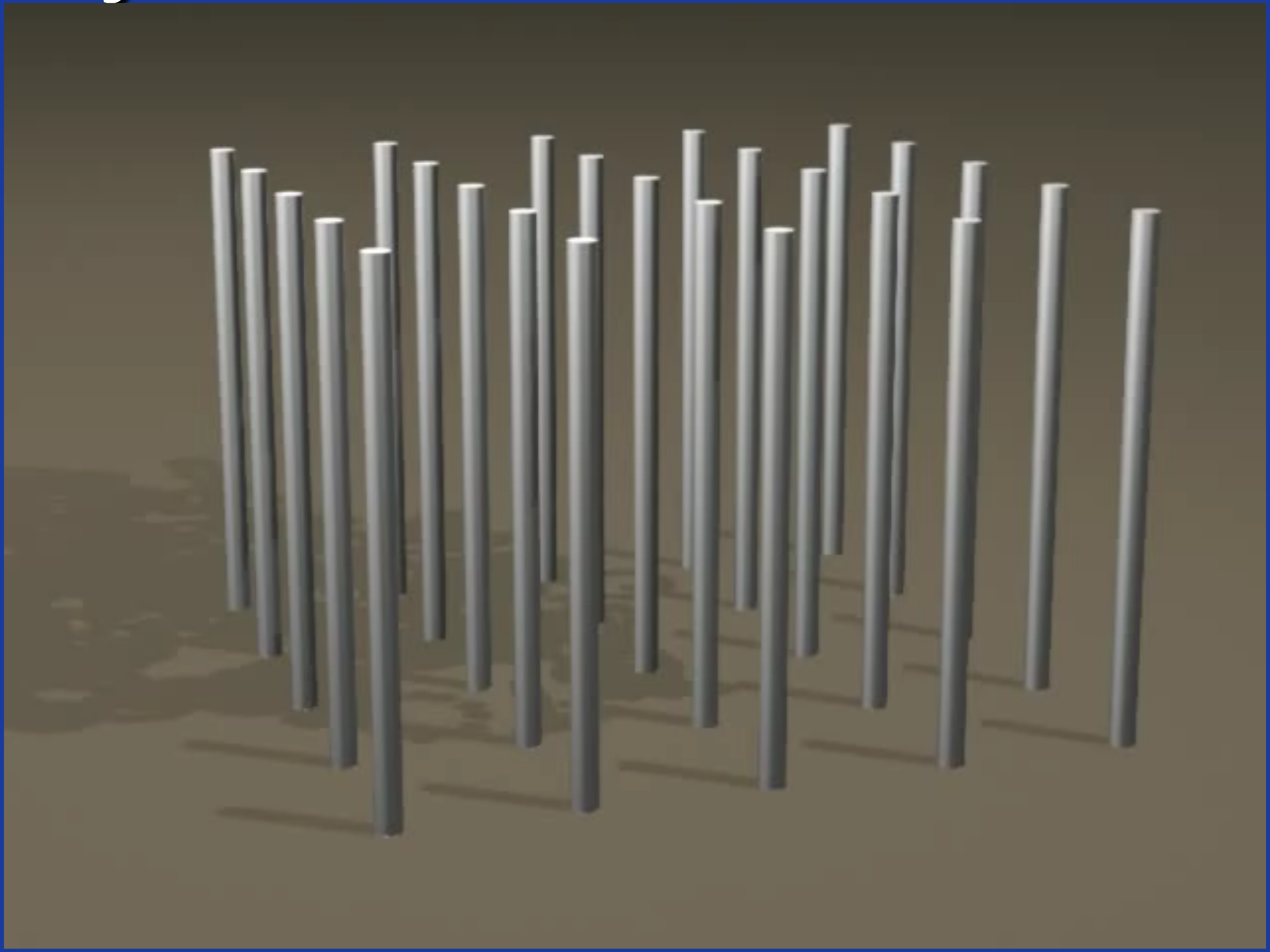
Differential Equations
&
Particle Systems

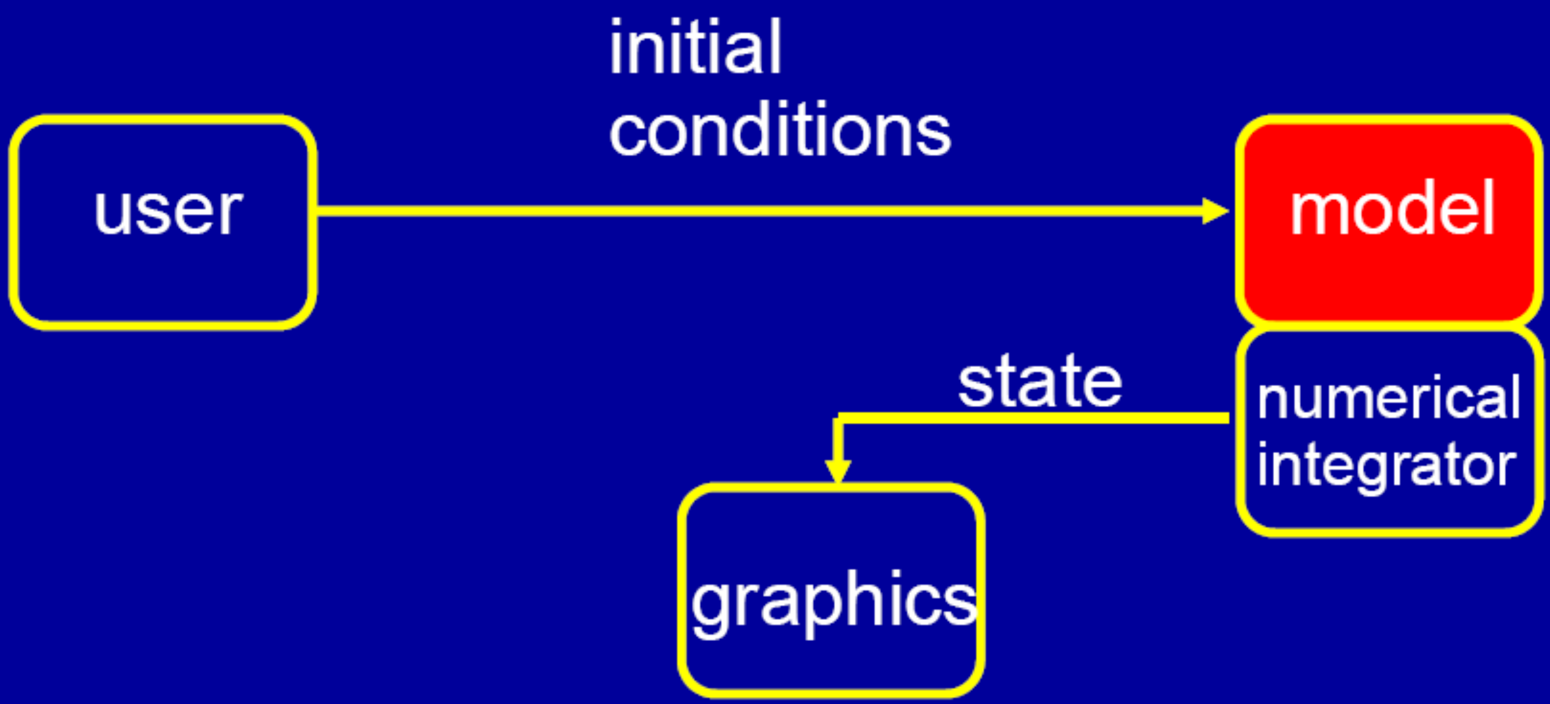
Thanks to Trueille, Popovic, Baraff, Witkin

Physics-based Animation



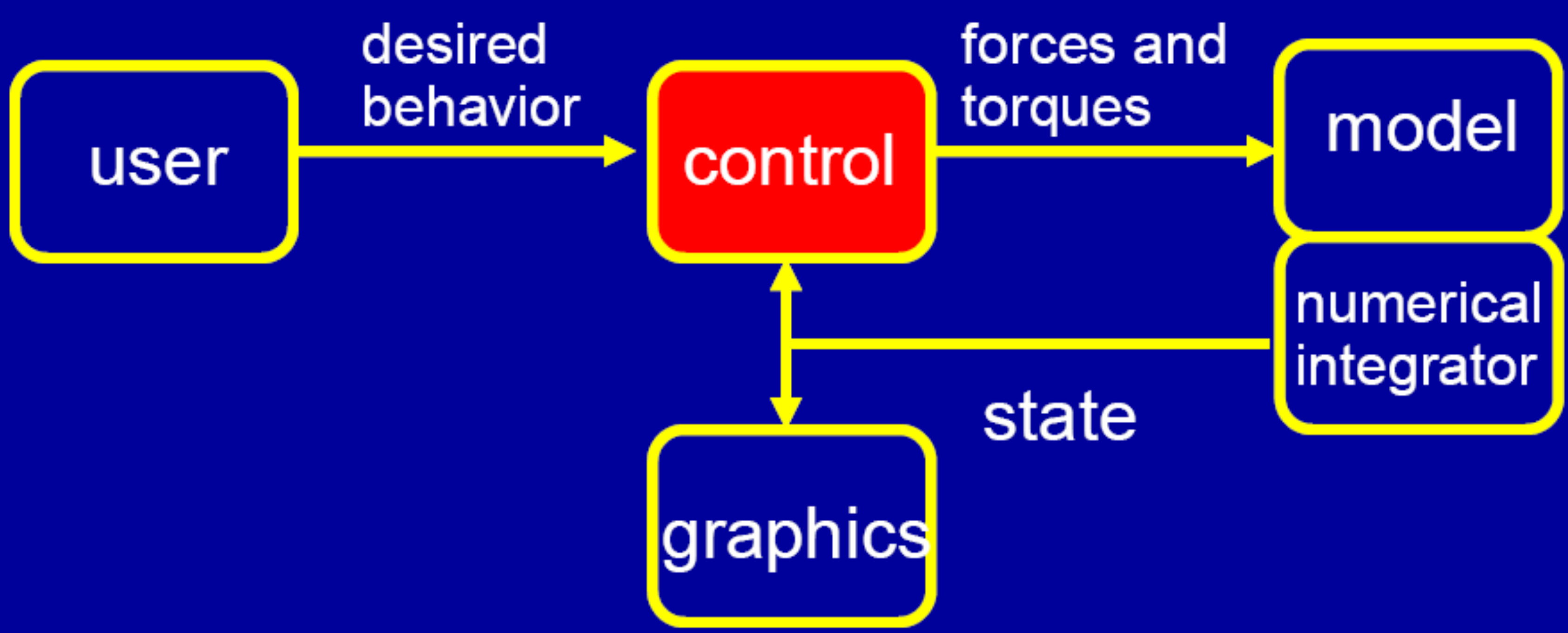
http://physbam.stanford.edu/~fedkiw/animations/large_pile.avi

Passive—no muscles or motors



Particle systems
Leaves
Water
Smoke
Clothing

Active—internal sources of energy

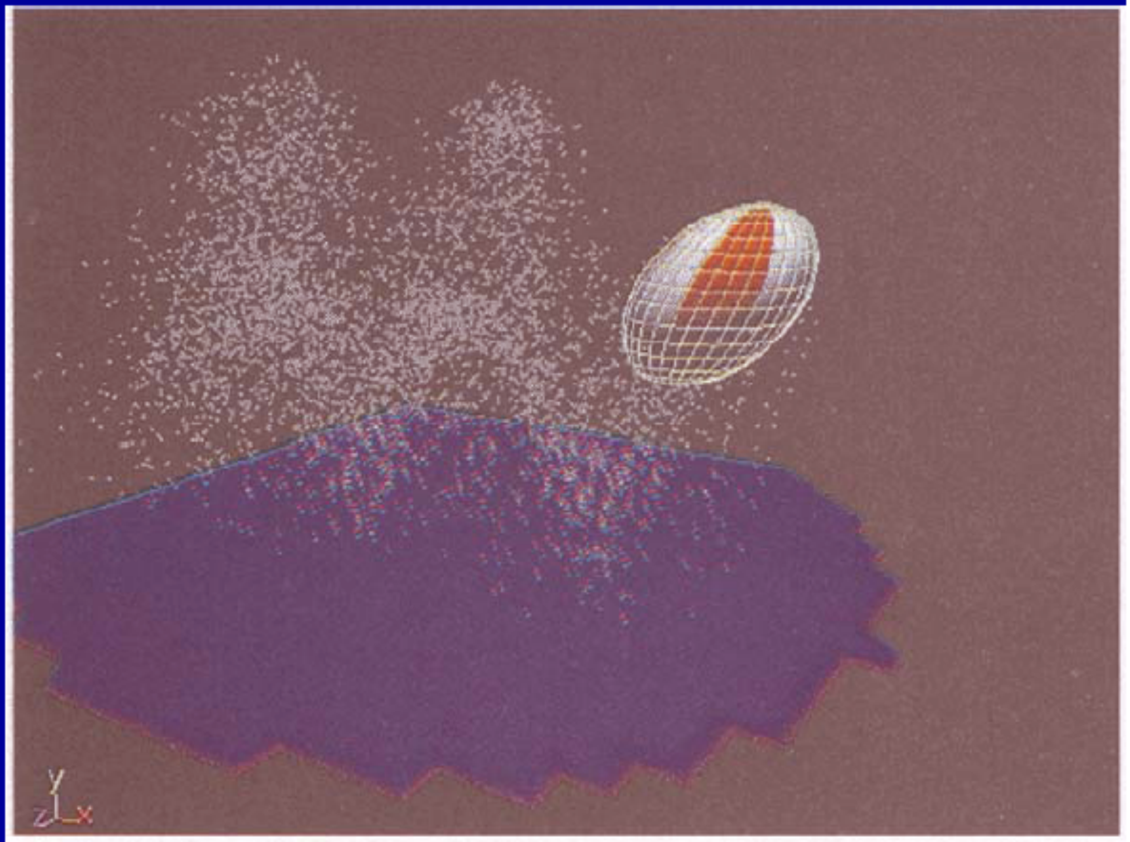


Running human
Trotting dog
Swimming fish

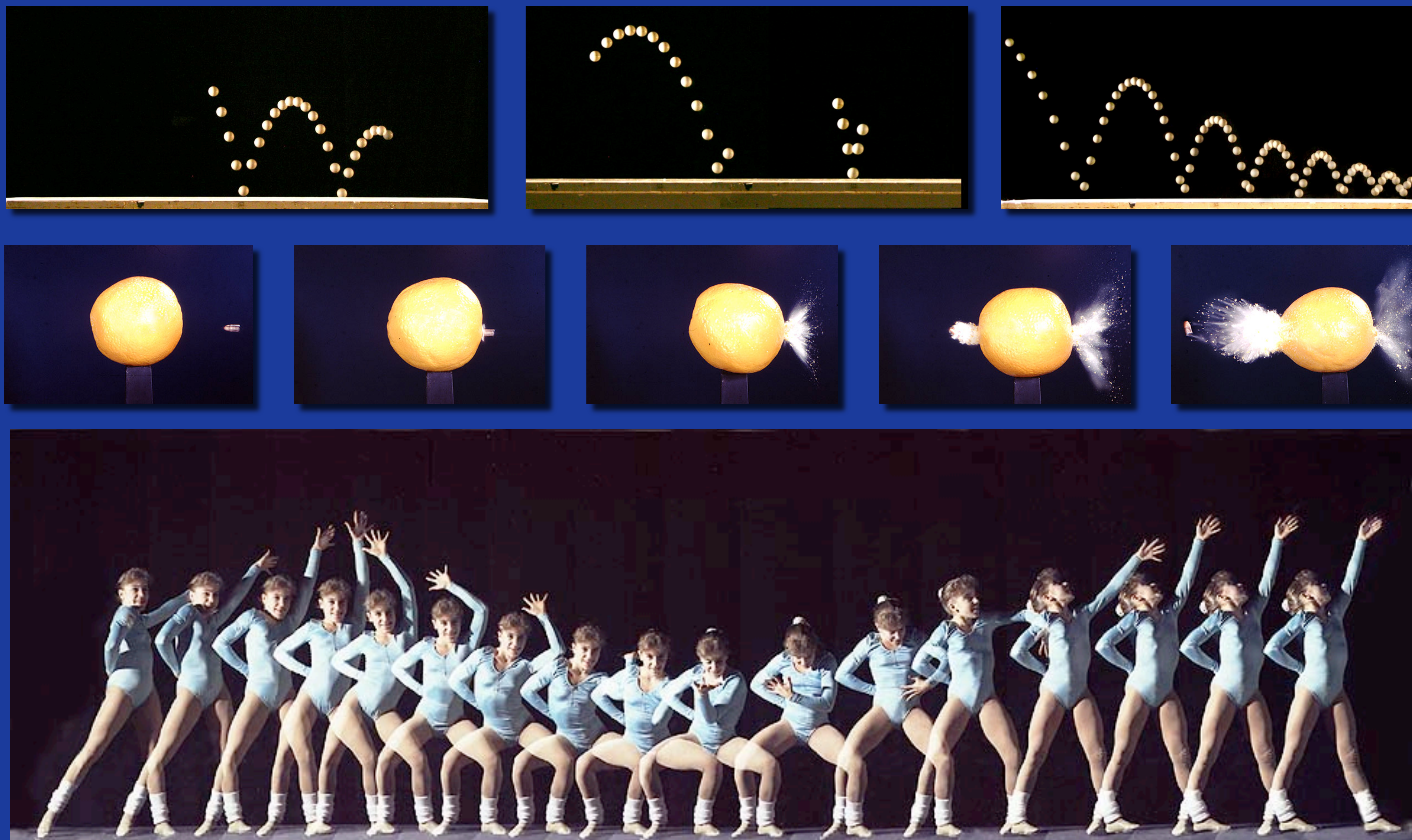
Dynamics

- 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

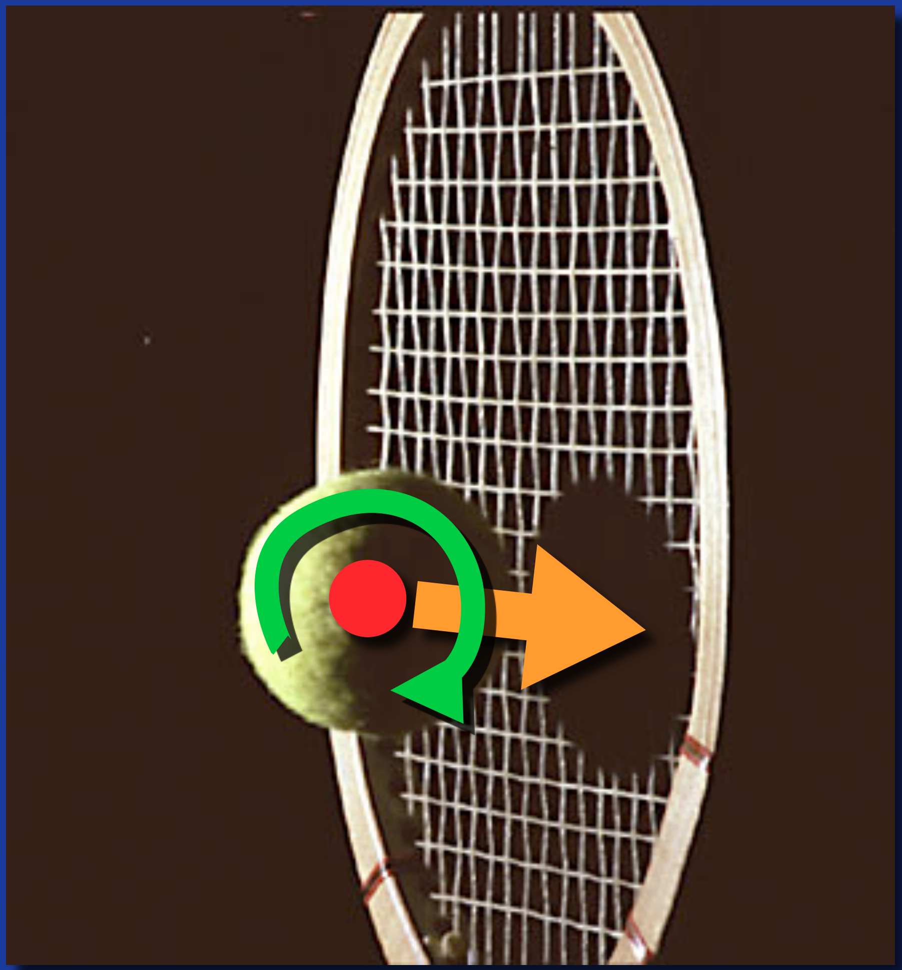


Describing Physics



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

What variables do we need?

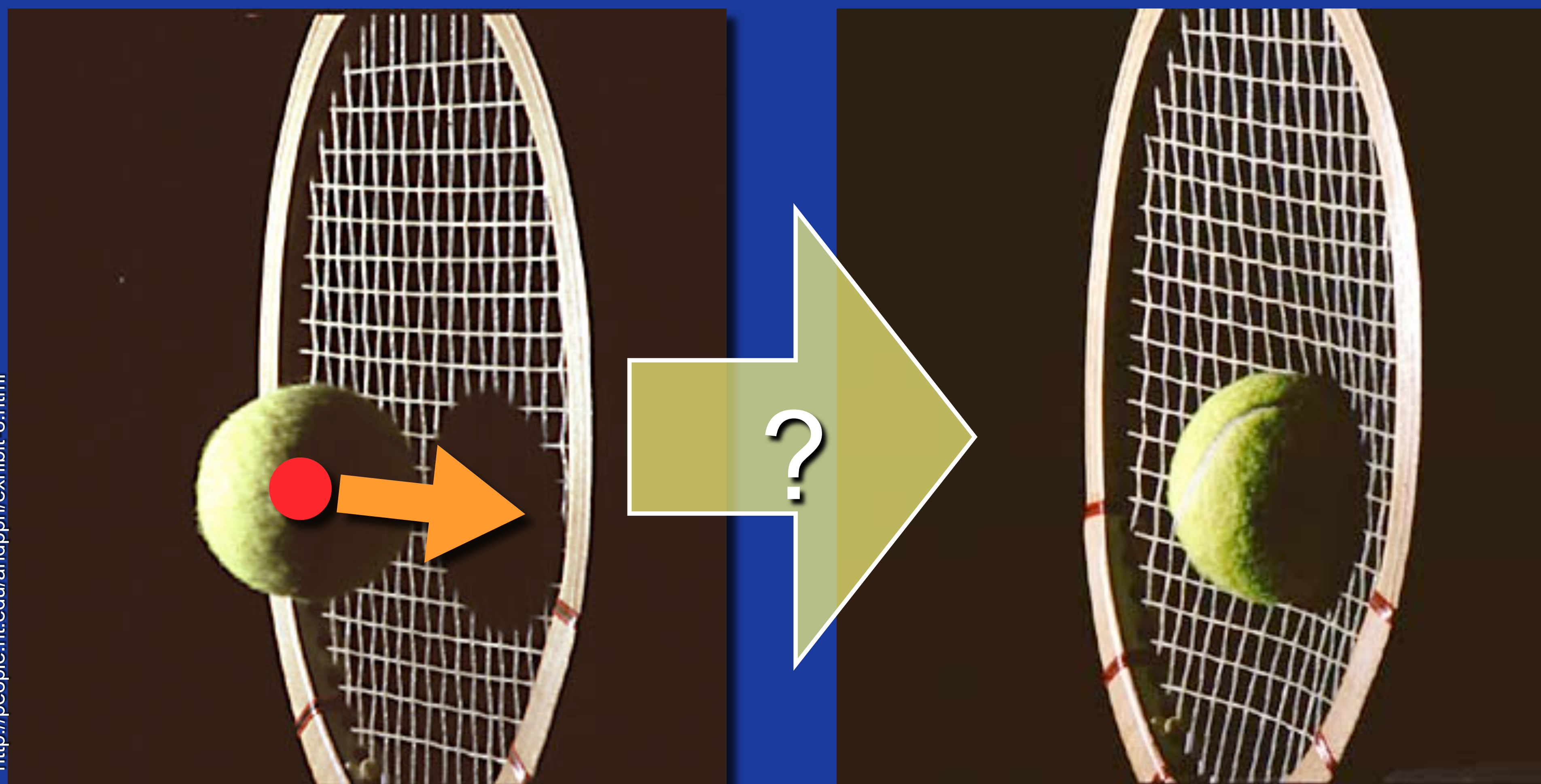


http://people.rit.edu/andpph/exhibit-8.html

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

What Happens Next?

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



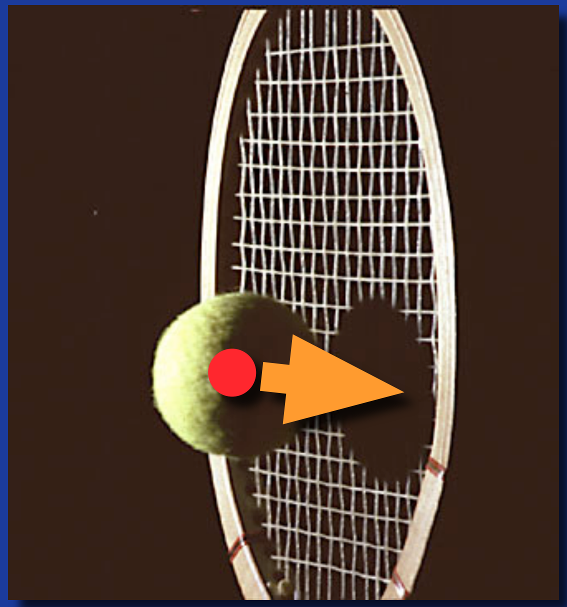
- Position
- Velocity

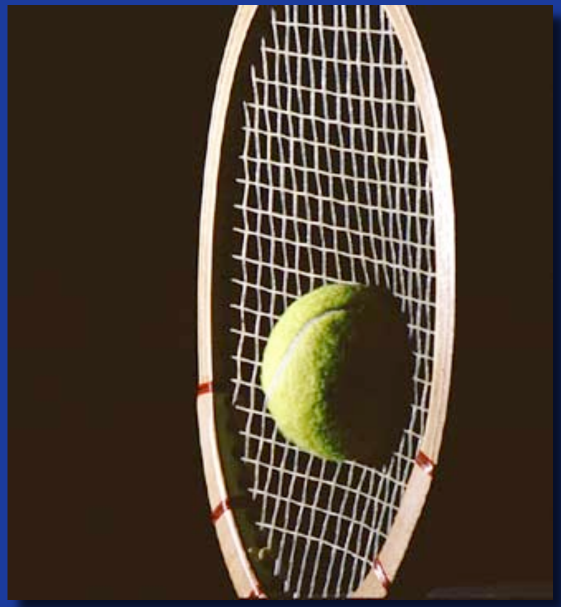
$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

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

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

Differential Equations





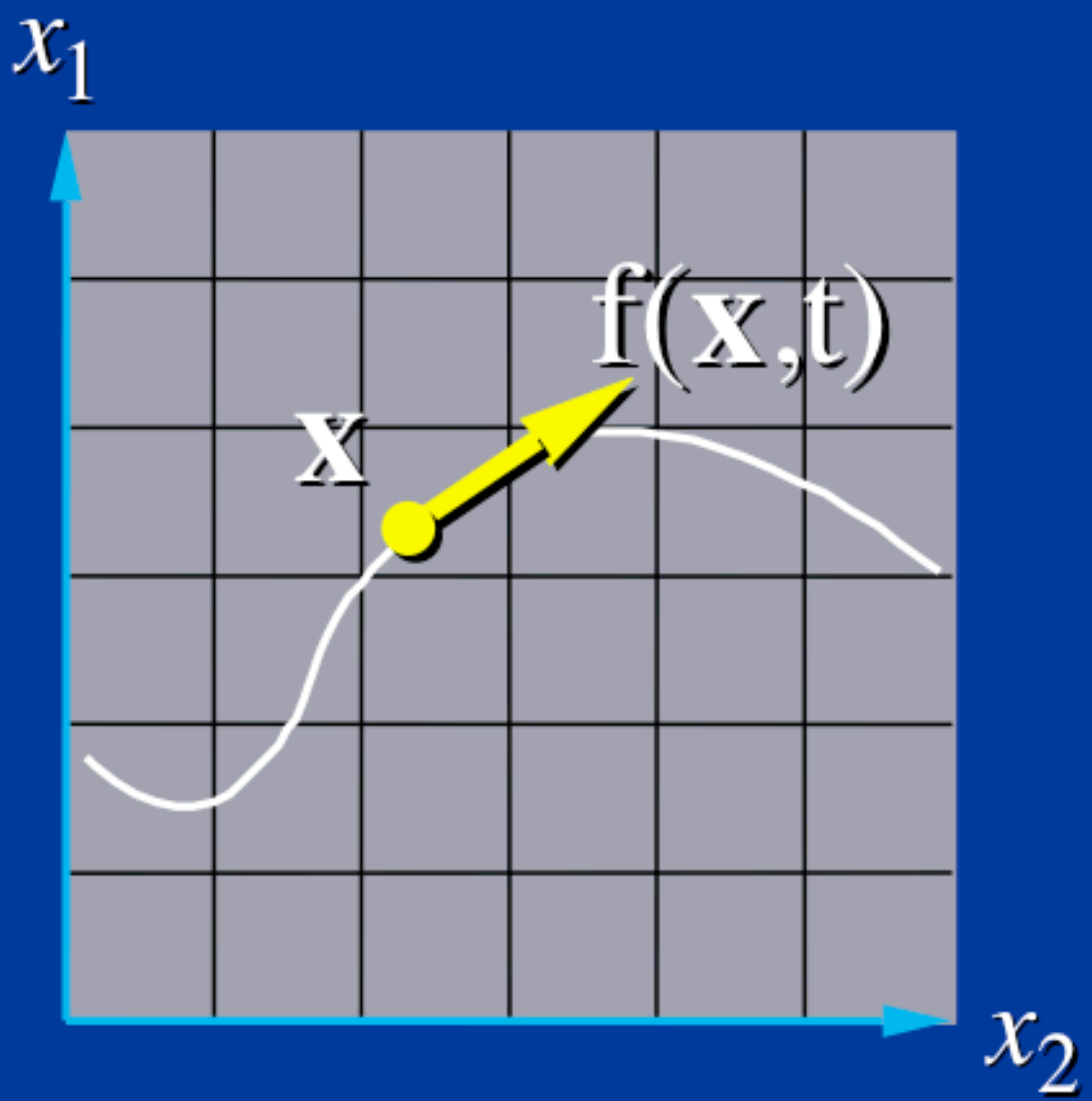
$$\dot{x} = f(x)$$

Differential Equation Basics

Andrew Witkin

PIXAR ANIMATION STUDIOS

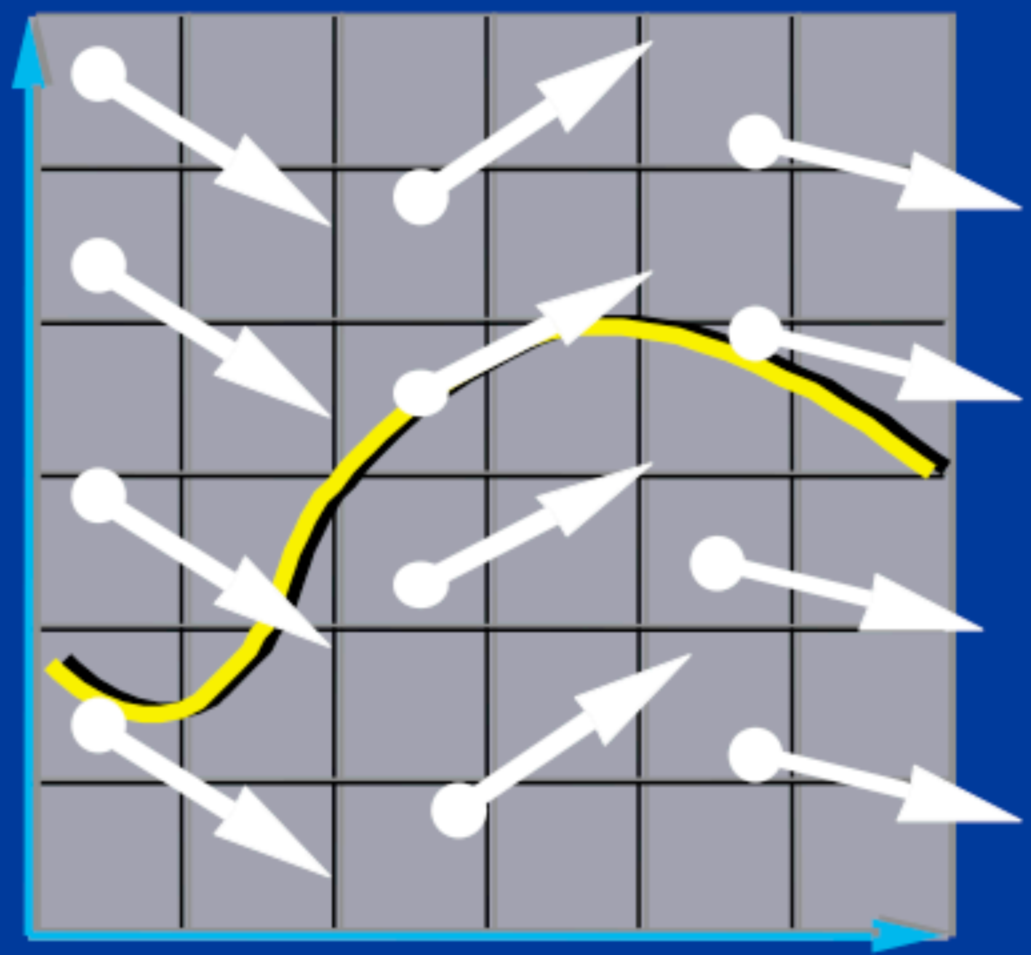
A Canonical Differential Equation



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

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

Vector Field



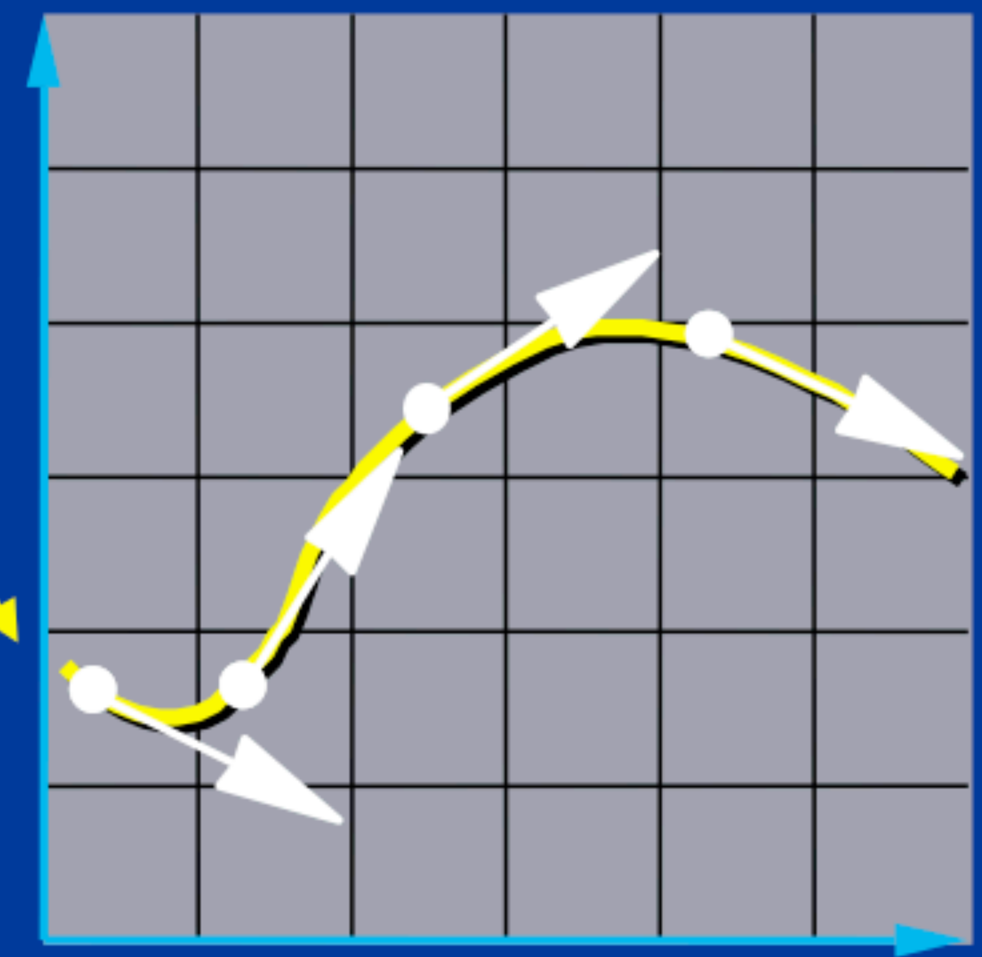
The differential equation

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

defines a vector field over x.

Integral Curves

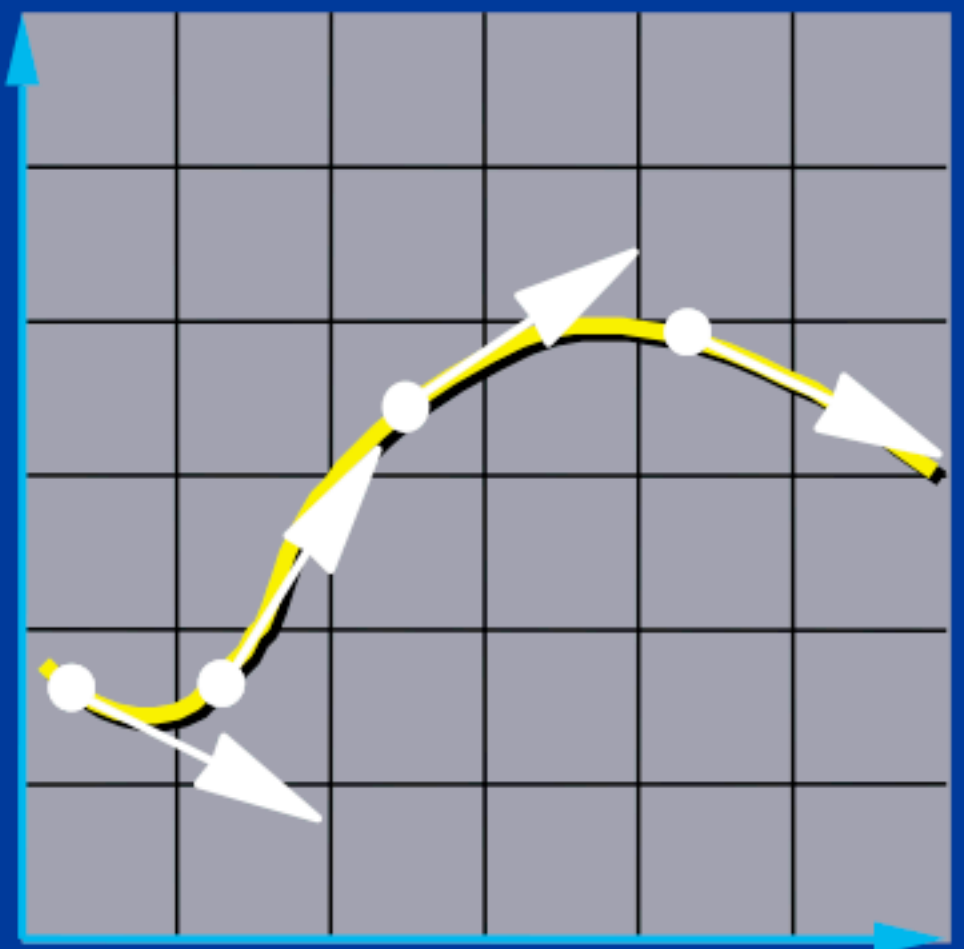
Start Here



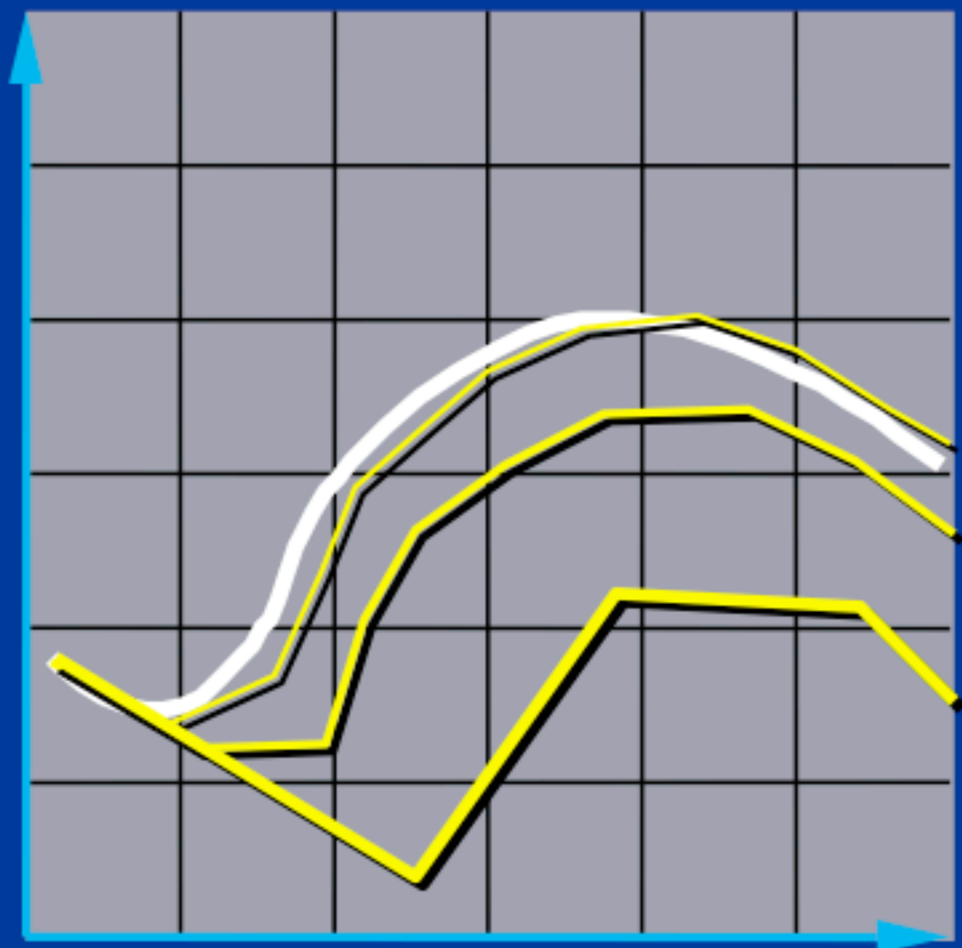
Pick any starting point, and follow the vectors.

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

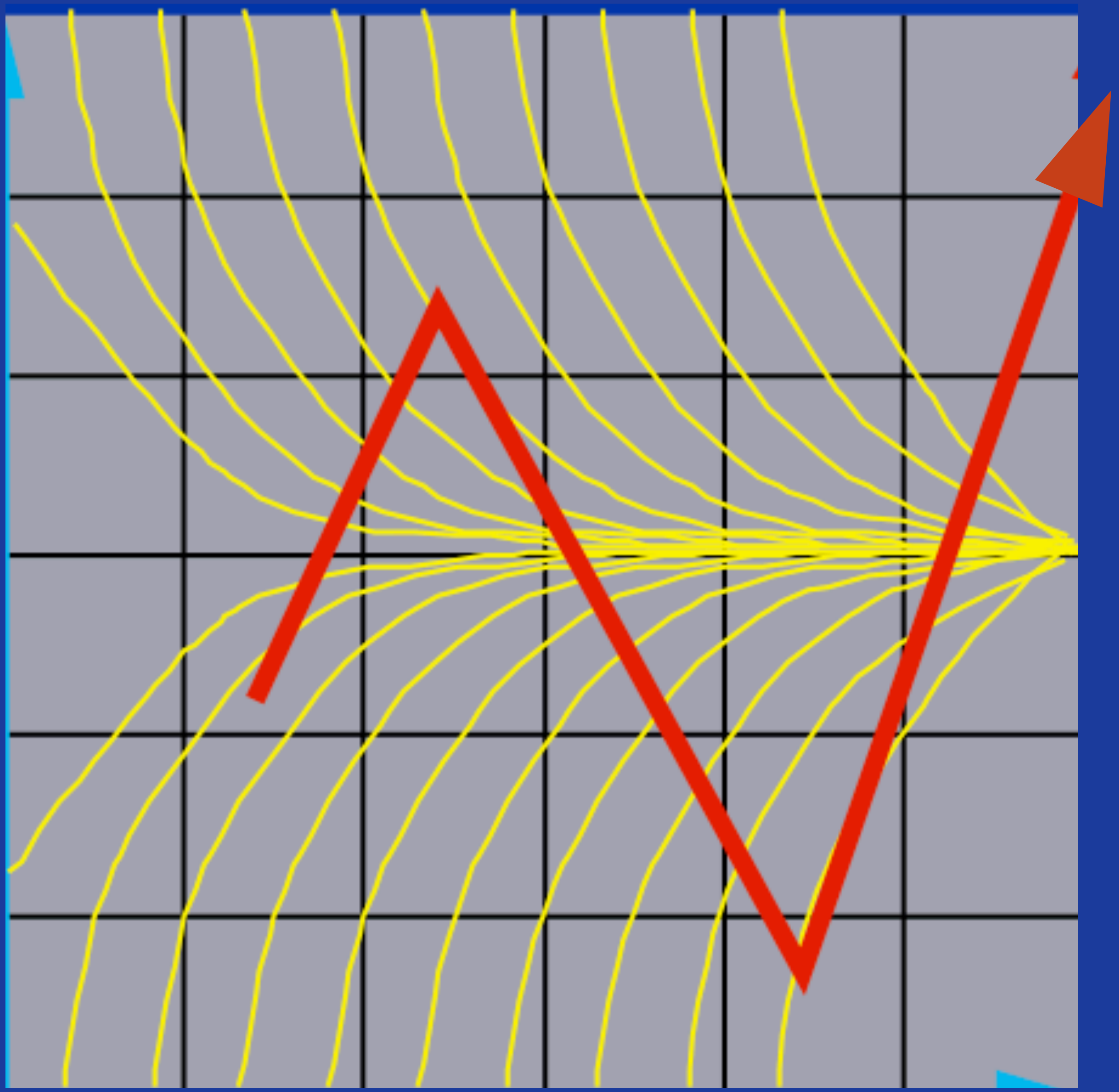
Error turns x(t) from a circle into the spiral of your choice.

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

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

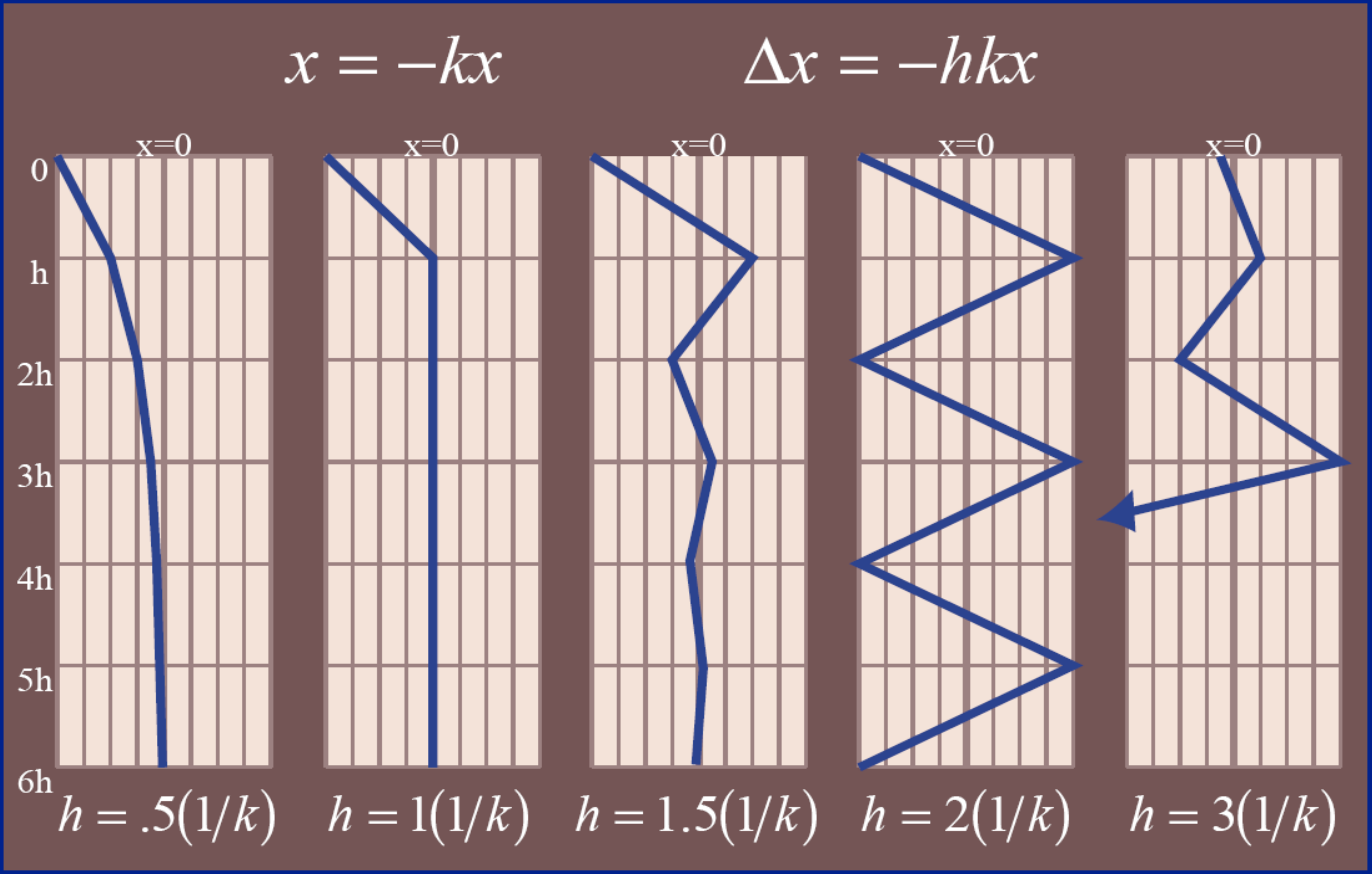
Consider Taylor Expansion about x(t)...

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

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



$h > 1/\mathrm{k}$: oscillate. $h > 2/k$: explode!

The Midpoint Method

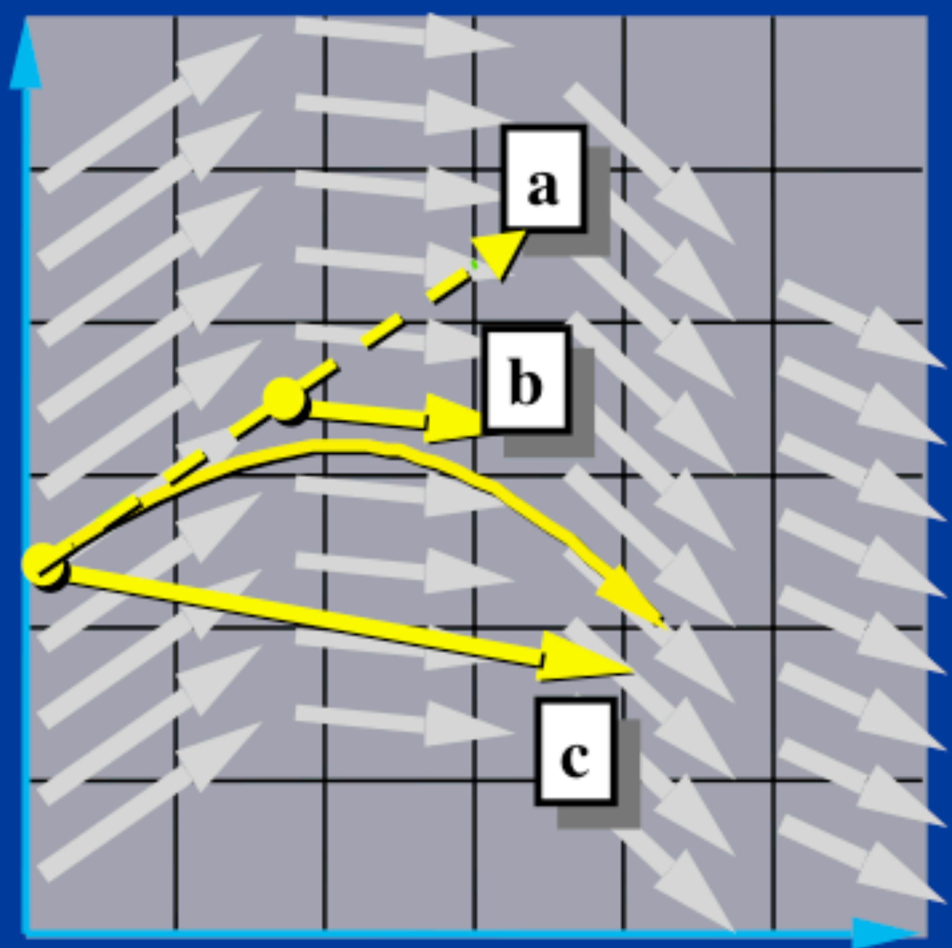
- Also known as second order Runge-Kutta:

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

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

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

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

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

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

Very popular

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

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

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

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

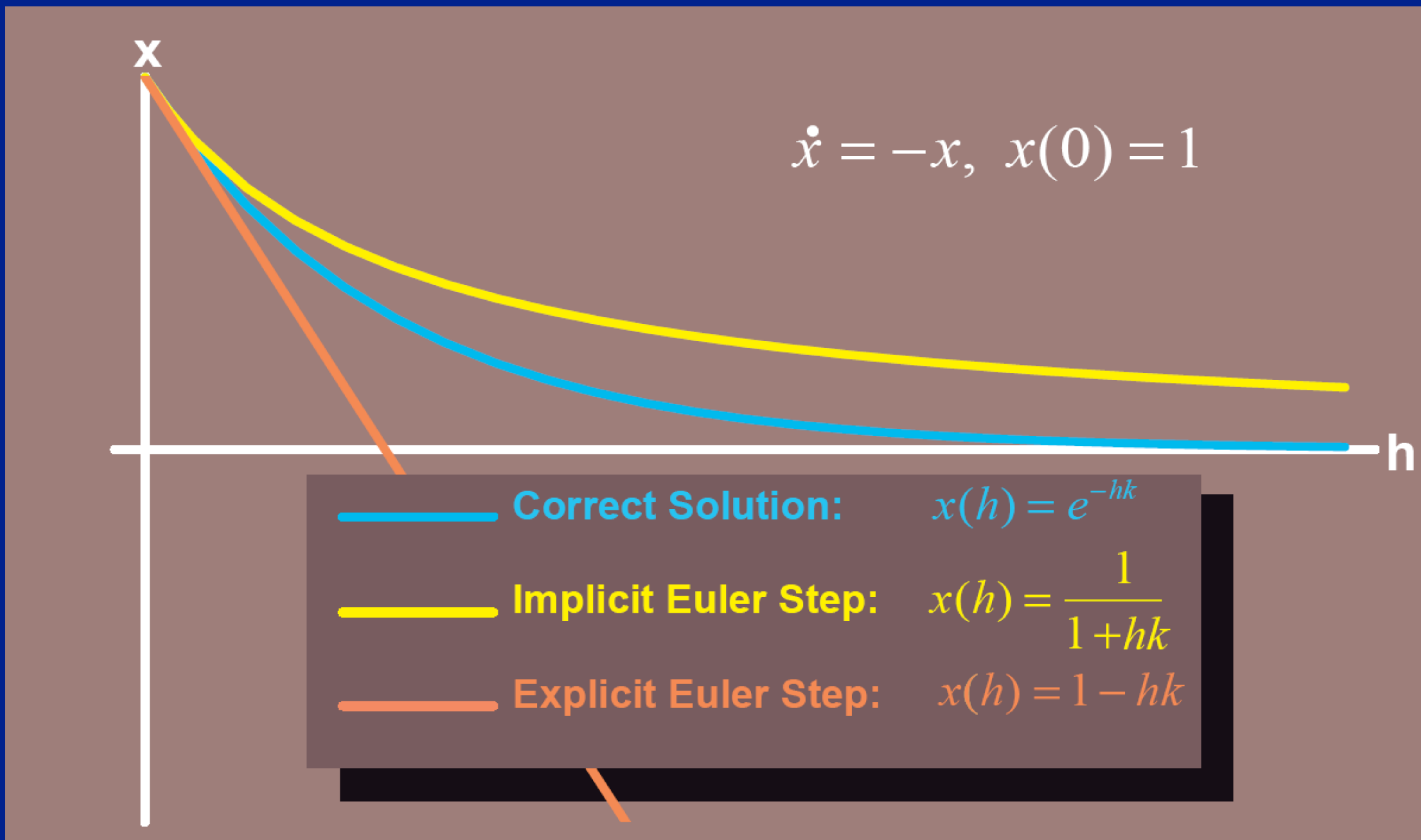
$$x(t_0 + h) = x(t_0) + h\dot{x}(t_0)$$

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

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

$$
\begin{aligned}
x(t+h) &= x(t) + h\,\dot{x}(t+h) \\
&= x(t) - hkx(t+h) \\
&= \frac{x(t)}{1+hk}
\end{aligned}
$$

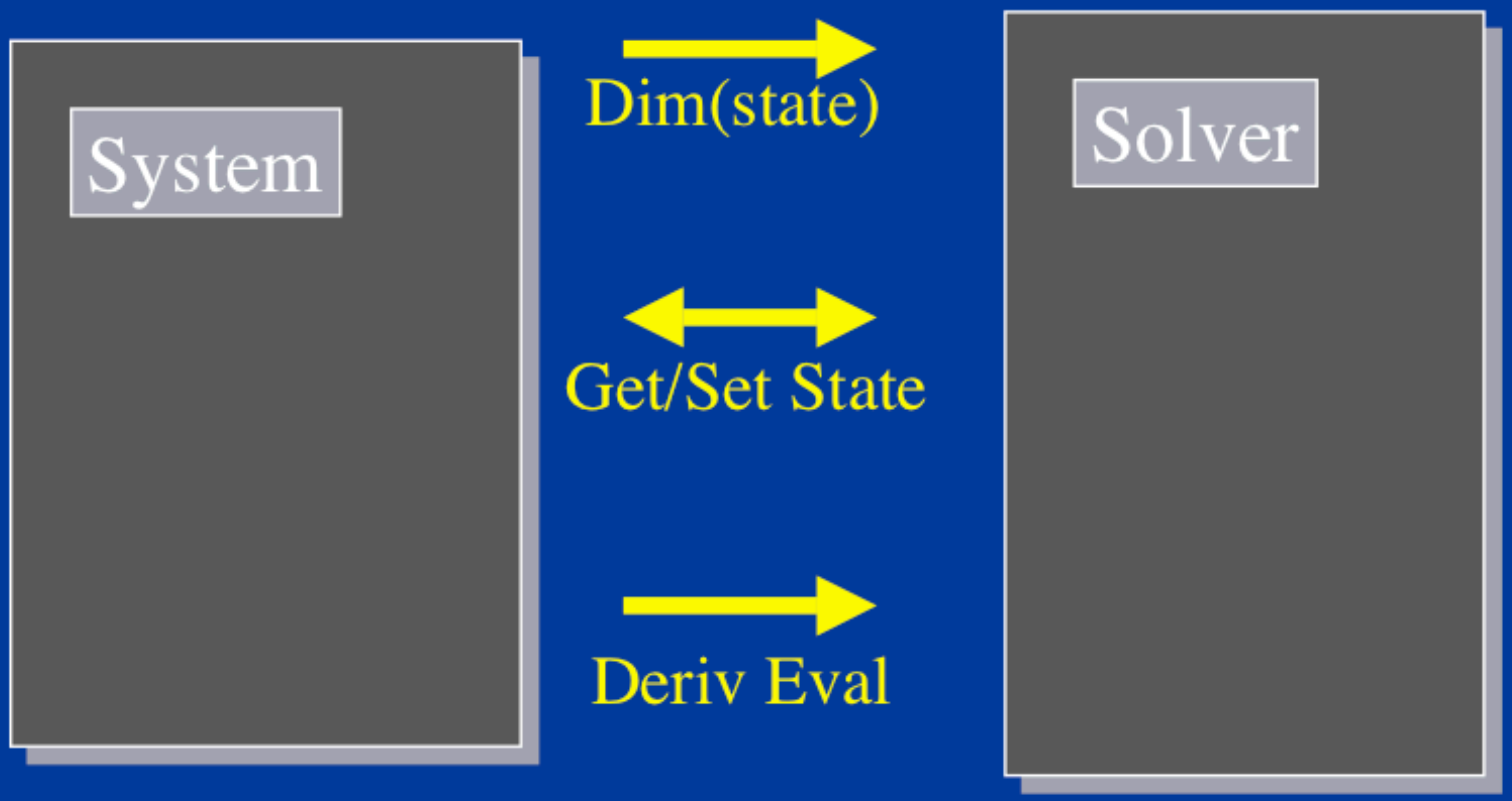
One Step: Implicit vs. Explicit



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

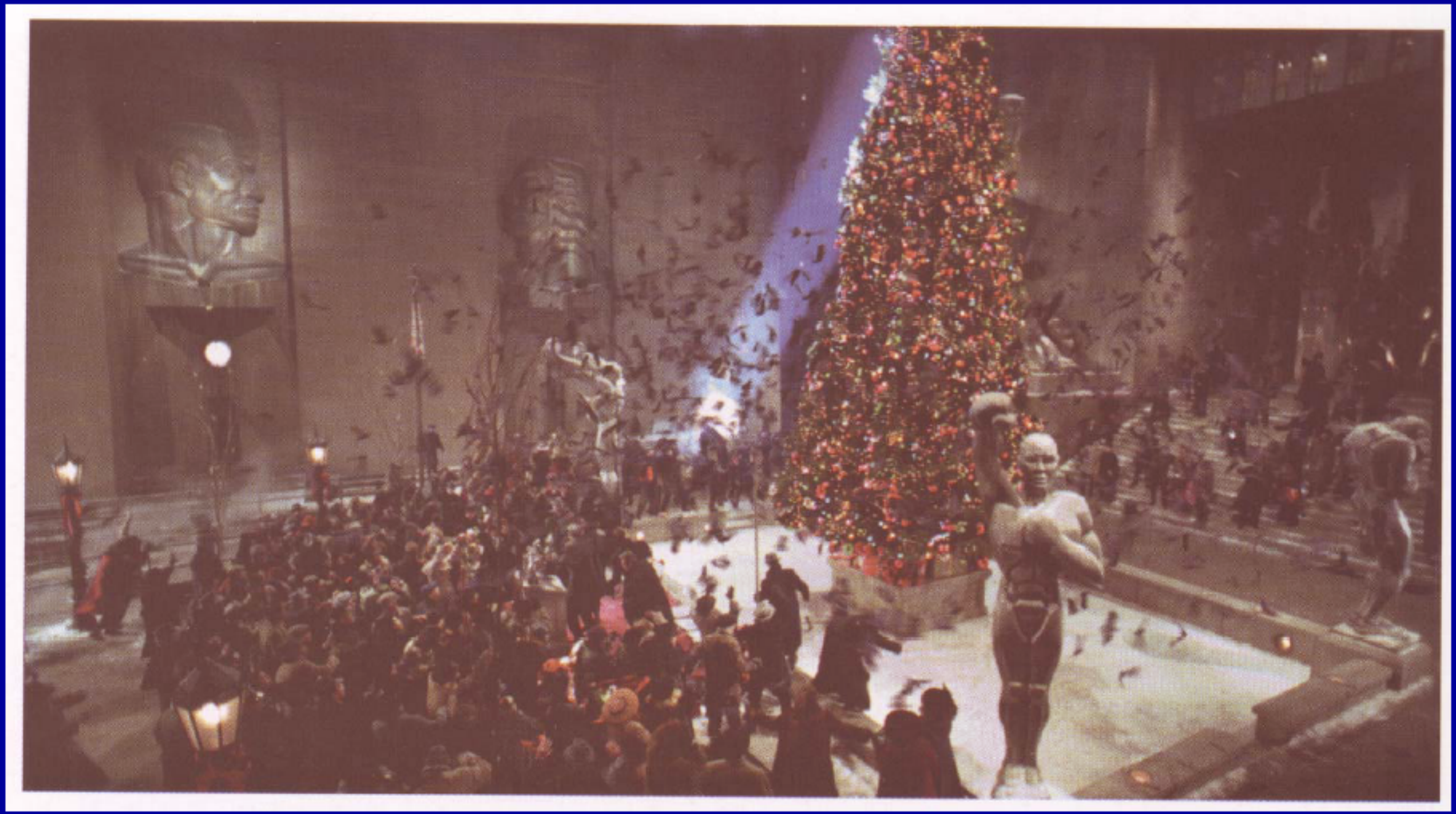
```
void eulerStep(Sys sys, float h) {
   float t = getTime(sys);
   vector<float> x0, deltaX;

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

Particle Systems

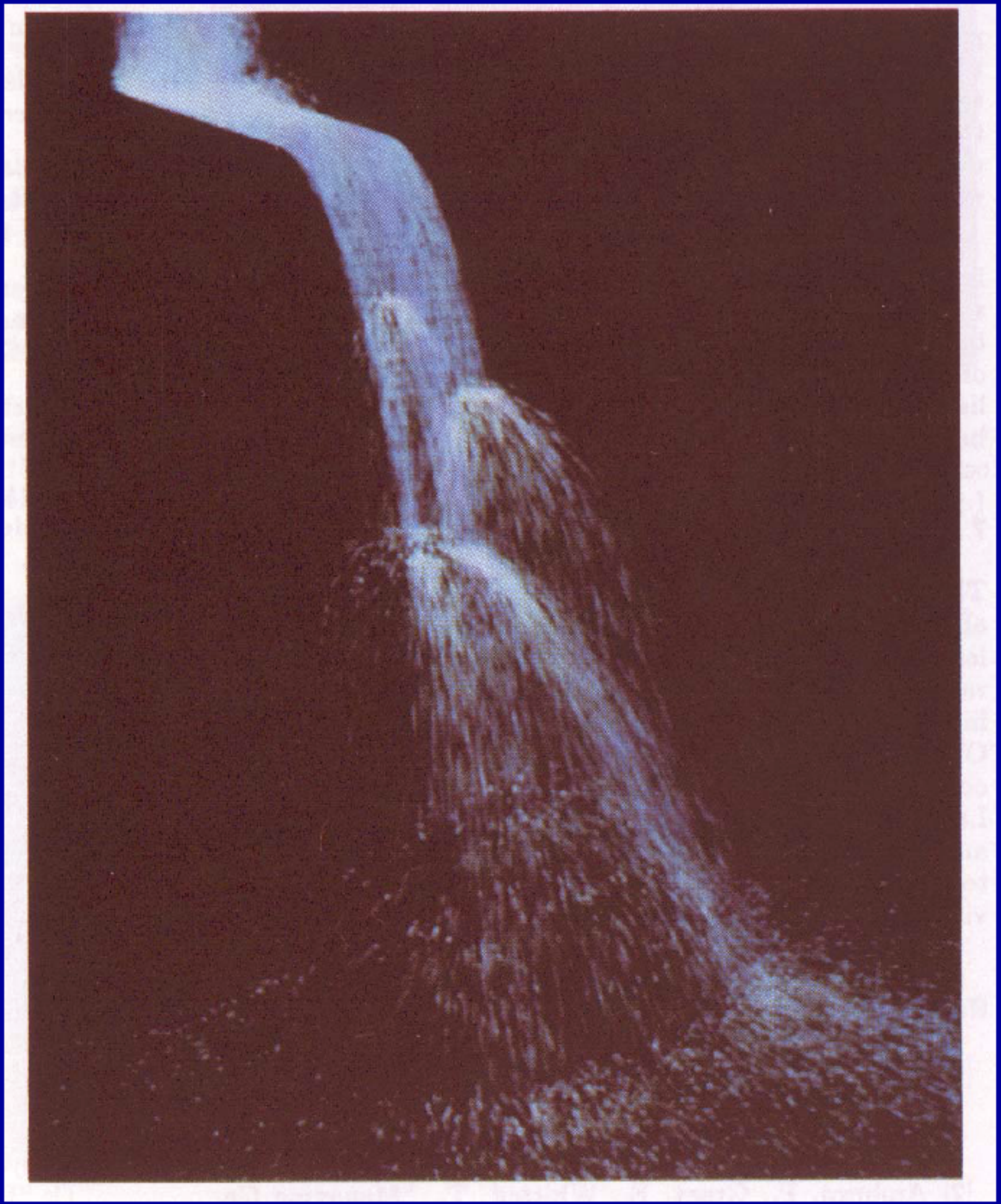
Particle Systems

Clouds
Smoke
Fire
Waterfalls
Fireworks



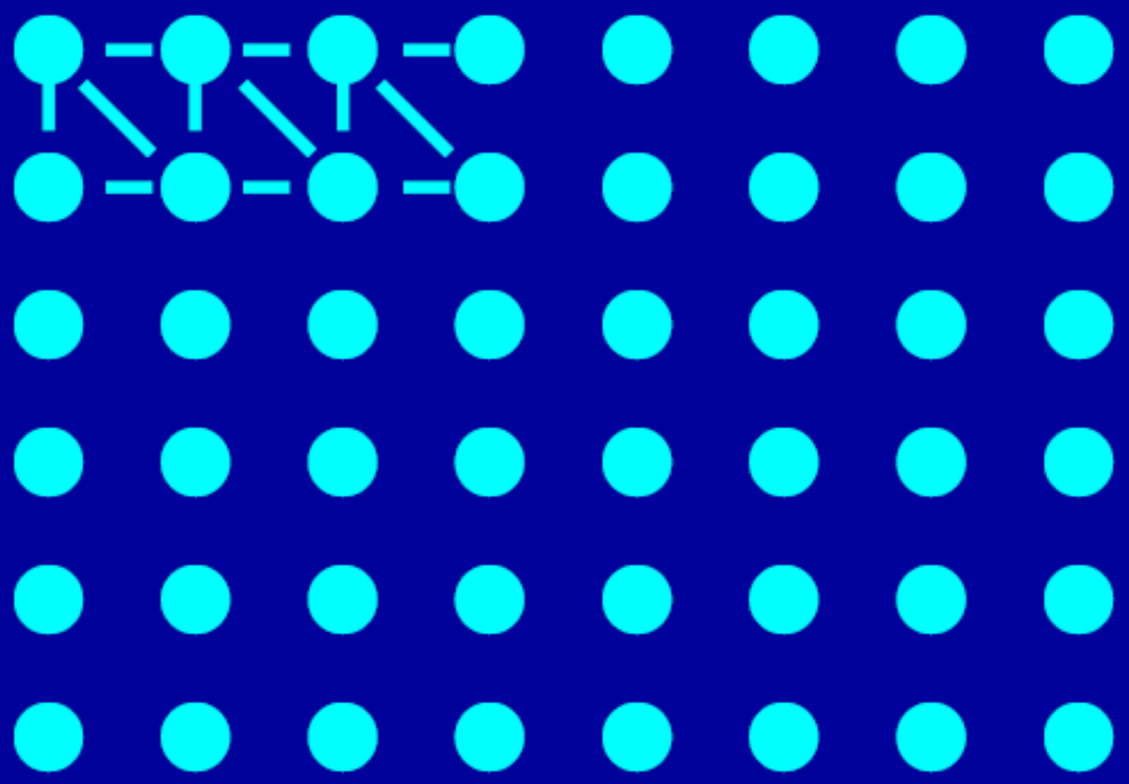
Reeves ’83, the Wrath of Khan
Batman Returns, using Reynold’s flocking algorithms

Karl Sims, Particle Dreams

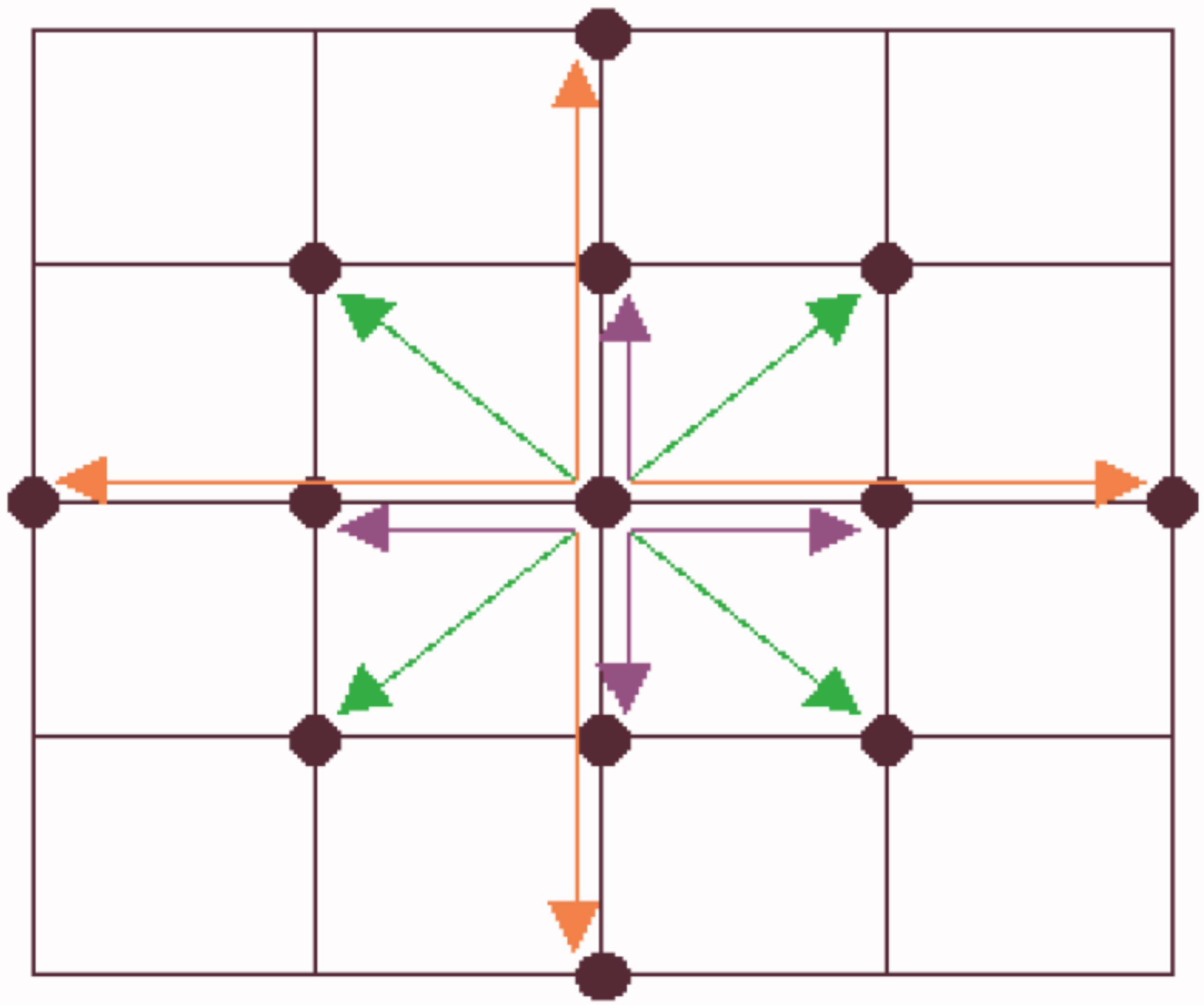


Spring-Mass Systems

Cloth in 2D
Jello in 3D



Cloth Simulation



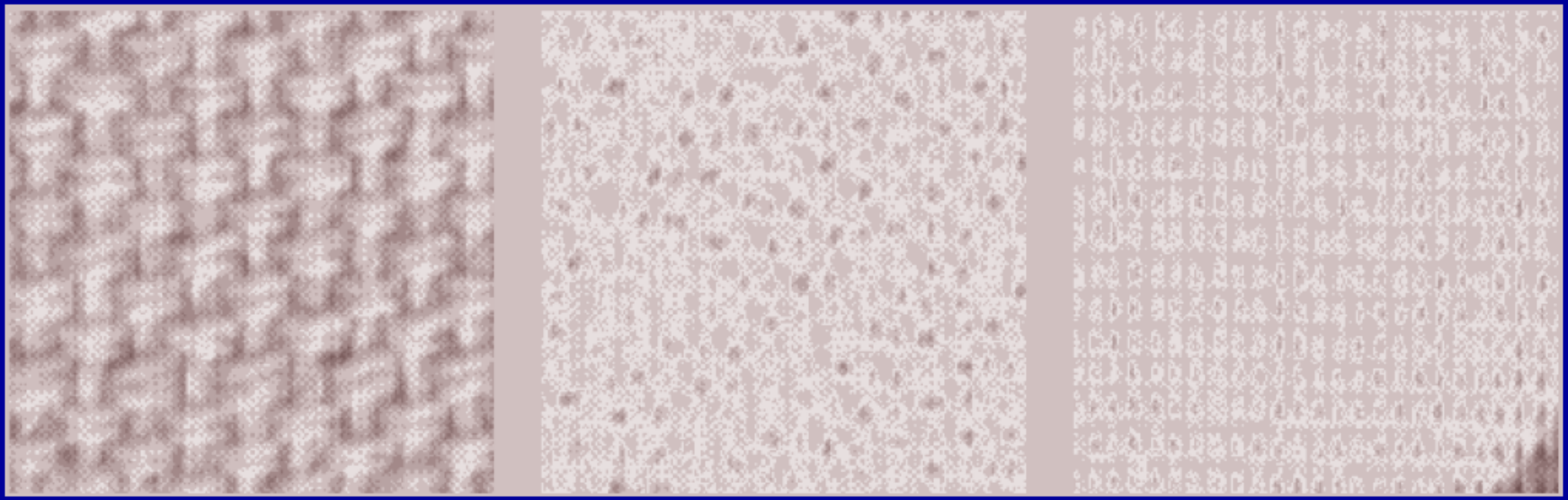
Cloth forces:

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