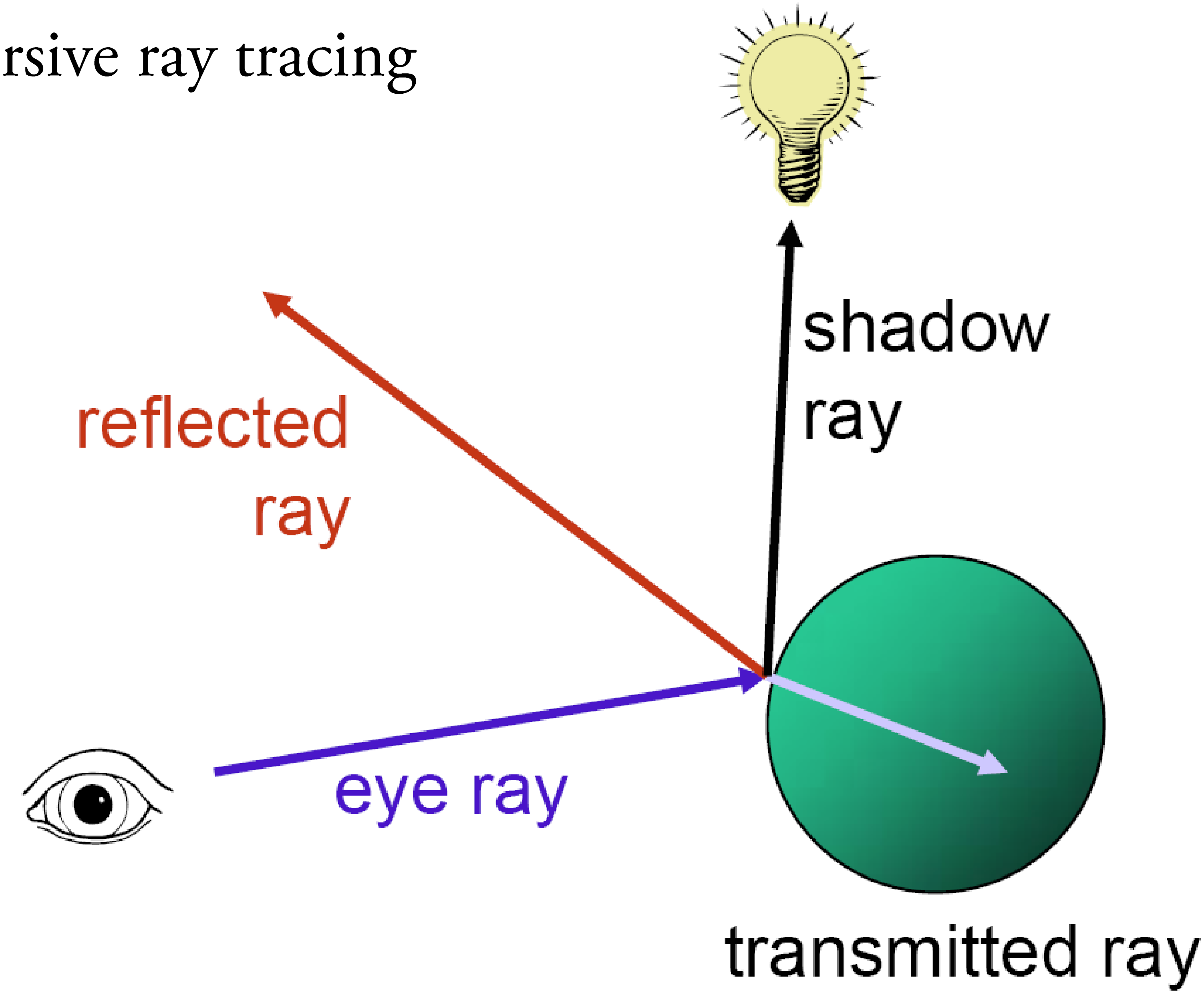
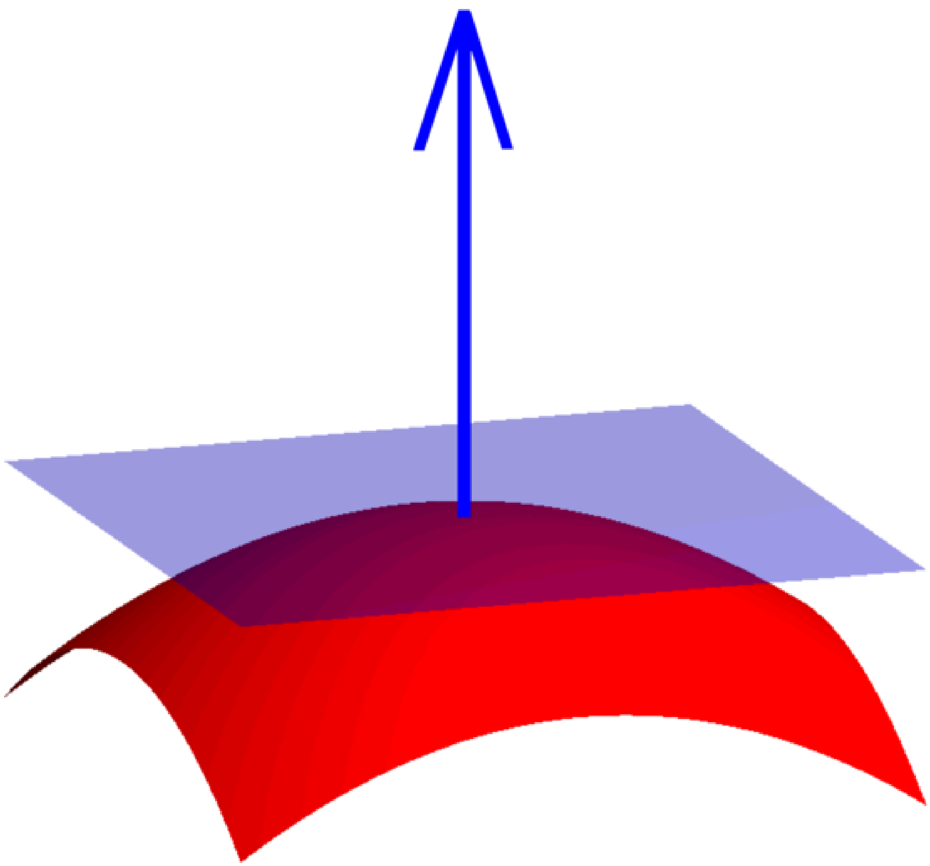
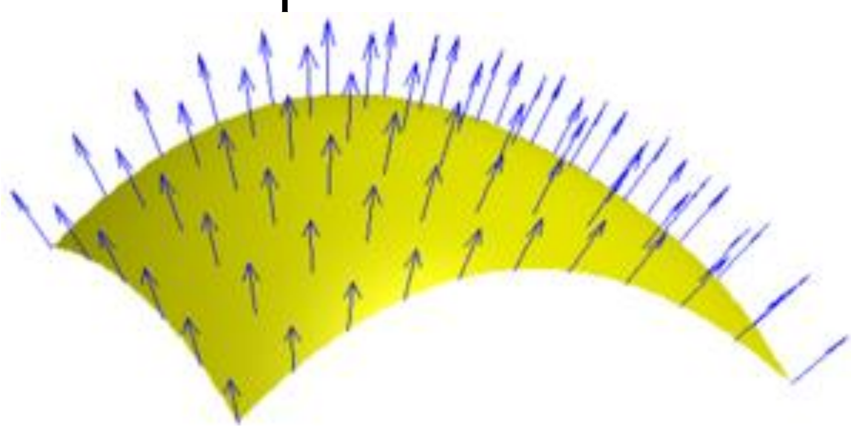
# Recursive ray tracing



# Surface Normals

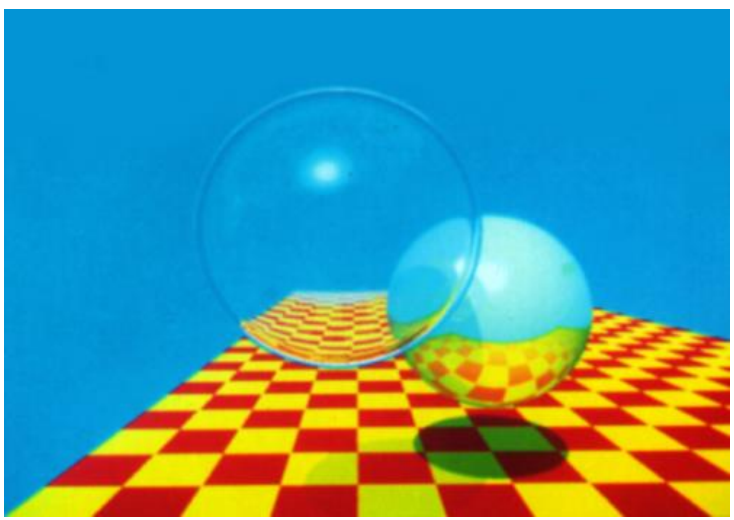
- A surface normal at a point is a vector from that point perpendicular to the tangent plane at that point
- Images from Wikipedia

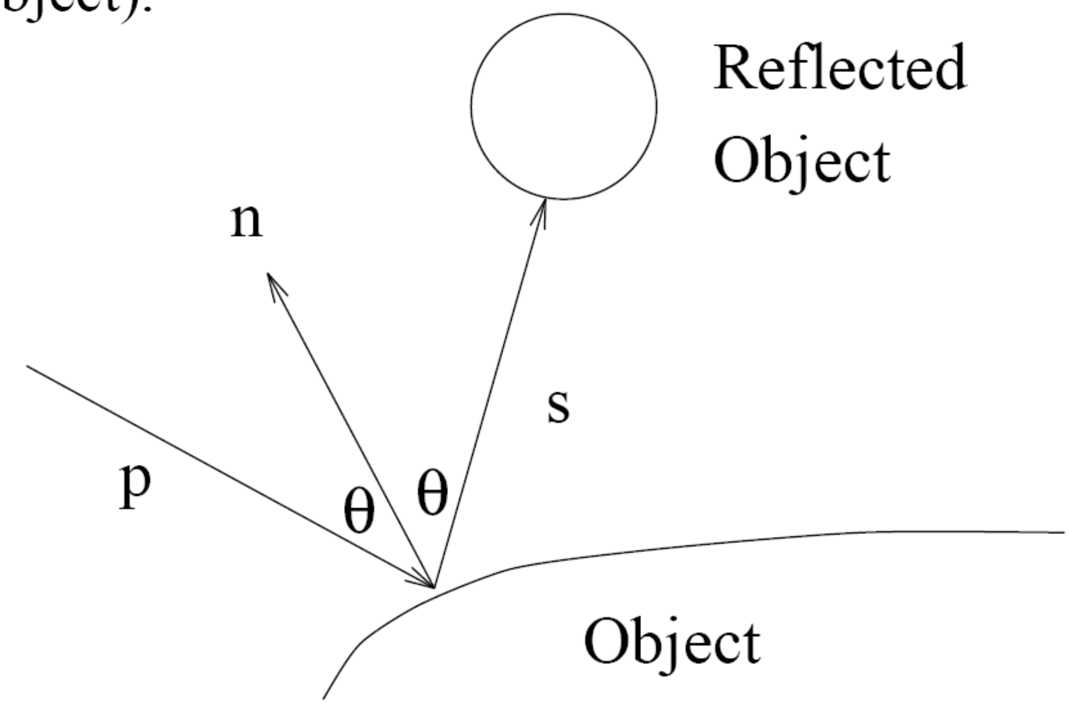




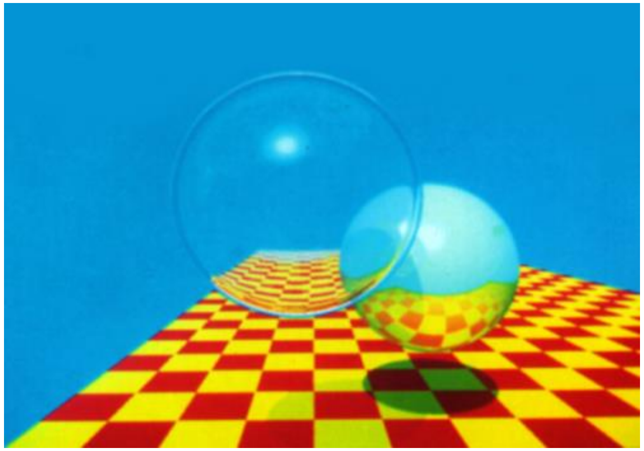
# Reflections

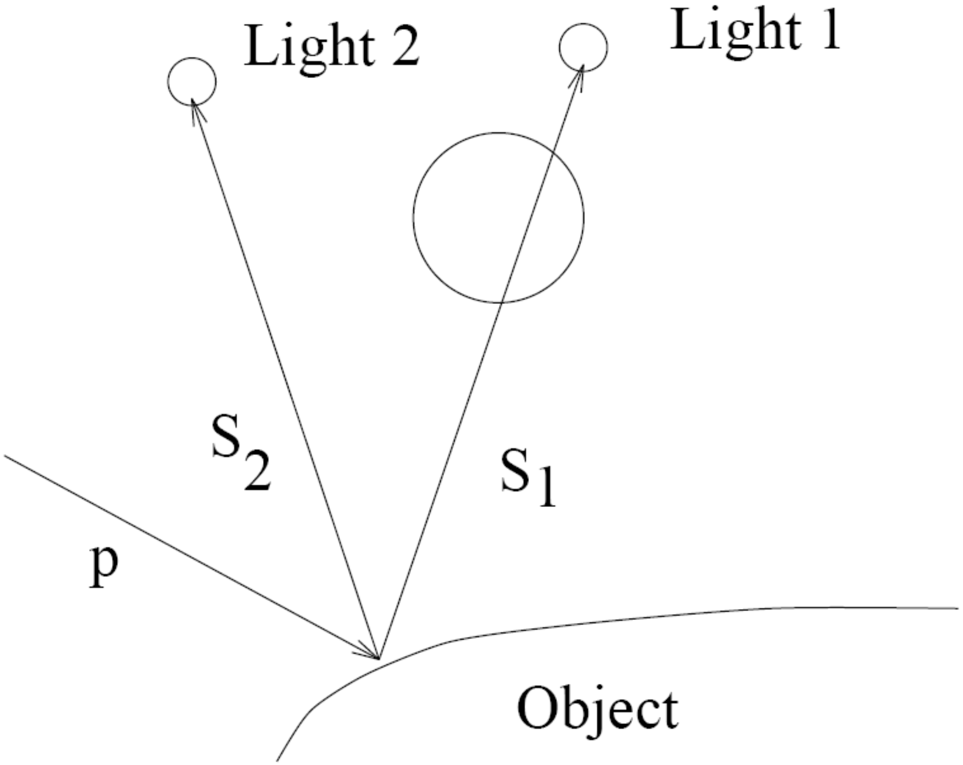
n = surface normal.
$\theta$ = angle of incidence (physics of reflection)
p = primary ray (from pixel).
s = secondary ray (from object).





# Shadows





# Transparency

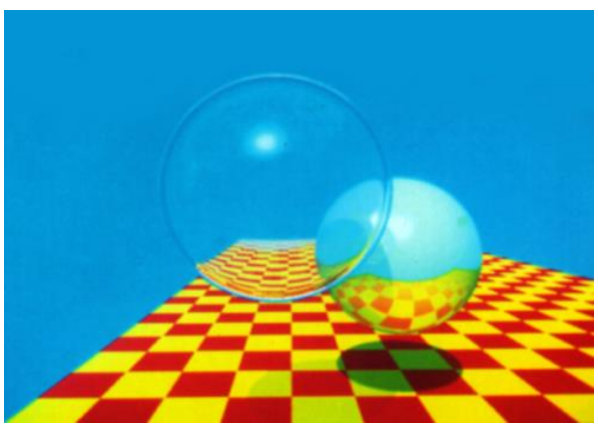
$\theta$ = angle of refraction (physics of light moving from one medium to another.)

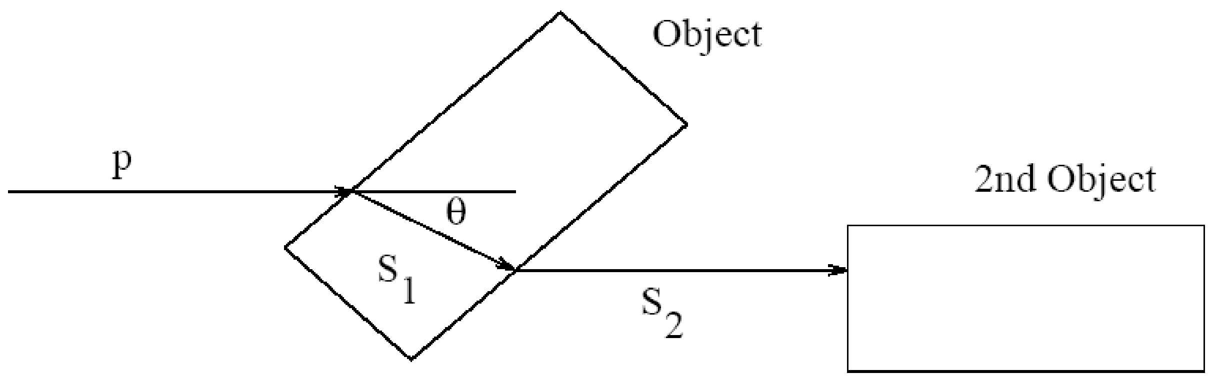
Snell's Law

p = primary ray.

$S_1$ = first secondary ray.

$S_2$ = second secondary ray.



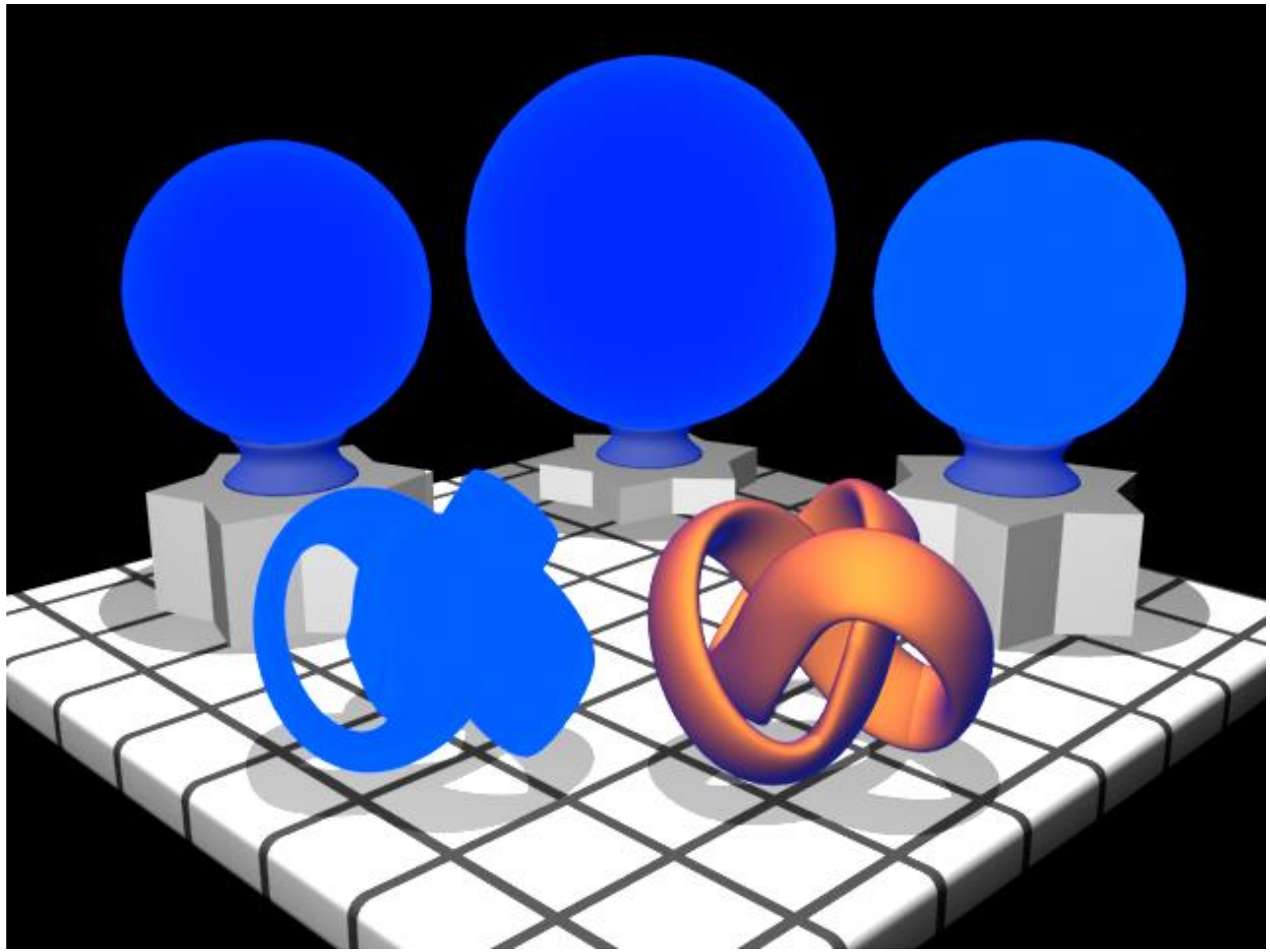


# Recursive Ray Tracing

- Each reflected or refracted ray is treated like a primary ray – i.e. can spawn shadow rays, reflected rays and refracted rays
- Recursive cut-off (e.g. what if a ray bounces between two mirrors – code does not terminate)
- How does that affect things?

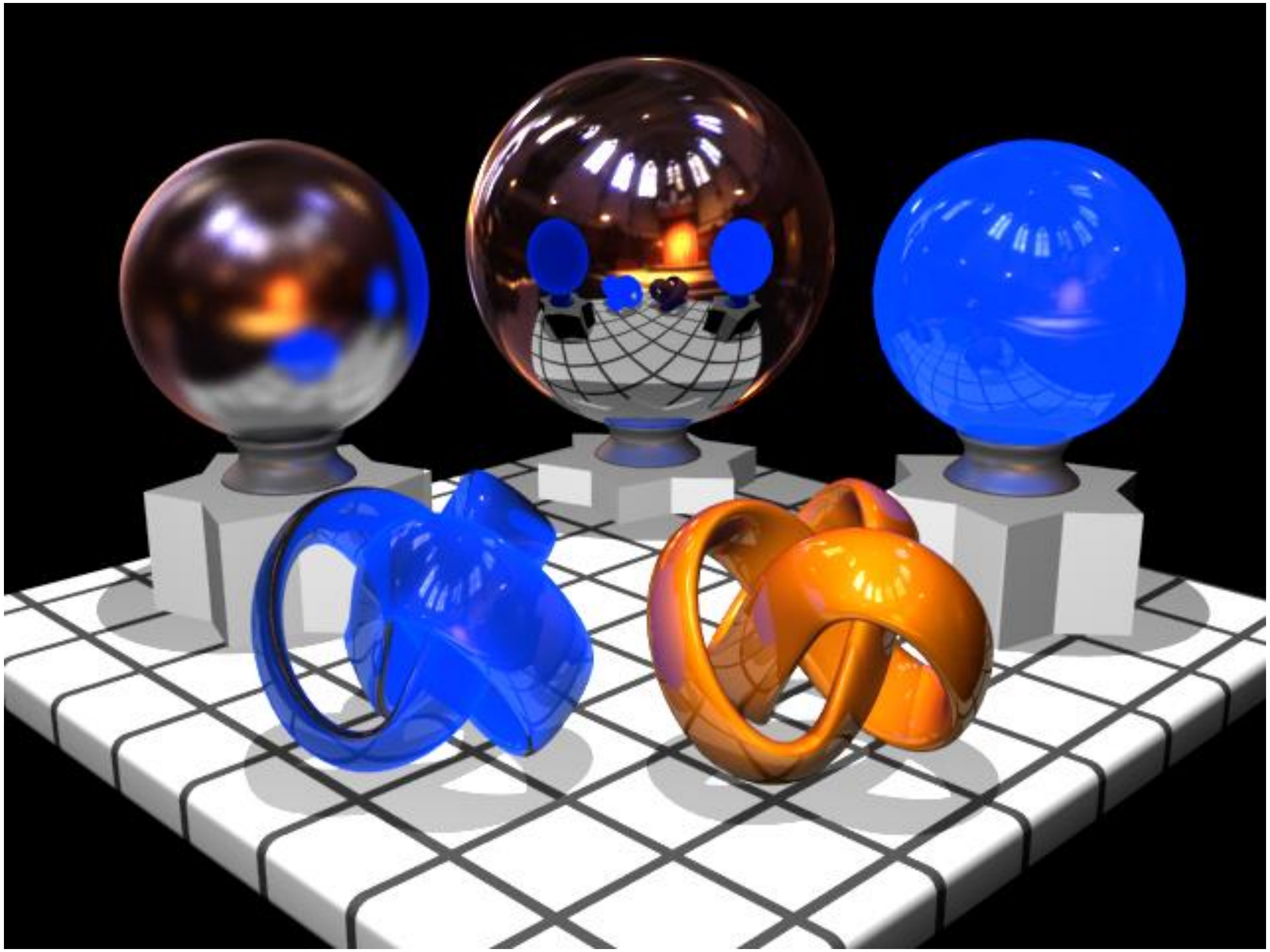
# Recursive Depth

- Primary rays (1)



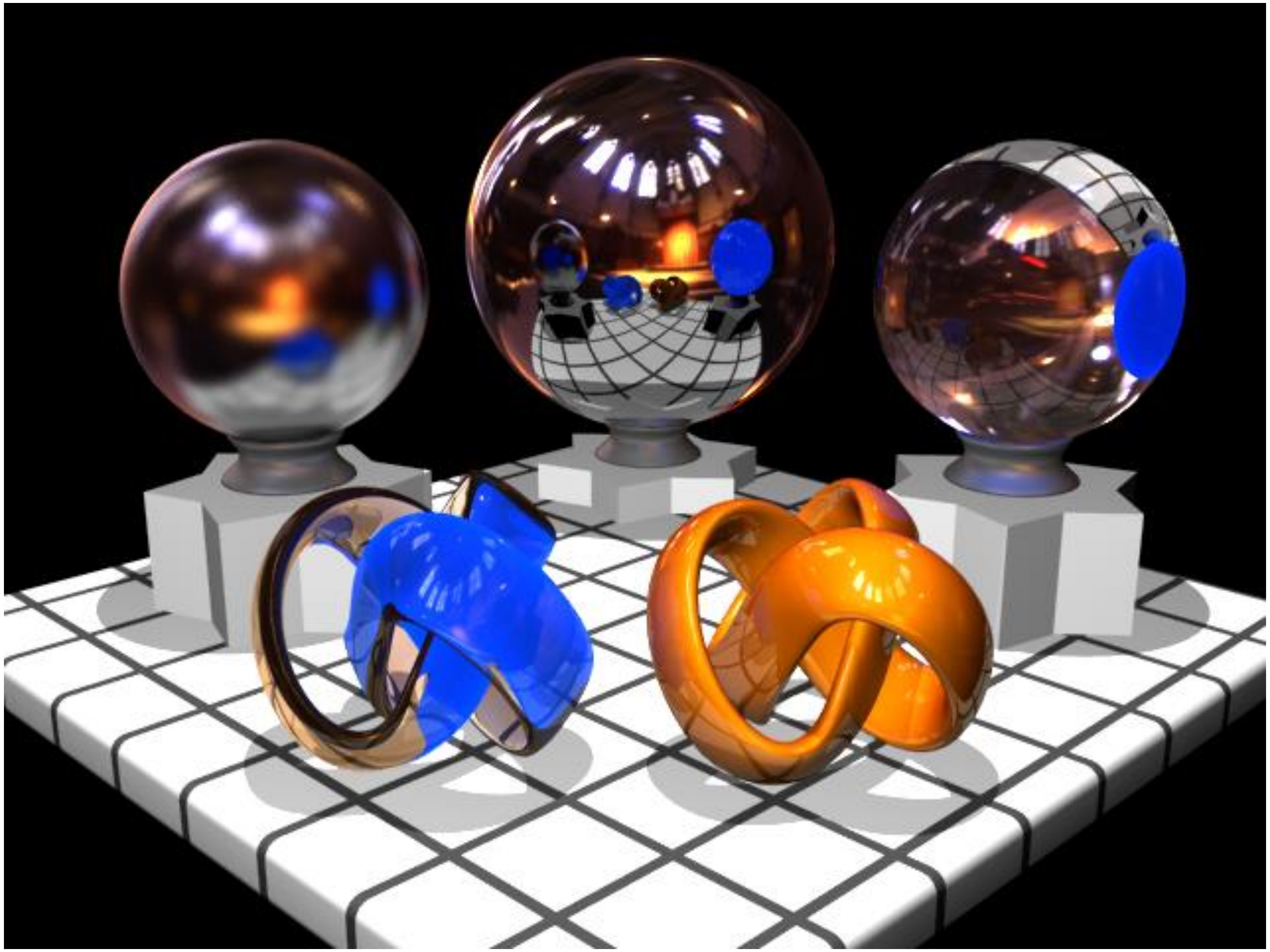
# Recursive Depth

- Depth= 2



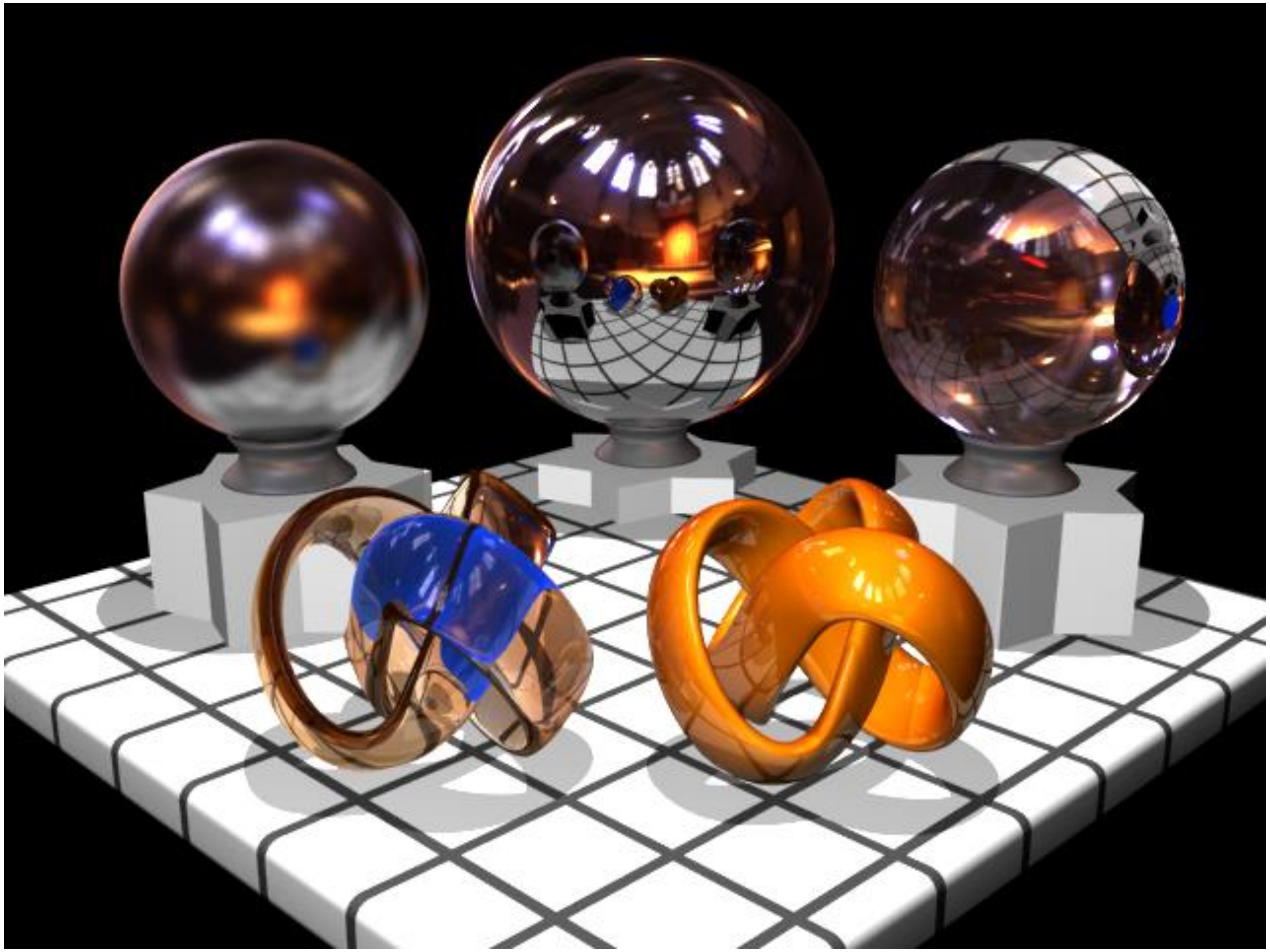
# Recursive Depth

- Depth=
3



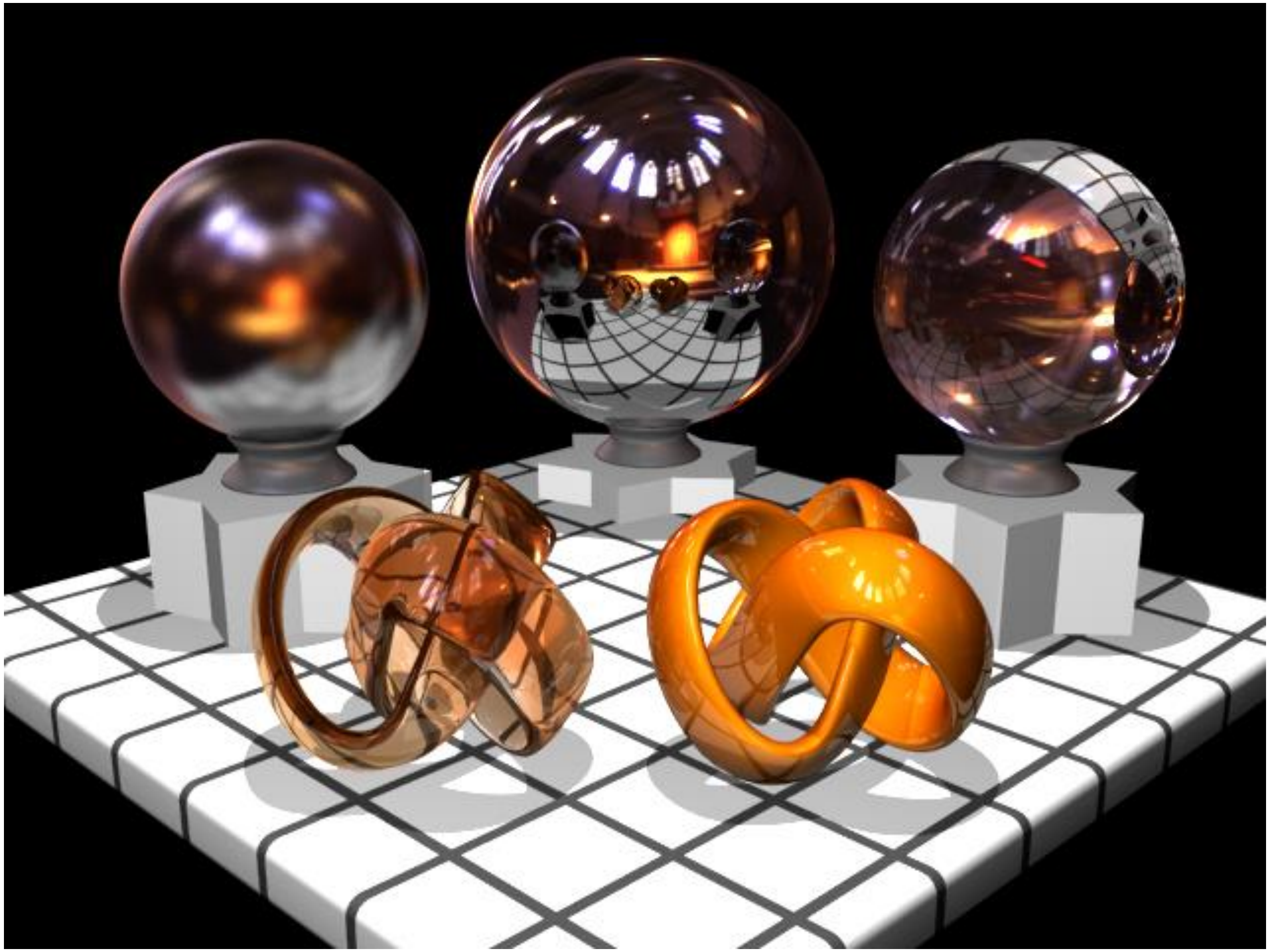
# Recursive Depth

- Depth= 5



# Recursive Depth

- Depth= 10

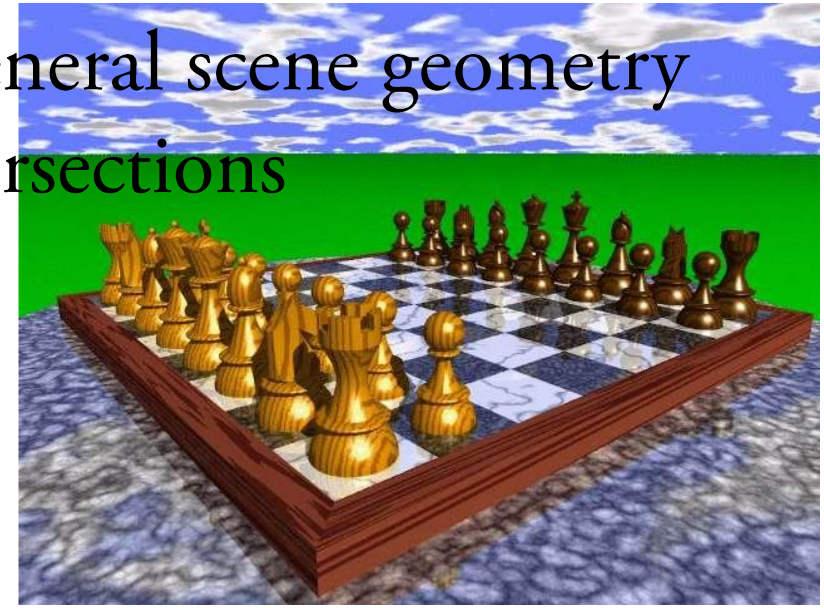


# Computational Expense

- No reflections, shadows or transparency
- Chess board – 32 pieces with 100 triangles each, extra 20 triangles for board, plane, sky etc. = 3220 triangles
- Image, 1000x1000=1 million pixels
- Algorithm – for each pixel, solve ray with each triangle. Select the closest hit
- 1000x1000x3220=3.22 billion intersection calculations!!

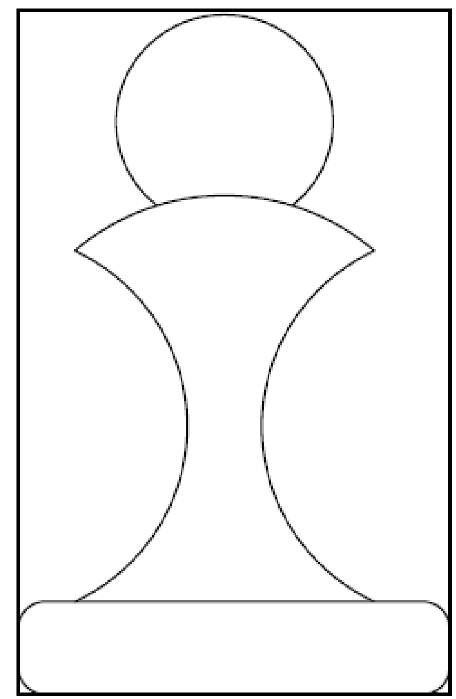
# Bounding Volumes

- Artificially surround each chess piece by a cylinder (easy to calculate ray-cylinder intersection)
- What does a ray hit?
- First trace it against the general scene geometry and 32 cylinders = 52 intersections



# Example

- Find closest hit point
- If the closest hit point is a cylinder, then intersect ray with 100 triangles within.
- Many rays will require just these 152 intersections
- Some will “miss” the chess piece although they hit the cylinder



# Example

- Assume 60% of rays miss cylinders, 30% hit a triangle in the first cylinder and 10% hit a triangle in the second cylinder.
- Therefore intersections ≈
  0.6x1000x1000x52
  0.3x1000x1000x152+
  0.1x1000x1000x252=102 million (about 30 times faster in this case)
- Why stop there? – hierarchical bounding volumes

# Hierarchical bounding volumes

- 1 box around the whole of the chess board and pieces (= 6 intersection tests)
- 3 other planes for the sky and two ground textures (=3 intersection tests)
- Inside the chess box, place 2 boxes – one over the white pieces, and one over the black (each 6 tests)
- Inside those have the 32 cylinders (1 test)
- Inside each cylinder have 3 cylinders (3 tests)
- Inside each of those have 30-40 triangles (30-40 tests)

# Hierarchical bounding volumes

- At the first level of our hierarchy, 9 tests are used to see if we have hit sky, ground or are in the region of the board.
- If we hit board region, another 6 tests see if we have hit white or black chess pieces
- If we have hit white, another 16 tests see if we have hit a chess piece
- Another 3 determine which part
- And a maximum of 40 to determine the triangle
- Total for a hit chess piece (previously 152) = 74

# Example

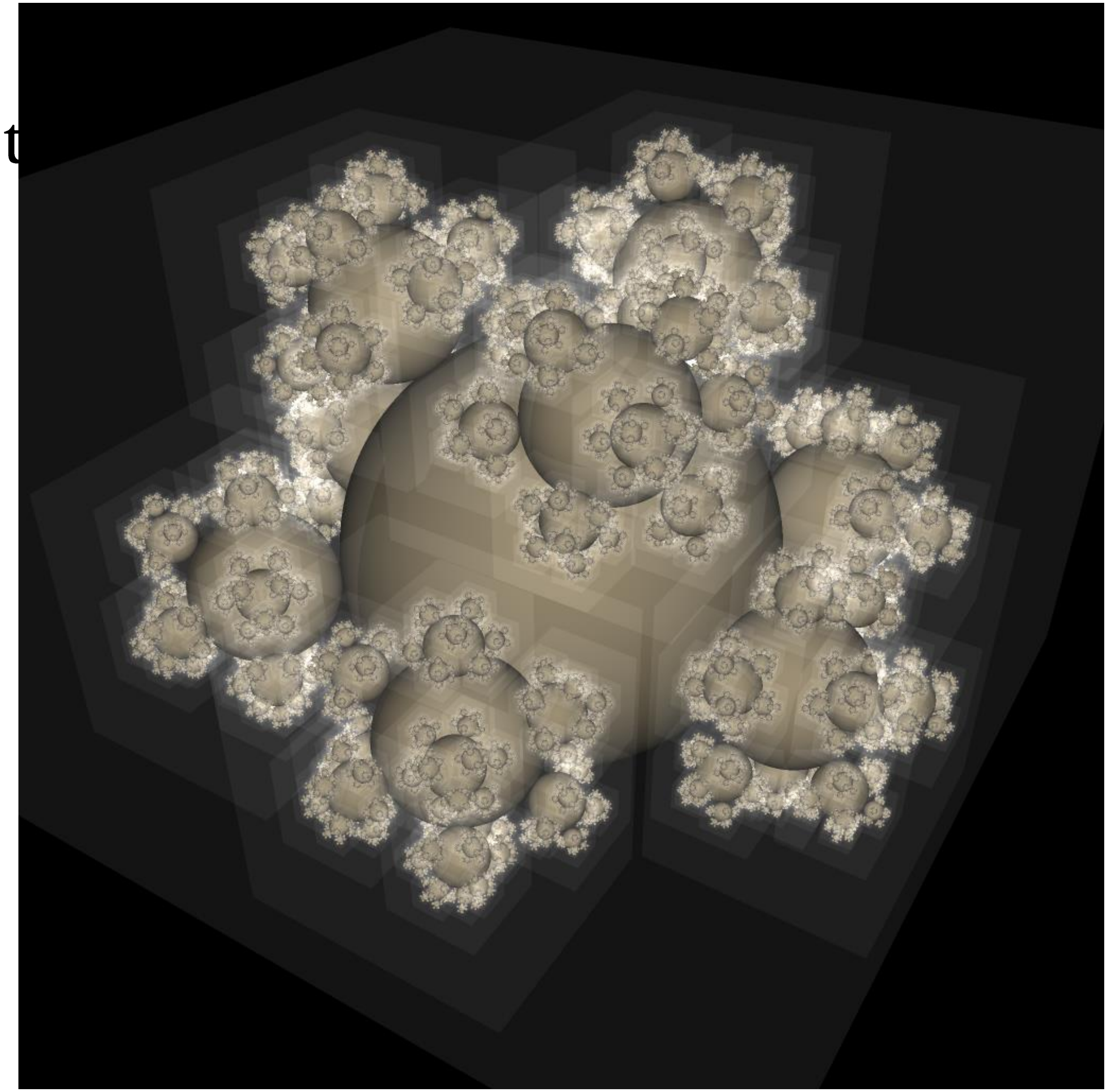
- Using same assumptions as before: 60% of rays miss board, 30% hit a triangle in the first cylinder and 10% hit a triangle in the second cylinder.
- Therefore intersections ≈
  0.6x1000x1000x9
  0.3x1000x1000x74+
  0.1x1000x1000x117=39.3 million (about 100 times faster in this case)

# Hierarchical Bounding Volumes

- Advantages
  - Fast ray tracing times
  - User can decide on tight fitting objects
  - User can decide logical splitting of scene
- Disadvantages
  - User interaction required
  - Could take a lot more time than is saved
  - Not all scene modellers will understand BVs

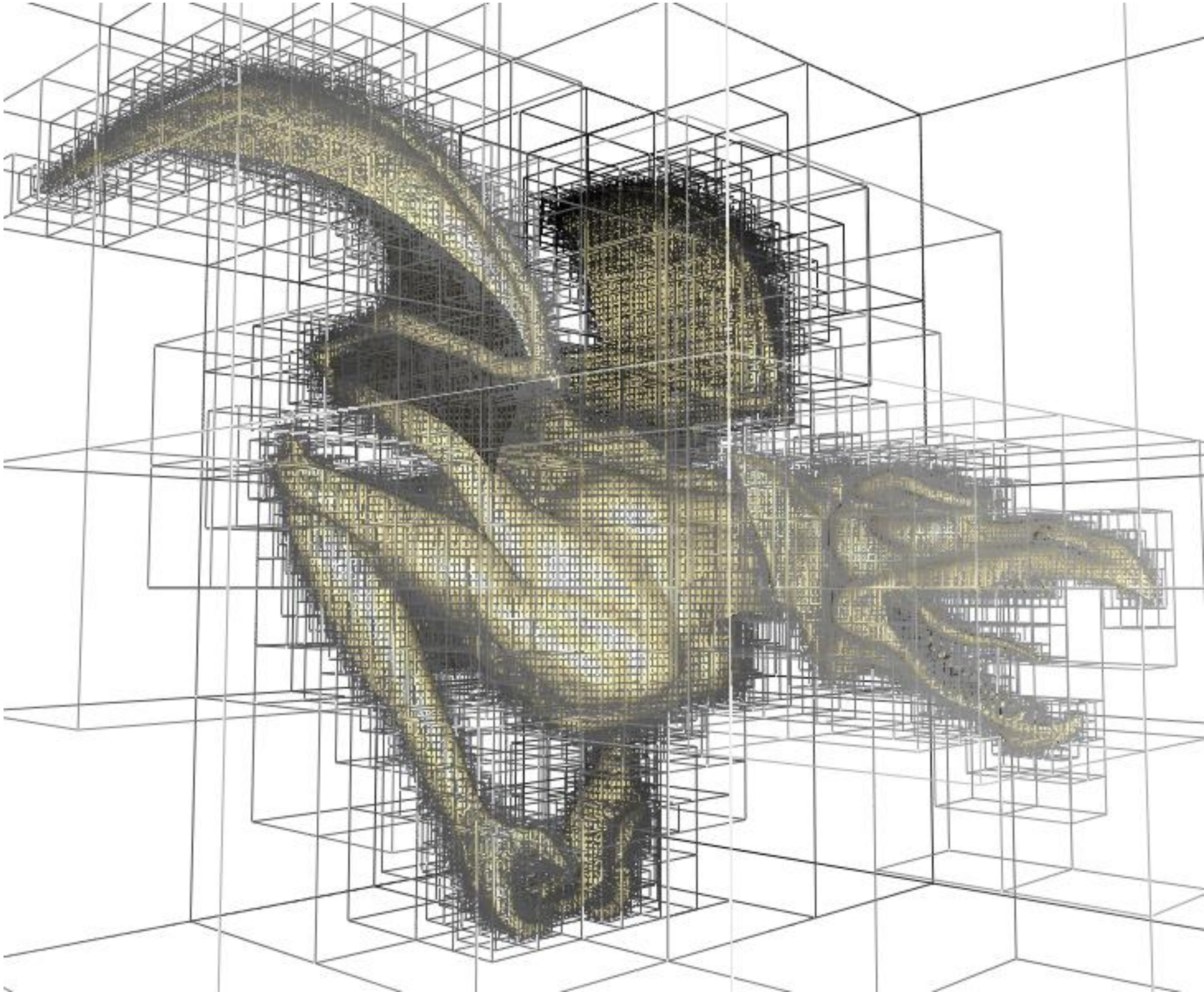
# Hierarchical Bounding Volumes

- Andreas Diet

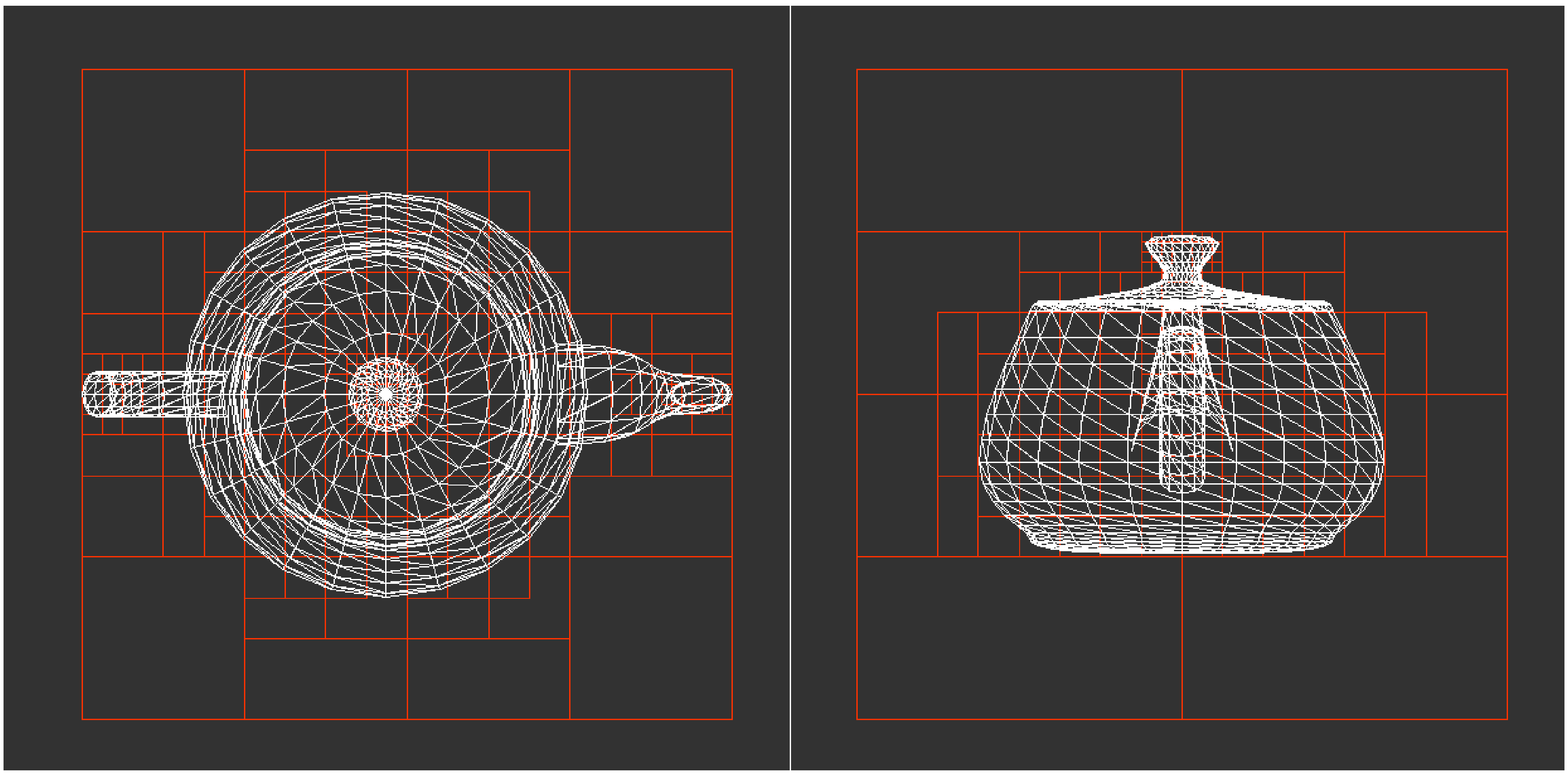


# Octrees

- Image from Sylvian Lefebvre
- (max depth= 12)



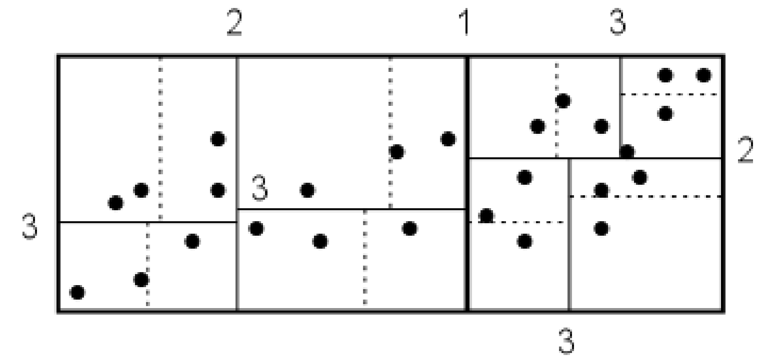
# Octrees



# Others

- kd-trees (cut is not necessarily in the centre)
- bsp-trees (cut is a plane – not a cube)

**kD-Trees**




Split longer dimension near data median