14. 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 (online).
- 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:

Overview

- A single particle
- Particle systems
- Forces: gravity, springs
- Collision detection

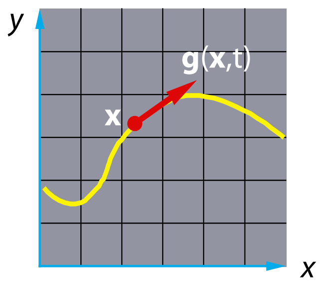
Particle in a flow field

We begin with a single particle with:

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

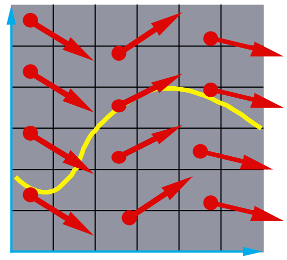
Suppose the velocity is actually dictated by some driving function **g**:

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



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?

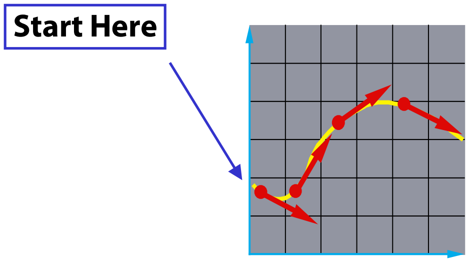
Diff eqs and integral curves

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:



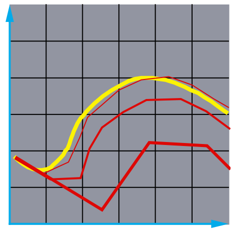
This is called an **intial value problem** and the solution is called an **integral curve**.

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 **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:

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

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.

Differential equation solver

Starting with:

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

Applying Euler method:

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

And making substitutions:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t)$$
$$\dot{\mathbf{x}}(t + \Delta t) = \dot{\mathbf{x}}(t) + \Delta t \cdot \frac{\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t)}{m}$$

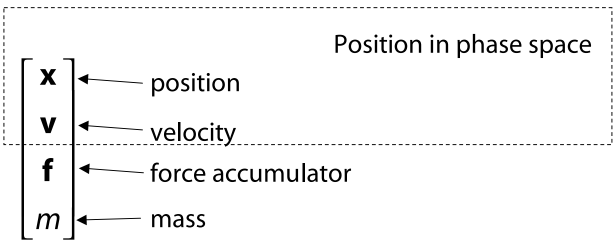
Writing this as an iteration, we have:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta t \cdot \mathbf{v}_i$$
$$\mathbf{v}_{i+1} = \mathbf{v}_i + \Delta t \cdot \frac{\mathbf{f}_i}{m}$$

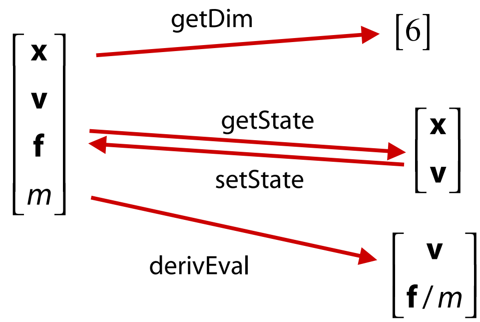
Again, performs poorly for large Δt.

Particle structure

How do we represent a particle?

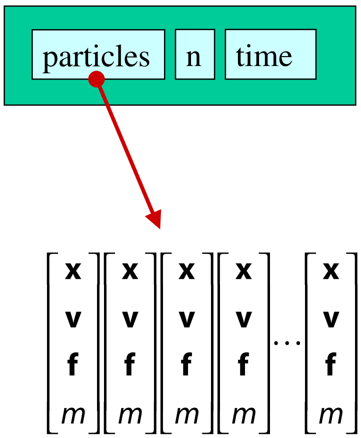


Single particle solver interface



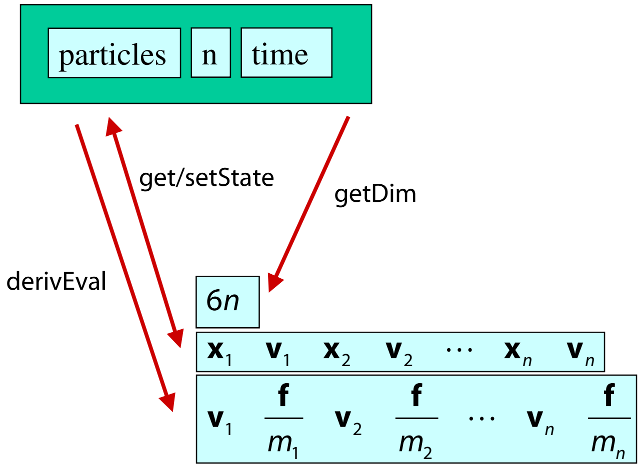
Particle systems

In general, we have a particle system consisting of *n* particles to be managed over time:



Particle system solver interface

For *n* particles, the solver interface now looks like:



Particle system diff. eq. solver

Thus, we start with:

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{x}_n \\ \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{f}_1 / m_1 \\ \vdots \\ \mathbf{v}_n \\ \mathbf{f}_n / m_n \end{bmatrix}$$

And can solve, using the Euler method:

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

Forces

Each particle can experience a force which sends it on its merry way.

Where do these forces come from? Some examples:

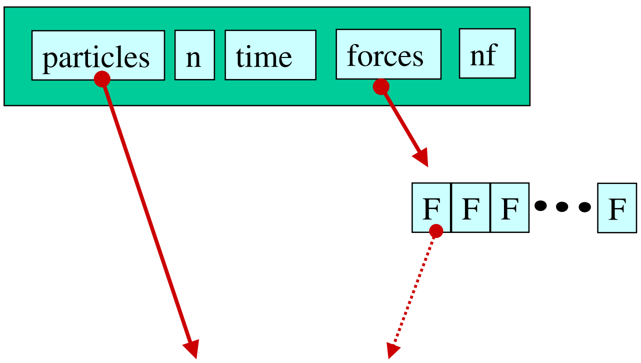
- Constant (gravity)
- Position/time dependent (force fields)
- Velocity-dependent (drag)
- N-ary (springs)

How do we compute the net force on a particle?

Particle systems with forces

Force objects are black boxes that point to the particles they influence and add in their contributions.

We can now visualize the particle system with force objects:



$$\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{f} \\ m \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{f} \\ m \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{f} \\ m \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{f} \\ m \end{bmatrix} \cdots \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{f} \\ m \end{bmatrix}$$

Gravity and viscous drag

The force due to **gravity** is simply:

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

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

Often, we want to slow things down with **viscous drag**:

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

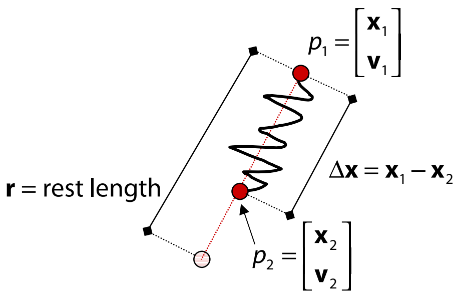
```
p->f -= F->k * p->v
```

Damped spring

A spring is a simple examples of an "N-ary" force.

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

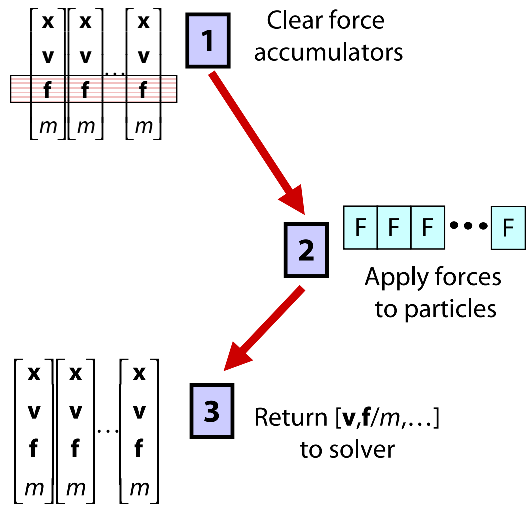
$$\mathbf{f}_2 = -\mathbf{f}_1$$



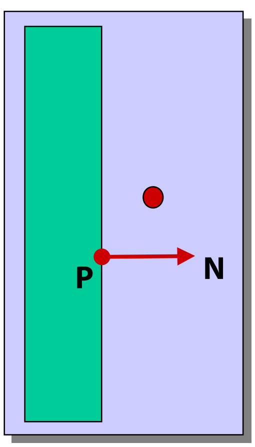
$$\Delta\mathbf{v} = \mathbf{v}_1 - \mathbf{v}_2$$

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



Bouncing off the walls

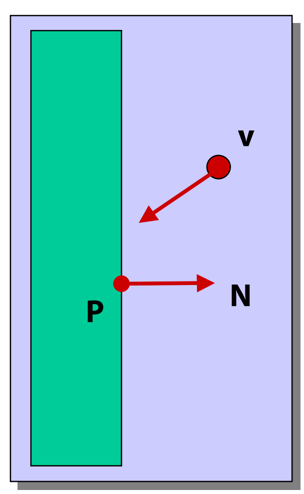


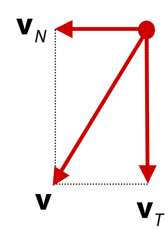
- Add-on for a particle simulator
- For now, just simple point-plane collisions

A plane is fully specified by any point **P** on the plane and its normal **N**.

Normal and tangential components

First we need to consider the normal and tangential components of a particle's velocity.



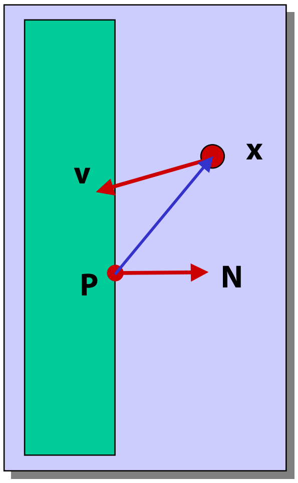


$$\mathbf{v}_N = (\mathbf{N}\cdot\mathbf{v})\mathbf{N}$$

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

Collision Detection

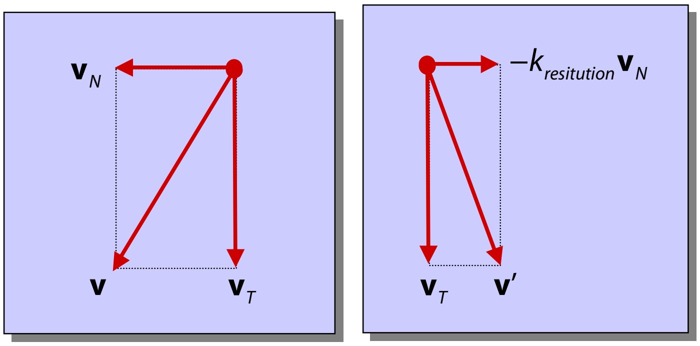
To detect collisions, we need the equation for a plane:



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

$\mathbf{N} \cdot \mathbf{v} < 0$ heading in

Collision Response



before after

$$\mathbf{v}' = \mathbf{v}_T - k_{restitution}\mathbf{v}_N$$

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