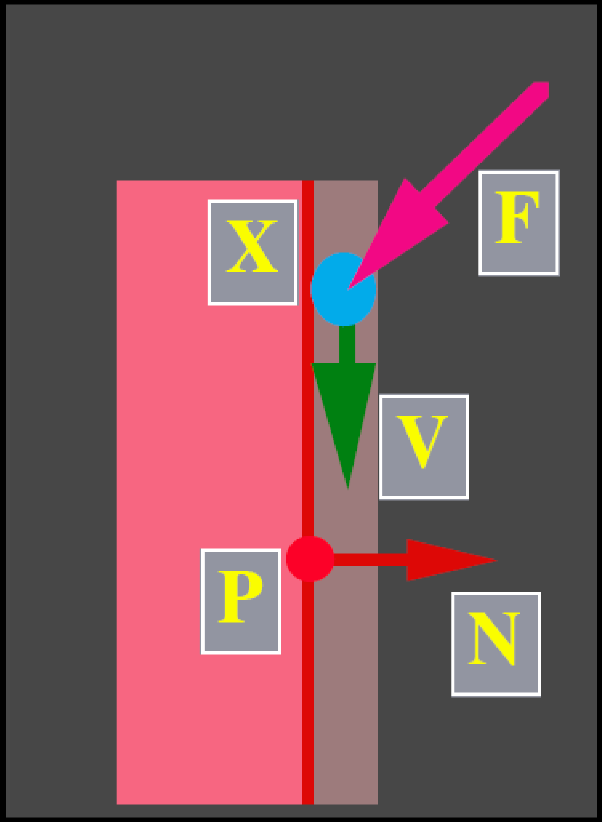
Particle System



Particles

- Particles are objects modeled as point masses
- Particle properties:
  - mass
  - position
  - velocity
  - force accumulator
  - age, lifespan
  - rendering properties
  - etc.

See also a nice introduction to particle systems with code by Daniel Shiffiman:
https://youtu.be/vdgiqMkFygc
https://natureofcode.com/

Particles (cont.)

- Particles respond to forces
- We represent this using differential equations

$$\ddot{\mathbf{x}} = \frac{\mathbf{f}}{m}$$

2nd order ODE

phase space

$$\dot{\mathbf{x}} = \mathbf{v}$$

$$\dot{\mathbf{v}} = \frac{\mathbf{f}}{m}$$

1st order ODEs

Particle Systems

- Particle systems are collections of particles
- Particle systems can represent:
  - fire
  - smoke/clouds
  - water
  - cloth
  - soft bodies
  - flocks/crowds

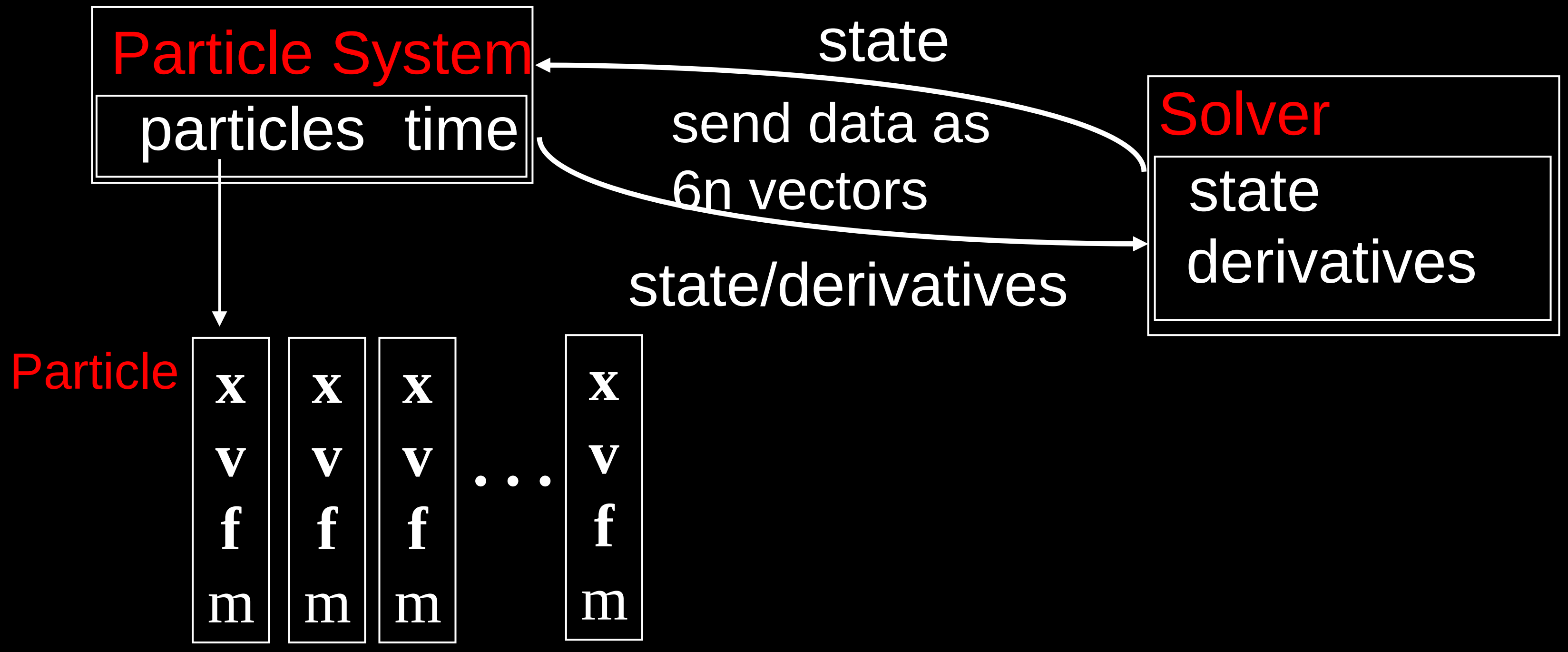
Example Videos

- Particle Dreams by Karl Sims (1988)
  https://youtu.be/t2an3xMuiew
- Pipe Dream by Wayne Lytle (Animusic,2001)

https://youtu.be/kTFBK7TltlE?t=13

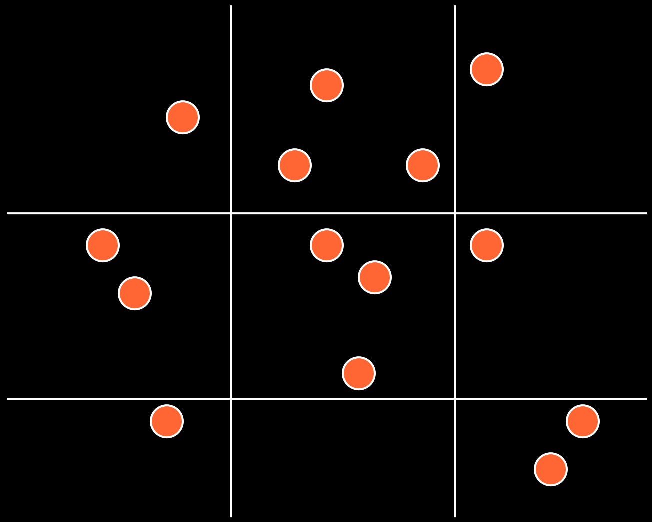
Particle Systems

- Separate the data structures and integration



Spatial Forces

- Forces that depend on nearby particles within a local region

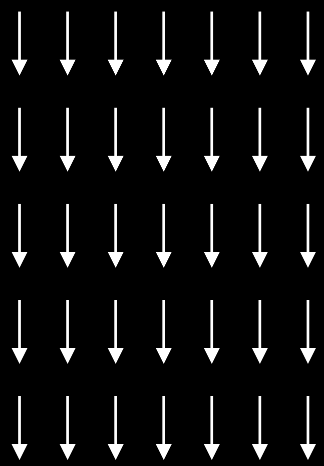


Gravity, Lennard-Jones and electric potentials

Spatial data structures can optimize computations

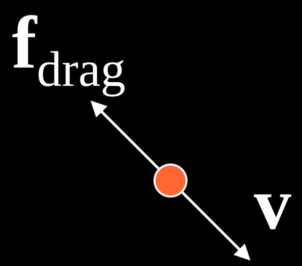
Unary Forces

- Forces that only depend on 1 particle



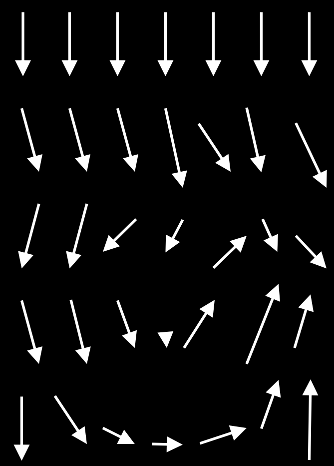
Gravity

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



Viscous Drag

$\mathbf{f} = -k_d\mathbf{v}$

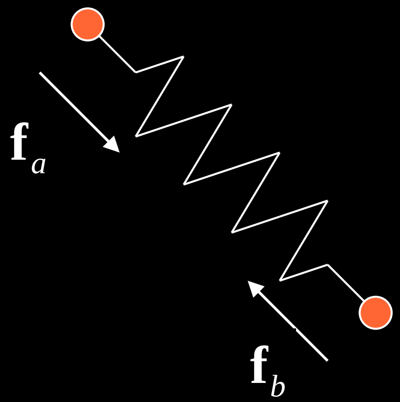


Wind Fields

$\mathbf{f} = k\mathbf{v}_{\mathrm{wind}}$

n-ary Forces

- Forces that depend on *n* particles
- Example: binary forces—spring and damper



Springs

$$\mathbf{f}_a = -k_s(|\mathbf{x}_a - \mathbf{x}_b| - l_0)\frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|}$$

$$-k_d\left(\frac{(\mathbf{v}_a - \mathbf{v}_b)\cdot(\mathbf{x}_a - \mathbf{x}_b)}{|\mathbf{x}_a - \mathbf{x}_b|}\right)\frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|}$$

Spring Force

- If particle is located farther than the rest position, the spring force needs to pull it back

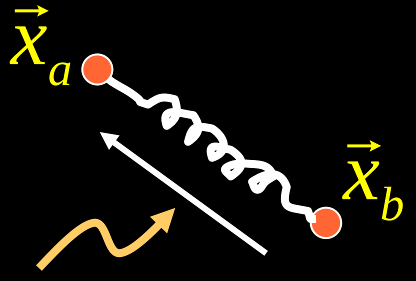
$$\Delta l = \left|\vec{x}_a - \vec{x}_b\right| - r > 0 \qquad \vec{f}_a = -(k\Delta l)\vec{l}, \quad k > 0$$

- If the particle is located nearer than the rest position, the spring force needs to push it away

$$\Delta l = \left|\vec{x}_a - \vec{x}_b\right| - r < 0 \qquad \vec{f}_a = -(k\Delta l)\vec{l}, \quad k > 0$$

- $\vec{f}_a = -k_s(\left|\vec{x}_a - \vec{x}_b\right| - r)\vec{l}, \quad k_s > 0$

$$= -k_s(\left|\vec{x}_a - \vec{x}_b\right| - r)\frac{\vec{x}_a - \vec{x}_b}{\left|\vec{x}_a - \vec{x}_b\right|}$$



$$\vec{l} = \frac{\vec{x}_a - \vec{x}_b}{\left|\vec{x}_a - \vec{x}_b\right|}$$

Damper Force

- If two particles are departing, the damper force needs to pull them back

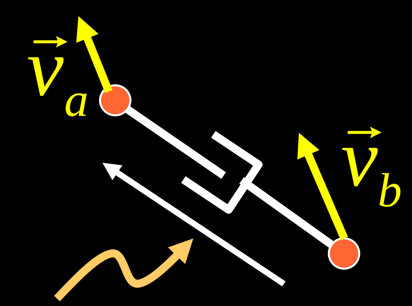
$$\Delta v = (\vec{v}_a - \vec{v}_b) \cdot \vec{l} > 0 \qquad \vec{f}_a = -(k\Delta v)\vec{l}, \quad k > 0$$

- If two particles are approaching, the damper force needs to push them away

$$\Delta v = (\vec{v}_a - \vec{v}_b) \cdot \vec{l} < 0 \qquad \vec{f}_a = -(k\Delta v)\vec{l}, \quad k > 0$$

- $$\vec{f}_a = -k_d((\vec{v}_a - \vec{v}_b) \cdot \vec{l})\vec{l}, \quad k_d > 0$$

$$= -k_d \frac{(\vec{v}_a - \vec{v}_b) \cdot (\vec{x}_a - \vec{x}_b)}{|\vec{x}_a - \vec{x}_b|} \frac{(\vec{x}_a - \vec{x}_b)}{|\vec{x}_a - \vec{x}_b|}$$



$$\vec{l} = \frac{\vec{x}_a - \vec{x}_b}{|\vec{x}_a - \vec{x}_b|}$$

Notes on Damping Forces

- According to the law of energy conservation, a particle system consists of only masses and springs keep bouncing from each other after external forces disappear

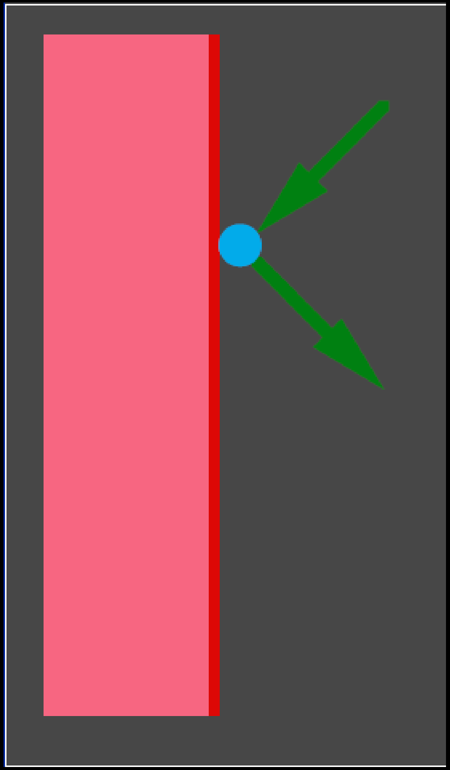
- Damping/viscous drag force resist motion, making a particle system gradually come to rest in the absence of external forces

Notes on Damping Forces (cont.)

- It is highly recommended that at least a small amount of damping is applied to each particle

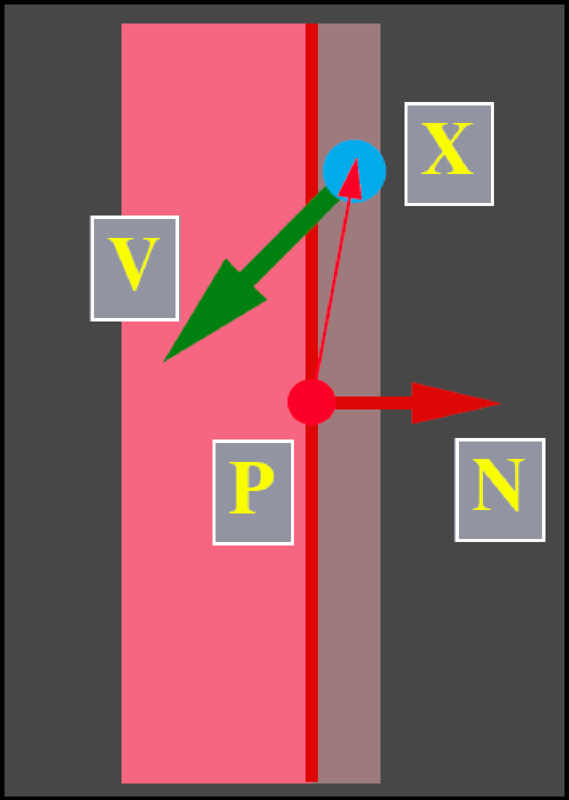
- Excessive damping, however, makes a particle appear that floating in molasses (energy dissipates out too quickly, not responsive)

Collision Detection and Response



- Later class: rigid body collision and contact
- For now, just simple point-plane collisions

Collision Detection



$$\|\mathbf{N}\| = 1$$

- Determine when a particle has collided
  - Close to the wall

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

  - Heading in

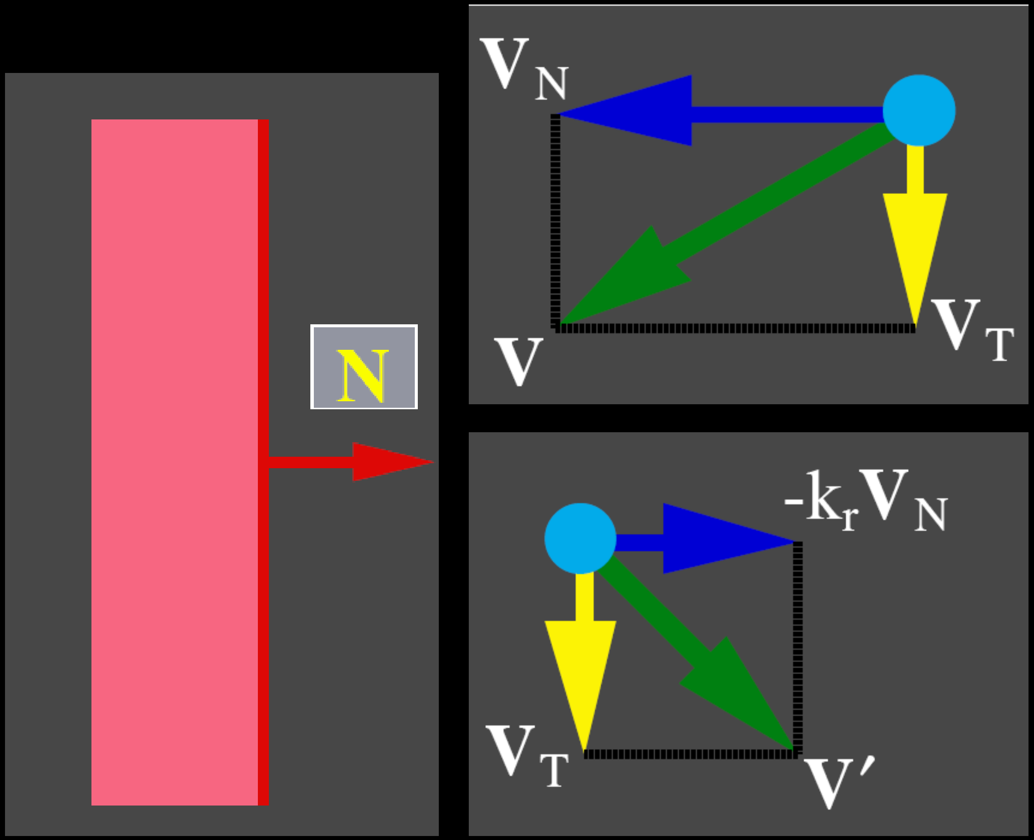
$$\mathbf{N} \cdot \mathbf{v} < 0$$

Collision Response

- What should we do when a particle has collided (or penetrated)?
- The correct thing to do is rollback the simulation to the exact point of contact
- A less accurate but easier alternative is to just modify positions and velocities

Collision Response

Assume no friction…



- Before

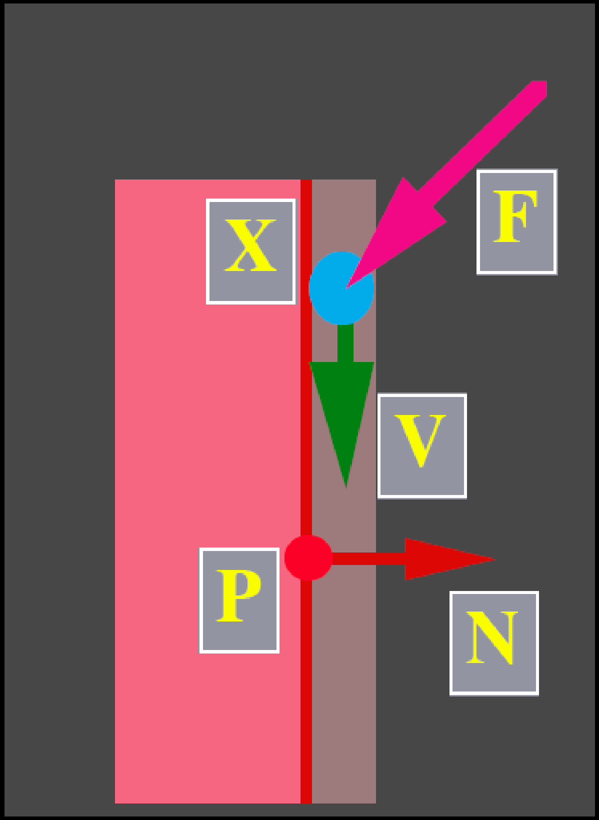
$$\mathbf{v} = \mathbf{v}_N + \mathbf{v}_T$$

- After

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

coefficient of restitution

Contact Conditions



$$\|\mathbf{N}\| = 1$$

- On the wall

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

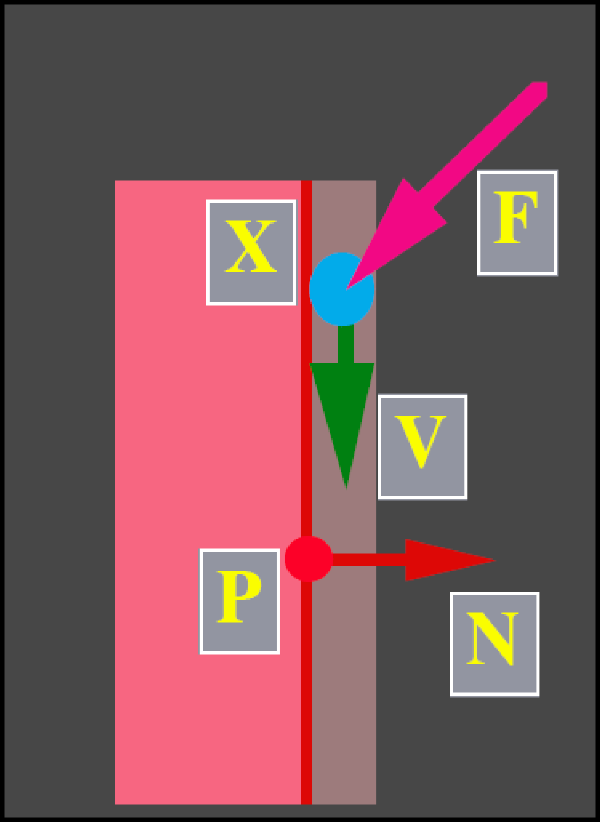
- Moving along the wall

$$|\mathbf{N} \cdot \mathbf{v}| < \varepsilon$$

- A force **f**, e.g., gravity, pushes the particle into the wall

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

Contact Forces



- When the particle is on the collision surface a contact force resists penetration

$$\mathbf{f}^{c} = -(\mathbf{N}\cdot\mathbf{f})\,\mathbf{N} \qquad \mathbf{N}\cdot\mathbf{f} < 0$$

- Contact forces do not resist leaving the surface

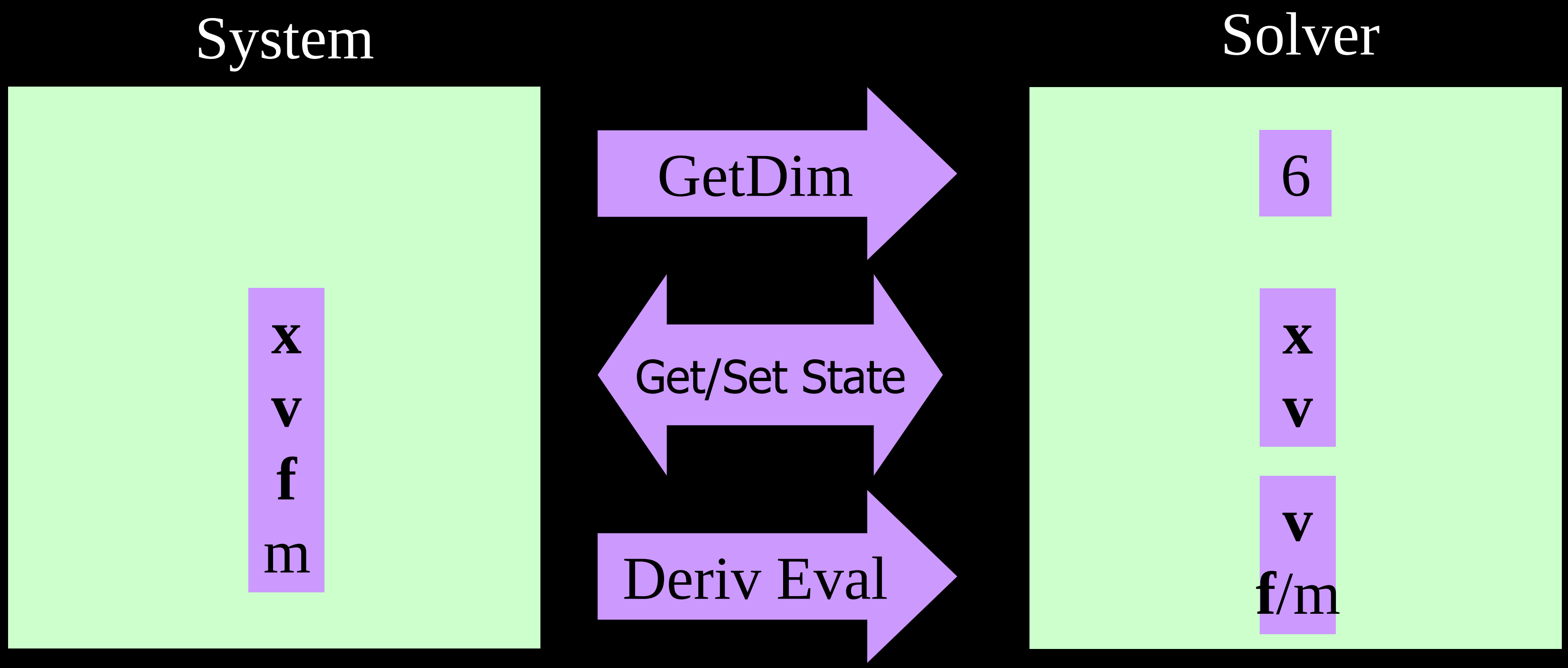
$$\mathbf{f}^{c} = 0 \qquad \mathbf{N}\cdot\mathbf{f} > 0$$

- Simple friction can be modeled

$$\mathbf{f}^{f} = -k_{f}(-\mathbf{N}\cdot\mathbf{f})\,v_{t} \qquad \mathbf{N}\cdot\mathbf{f} < 0$$

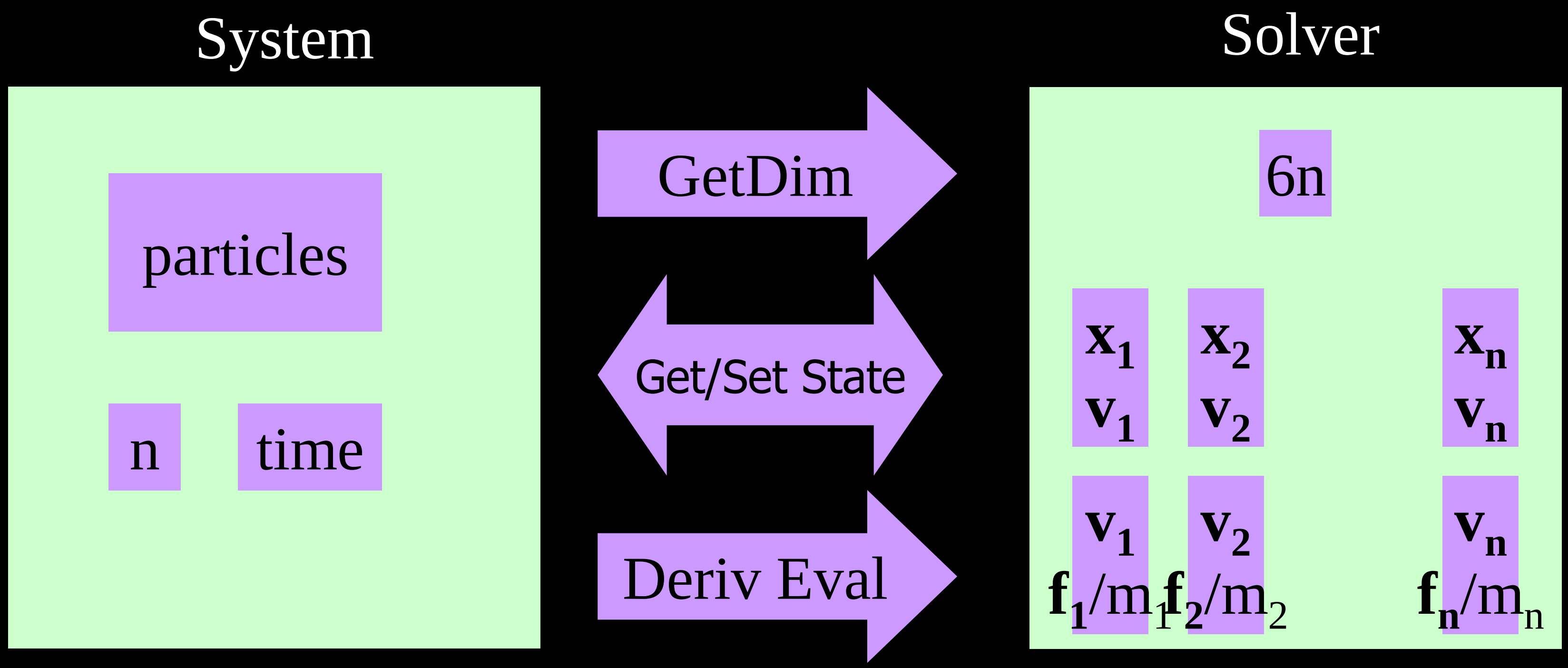
Implementation

- Solver Interface



Implementation (cont.)

- Solver Interface



Physics Engine for Dynamics Simulation

- Open Dynamics Engine (ODE)
  - Rigid body dynamics, robotics
- Bullet
  - Rigid and soft body dynamics, e.g., rope and cloth
- PhysX
  - Nvidia
- Unreal
- Unity

For more details on some engines, please see the article at https://www.goodfirms.co/blog/applications-physics-engine-animation