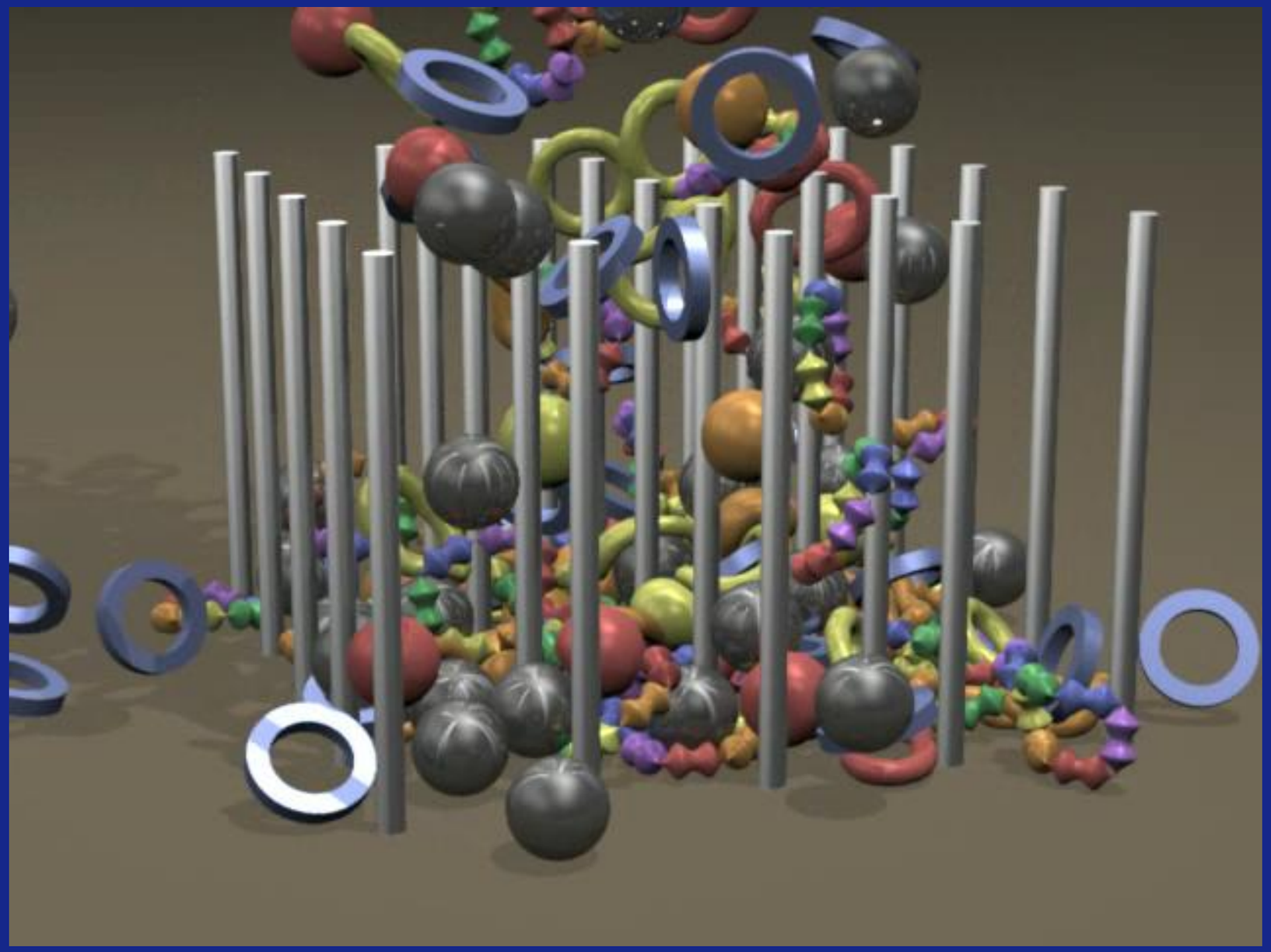
Differential Equations
&
Particle Systems

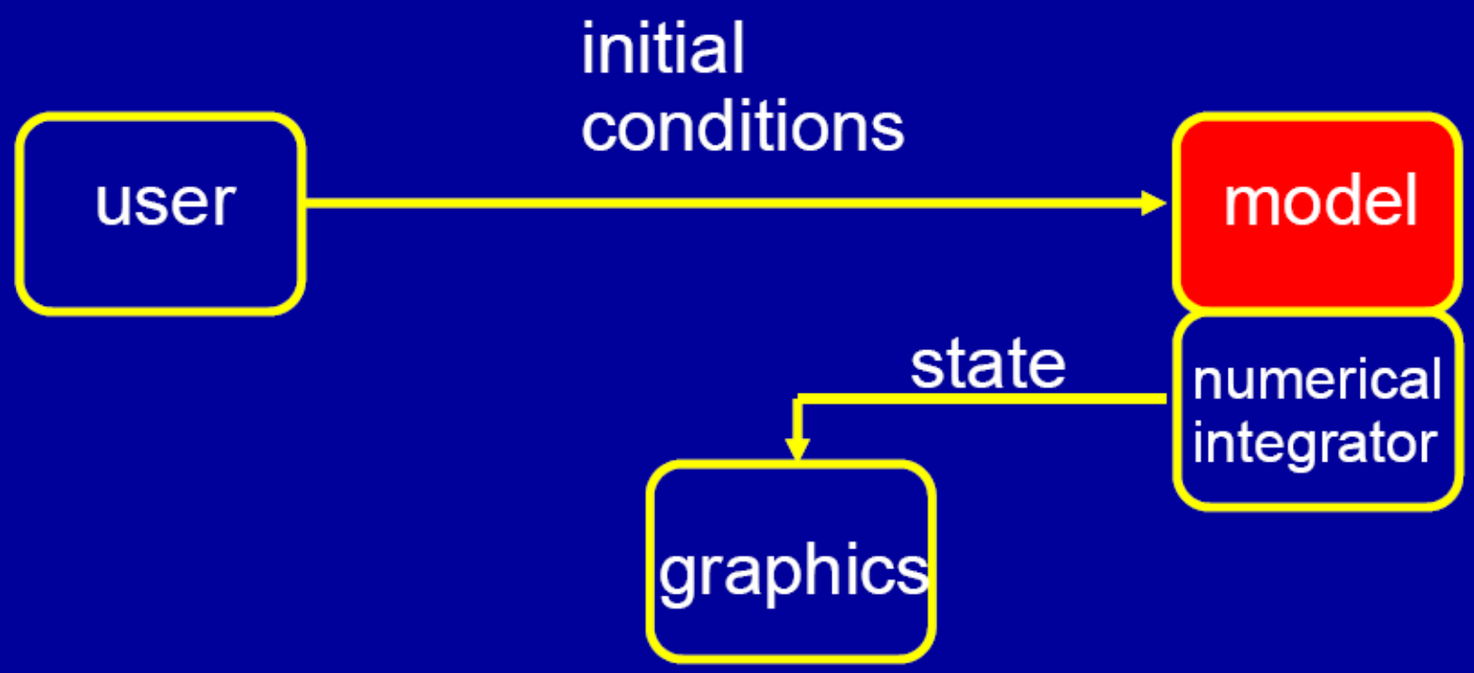
Thanks to Trueille, Popovic, Baraff, Witkin

Physics-based Animation



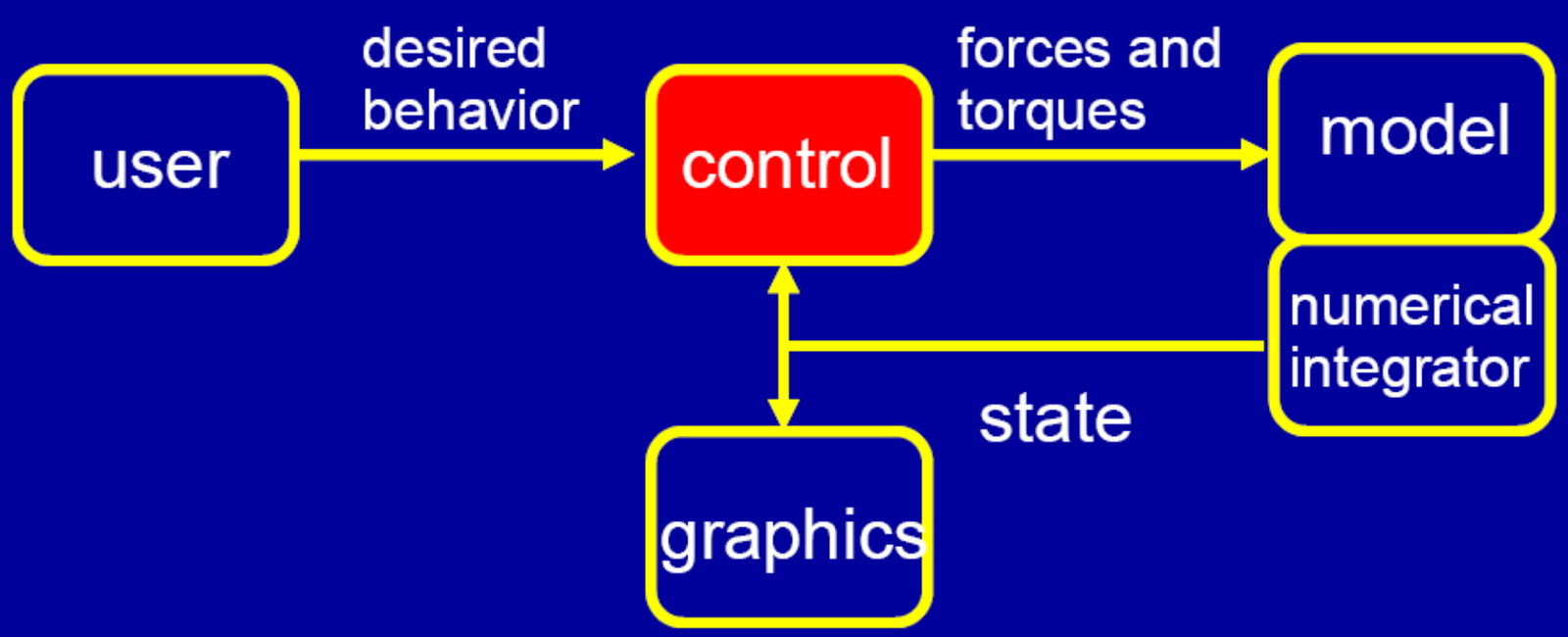
http://physbam.stanford.edu/~fedkiw/animations/large_pile.avi

Passive—no muscles or motors



Particle systems
Leaves
Water
Smoke
Clothing

Active—internal sources of energy

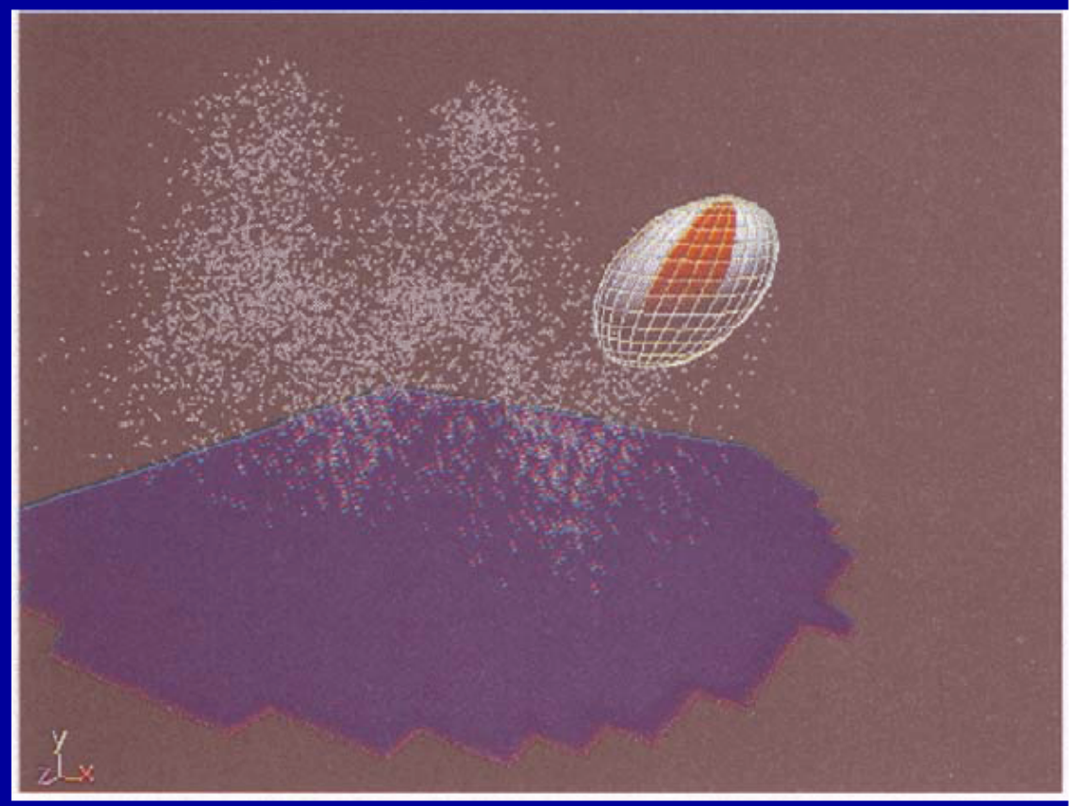


Running human
Trotting dog
Swimming fish

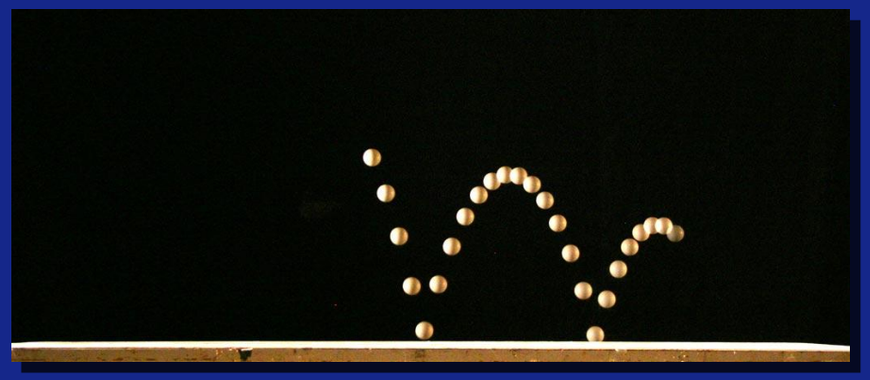
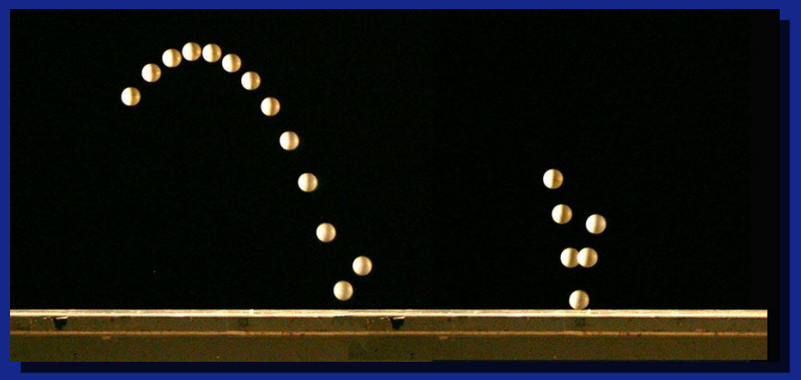
Dynamics

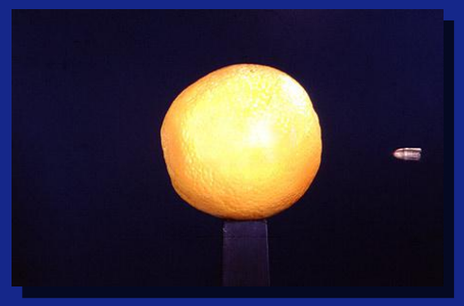
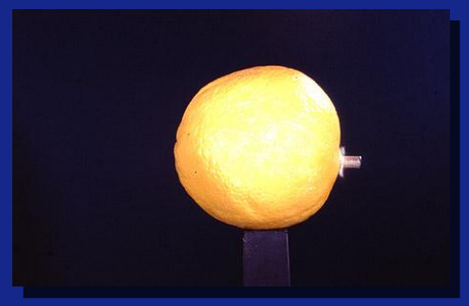
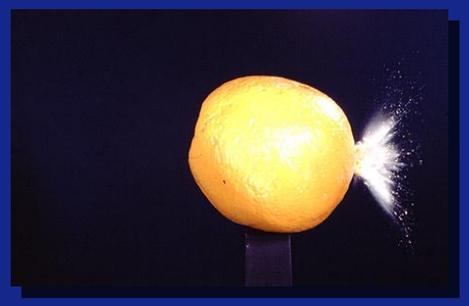
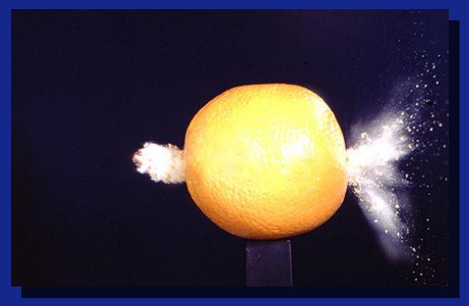
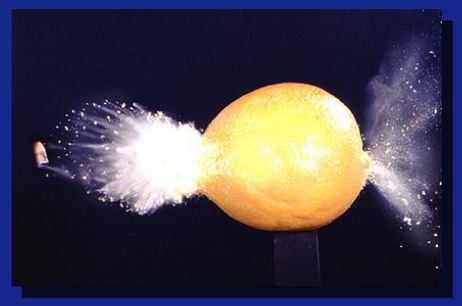
- Generate motion by specifying mass and force, apply physical laws (e.g., Newton's laws)
  - particles
  - soft objects
  - rigid bodies
- Simulates physical phenomena
  - gravity
  - momentum (inertia)
  - collisions
  - friction
  - fluid flow (drag, turbulence, ...)
  - solidity, flexibility, elasticity
  - fracture

Maya Dynamics



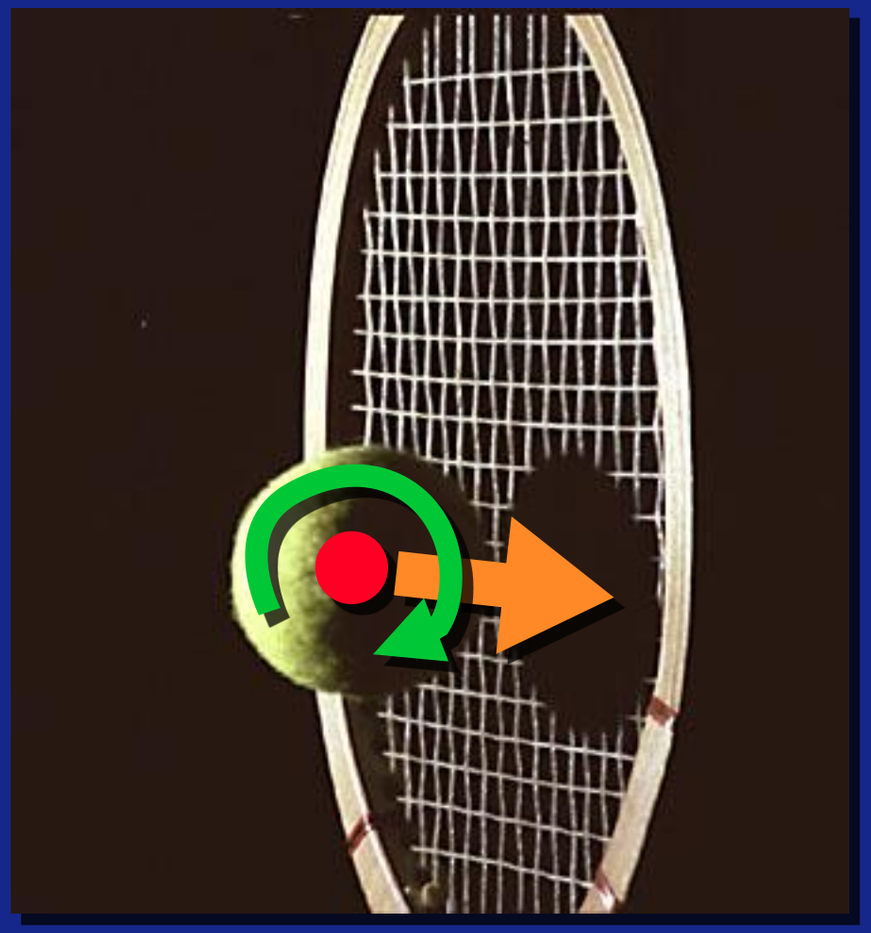
Describing Physics

source: http://people.rit.edu/andpph/exhibit-8.html

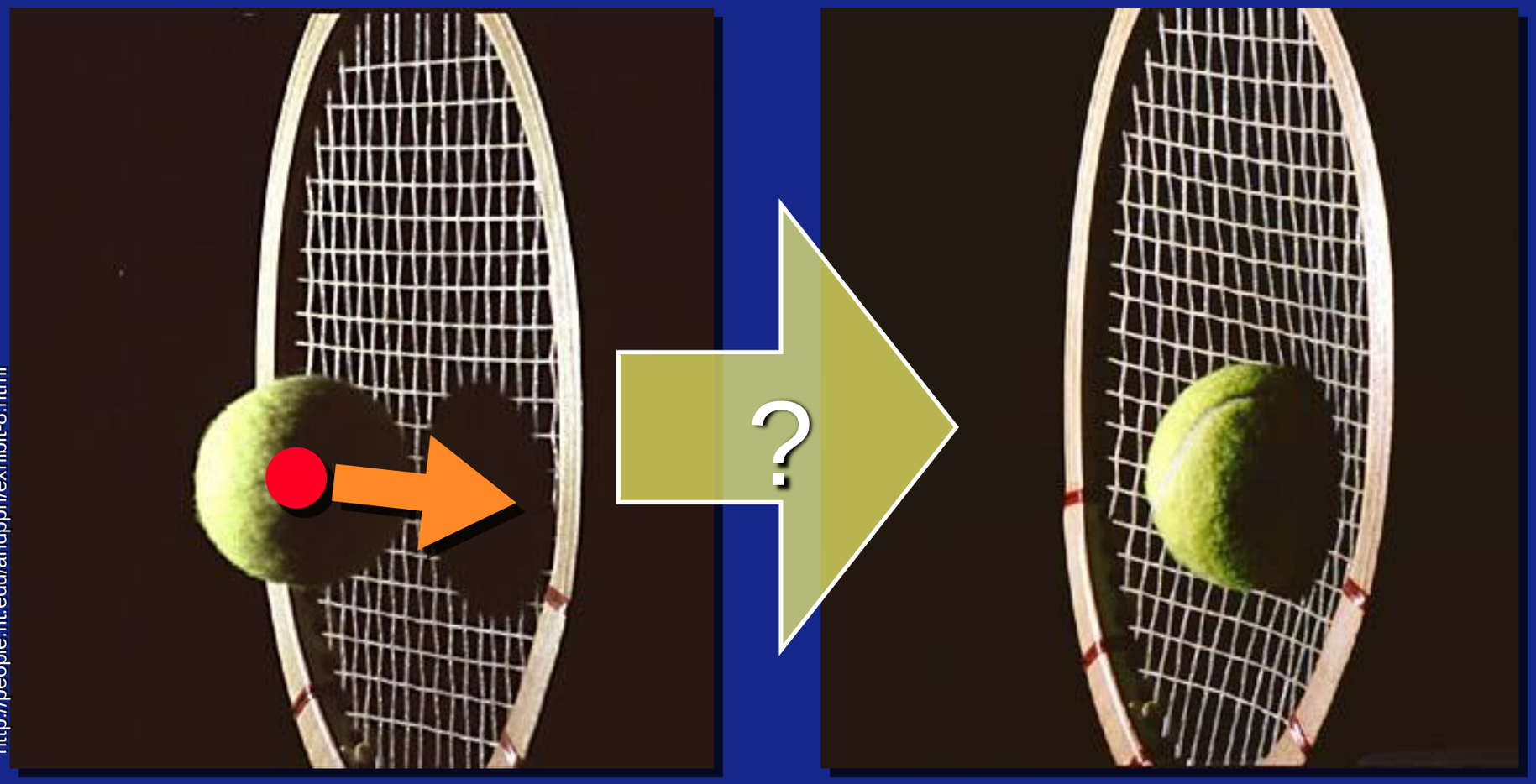
What variables do we need?



http://people.rit.edu/andpph/exhibit-8.html

Static	Dynamic
• Radius	• Position
• Mass	• Velocity
• Racquet Info	• Rotation?

What Happens Next?



http://people.rit.edu/andpph/exhibit-8.html

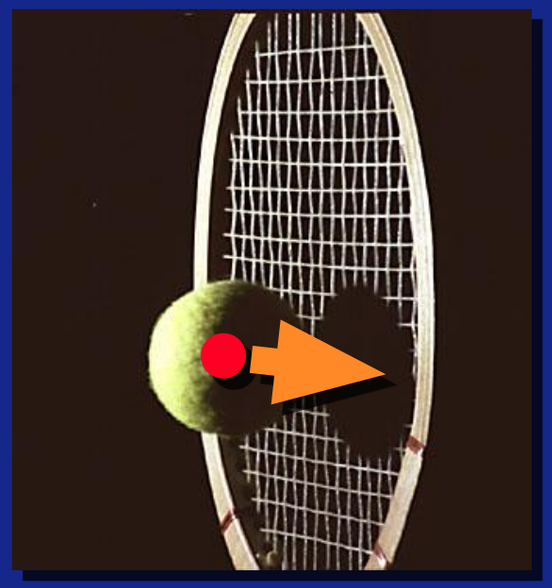
- Position
- Velocity

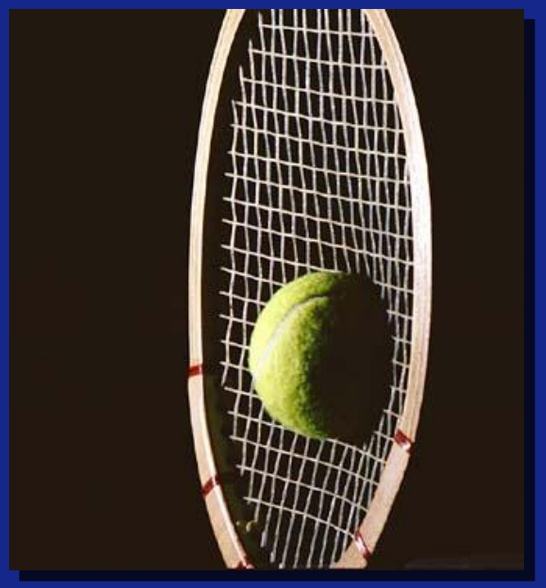
$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Discrete Time: $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t)$

Continuous Time: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$

Differential Equations





$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

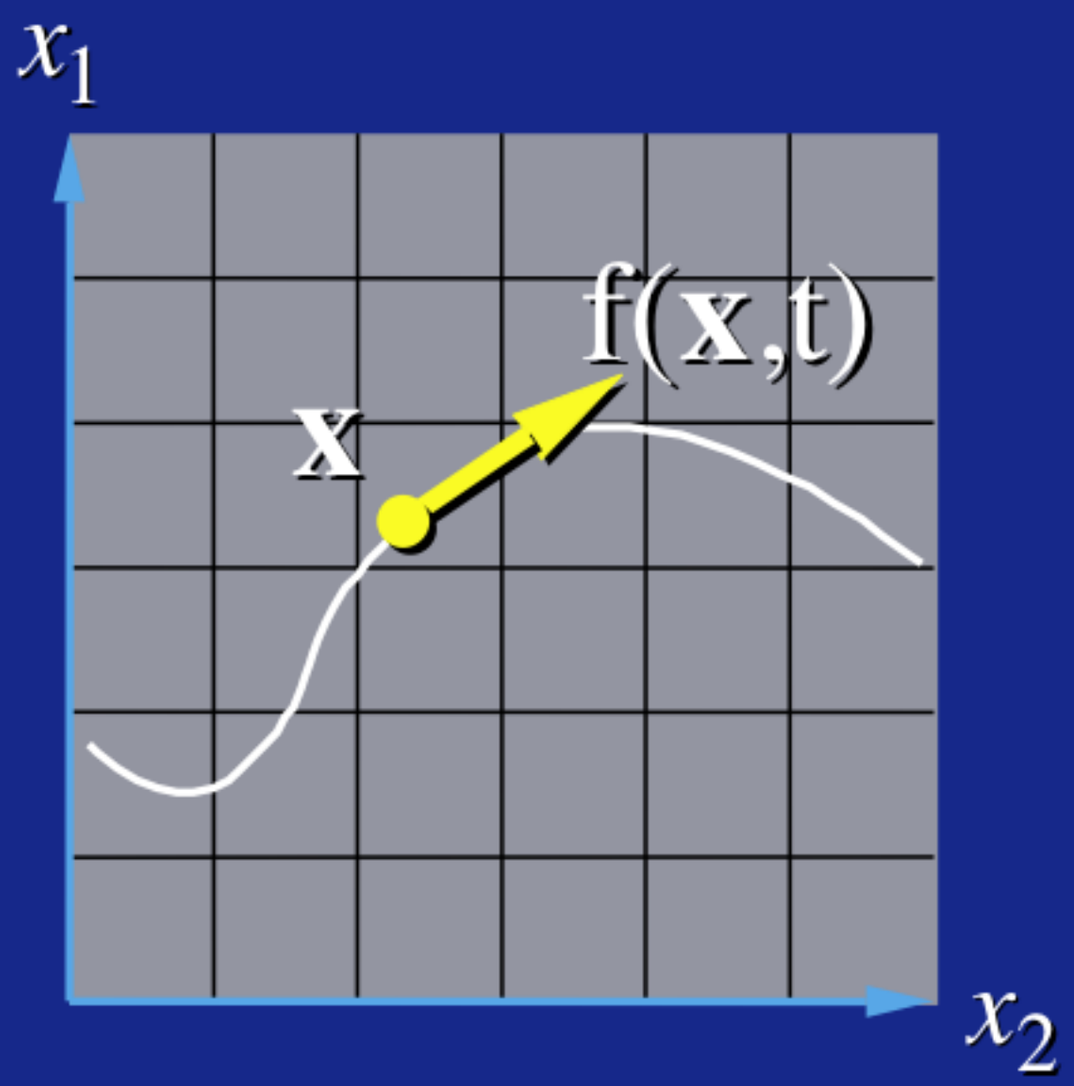
Differential Equation Basics

Andrew Witkin

PIXAR
ANIMATION STUDIOS

SIGGRAPH 2001 COURSE NOTES SBI1 PHYSICALLY BASED MODELING

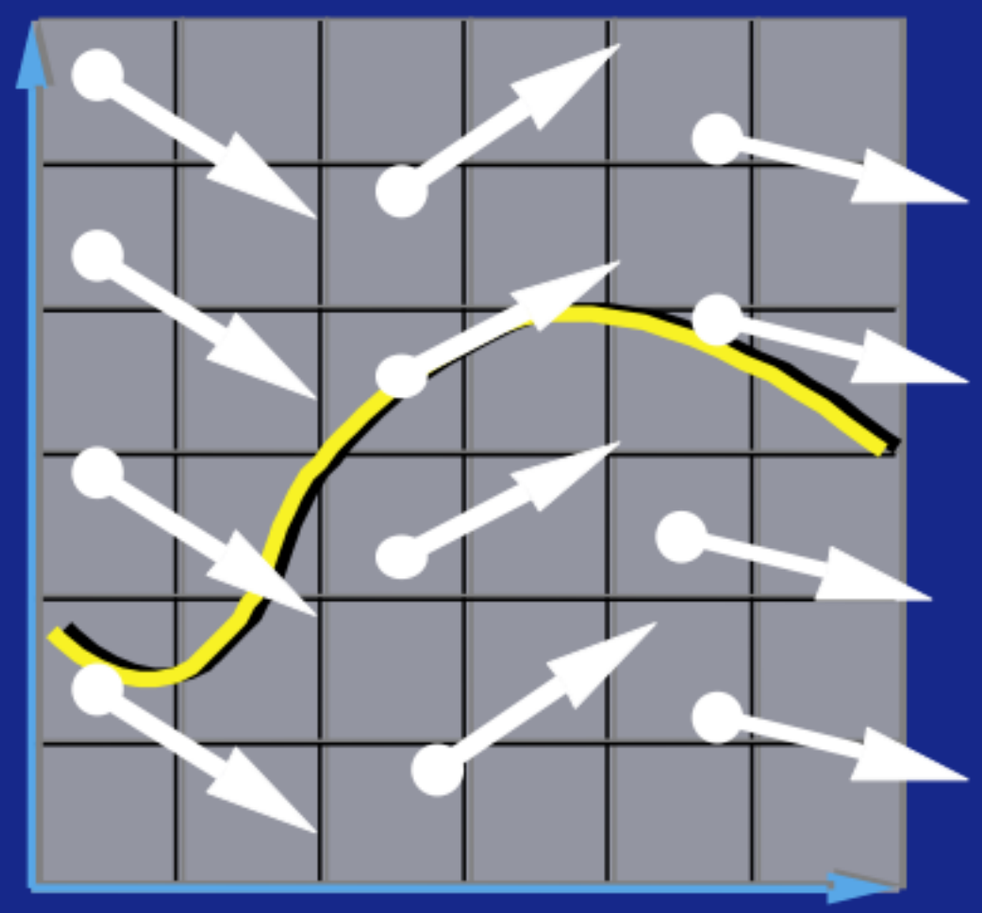
A Canonical Differential Equation



$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$$

- **x(t): a moving point.**
- **f(x,t): x's velocity.**

Vector Field



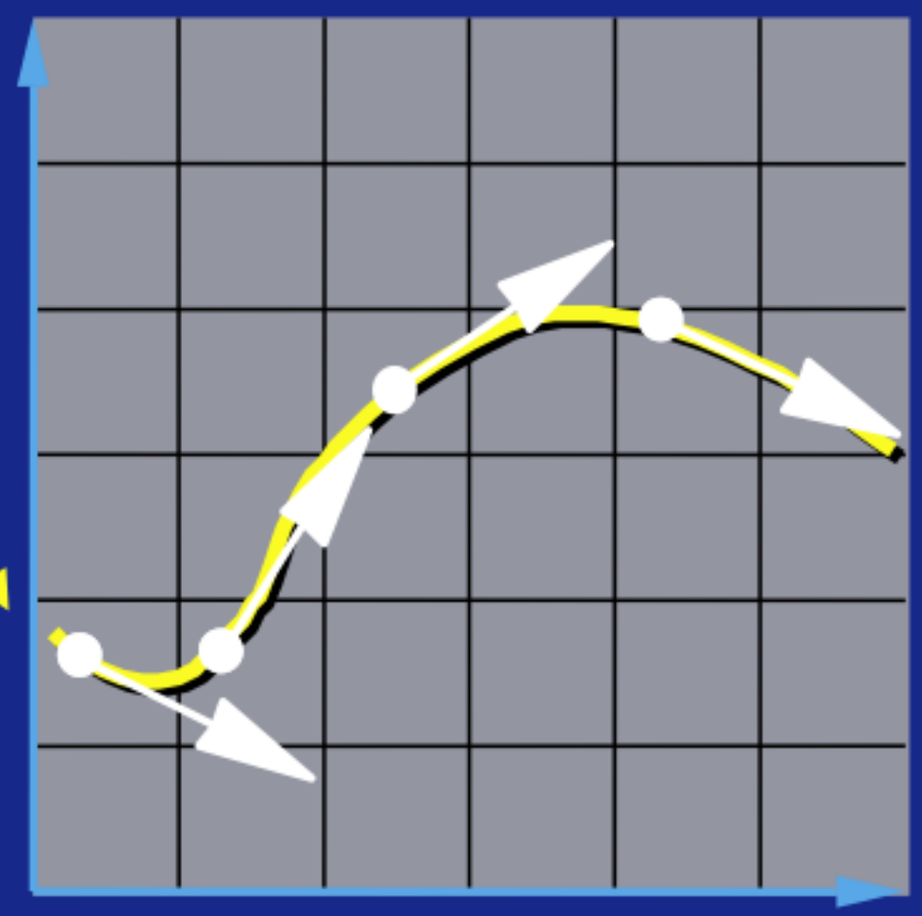
The differential equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$$

defines a vector field over x.

Integral Curves

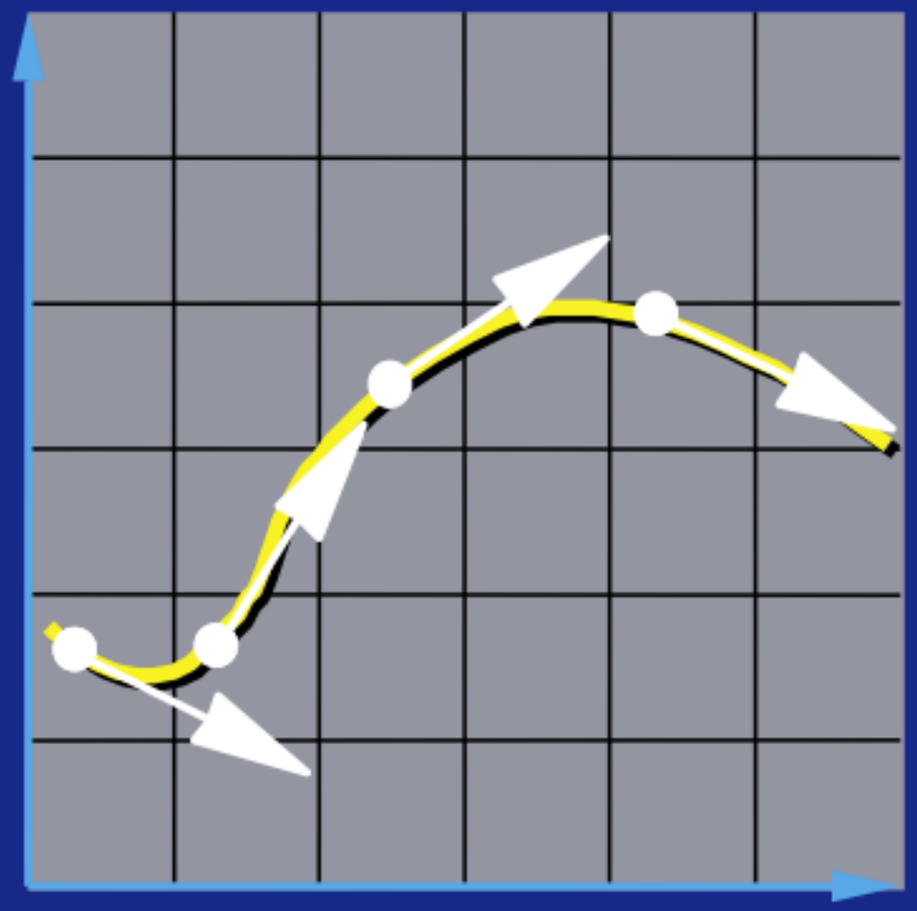
Start Here



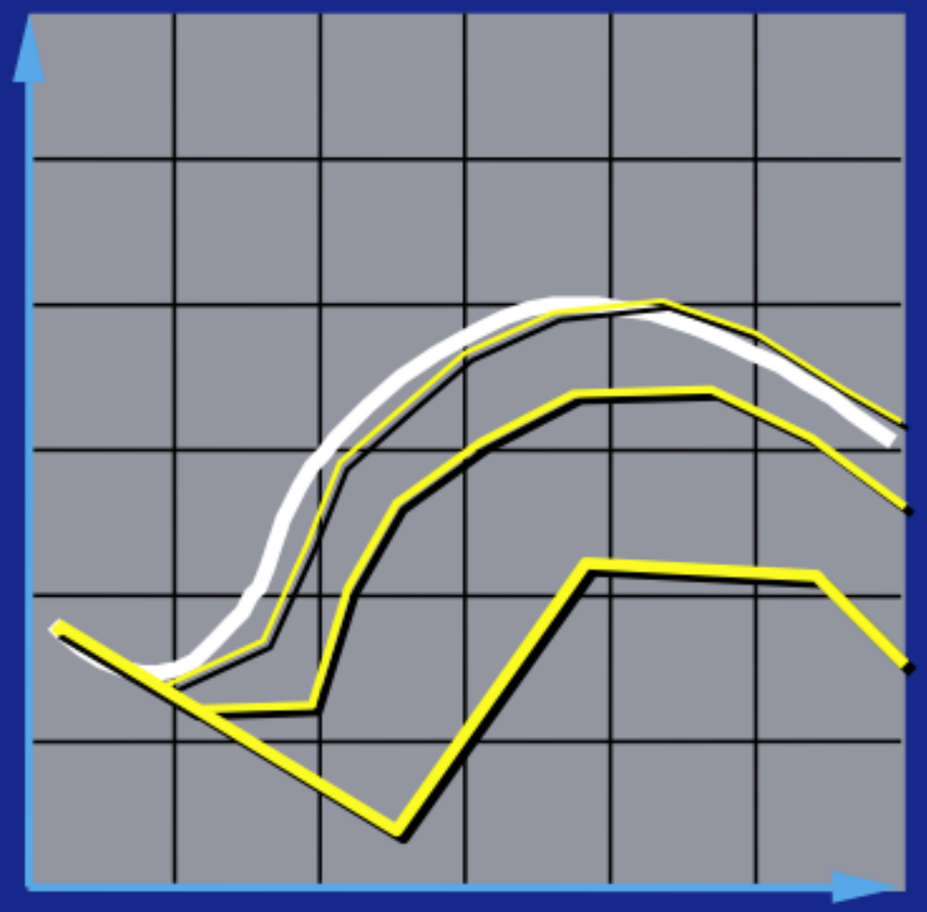
Pick any starting point,
and follow the vectors.

Initial Value Problems

Given the starting point, follow the integral curve.



Euler's Method



- Simplest numerical solution method
- Discrete time steps
- Bigger steps, bigger errors.

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{f}(\mathbf{x}, t)$$

Two Problems

- Accuracy
- Instability

Accuracy

Consider the equation:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{x}$$

What do the integral curves look like?

Problem I: Inaccuracy

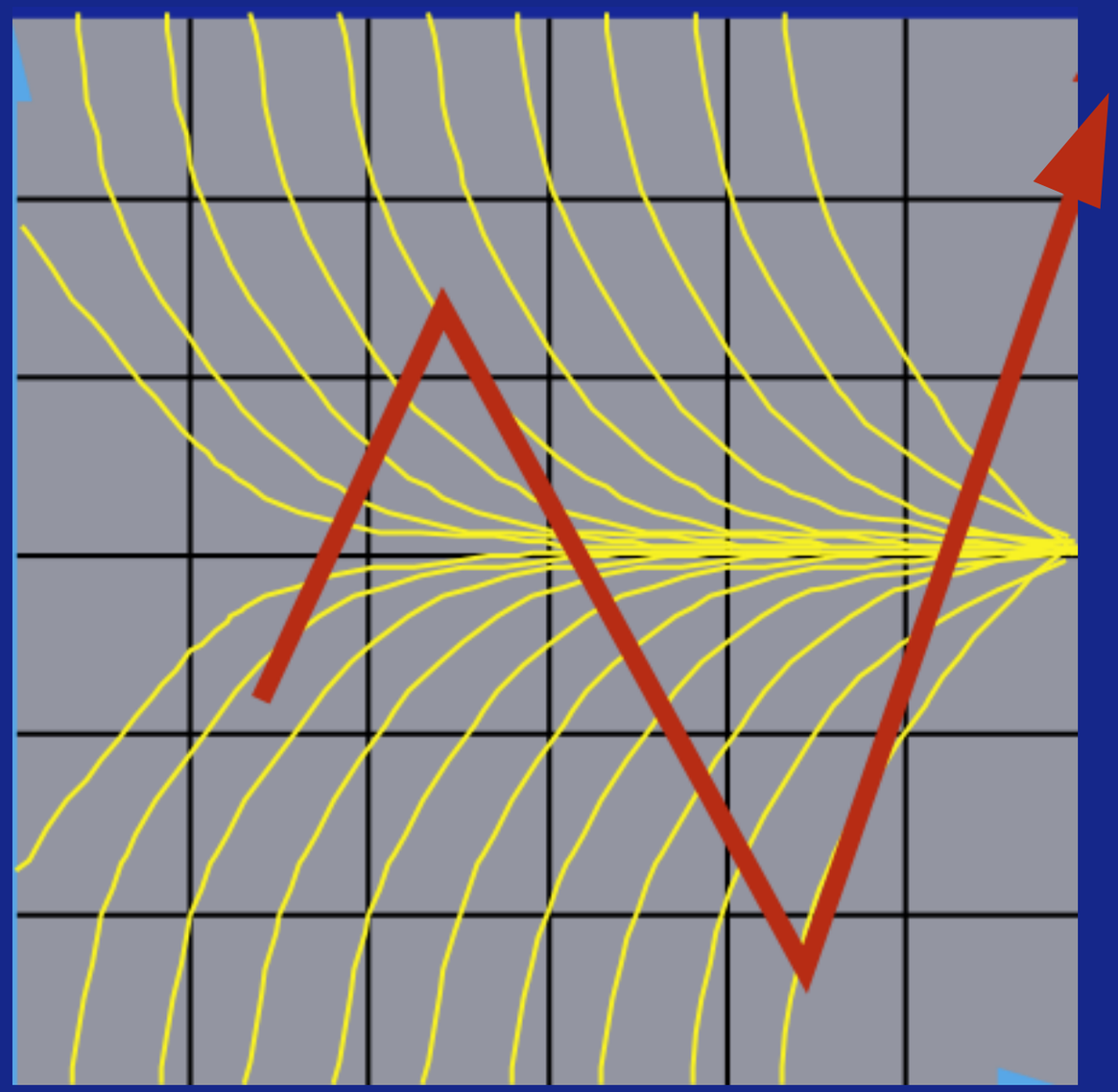
Error turns x(t) from a circle into the spiral of your choice.

Problem 2: Instability

- Consider the following system:

$$\begin{cases} \dot{x} = -x \\ x(0) = 1 \end{cases}$$

Problem 2: Instability



to Neptune!

Accuracy of Euler Method

$$\dot{x} = f(x)$$

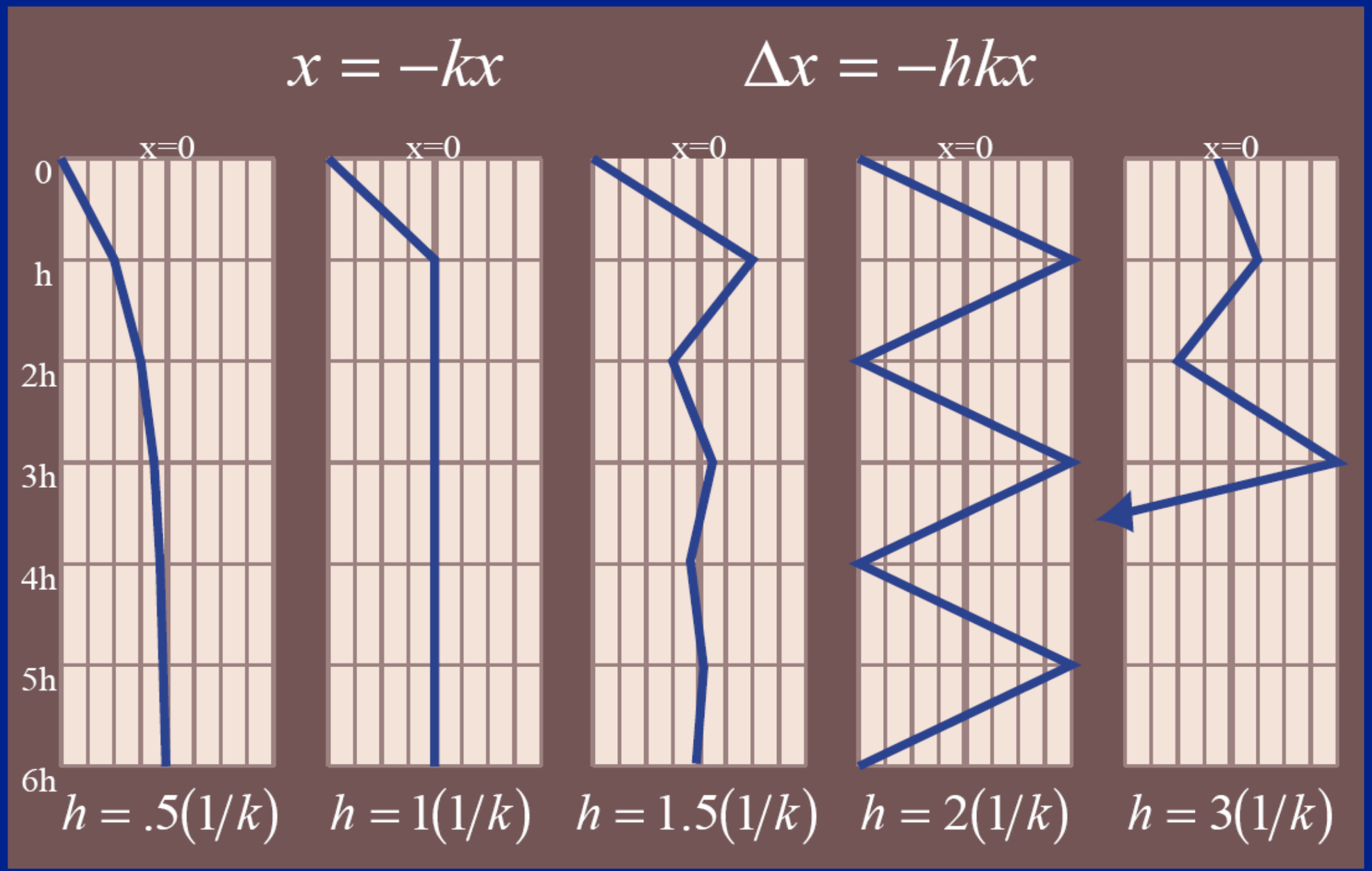
Consider Taylor Expansion about x(t)...

$$x(t+h) = \overbrace{\underbrace{x(t)}_{\text{constant}} + \underbrace{h f(x(t))}_{\text{linear}}}^{\text{Euler step}} + \overbrace{\underbrace{O(h^2)}_{\text{everything else}}}^{\text{error}}$$

Euler's method has error $O(h^2)$... first order.

How can we get to $O(h^3)$ error?

Euler's method has a speed limit



$h > 1/k$: oscillate. $h > 2/k$: explode!

The Midpoint Method

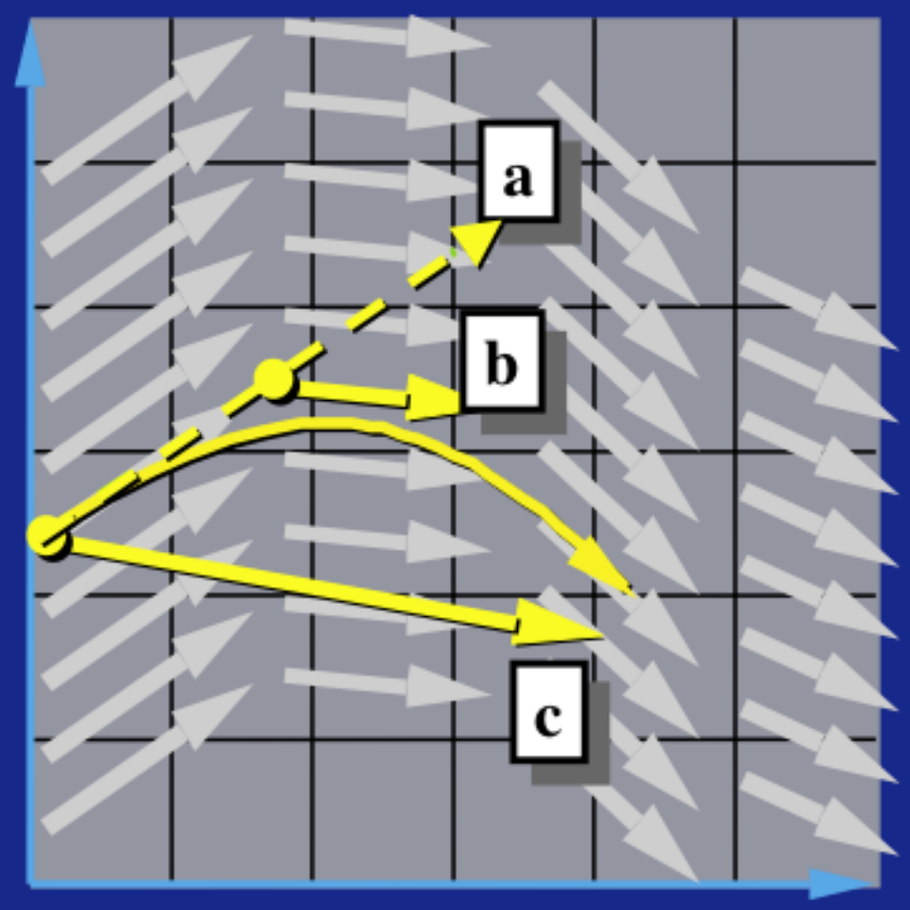
- Also known as second order Runge-Kutta:

$$k_1 = h(f(x_0, t_0))$$

$$k_2 = hf(x_0 + \frac{k_1}{2}, t_0 + \frac{h}{2})$$

$$x(t_0 + h) = x_0 + k_2 + O(h^3)$$

The Midpoint Method



a. Compute an Euler step

$$\Delta \mathbf{x} = \Delta t\, \mathbf{f}(\mathbf{x}, t)$$

b. Evaluate f at the midpoint

$$\mathbf{f}_{\text{mid}} = \mathbf{f}\left(\frac{\mathbf{x} + \Delta \mathbf{x}}{2}, \frac{t + \Delta t}{2}\right)$$

c. Take a step using the midpoint value

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{f}_{\text{mid}}$$

4th-Order Runge-Kutta

$$k_1 = hf(x_0, t_0)$$

$$k_2 = hf(x_0 + \frac{k_1}{2}, t_0 + \frac{h}{2})$$

$$k_3 = hf(x_0 + \frac{k_2}{2}, t_0 + \frac{h}{2})$$

$$k_4 = hf(x_0 + k_3, t_0 + h)$$

$$x(t_0 + h) = x_0 + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(h^5)$$

Very popular

q-Stage Runge-Kutta

General Form:

$$x(t_0 + h) = x_0 + h \sum_{i=1}^{q} w_i k_i$$

where:

$$k_i = f\left(x_0 + h \sum_{j=1}^{i-1} \beta_{ij} k_j \right)$$

Find the constant that ensure accuracty $O(h^n)$.

stability is all stability is all stability is all

- If your step size is too big, your simulation blows up. It isn't pretty.
- Sometimes you have to make the step size so small that you never get anyplace.
- Nasty cases: cloth, constrained systems.

Implicit Euler Method

$$x(t_0 + h) = x(t_0) + h\dot{x}(t_0)$$

$$x(t_0 + h) = x(t_0) + h\dot{x}(t_0 + \Delta t)$$

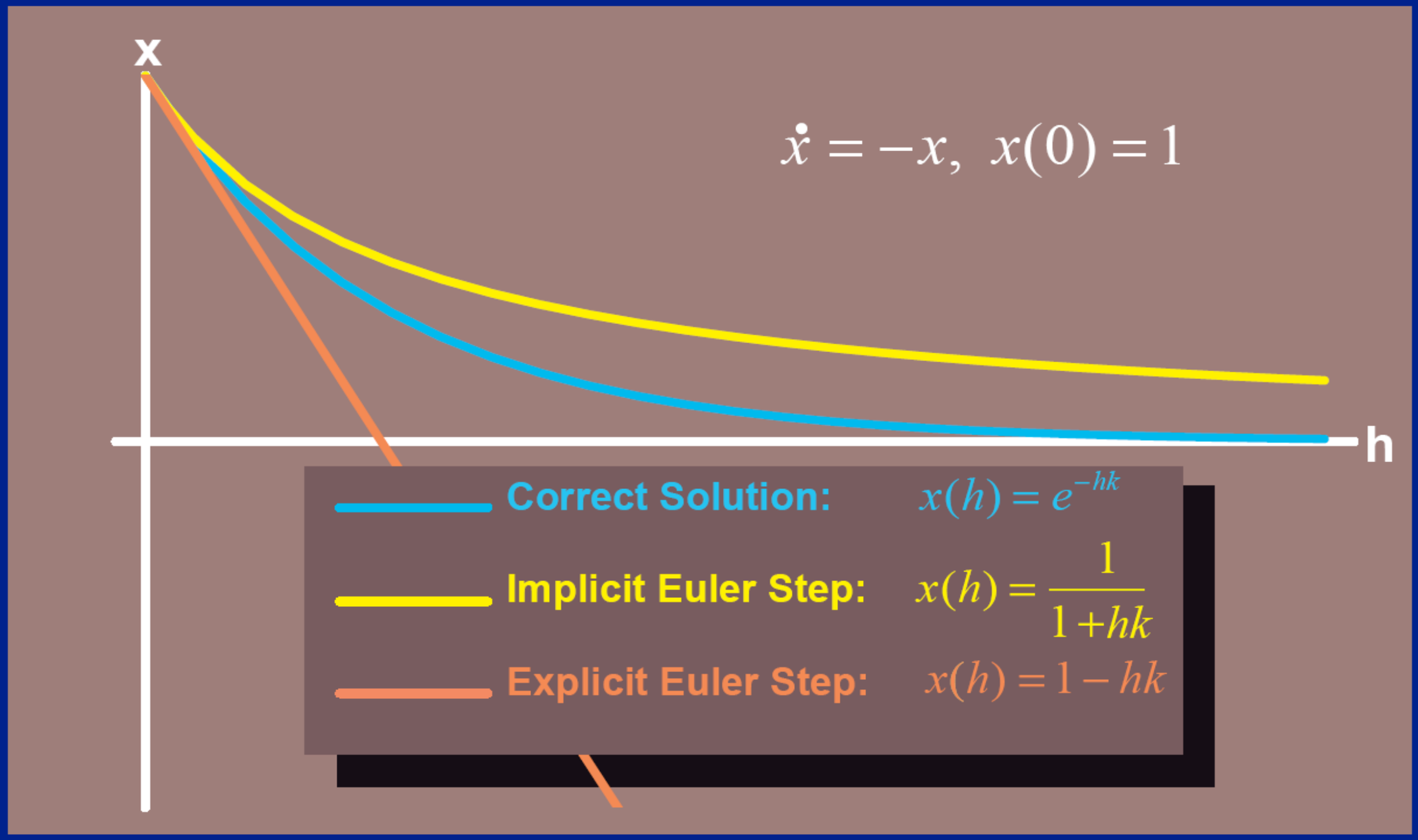
Implicit Euler for $\dot{x} = -kx$

$$x(t+h) = x(t) + h\dot{x}(t+h)$$

$$= x(t) - hkx(t+h)$$

$$= \frac{x(t)}{1+hk}$$

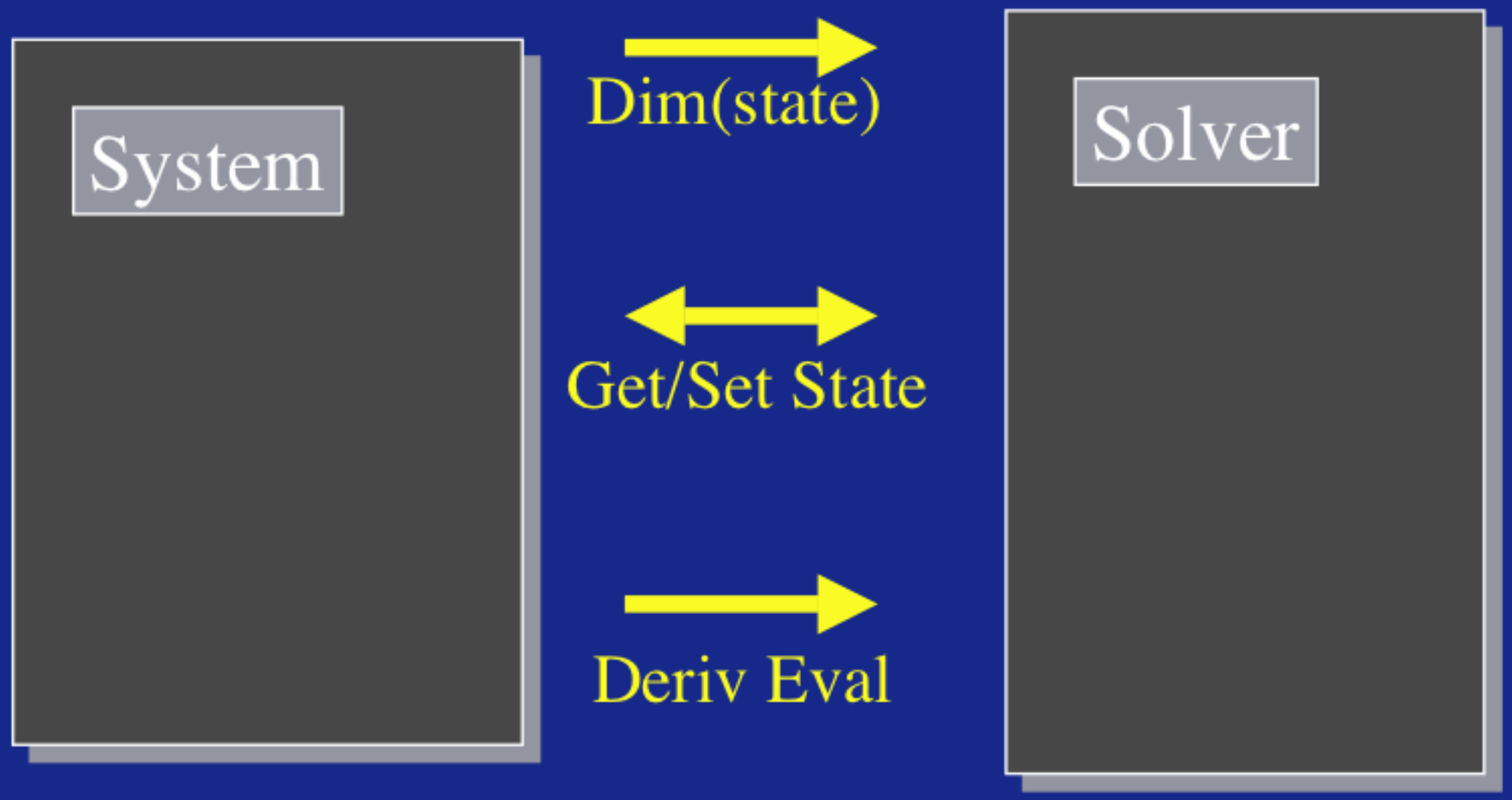
One Step: Implicit vs. Explicit



Modular Implementation

- Generic operations:
  - Get dim(x)
  - Get/set x and t
  - Deriv Eval at current (x,t)
- Write solvers in terms of these.
  - Re-usable solver code.
  - Simplifies model implementation.

Solver Interface



A Code Fragment

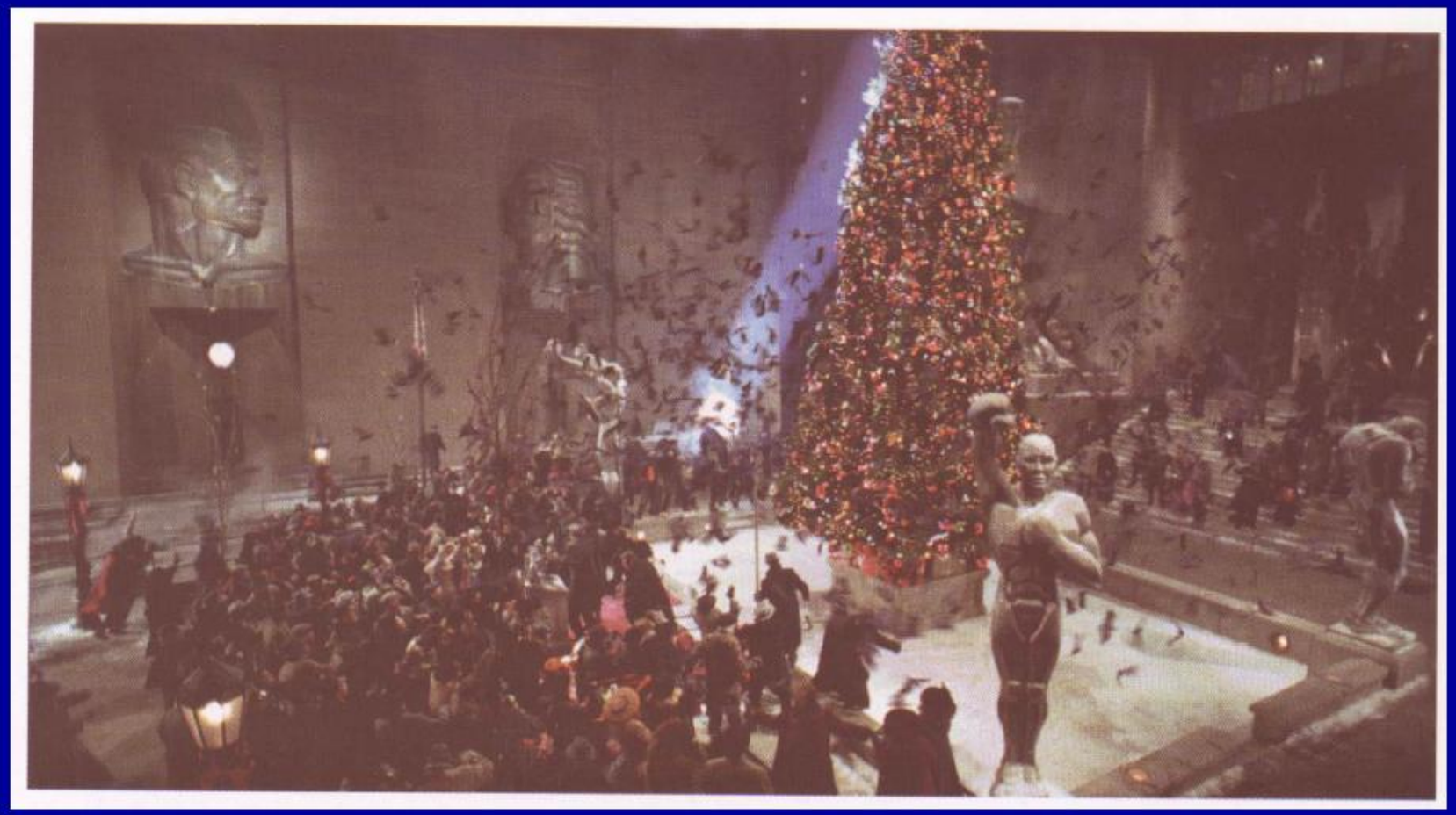
```
void eulerStep(Sys sys, float h) {
   float t = getTime(sys);
   vector<float> x0, deltaX;

   t = getTime(sys);
   x0 = getState(sys);
   deltaX = derivEval(sys,x0, t);
   setState(sys, x0 + h*deltaX, t+h);
}
```

Particle Systems

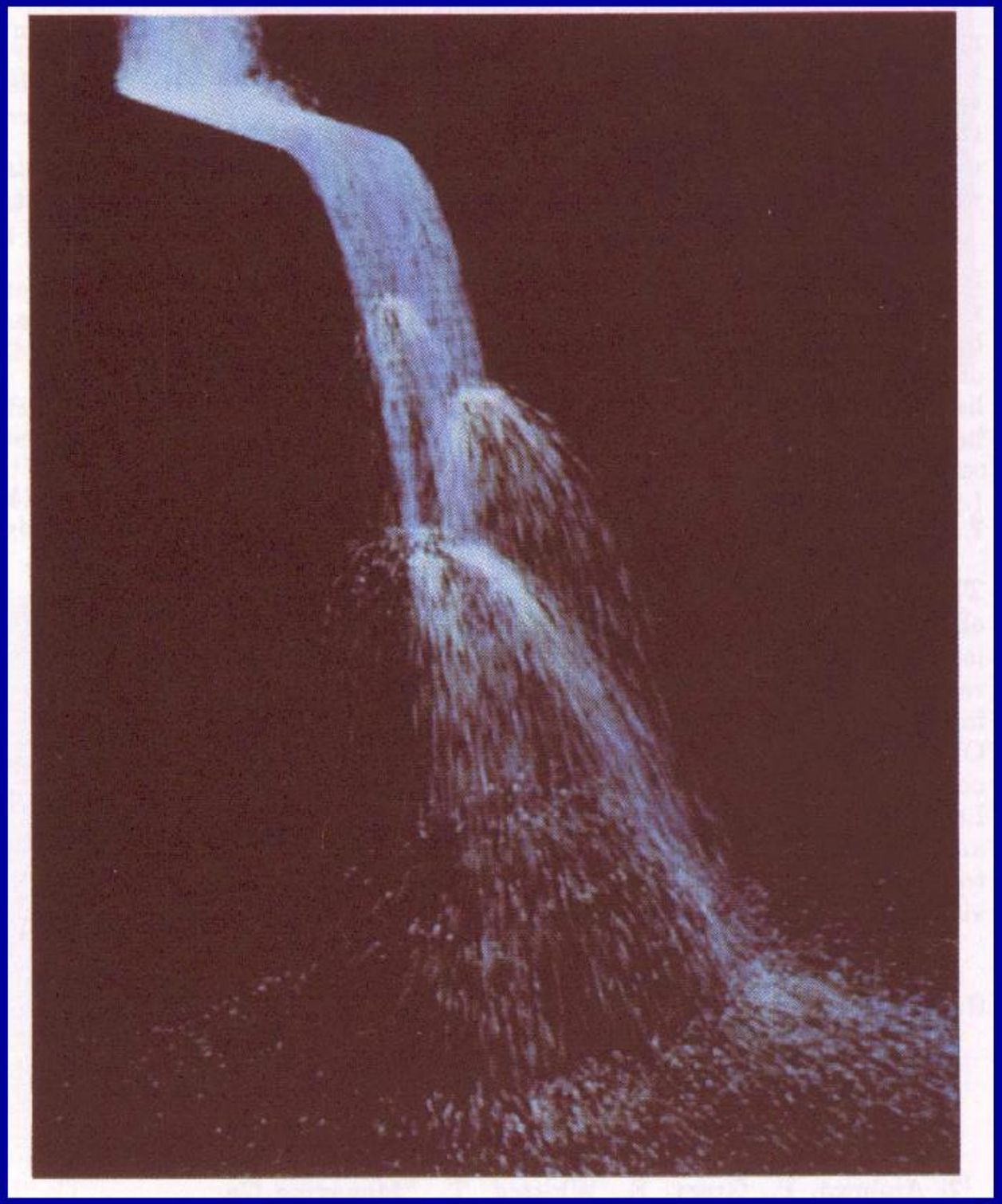
Particle Systems

Clouds
Smoke
Fire
Waterfalls
Fireworks



Reeves ’83, the Wrath of Khan
Batman Returns, using Reynold’s flocking algorithms

Karl Sims, Particle Dreams



What are particle systems?

A **particle system** is a collection of point masses that obeys some physical laws (e.g, gravity or spring behaviors).

Particle systems can be used to simulate all sorts of physical phenomena:

- Smoke
- Snow
- Fireworks
- Hair
- Cloth
- Snakes
- Fish

Overview

1. One lousy particle
2. Particle systems
3. Forces: gravity, springs
4. Implementation

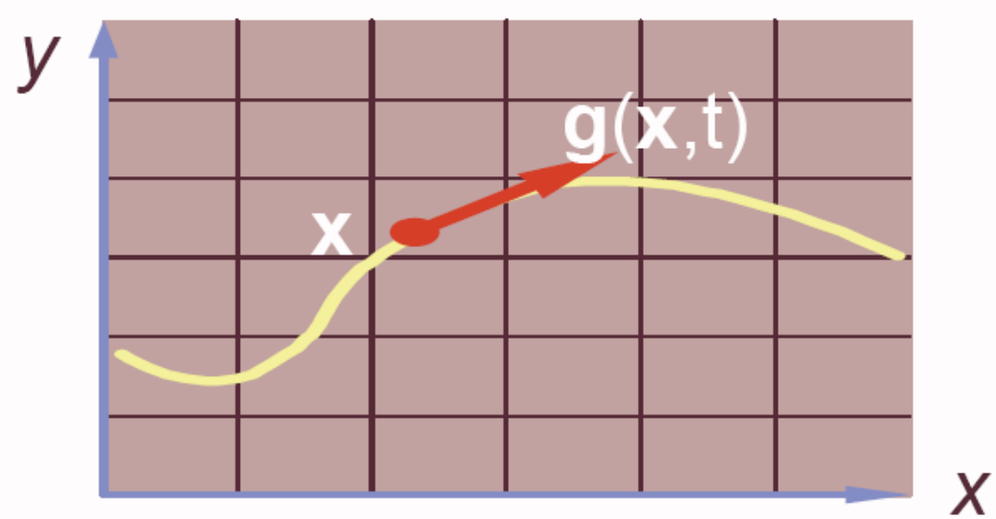
Particle in a flow field

We begin with a single particle with:

- Position, $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
- Velocity, $\mathbf{v} \equiv \dot{\mathbf{x}} = \dfrac{d\mathbf{x}}{dt} = \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix}$

Suppose the velocity is dictated by some driving function **g**:

$$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, t)$$



Particle in a force field

- Now consider a particle in a force field **f**.
- In this case, the particle has:
  - Mass, m
  - Acceleration, $\mathbf{a} \equiv \ddot{\mathbf{x}} = \dfrac{d\mathbf{v}}{dt} = \dfrac{d^2\mathbf{x}}{dt^2}$
- The particle obeys Newton's law: $\mathbf{f} = m\mathbf{a} = m\ddot{\mathbf{x}}$
- The force field **f** can in general depend on the position and velocity of the particle as well as time.
- Thus, with some rearrangement, we end up with:

$$\ddot{\mathbf{x}} = \frac{\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t)}{m}$$

Second order equations

This equation: $\ddot{\mathbf{x}} = \dfrac{\mathbf{f}(\mathbf{x}, \mathbf{v}, t)}{m}$

is a **second order differential equation**.

Our solution method, though, worked on first order differential equations.

We can rewrite this as: $\begin{bmatrix} \dot{\mathbf{x}} = \mathbf{v} \\ \dot{\mathbf{v}} = \dfrac{\mathbf{f}(\mathbf{x}, \mathbf{v}, t)}{m} \end{bmatrix}$

where we have added a new variable **v** to get a pair of **coupled first order equations**.

Phase space

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$$

Concatenate **x** and **v** to make a 6-vector: position in **phase space**.

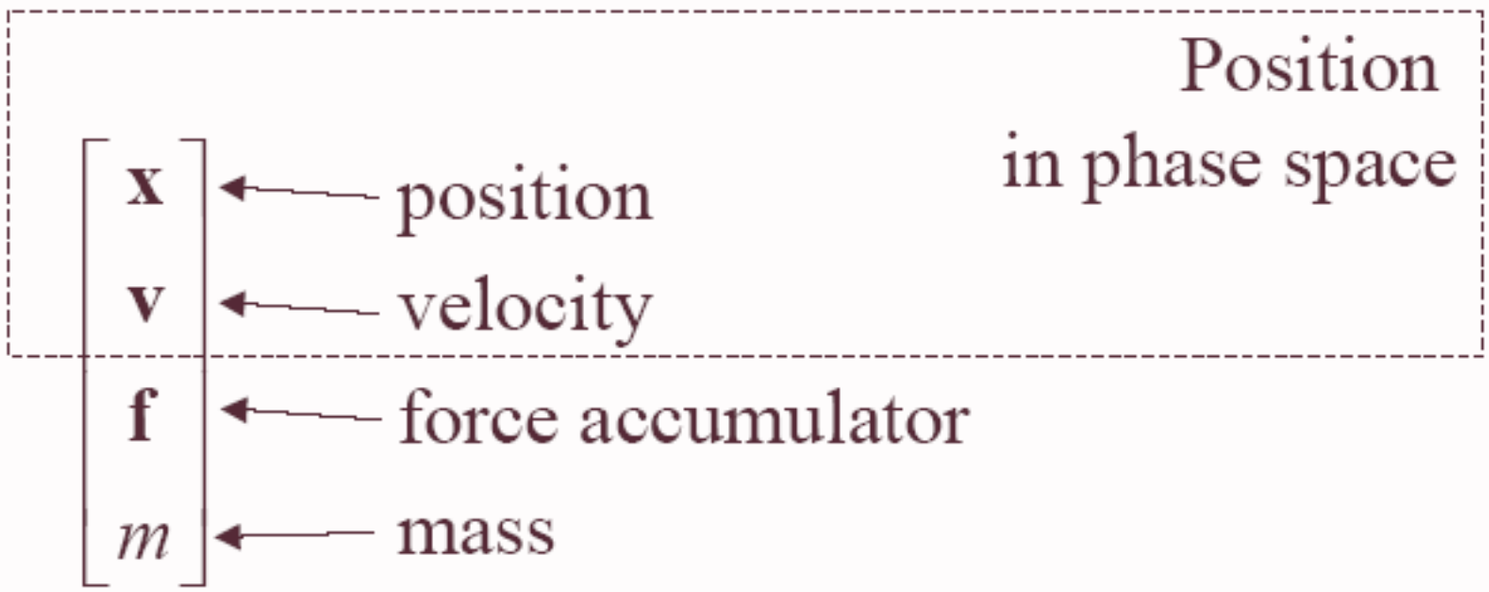
$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix}$$

Taking the time derivative: another 6-vector.

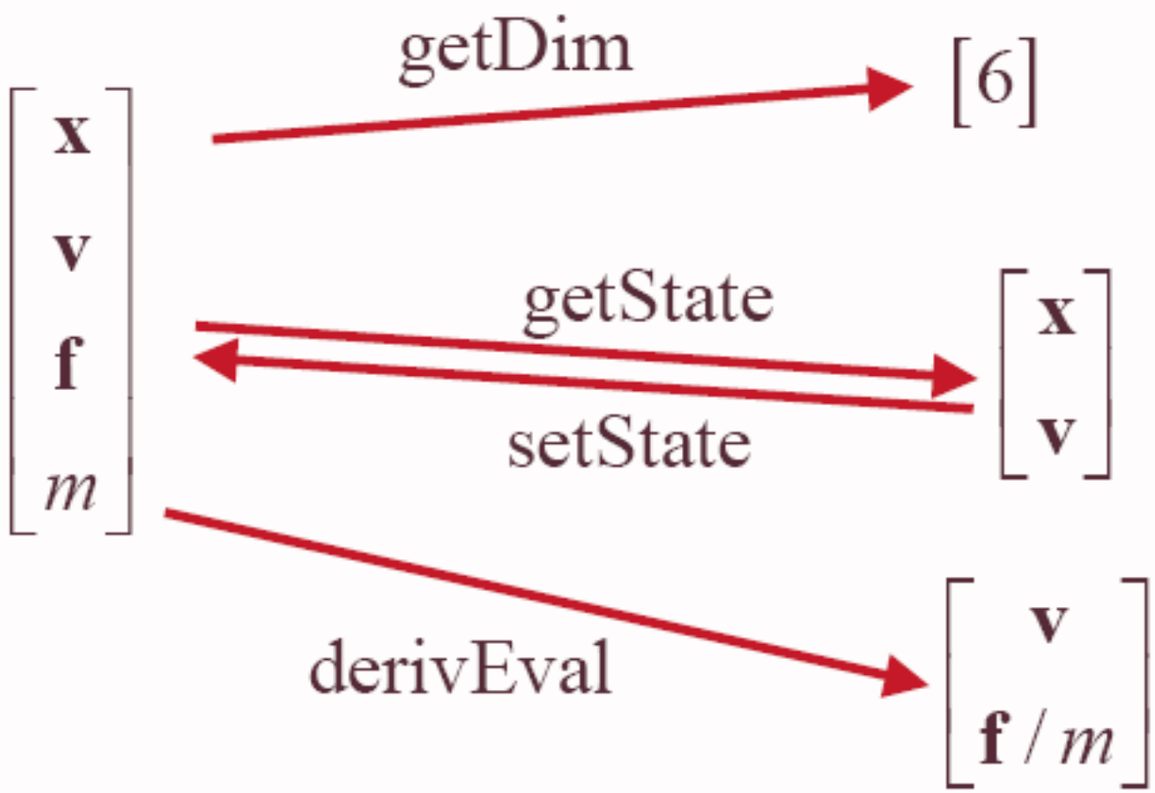
$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f} / m \end{bmatrix}$$

A vanilla 1st-order differential equation.

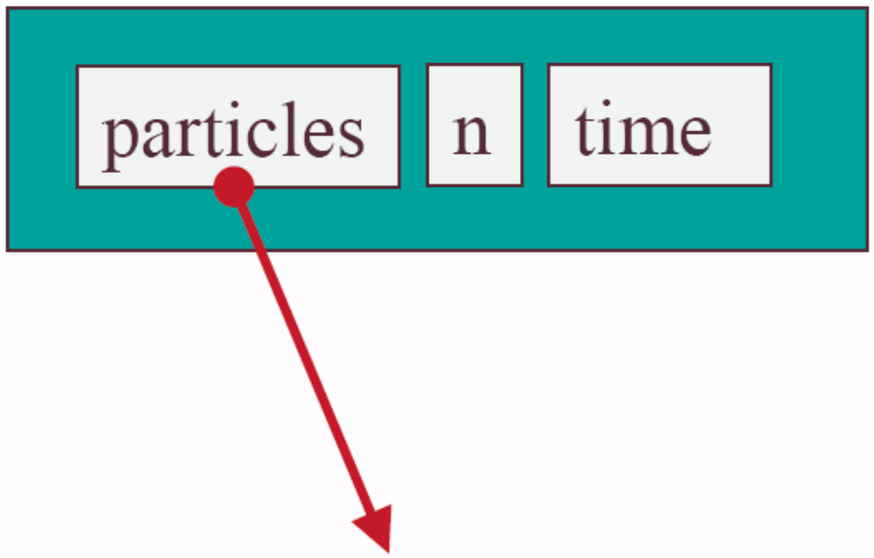
Particle structure



Solver interface

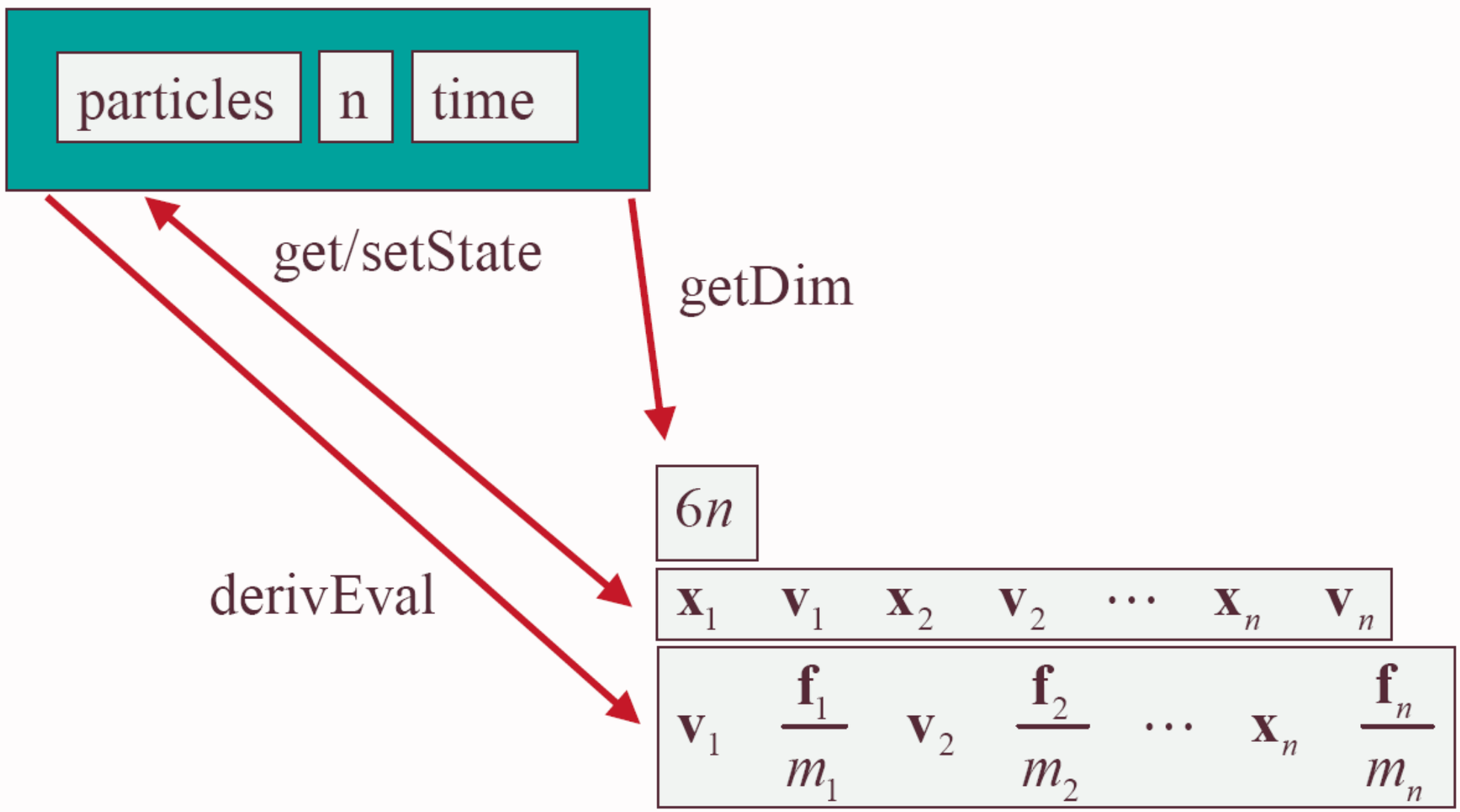


Particle systems



$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{v}_1 \\ \mathbf{f}_1 \\ m_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{v}_2 \\ \mathbf{f}_2 \\ m_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_3 \\ \mathbf{v}_3 \\ \mathbf{f}_3 \\ m_3 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{x}_n \\ \mathbf{v}_n \\ \mathbf{f}_n \\ m_n \end{bmatrix}$$

Solver interface



Forces

- Constant (gravity)
- Position/time dependent (force fields)
- Velocity-dependent (drag)
- N-ary (springs)

Gravity

Force law:

$$\mathbf{f}_{grav} = m\mathbf{G}$$

```
p->f += p->m * F->G
```

Viscous drag

Force law:

$$\mathbf{f}_{drag} = -k_{drag}\mathbf{v}$$

```
p->f -= F->k * p->v
```

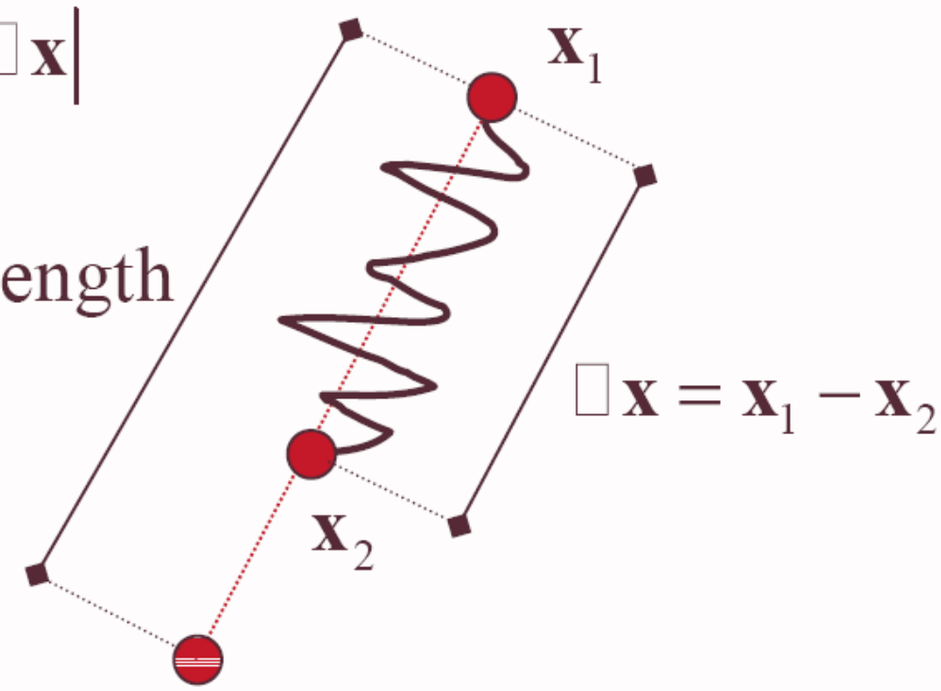
Damped spring

Force law:

$$\mathbf{f}_1 = -\left[k_s(|\Delta\mathbf{x}| - \mathbf{r}) + k_d \left(\frac{\Delta\mathbf{v}\Delta\mathbf{x}}{|\Delta\mathbf{x}|} \right) \right] \frac{\Delta\mathbf{x}}{|\Delta\mathbf{x}|}$$

$$\mathbf{f}_2 = -\mathbf{f}_1$$

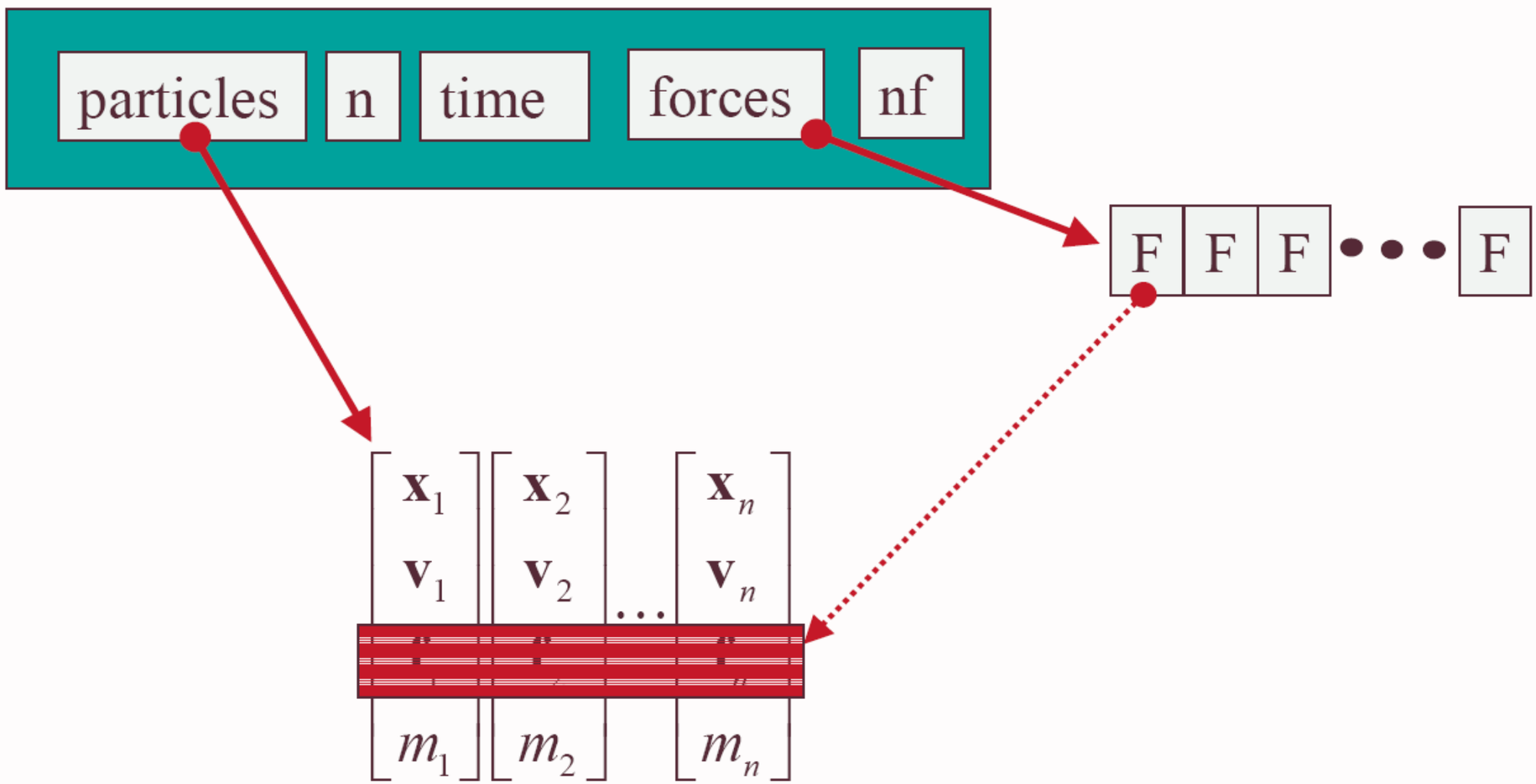
$\mathbf{r}$ = rest length



$$\Delta\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$$

$$\Delta\mathbf{v} = \mathbf{v}_1 - \mathbf{v}_2$$

Particle systems with forces



derivEval loop

1. Clear forces
   - Loop over particles, zero force accumulators
2. Calculate forces
   - Sum all forces into accumulators
3. Gather
   - Loop over particles, copying v and f/m into destination array

derivEval Loop

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{v}_1 \\ \mathbf{f}_1 \\ m_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{v}_2 \\ \mathbf{f}_2 \\ m_2 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{x}_n \\ \mathbf{v}_n \\ \mathbf{f}_n \\ m_n \end{bmatrix}$$

1

Clear force accumulators

2

F	F	F	• • •	F

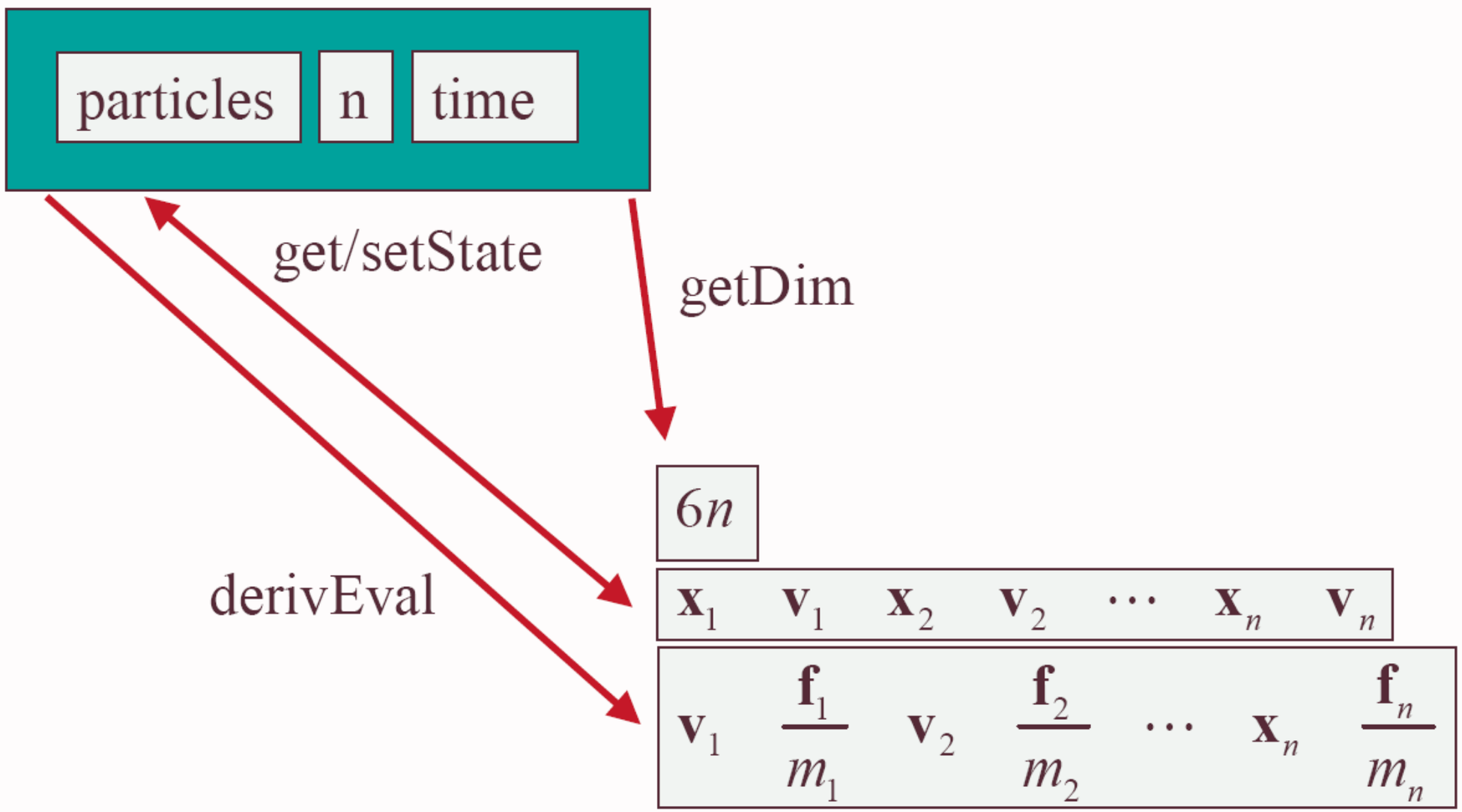
Apply forces to particles

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{v}_1 \\ \mathbf{f}_1 \\ m_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{v}_2 \\ \mathbf{f}_2 \\ m_2 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{x}_n \\ \mathbf{v}_n \\ \mathbf{f}_n \\ m_n \end{bmatrix}$$

3

Return [v,f/m,…] to solver

Solver interface



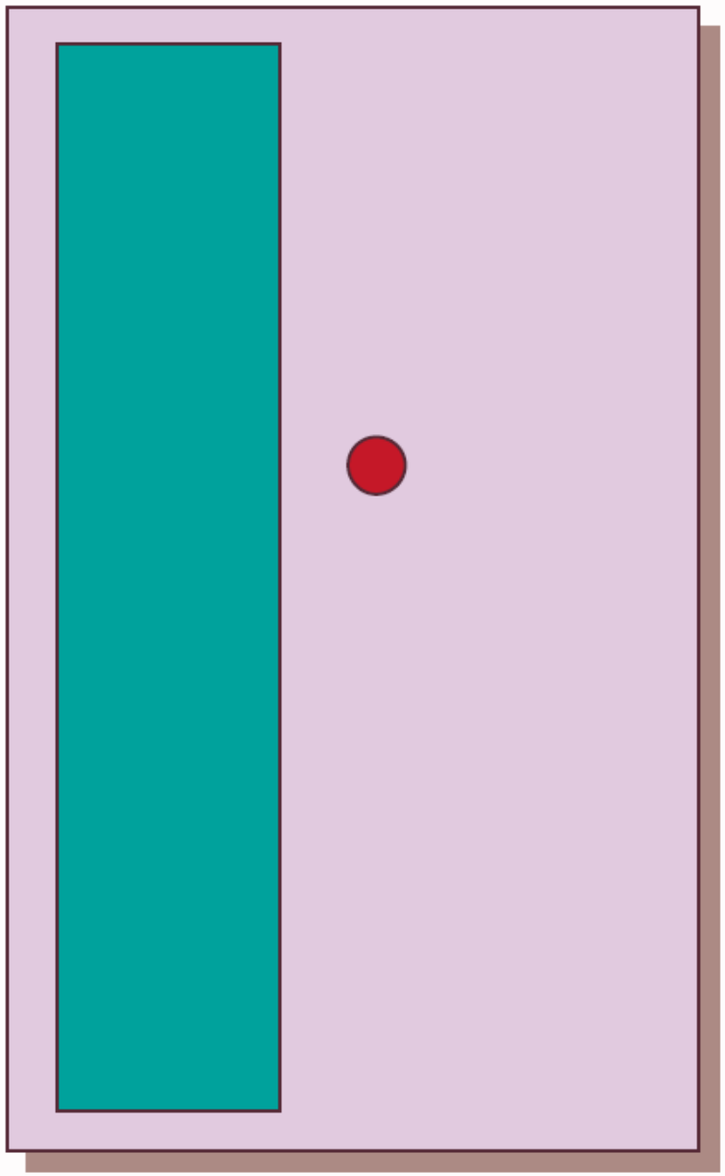
Differential equation solver

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f}/m \end{bmatrix}$$

Euler method:

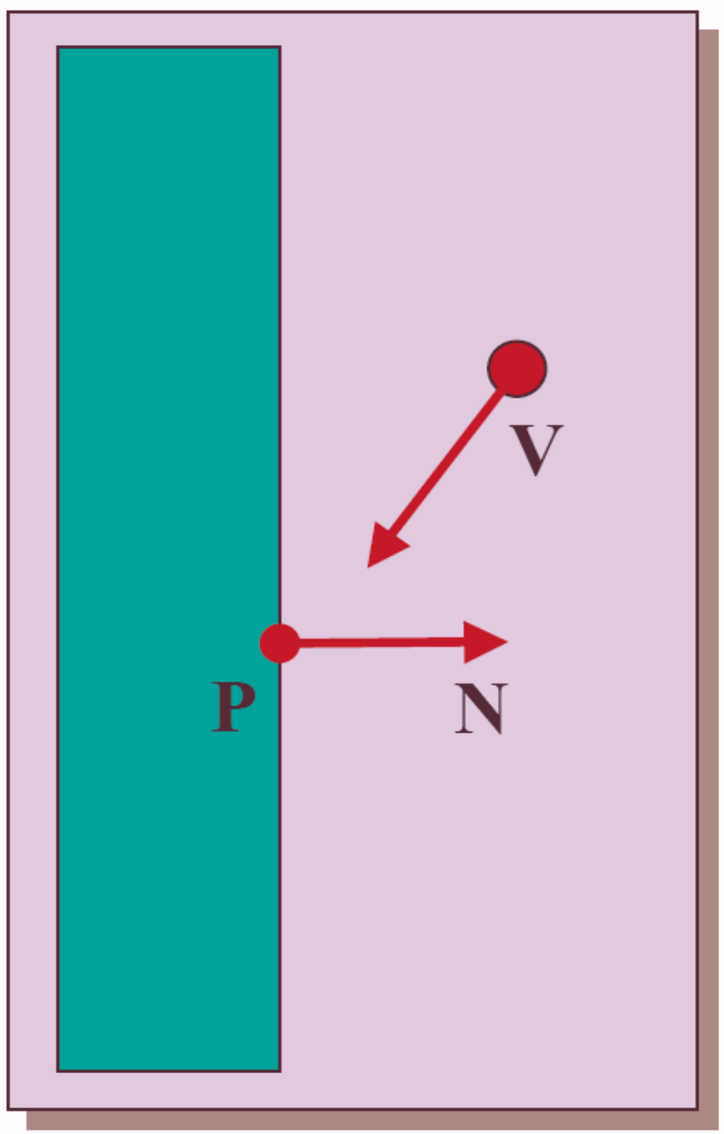
$$\begin{bmatrix} \mathbf{x}_1^{i+1} \\ \mathbf{v}_1^{i+1} \\ \vdots \\ \mathbf{x}_n^{i+1} \\ \mathbf{v}_n^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{i} \\ \mathbf{v}_1^{i} \\ \vdots \\ \mathbf{x}_n^{i} \\ \mathbf{v}_n^{i} \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{v}_1^{i} \\ \mathbf{f}_1^{i}/m_1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

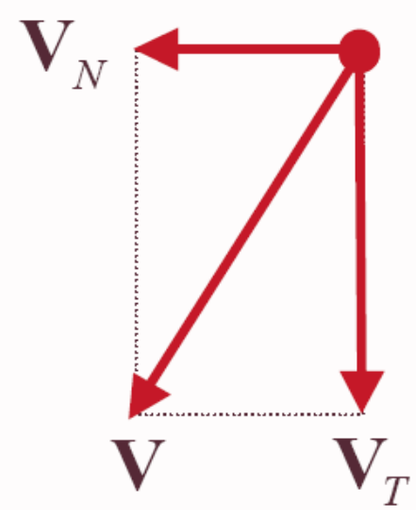
Bouncing off the walls



- Add-on for a particle simulator
- For now, just simple point-plane collisions

Normal and tangential components

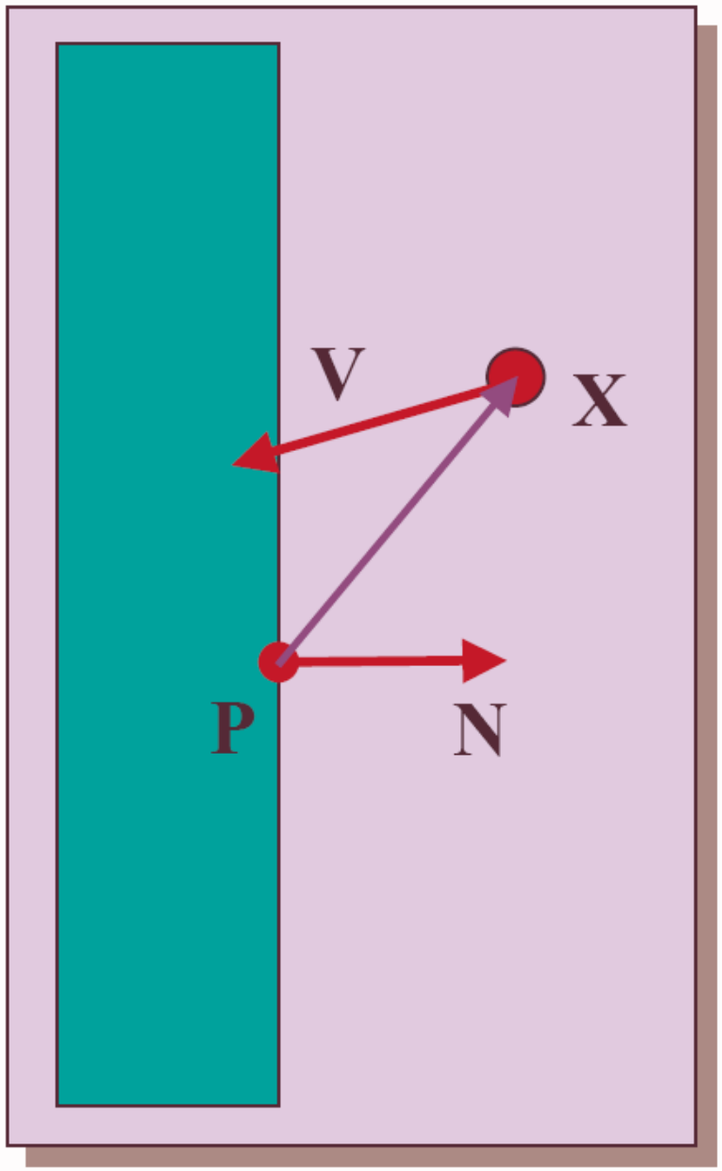




$$\mathbf{V}_N = (\mathbf{N} \cdot \mathbf{V})\mathbf{N}$$

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

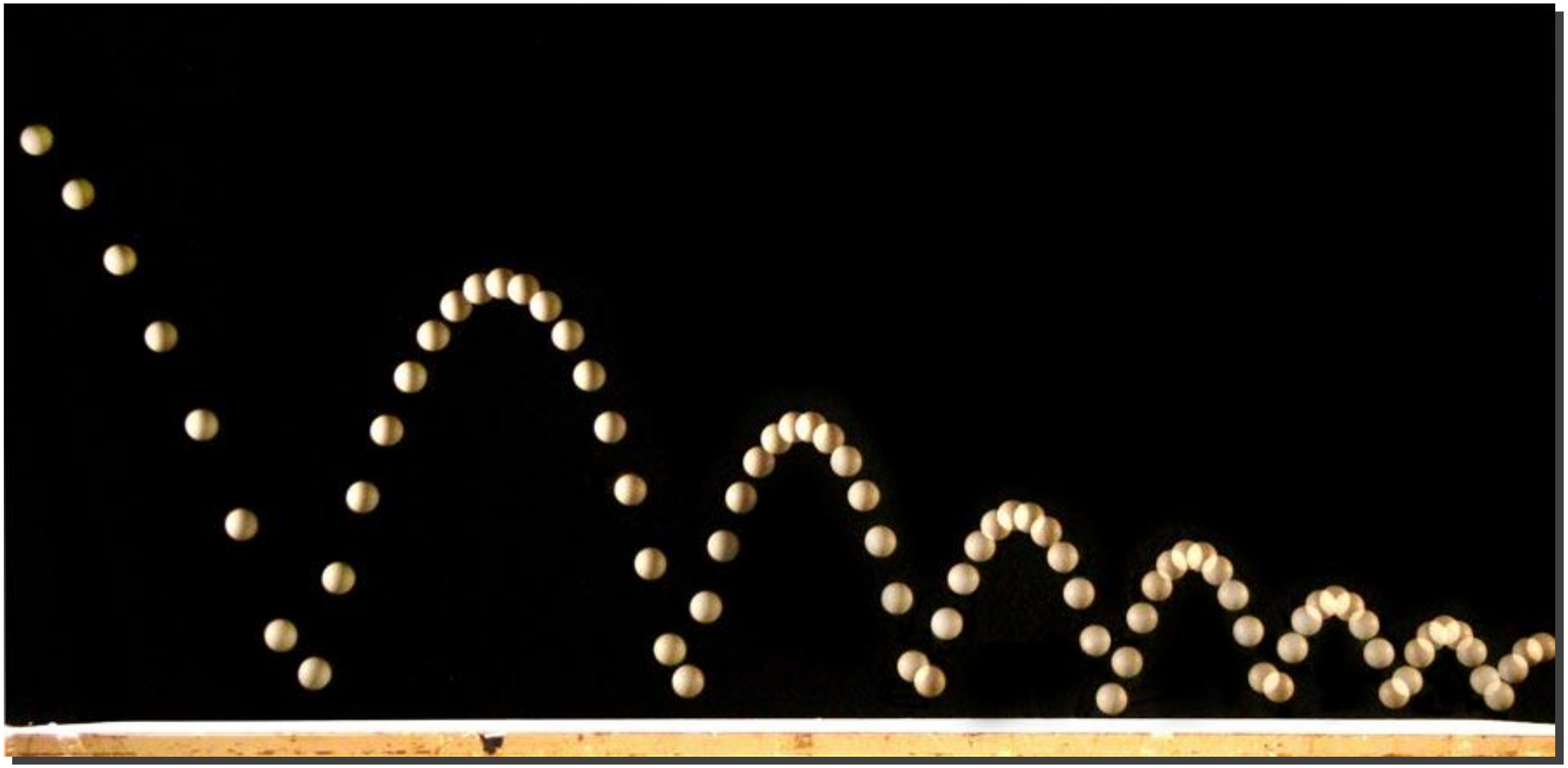
Collision Detection



$(\mathbf{X}-\mathbf{P})\cdot\mathbf{N}<\varepsilon$ Within ε of the wall

$\mathbf{N}\cdot\mathbf{V}<0$ Heading in

Collision Response



$$\mathbf{V}' = \mathbf{V}_T - k_r \mathbf{V}_N$$

Summary

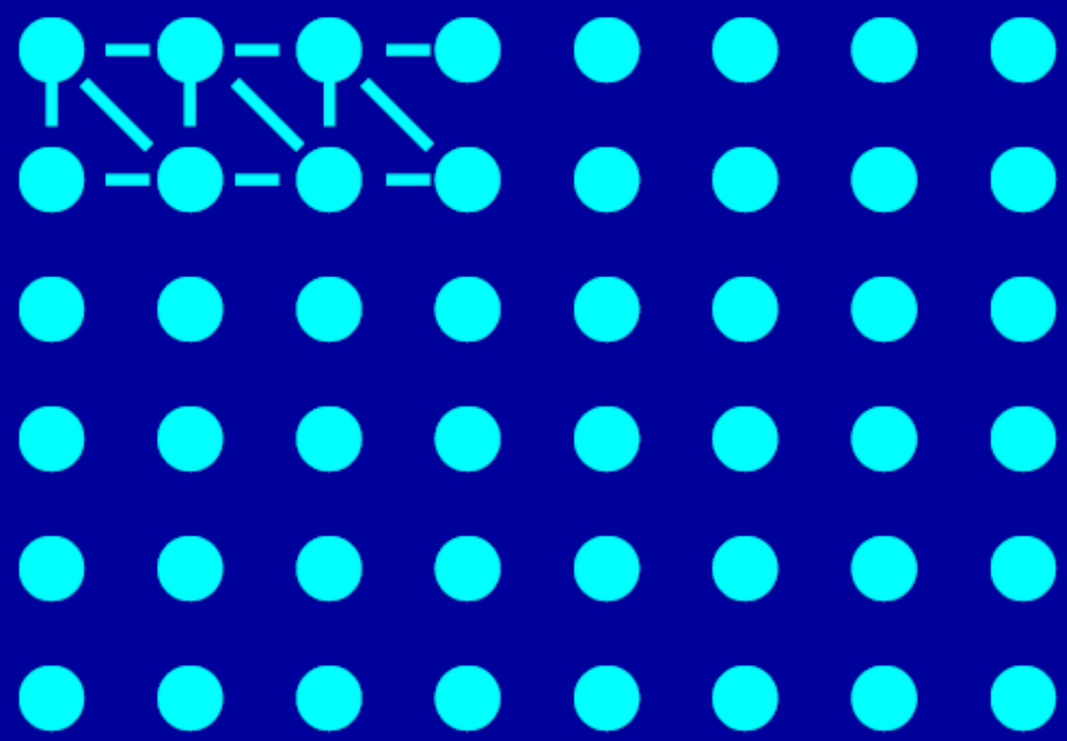
- Physics of a particle system
- Various forces acting on a particle
- Combining particles into a particle system
- Euler method for solving differential equations

Example

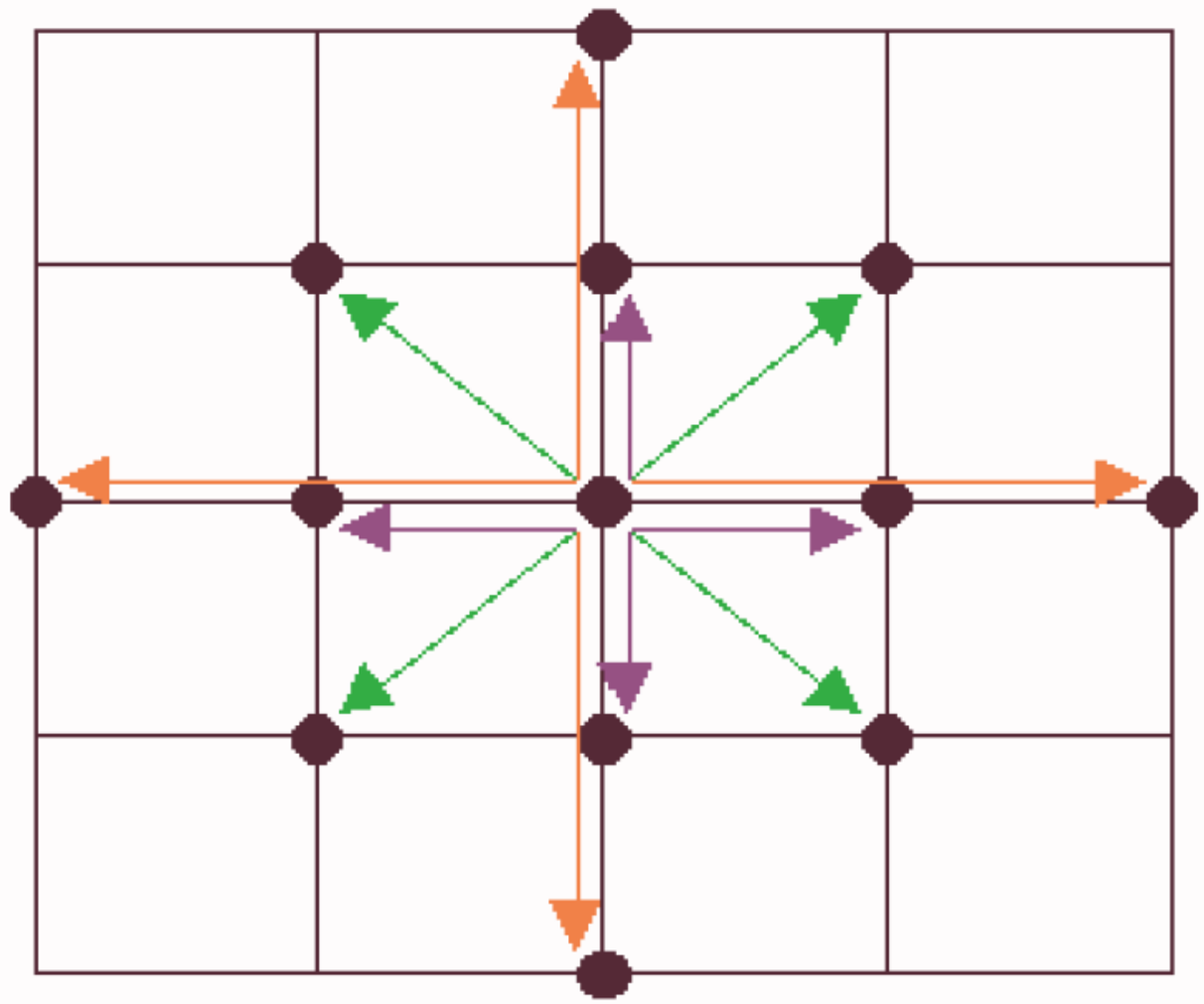
http://www.youtube.com/watch?v=3_fLO4xjTqg

Spring-Mass Systems

Cloth in 2D
Jello in 3D



Cloth Simulation



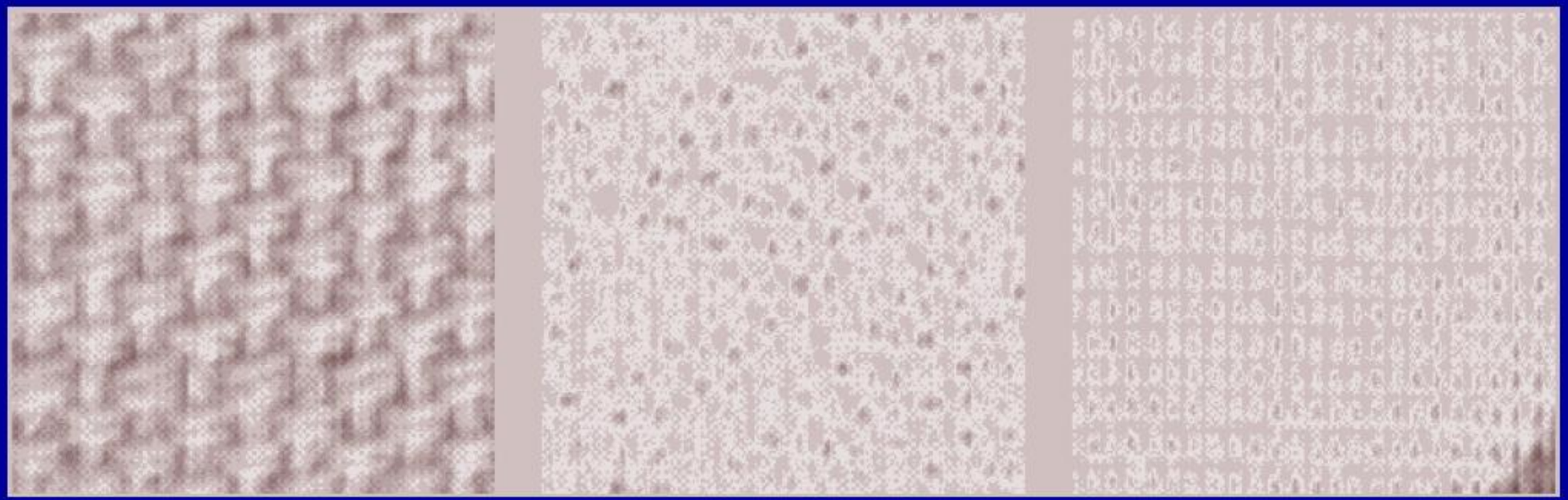
Cloth forces:

Blue (short horizontal & vertical) = stretch springs

Green (diagonal) = shear springs

Red (long horizontal & vertical) = bend springs

Cloth



Breen '95

Many types of cloth
Very different properties
Not a simple elastic surface
Woven fabrics tend to be very stiff
Anisotropic

Artificial Fish