CS559: Computer Graphics

Lecture 27: Animation, Depth & Motion Blur, Ray Tracing

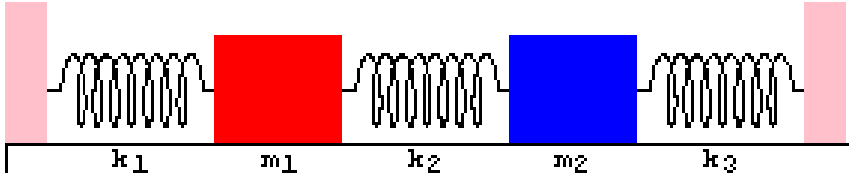
Li Zhang
Spring 2010

Slides from Brian Curless at U of Washington

Particle system diff. eq. solver

We can solve the evolution of a particle system again using the Euler method:

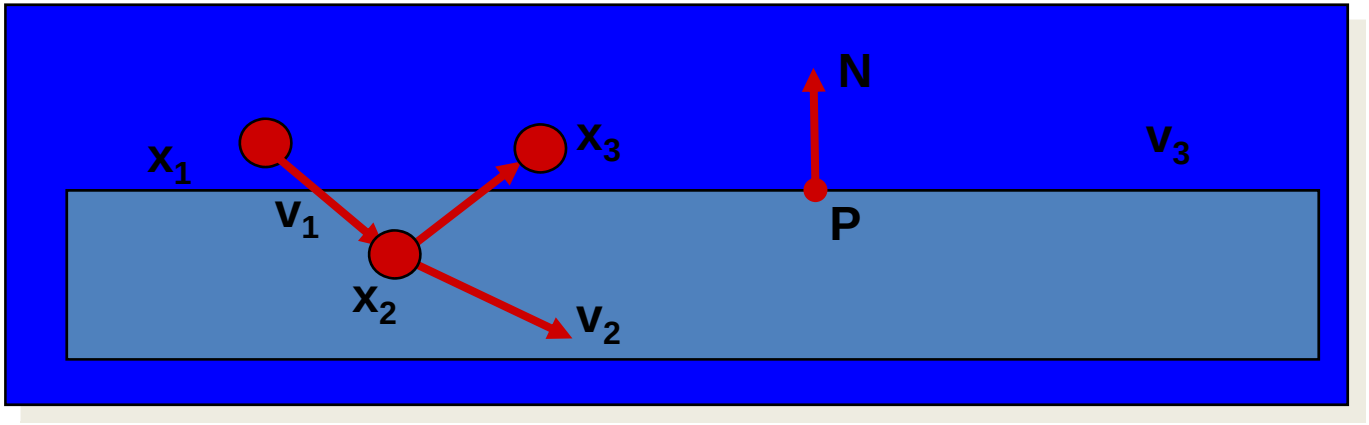
$$\begin{bmatrix} \mathbf{x}_1^{i+1} \\ \mathbf{v}_1^{i+1} \\ \vdots \\ \mathbf{x}_n^{i+1} \\ \mathbf{v}_n^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{i} \\ \mathbf{v}_1^{i} \\ \vdots \\ \mathbf{x}_n^{i} \\ \mathbf{v}_n^{i} \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{v}_1^{i} \\ \mathbf{f}_1^{i} / m_1 \\ \vdots \\ \mathbf{v}_n^{i} \\ \mathbf{f}_n^{i} / m_n \end{bmatrix}$$




```
void EulerStep(ParticleSystem p, float DeltaT){
        ParticleDeriv(p,temp1); /* get deriv */
        ScaleVector(temp1,DeltaT) /* scale it */
        ParticleGetState(p,temp2); /* get state */
        AddVectors(temp1,temp2,temp2); /* add -> temp2 */
        ParticleSetState(p,temp2); /* update state */
        p->t += DeltaT; /* update time */
}
```

Very simple collision response

- How do you decide when you've had a collision?

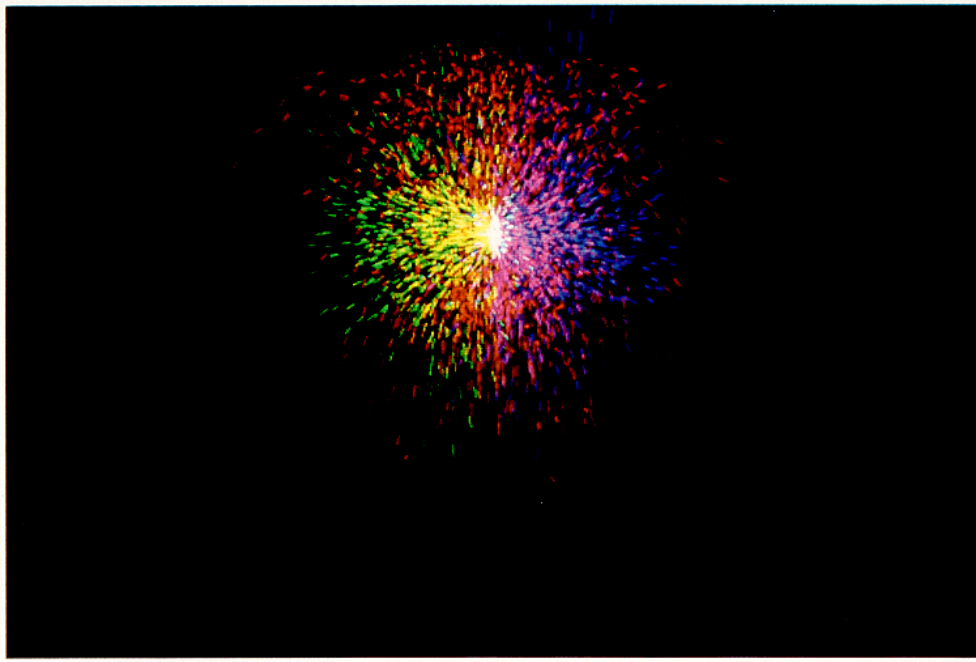


A problem with this approach is that particles will disappear under the surface.

Also, the response may not be enough to bring a particle to the other side of a wall.

Generate Particles

- Particle Attributes
  - initial position,
  - initial velocity (both speed and direction),
  - initial size,
  - initial color,
  - initial transparency,
  - shape,
  - lifetime.



WILLIAM T. REEVES, ACM Transactions on Graphics, Vol. 2, No. 2, April 1983

Generate Particles

- Initial Particle Distribution

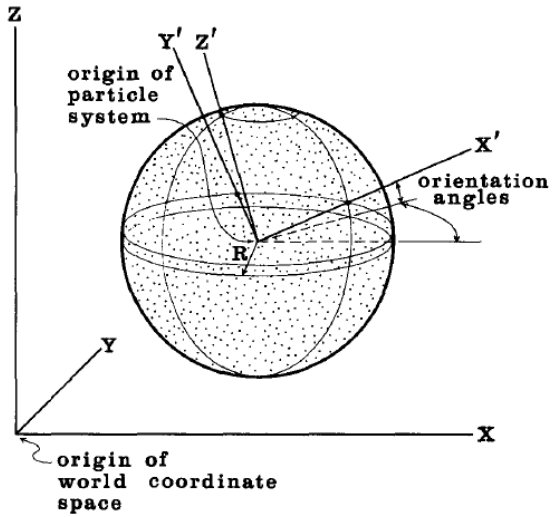
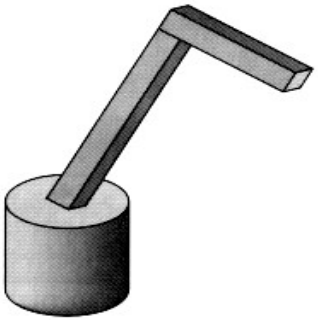


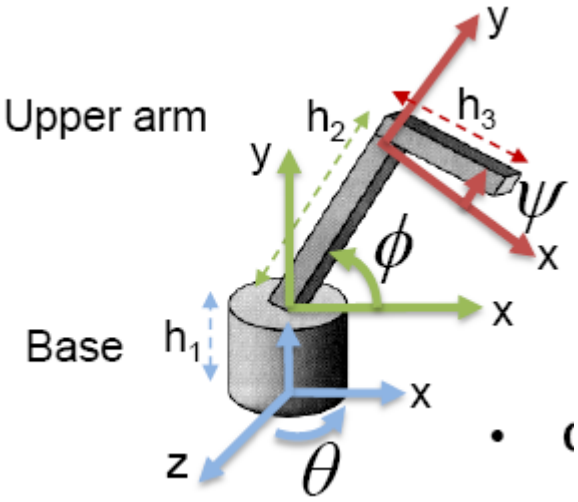
Fig. 1. Typical particle system with spherical generation shape.

- Particle hierarchy, for example
  - Skyrocket : firework
  - Clouds : water drops

Throwing a ball from a robot arm

- Let's say we had our robot arm example and we wanted to launch particles from its tip.





- How would we calculate initial speed?

  Q=R(theta)*T1*R(phi)*T2*R(psi)*P

  We want dQ/dt

Principles of Animation

- The following are a set of principles to keep in mind:
  1. Squash and stretch
  2. Staging
  3. Timing
  4. Anticipation
  5. Follow through
  6. Secondary action
  7. Straight-ahead vs. pose-to-pose vs. blocking
  8. Arcs
  9. Slow in, slow out
  10. Exaggeration
  11. Appeal

Squash and stretch (cont'd)

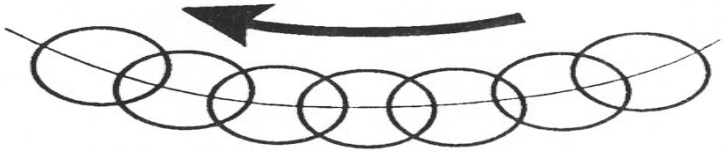


FIGURE 4a. In slow action, an object's position overlaps from frame to frame which gives the action a smooth appearance to the eye.

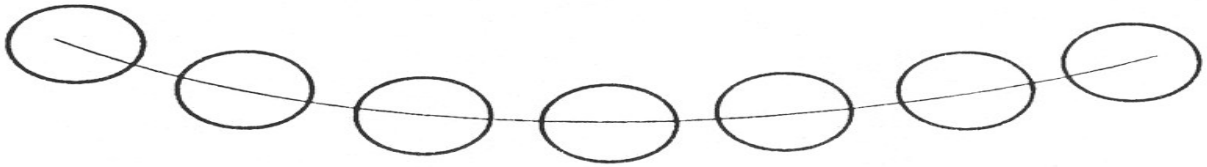


FIGURE 4b. Strobing occurs in a faster action when the object's positions do not overlap and the eye perceives seperate images.

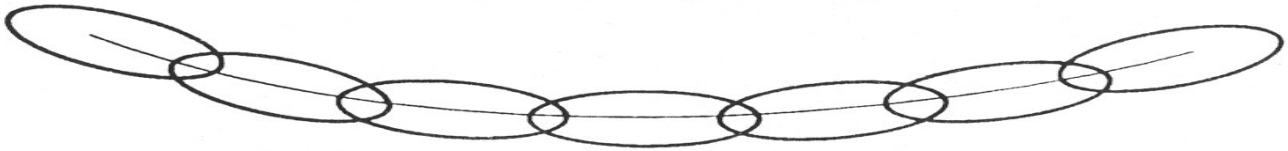
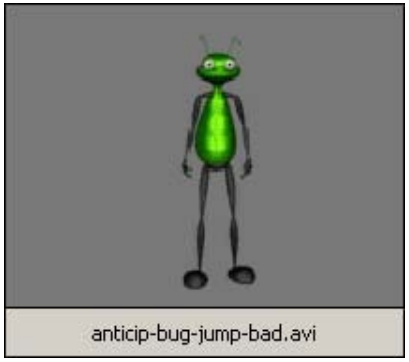
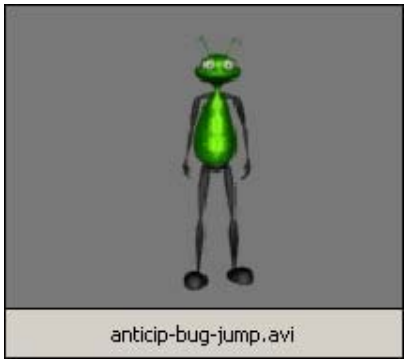


FIGURE 4c. Stretching the object so that it's positions overlap again will relieve the strobing effect.

Anticipation

- An action has three parts: anticipation, action, reaction.
- Anatomical motivation: a muscle must extend before it can contract.

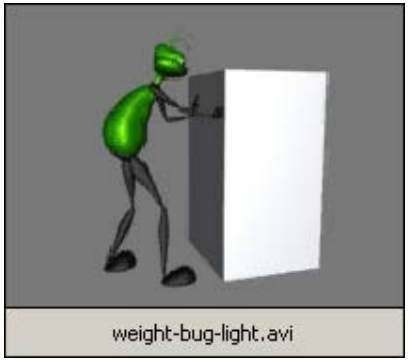


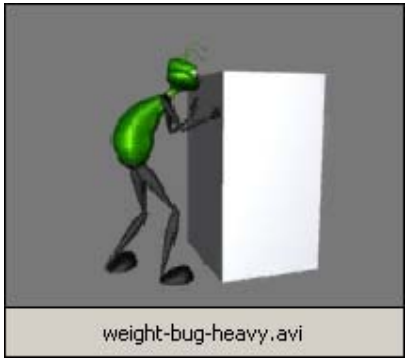


- Watch: bugs-bunny.virtualdub.new.mpg
- Prepares audience for action so they know what to expect.
- Directs audience's attention.

Anticipation (cont'd)

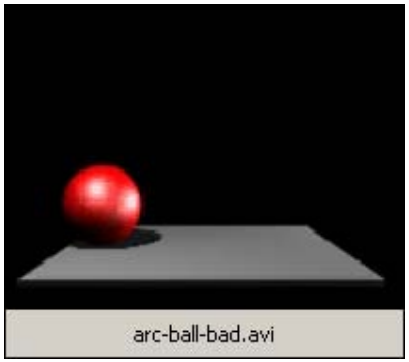
- Amount of anticipation (combined with timing) can affect perception of speed or weight.

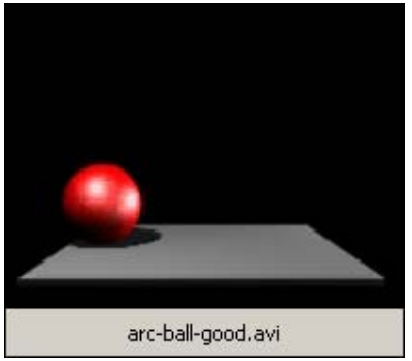




Arcs

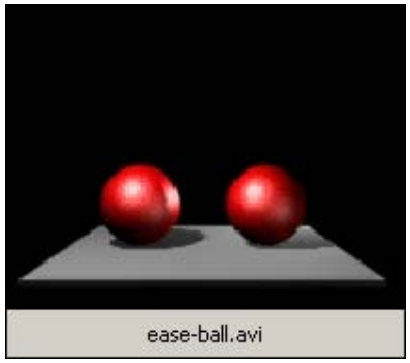
- Avoid straight lines since most things in nature move in arcs.

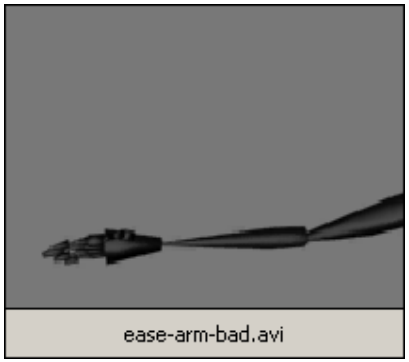


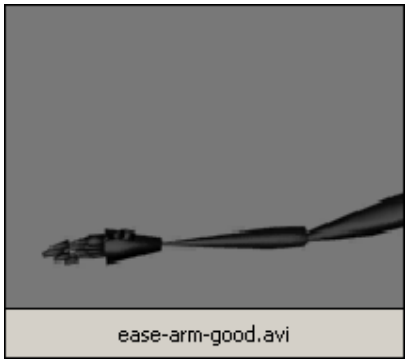


Slow in and slow out

- An extreme pose can be emphasized by slowing down as you get to it (and as you leave it).
- In practice, many things do not move abruptly but start and stop gradually.

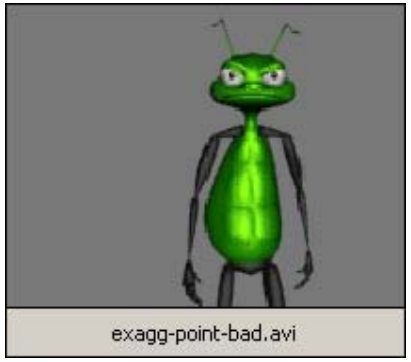


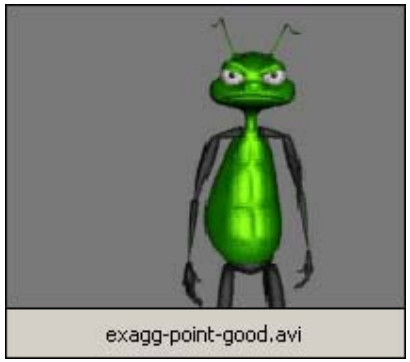




Exaggeration

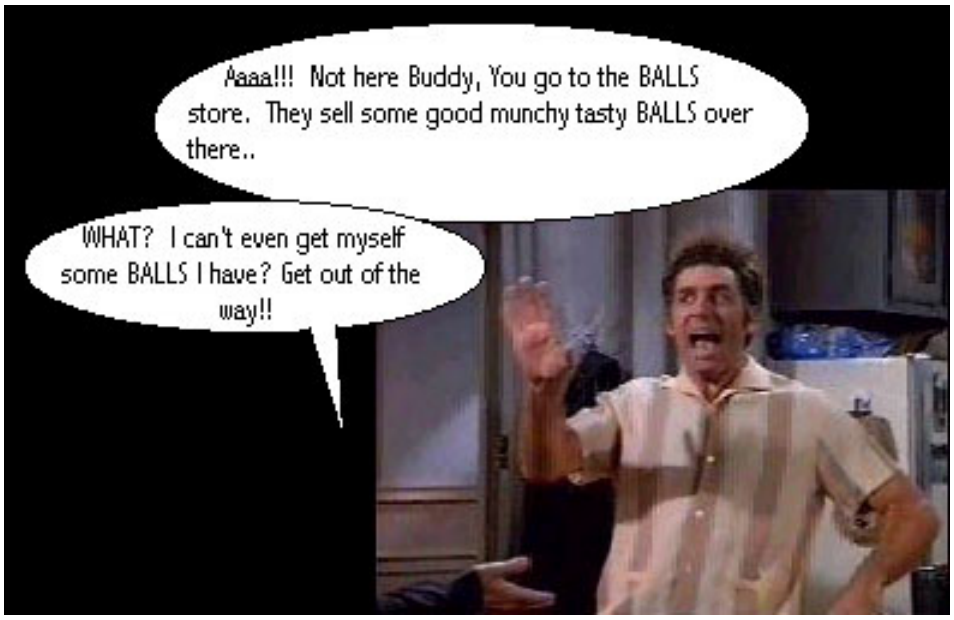
- Get to the heart of the idea and emphasize it so the audience can see it.





Exaggeration

- Get to the heart of the idea and emphasize it so the audience can see it.



Appeal

- The character must interest the viewer.
- It doesn't have to be cute and cuddly.
- Design, simplicity, behavior all affect appeal.
- Example: Luxo, Jr. is made to appear childlike.

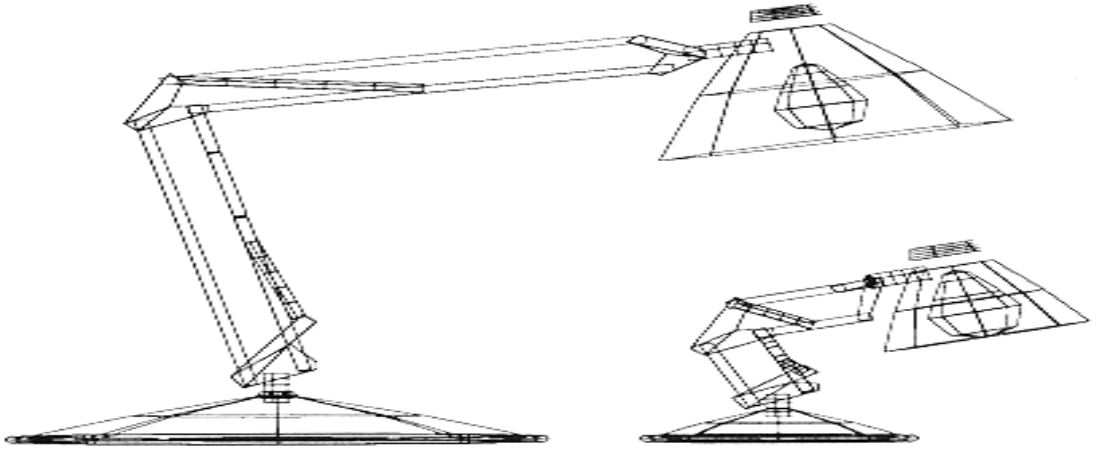


FIGURE 11. Varying the scale of different parts of Dad created the child-like proportions of Luxo Jr.

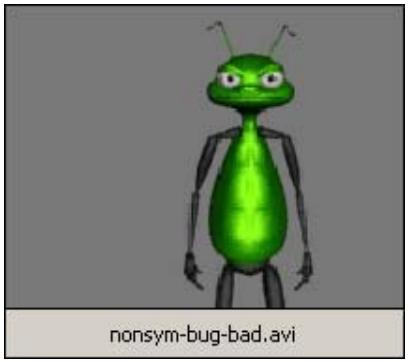
http://www.youtube.com/watch?v=HDuRXvtImQ0&feature=related

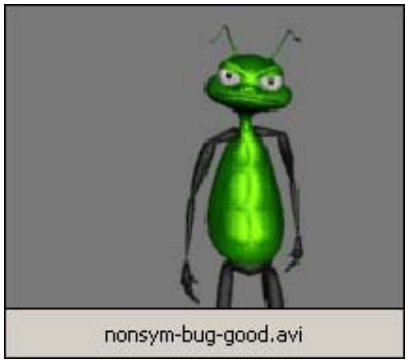
Appeal (cont'd)

- Note: avoid perfect symmetries.

Appeal (cont’d)

- Note: avoid perfect symmetries.





Ray Tracing

Shirley Ch 4

Slides are from Ravi Ramamoorthi's graphics class at Columbia U

Effects needed for Realism

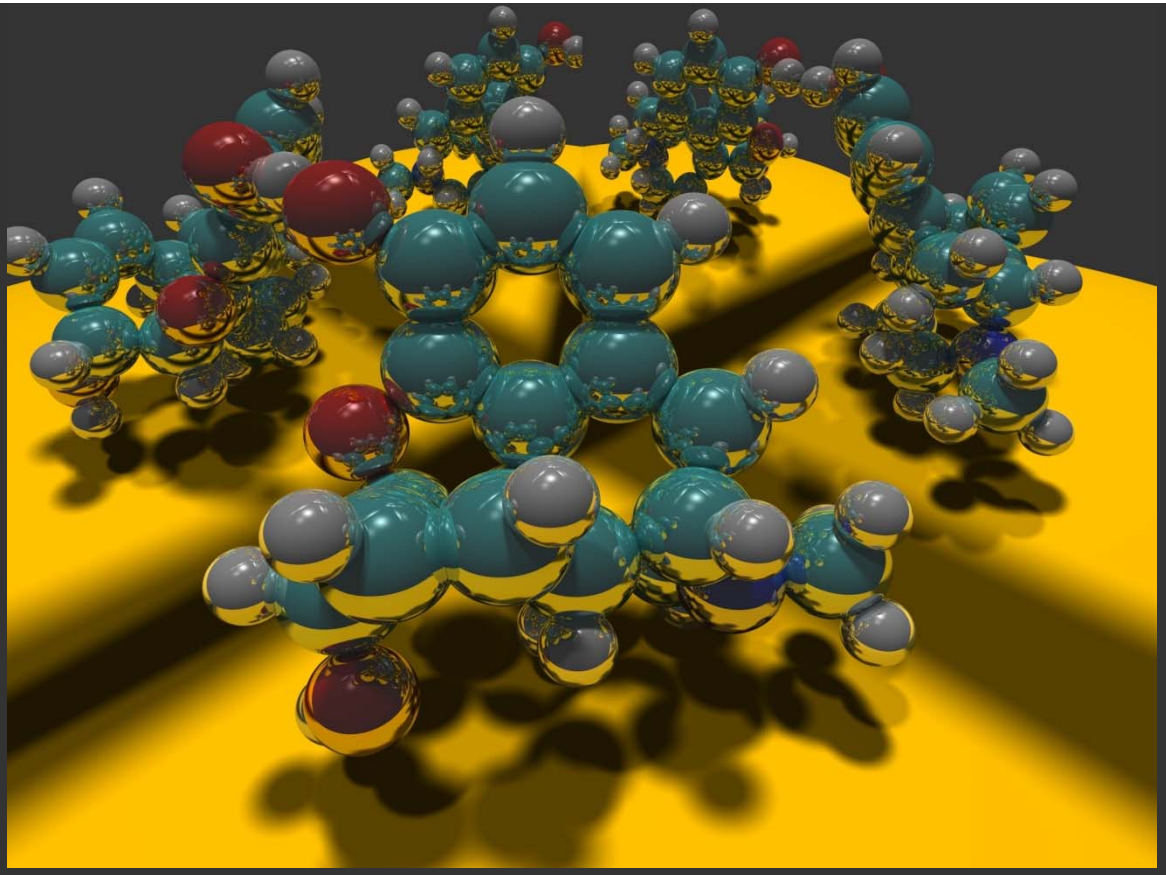


Image courtesy
Paul Heckbert 1983

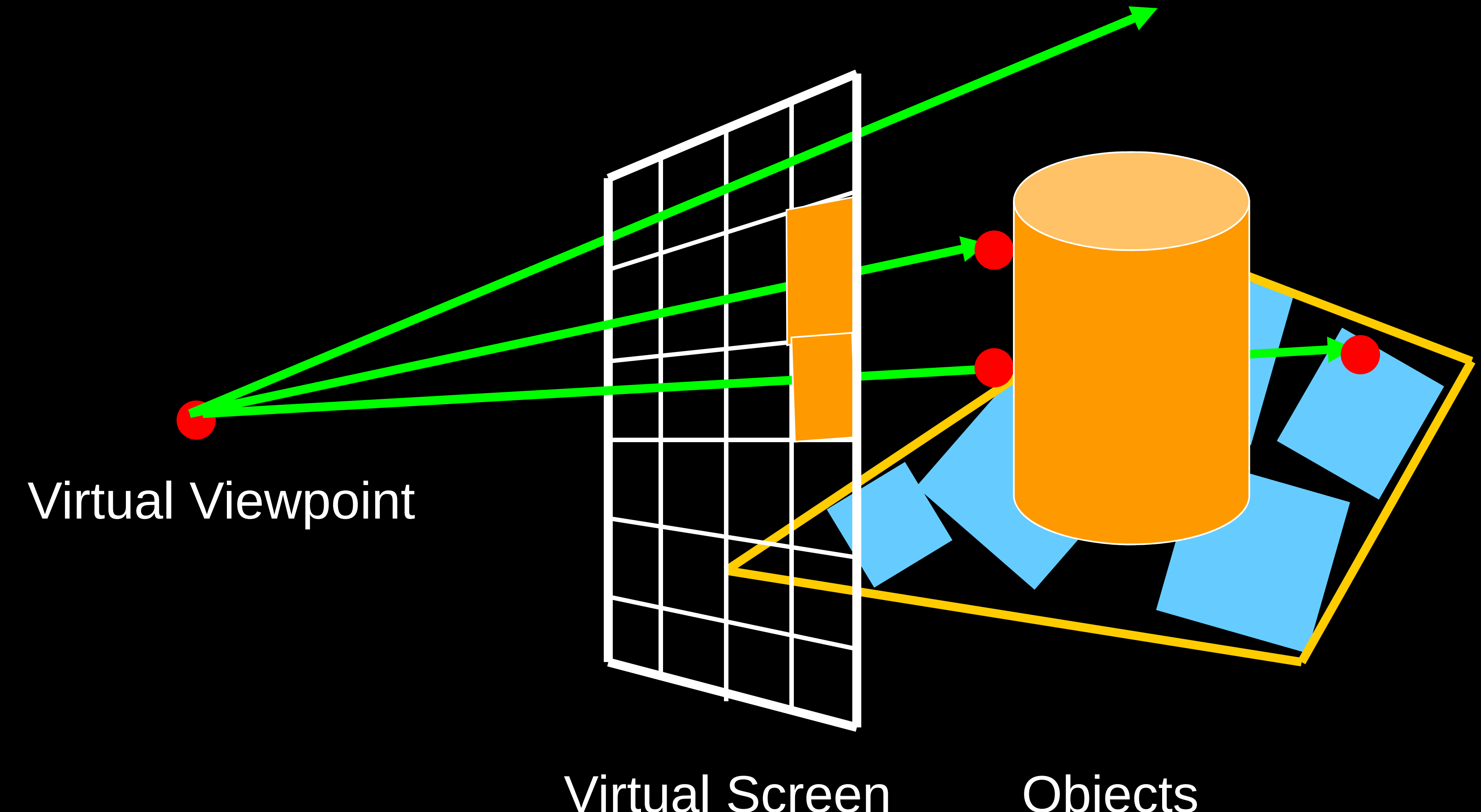
- Reflections (Mirrors and Glossy)
- Transparency (Water, Glass)
- Interreflections (Color Bleeding)
- (Soft) Shadows

- Complex Illumination (Natural, Area Light)
- Realistic Materials (Velvet, Paints, Glass)
- And many more

Ray Tracing

- Different Approach to Image Synthesis as compared to Hardware pipeline (OpenGL)
  - OpenGL : Object by Object
  - Ray Tracing : Pixel by Pixel
- Advantage:
  - Easy to compute shadows/transparency/etc
- Disadvantage:
  - Slow (in early days)

Basic Version: Ray Casting



Ray intersects object: shade using color, lights, materials

Multiple intersections: Use closest one (as does OpenGL)

Ray Casting

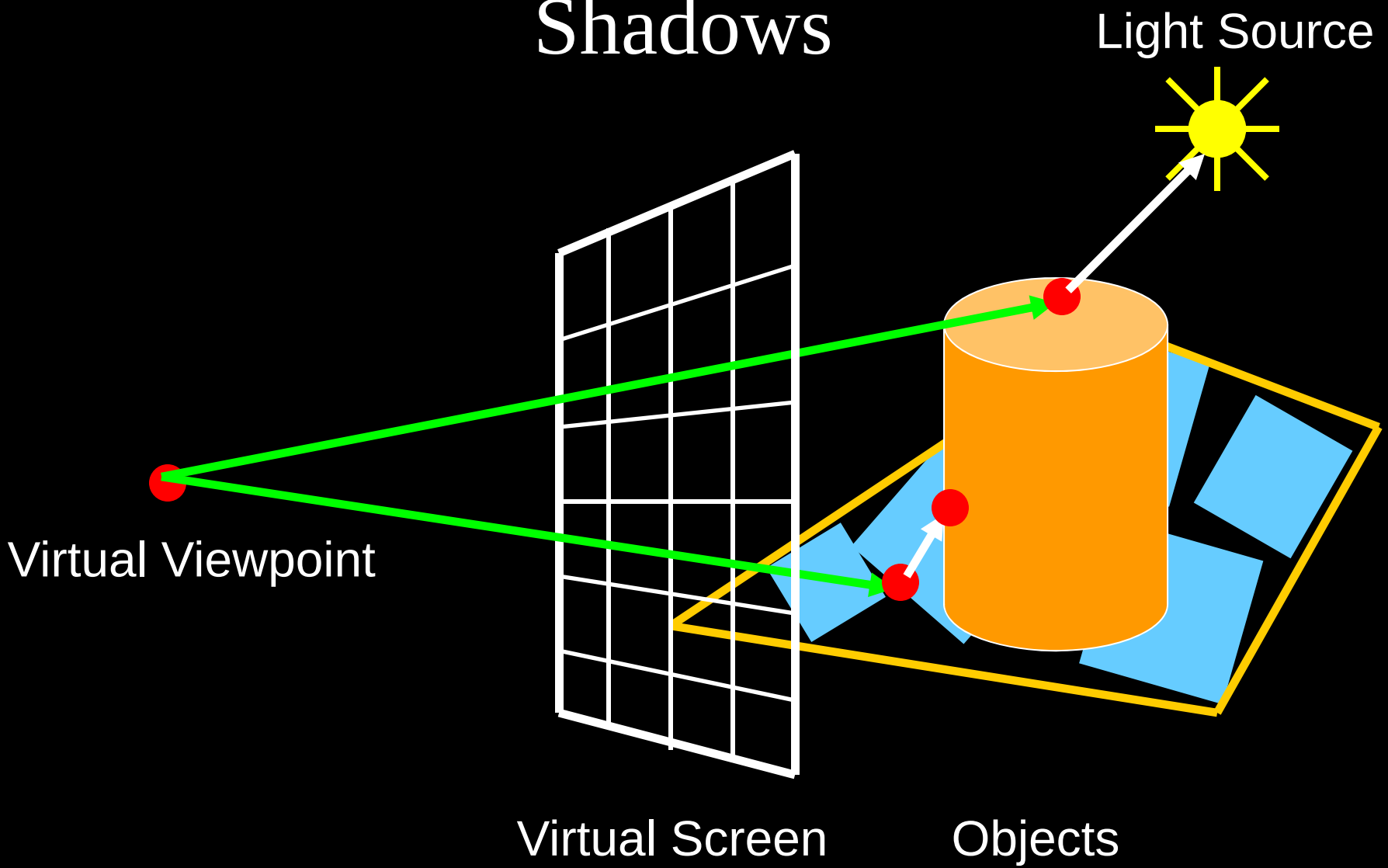
Produce same images as with OpenGL

- Visibility per pixel instead of Z-buffer
- Find nearest object by shooting rays into scene
- Shade it as in standard OpenGL

Comparison to hardware scan-line

- Per-pixel evaluation, per-pixel rays (not scan-convert each object). On face of it, costly
- But good for walkthroughs of extremely large models (amortize preprocessing, low complexity)
- More complex shading, lighting effects possible

Shadows



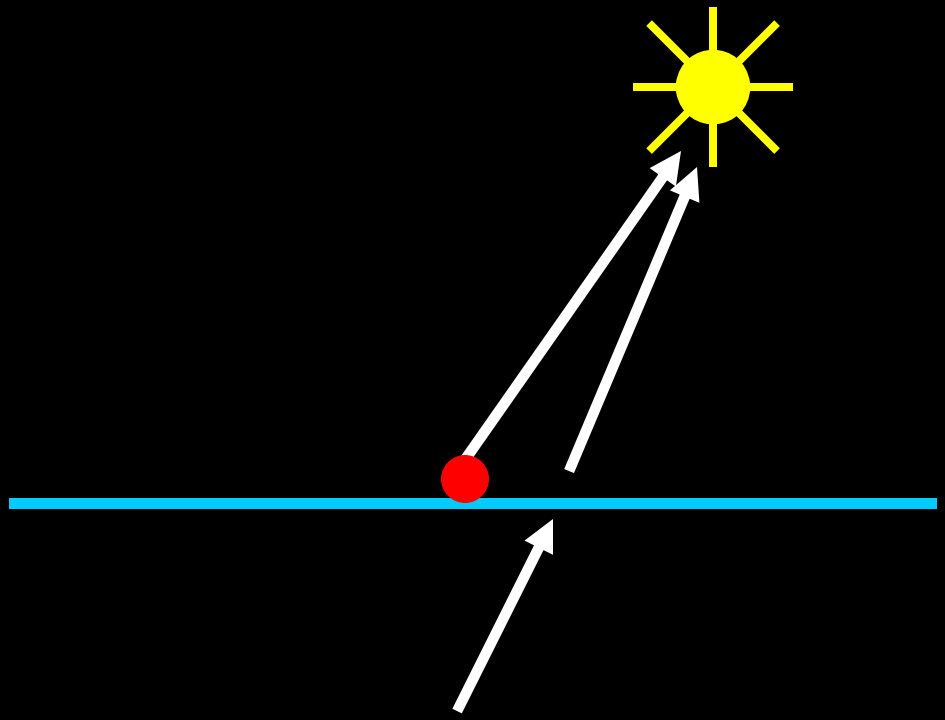
Shadow ray to light is blocked by object → shadow

Shadow ray to light is not blocked by object → visible

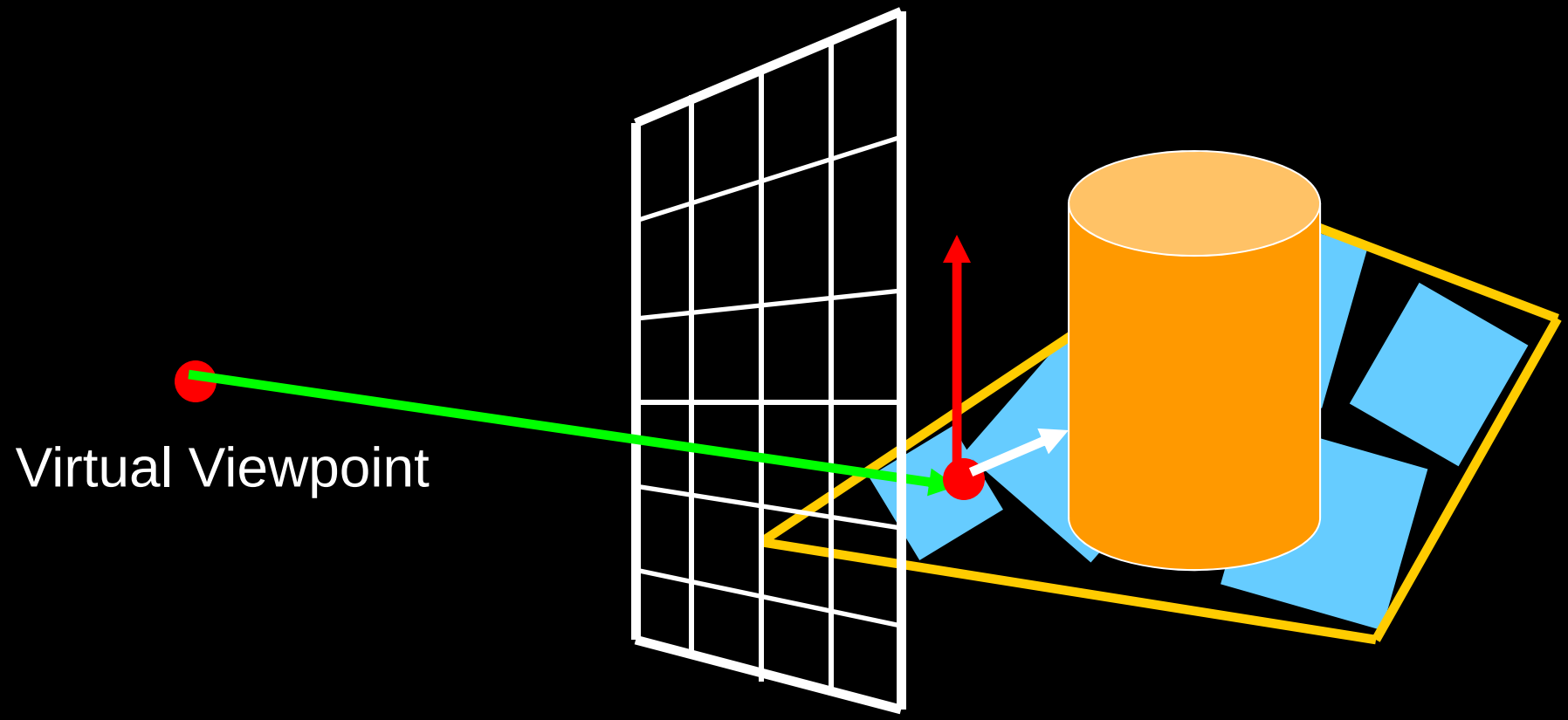
10.5 in textbook

Shadows: Numerical Issues

- Numerical inaccuracy may cause intersection to be below surface (effect exaggerated in figure)
- Causing surface to incorrectly shadow itself
- Move a little towards light before shooting shadow ray



Mirror Reflections/Refractions

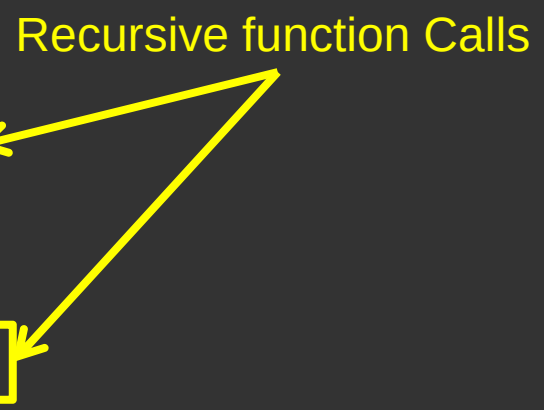


Generate reflected ray in mirror direction,
Get reflections and refractions of objects

Recursive Ray Tracing (Core Idea)

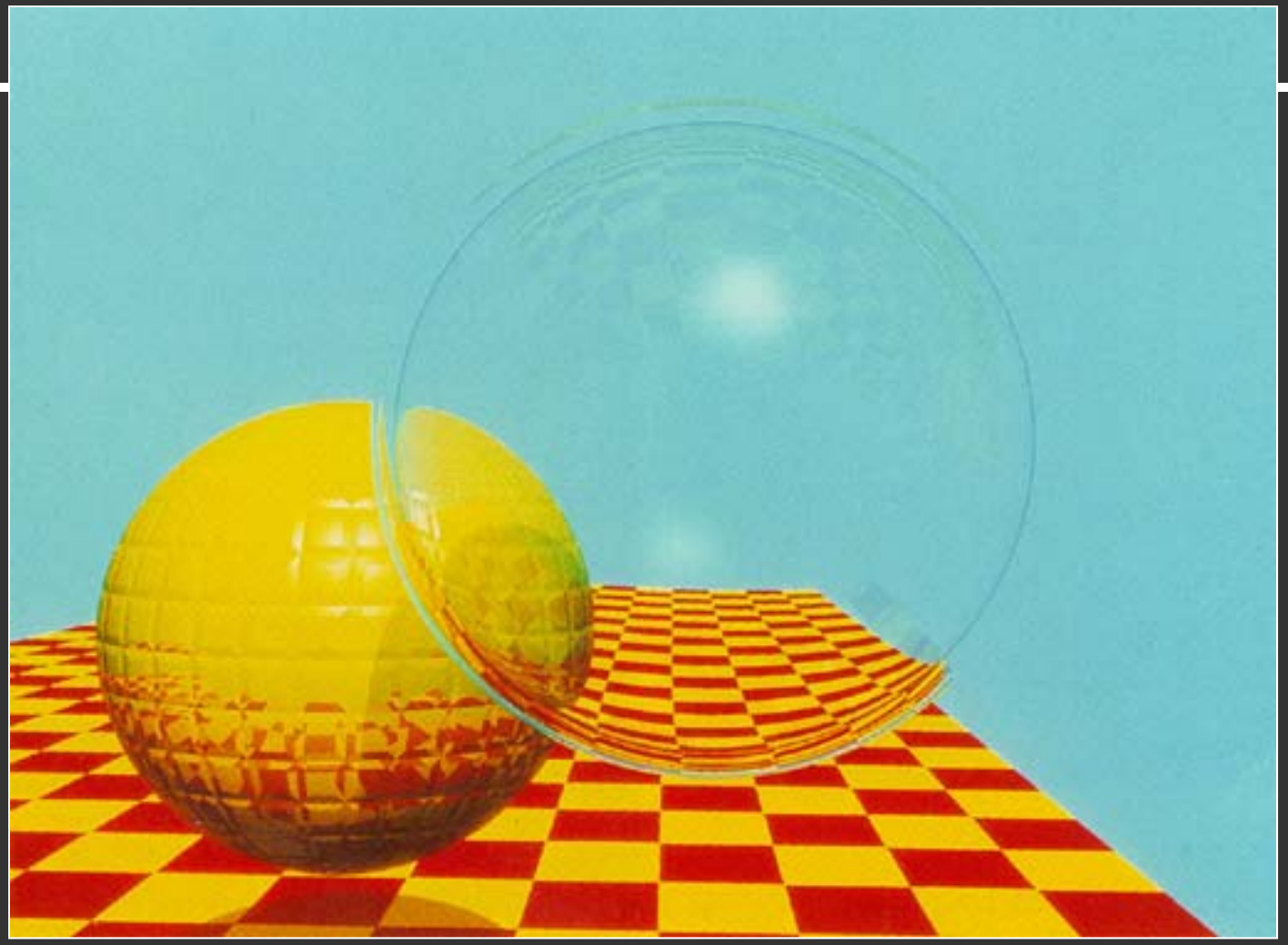
For each pixel

- Trace Primary Eye Ray, find intersection
- Trace Secondary Shadow Ray(s) to all light(s)
  - Color = Visible1 ? Illumination Model(light1) : 0 ;
  - Color += Visible2 ? Illumination Model(light2) : 0 ;
  - …
- Trace Reflected Ray
  - Color += reflectivity * Color of reflected ray
- Trace Refracted Ray
  - Color += transparency * Color of refracted ray

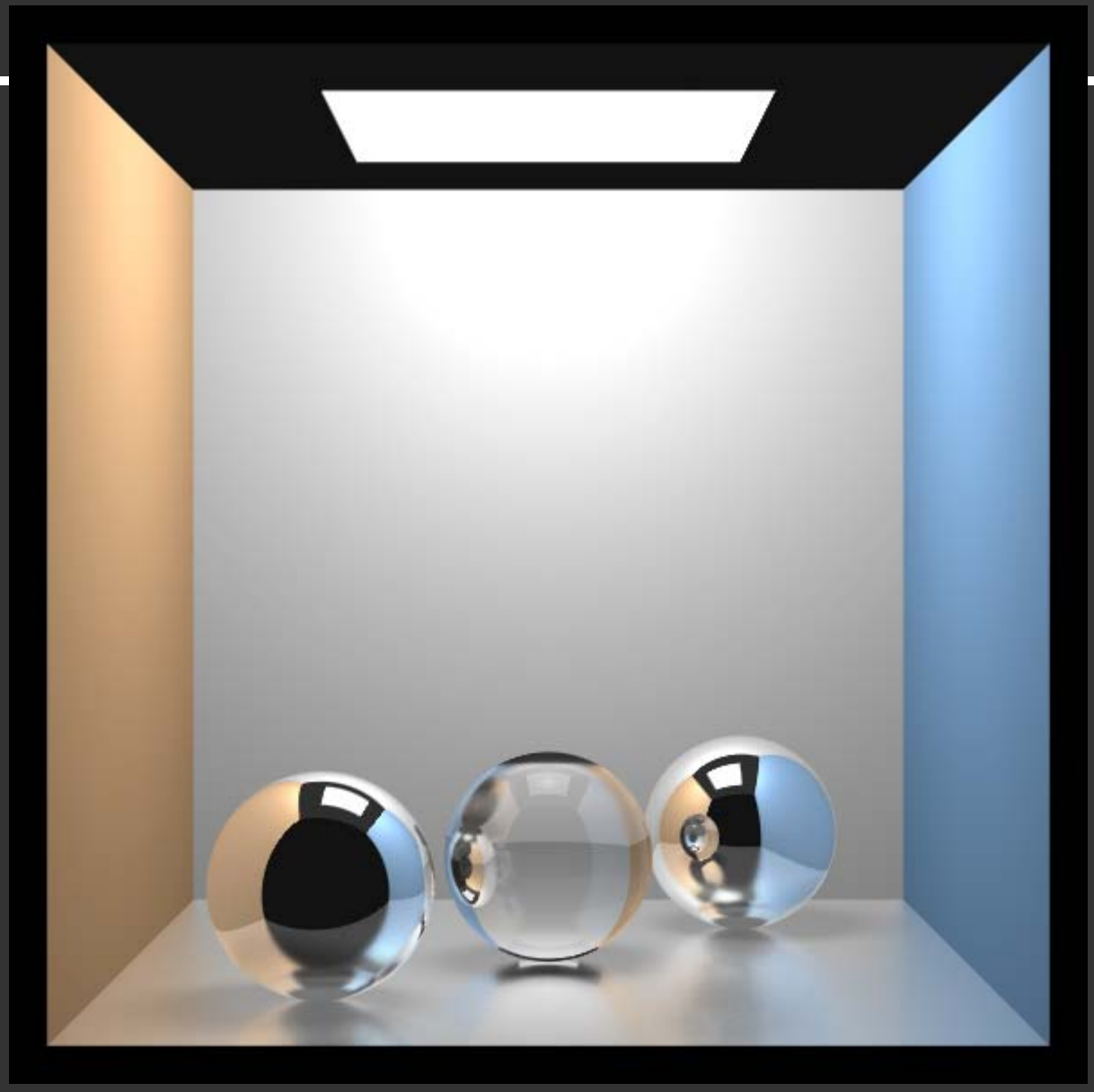


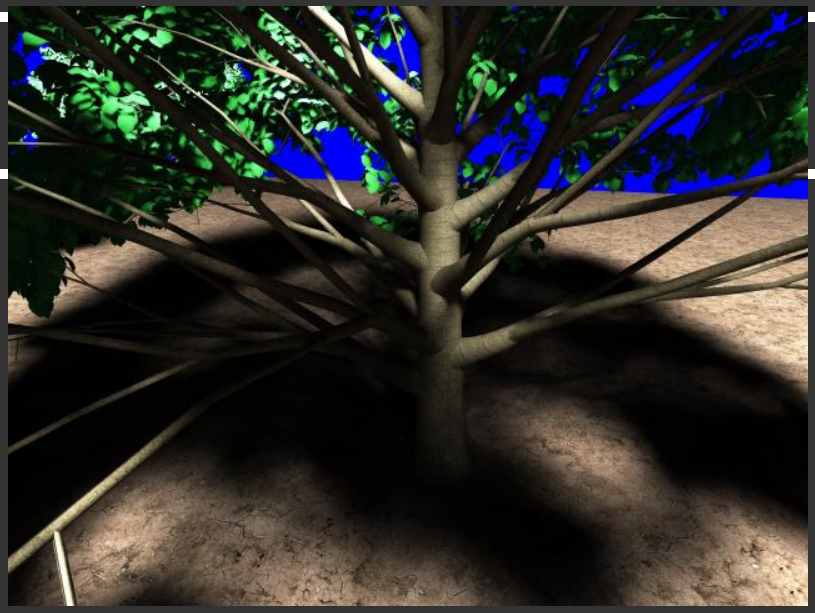
Problems with Recursion

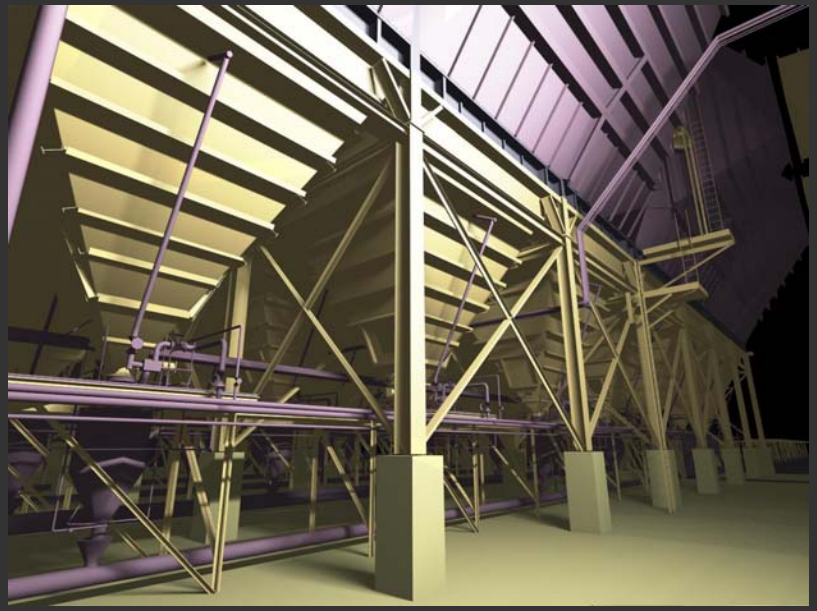
- Reflection rays may be traced forever
- Generally, set maximum recursion depth



Turner Whitted 1980







Effects needed for Realism

- (Soft) Shadows
- Reflections (Mirrors and Glossy)
- Transparency (Water, Glass)
- Interreflections (Color Bleeding)
- Complex Illumination (Natural, Area Light)
- Realistic Materials (Velvet, Paints, Glass)

Discussed in this lecture

Not discussed so far but possible with distribution ray tracing (13.4)

Hard (but not impossible) with ray tracing; radiosity methods

How to implement Ray tracing?

- Ray parameterization
- *Ray-Surface Intersection*

Ray/Object Intersections

- Heart of Ray Tracer
  - One of the main initial research areas
  - Optimized routines for wide variety of primitives
- Various types of info
  - Shadow rays: Intersection/No Intersection
  - Primary rays: Point of intersection, material, normals, Texture coordinates

Example

- Sphere
  - How to decide there is an intersection?
- Triangle
  - How to decide the intersection is inside?
- Polygon
  - How to decide the intersection is inside?

- How about an ellipsoid?

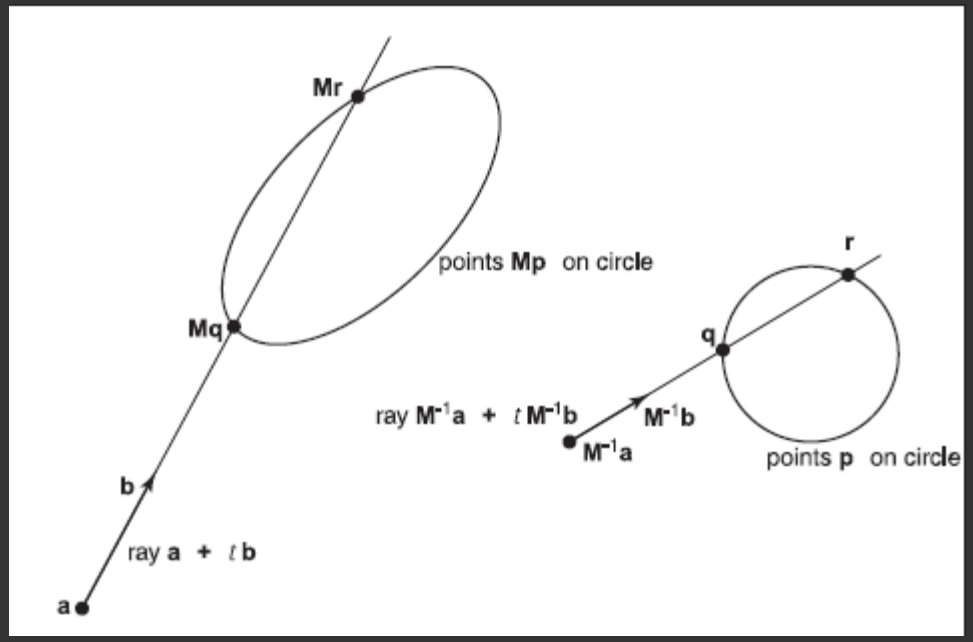
Ray-Tracing Transformed Objects

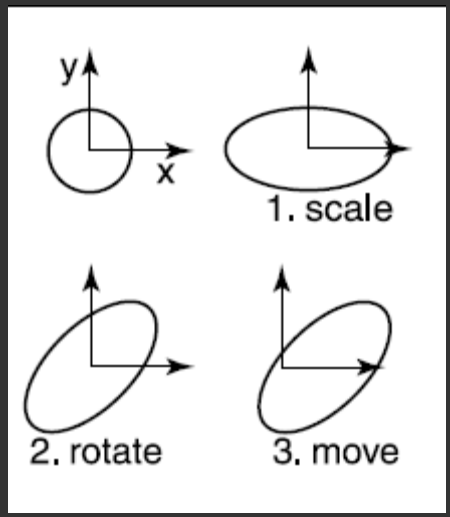
We have an optimized ray-sphere test

- But we want to ray trace an ellipsoid…

Solution: Ellipsoid transforms sphere

- Apply inverse transform to ray, use ray-sphere





Acceleration

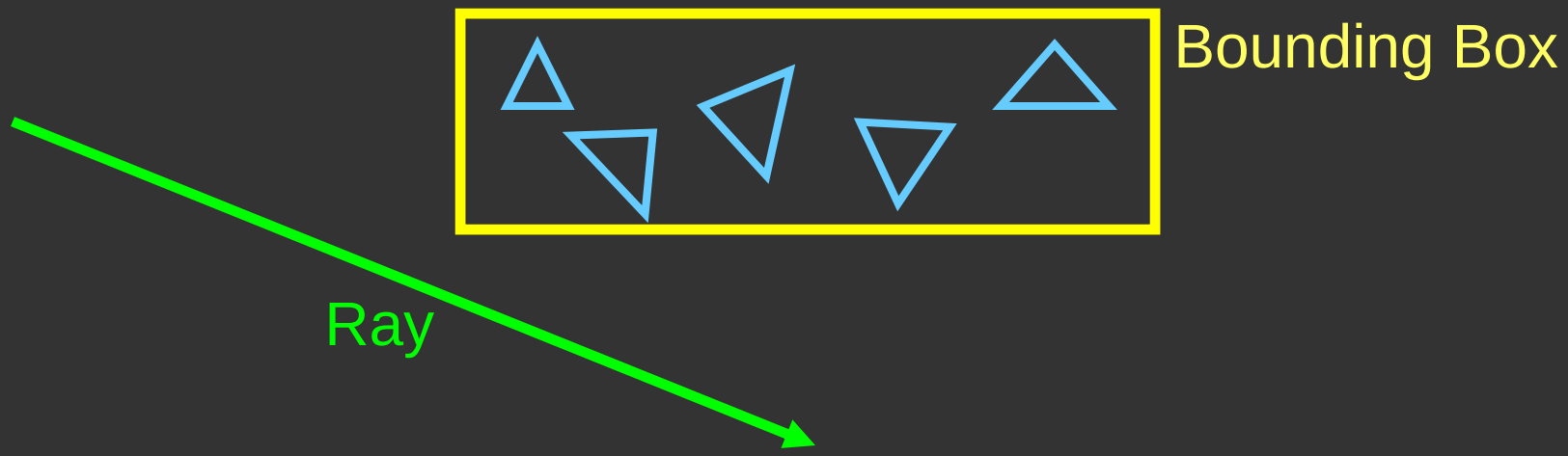
Testing each object for each ray is slow

- Faster Intersections
  - Optimized Ray-Object Intersections
  - ***Fewer Intersections***

Acceleration Structures

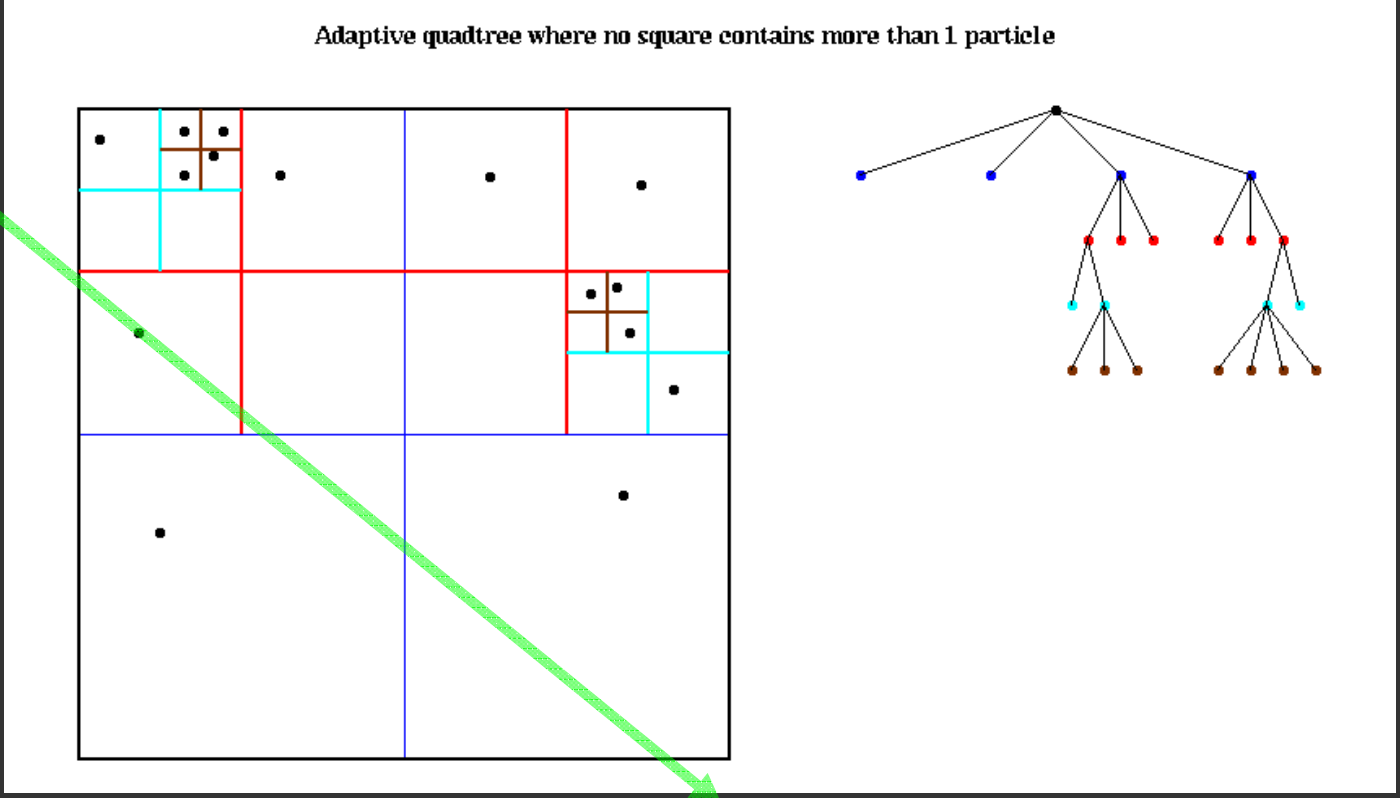
Bounding boxes (possibly hierarchical)

If no intersection bounding box, needn't check objects

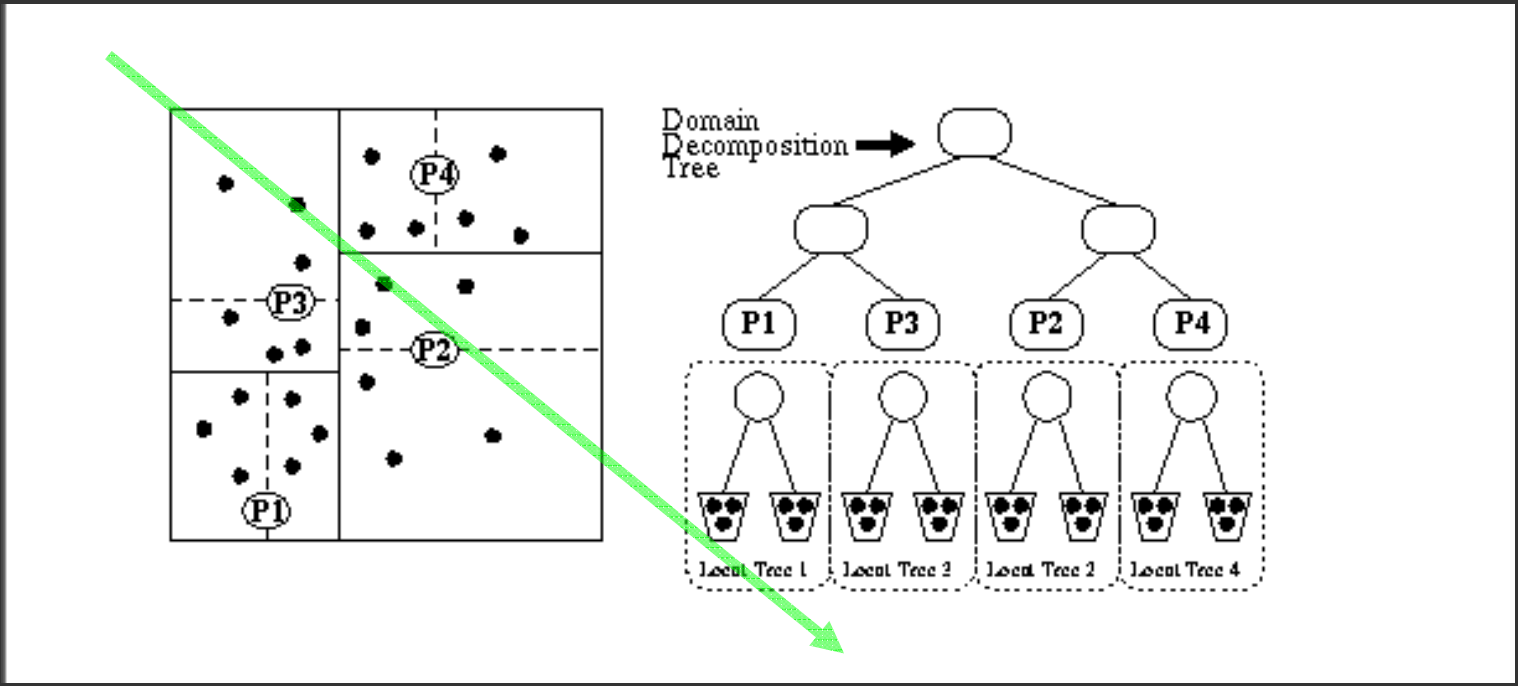


Different Spatial Hierarchies (Oct-trees, kd trees, BSP trees)

Octree


Adaptive quadtree where no square contains more than 1 particle

K-d tree


P4
P3
P2
P1
Domain
Decomposition
Tree
P1
P3
P2
P4
Local Tree 1
Local Tree 2
Local Tree 3
Local Tree 4