Computer Graphics (CS 543)
Lecture 13c
Ray Tracing Overview

Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

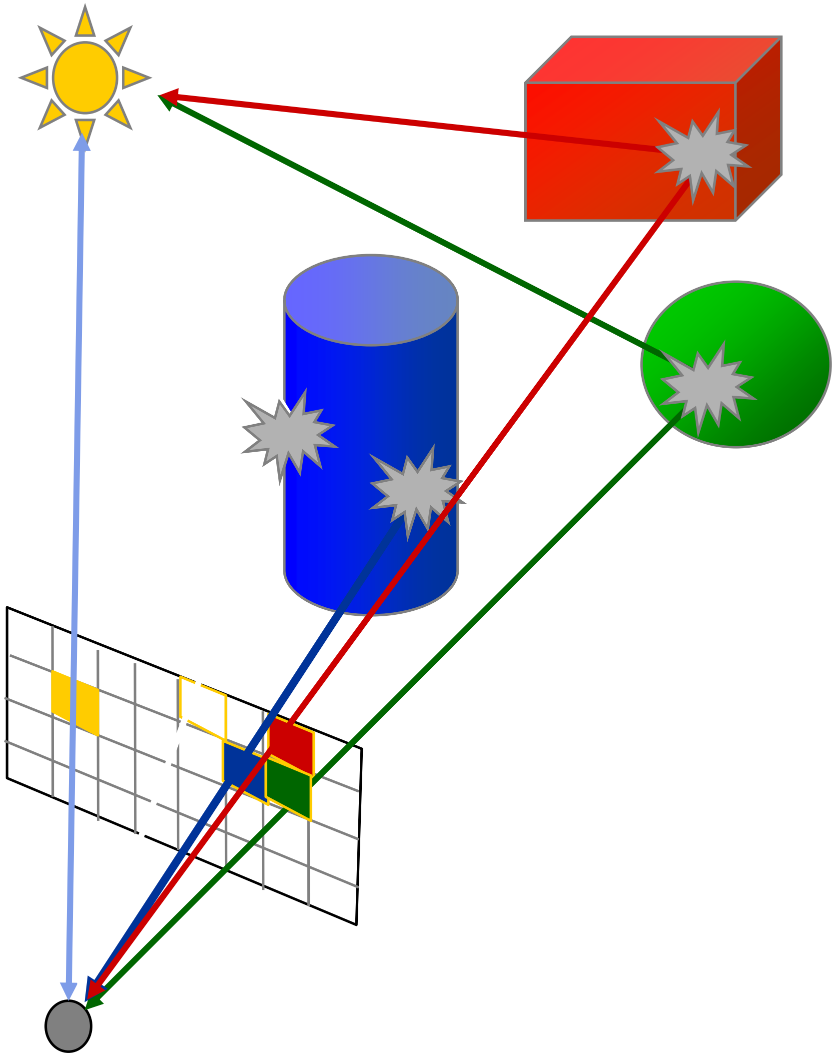
Raytracing

- Global illumination-based rendering method
- Simulates rays of light, natural lighting effects
- Because light path is traced, handles effects tough for openGL:
  - Shadows
  - Multiple inter-reflections
  - Transparency
  - Refraction
  - Texture mapping
- Newer variations… e.g. photon mapping (caustics, participating media, smoke)
- **Note:** raytracing can be semester graduate course
- Today: start with high-level description

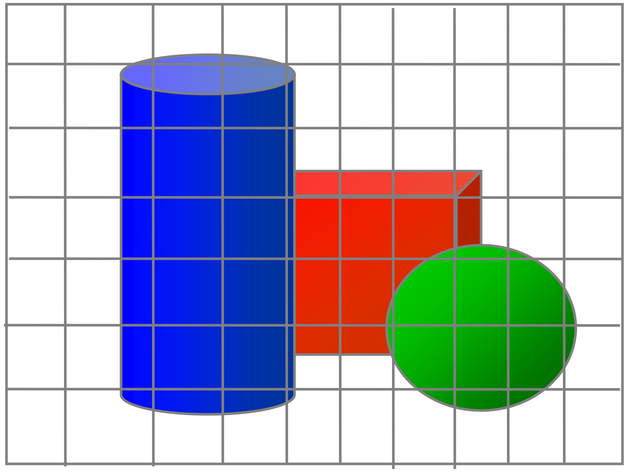
Raytracing Uses

- Entertainment (movies, commercials)
- Games (pre-production)
- Simulation (e.g. military)
- Image: Internet Ray Tracing Contest Winner (April 2003)

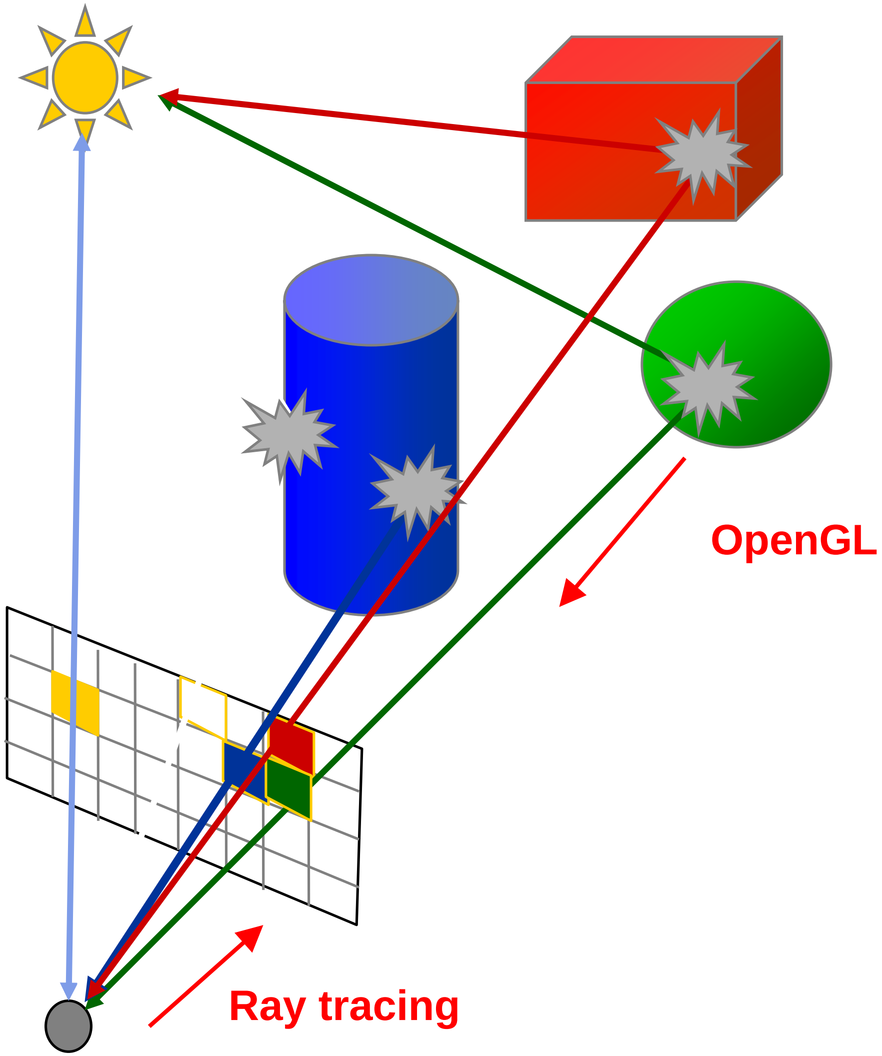
Ray Casting (Appel, 1968)



direct illumination (One bounce)
OpenGL does this too



Ray Tracing Vs OpenGL

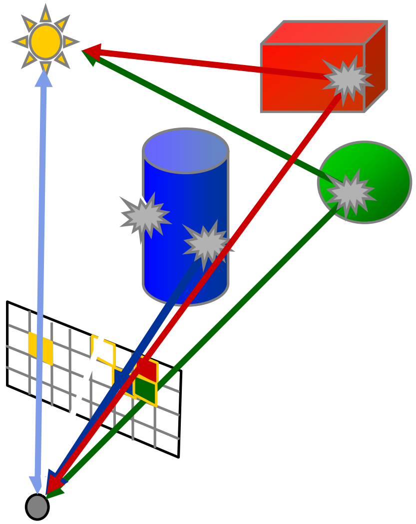


- OpenGL is object space rendering
  - start from world objects, transform, project, rasterize them

- Ray tracing is image space method
  - Start from pixel, what do you see through this pixel?

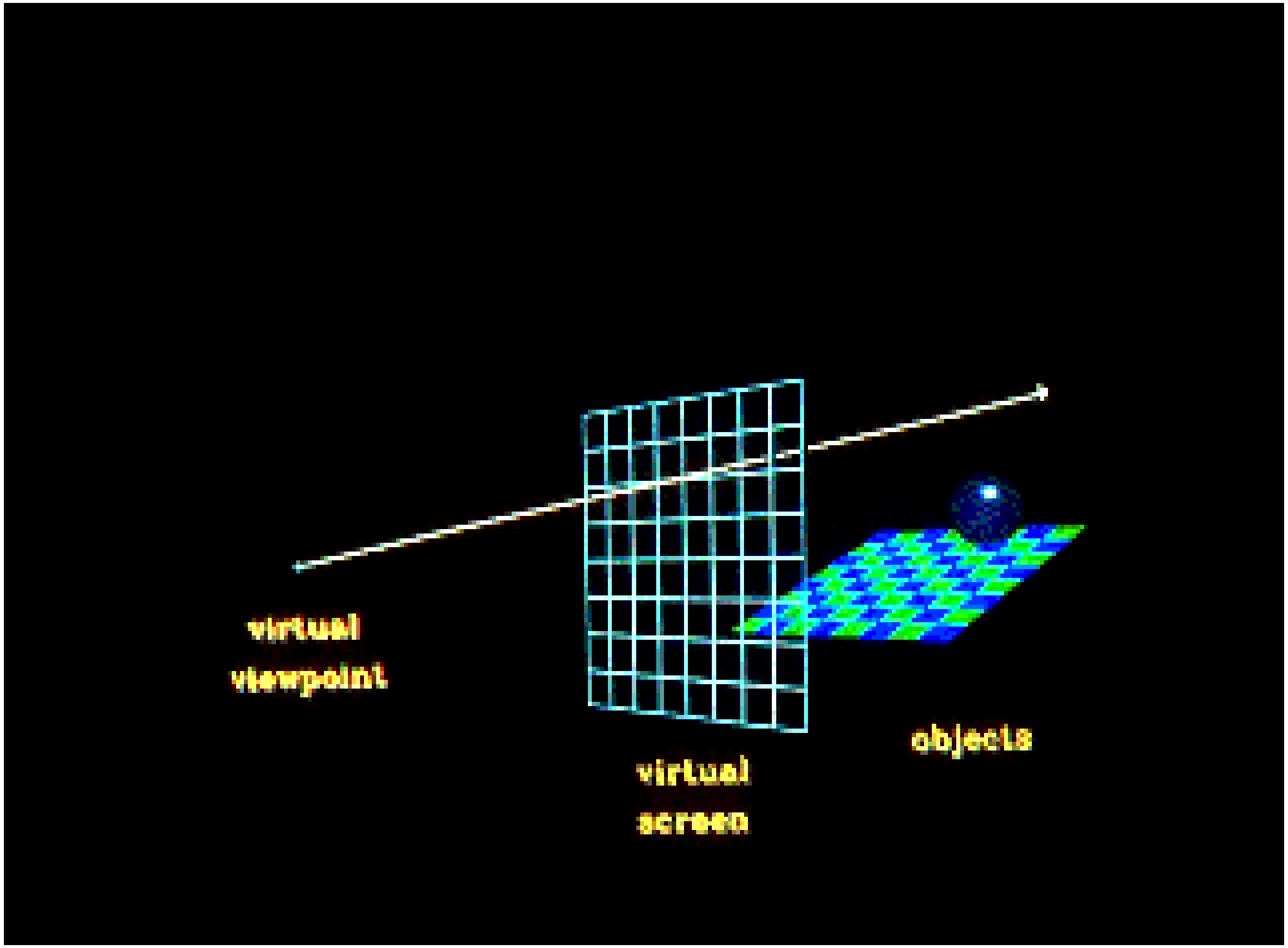
How Raytracing Works

- Looks through each pixel (e.g. 640 x 480)
- Determines what eye sees through pixel



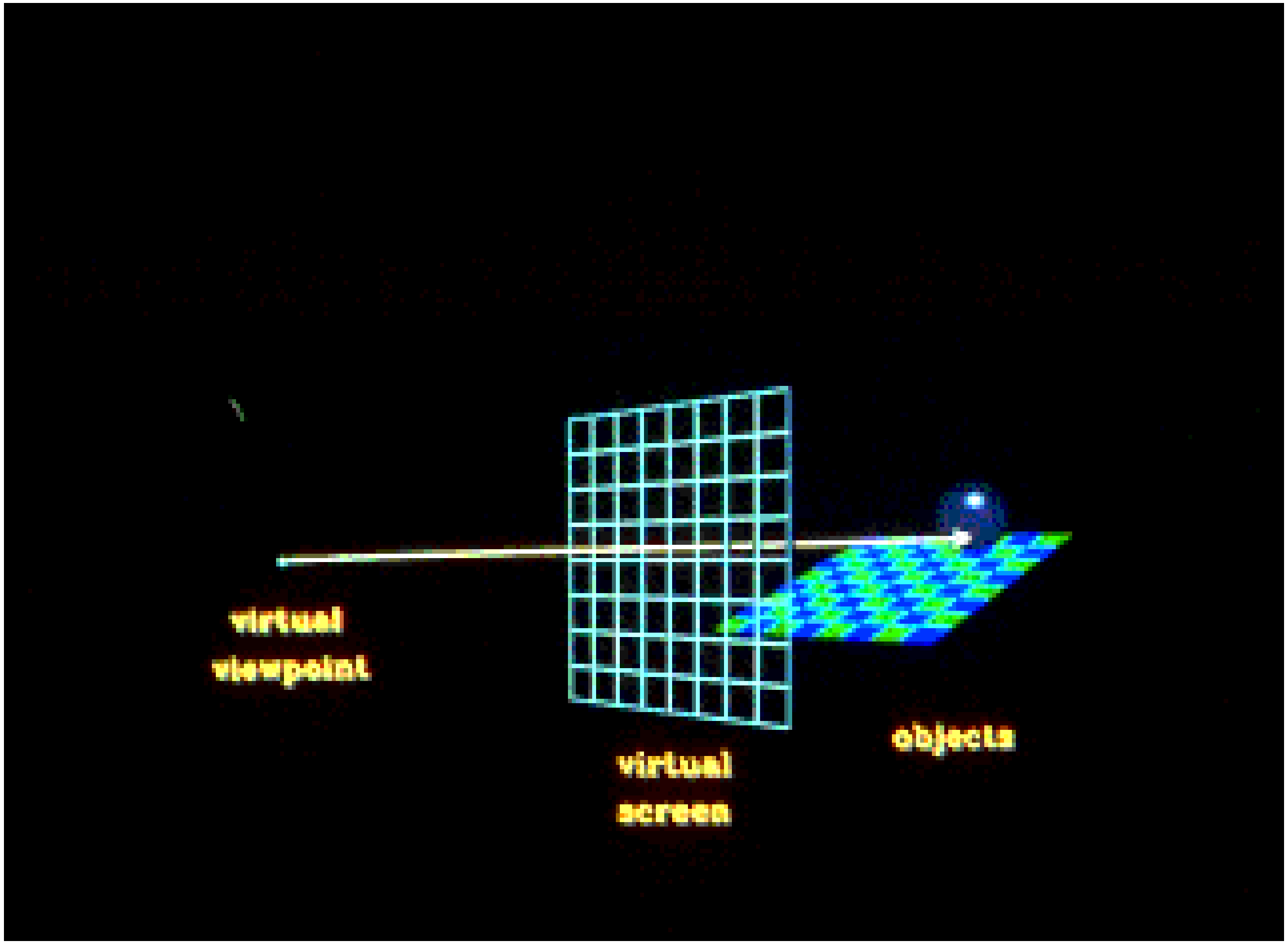
- Basic idea:
    - Trace light rays: eye -> pixel (image plane) -> scene
    - Does ray intersect any scene object in this direction?
        - Yes? Render pixel using object color
        - No? Renders the pixel using the background color
- Automatically solves hidden surface removal problem

Case A: Ray misses all objects



Render pixel using Background color

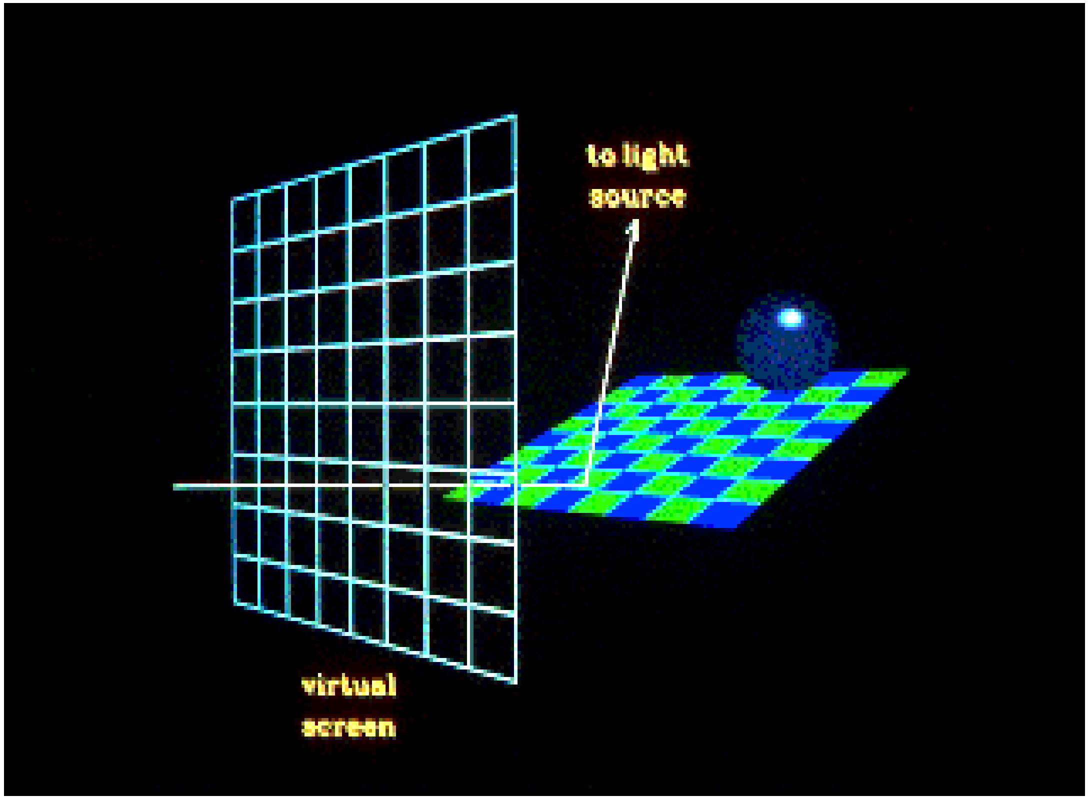
Case B: Ray hits an object



Render pixel using Object's color

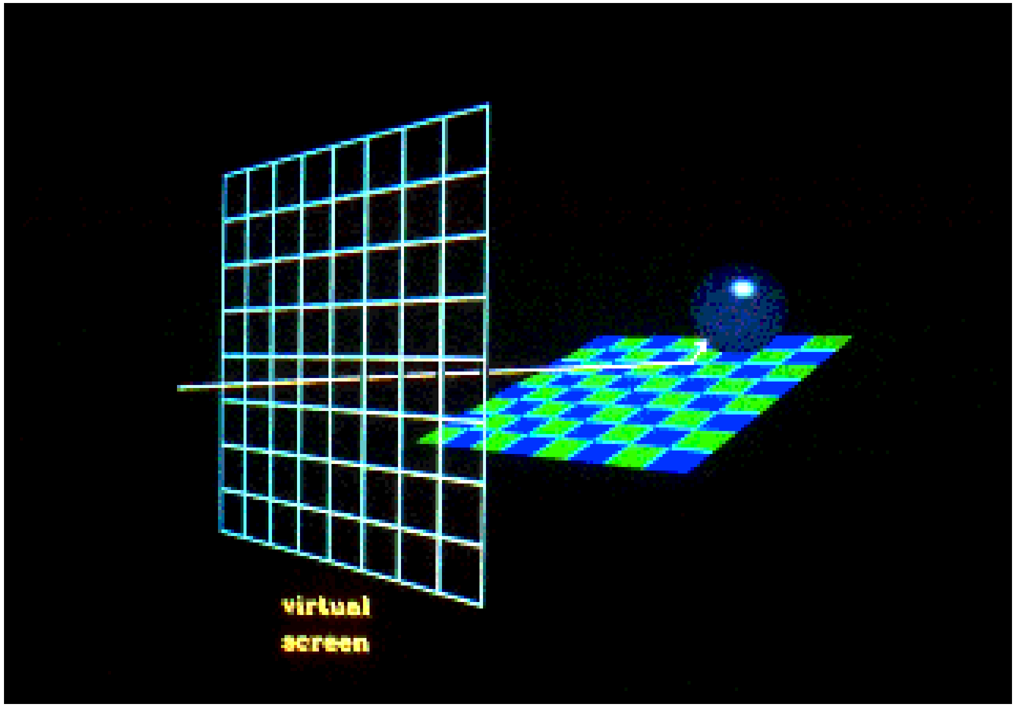
Case B: Ray hits an object

- **Ray hits object:** Check if hit point is in shadow, build secondary ray (shadow ray) towards each light source

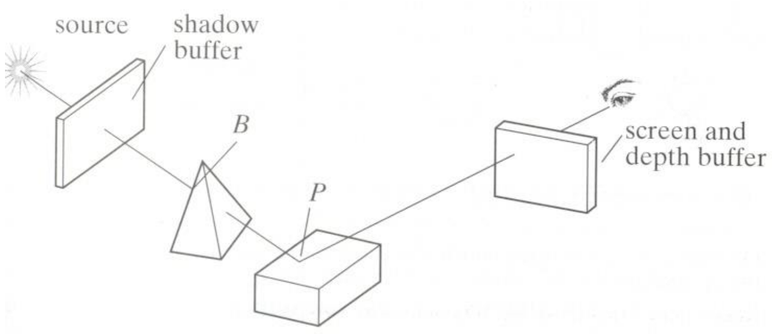


Case B: Ray hits an object

- If shadow ray hits another object before light source: first intersection point is in shadow of the second object (use only ambient)
- Otherwise, not in shadow. (use ambient + diffuse + specular)

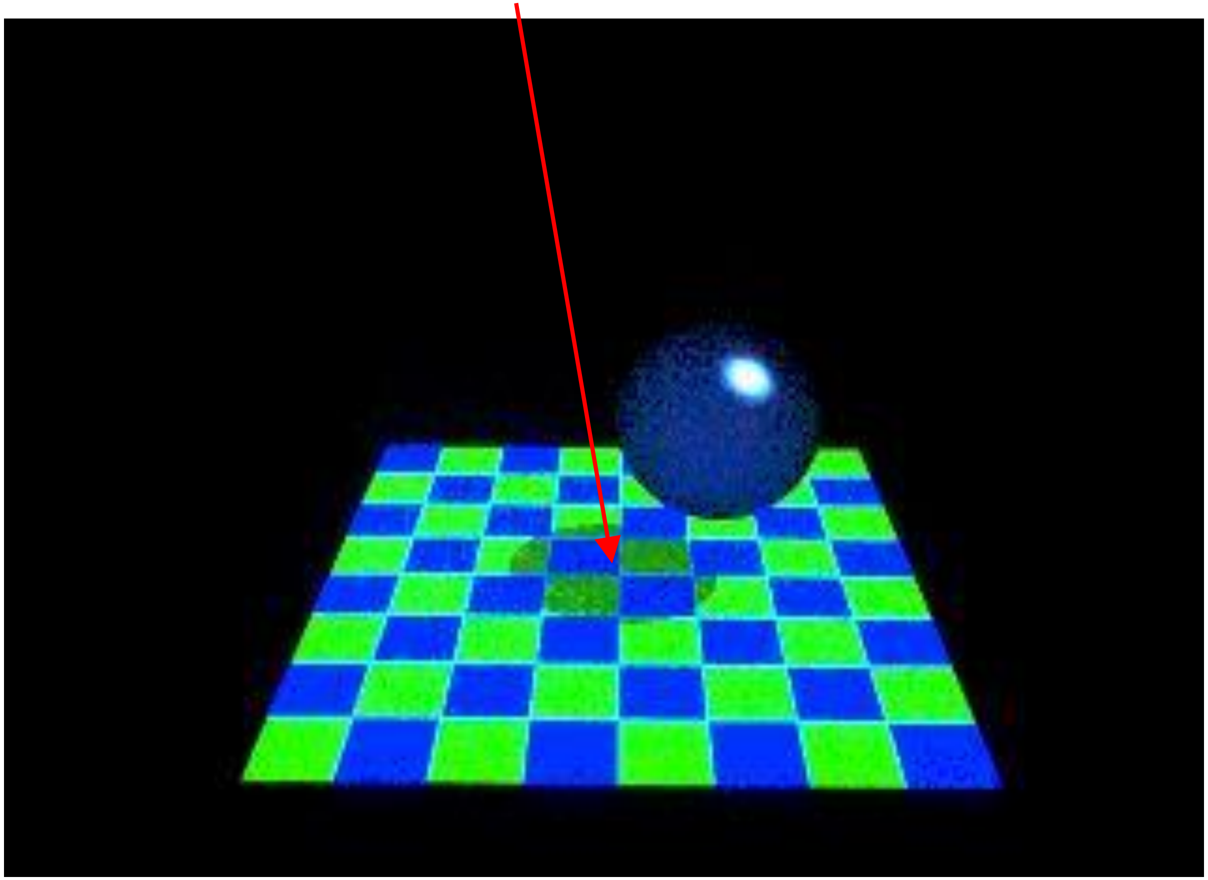


Recall: P in shadow of B



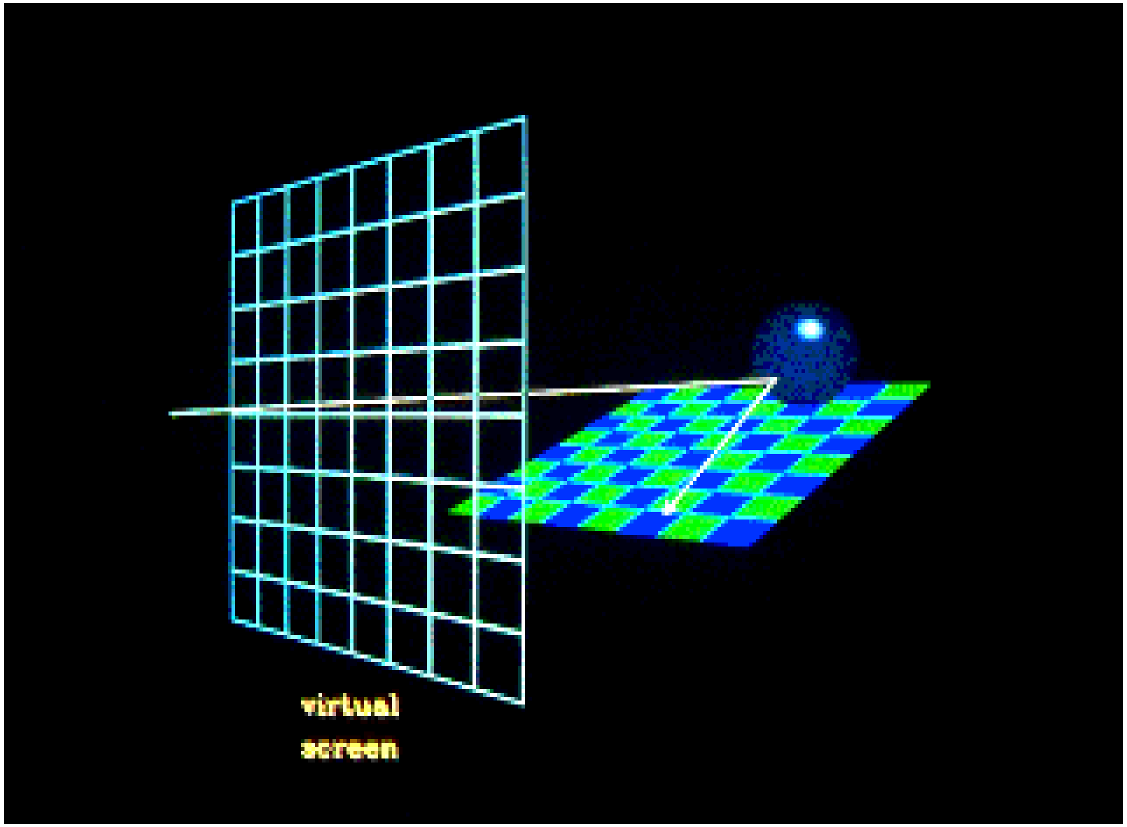
Case B: Ray hits an object

- First Intersection point in the shadow of the second object is the shadow area.

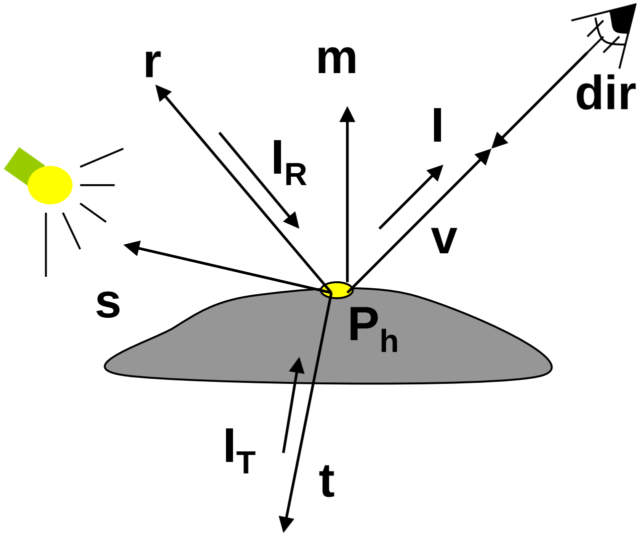


Reflected Ray

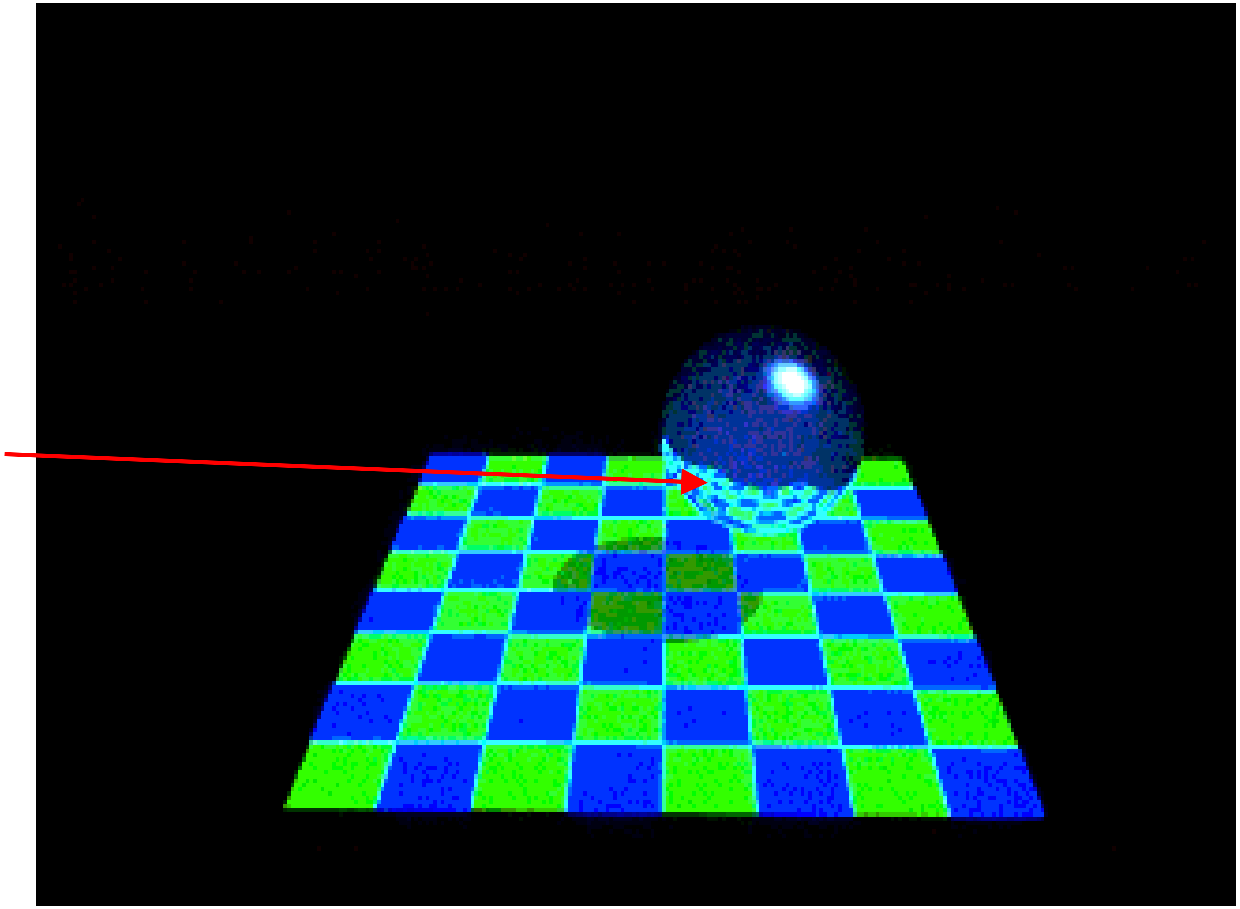
- When a ray hits an object, a reflected ray is generated which is tested against all of the objects in the scene.



Recall: Reflected Ray r, in mirror direction



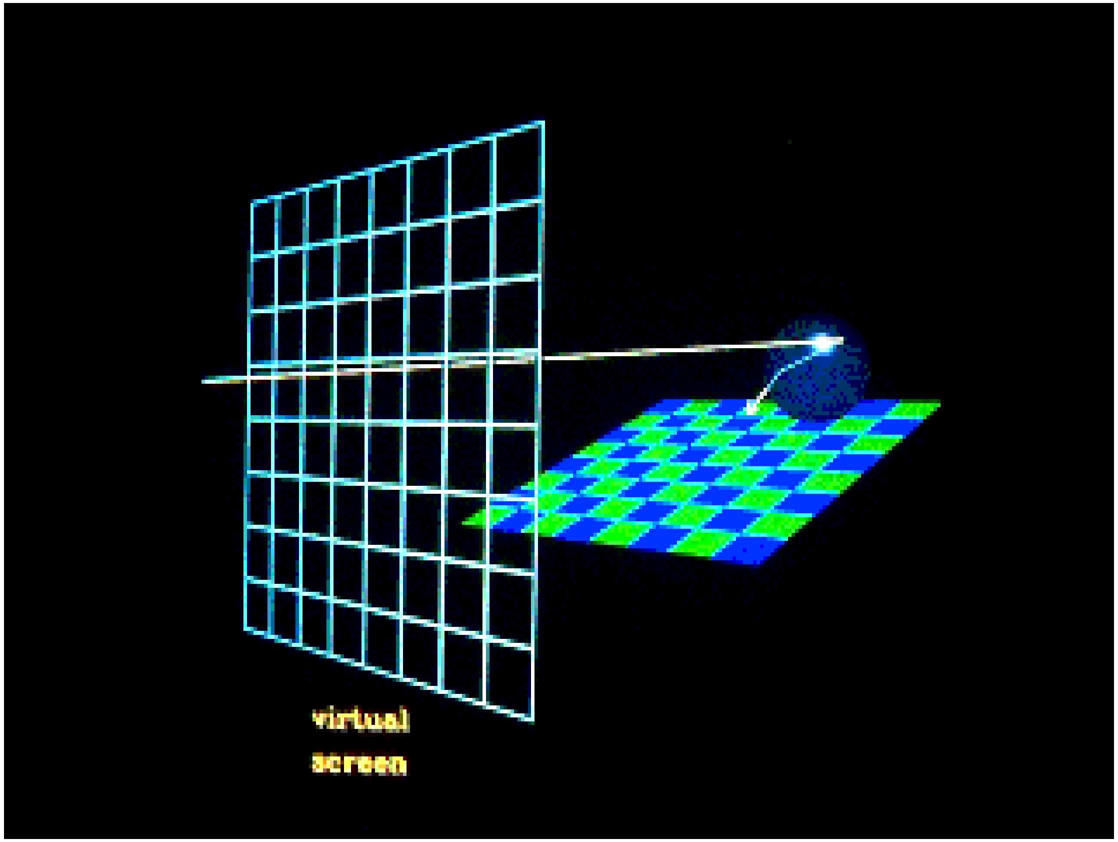
Reflection: Contribution from the reflected ray

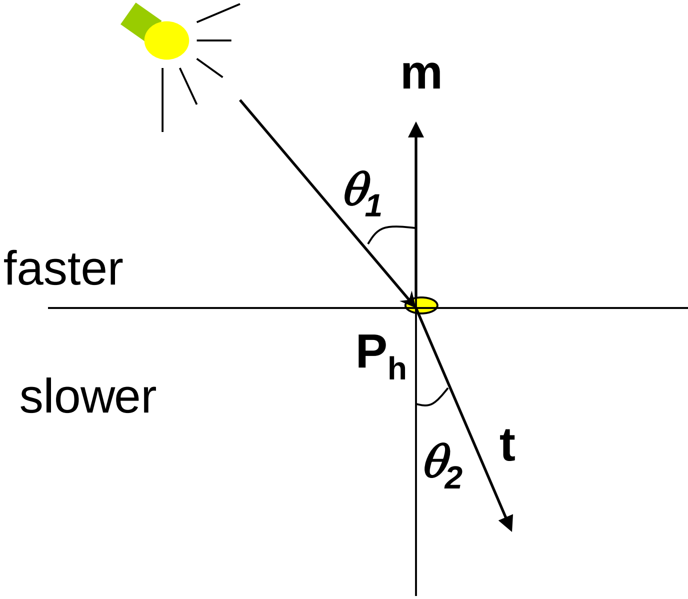


Ambient + Diffuse + Specular + Reflected

Transparency

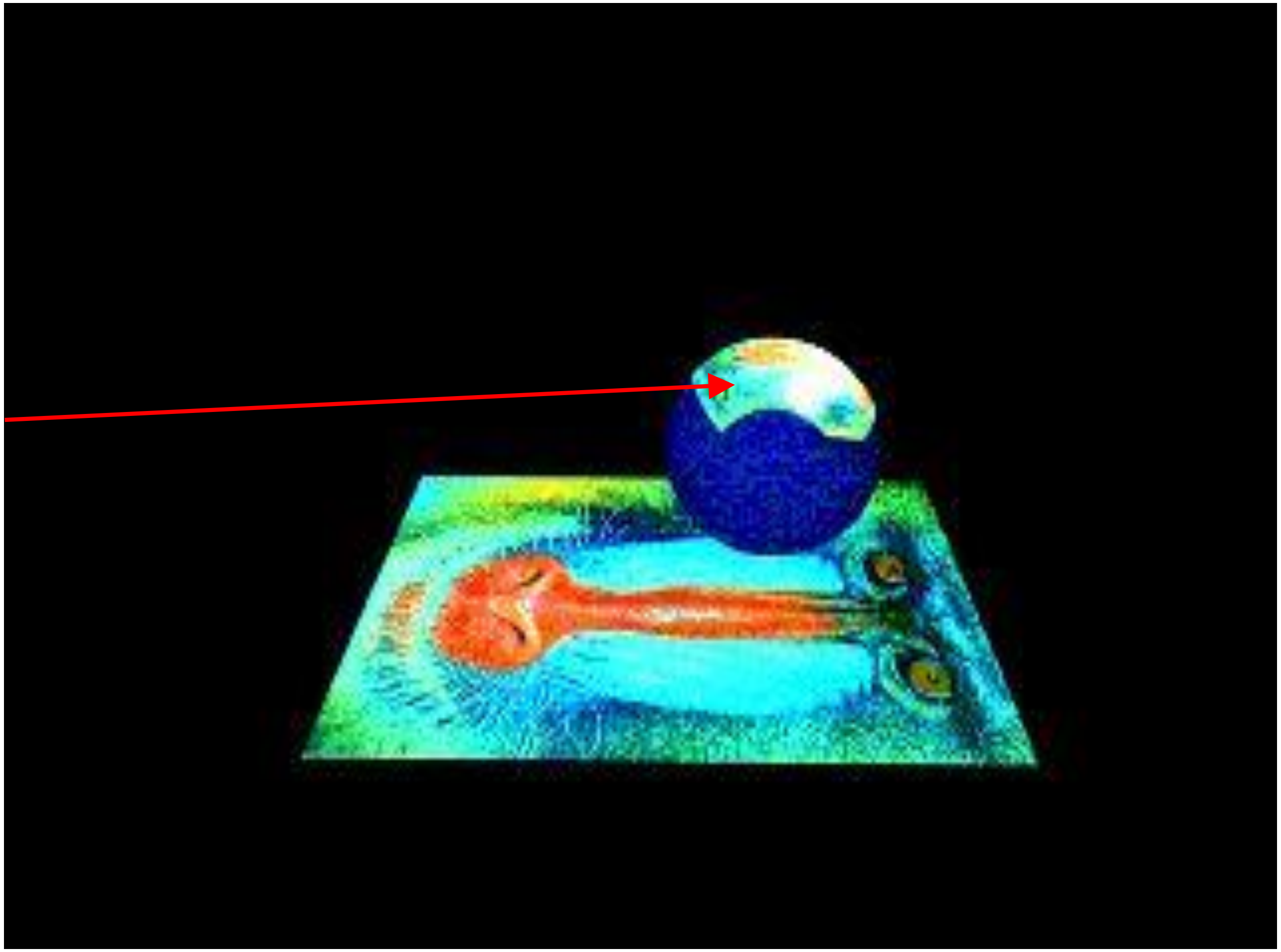
- If intersected object is transparent, transmitted ray is generated and tested against all the objects in the scene.





Recall: Transmitted Ray t, Using Snell's law

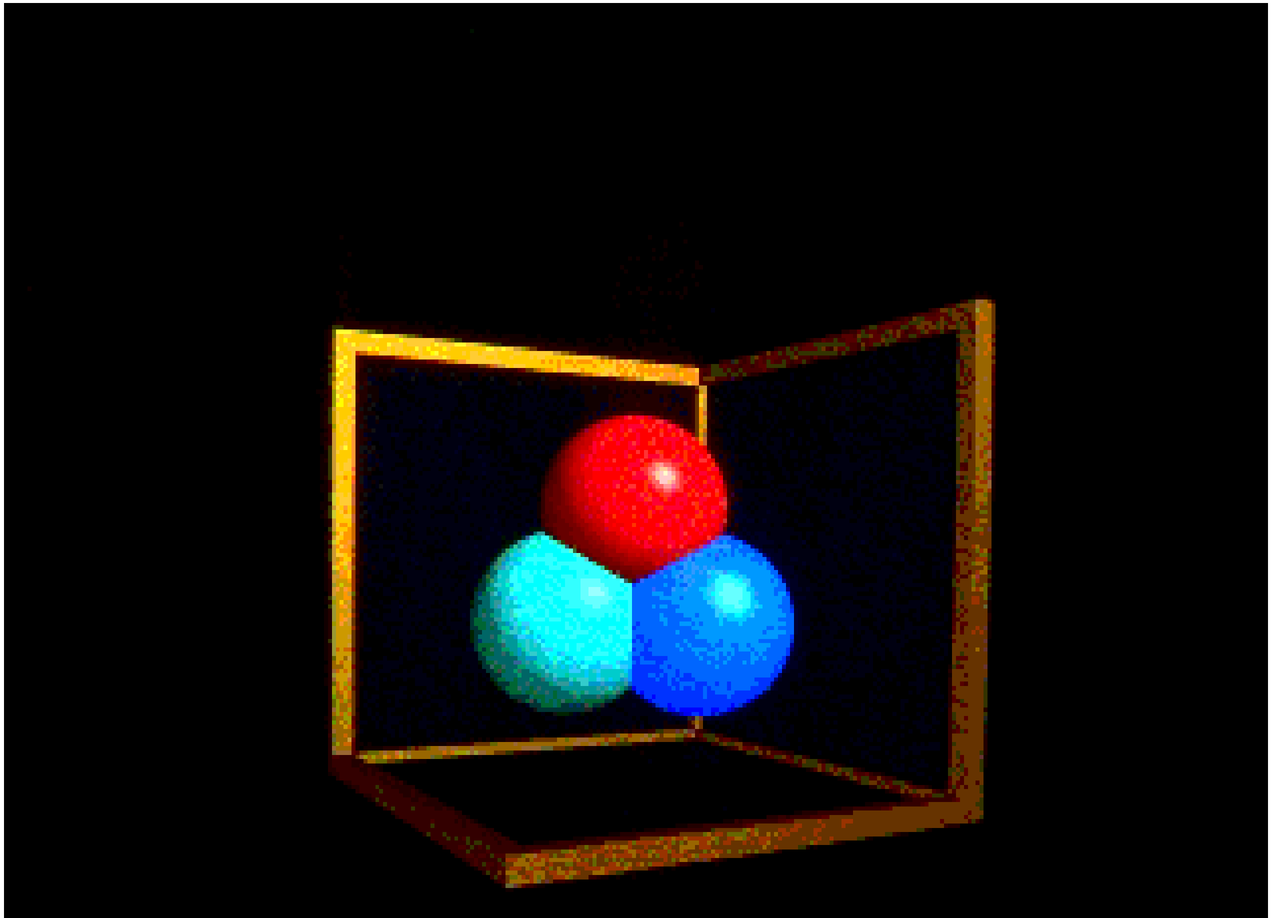
Transparency: Contribution from transmitted ray



Ambient + Diffuse + Specular + Reflected + Transmitted

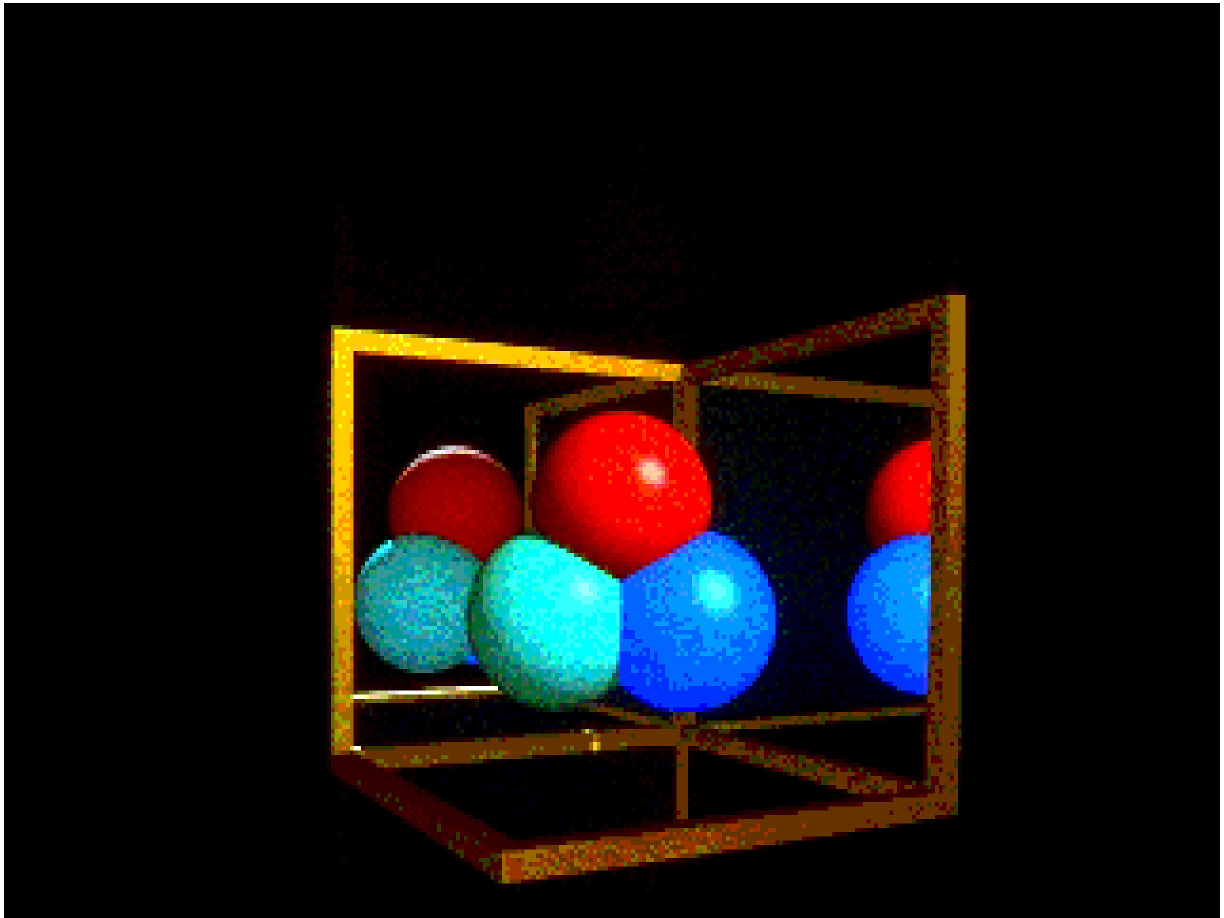
Reflected Ray: Recursion

Reflected rays can generate other reflected rays that can generate other reflected rays, etc. Case A: *Scene with no reflection rays*



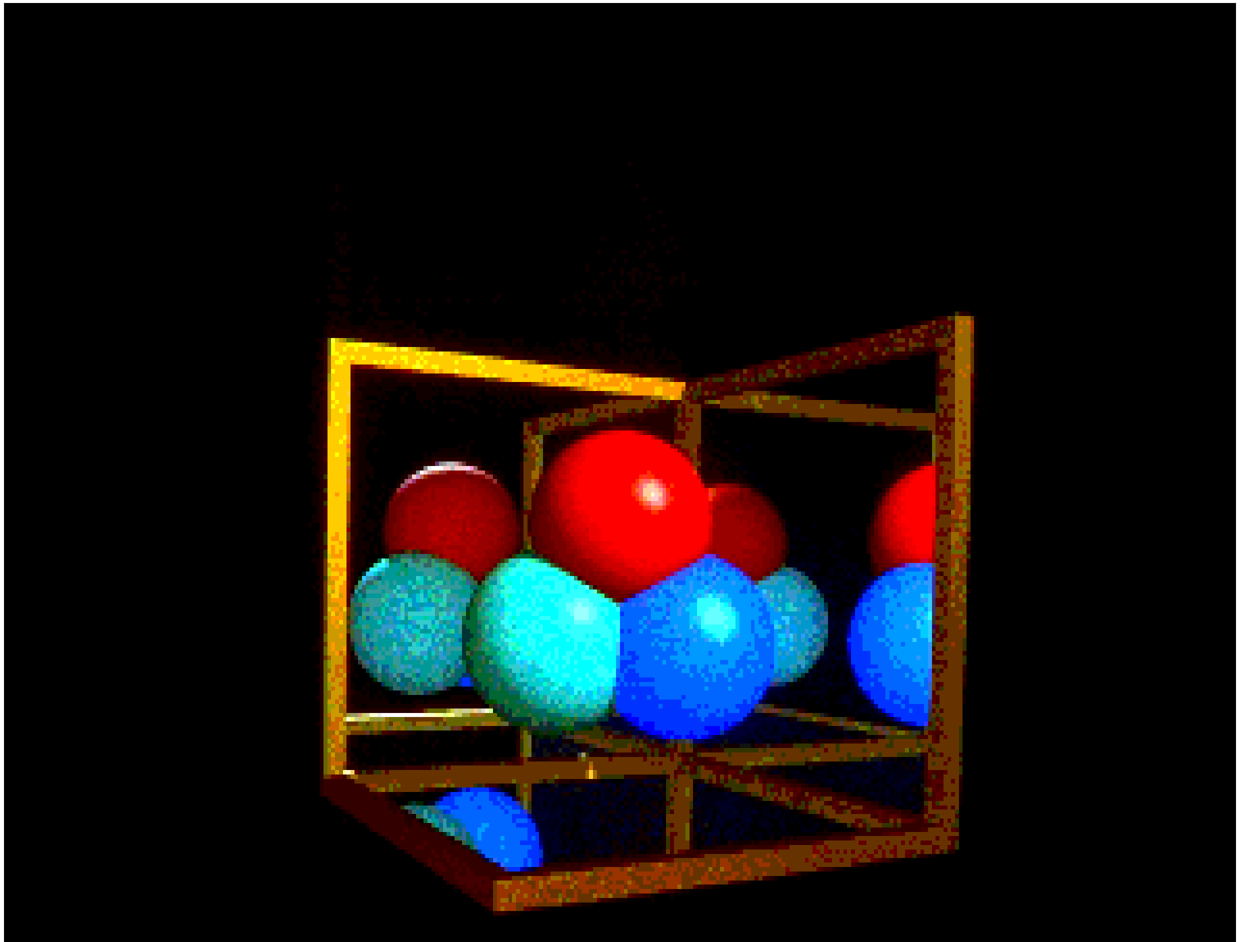
Reflected Ray: Recursion

Case B: Scene with one layer of reflection



Reflected Ray: Recursion

Case C: Scene with two layers of reflection



Ray Tree

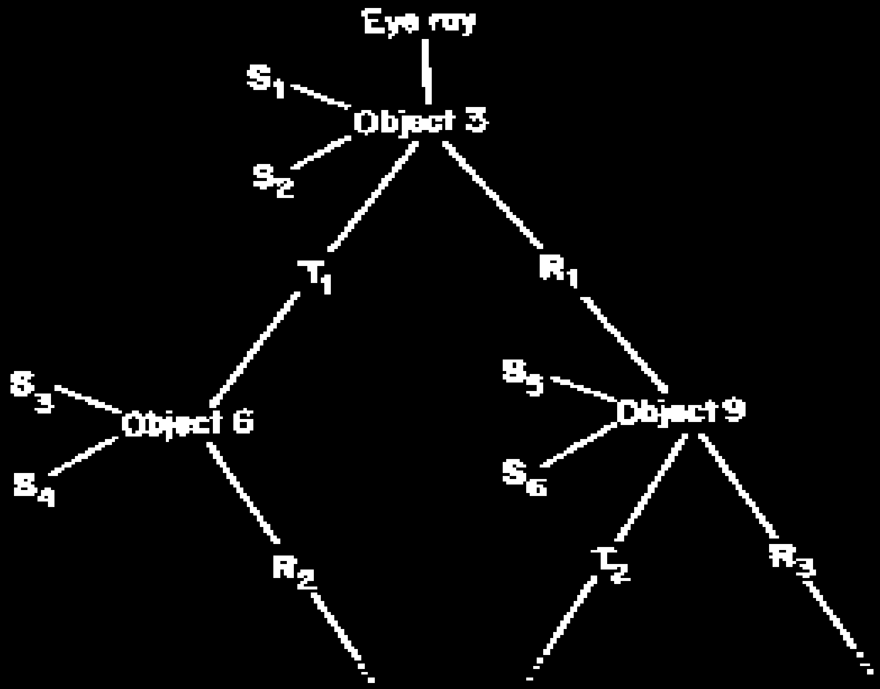
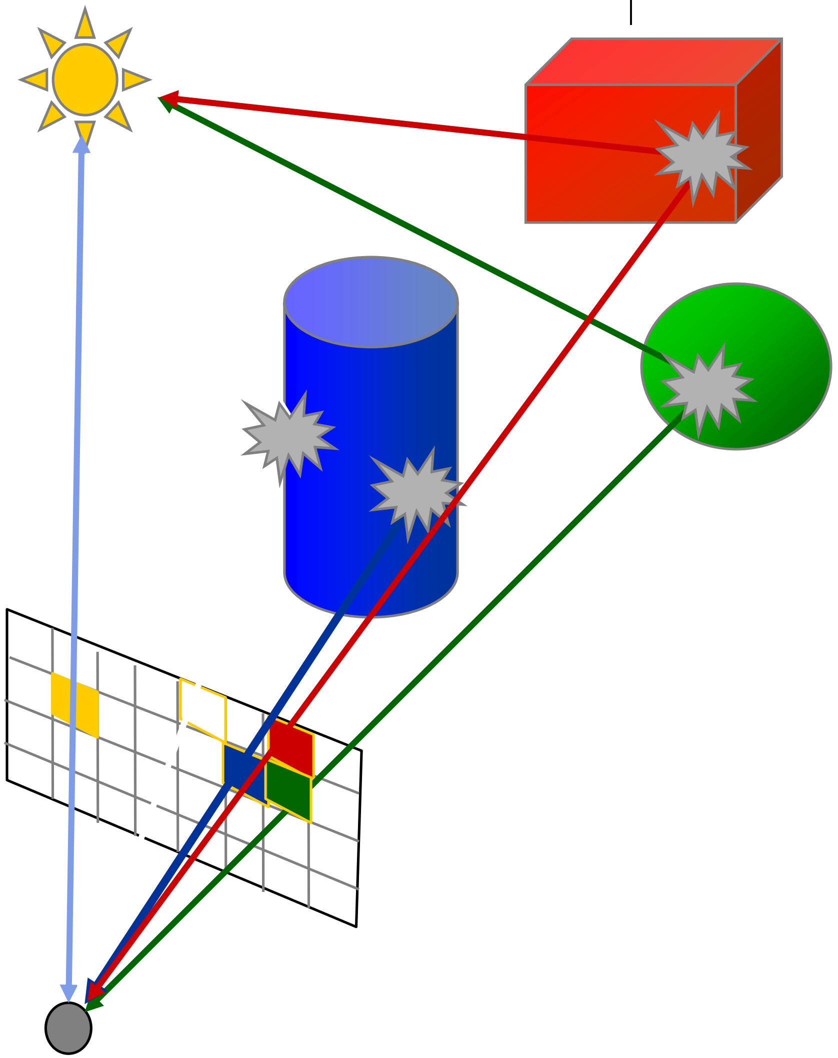


Fig. 12. The ray tree in schematic form.

- Reflective and/or transmitted rays are continually generated until ray leaves the scene without hitting any object or a preset recursion level has been reached.

Find Object Intersections with rc-th ray

- Much of ray tracing work is in finding ray-object intersections
- Break into two parts
  - Find intersection with untransformed, generic (dimension 1) shape first
  - Later: deal with transformed objects
- Express ray, objects (sphere, cube, etc) mathematically
- Ray tracing idea:
  - put ray mathematical equation into object equation
  - determine if valid intersection occurs
  - Object with smallest hit time is object seen through pixel



Find Sphere Intersections with rc-th ray

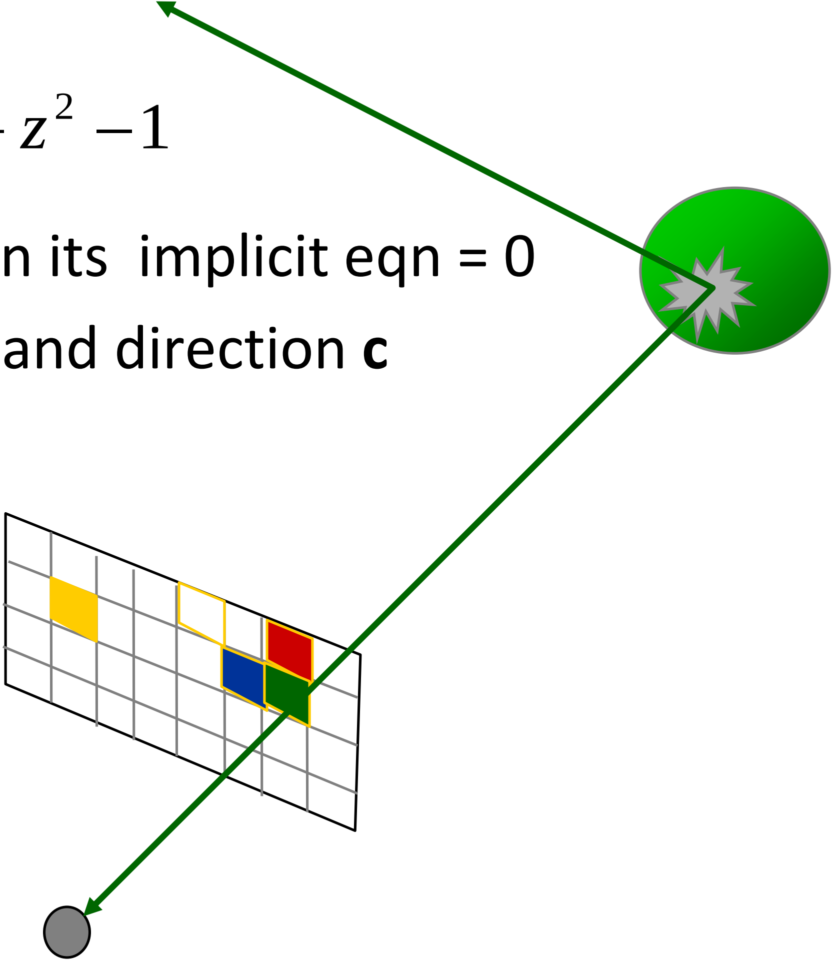
- Ray generic object intersection best found by using implicit form of each shape. E.g. generic sphere is

$$F(x, y, z) = x^2 + y^2 + z^2 - 1$$

- Approach: ray r(t) hits a surface when its implicit eqn = 0
- So for ray with starting point S (eye) and direction **c**

$$r(t) = S + \mathbf{c}t$$

$$F(S + \mathbf{c}t_{hit}) = 0$$



Ray Intersection with Generic Sphere

- Generic sphere has form

$$x^2 + y^2 + z^2 = 1$$

$$x^2 + y^2 + z^2 - 1 = 0$$

$$F(x, y, z) = x^2 + y^2 + z^2 - 1$$

$$F(P) = |P|^2 - 1$$

- Substituting S + **c**t in *F(P) = 0,* we get

$$|S + \mathbf{c}t|^2 - 1 = 0$$

$$|\mathbf{c}|^2 t^2 + 2(S \cdot \mathbf{c})t + (|S|^2 - 1) = 0$$

- This is a quadratic equation of the form $At^2 + 2Bt + C = 0$ where $A = |\mathbf{c}|^2$, $B = S.\mathbf{c}$ and $C = |S|^2 - 1$

Ray Intersection with Generic Sphere

- Solving

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

- If discrimant ($B^2 - AC$) is negative, no solutions, ray misses sphere
- If discriminant ($B^2 - AC$) is zero, ray grazes sphere at one point and hit time is –B/A
- If discriminant ($B^2 - AC$) is +ve, two hit times t1 and t2 (+ve and –ve) discriminant

Ray-Object Intersections

- Object equations and hence intersections vary, depend on parametric equations of object
  - Ray-Sphere Intersections
  - Ray-Plane Intersections
  - Ray-Polygon Intersections
  - Ray-Box Intersections
  - Ray-Quadric Intersections

  (cylinders, cones, ellipsoids, paraboloids)

Accelerating Ray Tracing

- Ray Tracing is time-consuming because of intersection calculations
- Each intersection requires from a few (5-7) to many (15-20) floating point (fp) operations
- Example: for a scene with 100 objects and computed with a screen resolution of 512 x 512, assuming 10 fp operations per object test there are about 250,000 X 100 X10 = 250,000,000 fp operations.
- Solutions:
  - Use faster machines
  - Use specialized hardware, especially parallel processors or graphics card
  - Speed up computations by using more efficient algorithms
  - Reduce the number of ray - object computations

Reducing Ray-Object Intersections

- Adaptive Depth Control: Stop generating reflected/transmitted rays when computed intensity becomes less than certain threshold.

- Bounding Volumes:
  - Enclose groups of objects in sets of hierarchical bounding volumes
  - First test for intersection with the bounding volume
  - Then only if there is an intersection, against the objects enclosed by the volume.

- First Hit Speed-Up: use modified Z-buffer algorithm to determine the first hit.

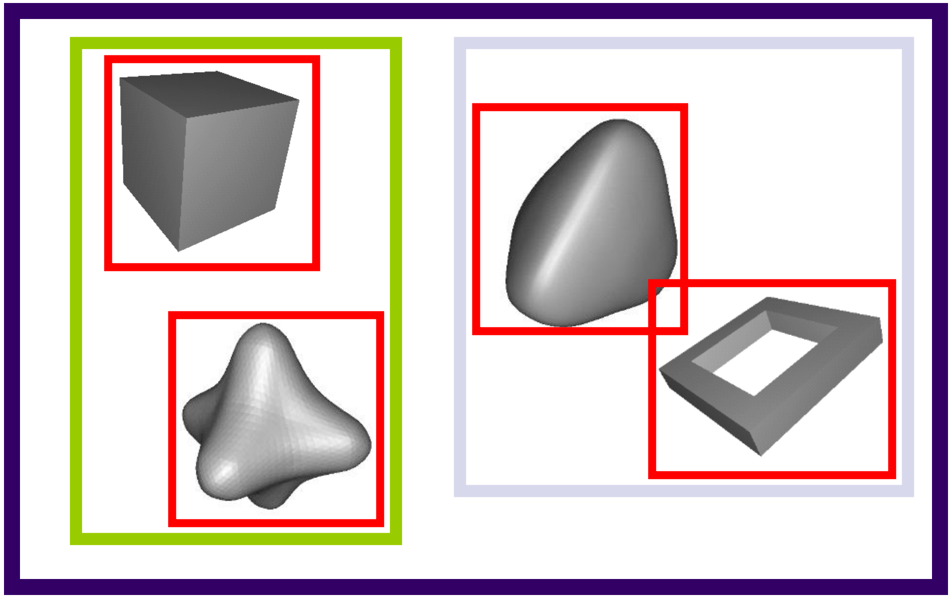
Popular Spatial Acceleration Structures

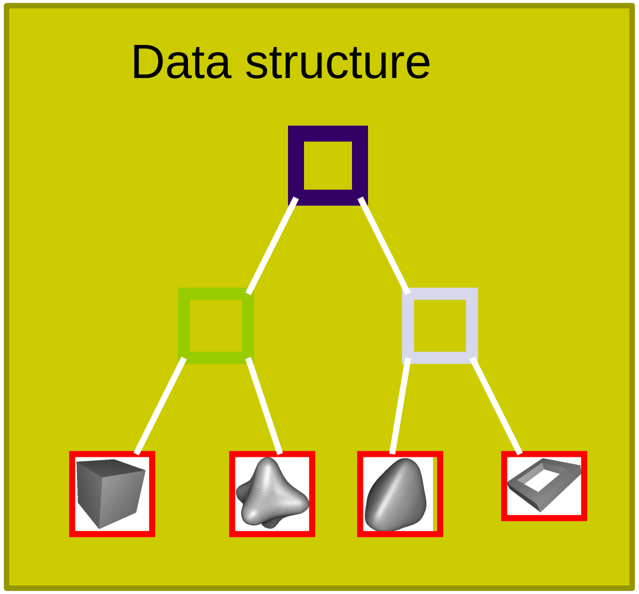
- **Spatial Data Structures:** manage scene geometry
  - Bounding Volume Hierarchies
  - BSP Trees
  - Octrees
  - Scene Graphs

How?

- Organizes geometry in some hierarchy

In 2D space





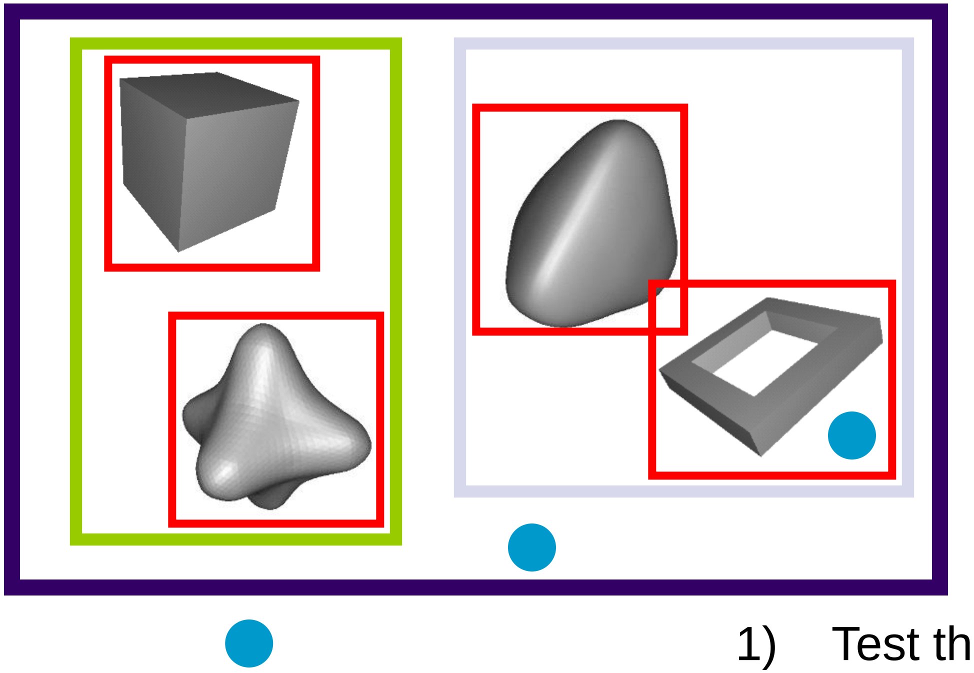
Data structure

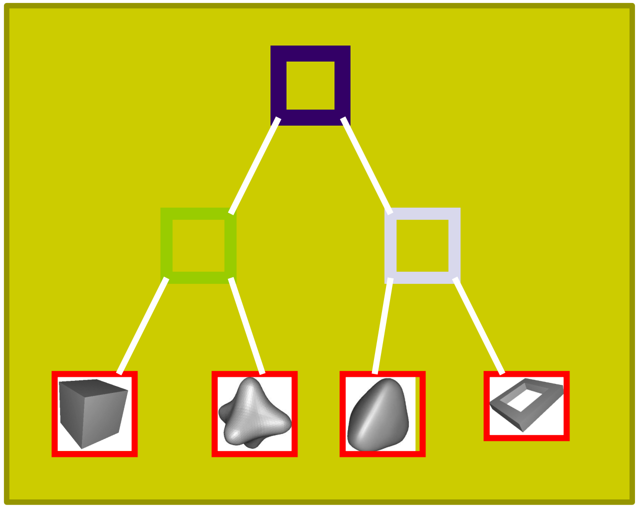
Bounding Volume Hierachy

Basic idea: Test bigger volumes first.
If no hit, avoid testing smaller volumes inside it

What's the point?
An example

- Assume we click on screen, and want to find which object we clicked on



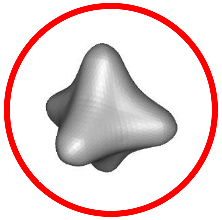


1) Test the root first
2) Descend recursively as needed
3) Terminate traversal as soon as possible

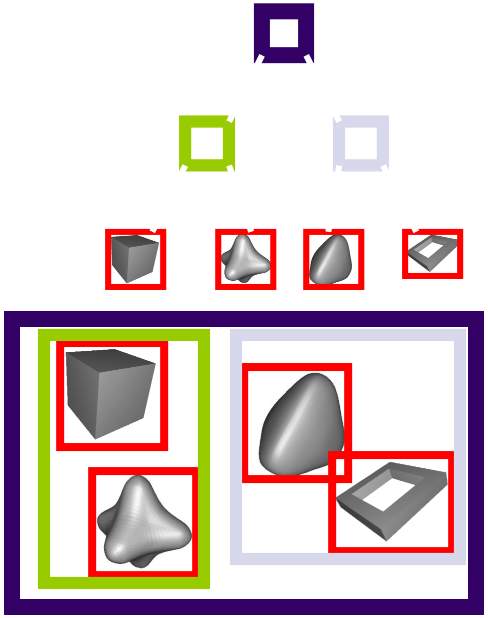
In general: get O(log n) instead of O(n)

Bounding Volume Hierarchy (BVH)

- Use simple shapes to enclose complex geometry
- Most common bounding volumes (BVs):
  - Spheres, boxes (AABB and OBB)



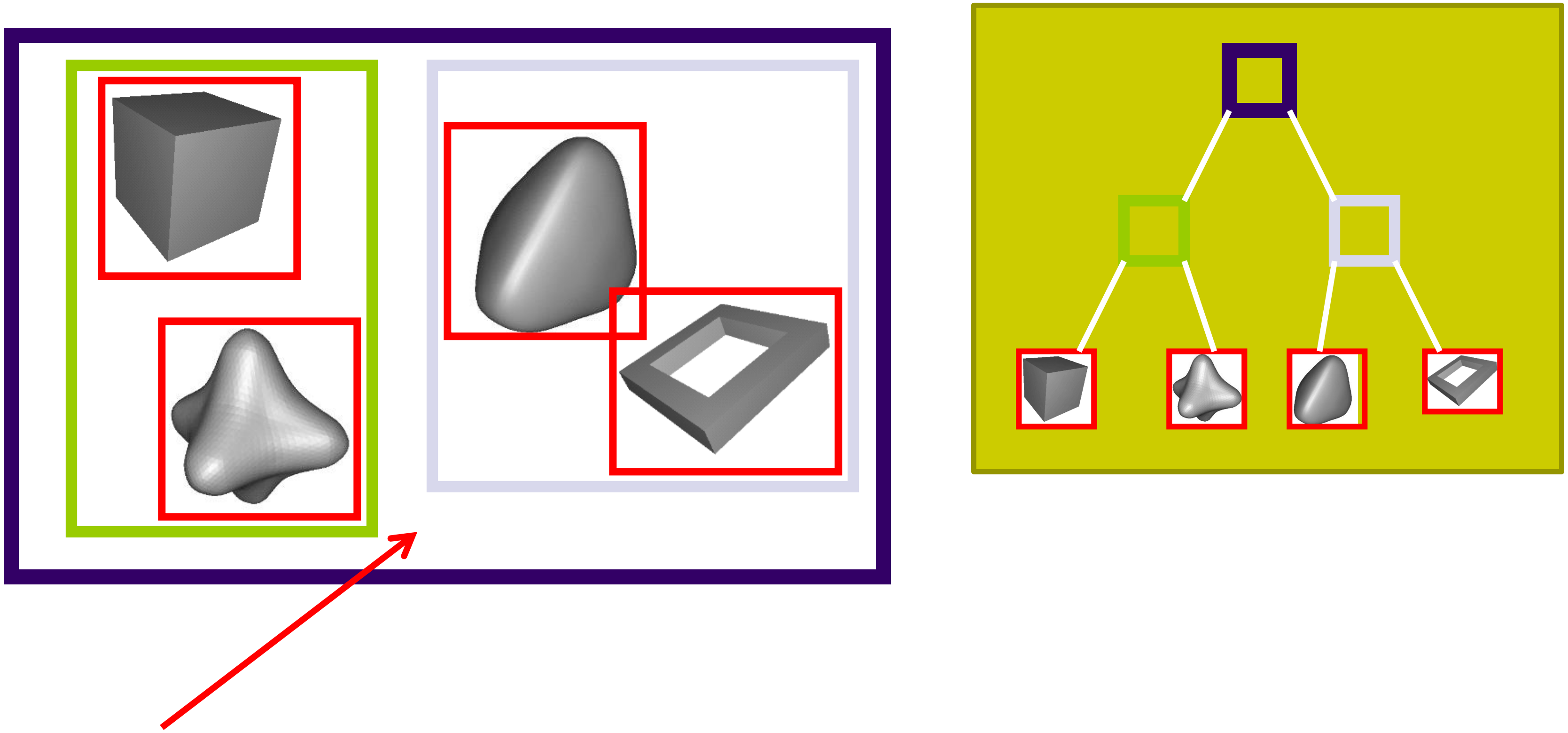
- The BV does not contibute to the rendered image -- rather, encloses an object



- The data structure is a *k*-ary tree
  - Leaves hold geometry
  - Internal nodes have at most *k* children
  - Internal nodes hold BVs that enclose all geometry in its subtree

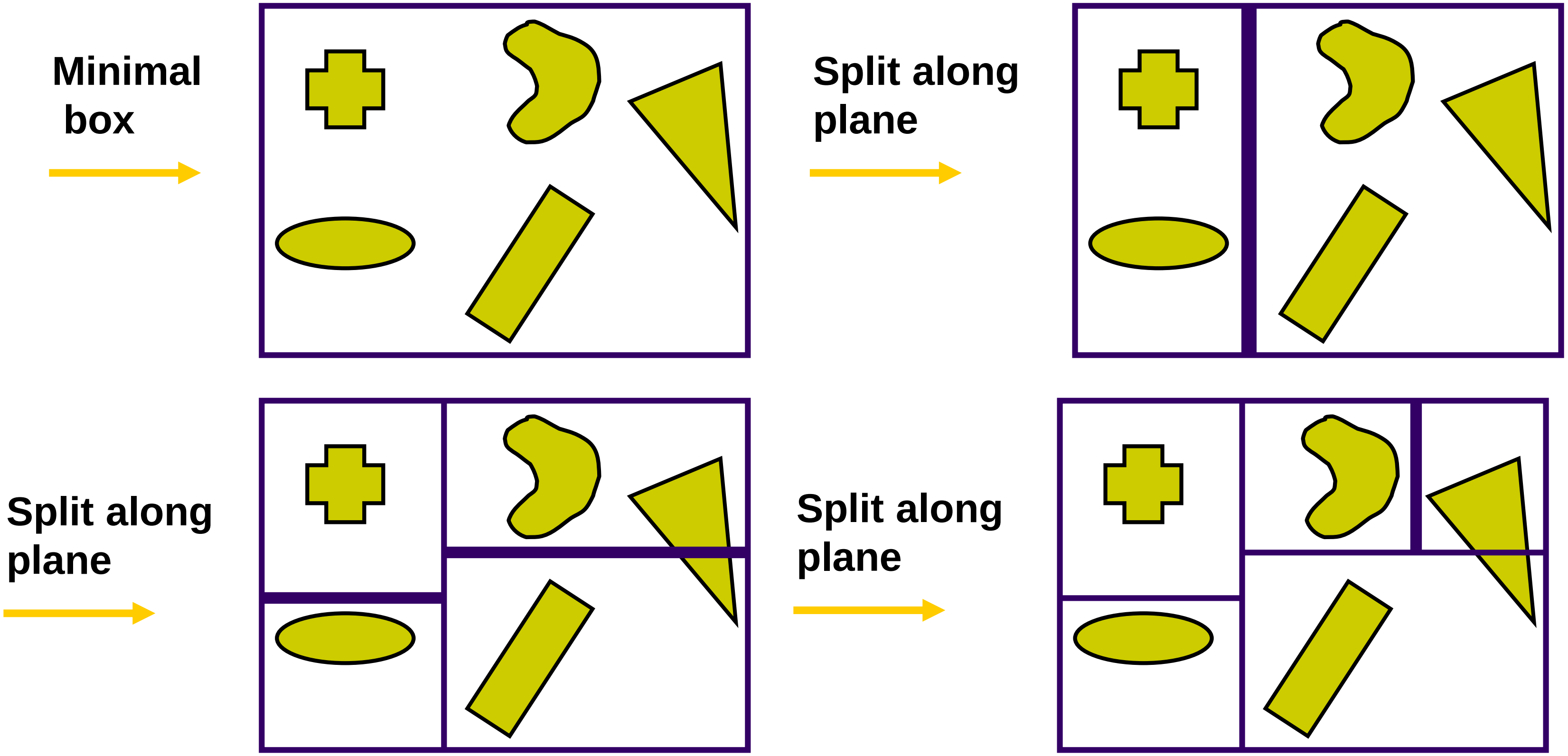
Example Application of BVH: Intersection Testing in RT

- Enclose scene geometry in BVH
- Cube/box much easier to test for intersections
- Large time savings if ray misses portions of scene



Axis-Aligned BSP tree

- General idea:
  - Divide space with a plane
  - Sort geometry into the space it belongs
  - Can only make a splitting plane along x,y, or z



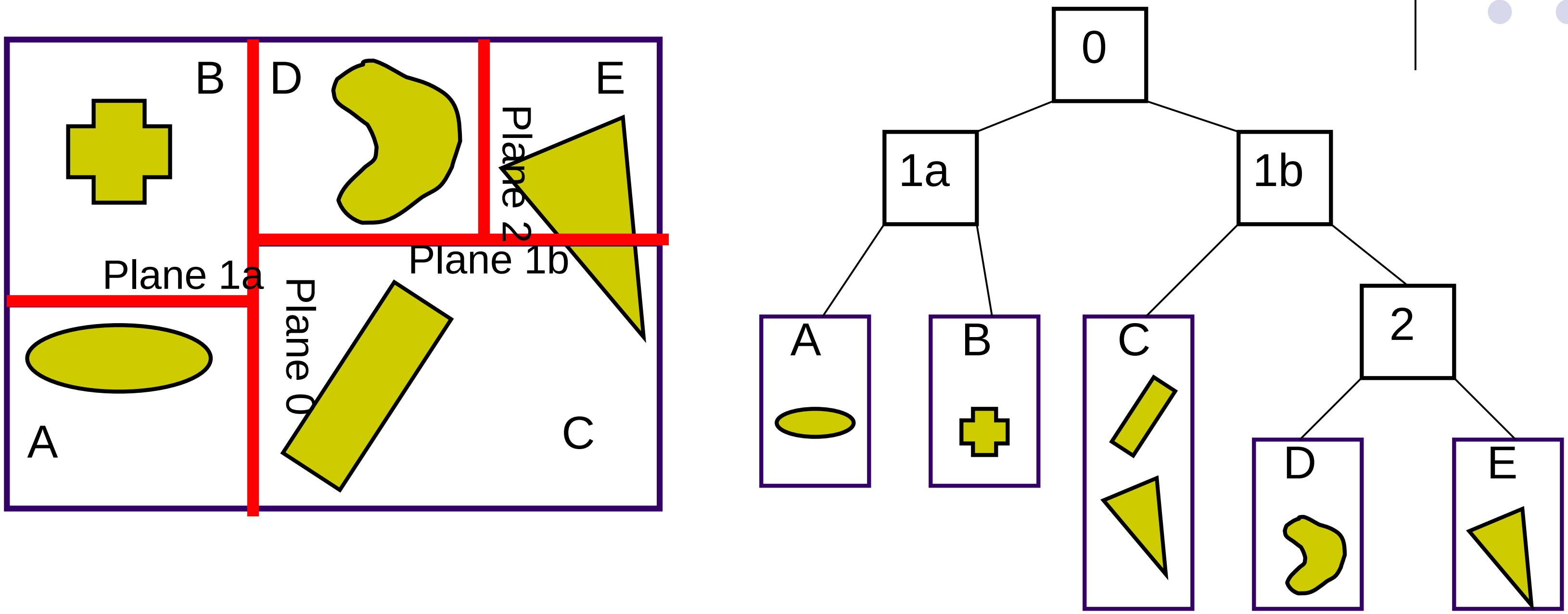
Minimal box

Split along plane

Split along plane

Split along plane

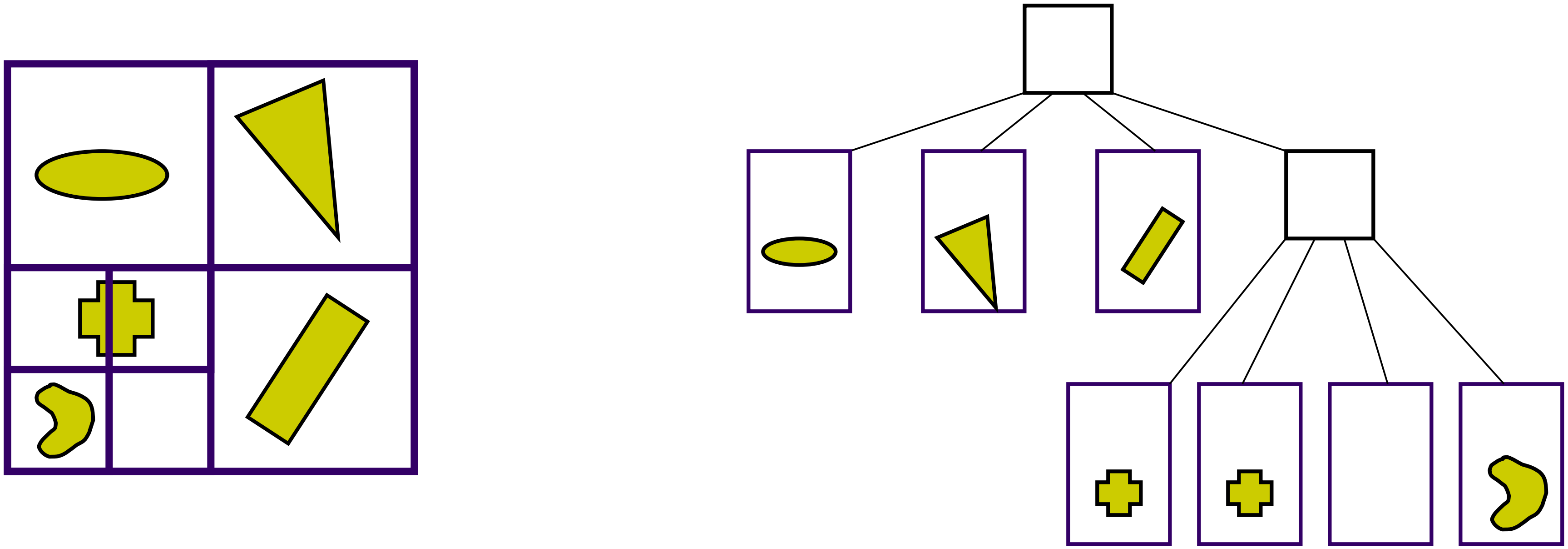
Axis-Aligned BSP tree



- Each internal node holds a divider plane
- Leaves hold geometry
- Differences compared to BVH
    - Encloses **entire space**
    - BVHs can use any desirable type of BV

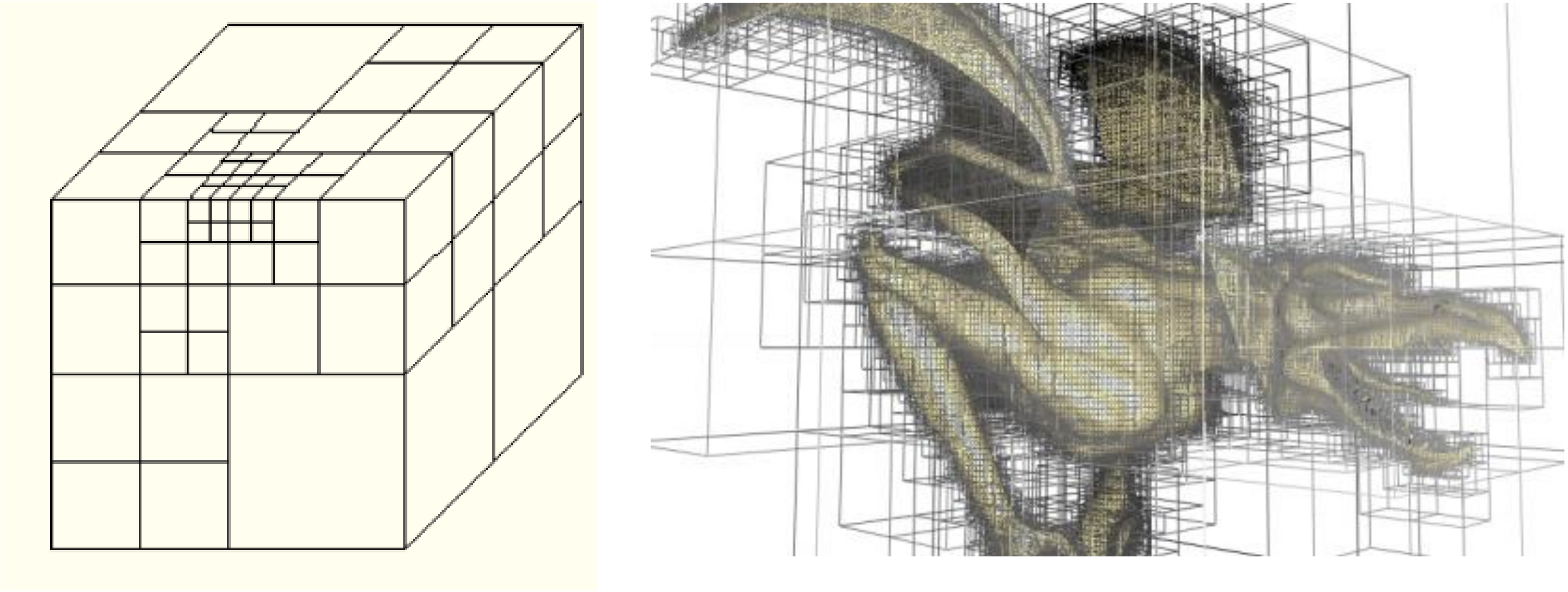
Octrees

- Similar to axis-aligned BSP trees but **regular (split in middle)**
- Variants:
  - Quadtree (2D) below and octree (3D)



- Quadtree

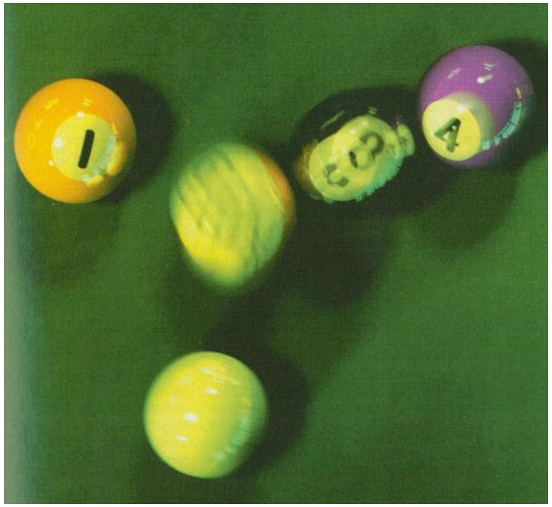
Example of Octrees

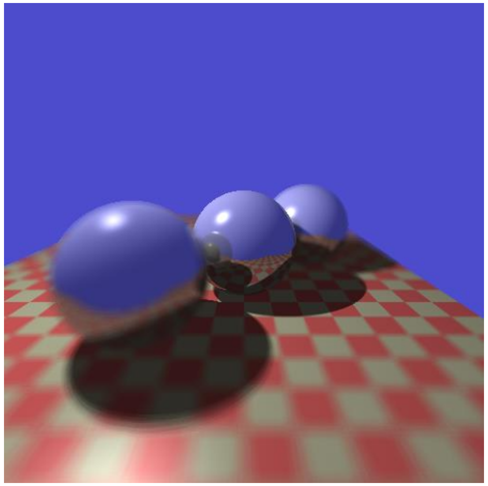


- In 3D each square (or rectangle) becomes a box, and 8 children

Making Ray Tracing Look Real

- **Antialiasing**
  - Cast multiple rays from eye through same point in each pixel
- **Motion blur**
  - Introduce time, motion
  - Each ray intersects scene objects at different time
  - Add camera shutter speed, reconstruction filter controls
- **Depth of Field**
  - Simulate camera better
    - f-stop
    - focus
- **Other effects** (soft shadow, glossy, etc)

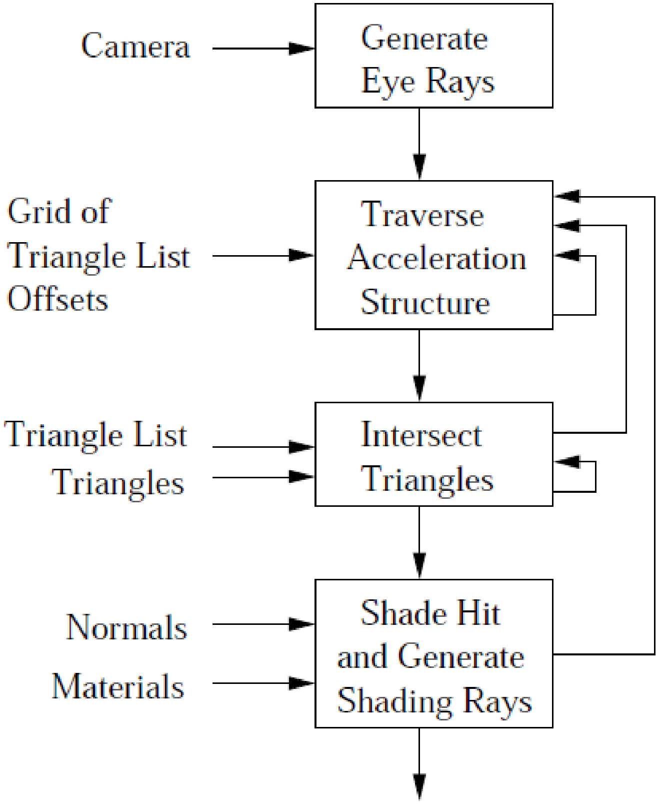




Real Time Ray Tracing

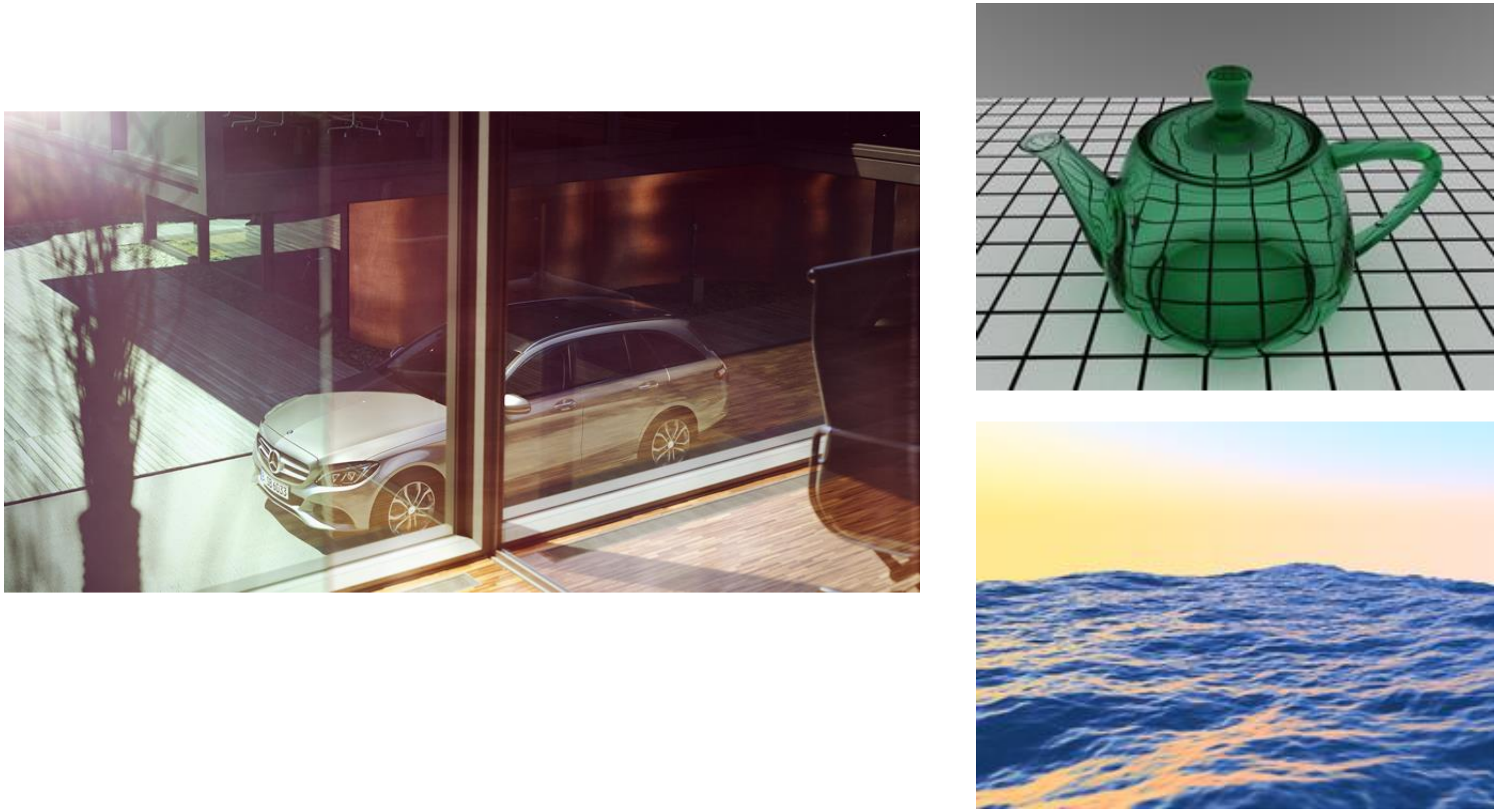
Ref: T. Purcell *et al*, Ray Tracing on Programmable Graphics Hardware, ACM Transactions on Graphics (TOG) 21 (3), pgs 703-712

- Multi-pass rendering: Ray tracer using 4 shaders

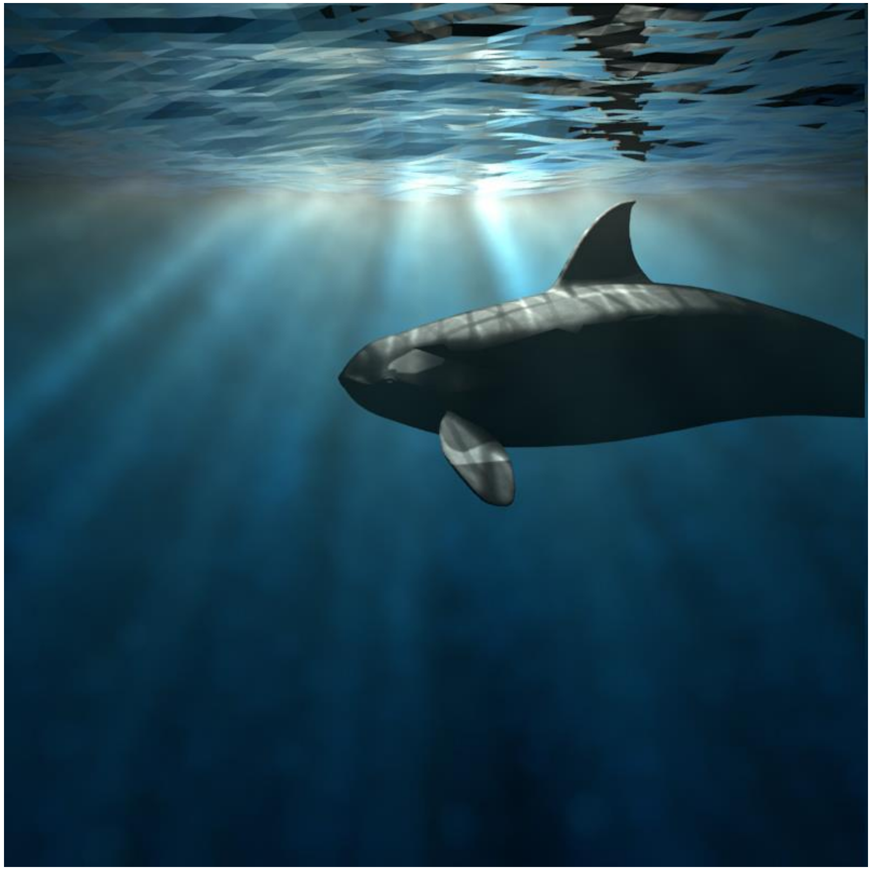


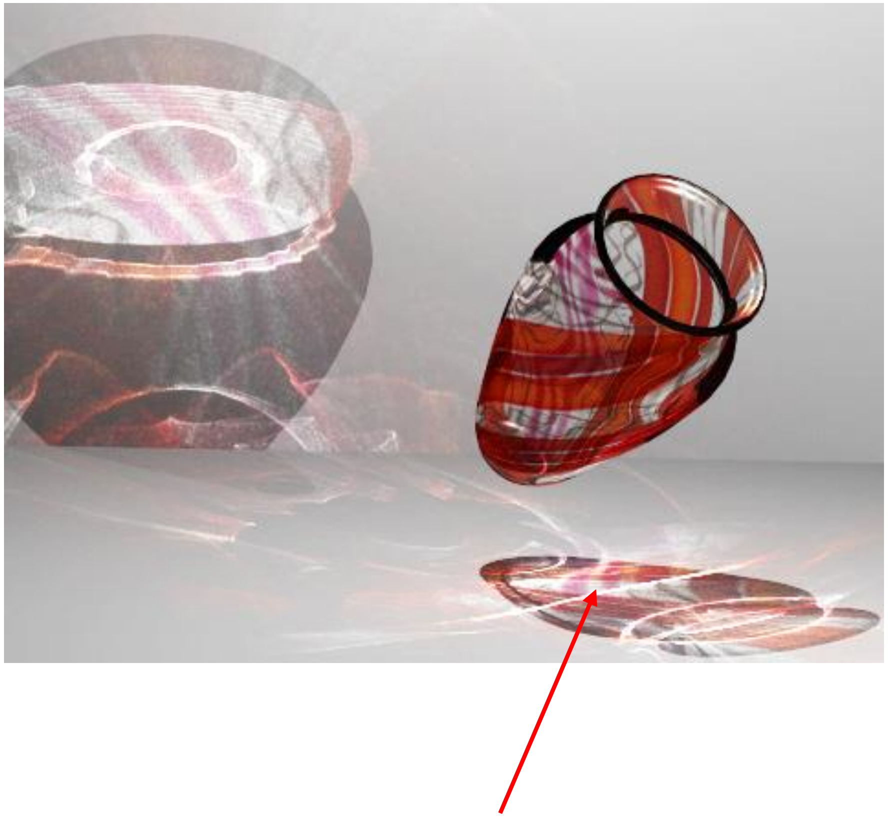
Nvidia Optix Real Time Ray Tracer

- Nvidia software/SDK, available on their website
  - http://developer.nvidia.com/object/optix-home.html
- Needs high end Nvidia graphics card



Photon mapping examples





Caustics

Images: courtesy of Stanford rendering contest

Photon Mapping

- Simulates the transport of individual photons (Jensen ’95-’96)
- Good for effects ray tracing can’t, especially those requiring tracing from light source:
  - Caustics
  - Light through volumes (smoke, water, marble, clouds)
- Two pass algorithm
  - Pass 1 - **Photon tracing** (generate photon map)
  - Pass 2 – **Rendering** scene using photon map

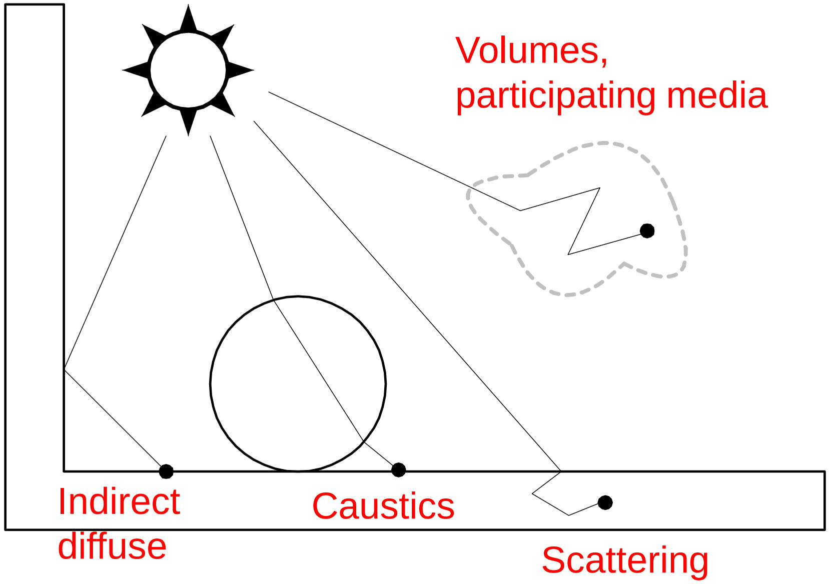


Illustration is based on figures from Jensen[1].

Photon Tracing

Photon scattering

- Emitted photons are probabilistically FROM LIGHT SOURCE, scattered through the scene and are eventually absorbed.
- Photon hits surface: can be reflected, refracted, or absorbed
- Photon hits volume: can be scattered or absorbed
- Store photons at surface/volume in kd-tree (photon maps)

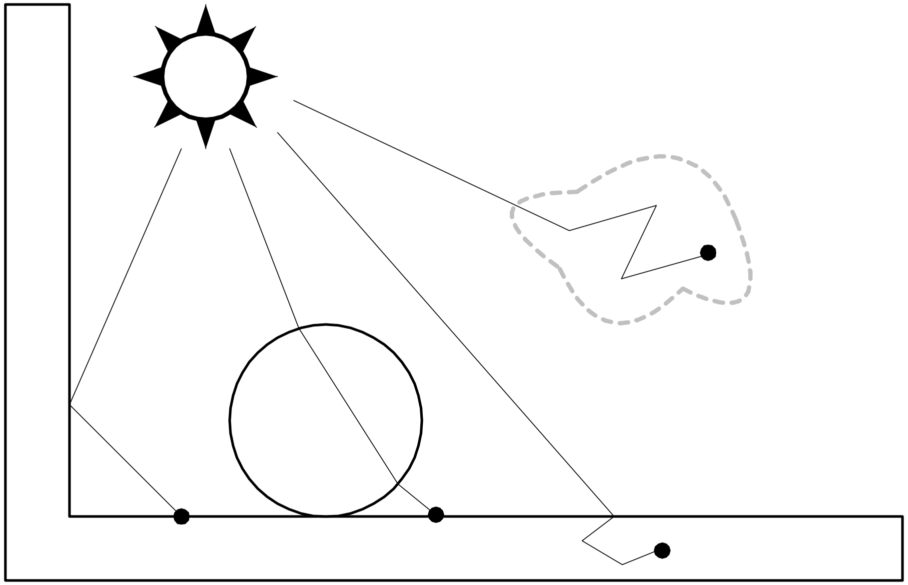
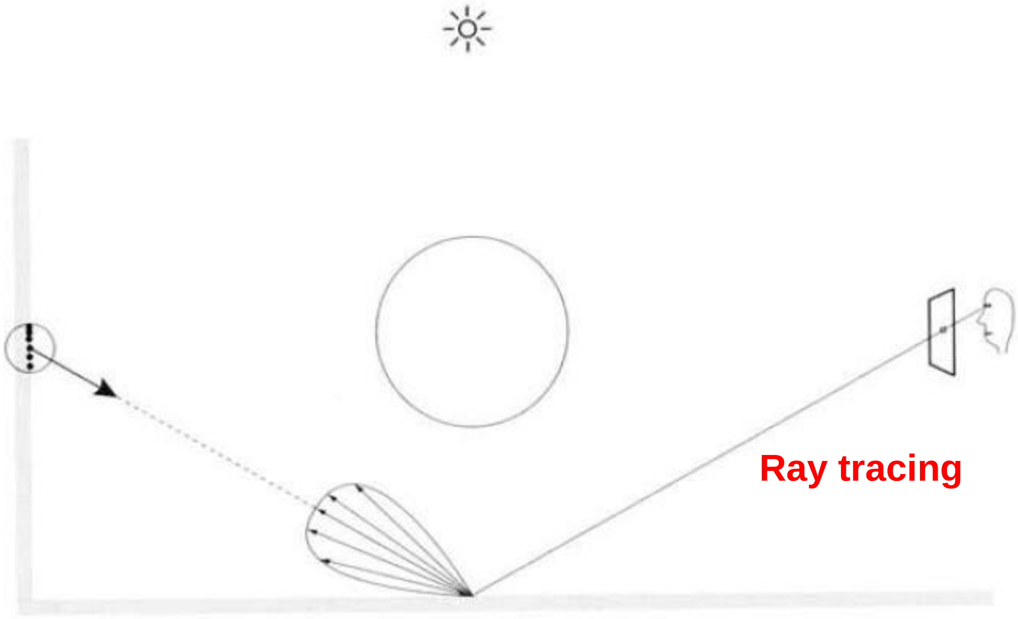


Illustration is based on figures from Jensen[1].

Photon mapping: Pass 2 - Rendering

- Use ray tracing to render scene using information in the photon maps to estimate:
  - Indirect diffuse lighting
  - Reflected radiance at surfaces
  - Scattered radiance from volumes and translucent materials
  - Illumination in volumes, caustics



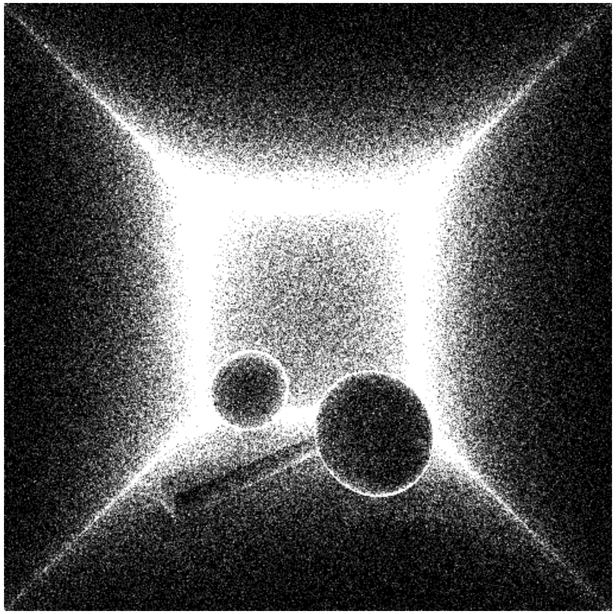
Estimate illumination in photon map

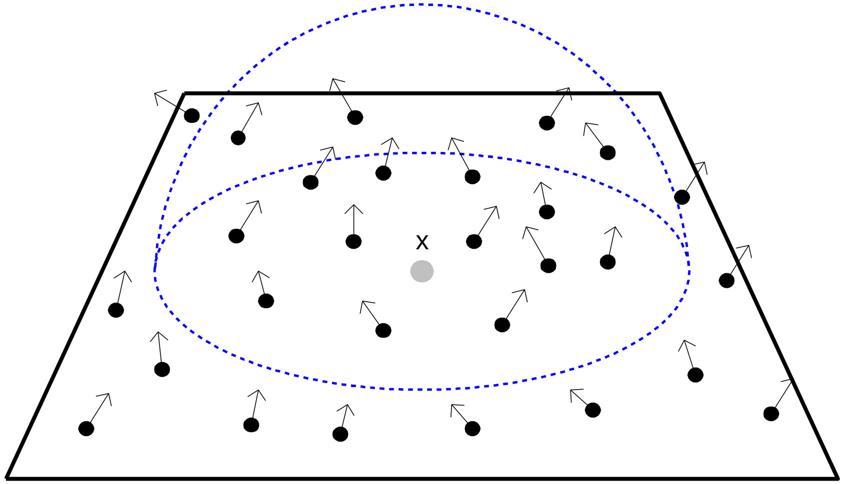
Ray tracing

Photon Tracing

Pass 2 - Rendering

- Imagine ray tracing a hitpoint x
- Information from photon maps used to estimate radiance from x
- Radius of circle required to encountering N photons gives radiance estimate at x

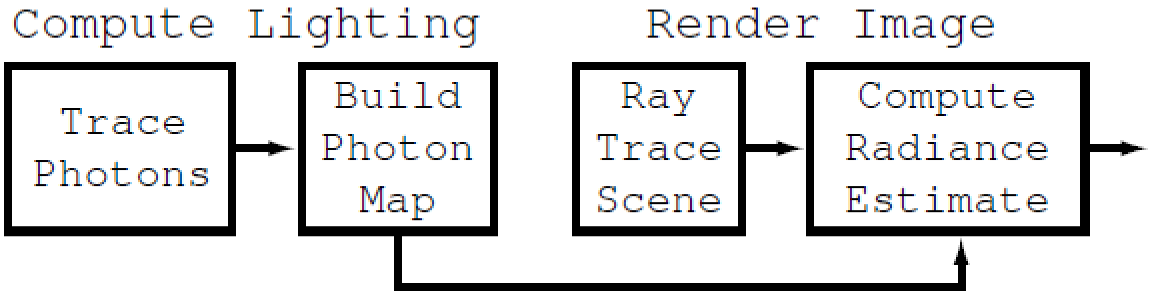




Real Time Photon mapping

Ref: T. Purcell *et al*, Photon mapping on programmable graphics hardware, Graphics Hardware 2003

- Similar idea to real-time ray tracing.
- Photon mapping as multi-pass shading



References

- Hill and Kelley, Computer Graphics using OpenGL, 3rd edition, Chapter 12
- Akenine-Moller, Eric Haines and Naty Hoffman, Real Time Rendering (3rd edition)