Particle Systems

Reading

- Required:
  - Witkin, *Particle System Dynamics*, SIGGRAPH ’97 course notes on Physically Based Modeling.

- Optional
  - Witkin and Baraff, *Differential Equation Basics*, SIGGRAPH ’97 course notes on Physically Based Modeling.
  - Hocknew and Eastwood. *Computer simulation using particles*. Adam Hilger, New York, 1988.
  - Gavin Miller. “The motion dynamics of snakes and worms.” *Computer Graphics* 22:169-178, 1988.

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

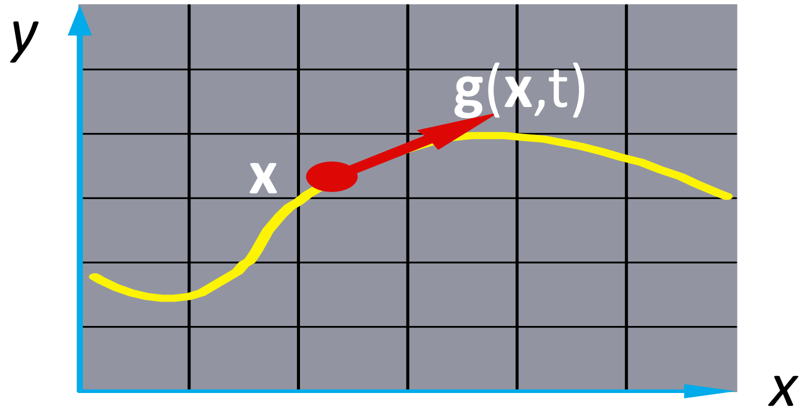
We begin with a single particle with:

– Position, $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

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

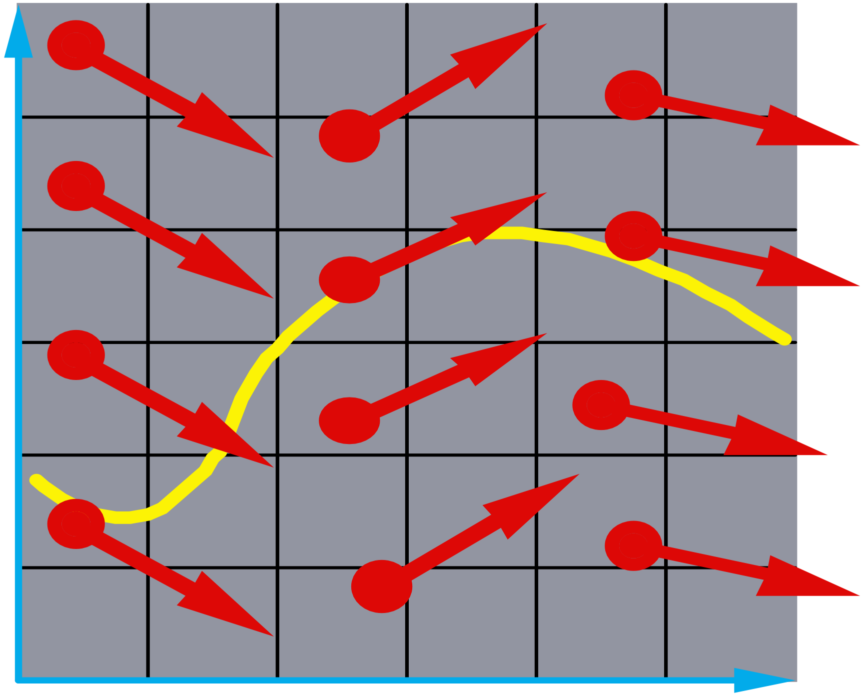
Suppose the velocity is dictated by some driving function **g**:

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



Vector fields

At any moment in time, the function **g** defines a vector field over **x**:

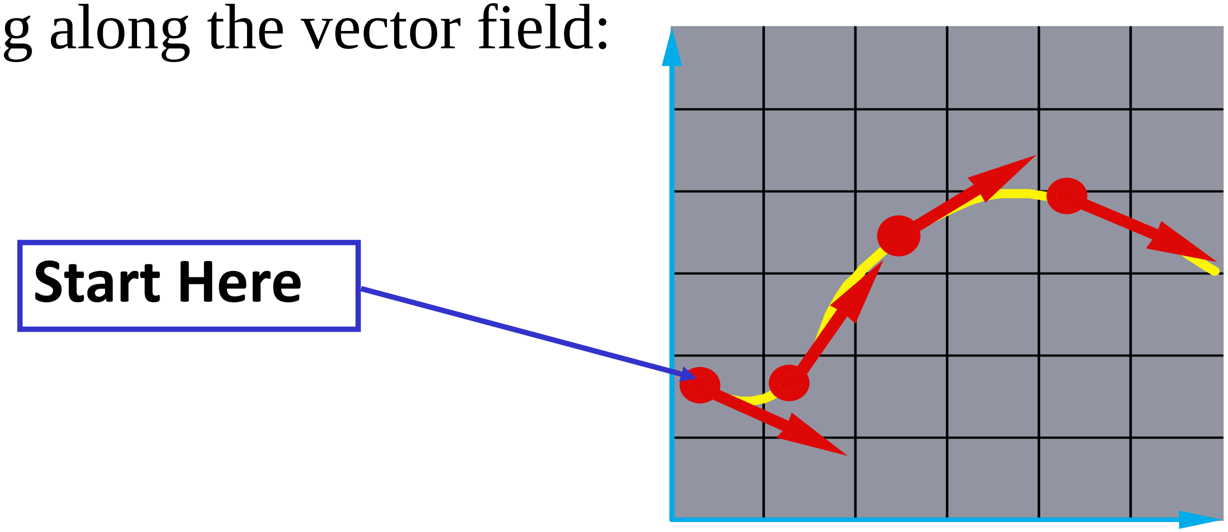


How does our particle move through the vector field?

Diff eqs and integral curves

The equation $\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, t)$ is actually a **first order differential equation**.

We can solve for **x** through time by starting at an initial point and stepping along the vector field:



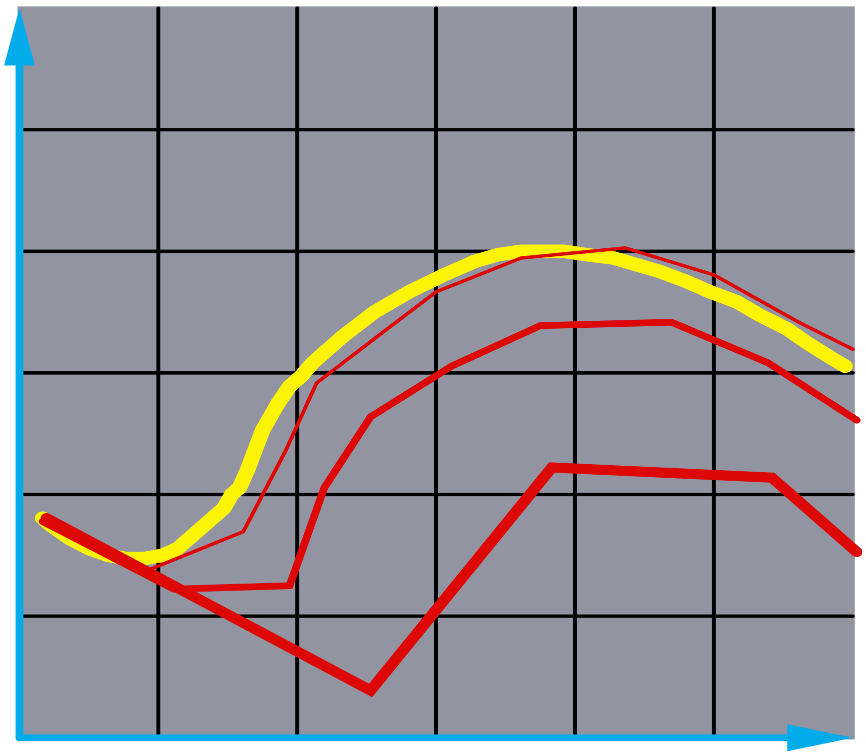
This is called an **intial value problem** and the solution is called an **integral curve**.

Euler's method

One simple approach is to choose a time step, Δt, and take linear steps along the flow:

$$\begin{aligned}\mathbf{x}(t+\Delta t) &= \mathbf{x}(t)+\Delta t \cdot \dot{\mathbf{x}}(t) \\ &= \mathbf{x}(t)+\Delta t \cdot \mathbf{g}(\mathbf{x},t)\end{aligned}$$

This approach is called **Euler's method** and looks like:



Properties:

- Simplest numerical method
- Bigger steps, bigger errors

Need to take pretty small steps, so not very efficient. Better (more complicated) methods exist, e.g., "Runge-Kutta."

Particle in a force field

- Now consider a particle in a force field $\mathbf{f}$.
- In this case, the particle has:
  - Mass, m
  - Acceleration, $\mathbf{a} \equiv \ddot{\mathbf{x}} = \frac{d\mathbf{v}}{dt} = \frac{d^2\mathbf{x}}{dt^2}$
- The particle obeys Newton's law: $\mathbf{f} = m\mathbf{a} = m\ddot{\mathbf{x}}$
- The force field $\mathbf{f}$ 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.

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

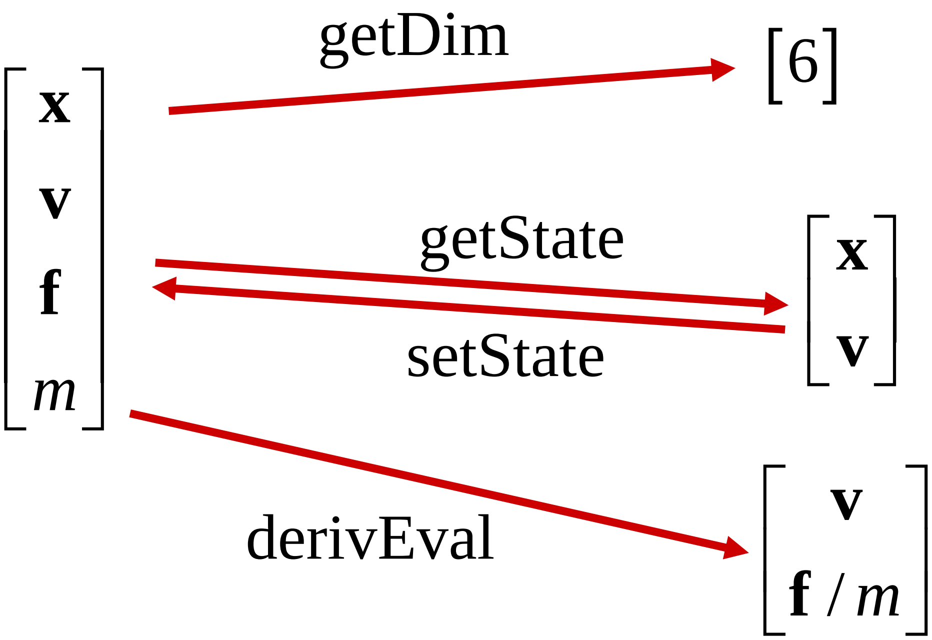
A vanilla 1st-order differential equation.

Particle structure

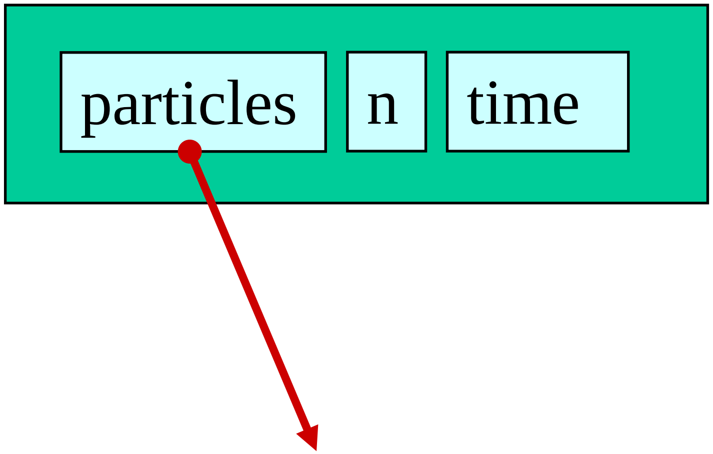
Position
in phase space

$$\left[\begin{array}{c}\mathbf{x}\\ \mathbf{v}\\ \mathbf{f}\\ m\end{array}\right] \begin{array}{l}\leftarrow \text{position}\\ \leftarrow \text{velocity}\\ \leftarrow \text{force accumulator}\\ \leftarrow \text{mass}\end{array}$$

Solver interface

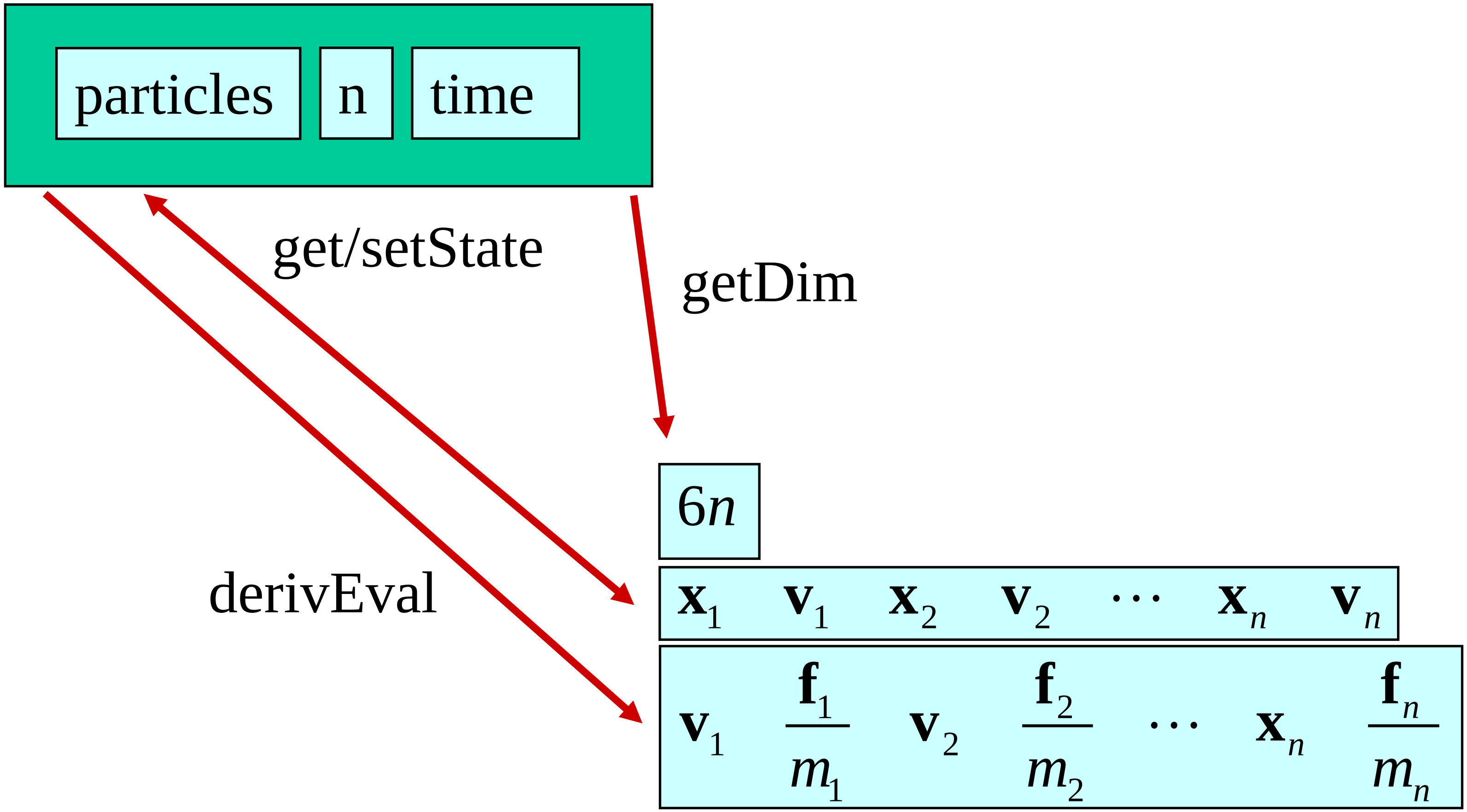


Particle systems



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

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
```

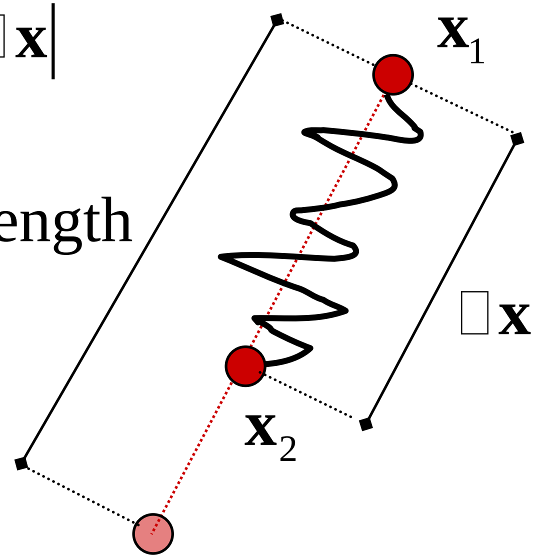
Damped spring

Force law:

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

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

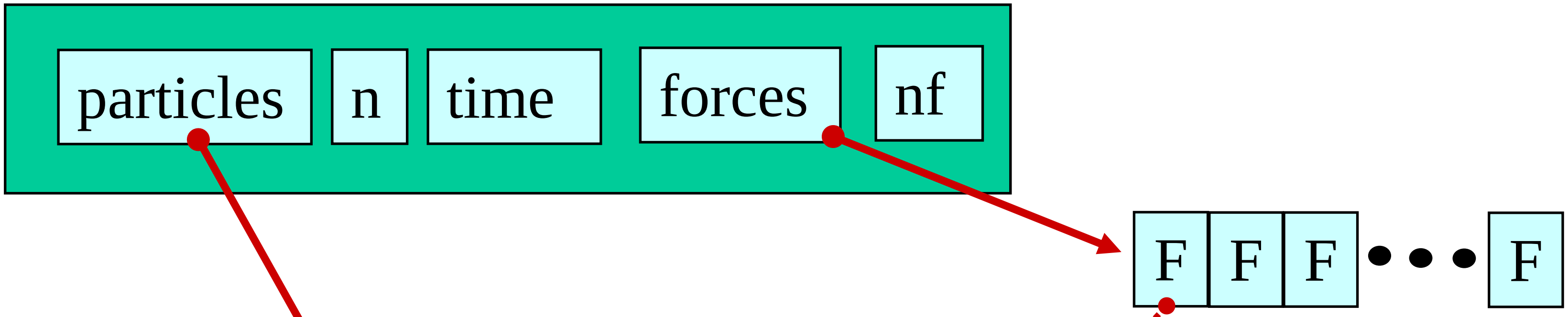
$\mathbf{r}$ = rest length



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

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

Particle systems with forces

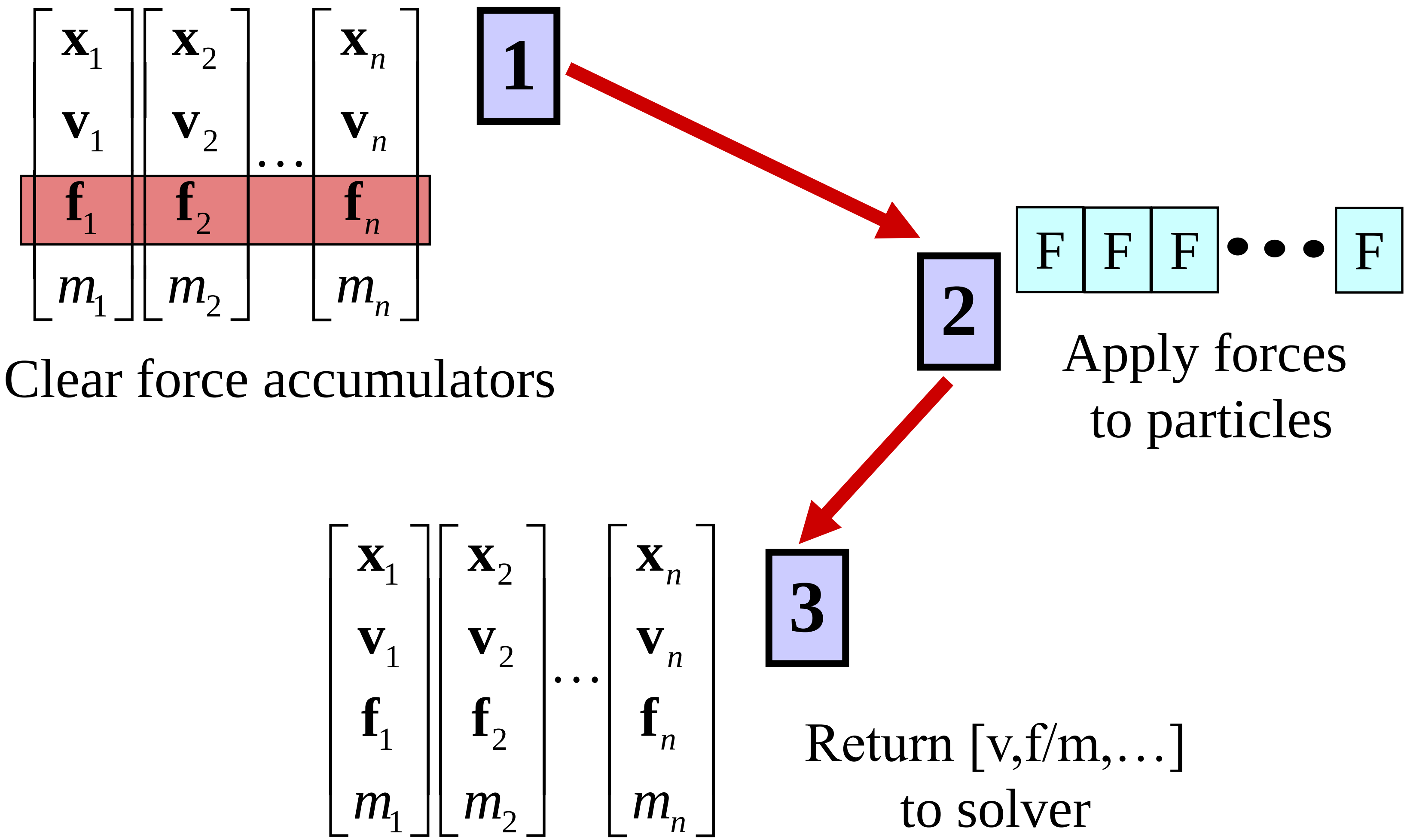


$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{v}_1 \\ \mathbf{f}_1 \\ m_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{v}_2 \\ \mathbf{f}_2 \\ m_2 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{x}_n \\ \mathbf{v}_n \\ \mathbf{f}_n \\ m_n \end{bmatrix}$$

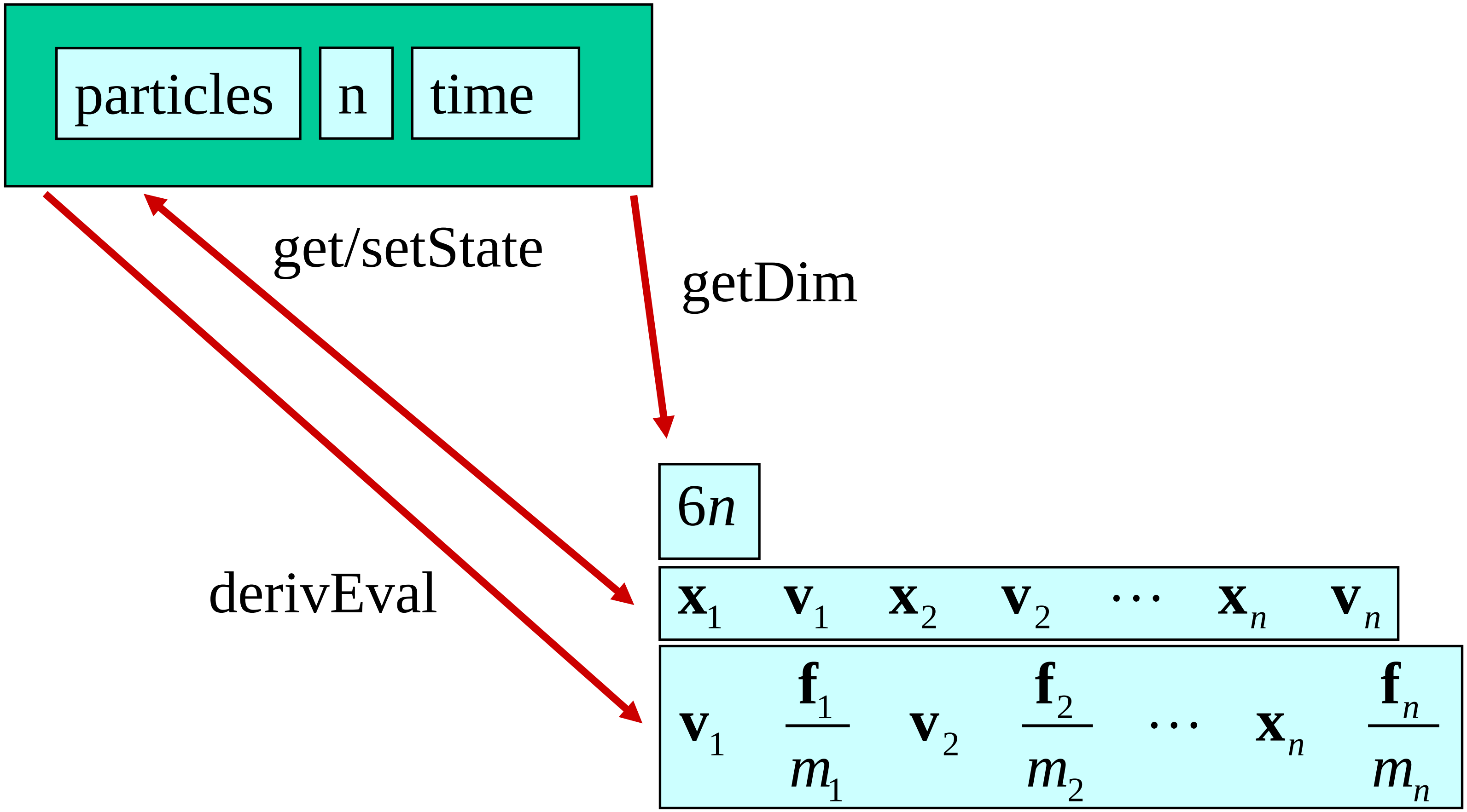
derivEval loop

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



Solver interface

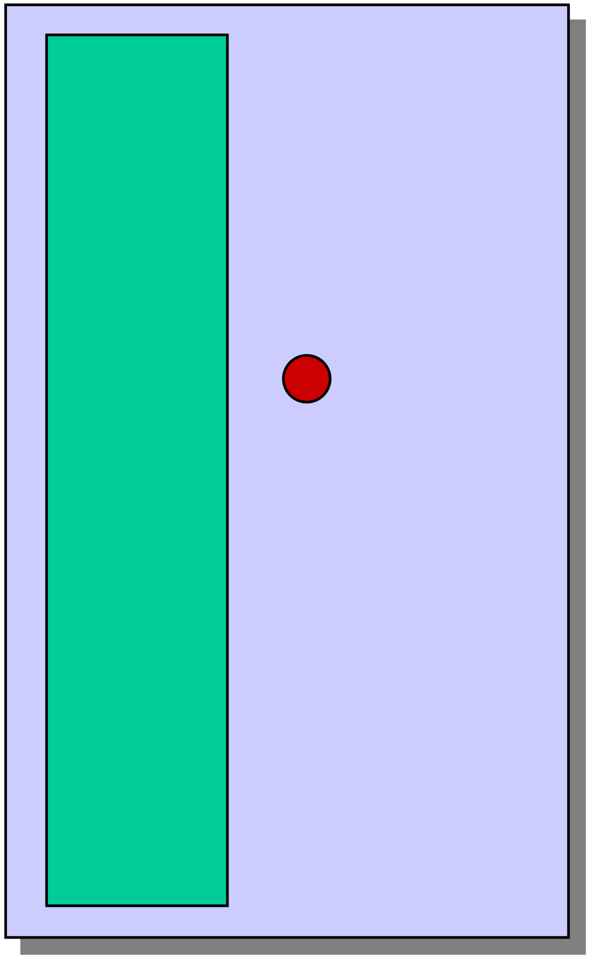


Differential equation solver

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

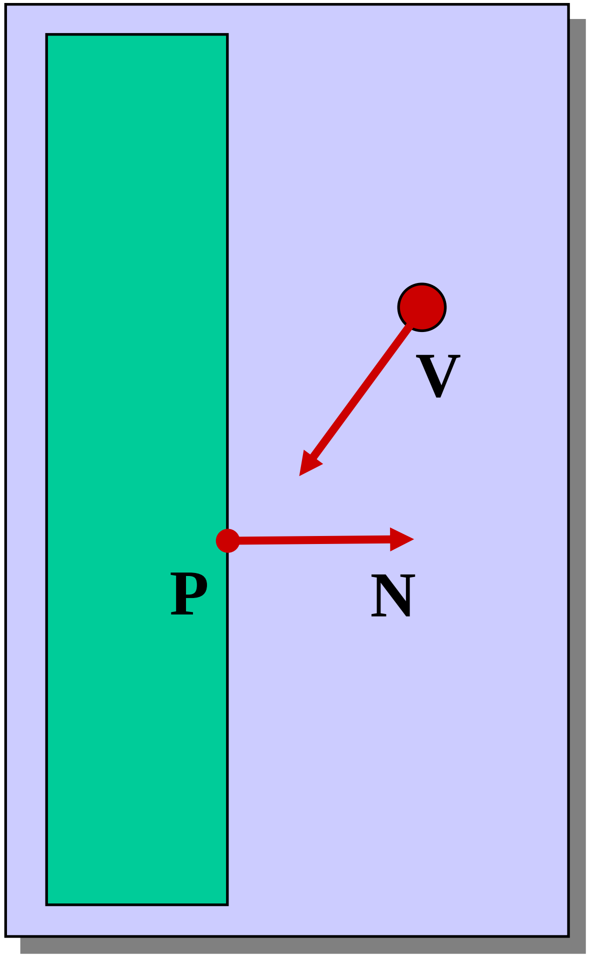
Euler method:
$$\begin{bmatrix} \mathbf{x}_1^{i+1} \\ \mathbf{v}_1^{i+1} \\ \vdots \\ \mathbf{x}_n^{i+1} \\ \mathbf{v}_n^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{i} \\ \mathbf{v}_1^{i} \\ \vdots \\ \mathbf{x}_n^{i} \\ \mathbf{v}_n^{i} \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{v}_1^{i}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

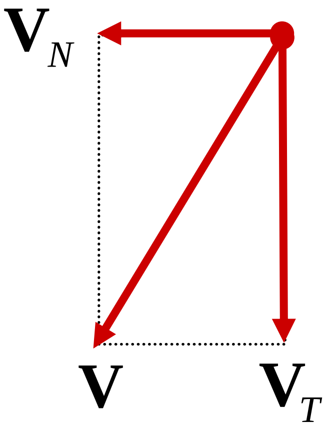
Bouncing off the walls



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

Normal and tangential components

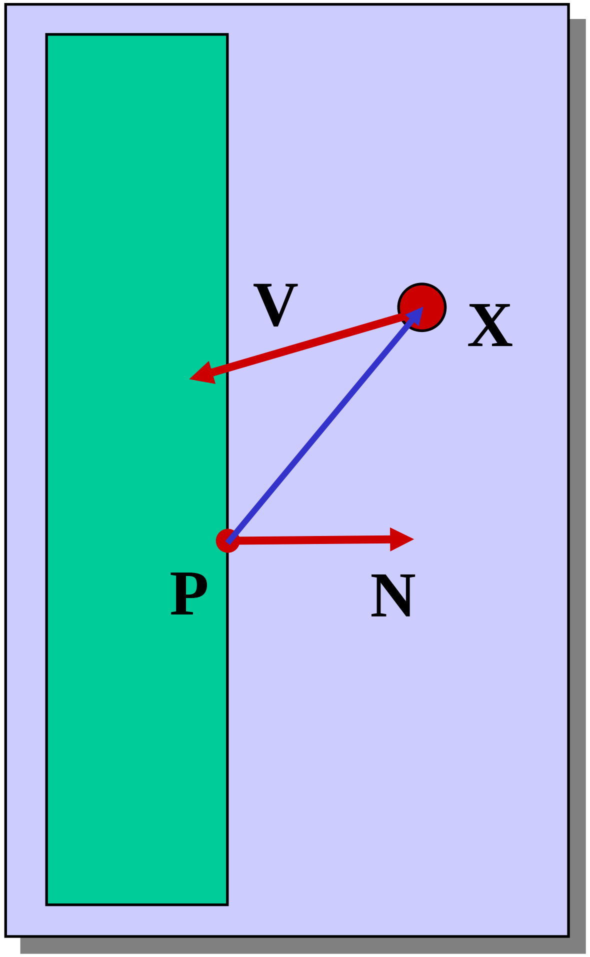




$$\mathbf{V}_N = (\mathbf{N}\cdot\mathbf{V})\mathbf{N}$$

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

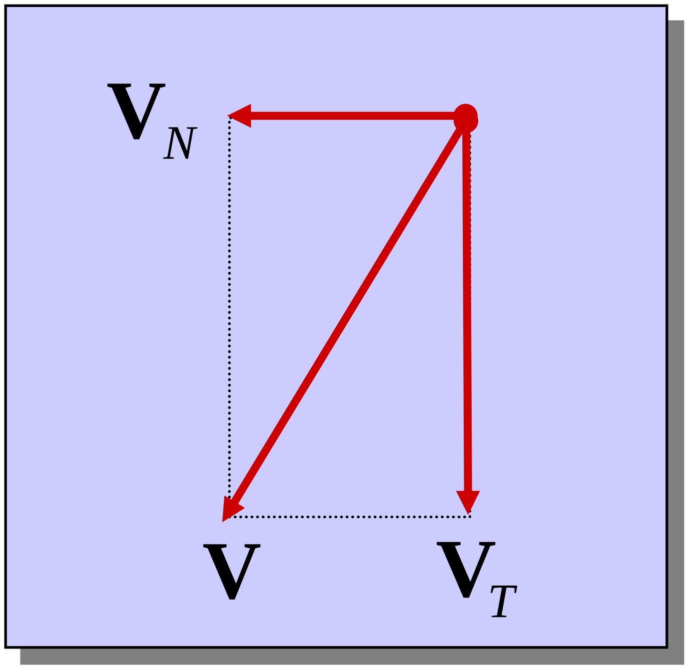
Collision Detection



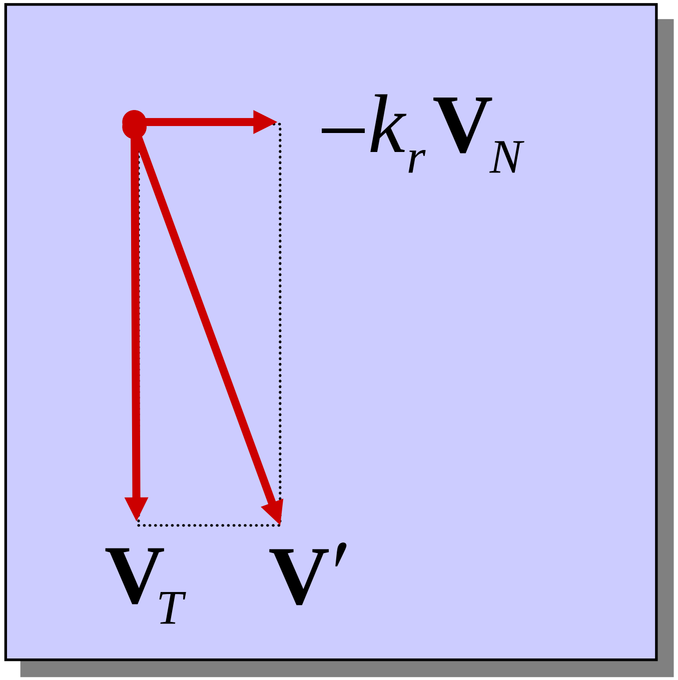
$(\mathbf{X}-\mathbf{P})\cdot\mathbf{N} < \varepsilon$ Within ε of the wall

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

Collision Response



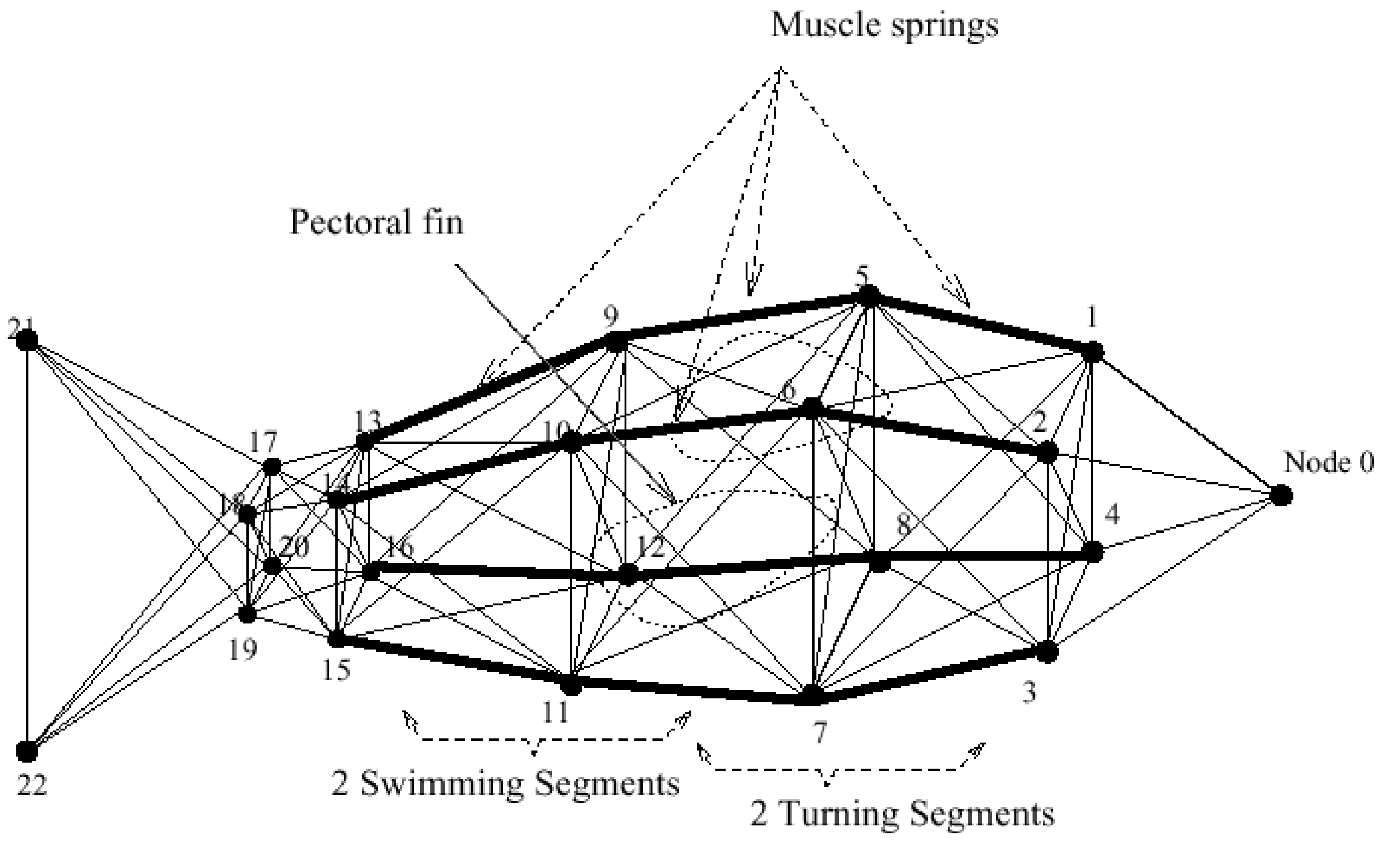
before



after

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

Artificial Fish



Related Research

- Determining dynamic parameters for cloth simulation

Summary

What you should take away from this lecture:

- The meanings of all the **boldfaced** terms
- Euler method for solving differential equations
- Combining particles into a particle system
- Physics of a particle system
- Various forces acting on a particle
- Simple collision detection