Particle Dynamics

Reading

Required:

- Witkin, Particle System Dynamics, SIGGRAPH ‘97 course notes on Physically Based Modeling (on which lectures are based)
  - available on the course web
- Angel, pp. 467-481 (more readable)
  - available in library reserve

Optional:

- 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

What are particle systems?

- Note that although the dynamics of a simple particle system are based on each particle being treated as a point mass, the user can specify how each particle is rendered.
- Each particle may represent a person in a crowd scene, or a molecule in a chemical-synthesis application, or a portion of a cloth piece in the simulation of a flag blowing in the wind.

Overview

- One lousy particle
- Particle systems
- Forces: gravity, springs
- Implementation

Particle in a flow field

- Let's consider the 2D case first.
- We begin with a single particle with
  - position,

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

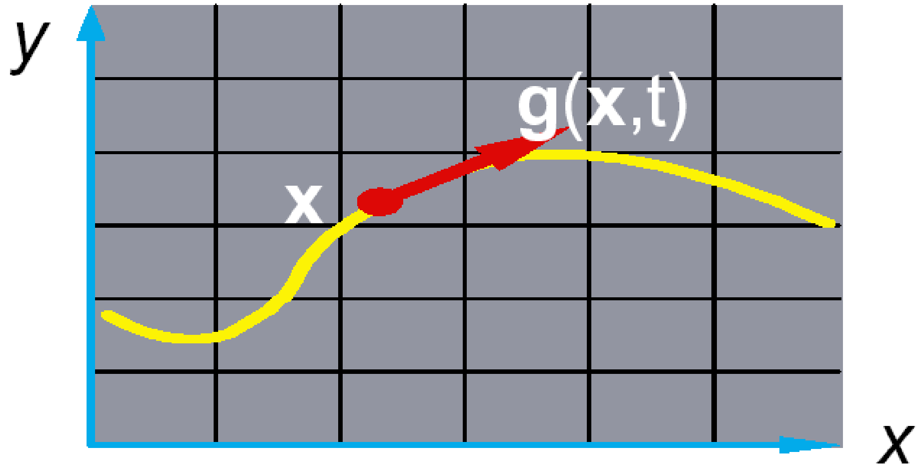
  - velocity,

$$\mathbf{v} \equiv \dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix}$$

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

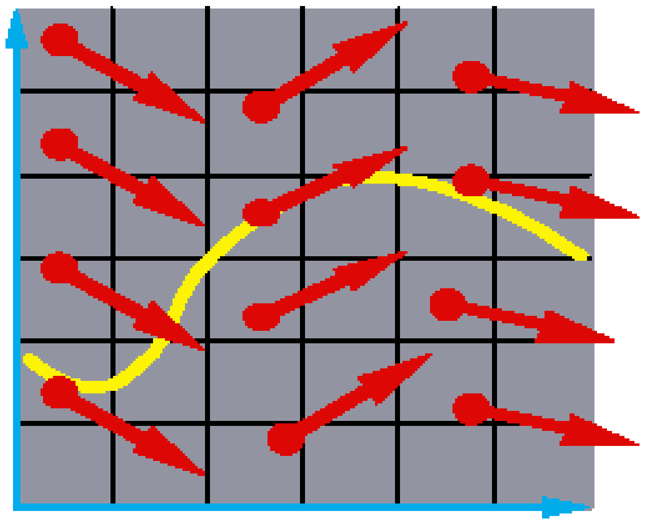
$$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, t)$$

$$\mathbf{g}(x,y,t)$$



Vector fields

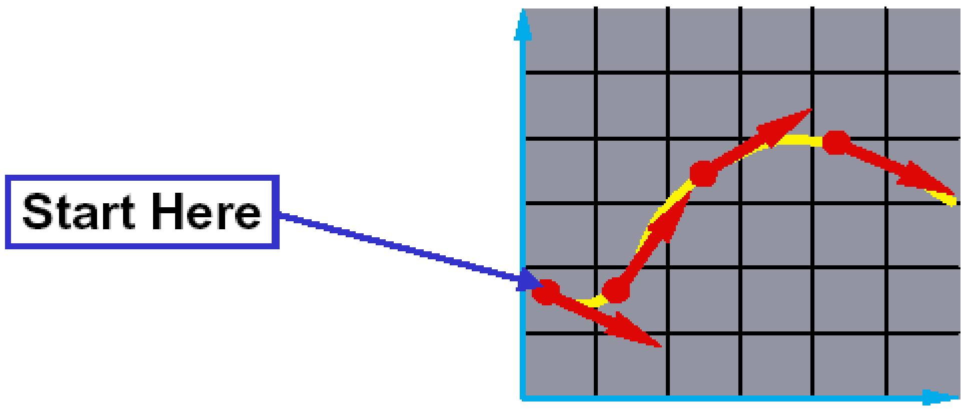
- At any moment in time *t*, the function **g** defines a vector field over **x**:



- How does our particle move through the vector field?

ODE and Integral Curves

- The equation $\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, t)$ is actually a **first order (ordinary) differential equation**.
- We can solve for **x(t)** through time by starting at an initial point and stepping along the vector field:

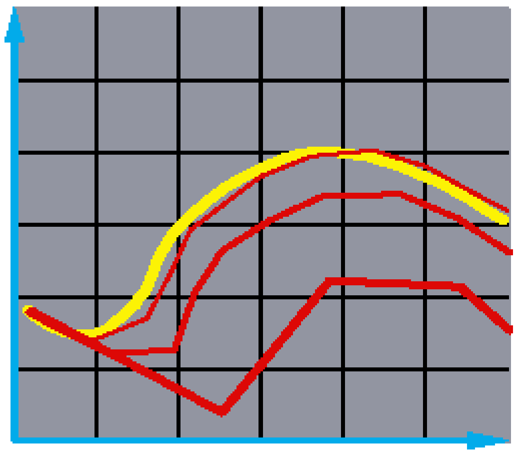


- This is called an **initial value problem** and the solution is called an **integral curve**.

Euler’s Method

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

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



- This approach is called Euler’s method.
- Properties:
  - Simple 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 **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 **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

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

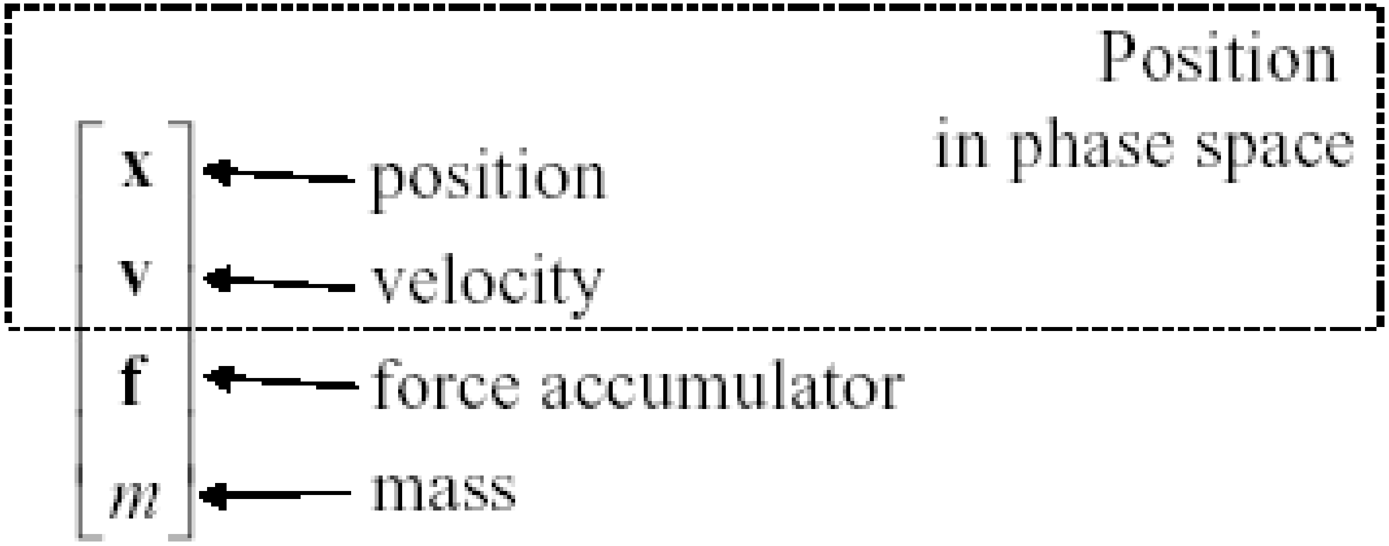
$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix}$$

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

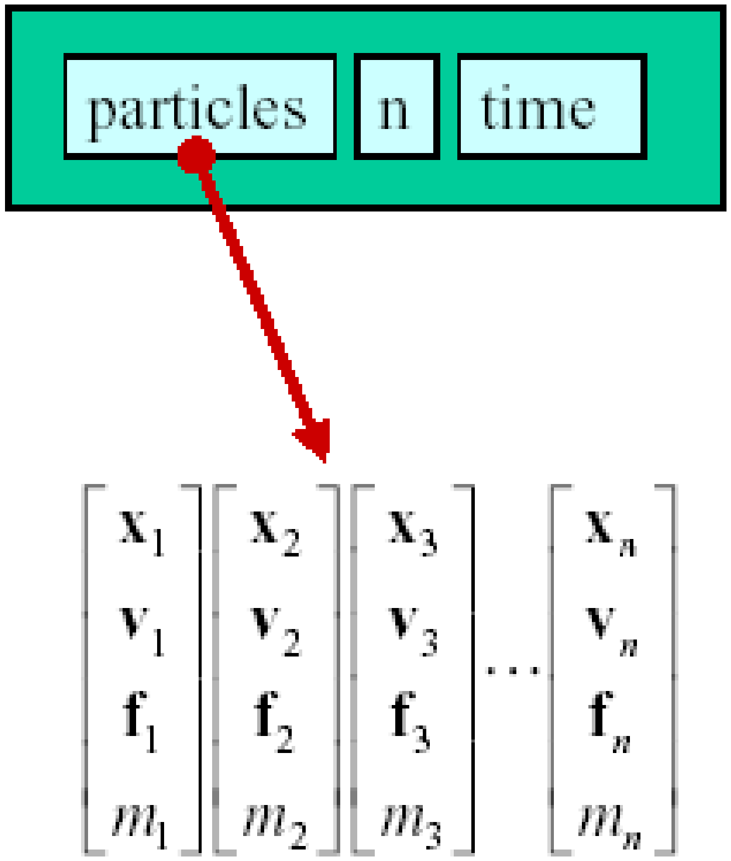
Let's consider the 3D case from now on.

- Concatenate **x** and **v** to make a 6-vector: position in **phase space**.
- Taking the time derivative: another 6-vector
- A 1st-order differential equation.

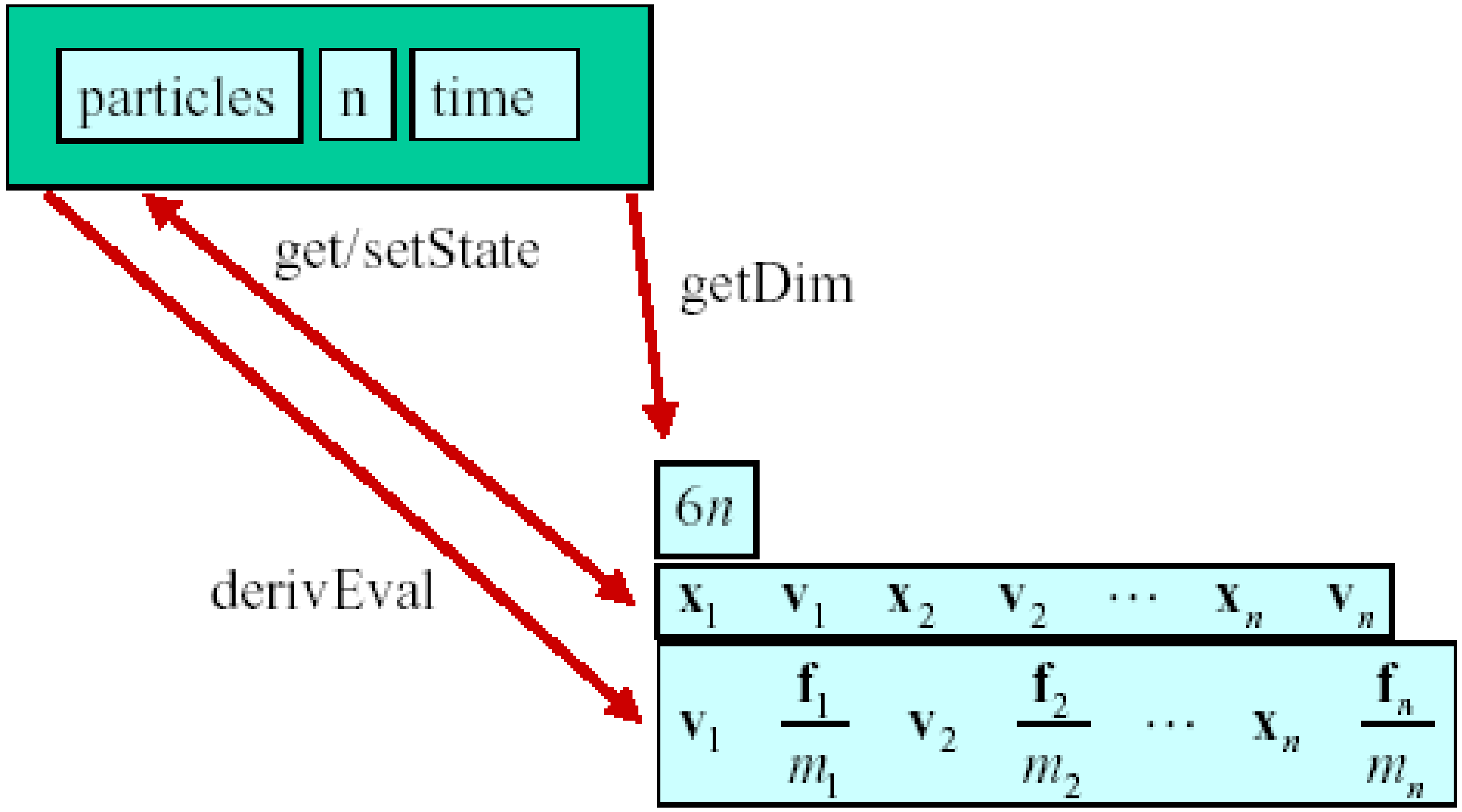
Particle Structure



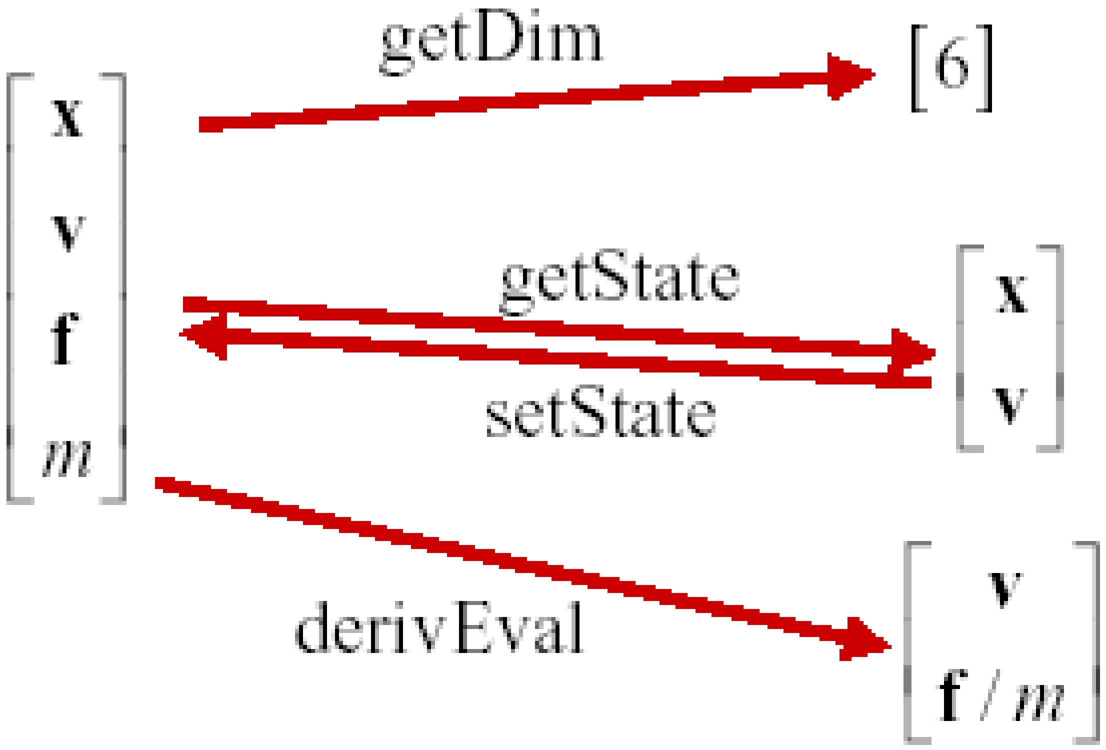
Particle Systems


particles
n
time
$\mathbf{x}_1$ $\mathbf{v}_1$ $\mathbf{f}_1$ m_1
$\mathbf{x}_2$ $\mathbf{v}_2$ $\mathbf{f}_2$ m_2
$\mathbf{x}_3$ $\mathbf{v}_3$ $\mathbf{f}_3$ m_3
$\cdots$
$\mathbf{x}_n$ $\mathbf{v}_n$ $\mathbf{f}_n$ m_n

Solver Interface


particles
n
time
get/setState
getDim
derivEval
$6n$
$\mathbf{x}_1 \quad \mathbf{v}_1 \quad \mathbf{x}_2 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{x}_n \quad \mathbf{v}_n$
$\mathbf{v}_1 \quad \frac{\mathbf{f}_1}{m_1} \quad \mathbf{v}_2 \quad \frac{\mathbf{f}_2}{m_2} \quad \cdots \quad \mathbf{x}_n \quad \frac{\mathbf{f}_n}{m_n}$

Solver Interface


getDim
[6]
getState
setState
derivEval
$\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\\ \mathbf{f}\\ m\end{bmatrix}$
$\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix}$
$\begin{bmatrix}\mathbf{v}\\ \mathbf{f}/m\end{bmatrix}$

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
```

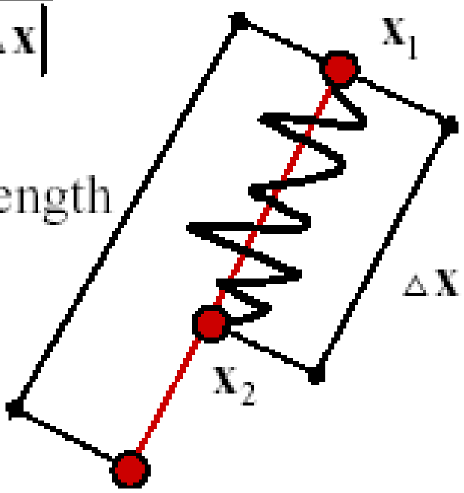
Damped spring

- Force law (Hooke's Law):

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

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

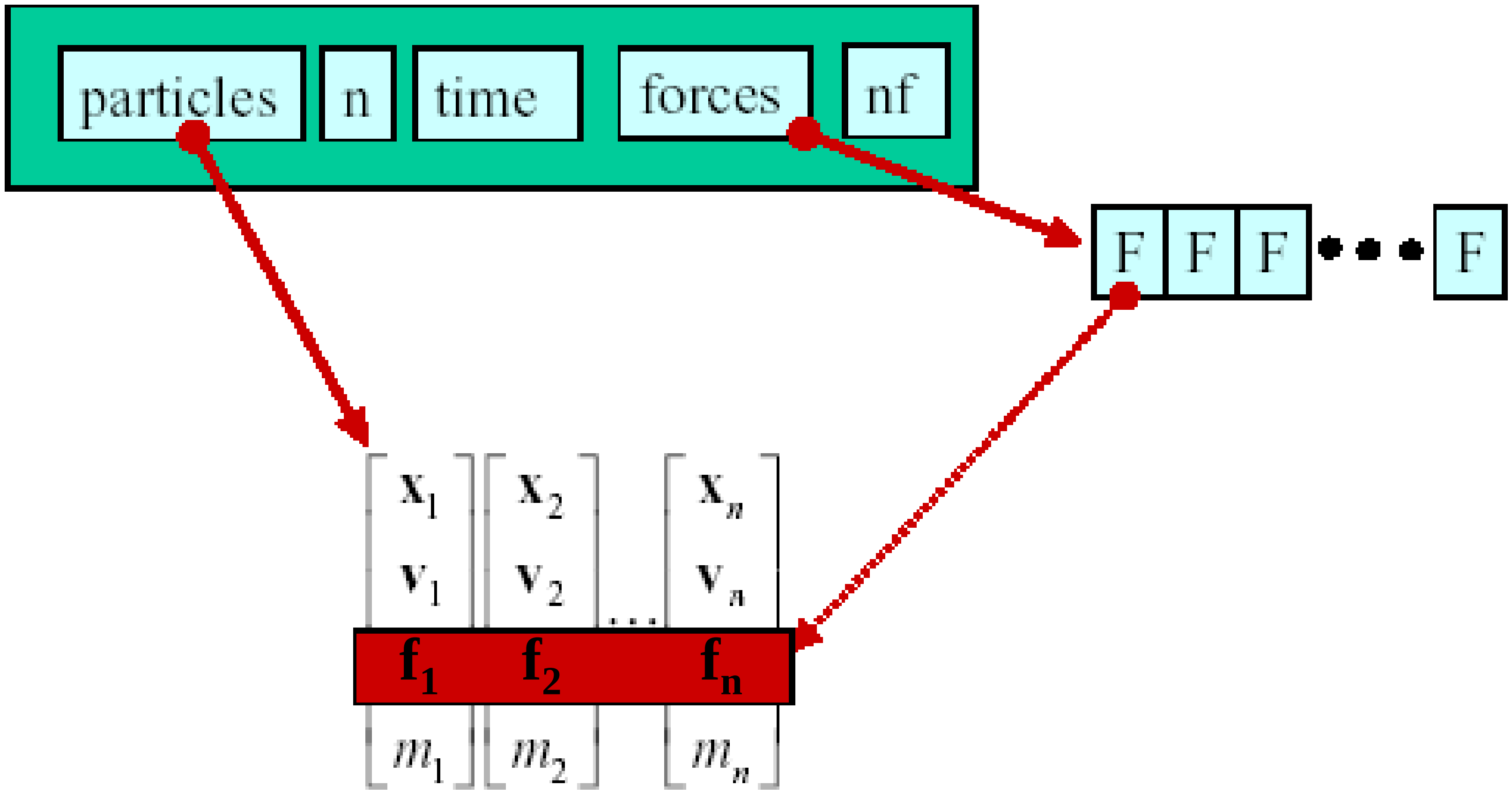
$\mathbf{r}$ = rest length



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

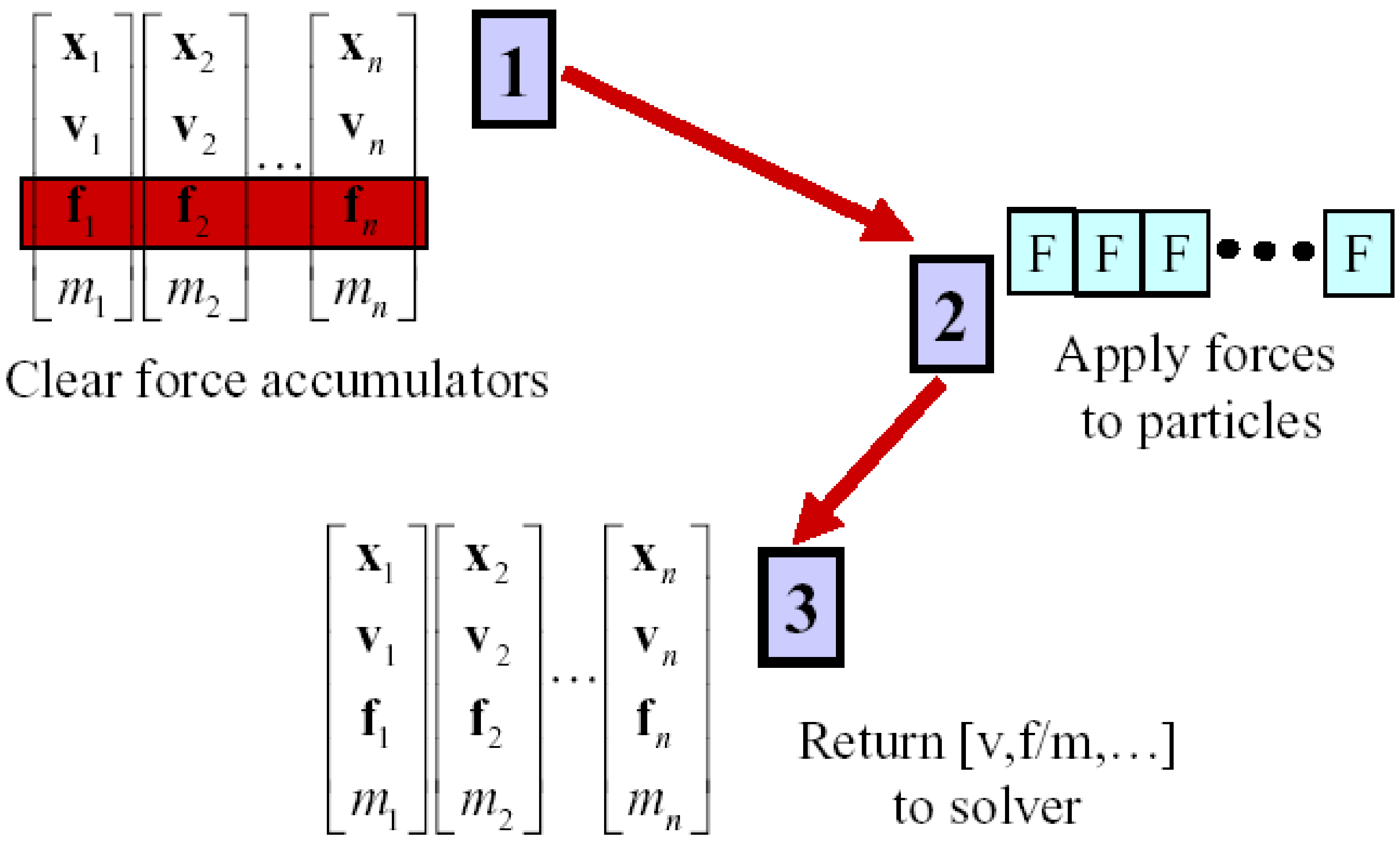
$\Delta\mathbf{v} = \mathbf{v}_1 - \mathbf{v}_2$

Particle systems with forces



particles
n
time
forces
nf
F
F
F
F
$\mathbf{x}_1$ $\mathbf{v}_1$ $\mathbf{f}_1$ m_1
$\mathbf{x}_2$ $\mathbf{v}_2$ $\mathbf{f}_2$ m_2
$\mathbf{x}_n$ $\mathbf{v}_n$ $\mathbf{f}_n$ m_n

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
3
$\begin{bmatrix}\mathbf{x}_1\\ \mathbf{v}_1\\ \mathbf{f}_1\\ m_1\end{bmatrix}\begin{bmatrix}\mathbf{x}_2\\ \mathbf{v}_2\\ \mathbf{f}_2\\ m_2\end{bmatrix}\cdots\begin{bmatrix}\mathbf{x}_n\\ \mathbf{v}_n\\ \mathbf{f}_n\\ m_n\end{bmatrix}$
Return [v,f/m,…] to solver

derivEval loop

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

Differential Equation Solver

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

Euler method:

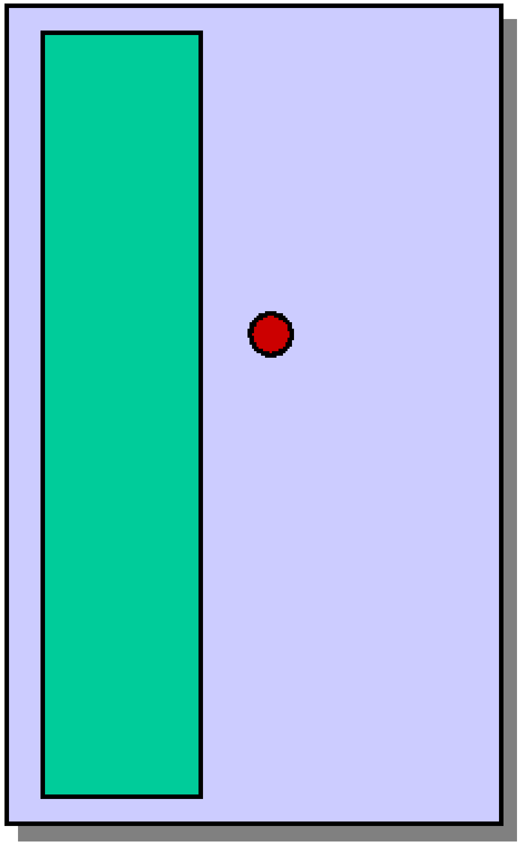
$$\begin{aligned} \mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \Delta t \cdot \dot{\mathbf{x}}(t) \\ &= \mathbf{x}(t) + \Delta t \cdot \mathbf{g}(\mathbf{x}, t) \end{aligned}$$

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

Summary

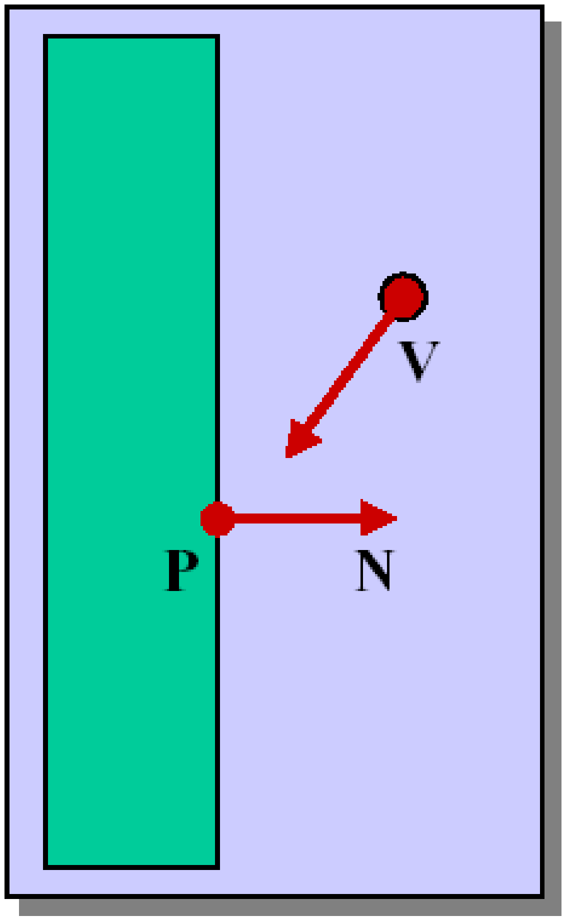
```
float time, delta;
float state[6n], force[3n];
state = get_initial_state();
for(time=t0;time<time_final;time+=delta) {
   /* compute forces */
   force=force_function (state,time);
   /* apply ODE solver */
   state = ode(force,state,time,delta);
   /* display result */
  render(state,time);
}
```

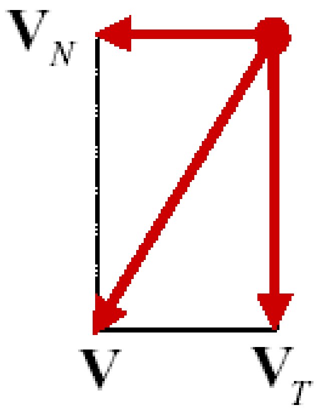
Bouncing off the walls



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

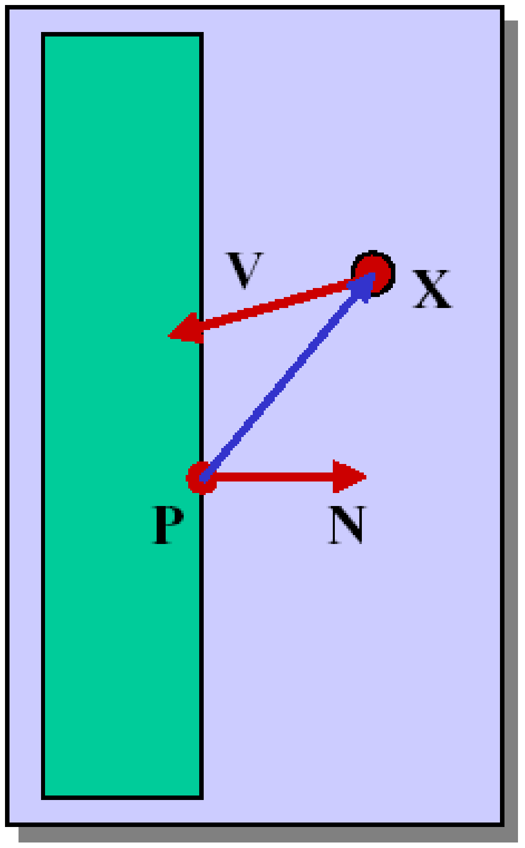
Normal and tangent components





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

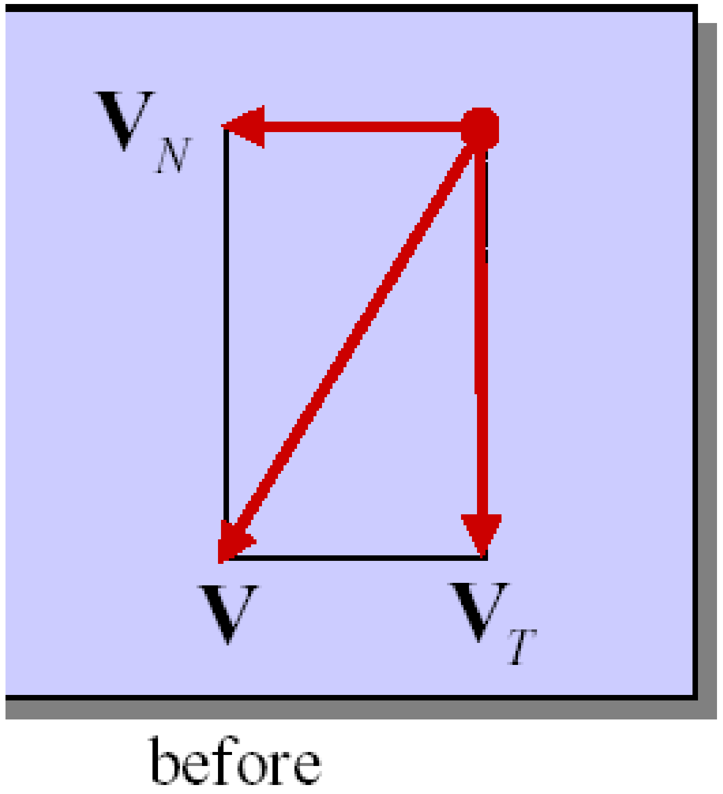
Collision Detection

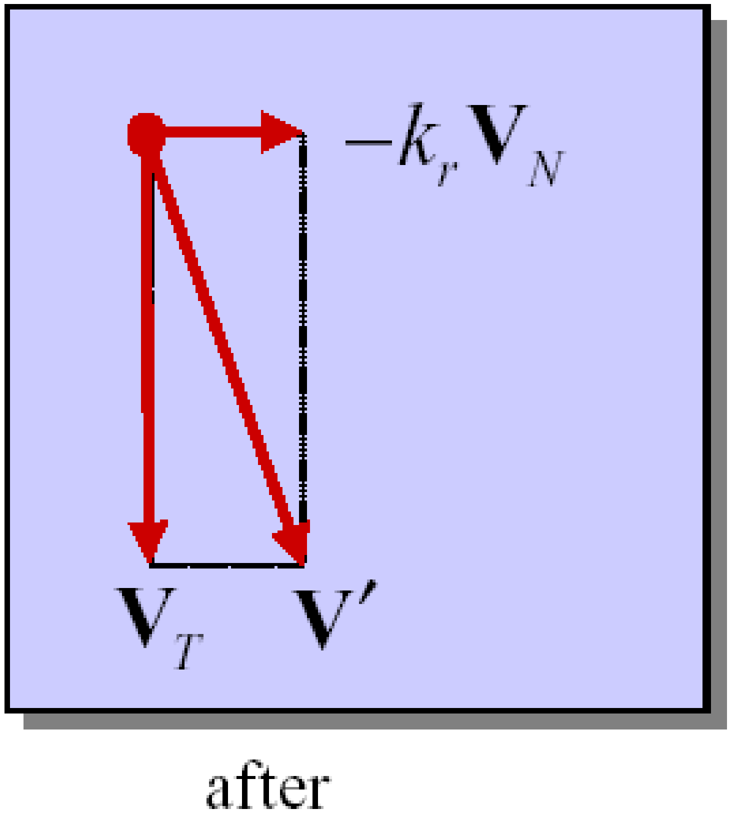


$(\mathbf{X}-\mathbf{P})\cdot\mathbf{N}<\varepsilon$ Within ε of the wall

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

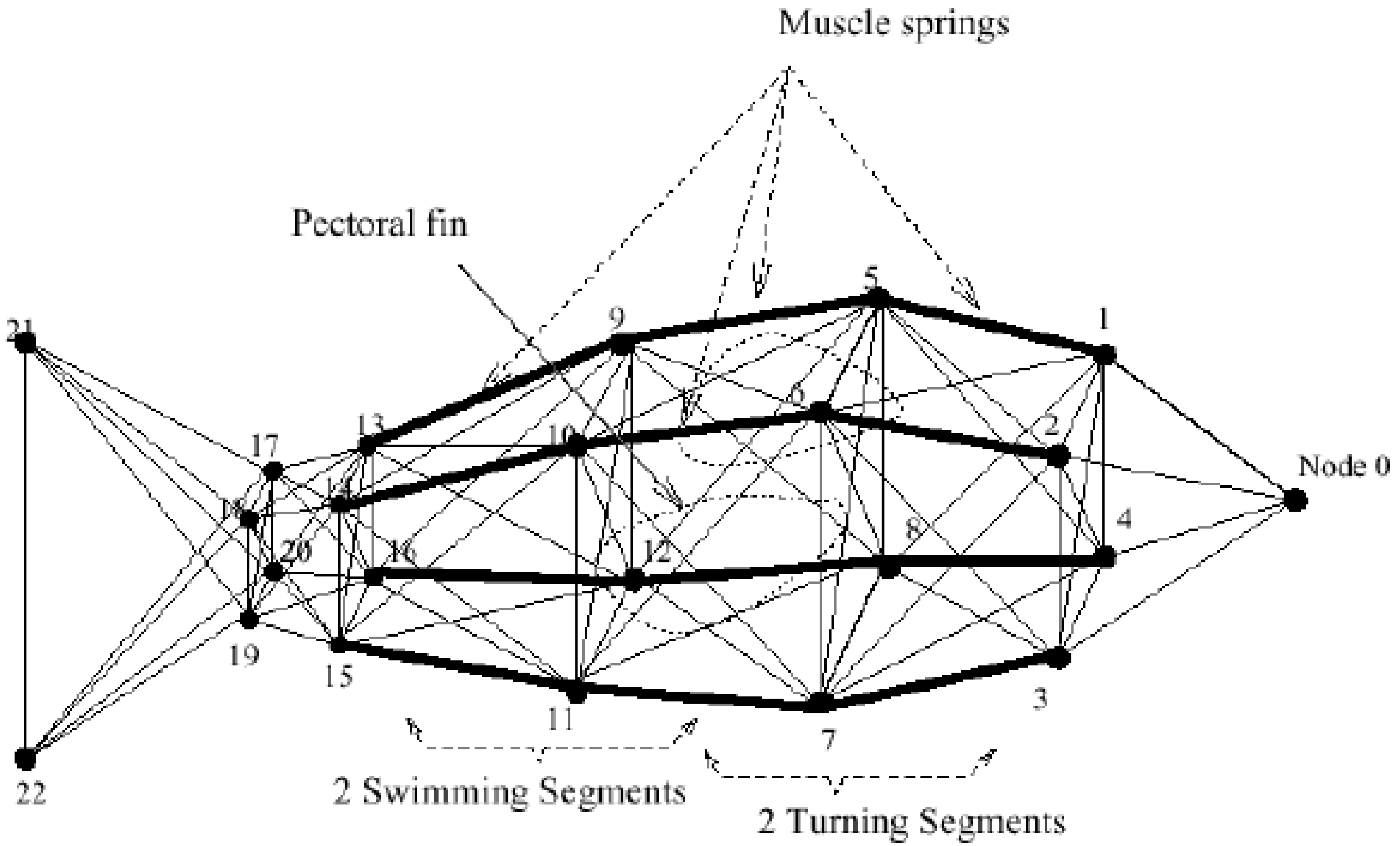
Collision Response





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

Example: Artificial Fish



Summary

What you should take from this lecture:

- The meaning of all the boldfaced terms
- Euler’s 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 and collision response