CS559: Computer Graphics

Lecture 36: Animation

Li Zhang

Spring 2008

Slides from Brian Curless at U of Washington

Today

- Particle Systems, Cartoon animation, ray tracing

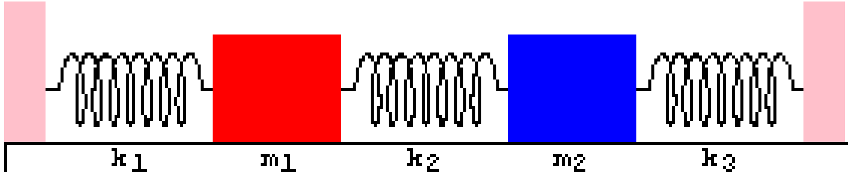
- Reading
  - (Optional) John Lasseter. Principles of traditional animation applied to 3D computer animation. Proceedings of SIGGRAPH (Computer Graphics) 21(4): 35-44, July 1987.

    http://portal.acm.org/citation.cfm?id=37407
  - (Optional) WILLIAM T. REEVES, ACM Transactions on Graphics, Vol. 2, No. 2, April 1983

    http://portal.acm.org/citation.cfm?id=357320

Particle system diff. eq. solver

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

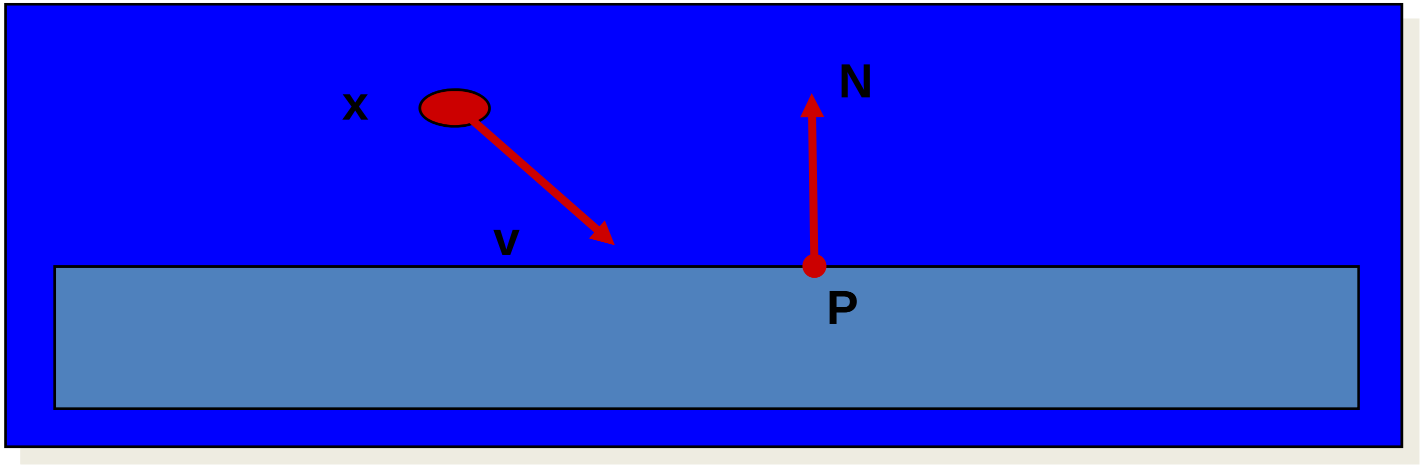


$$\begin{bmatrix} \mathbf{x}_1^{i+1} \\ \mathbf{v}_1^{i+1} \\ \vdots \\ \mathbf{x}_n^{i+1} \\ \mathbf{v}_n^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{i} \\ \mathbf{v}_1^{i} \\ \vdots \\ \mathbf{x}_n^{i} \\ \mathbf{v}_n^{i} \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{v}_1^{i} \\ \mathbf{f}_1^{i}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

```
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 */
}
```

Bouncing off the walls

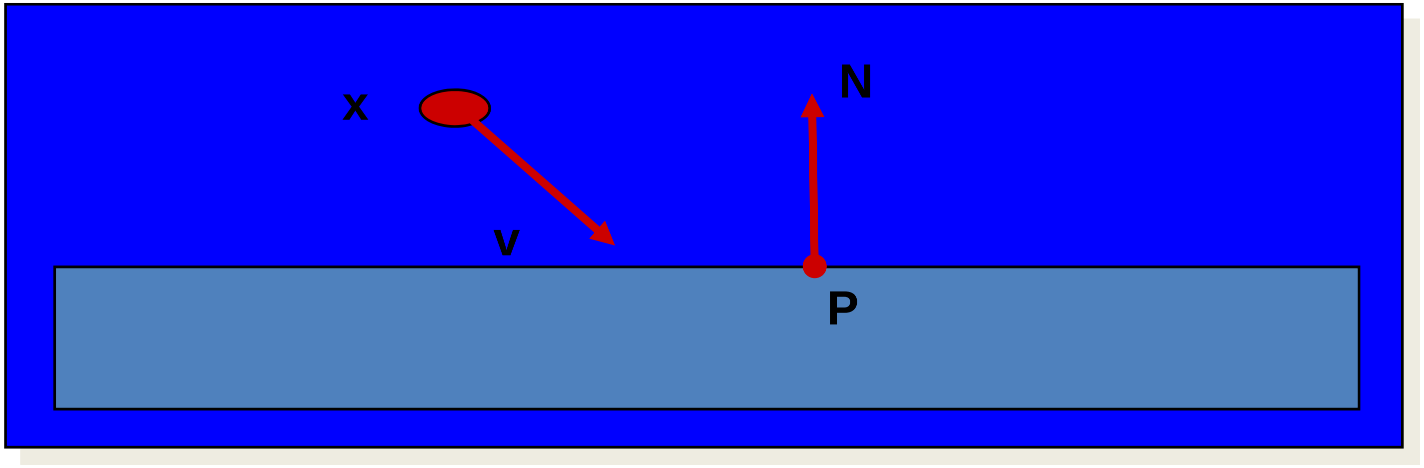
- Handling collisions is a useful add-on for a particle simulator.
- For now, we'll just consider simple point-plane collisions.



A plane is fully specified by any point **P** on the plane and its normal **N**.

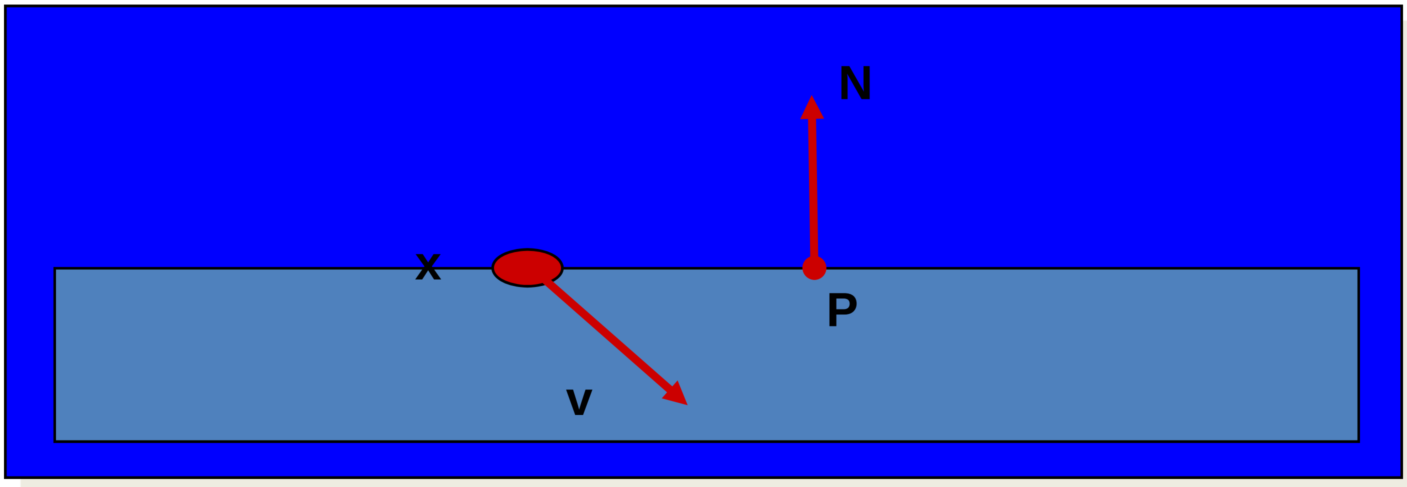
Collision Detection

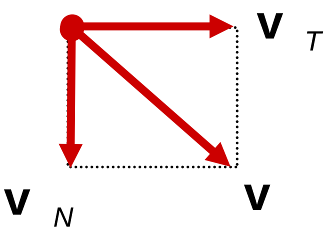
How do you decide when you've made **exact** contact with the plane?



Normal and tangential velocity

To compute the collision response, we need to consider the normal and tangential components of a particle's velocity.

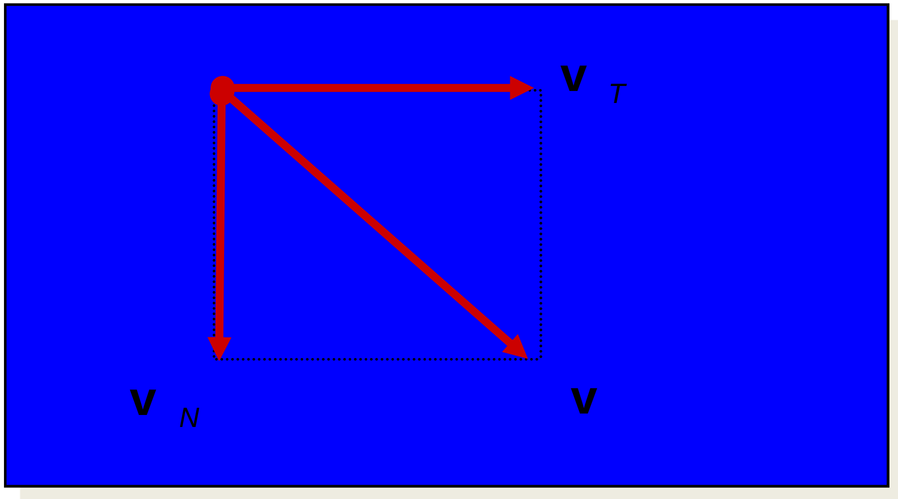




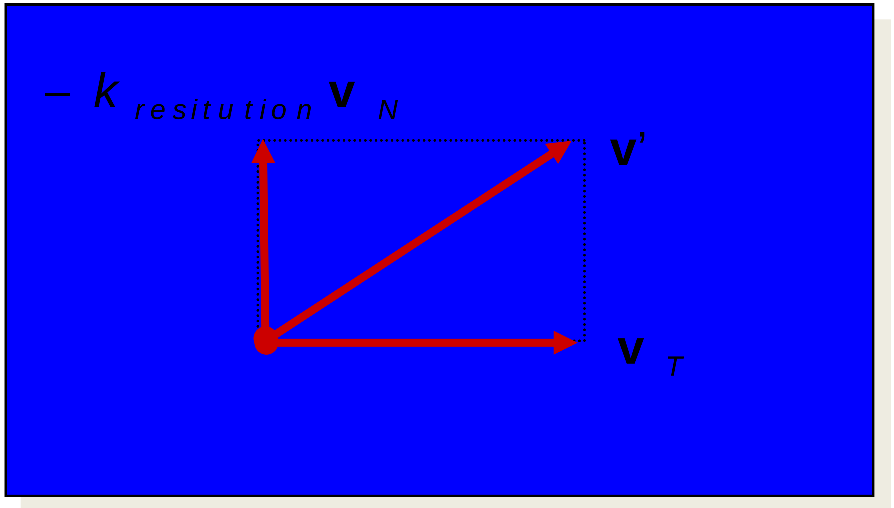
$$\mathbf{v}_N = (\mathbf{N} \cdot \mathbf{v})\mathbf{N}$$

$$\mathbf{v}_T = \mathbf{v} - \mathbf{v}_N$$

Collision Response



before



after

The response to collision is then to immediately replace the current velocity with a new velocity:

$$\mathbf{v}' = \mathbf{v}_T - k_{restitution}\mathbf{v}_N$$

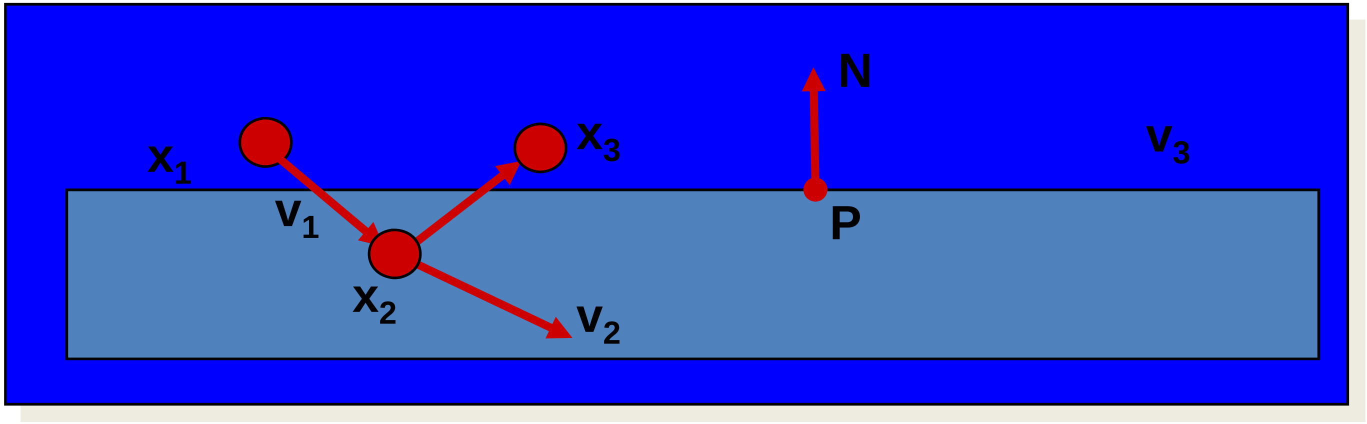
The particle will then move according to this velocity in the next timestep.

Collision without contact

- In general, we don’t sample moments in time when particles are in *exact* contact with the surface.
- There are a variety of ways to deal with this problem.
- A simple alternative is to determine if a collision must have occurred in the past, and then pretend that you’re currently in exact contact.

Very simple collision response

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

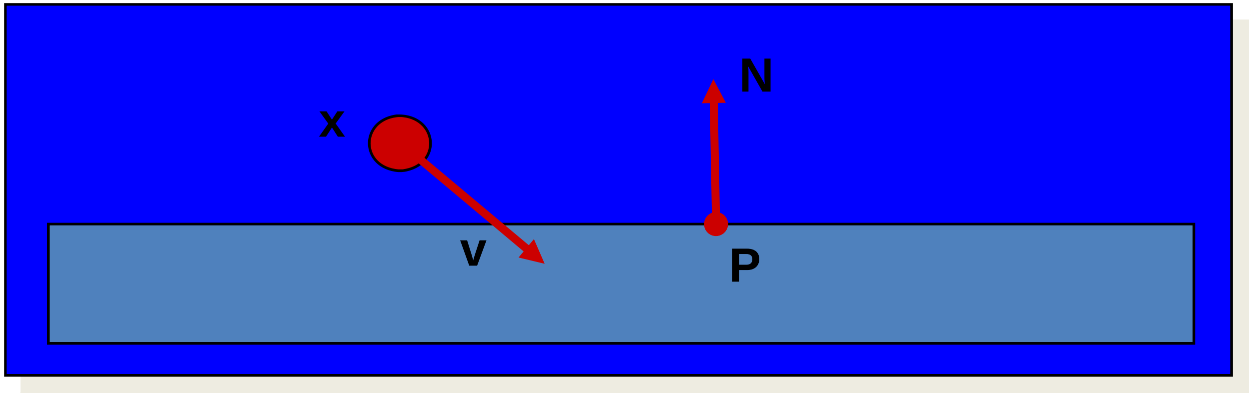


A problem with this approach is that particles will disappear under the surface.

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

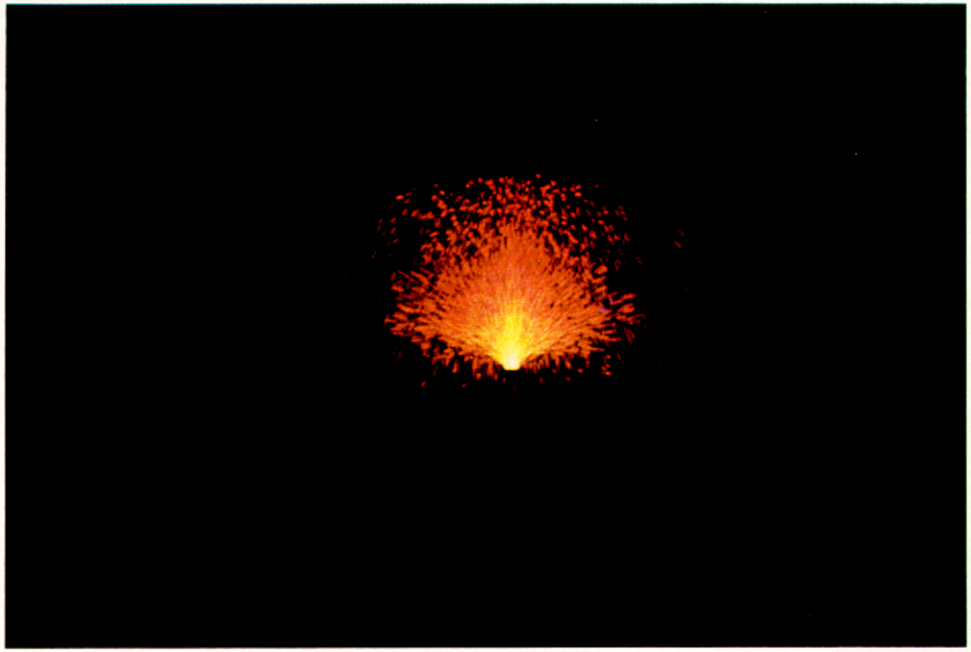
More complicated collision response

- Another solution is to modify the update scheme to:
  - detect the future time and point of collision
  - reflect the particle within the time-step



Generate Particles

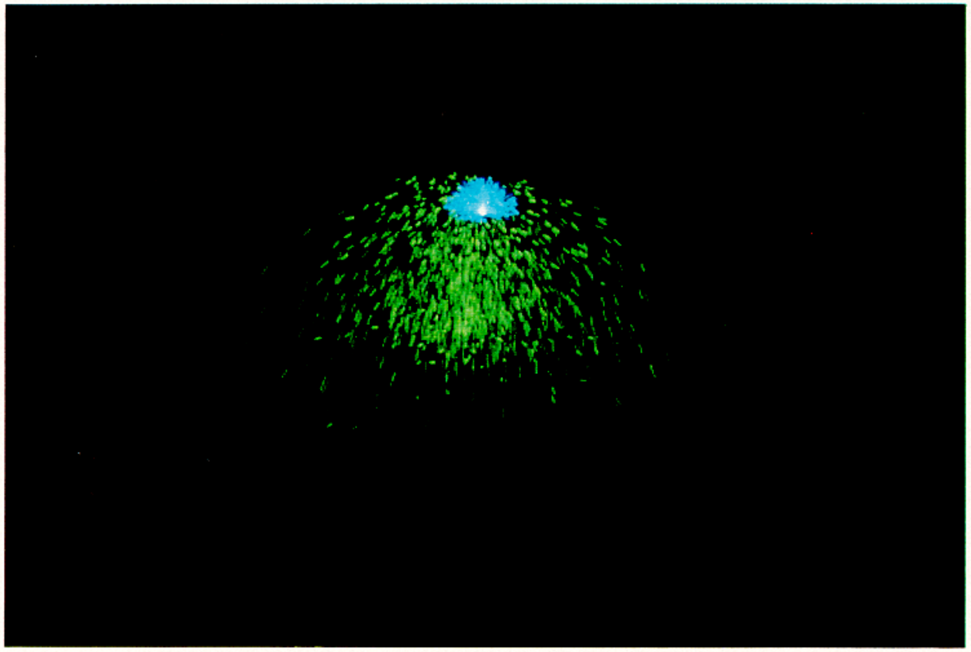
- Particle Attributes
  - initial position,
  - initial velocity (both speed and direction),
  - initial size,
  - initial color,
  - initial transparency,
  - shape,
  - lifetime.



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

Generate Particles

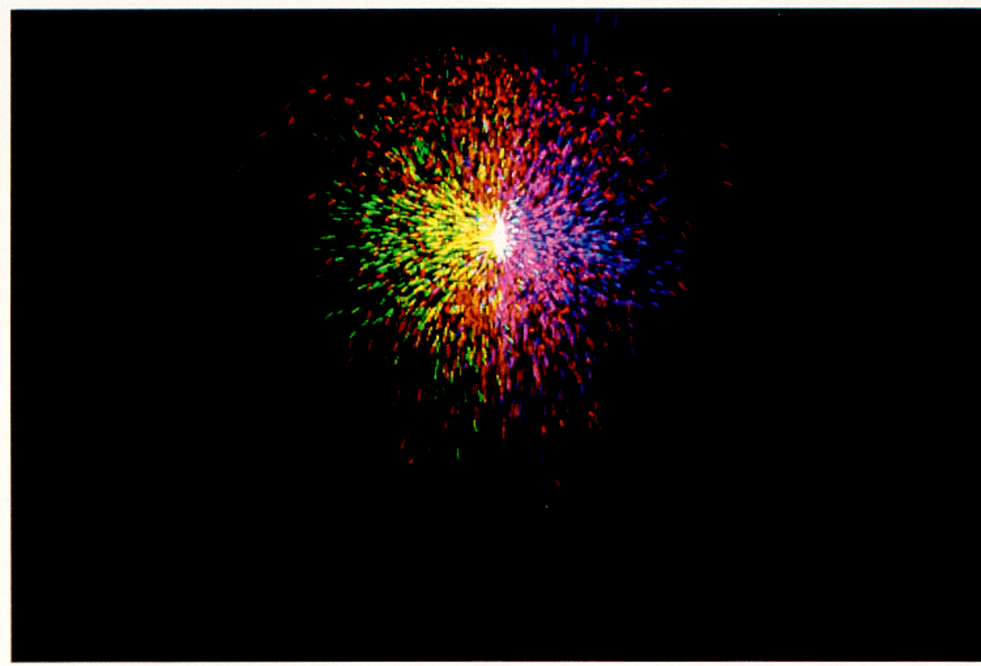
- Particle Attributes
  - initial position,
  - initial velocity (both speed and direction),
  - initial size,
  - initial color,
  - initial transparency,
  - shape,
  - lifetime.



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

Generate Particles

- 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

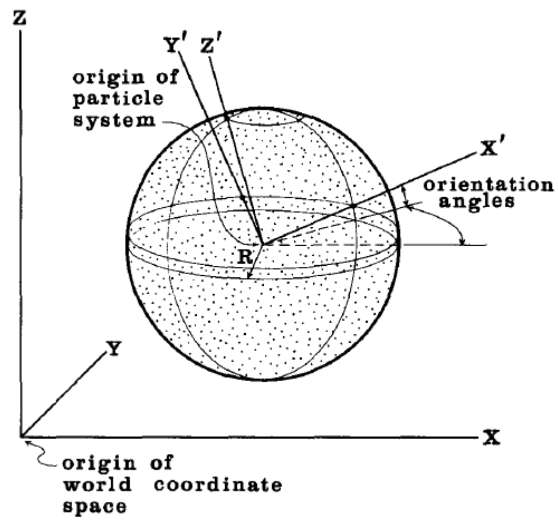
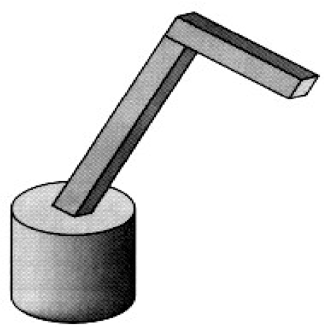


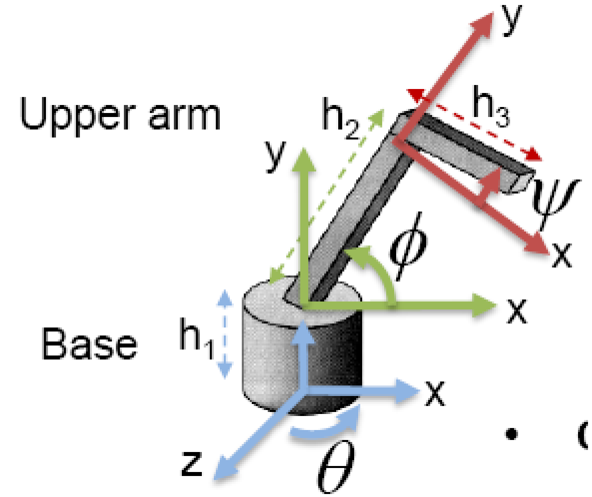
Fig. 1. Typical particle system with spherical generation shape.

- 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

- **Goal**: make characters that move in a convincing way to communicate personality and mood.

- Walt Disney developed a number of principles.
  - ~1930

- Computer graphics animators have adapted them to 3D animation.

John Lasseter. Principles of traditional animation applied to 3D computer animation. Proceedings of SIGGRAPH (Computer Graphics) 21(4): 35-44, July 1987.

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

- **Squash**: flatten an object or character by pressure or by its own power.
- **Stretch**: used to increase the sense of speed and emphasize the squash by contrast.
- Note: keep volume constant!

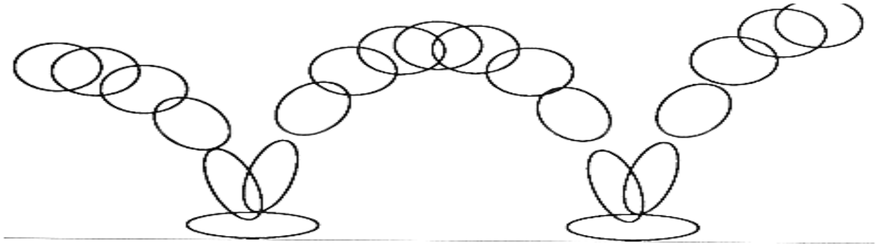


FIGURE 2. Squash & stretch in bouncing ball.

- http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/principles/squash_and_stretch.htm
- http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/principles/bouncing_ball_example_of_slow_in_out.htm

Squash and stretch (cont’d)

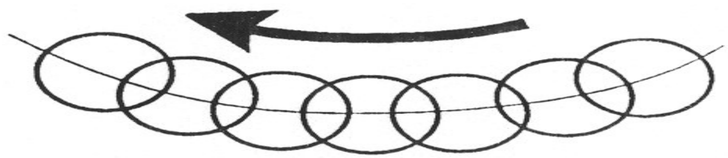


FIGURE 4a. In slow action, an object's position overlaps from frame to frame which gives the action a smooth appearance to the eye.

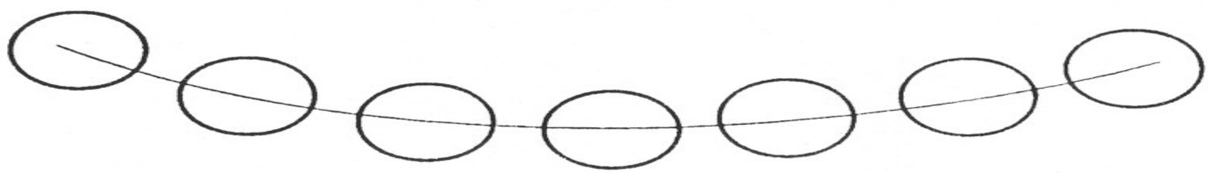


FIGURE 4b. Strobing occurs in a faster action when the object's positions do not overlap and the eye perceives seperate images.

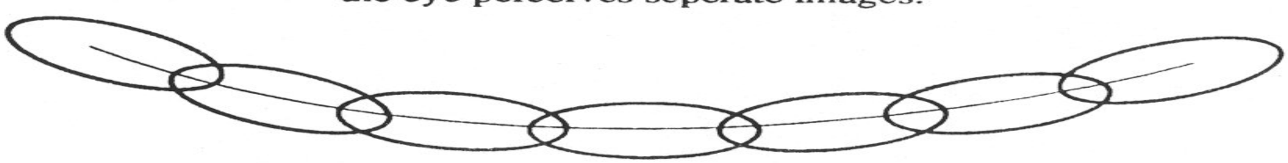
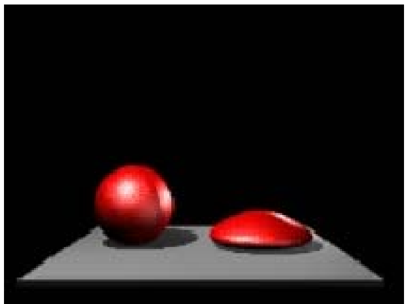
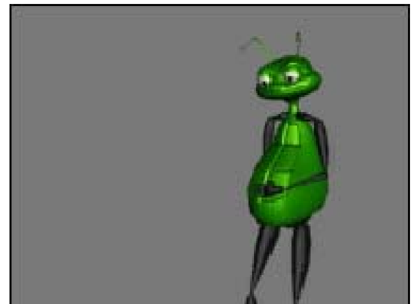


FIGURE 4c. Stretching the object so that it's positions overlap again will relieve the strobing effect.

Squash and stretch (cont'd)



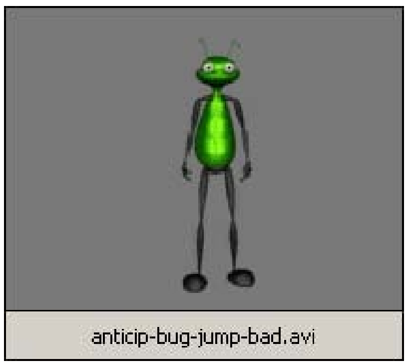
squash-ball.avi

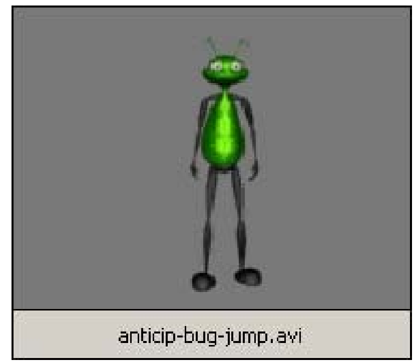


bug-squash-tummy.avi

Anticipation

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

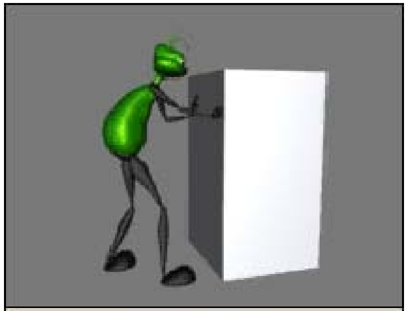




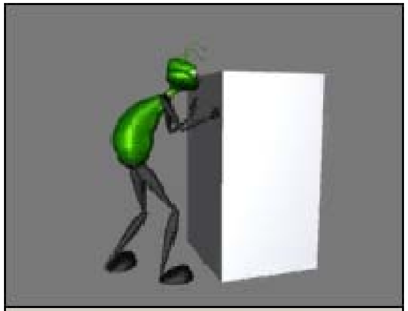
- Watch: bugs-bunny.virtualdub.new.mpg
- Prepares audience for action so they know what to expect.
- Directs audience's attention.

Anticipation (cont'd)

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



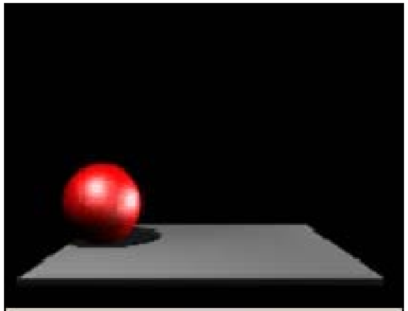
weight-bug-light.avi

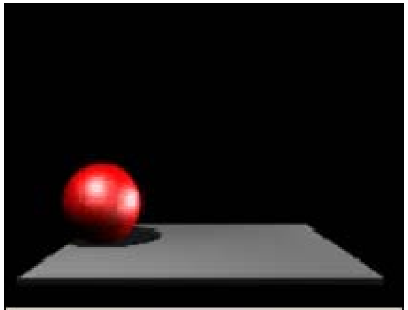


weight-bug-heavy.avi

Arcs

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


arc-ball-bad.avi

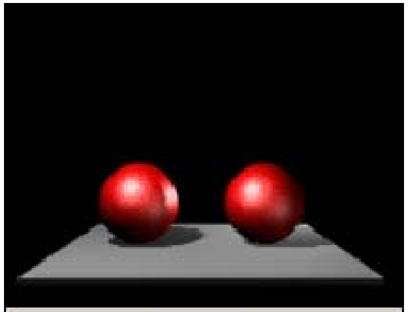

arc-ball-good.avi

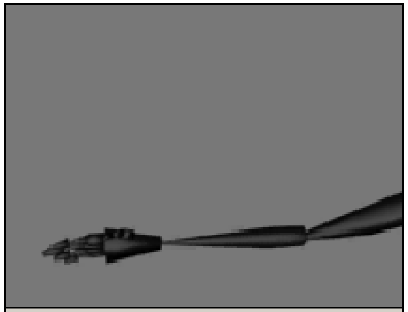

arc-headturn-bad.avi

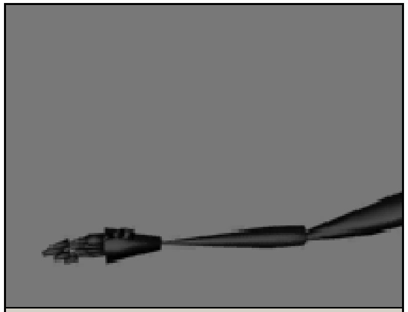

arc-headturn-good.avi

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.

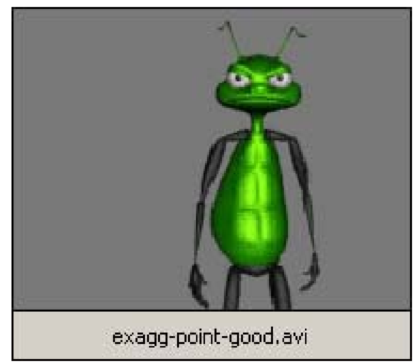

ease-ball.avi


ease-arm-bad.avi


ease-arm-good.avi

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.

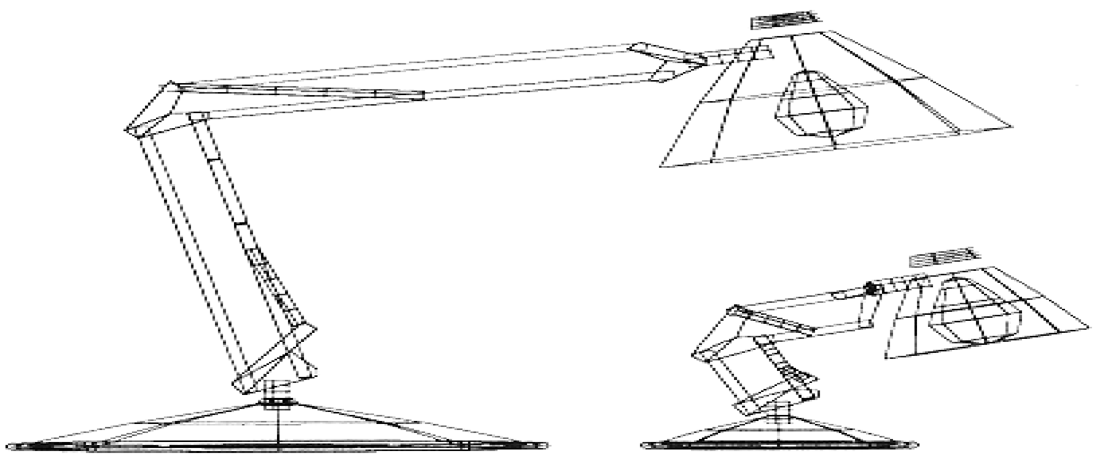
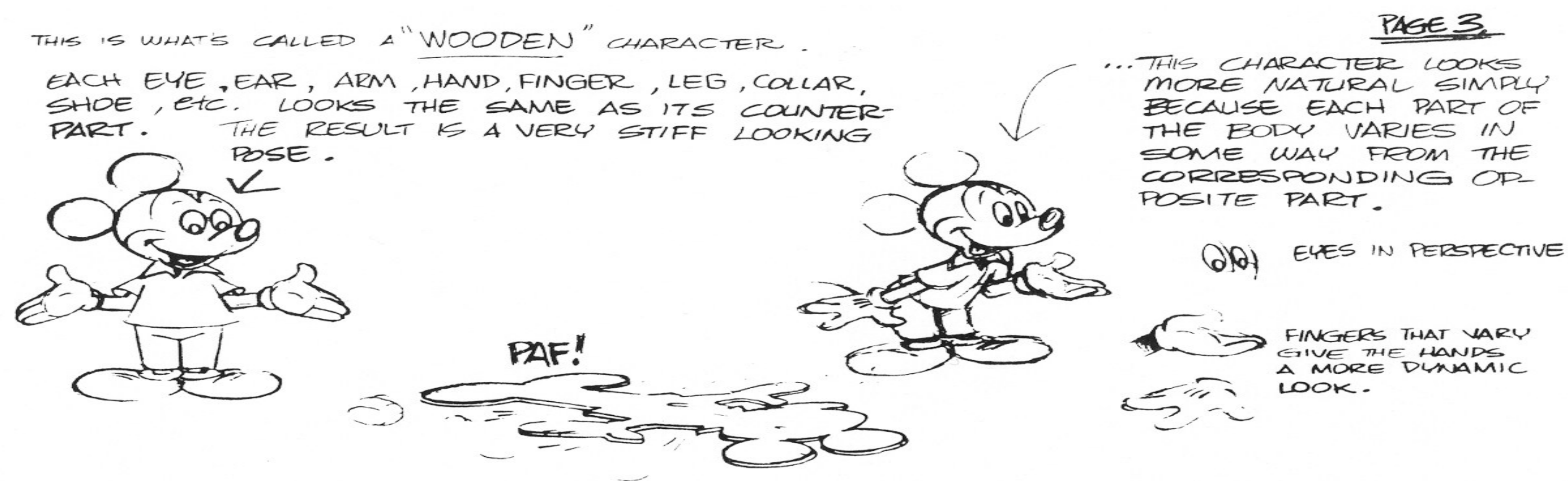


FIGURE 11. Varying the scale of different parts of Dad created the child-like proportions of Luxo Jr.

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

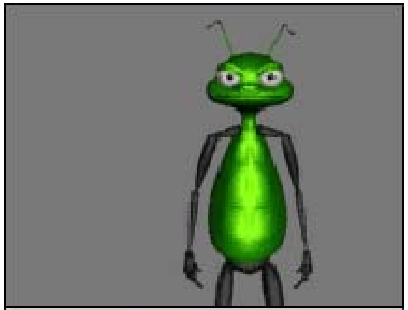
Appeal (cont'd)

- Note: avoid perfect symmetries.

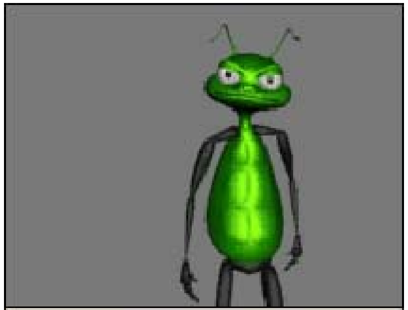


Appeal (cont'd)

- Note: avoid perfect symmetries.



nonsym-bug-bad.avi



nonsym-bug-good.avi

Ray Tracing

Reading: Shirley Ch 10.1 --- 10.8

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

Effects needed for Realism

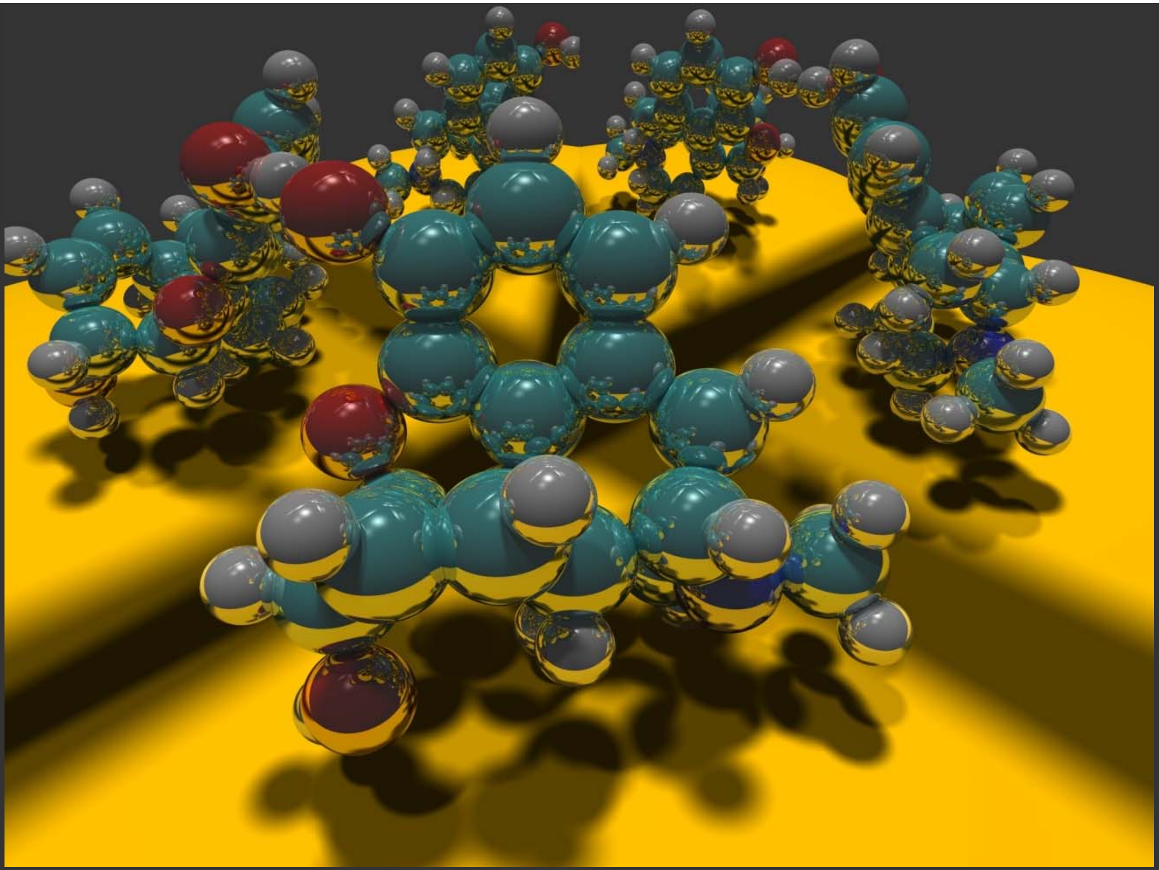


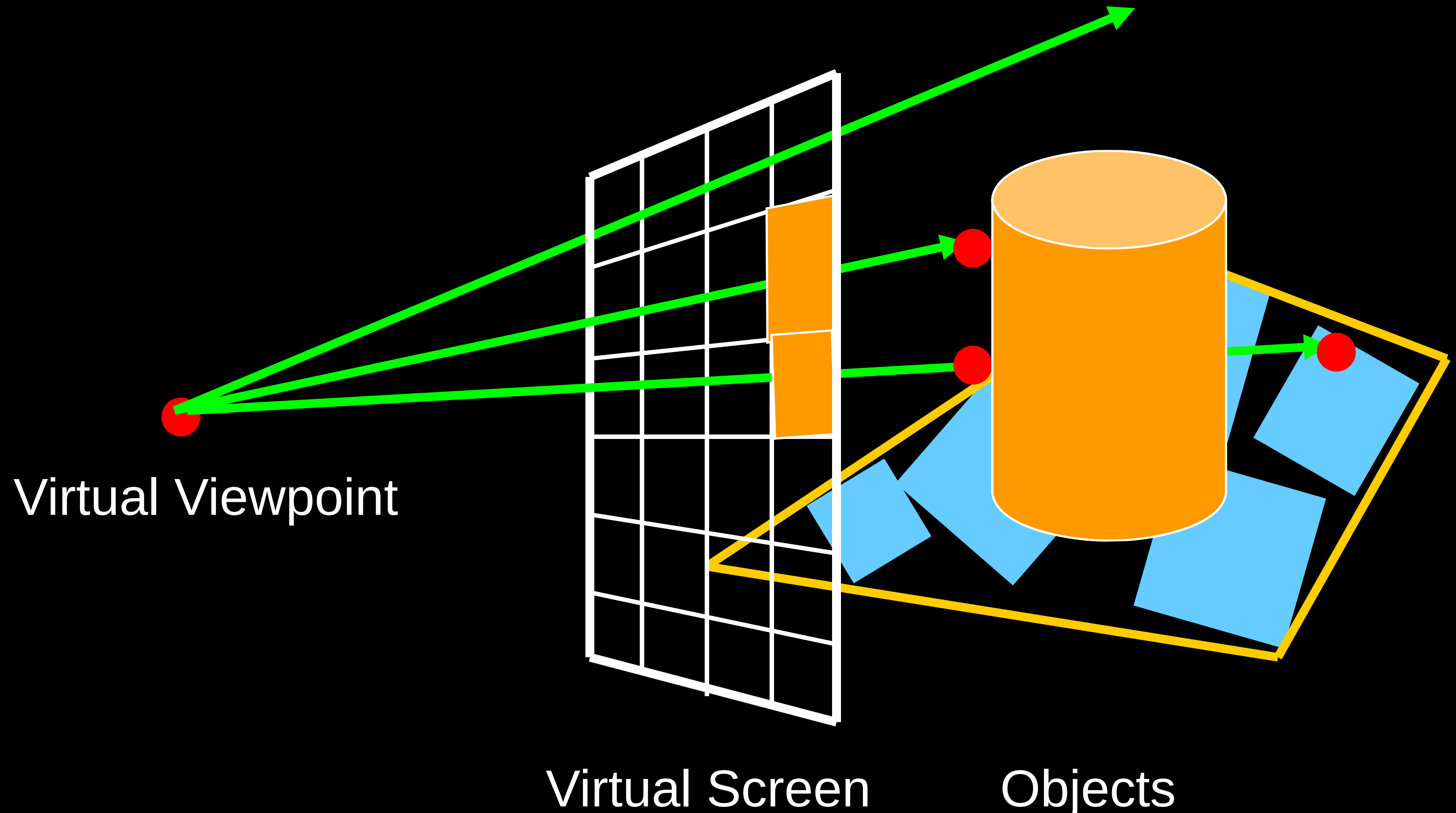
Image courtesy
Paul Heckbert 1983

- 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

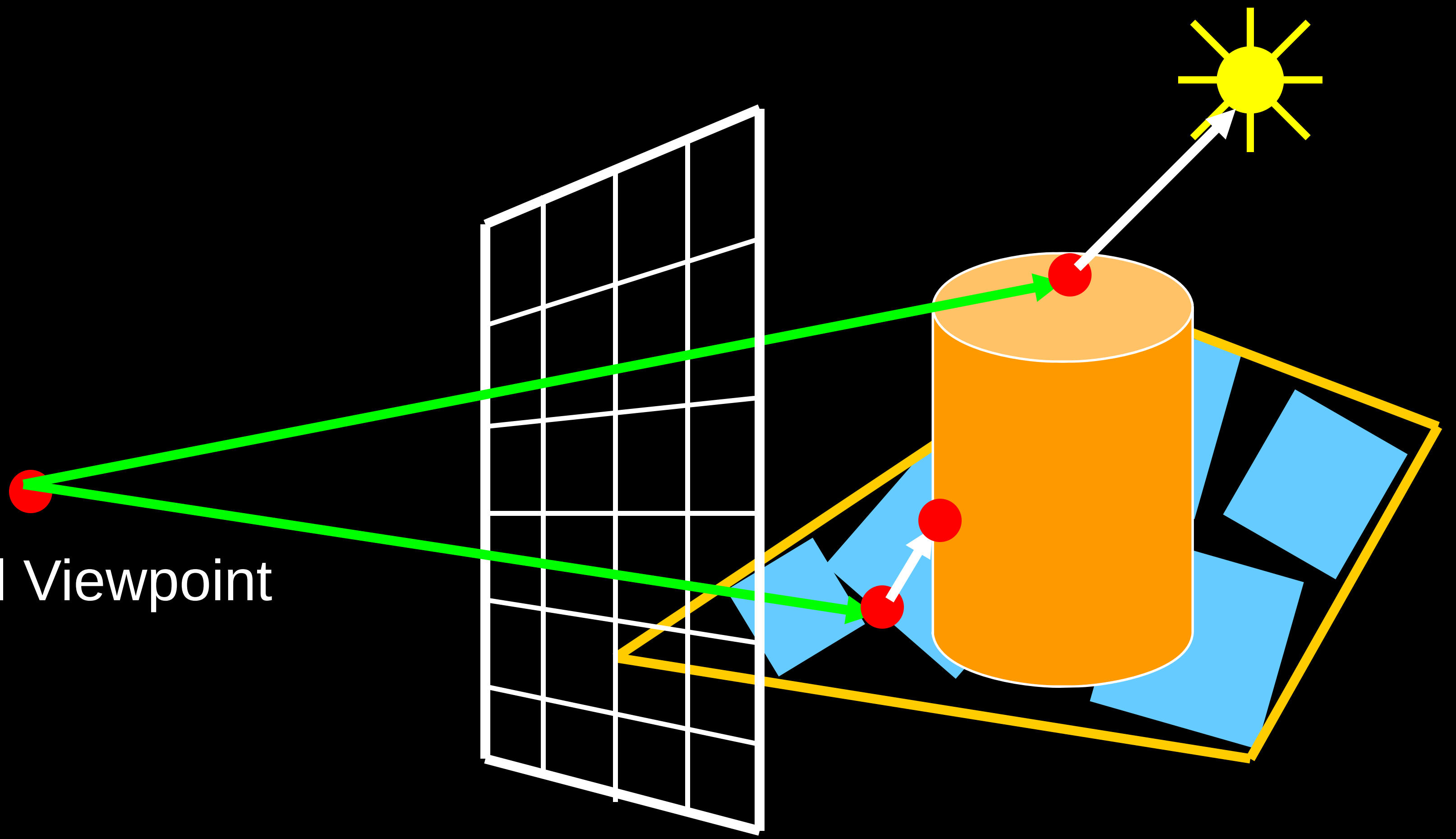
- Visibility per pixel instead of Z-buffer
- Find nearest object by shooting rays into scene
- Shade it as in standard OpenGL

Section 10.1-10.2 in text (we show visually, omitting math)

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: object in shadow

Shadow ray to light is unblocked: object visible

10.5 in textboo

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