CS888 Advanced Topics in Computer Graphics:

Physics-Based Animation

Jan. 7, 2014

Physics-Based Animation

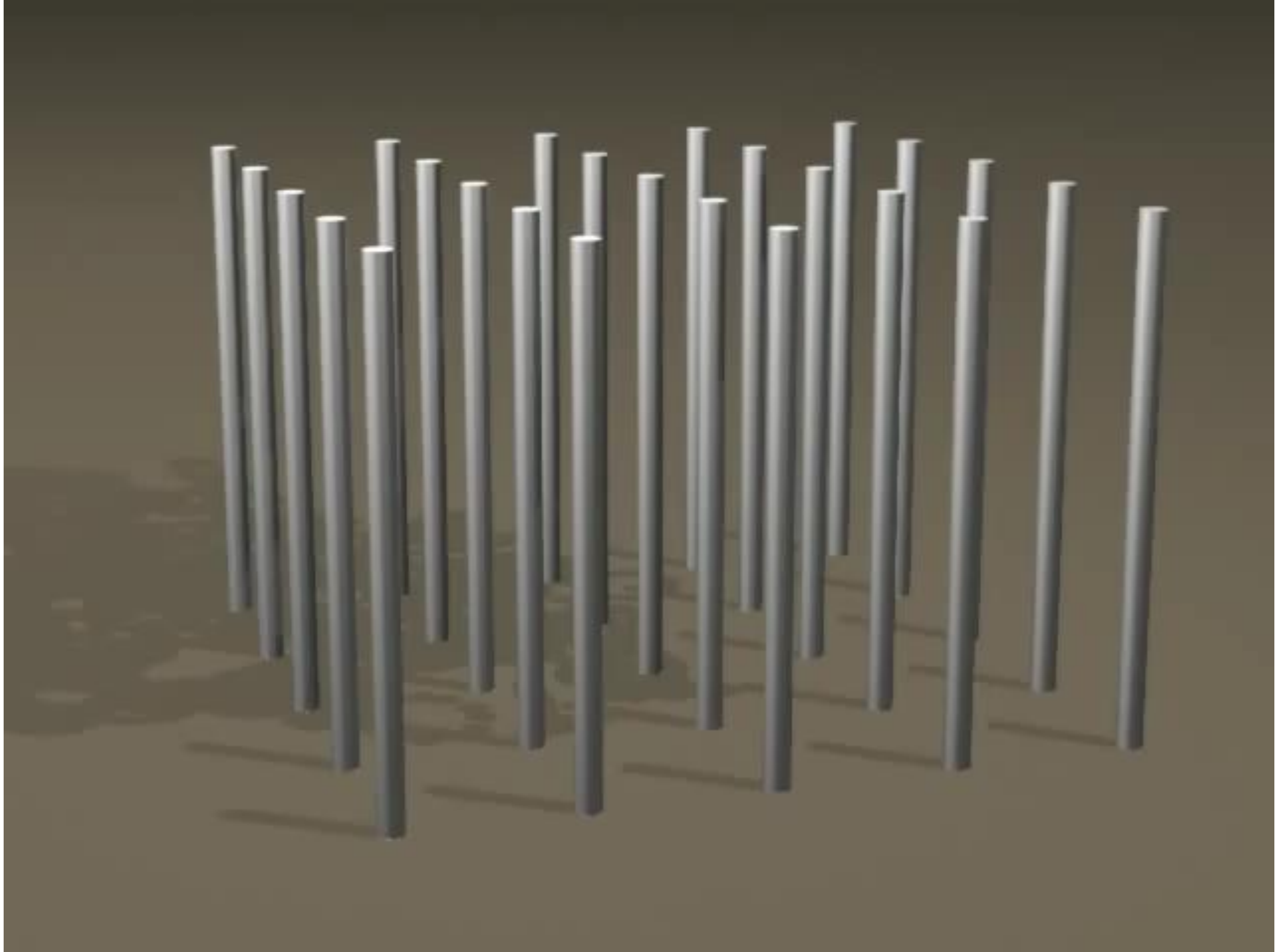
The use of physical simulation to generate animations of:

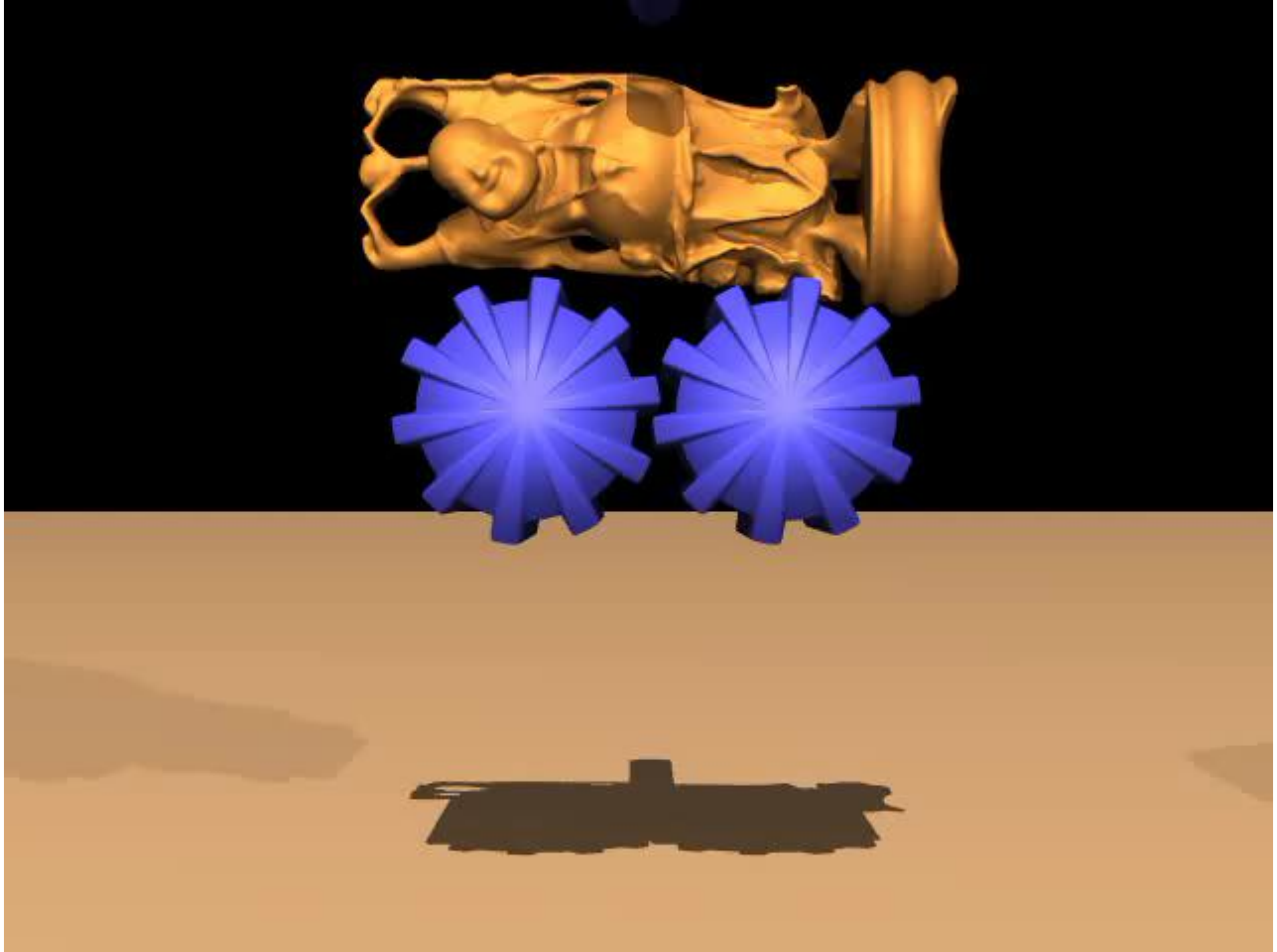
- Rigid bodies: “Perfectly” stiff or rigid objects
- Deformable objects: flesh, rubber, jello
- Shells/plates: Cloth, paper, sheet metal, plant leaves
- Rods/beams: Hair, strands, cords, slender tree branches
- Gases: Air, fire, smoke, explosions, bubbles
- Liquids: Water, oil, honey, slime, goop, oceans, waves

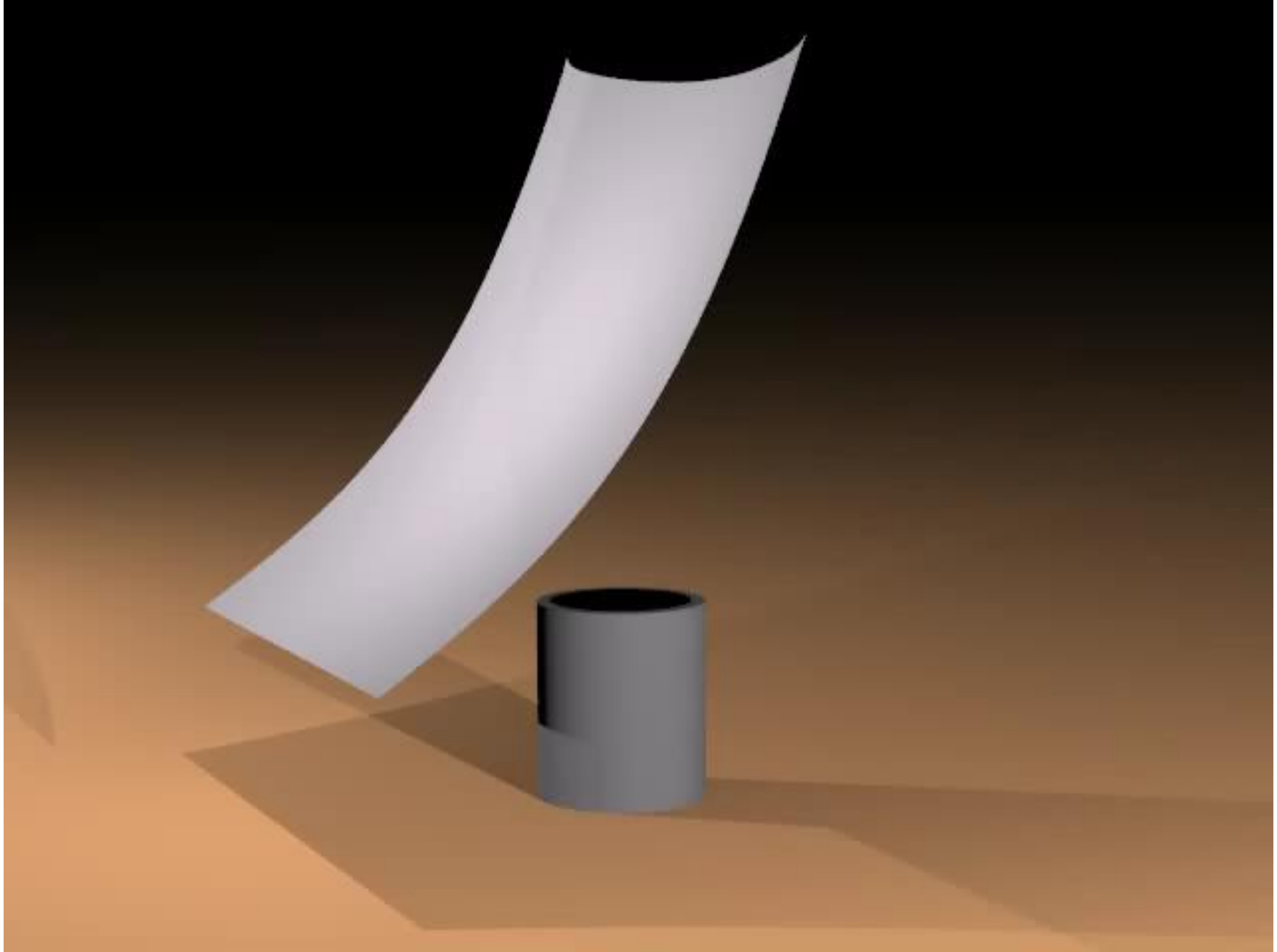
…and (m)any other visually interesting physical phenomena.

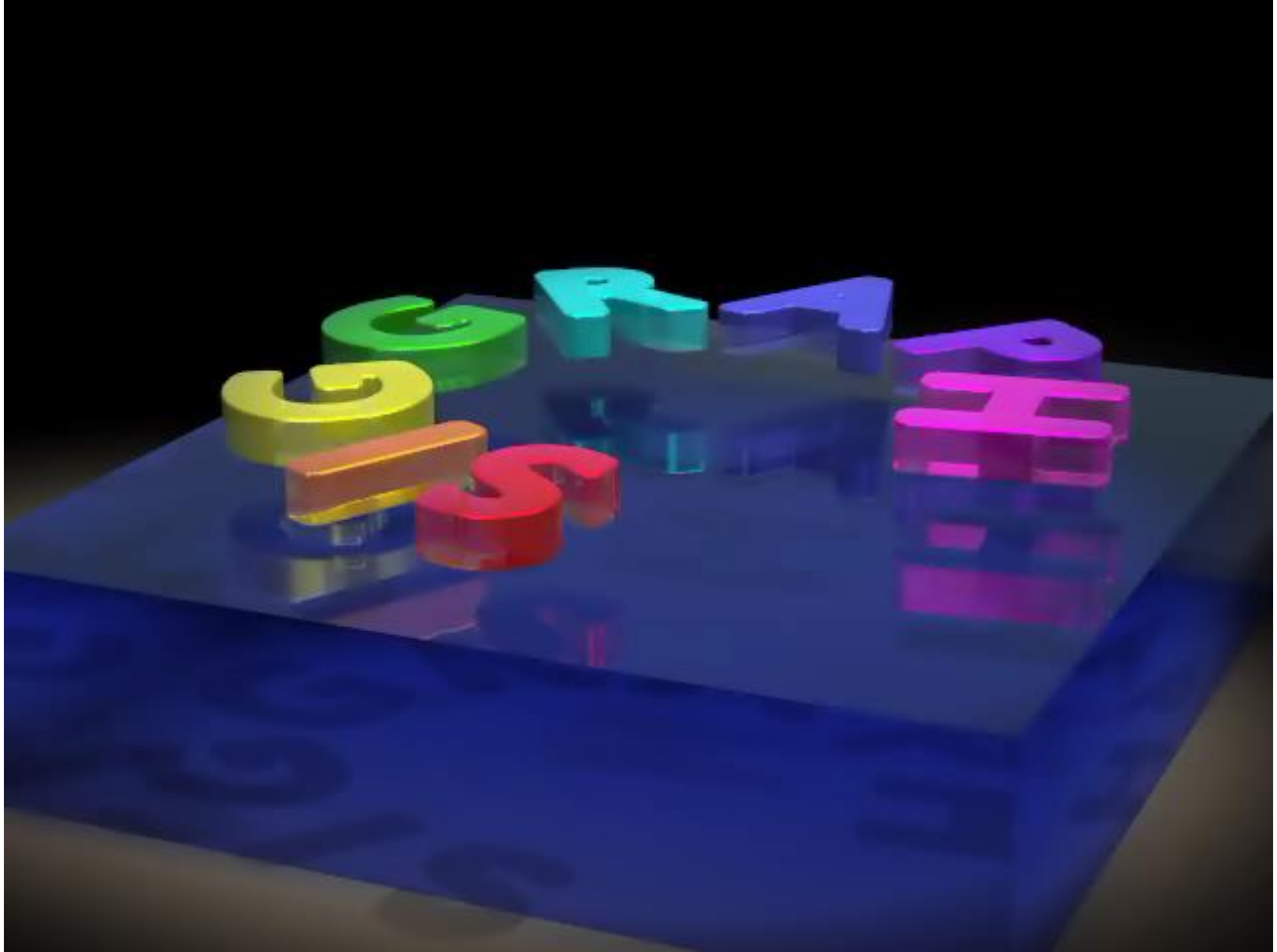
Why Use Simulated Physics?

- Too many *degrees of freedom* to model by hand.
- Humans are good at spotting physical irregularities ("weird" motion).
- Save artists time (avoid "simulating" in their heads!); can instead focus on characters, story, aesthetics, etc.
- Directly capturing real motion (via video camera or motion capture, etc.) can be limiting.
- Simulation is often cheaper, safer, and makes otherwise "impossible" animations feasible.
- For interactive applications, animations must respond on-the-fly in a flexible way.



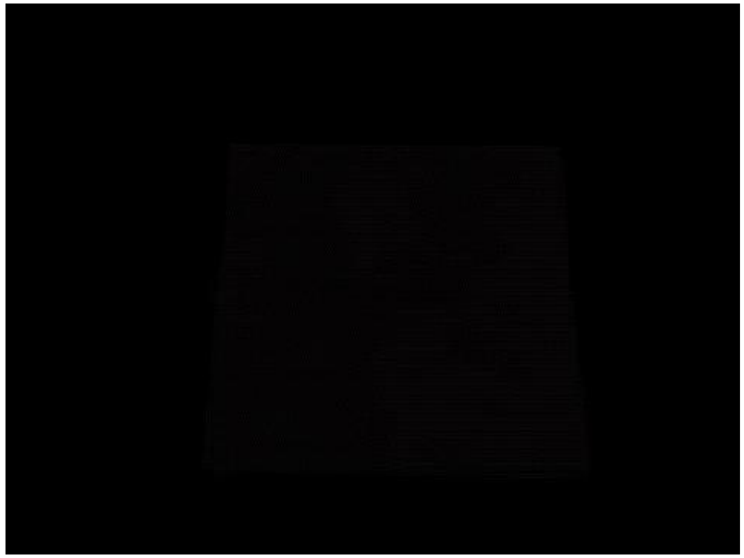





SIGGRAPH

Applications

- Computer gaming
- Visual effects & animated movies
- Various interactive tools
  - Virtual surgery (training)
  - Educational software
  - Virtual fitting room
  - Interactive/iterative design software
    - architecture, tailoring, etc.



Course Organization

Tues/Thurs at 10:00-11:20am in DC 3313.

Instructor: Christopher Batty (DC 3605)
E-mail: christopher.batty@uwaterloo.ca
Feel free to send questions/comments, put “CS888” in the subject.

Office hours by appointment.

Course web page:

http://www.cs.uwaterloo.ca/~c2batty/courses/CS888_2014/

Course Organization

- Primarily seminar-style – paper presentations and group discussion.
- A few lectures to set the stage.
- Course project – implement a physical simulation technique.
- One paper review per week.

- See preliminary schedule on the website. Roughly: first 2/3 on solids, last 1/3 on fluids.

Grade Breakdown

- Project: 40%
- Presentations: 25%
- Reviews: 20%
- Participation/discussion: 15%

Late penalty of 20% per day.

Attendance is expected to all classes. If a class *must* be missed for research (conference, deadline, etc.), notify me one week prior.

Background

- You should have some familiarity with computer graphics and numerical methods.
- I’ll cover some background in the first few lectures.
- A good general source is Baraff & Witkin’s SIGGRAPH course notes.
- Links to additional (optional) material are on the web site.

Presentations

Describe :

- Key novel elements of the paper, and their significance.
- Relationship to similar work.
- Strengths and weaknesses.
- Future directions.

Length: 20-25 minutes.

Probably 2-3 presentations each.

Steve Mann has some advice for giving talks:

http://www.cgl.uwaterloo.ca/~smann/Talks/CGL.98.11.24/

http://www.cgl.uwaterloo.ca/~smann/GSInfo/talk_guidelines.html

Presentations

See the list of topics (by week) and papers to choose from on the course website.

Email me your top 3 preferred slots for the first round of presentations. No guarantees. First slot is Jan 16.

Assuming 10 students, the first round topics are:

- Rigid bodies
- Deformables & FEM/FVM
- Cloth & shells

Course Project

- Pick a method or technique for a physical system, and implement it.
- Should be non-trivial, but need not be *novel*.
- Solo or with a partner.
- Can rely on existing code/libraries, but must be documented.

Course Project: Deliverables

- 1-2 page project proposal – due Feb. 14.
- Short presentation & demo during the last week of class (Apr. 1 or 3).
- Final submission – tentatively due April 4. (Won’t be earlier.)
  - Final report (PDF) in SIGGRAPH paper format.
  - An animation clip illustrating the project results.
  - Code.

Paper Reviews

- Pick one of the papers to be presented each week.
- Write a SIGGRAPH-style review of the paper (I'll post a form to follow.)
- Due Sunday at 10pm prior to the week of the presentation.
- Starting with the papers from the 3rd week.

Questions for you…

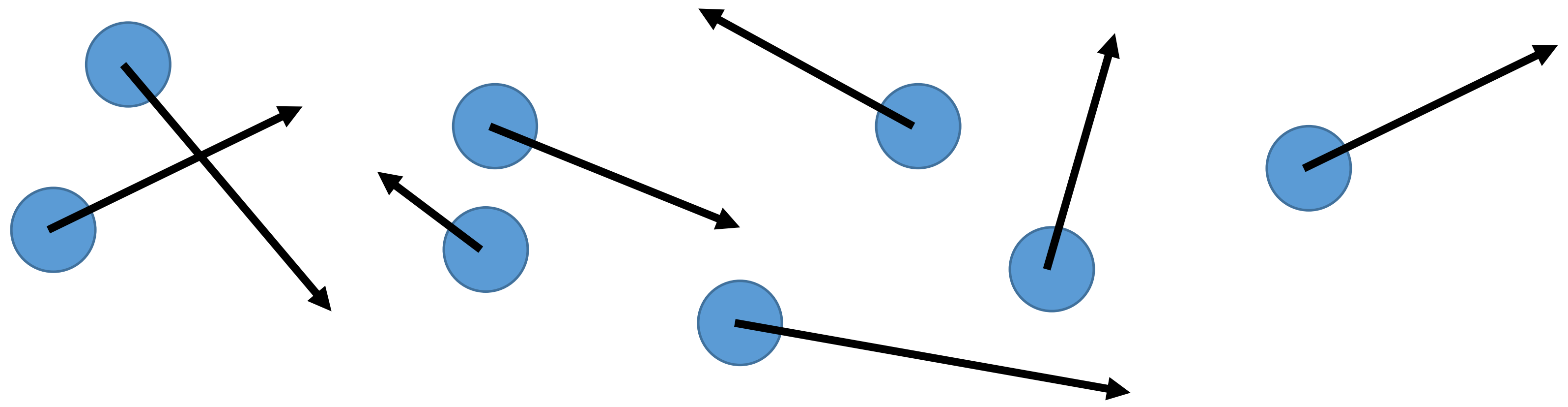
1. What level of computer graphics courses have you taken (at Waterloo or elsewhere)?
2. What level of scientific computing / numerical methods courses have you taken?
3. Summarize any other relevant background or experience.
4. What topic(s) are you most interested in?
5. What do you anticipate being most challenging about the course?

Introductions

Particle Systems

Particle Systems

Particle system: A collection of particles that obey rules dictating their creation, movement, deletion, and other attributes and behaviors.

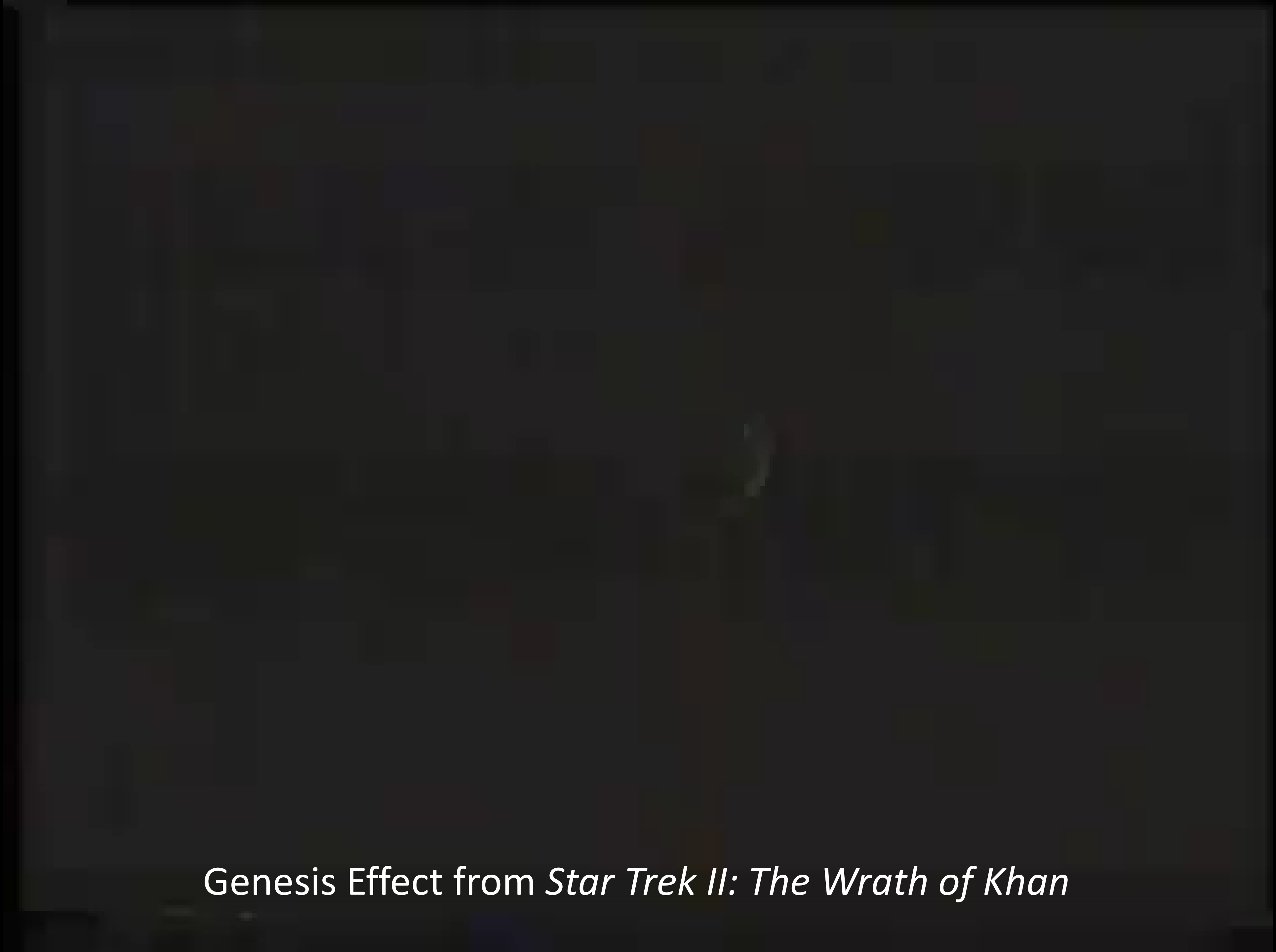


Particle Systems

- Often used to model “fuzzy”/complex phenomena, with…
  - ill-defined or changing boundaries
  - chaotic motion
  - e.g., fire, waterfalls, dust, clouds, flocking animals, etc.

- Common in 3D software (Maya, 3DSMax, Houdini, etc.)

- First used in 1982’s “Star Trek II: The Wrath of Khan” to model a fiery explosion transforming a planet.
- Another classic animation is Karl Sims’ “Particle Dreams” (1988).

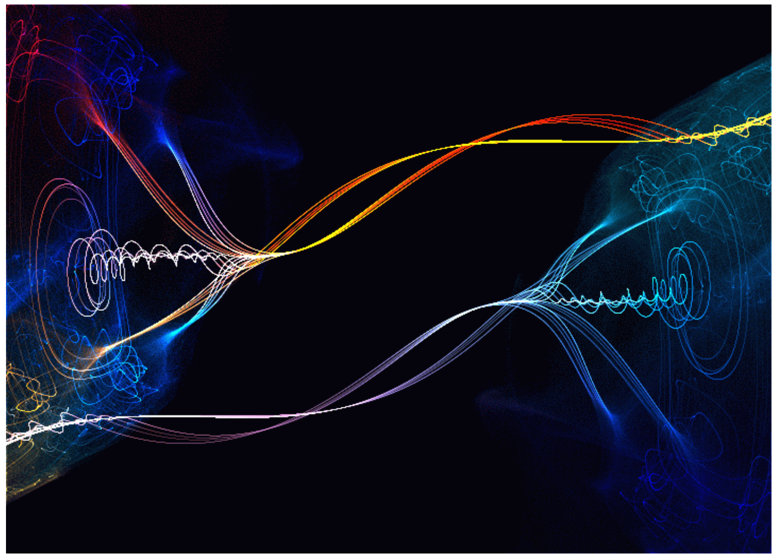
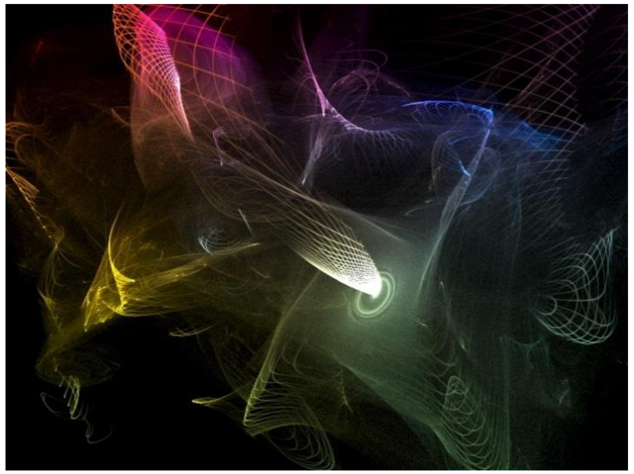
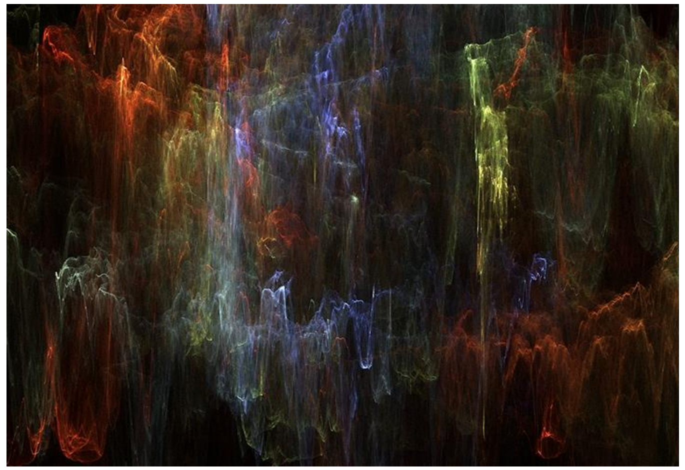
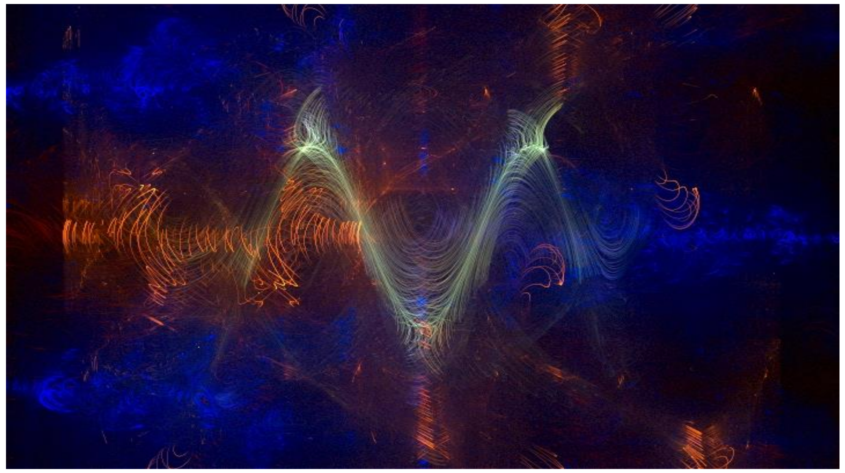

Genesis Effect from *Star Trek II: The Wrath of Khan*

PARTICLE DREAMS

Karl Sims

Optomystic

More Recent Example – “Spore”

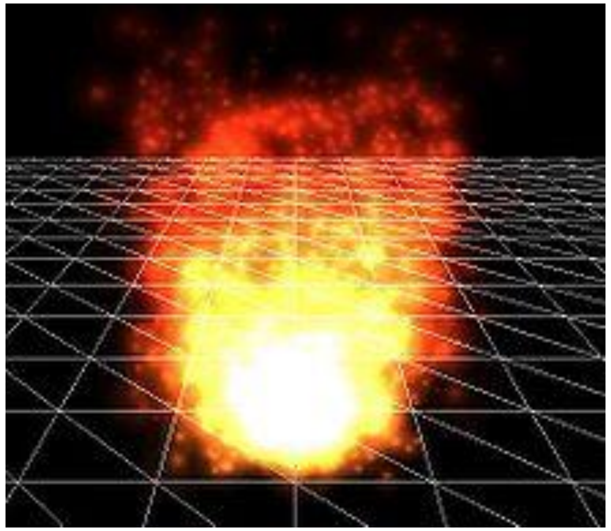





http://www.imagesavant.com/

Particle Systems

Possible particle data/attributes:

- Position (x,y,z)
- **Velocity**
- Color
- State
- Age
- Temperature
- etc. (whatever else you like!)



Particle Systems

At each frame:

- Create new particles and assign initial attributes.
- Update existing particle position/velocity/attributes according to chosen rules.
- Delete “expired” (old) particles.

Rules can also incorporate some randomness.

Designing the rules is where the art (and maybe science) comes in.

Example: Star Trek II: Genesis Effect

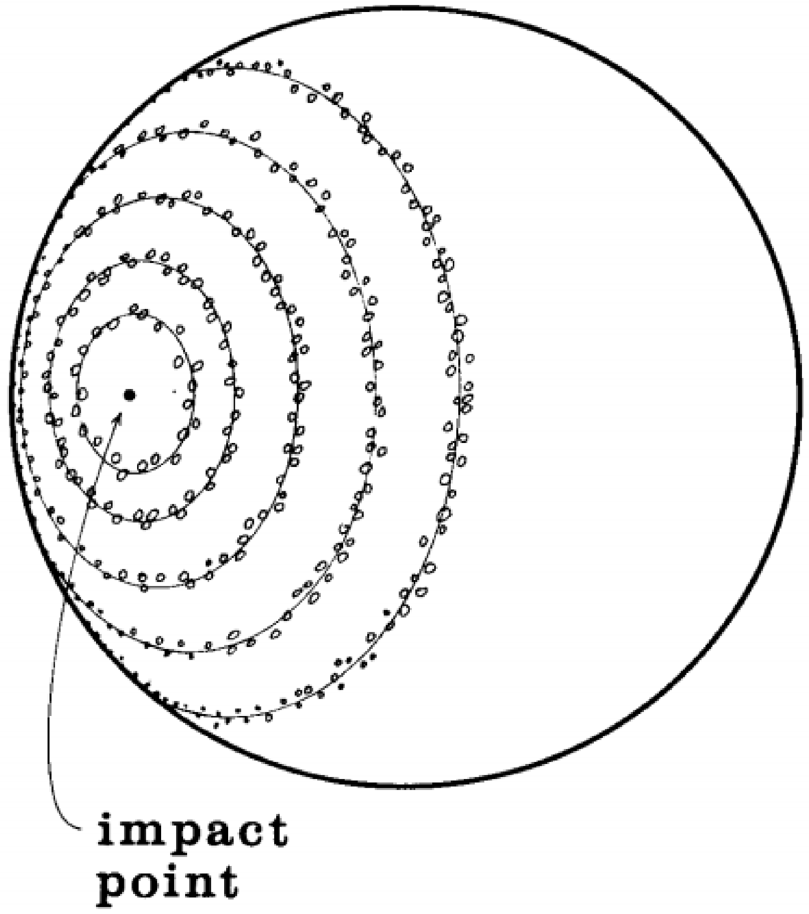


Fig. 2. Distribution of particle systems on the planet's surface.

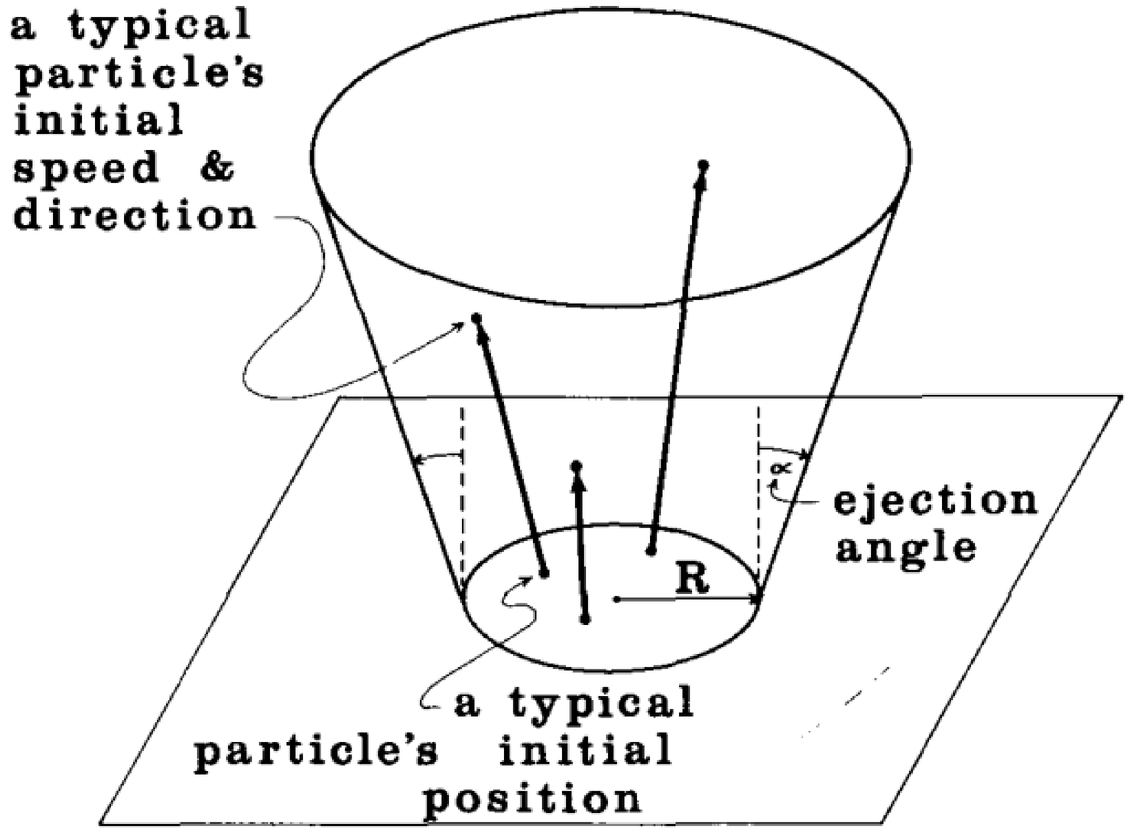


Fig. 3. Form of an explosion-like particle system.

From [Reeves 1983]

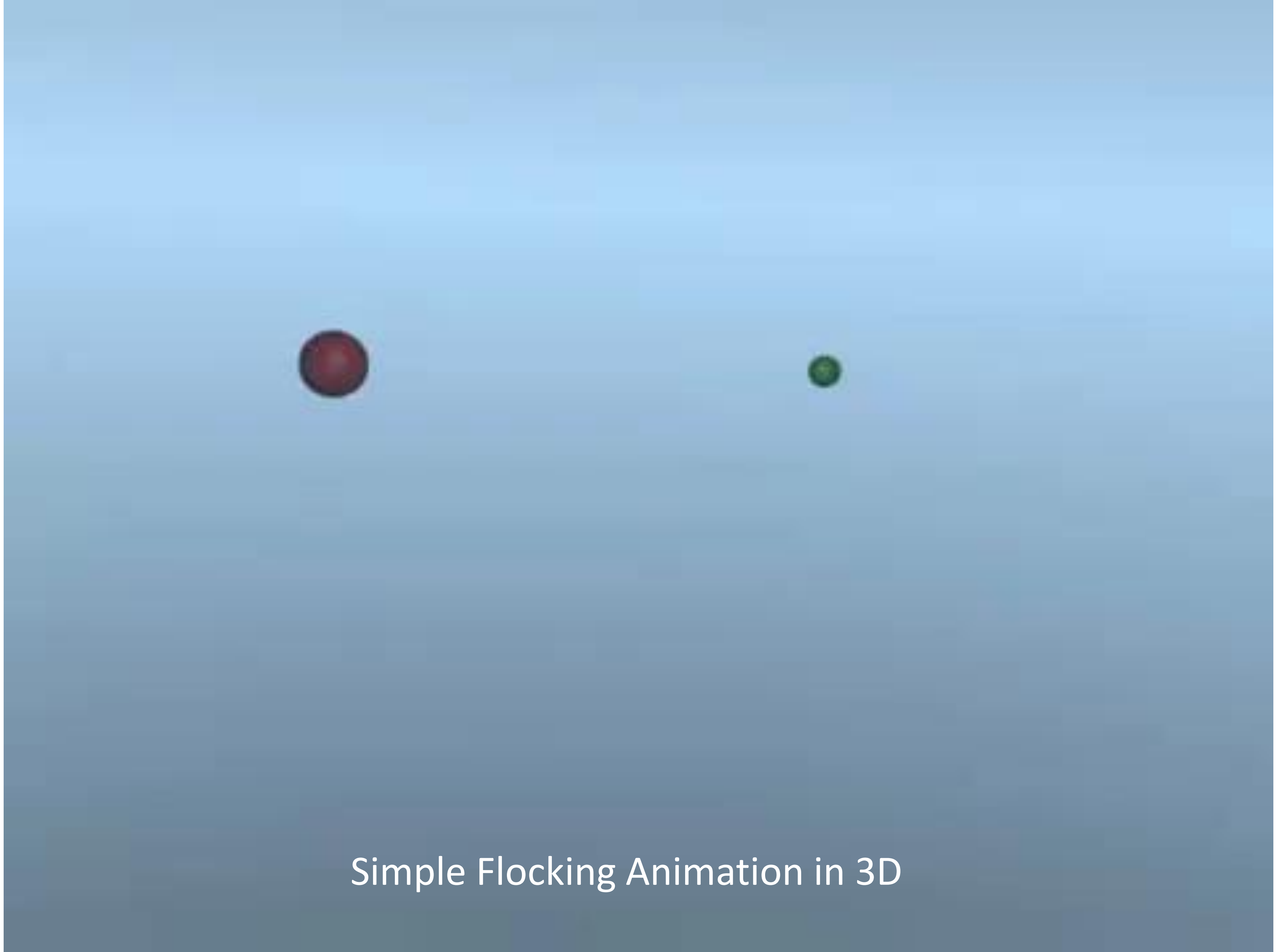
Flocking (“Boids”)

Simple rules relating to interactions between nearby particles can yield emergent, flocking-like behaviour.

- Collision Avoidance (separation)
- Velocity Matching (alignment)
- Flock Centering (cohesion)

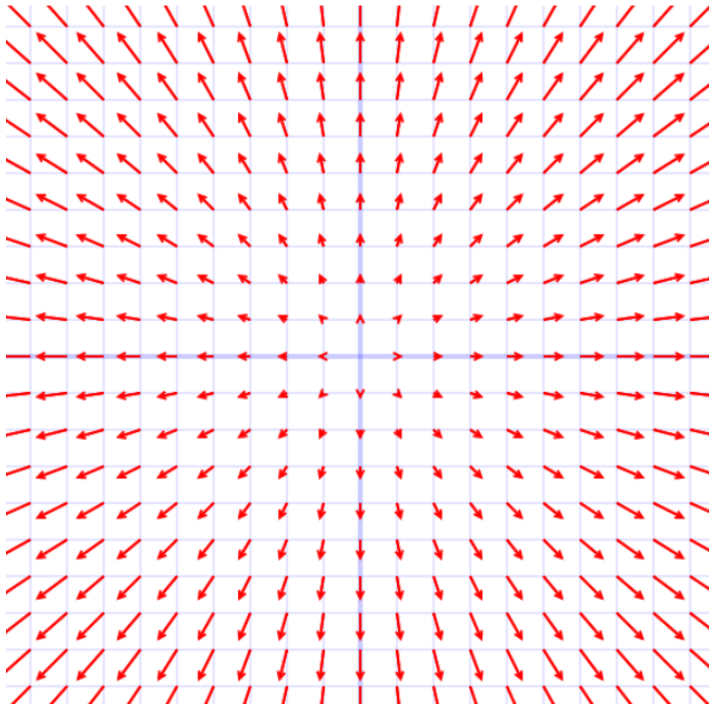
For details see:

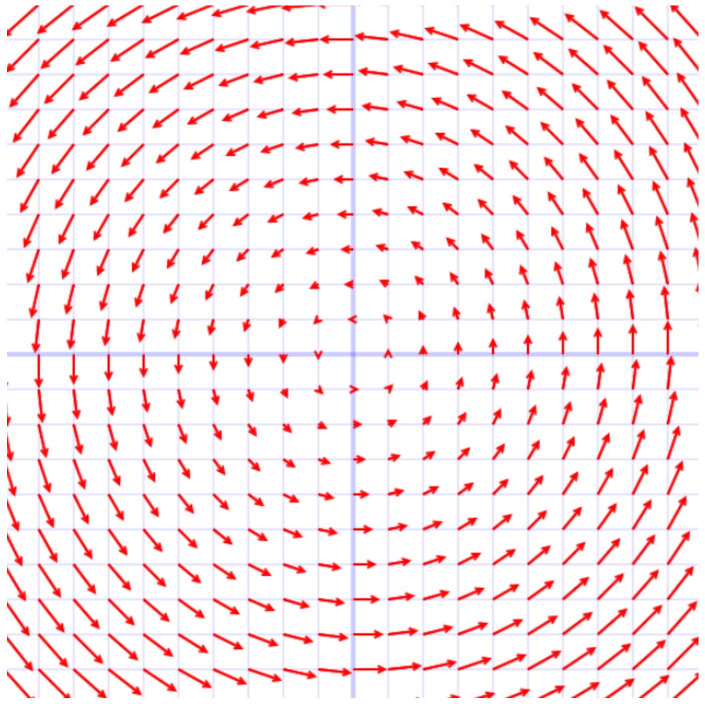
“Flocks, herds and schools: A distributed behavioral model.” [Reynolds, 1987]

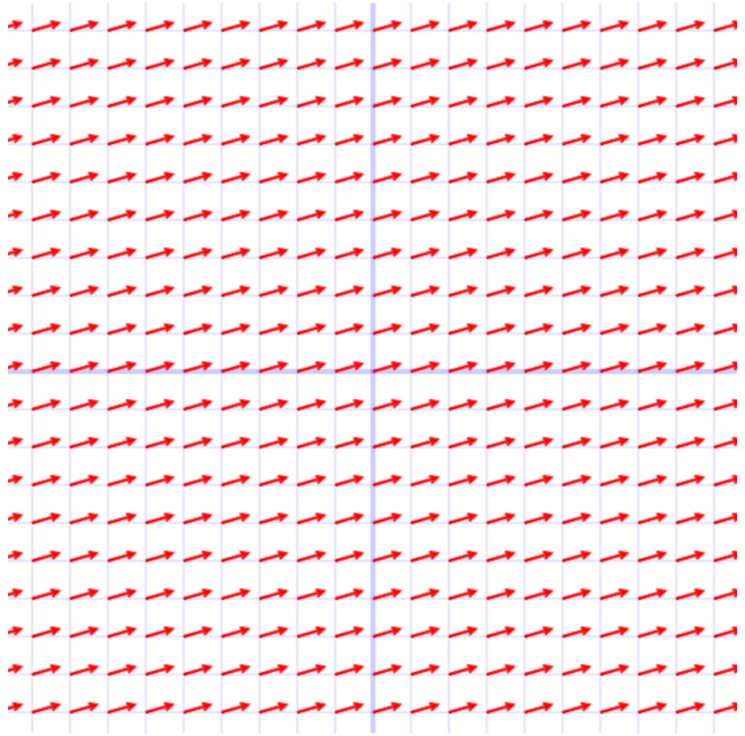

Simple Flocking Animation in 3D

Vector Fields

Often particle motion is dictated by a function that takes a 3D position and returns a 3D velocity vector, i.e., a vector field.


Radial expansion: $V = (x, y)$


Rotational Vortex: $V = (-y, x)$


Constant Wind: $V = (7,2)$

Basic Time Integration

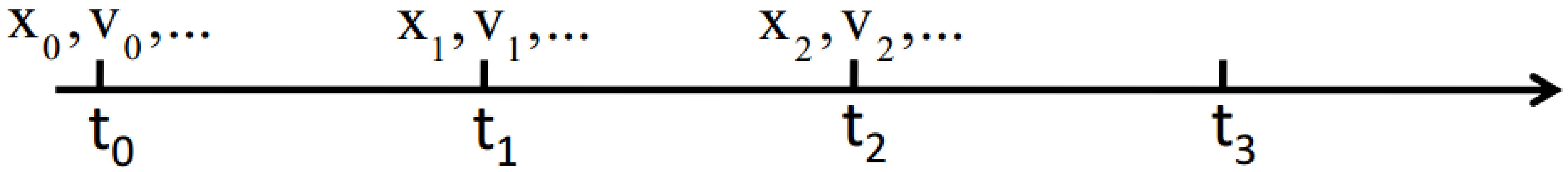
Solving For Particle Motion

Given a particle P at time T=t with:

- Current position X=(x,y)
- Velocity function V(X,t) = (u,v)

…how do we determine the new particle position at time T = t+Δt?

This task is called *time integration*. Δt is the time step.



Time Integration (for 1st order dynamics)

Recall: velocity V is the time derivative of position X.

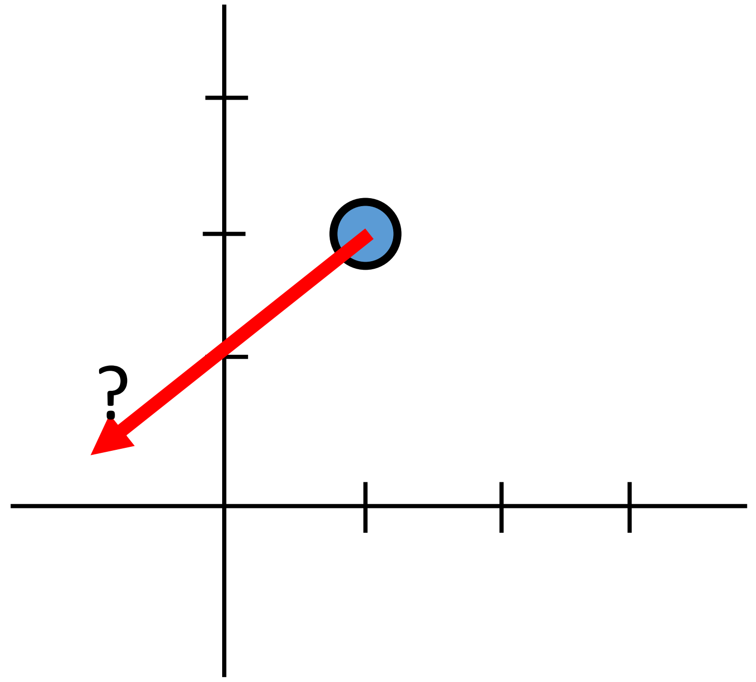
i.e., rate of change of the particle position with respect to time.

$$\frac{dX}{dt} = V$$

This is a *differential equation* relating X and V by a (time) derivative.
Given initial values for X, solve for X at subsequent times.

Time Integration

e.g., consider a particle with current position X = (1,2) and (constant) velocity V = (-1,-1) m/s, taking a *time step* of length Δt = 0.5 s.



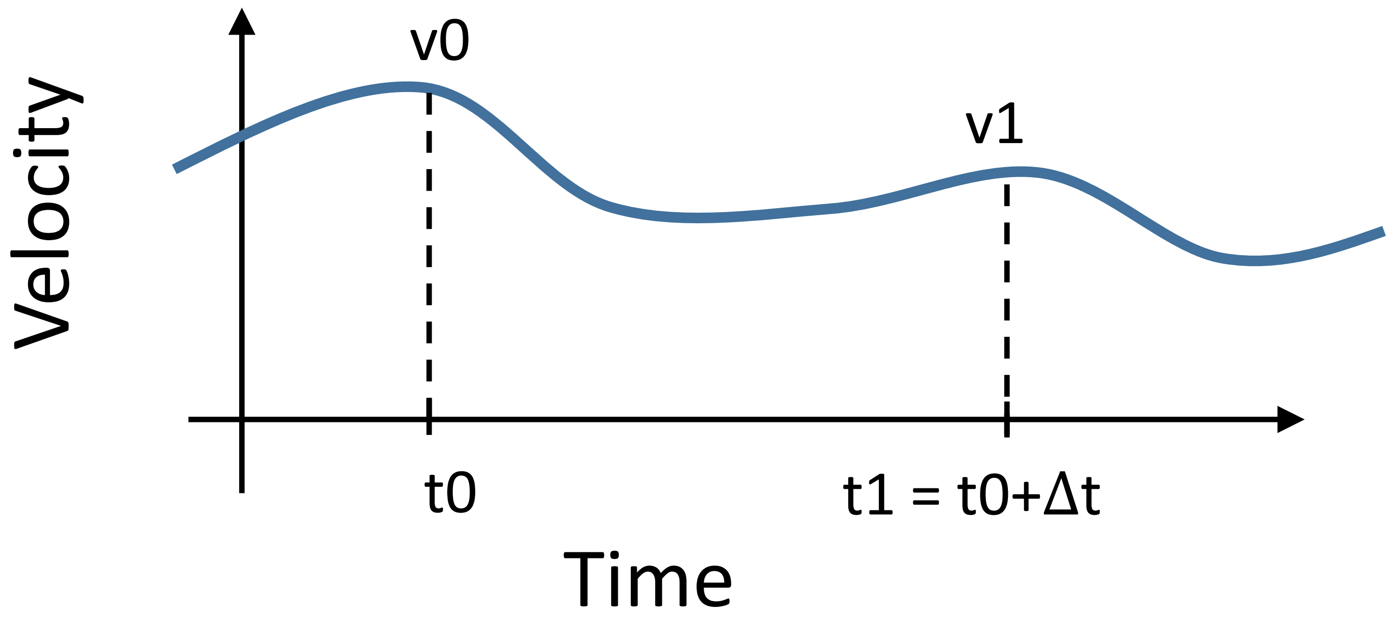
$$X^{t+\Delta t} = X^t + V\Delta t$$

Solution: $X^{t+\Delta t}$ = (0.5, 1.5)

Time Integration

Finding the new position requires *integrating velocity over time*.

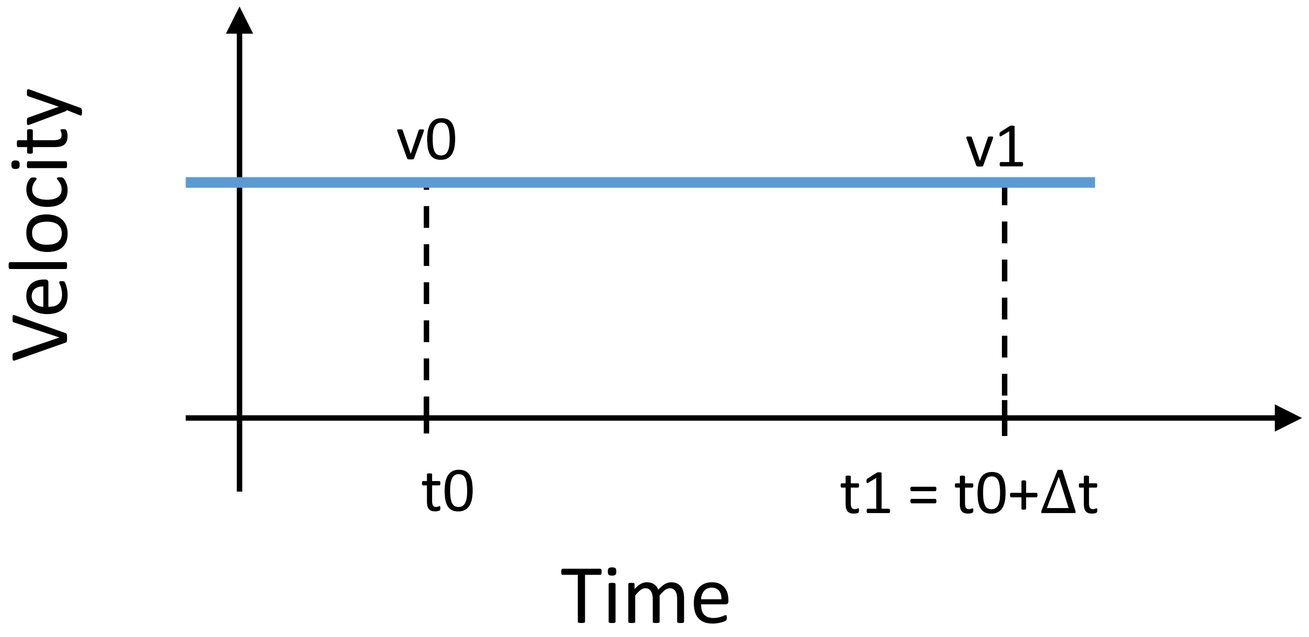
$$X^{t+\Delta t} = X^t + \int_t^{t+\Delta t} V dT$$



i.e., find the area under the curve.

Time Integration

In our example, velocity was a constant, so the (rectangular) area was *exactly* $V\Delta t$.



$$X^{t+\Delta t} = X^{t} + V\Delta t$$

What About *Time-Varying* Velocity?

Velocity could depend on many factors, including current time, position, state…

$$\frac{dX}{dt} = V(t, X(t), \dots)$$

e.g., $V = (17t\log(t)\tan(y), \operatorname{arcsinh}(t)^t x^2)$.

In general, we can't solve the integral exactly / analytically. We must approximate.

Numerical Integration

We will use *numerical integration*.

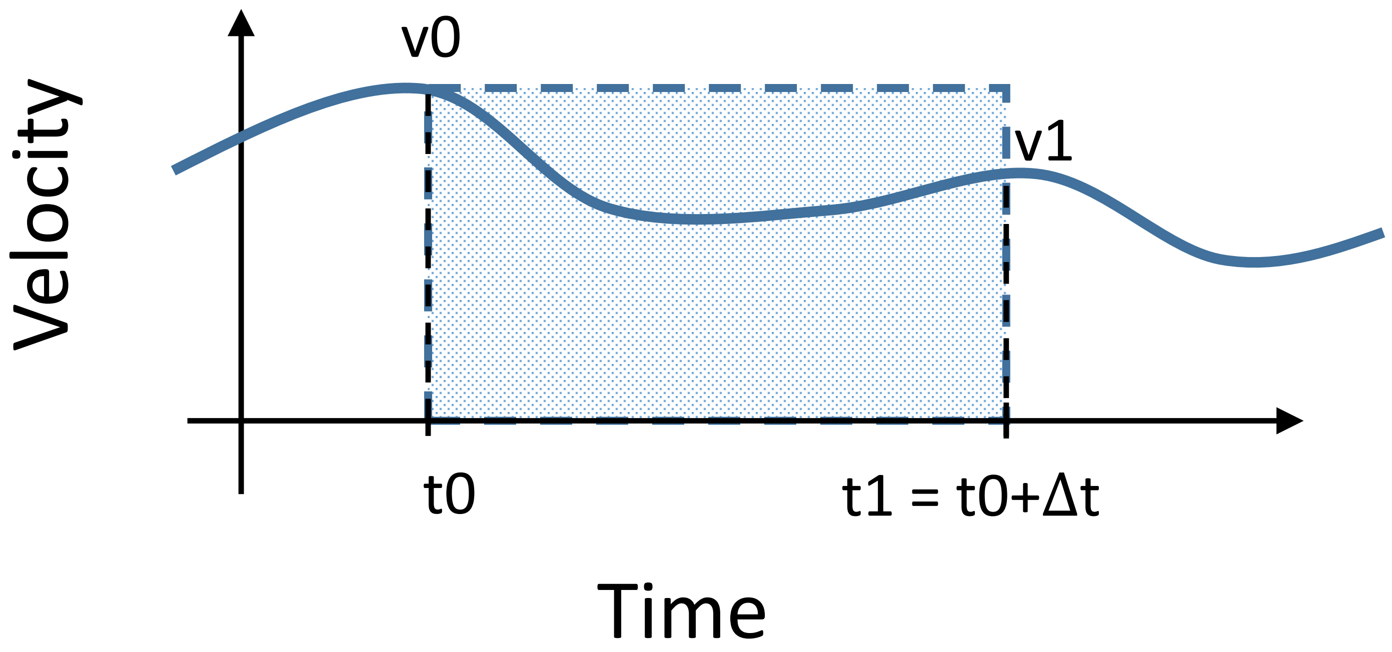
First idea: “Ignore” that the velocity may change during the time step. Then…

$$X^{t+\Delta t} = X^t + V(t)\Delta t$$

i.e., Evaluate V at the **current time** t, and use it to take only a single step. Repeat on the next step.

Numerical Integration

Effectively, we are approximating the true area as a rectangle, using the starting velocity, v0.



Forward Euler

This simple scheme is called Forward Euler.

$$X^{t+\Delta t} = X^{t} + V(t, X(t))\Delta t$$

Example:

- X(t=0) = (0, 1)
- V = (-y, x),
- Δt=0.5

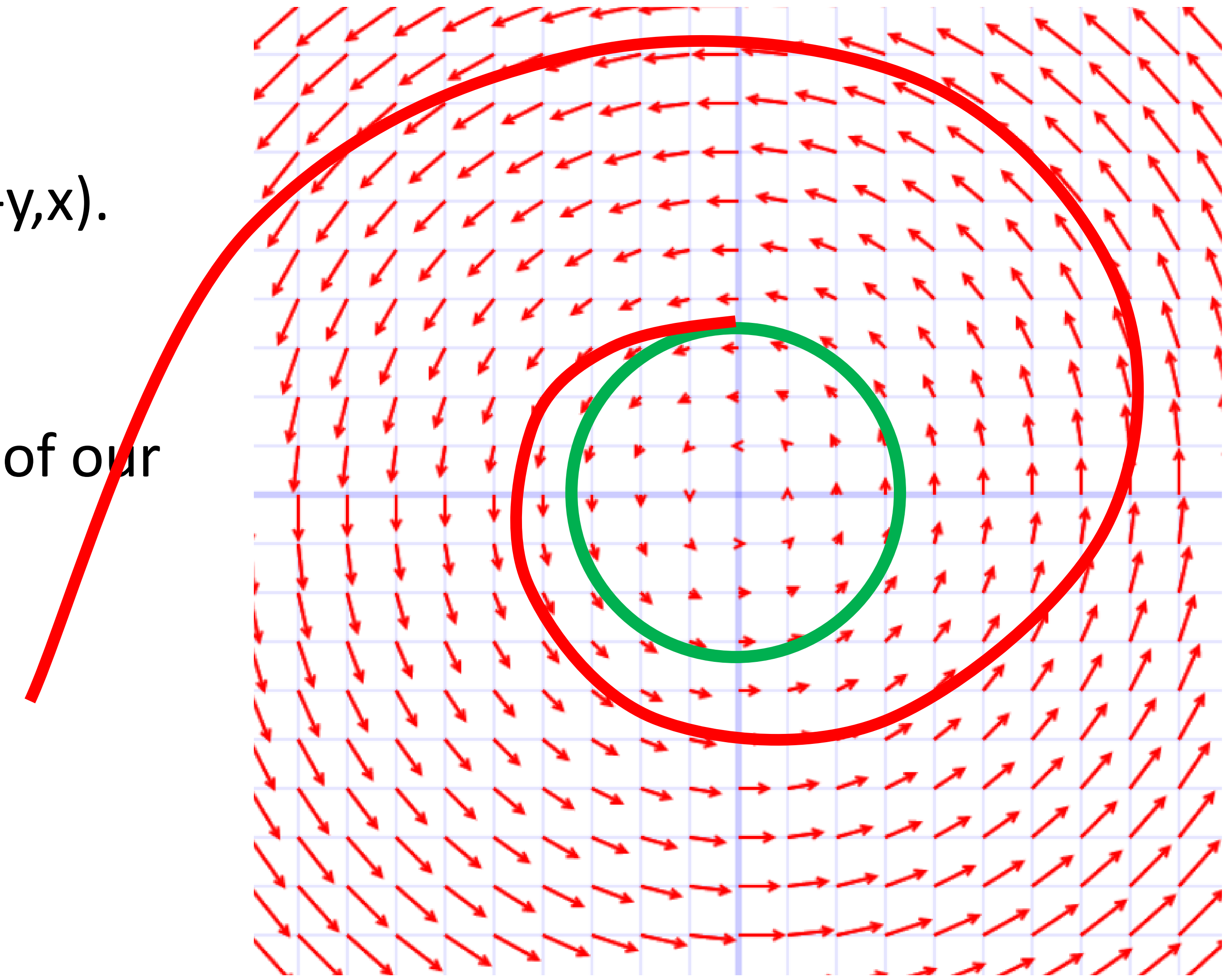
Find X(t=1.5).

X(0.5) = (0,1) + 0.5(-1,0) = (-0.5, 1)
X(1) = (-0.5, 1) + 0.5 (-1, -0.5) = (-1, 0.75)
X(1.5) = (-1,0.75) + 0.5(-0.75,-1) = (-1.375, 0.25)
…

Vector Fields

This is the vector field V = (-y,x).

Compare the expected true trajectory to the behaviour of our numerical solution…

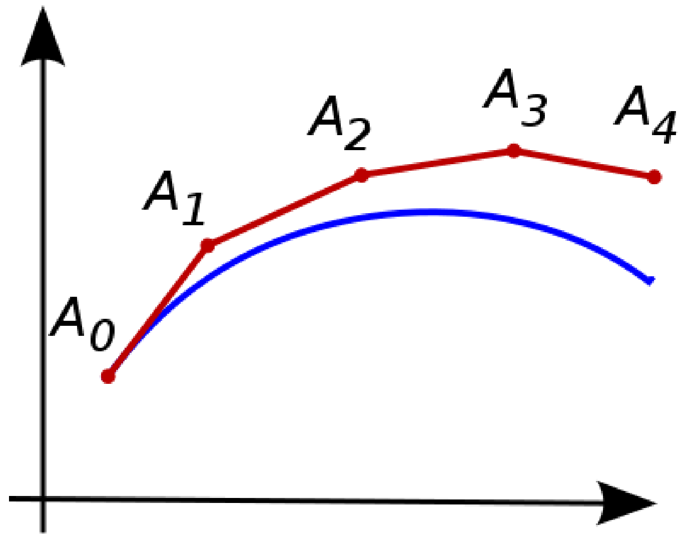


Forward Euler – Points to Note

1. Accumulated error can cause the numerical solution to drift away from the true solution.

2. But, the smaller the time step Δt, the more accurate the approximate trajectory becomes.

3. If the time step is too large, the result can "blow up" and yield garbage answers. Forward Euler has a (problem-dependent) maximum *stable* time step…



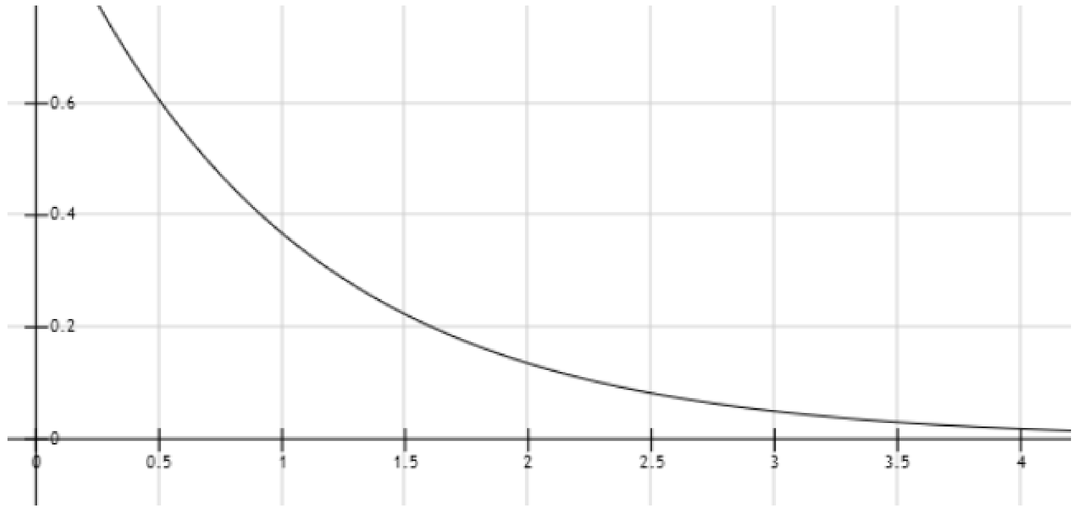
Forward Euler – Instability

Consider the 1D function: $\frac{dx}{dt} = -x$, with x(t=0) = 1.

True solution is: e^{-t}

Always positive, decays smoothly.

Numerical solution for $\Delta t = 3$?



Result: 1, -2, 4, -8, 16, etc.

Wrong!

The sign flips madly, the magnitude increases instead of decreasing.

Other Time Integration Schemes

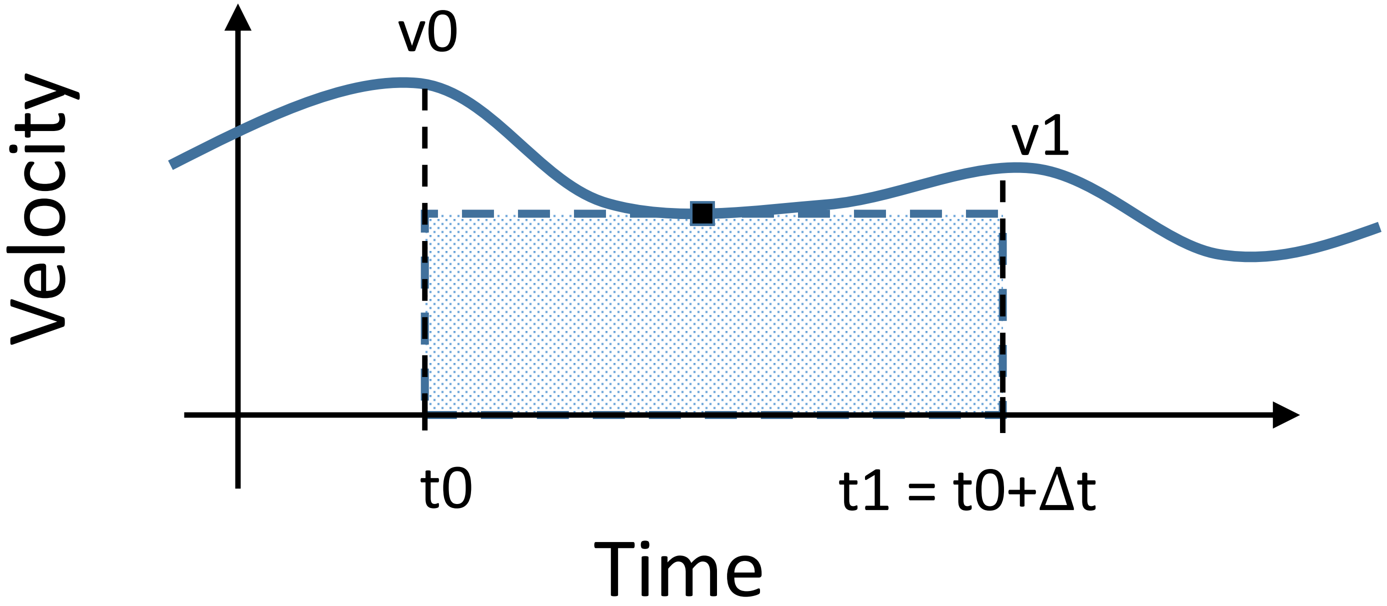
Forward Euler uses the velocity at the *start* of a time step to perform the integration.

Other common schemes use the (possibly approximate) velocity at the *middle*, *end*, and/or *other* instants within a time step to increase accuracy and stability.

e.g. midpoint method, trapezoidal rule, implicit Euler, Runge Kutta schemes, etc.

(Explicit) Midpoint method

Use the velocity at the *midpoint* to estimate the integral. (Also known as 2nd order Runge Kutta or RK2).



(Explicit) Midpoint method

Estimate the midpoint position halfway through a time step:

$$X^{mid} = X^t + \frac{V(t, X^t)\Delta t}{2}$$

Then, use the velocity evaluated at the midpoint to determine the final position.

$$X^{t+\Delta t} = X^t + V\left(t + \tfrac{1}{2}\Delta t, X^{mid}\right)\Delta t$$

E.g., Try the midpoint method on the velocity V = (-y,x).

Sketch this out

Adding Some Physics

Newton’s 2nd Law

Rather than prescribe velocities, we often want to use physics (classical mechanics) to solve for *both* X *and* V, given a set of applied forces, F.

First, assign each particle *i* some fixed mass, M_i.

Then, recall Newton’s 2nd law: Force = Mass x Acceleration.

Forces

What physical forces might we use to drive a particle system?

- Gravity
- Wind
- Springs / Elasticity
- Damping / Viscosity
- Friction
- Collisions/Contact
- …

Given the set of forces $F_1, F_2, \dots, F_n$, sum to get net force on a particle.

2nd order dynamics

Earlier, we had a given velocity, V, dictating how we update position X.

$$\frac{dX}{dt} = V$$

Now, we instead have given forces, F, and Newton's 2nd law, F=ma.

Recalling that acceleration is the 2nd time derivative of position, we have a new (2nd order) differential equation…

$$m\frac{d^2X}{dt^2} = F$$

2nd order dynamics

We can split this into two 1st order equations…

$$m\frac{d^2X}{dt^2} = F \quad \Longrightarrow \quad m\frac{dV}{dt} = F \qquad \frac{dX}{dt} = V$$

Then time integrate each of these (using e.g. forward Euler).

Forward Euler, revisited

Position Update:

$$\frac{dX}{dt} = V \quad \Longrightarrow \quad X^{t+\Delta t} = X^t + V^t \Delta t$$

Velocity Update:

$$m\frac{dV}{dt} = F \quad \Longrightarrow \quad V^{t+\Delta t} = V^t + \frac{F(t, X^t)}{m}\Delta t$$

Forces: Gravity

1. Earth-specific gravity (treated as a constant):

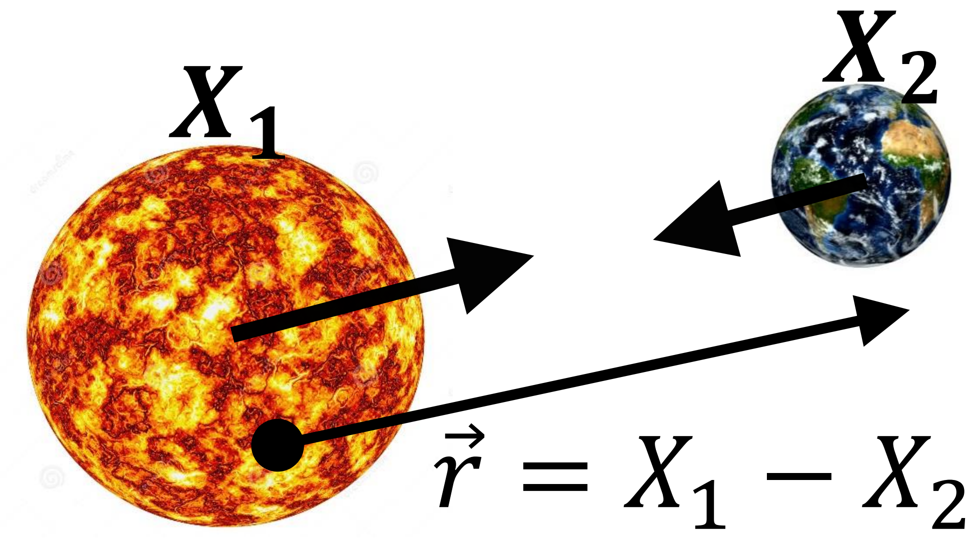
$$F = (0, -9.81, 0)\, m/s^2$$

Consider a 1D example: An initially stationary apple at a height of 10m falling under gravity, with time steps of length 0.1 seconds.

Forces: Gravity

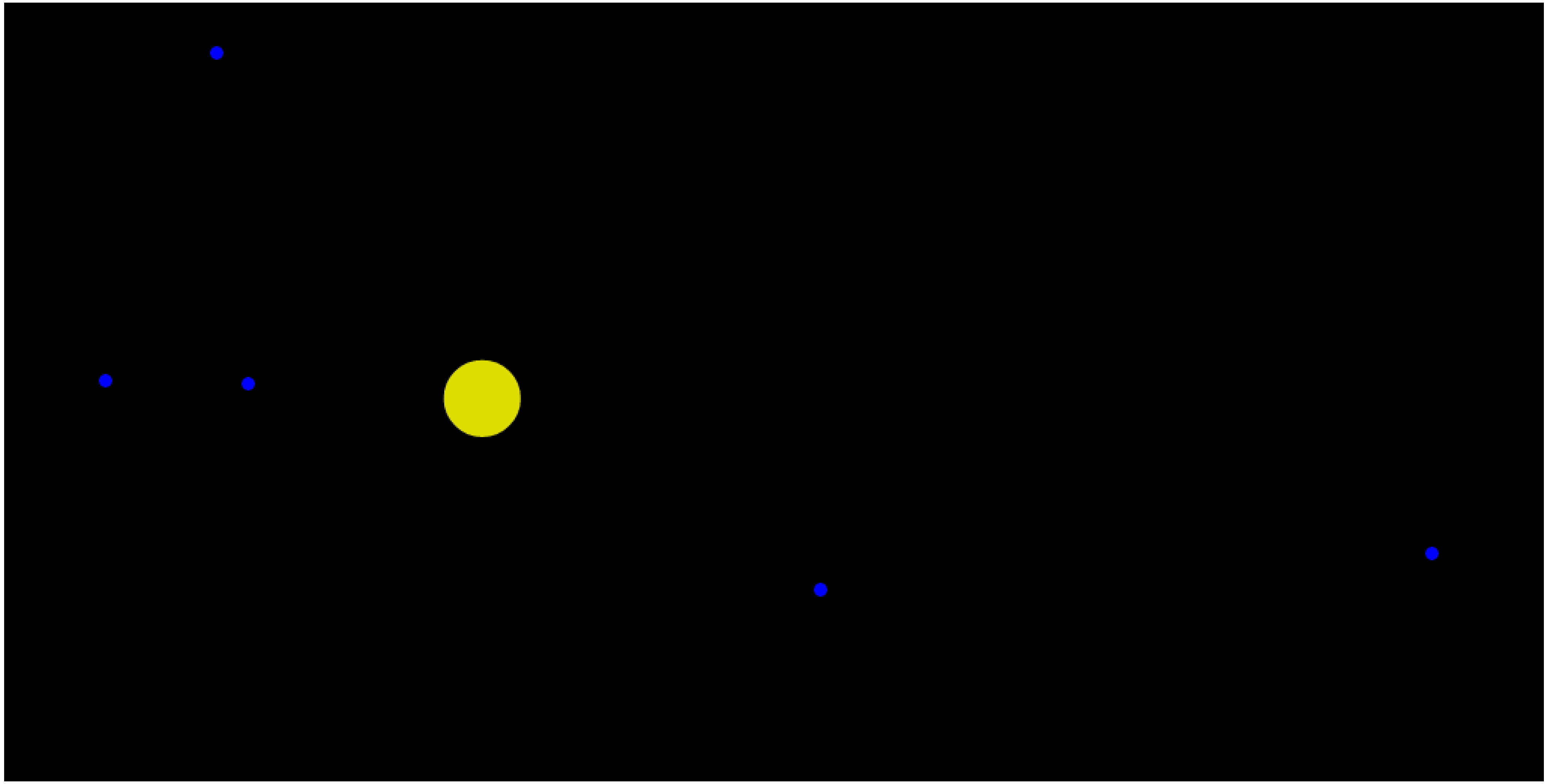
2. N-body gravitation:

$$F = -\frac{G m_1 m_2}{\|\vec{r}\|^3}\vec{r}$$



Gravitation Simulation

http://wxs.ca/js/jsgravity/



Forces: Springs!

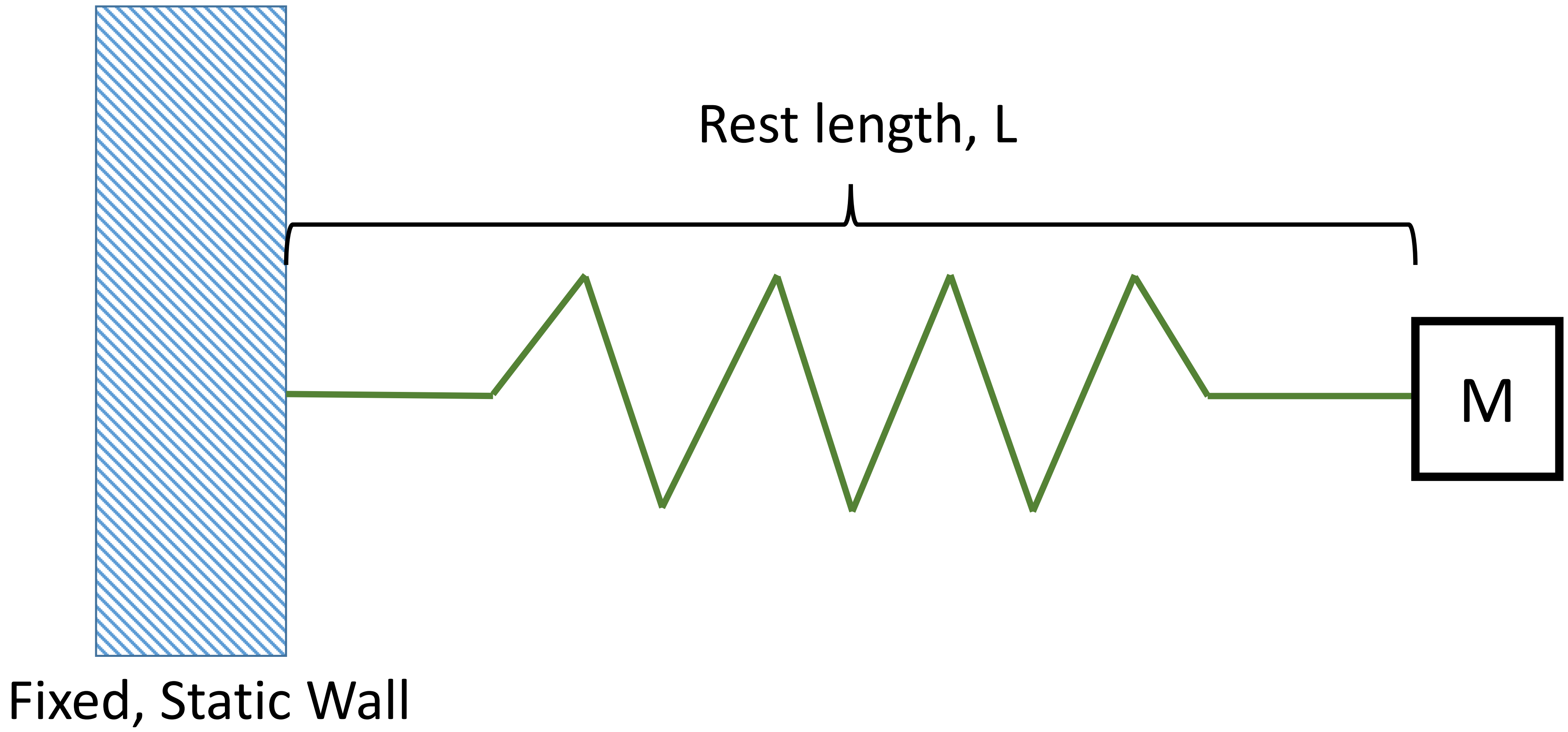
A simple way to model complex structures (e.g., hair, cloth, jello) is joining particles with *spring forces*.

AKA Mass-spring systems.

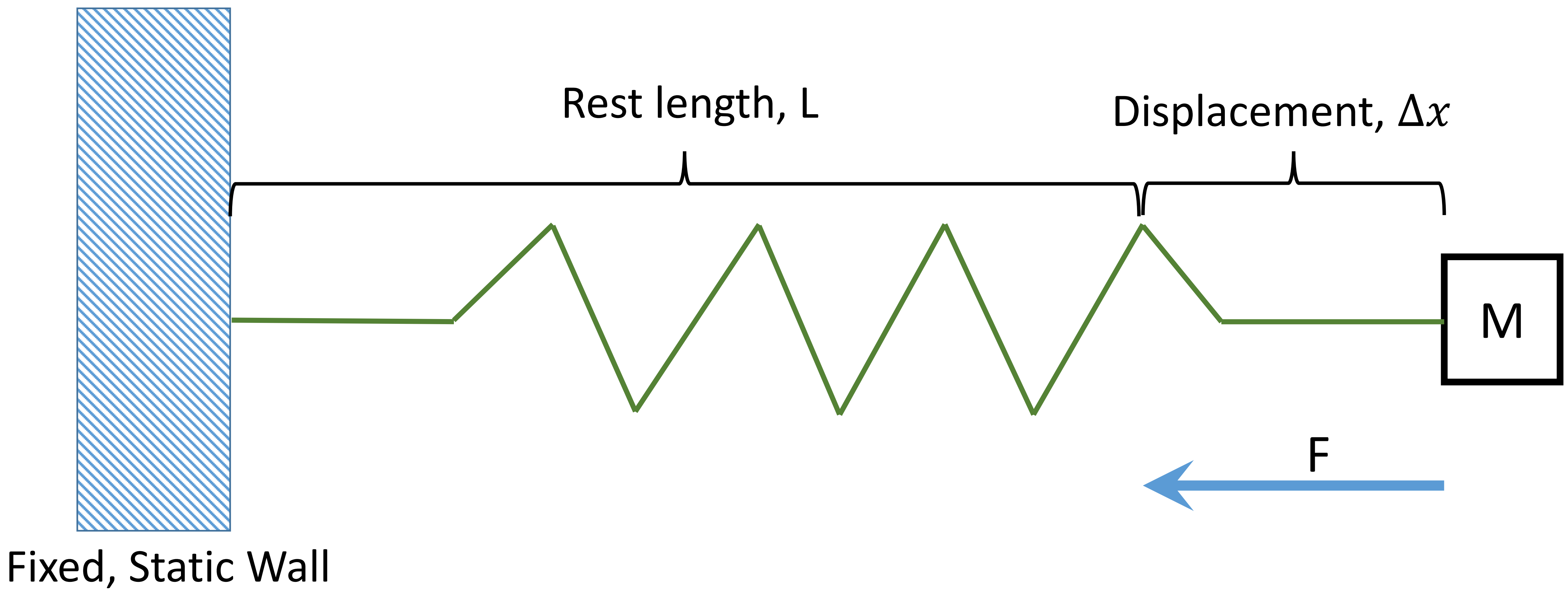
Each spring…

- connects two particles
- has a given *rest length*, *L*
- has a given “spring constant” or *stiffness*, *k*

Spring in 1D



Spring in 1D



Hooke’s Law for linear springs

The restoring force…

- Is linearly proportional to the amount of displacement (from the rest length).
- Acts in the opposite direction to the displacement.

$$F = -k\Delta x$$

where *k* is the proportionality constant that controls the spring stiffness. (Note: Stiffer materials typically require smaller timesteps!)

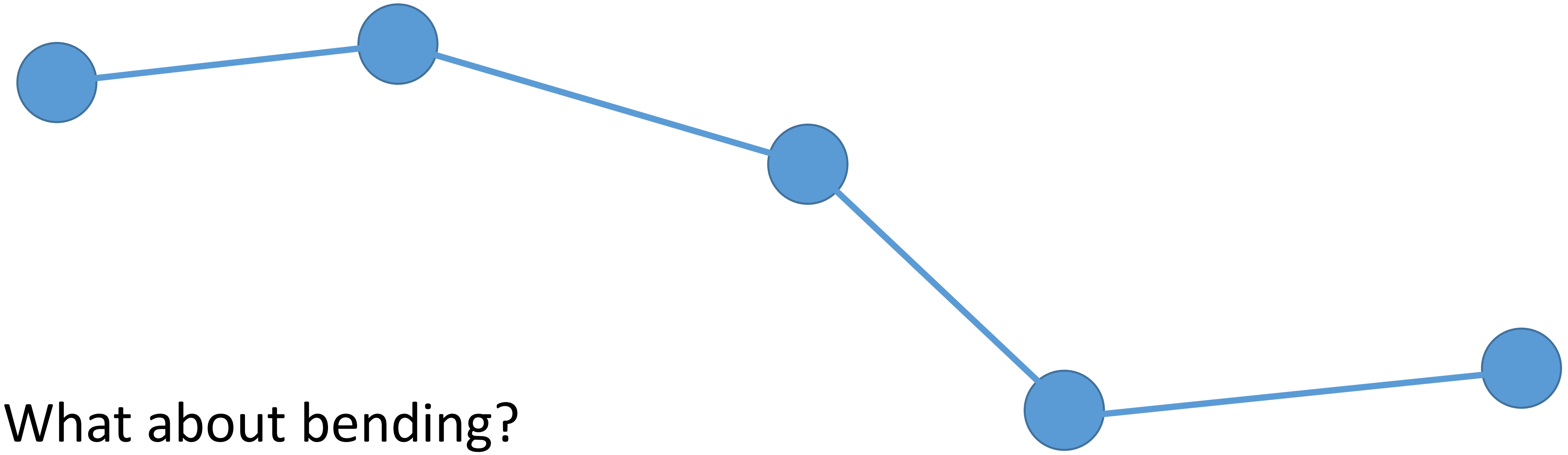
Hooke's Law in 3D

For a spring joining 2 particles with position vectors X_1 and X_2:

$$F_1 = -F_2 = -k\underbrace{(\|X_1 - X_2\| - L)}_{\text{Displacement}}\overbrace{\frac{X_1 - X_2}{\|X_1 - X_2\|}}^{\text{Direction}}$$

Springs for Hair and Cloth

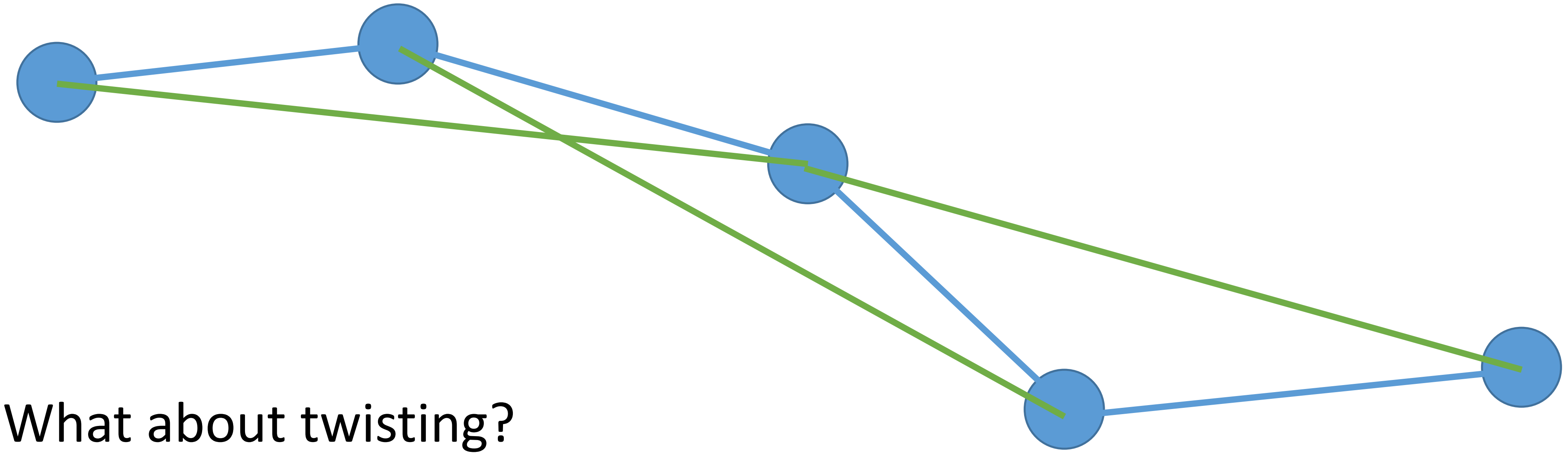
A single chain of masses and springs can model a strand of hair.



What about bending?

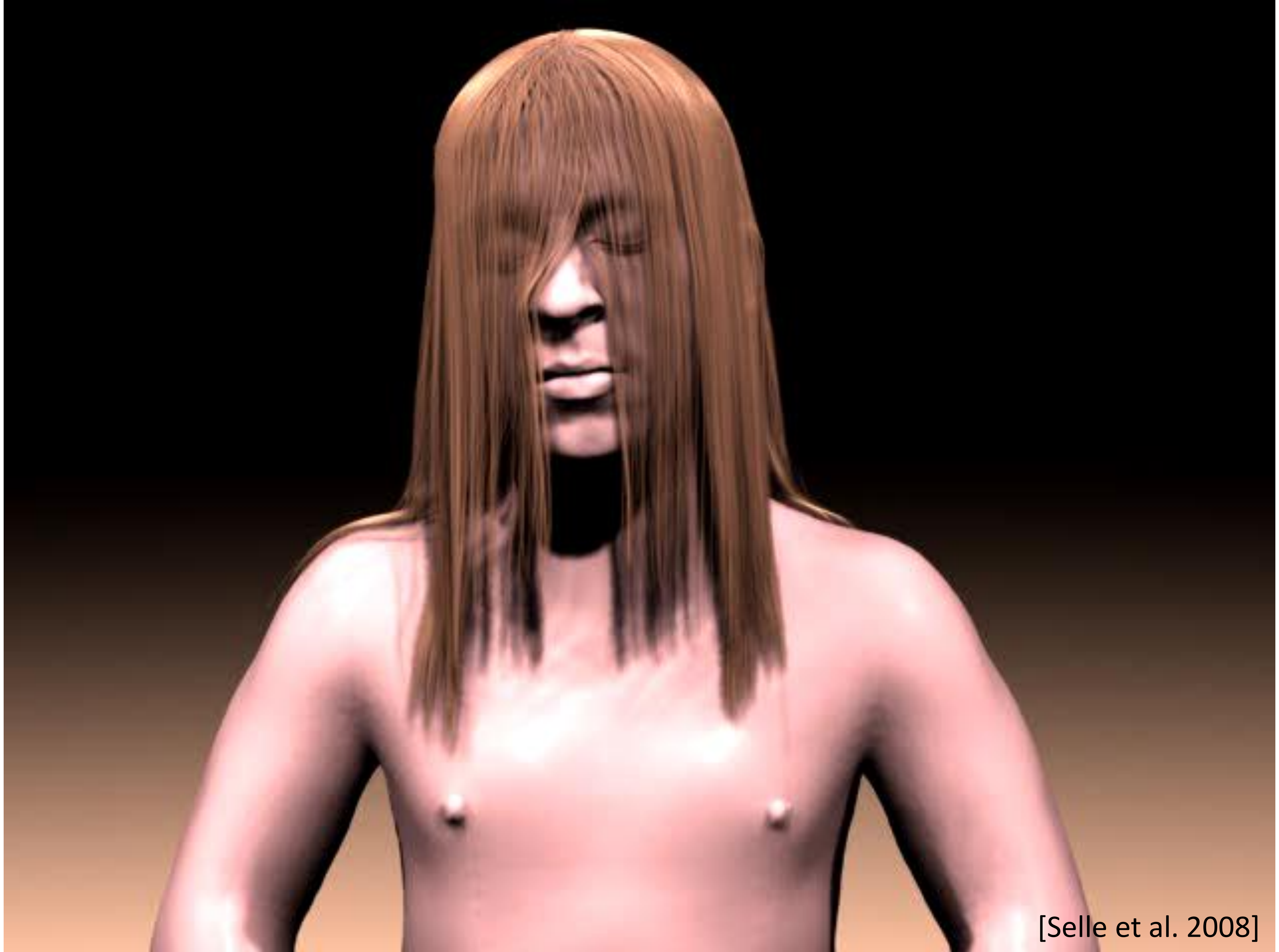
Springs for Hair and Cloth

Add *alternating* springs. This discourages the hair from collapsing when you bend it.



What about twisting?

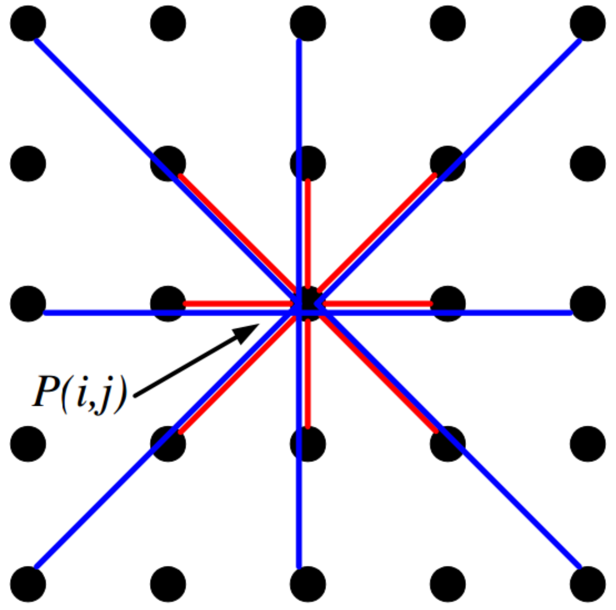
See “A Mass Spring Model for Hair Simulation” [Selle et al. 2008]


[Selle et al. 2008]

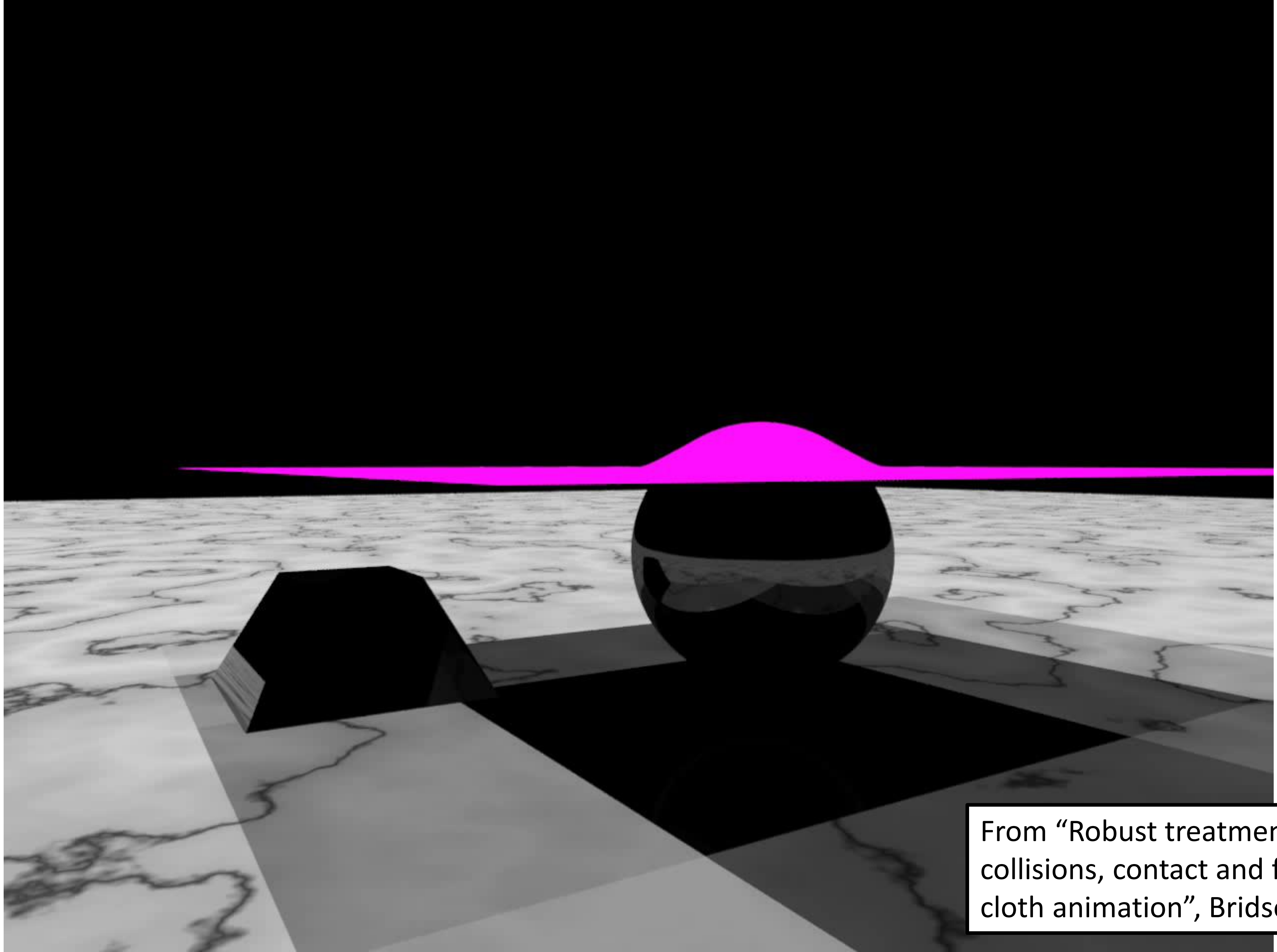
Springs for Cloth

Using a grid of particles, a similar approach has been used to model cloth.

- Red springs model stretching/shearing.
- Blue springs model bending.



From "Stable but responsive cloth", Choi and Ko, 2002.

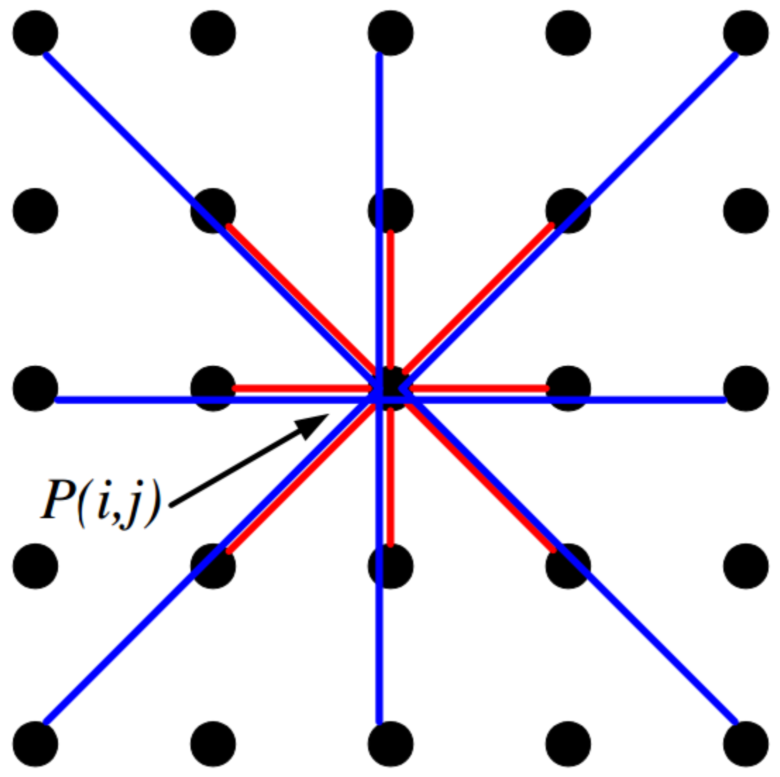


From “Robust treatment of collisions, contact and friction for cloth animation”, Bridson et al. 2002

Problems with mass-spring systems…

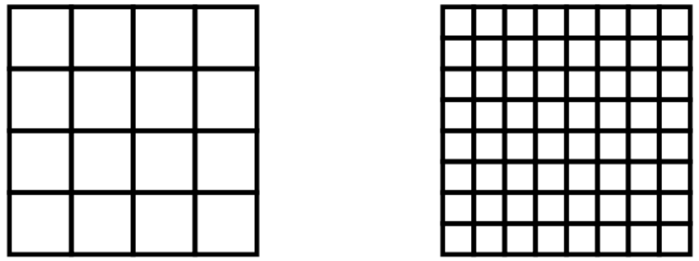
Can we control stretching and bending strength independently?

No! The bending springs also affect stretchiness of the material in an unclear way.



Problems with springs…

If we double the mesh resolution, will the behaviour remain the same?



Ideally, want the motion to approach a “true” solution as we *refine* the mesh. In general, springs do not.

More accurate *finite element* and *finite volume* approaches are often preferred in VFX now; games still mostly use mass-spring models.

Forces: Damping

It’s often helpful to gradually slow down the motion, so it doesn’t oscillate forever.

This models internal friction or energy dissipation.

A typical damping/drag force opposes the motion of the particle(s), and is proportional to the velocity.

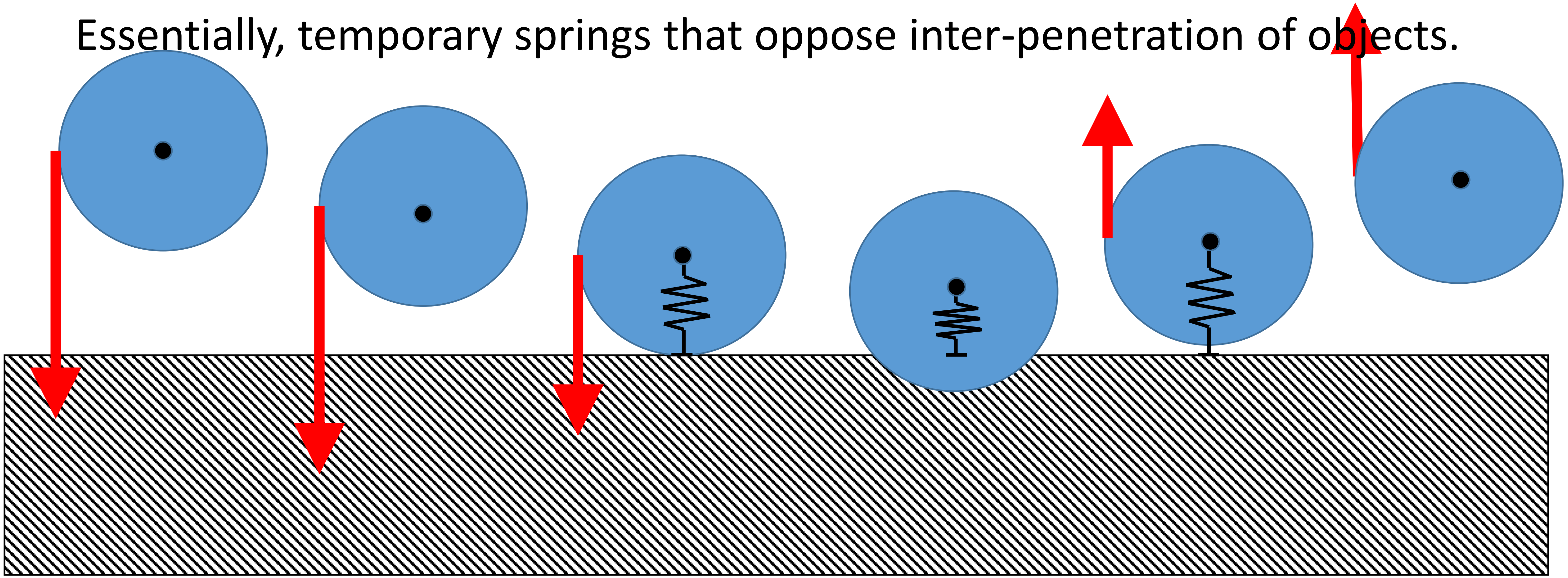
$$F = -k_d V$$

Note: Too much damping can yield unrealistic behavior.

Forces: Collisions

One strategy for modeling collisions is *penalty/repulsion forces*.

Essentially, temporary springs that oppose inter-penetration of objects.



Recap

- Particle systems can model diverse phenomena, in non-physical and physical ways.
- Time integration methods advance a simulation through time
  - e.g. forward Euler, midpoint, etc.
- By solving the equations of motion for particles and particle systems, we can capture more physically meaningful behaviours.

References

Reminder: A great reference to get more details is the set of SIGGRAPH course notes by Baraff and Witkin, from Pixar.

http://www.pixar.com/companyinfo/research/pbm2001/