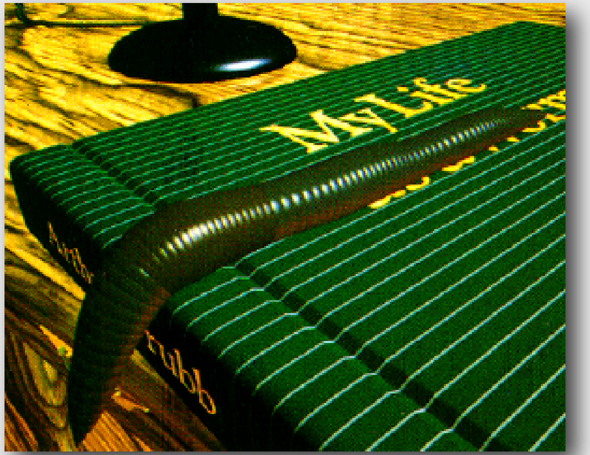
Particle dynamics





- Particle overview
- Second order motion
- Particle system
- Forces
- Constraints

Particle system

- Particles are objects that have mass, position, and velocity, but without spatial extent
- Particles are the easiest objects to simulate but they can be made to exhibit a wide range of objects

Particle basics

- Each particle has a position, mass, and velocity
  - maybe color, age, temperature
- Seeded randomly at start
  - maybe some created each frame
- Move each frame according to physics
- Eventually die when some condition met

Sparks from a campfire

- Add 2-3 particles at each frame
  - initialize position and temperature randomly
- Move in specified turbulent smoke flow and decrease temperature as evolving
- Render as a glowing dot
- Kill when too cold to glow visibly

Rendering

- Simplest rendering: color dots
- Animated sprites
- Deformable blobs
- Transparent spheres
- Shadows

A Newtonian particle

- First order motion is sufficient, if
  - a particle state only contains position
  - no inertia
  - particles are extremely light
- Most likely particles have inertia and are affected by gravity and other forces
- This puts us in the realm of second order motion

- Particle overview
- Second order motion
- Particle system
- Forces
- Constraints

Second-order ODE

What is the differential equation that describes the behavior of a mass point?

$$\mathbf{f} = m\mathbf{a}$$

What does $\mathbf{f}$ depend on?

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

Second-order ODE

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

This is not a first oder ODE because it has second derivatives

Add a new variable, $\mathbf{v}(t)$, to get a pair of coupled first order equations

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

Phase space

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Concatenate position and velocity to form a 6-vector: *position* in phase space

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

First order differential equation: *velocity* in the phase space

Linear analysis

- Linearly approximate acceleration

$$\mathbf{a}(\mathbf{x}, \mathbf{v}) \approx \mathbf{a}_0 - \mathbf{K}\mathbf{x} - \mathbf{D}\mathbf{v}$$

- Split up analysis into different cases
  - constant acceleration
  - linear acceleration

Constant acceleration

- Solution is

$$\mathbf{v}(t) = \mathbf{v}_0 + \mathbf{a}_0 t$$

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{a}_0 t^2$$

- $\mathbf{v}(t)$ only needs 1st order accuracy, but $\mathbf{x}(t)$ demands 2nd order accuracy

Linear acceleration

- Dependence on x and v dominates

$$\mathbf{a}(\mathbf{x}, \mathbf{v}) = -\mathbf{K}\mathbf{x} - \mathbf{D}\mathbf{v}$$

$$\frac{d}{dt}\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{K} & -\mathbf{D} \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} = \mathbf{A}\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$$

- Need to compute the eigenvalues of A

Linear acceleration

Assume α is an eigenvalue of $\mathbf{A}$, $\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}$ is the corresponding eigenvector

$$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{K} & -\mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \alpha \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}$$

The eigenvector of $\mathbf{A}$ has the form $\begin{bmatrix} \mathbf{u} \\ \alpha\mathbf{u} \end{bmatrix}$

Often, D is linear combination of K and I (Rayleigh damping)

That means K and D have the same eigenvectors

Linear acceleration

Assume **u** is an eigenvector for both **K** and **D**

If $\begin{bmatrix} \mathbf{u} \\ \alpha\mathbf{u} \end{bmatrix}$ is an eigenvector of A, following must be true

$$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{K} & -\mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \alpha\mathbf{u} \end{bmatrix} = \alpha \begin{bmatrix} \mathbf{u} \\ \alpha\mathbf{u} \end{bmatrix}$$

$$-\lambda_k \mathbf{u} - \alpha\lambda_d \mathbf{u} = \alpha^2 \mathbf{u}$$

$$\alpha = -\frac{1}{2}\lambda_d \pm \sqrt{(\frac{1}{2}\lambda_d)^2 - \lambda_k}$$

Eigenvalue approximation

- If **D** dominates

$$\alpha \approx -\lambda_d, 0$$

  - exponential decay

- If **K** dominates

$$\alpha \approx \pm\sqrt{-1}\sqrt{\lambda_k}$$

  - oscillation

Analysis

- Constant acceleration (e.g. gravity)
  - demands 2nd order accuracy for position
- Position dependence (e.g. spring force)
  - demands stability but low or zero damping
  - looks at imaginary axis
- Velocity dependence (e.g. damping)
  - demands stability, exponential decay
  - Looks at negative real axis

Constant acceleration

- Solution is

$$\mathbf{v}(t) = \mathbf{v}_0 + \mathbf{a}_0 t$$

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{a}_0 t^2$$

- $\mathbf{v}(t)$ only needs 1st order accuracy, but $\mathbf{x}(t)$ demands 2nd order accuracy

Explicit methods

- First-order Euler method
  - constant acceleration: bad (1st order)
  - position dependence: very bad (unstable)
  - velocity dependence: ok (conditionally stable)
- RK3 and RK4
  - constant acceleration: great (high order)
  - position dependence: ok (conditionally stable)
  - velocity dependence: ok (conditionally stable)

Implicit methods

- Implicit Euler method
  - constant acceleration: bad (1st order)
  - position dependence: ok (stable but damped)
  - velocity dependence: great (monotone)
- Trapezoidal rule
  - constant acceleration: great (2nd order)
  - position dependence: great (stable and no damp)
  - velocity dependence: good (stable, not monotone)

- Particle overview
- Second order motion
- Particle system
- Forces
- Constraints

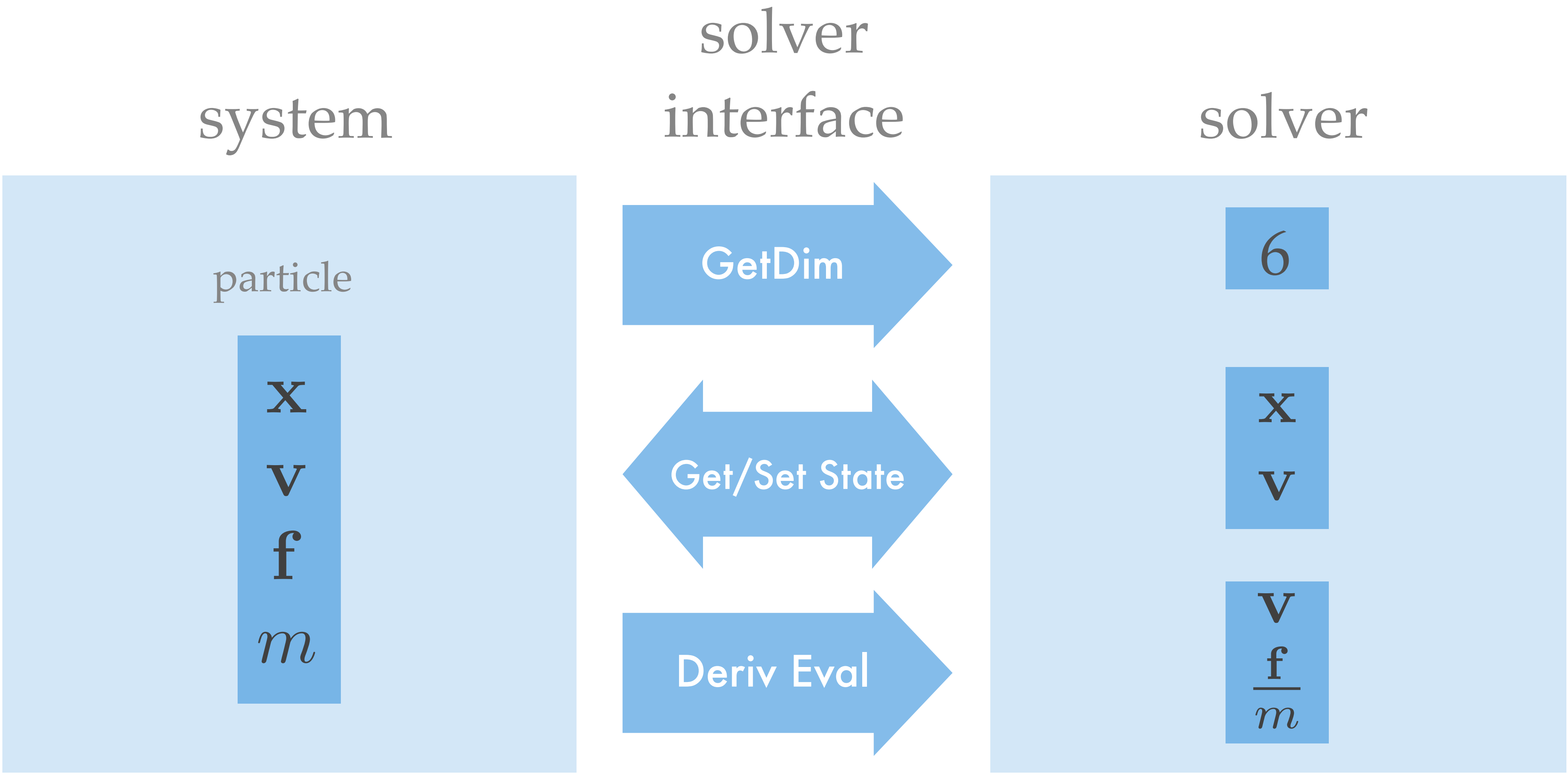
Particle structure

Particle

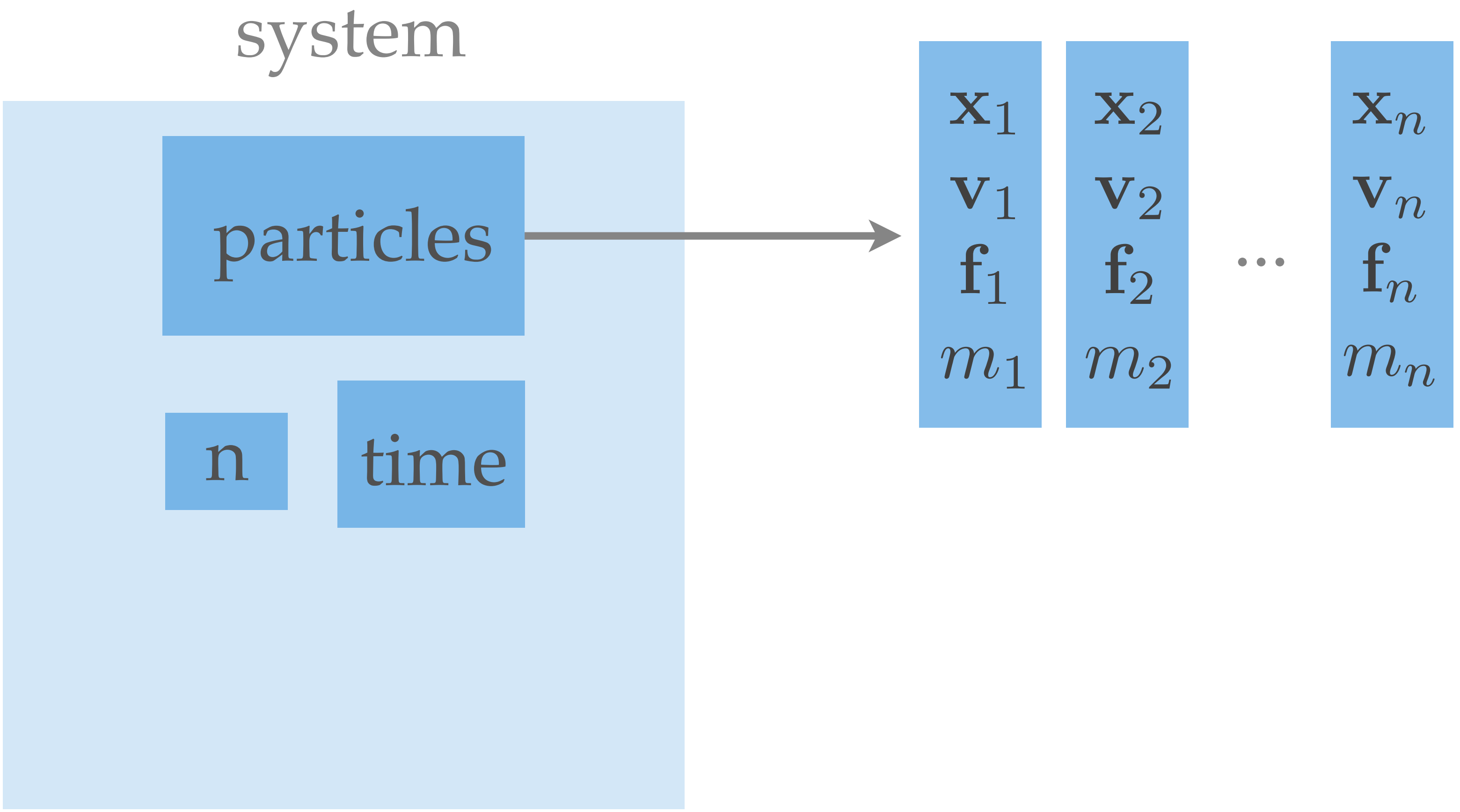
$\mathbf{x}$	position
$\mathbf{v}$	velocity
$\mathbf{f}$	force accumulator
m	mass

$\mathbf{x}$, $\mathbf{v}$ → a point in the phase space

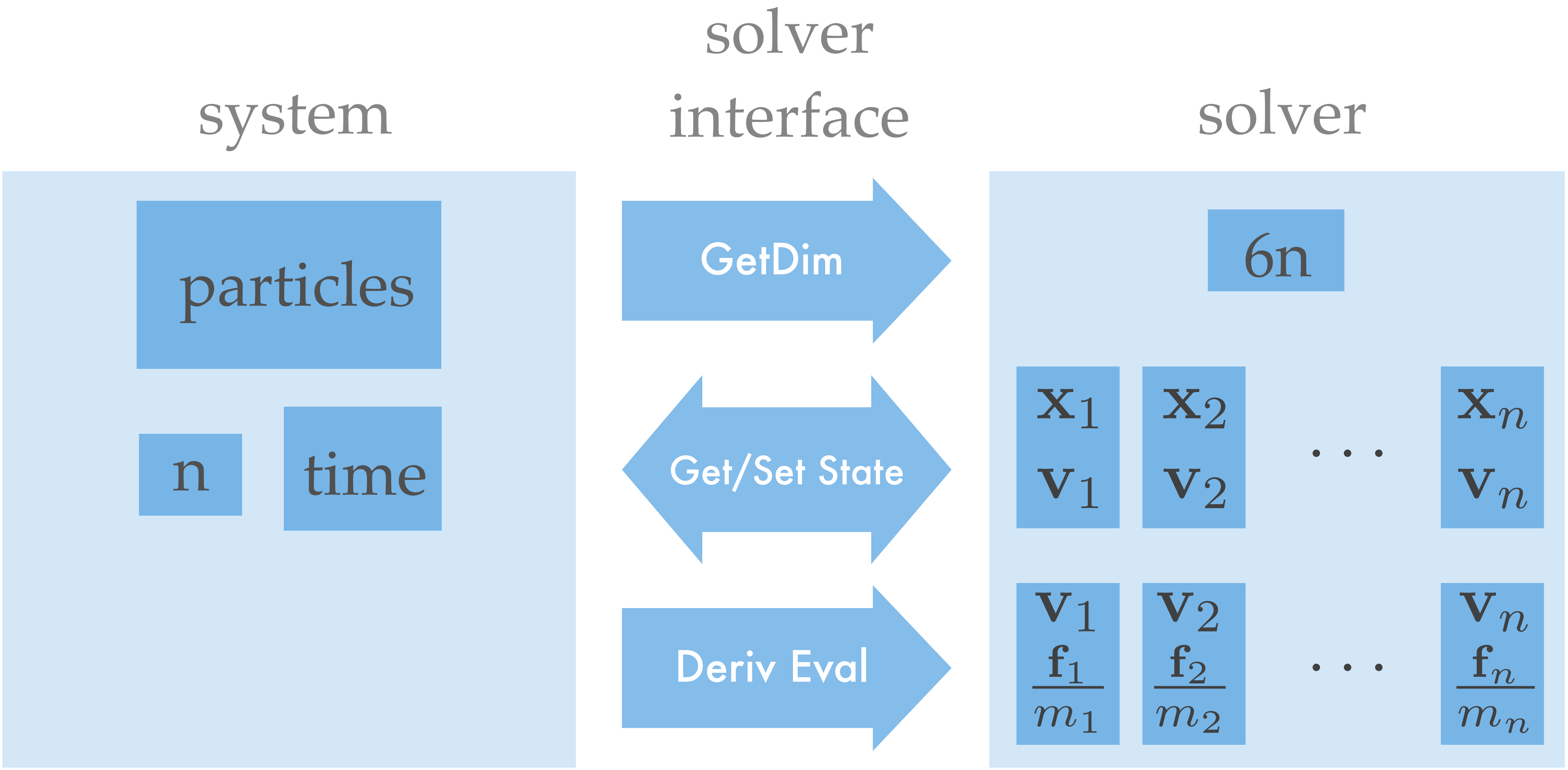
Solver interface



Particle system structure



Particle system structure



Deriv Eval

Clear forces: loop over particles, zero force accumulator

Calculate forces: sum all forces into accumulator

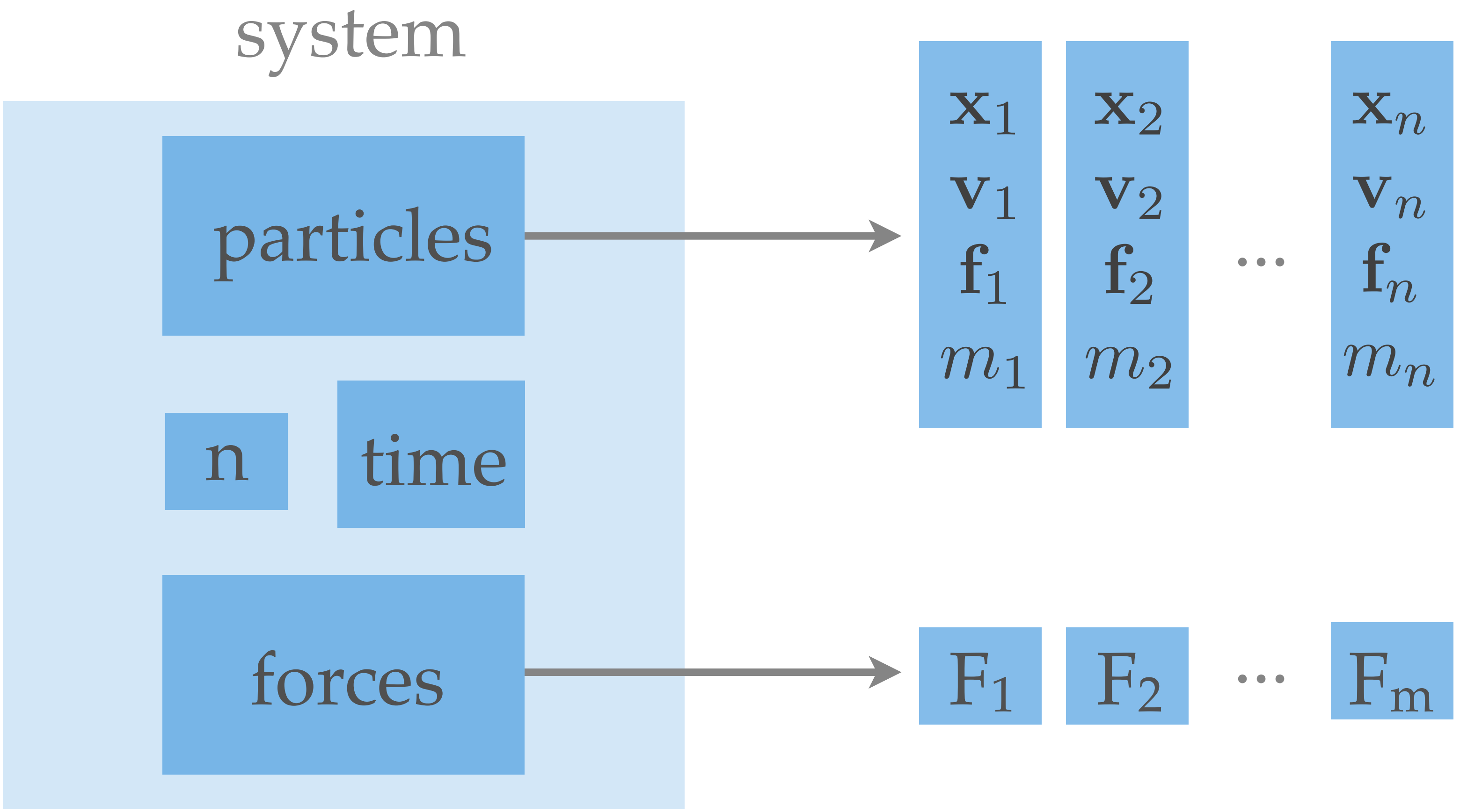
Gather: loop over particles, copy **v** and **f**/m into destination array

- Particle overview
- Second order motion
- Particle system
- Forces
- Constraints

Forces

- Constant
  - gravity
- Position/time dependent
  - force fields, springs
- Velocity dependent
  - drag

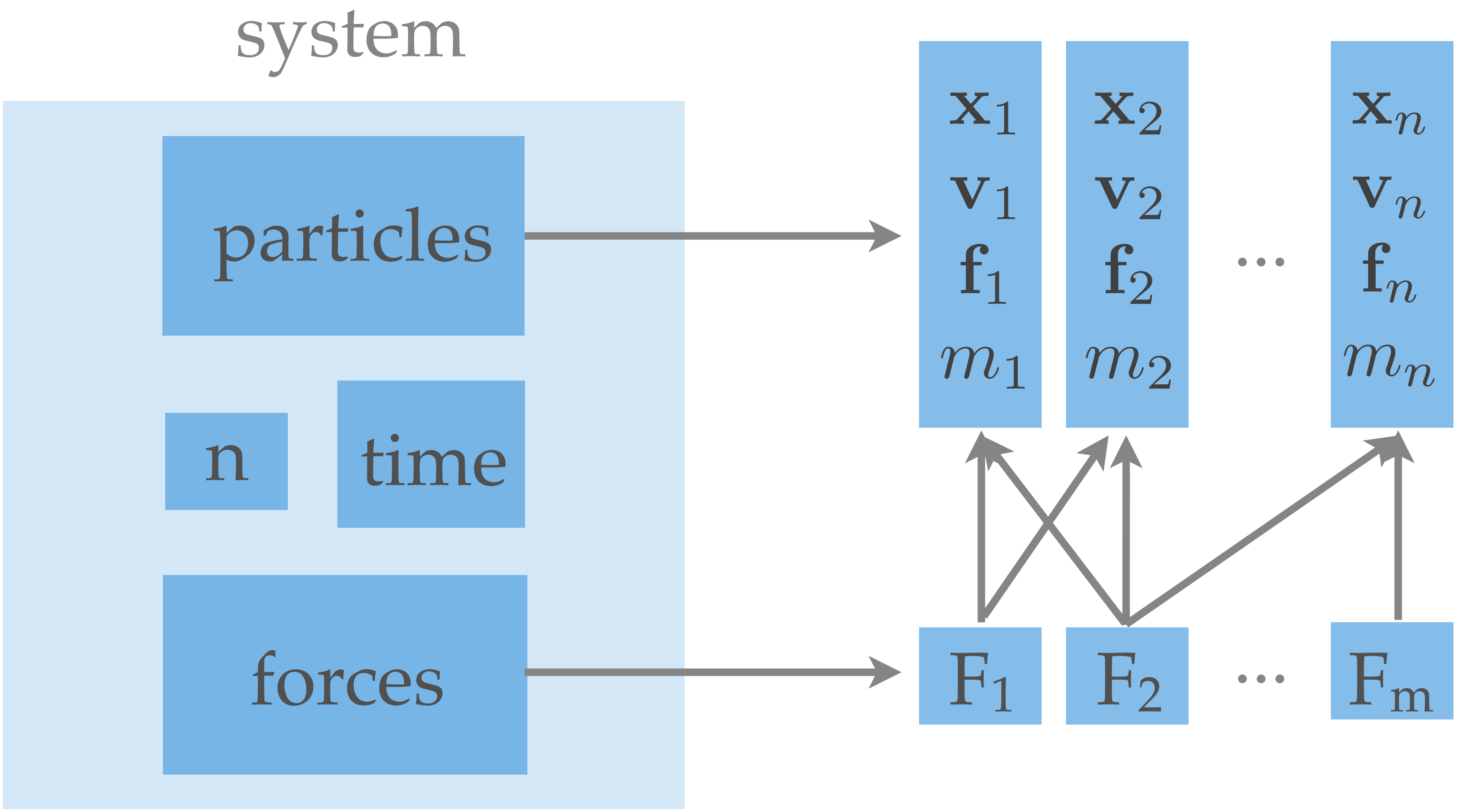
Particle systems with forces



Force structure

- Unlike particles, forces are heterogeneous (type-dependent)
- Each force object "knows"
  - which particles it influences
  - how much contribution it adds to the force accumulator

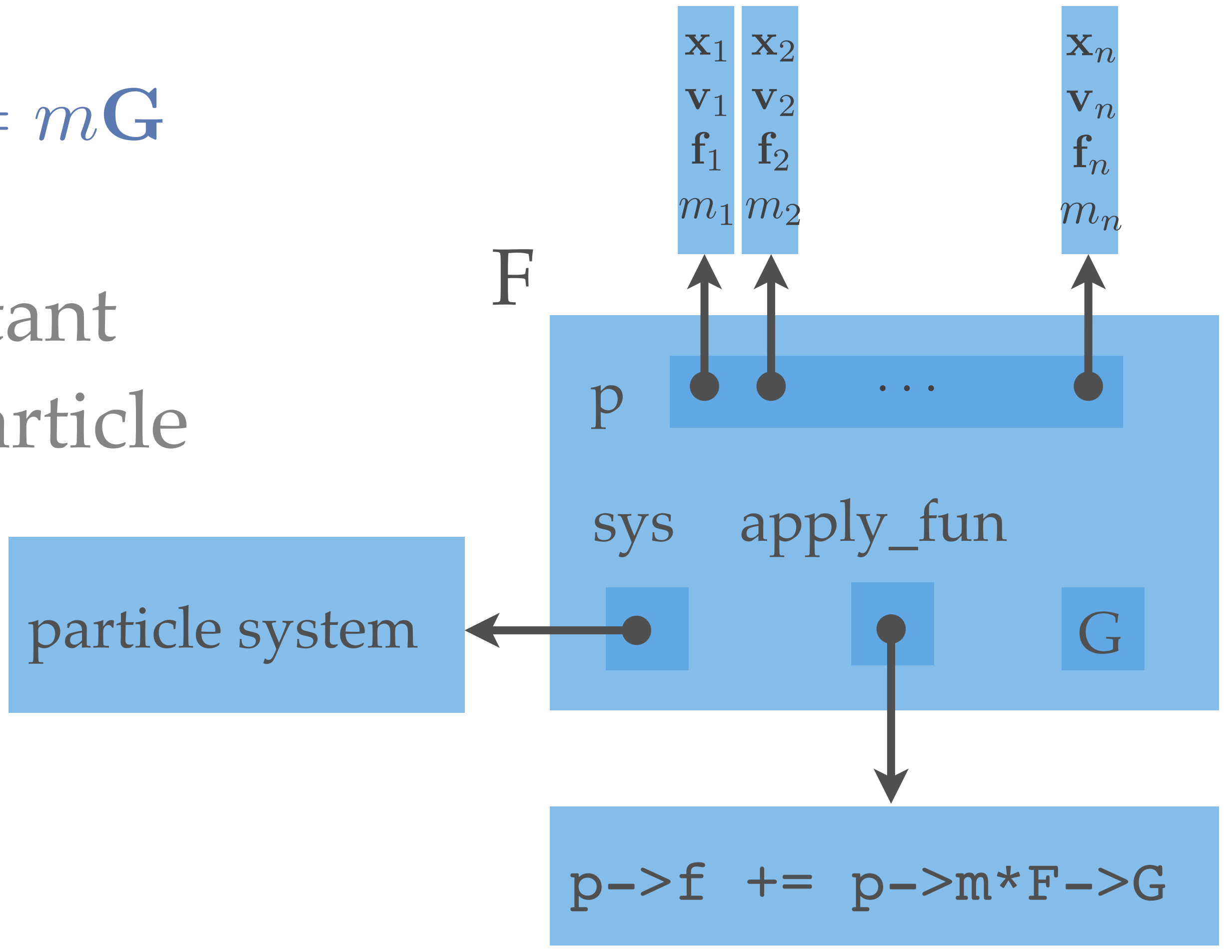
Particle systems with forces



Gravity

Unary force: $\mathbf{f} = m\mathbf{G}$

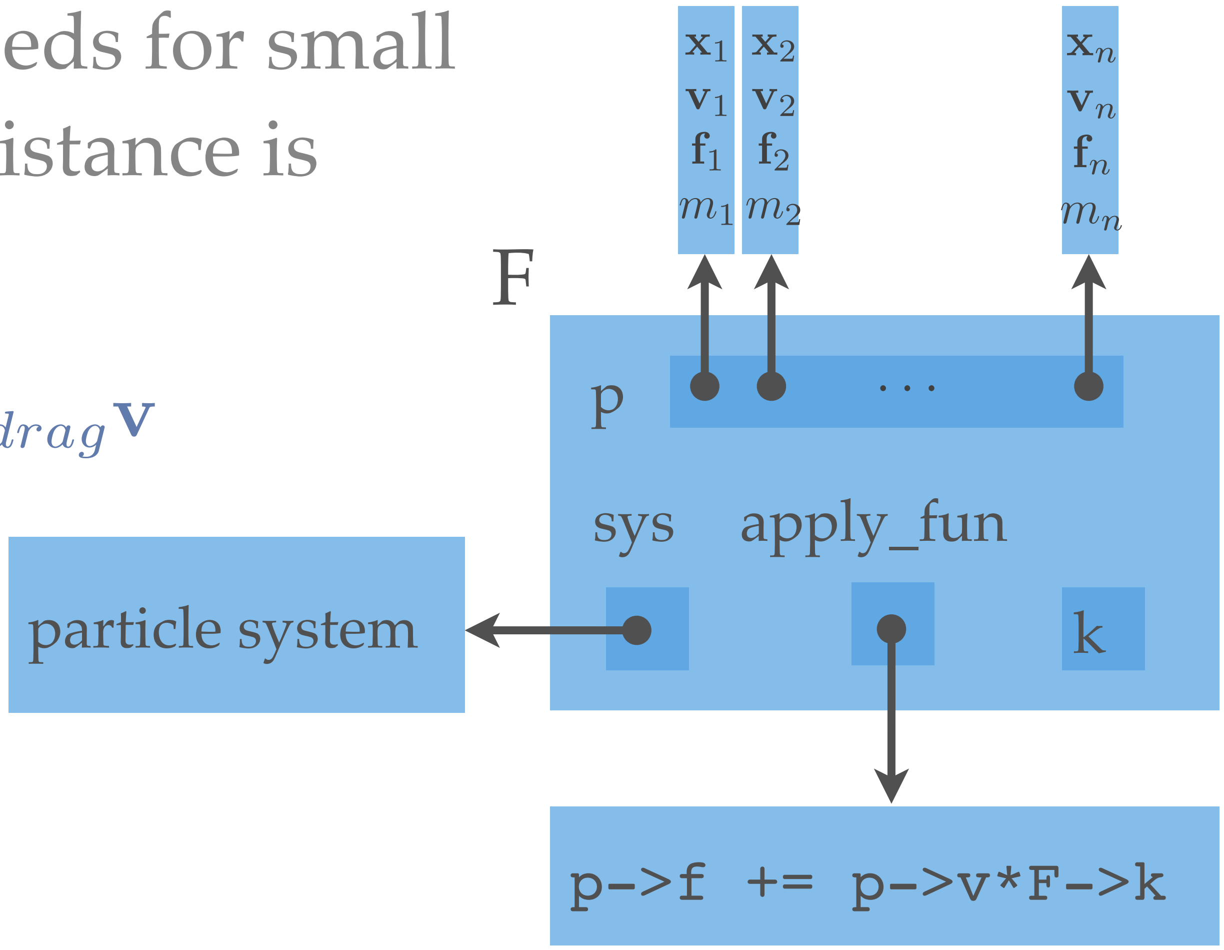
Exerting a constant force on each particle



Viscous drag

At very low speeds for small particles, air resistance is approximately:

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



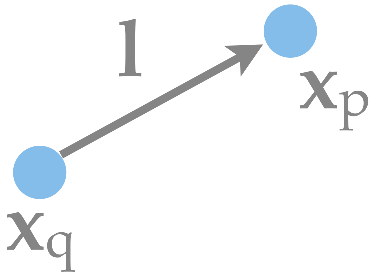
Attraction

Act on any or all pairs of particles, depending on their positions

$$\mathbf{f}_p = -k \frac{m_p m_q}{|\mathbf{l}|^2} \frac{\mathbf{l}}{|\mathbf{l}|}$$

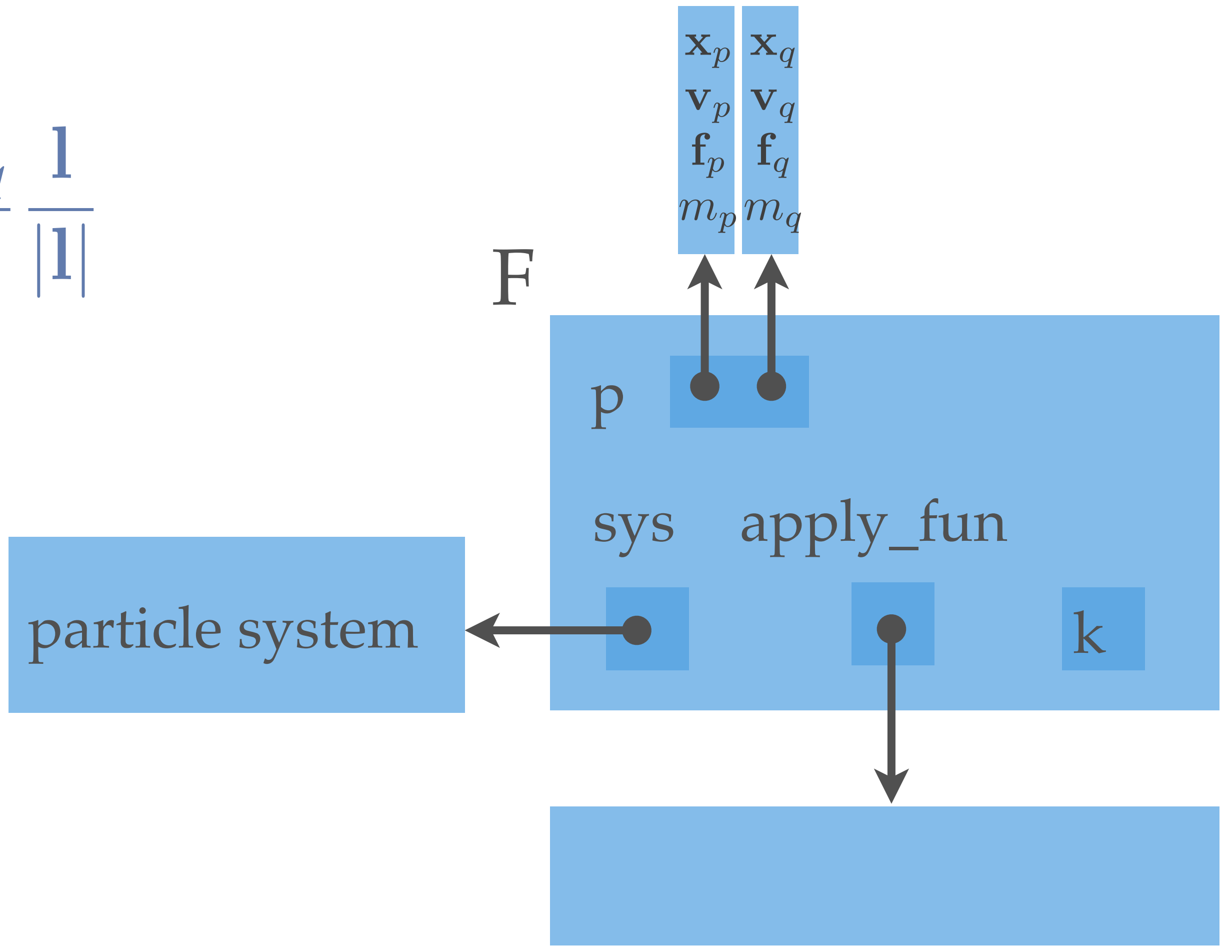
$$\mathbf{f}_q = -\mathbf{f}_p$$

$$\mathbf{l} = \mathbf{x}_p - \mathbf{x}_q$$



Attraction

$$\mathbf{f}_p = -k \frac{m_p m_q}{|\mathbf{l}|^2} \frac{\mathbf{l}}{|\mathbf{l}|}$$

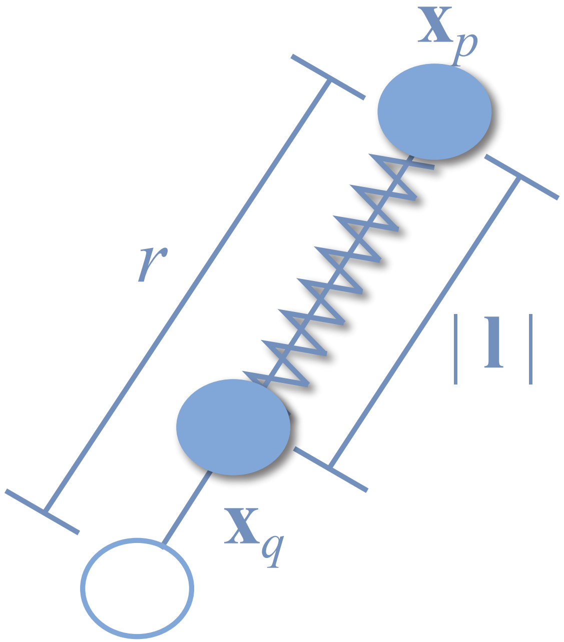


Damped spring

$$\mathbf{f}_p = -\left[k_s(|\mathbf{l}| - r) + k_d \frac{\dot{\mathbf{l}} \cdot \mathbf{l}}{|\mathbf{l}|}\right] \frac{\mathbf{l}}{|\mathbf{l}|}$$

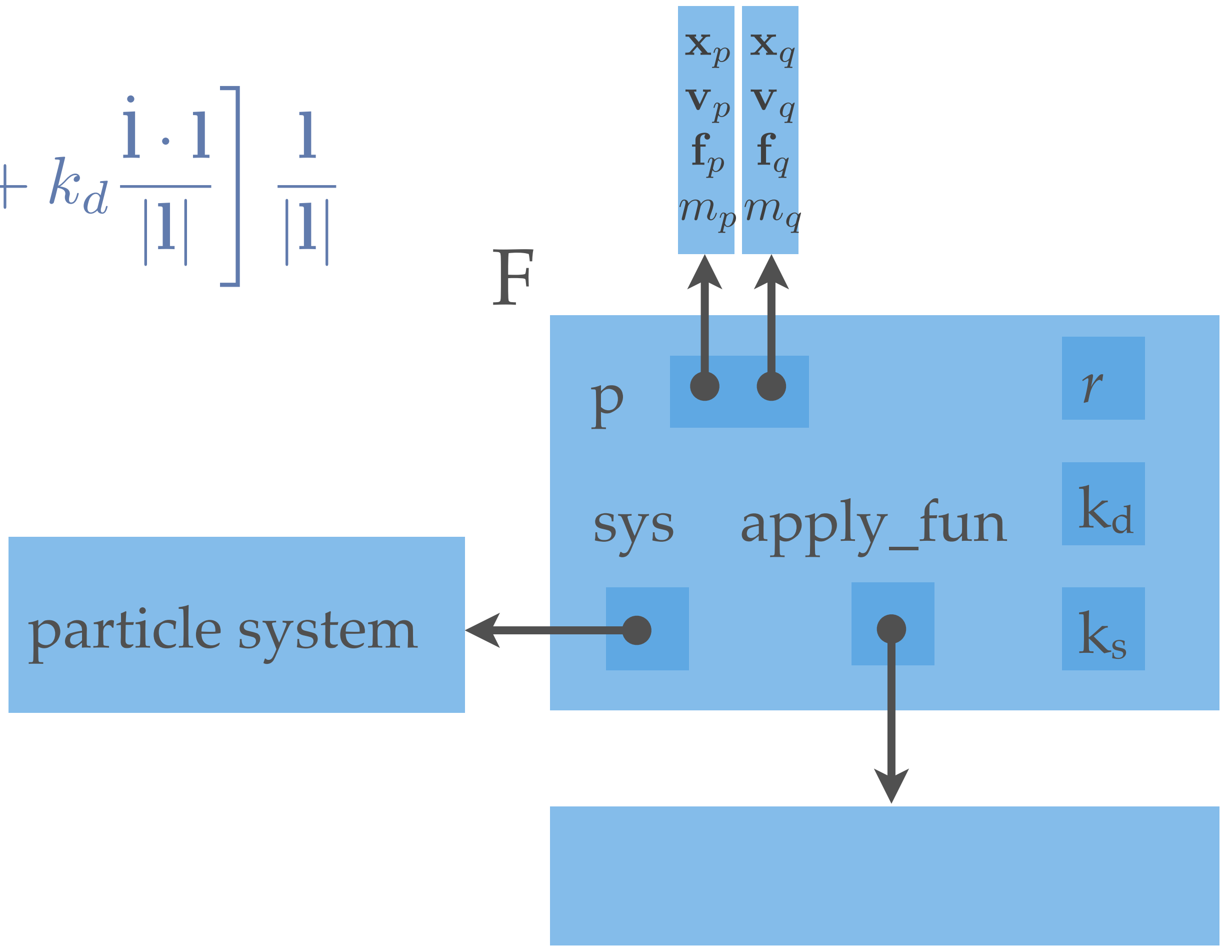
$$\mathbf{f}_q = -\mathbf{f}_p$$

$$\mathbf{l} = \mathbf{x}_p - \mathbf{x}_q$$



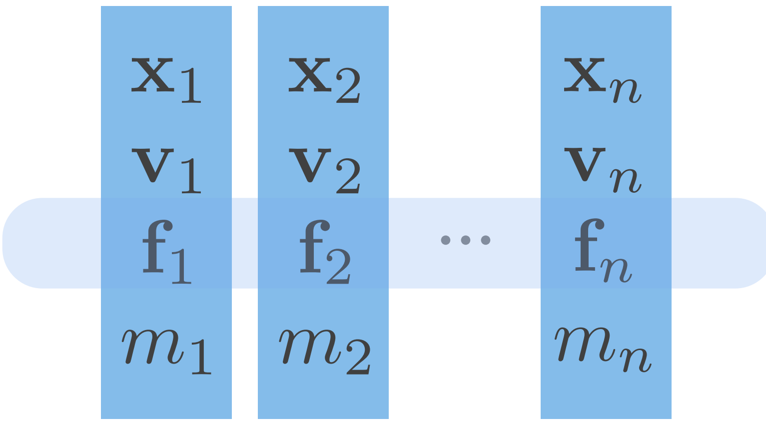
Damped spring

$$\mathbf{f}_p = -\left[k_s(|\mathbf{l}| - r) + k_d \frac{\dot{\mathbf{l}} \cdot \mathbf{l}}{|\mathbf{l}|}\right] \frac{\mathbf{l}}{|\mathbf{l}|}$$

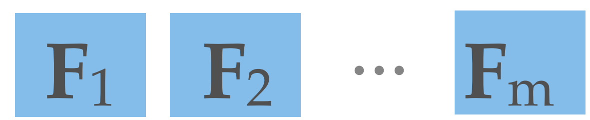


Deriv Eval

1. Clear force accumulators



2. Invoke `apply_force` functions



3. Return derivatives to solver

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

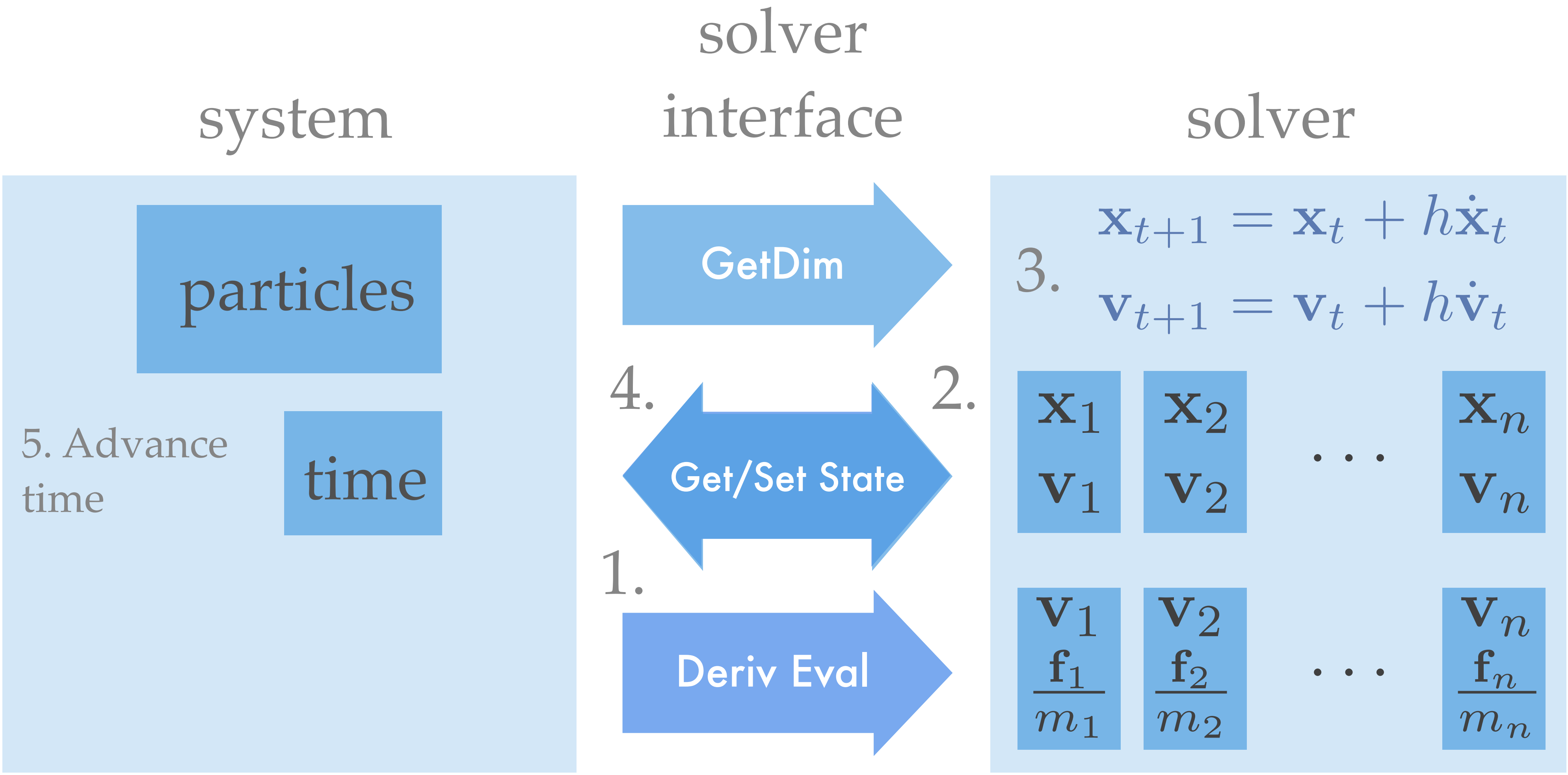
ODE solver

Euler's method: $\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + hf(\mathbf{x}, t)$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + h\dot{\mathbf{x}}_t$$

$$\mathbf{v}_{t+1} = \mathbf{v}_t + h\dot{\mathbf{v}}_t$$

Euler step

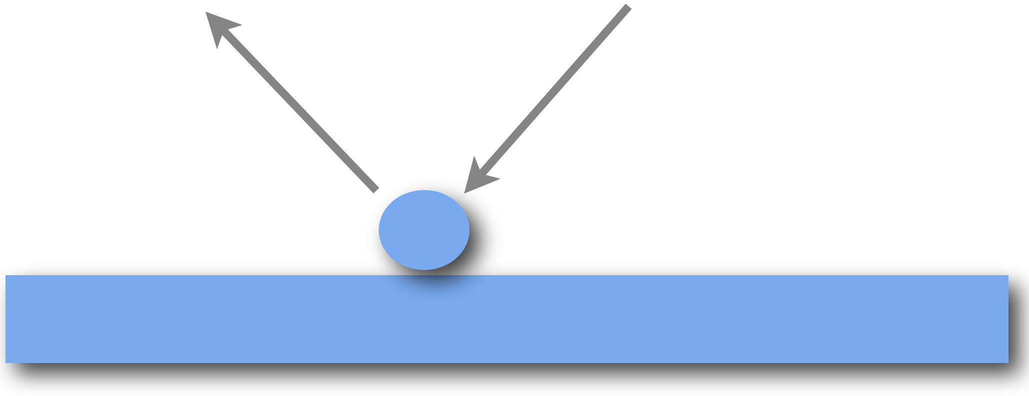


system
solver interface
solver
particles
time
5. Advance time
GetDim
Get/Set State
Deriv Eval
1.
2.
3.
4.
$\mathbf{x}_{t+1} = \mathbf{x}_t + h\dot{\mathbf{x}}_t$
$\mathbf{v}_{t+1} = \mathbf{v}_t + h\dot{\mathbf{v}}_t$
$\mathbf{x}_1$ $\mathbf{v}_1$
$\mathbf{x}_2$ $\mathbf{v}_2$
$\cdots$
$\mathbf{x}_n$ $\mathbf{v}_n$
$\mathbf{v}_1$ $\frac{\mathbf{f}_1}{m_1}$
$\mathbf{v}_2$ $\frac{\mathbf{f}_2}{m_2}$
$\cdots$
$\mathbf{v}_n$ $\frac{\mathbf{f}_n}{m_n}$

- Particle overview
- Second order motion
- Particle system
- Forces
- Constraints

Particle Interaction

- We will revisit collision when we talk about rigid body simulation
- For now, just simple point-plane collisions

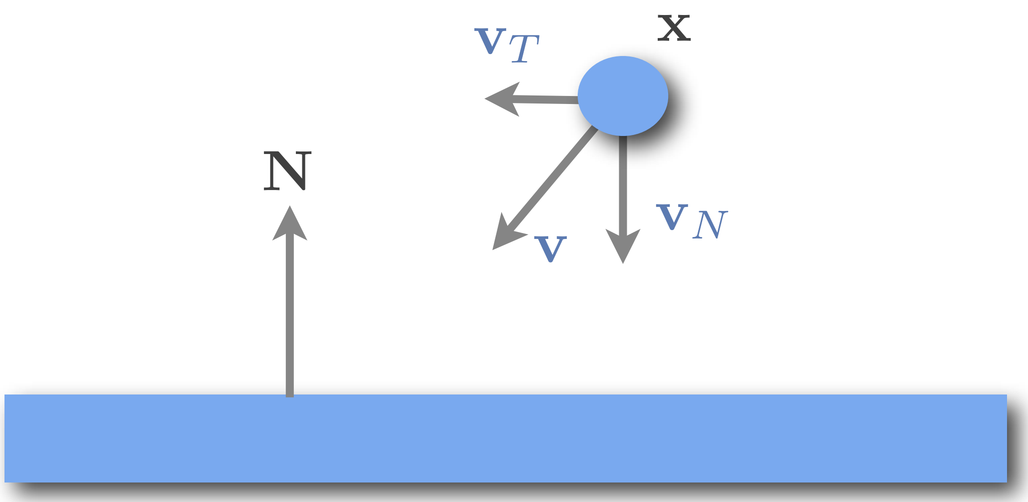


Collision detection

Normal and tangential components

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

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



Collision detection

Particle is on the legal side if

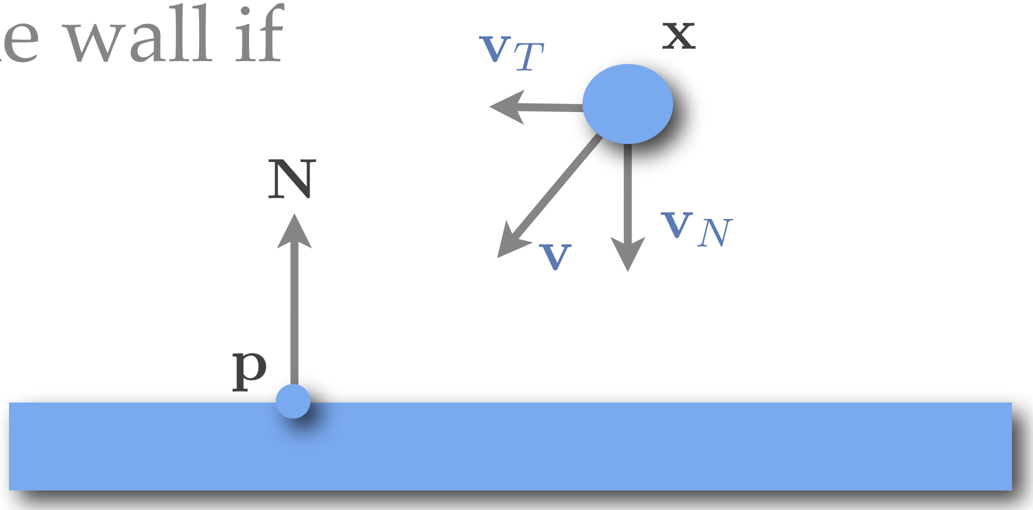
$$(\mathbf{x} - \mathbf{p}) \cdot \mathbf{N} \geq 0$$

Particle is within ϵ of the wall if

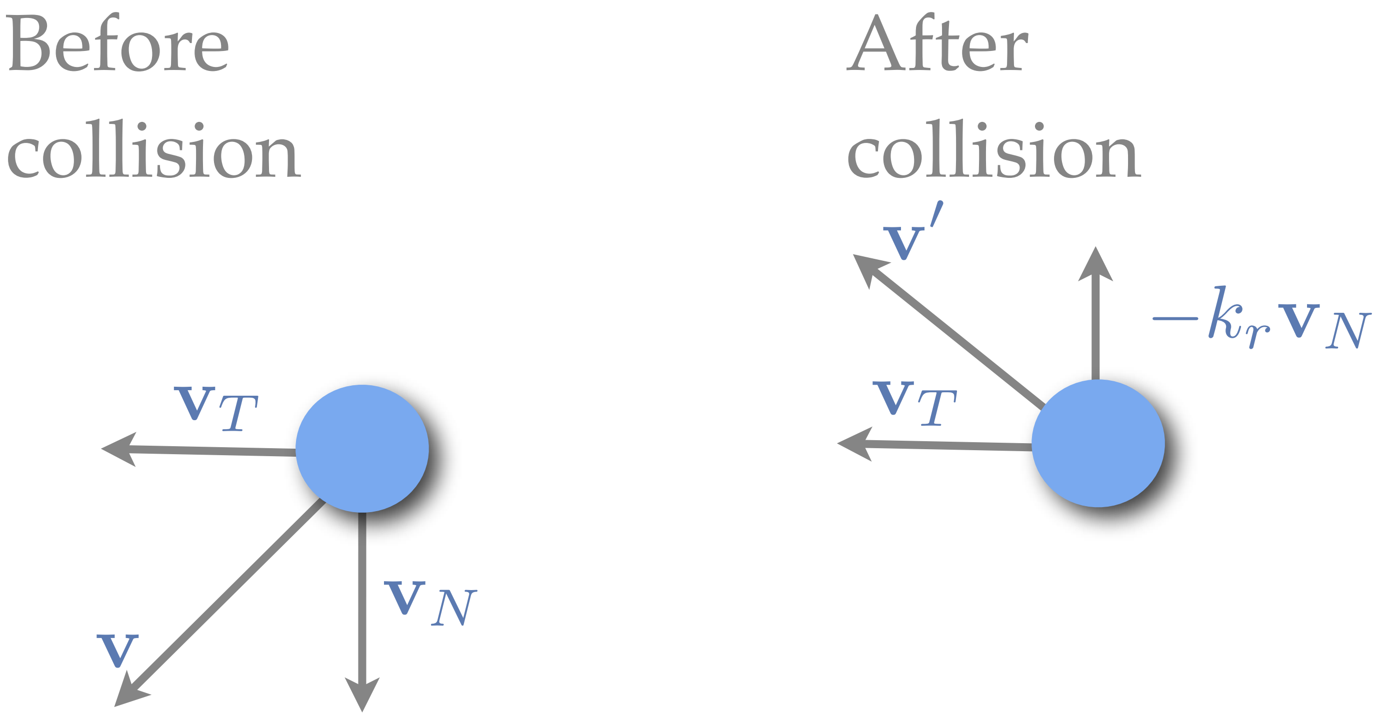
$$(\mathbf{x} - \mathbf{p}) \cdot \mathbf{N} < \epsilon$$

Particle is heading in if

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



Collision response



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

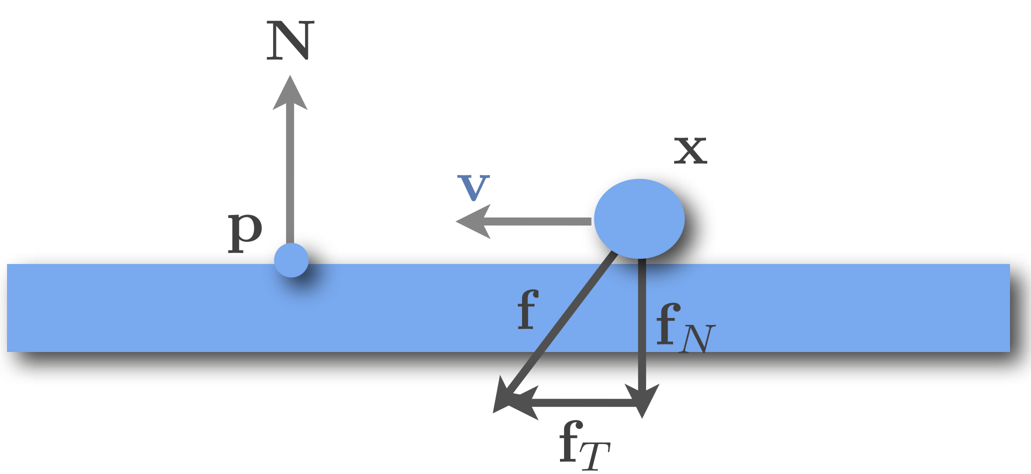
coefficient of restitution: $0 \leq k_r < 1$

Contact

Conditions for resting contact:

1. particle is on the collision surface
2. zero normal velocity

If a particle is pushed into the contact plane a contact force $\mathbf{f}_c$ is exerted to cancel the normal component of $\mathbf{f}$



What’s next?

- How do we enforce constraints on the particles?
- Read (optional): Particle animation and rendering using data parallel computation, SIG90, Karl Sims