Particle Systems

Andrew Witkin

PIXAR

ANIMATION STUDIOS

Overview

- One Lousy Particle
- Particle Systems
- Forces: gravity, springs …
- Implementation and Interaction
- Simple collisions

A Newtonian Particle

- Differential equation: f = ma
- Forces can depend on:
  - Position, Velocity, Time

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

Second Order Equations

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

Not in our standard form because it has 2nd derivatives

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

Add a new variable, **v,** to get a pair of coupled 1st 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.*

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

Velocity in Phase Space: another 6-vector.

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

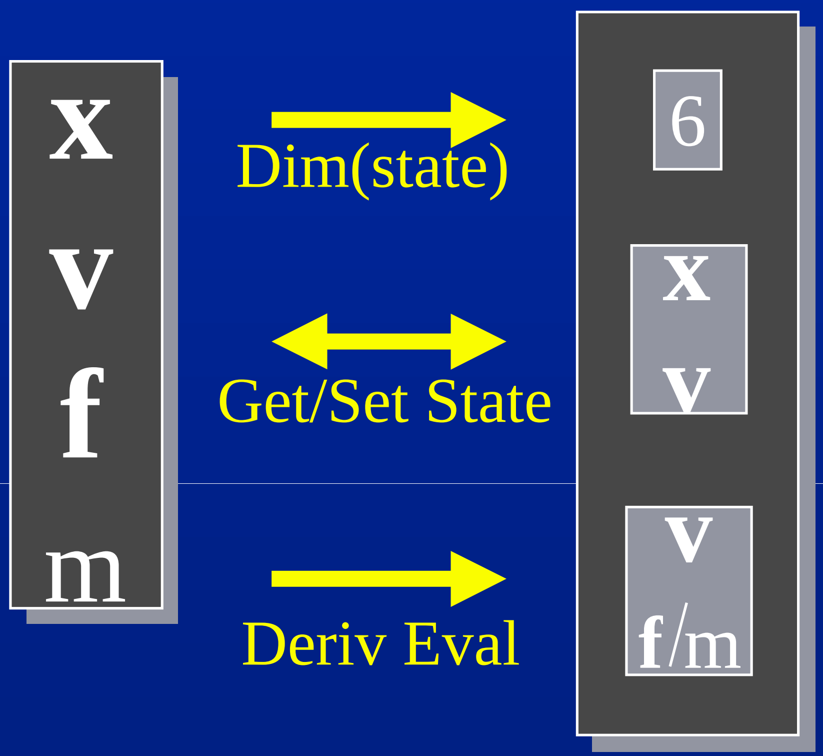
A vanilla 1st-order differential equation.

Particle Structure

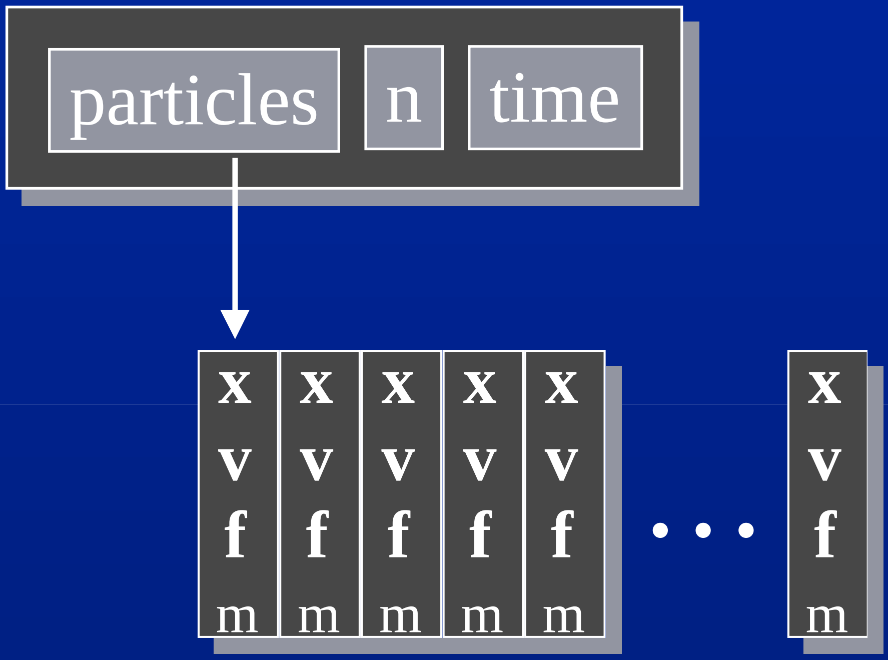
x — Position
v — Velocity
f — Force Accumulator
m — mass

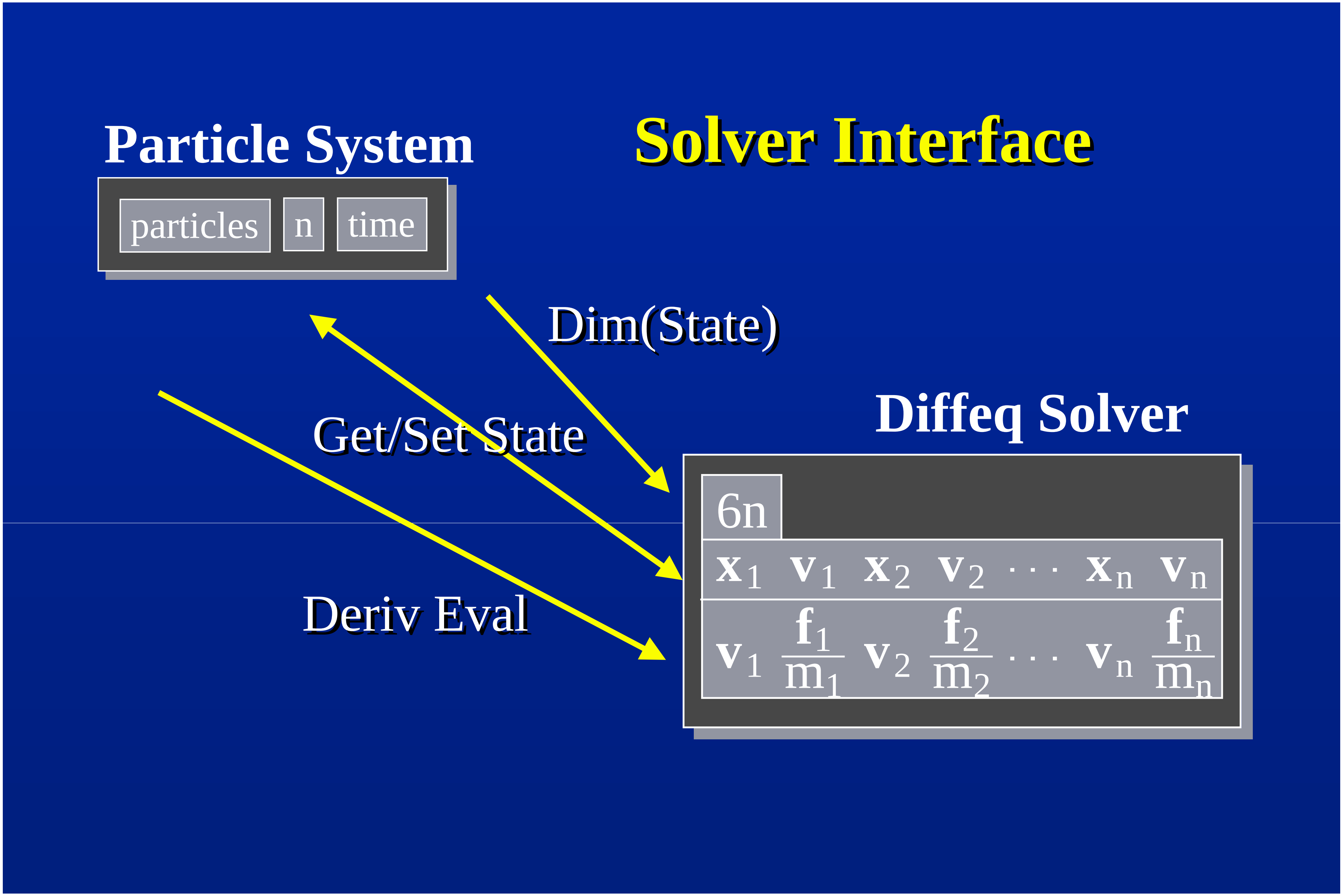
(x and v together:) Position in Phase Space

Solver Interface



Particle Systems




Solver Interface
Particle System
particles
n
time
Dim(State)
Get/Set State
Deriv Eval
Diffeq Solver
6n
x1 v1 x2 v2 · · · xn vn
v1 f1/m1 v2 f2/m2 · · · vn fn/mn

Deriv Eval 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.

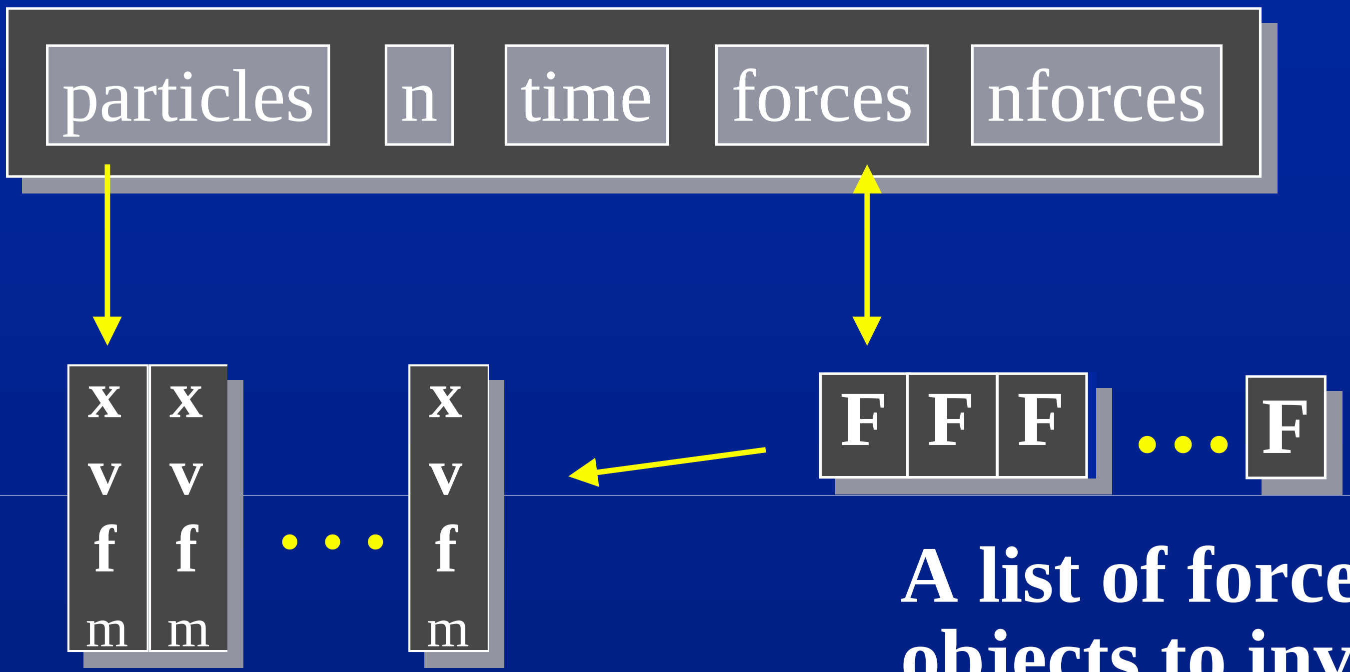
Forces

• Constant	gravity
• Position/time dependent	force fields
• Velocity-Dependent	drag
• n-ary	springs

Force Structures

- **Unlike particles, forces are heterogeneous.**
- **Force Objects:**
  - black boxes
  - point to the particles they influence
  - add in their own forces (type dependent)
- **Global force calculation:**
  - loop, invoking force objects

Particle Systems, with forces

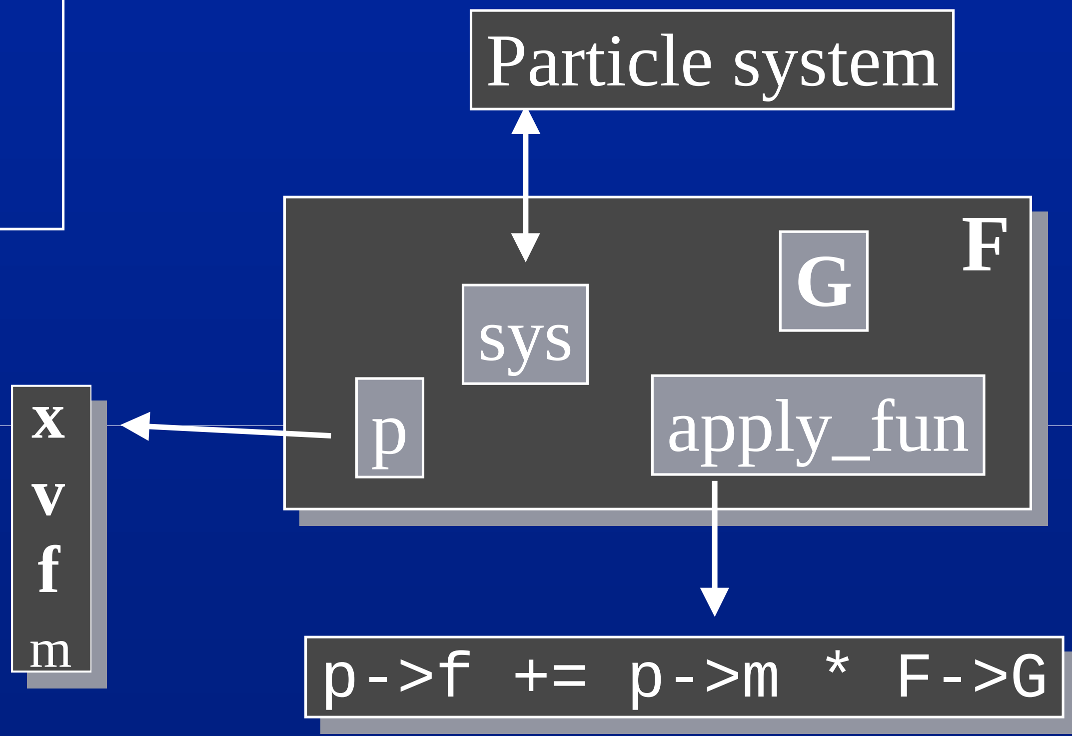


A list of force objects to invoke

Gravity

Force Law:

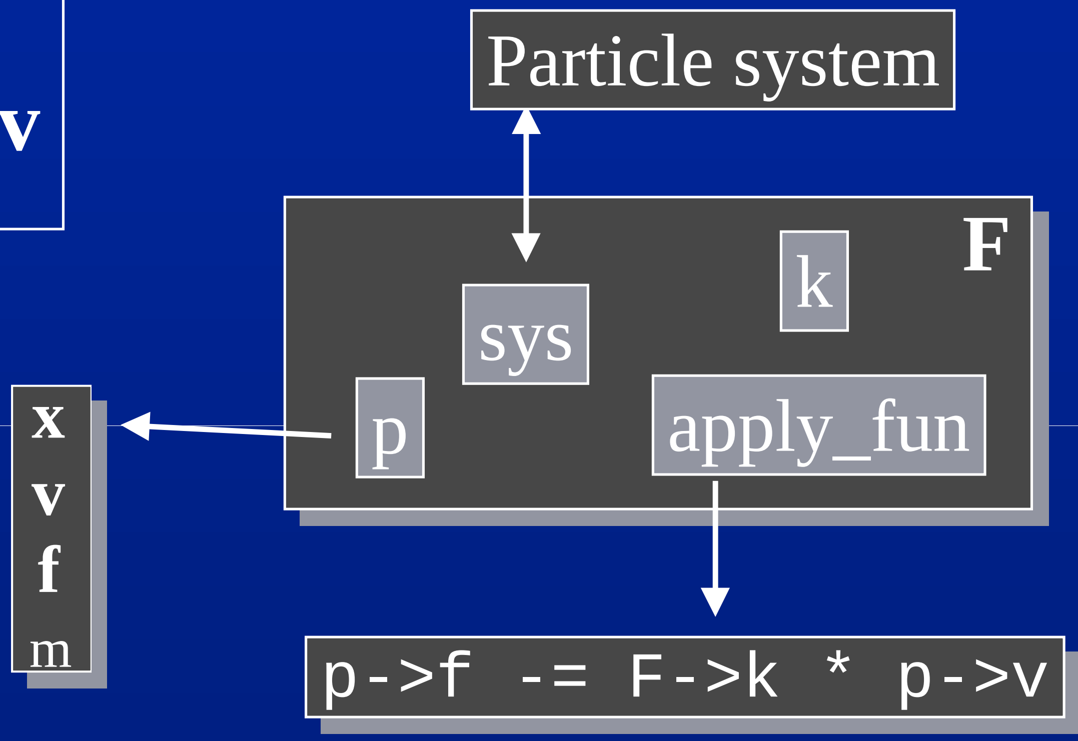
$$\mathbf{f}_{\text{grav}} = m\mathbf{G}$$



Viscous Drag

Force Law:

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

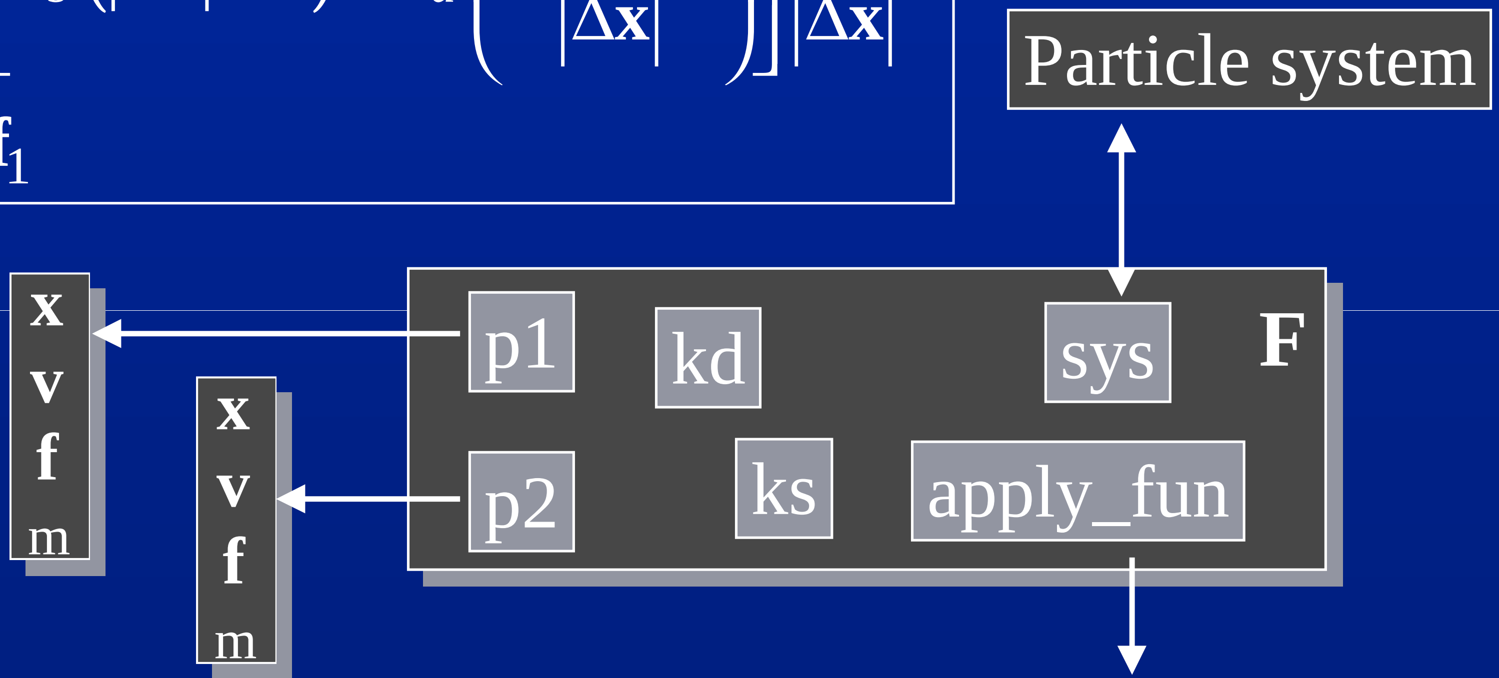


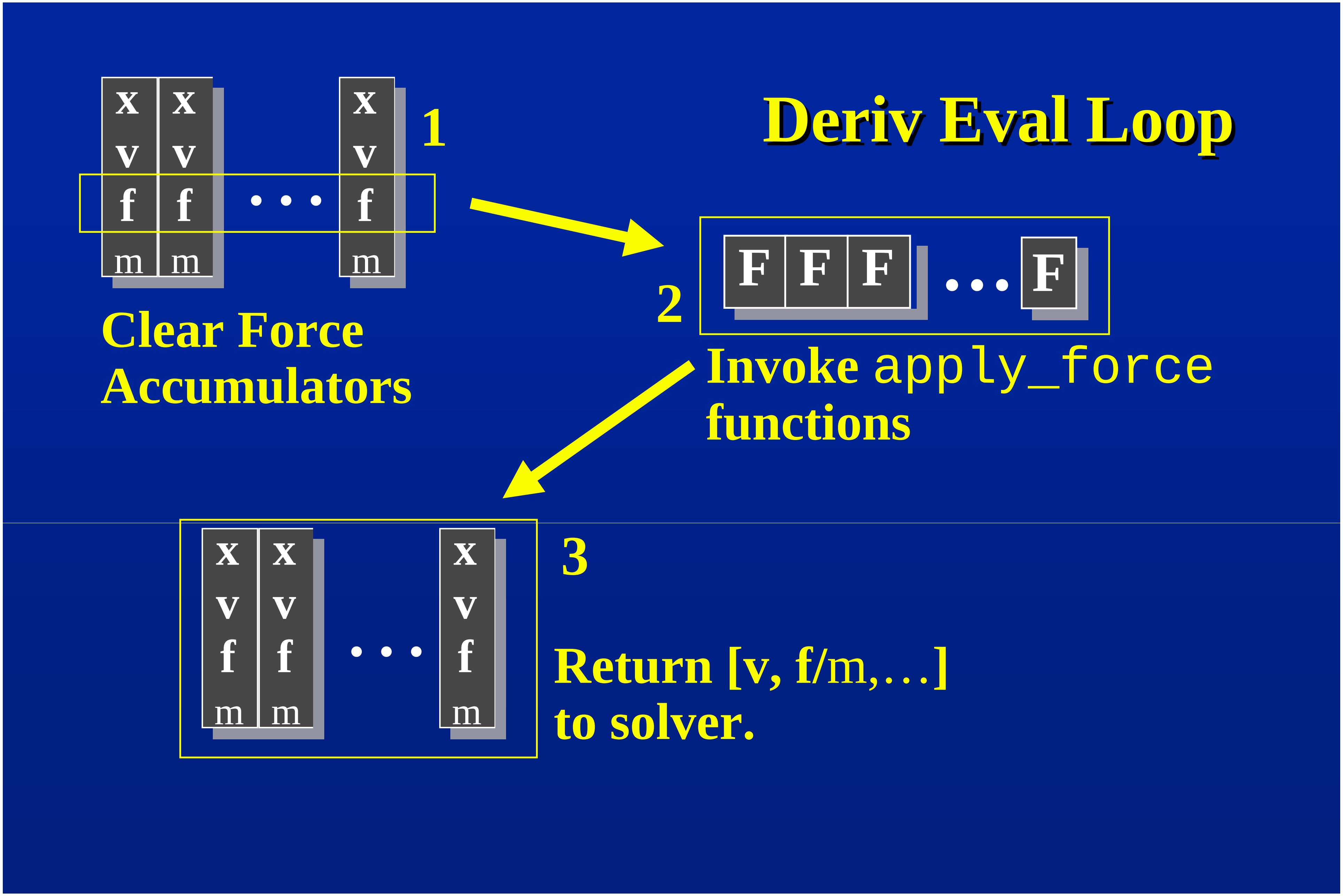
Damped Spring

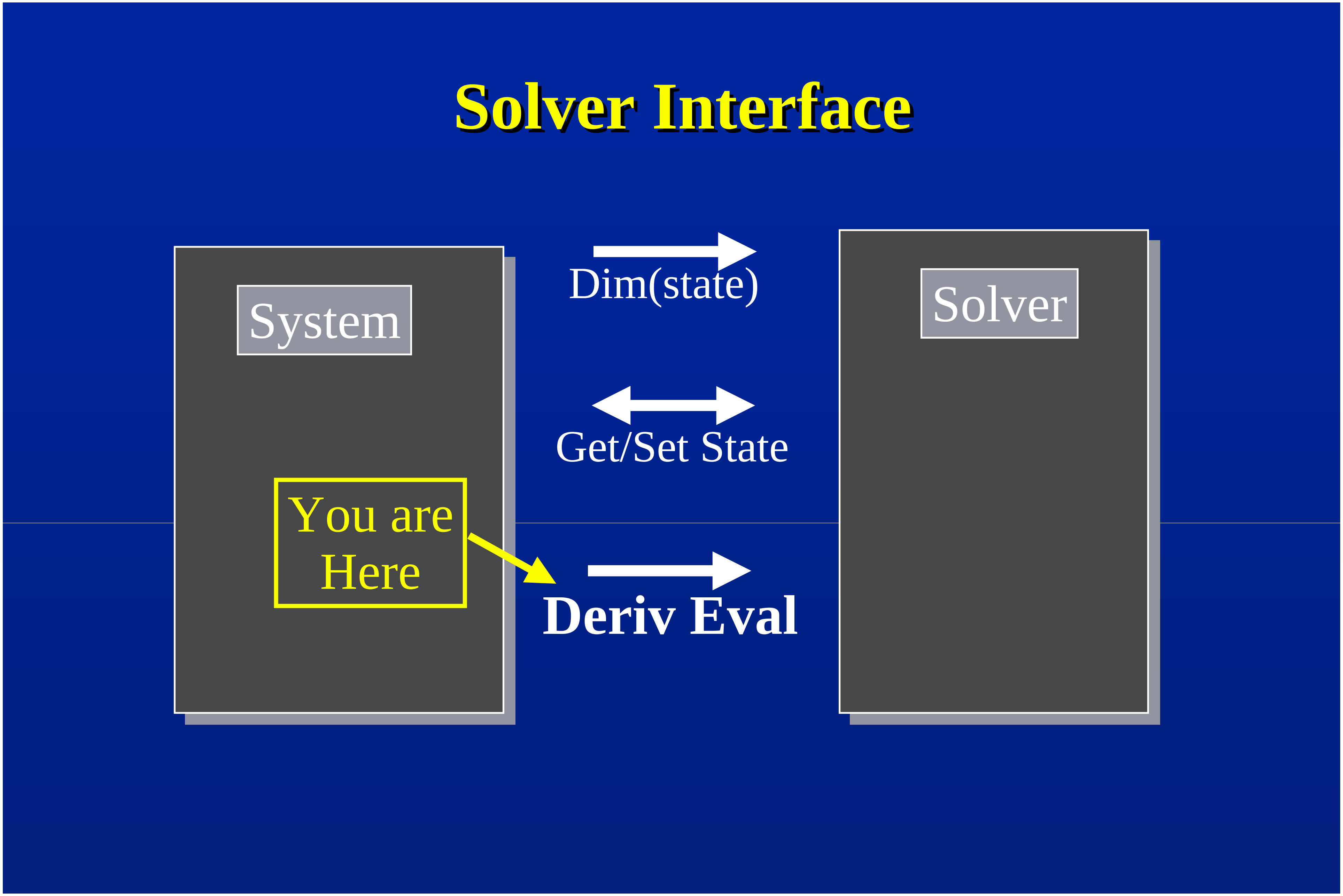
Force Law:

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

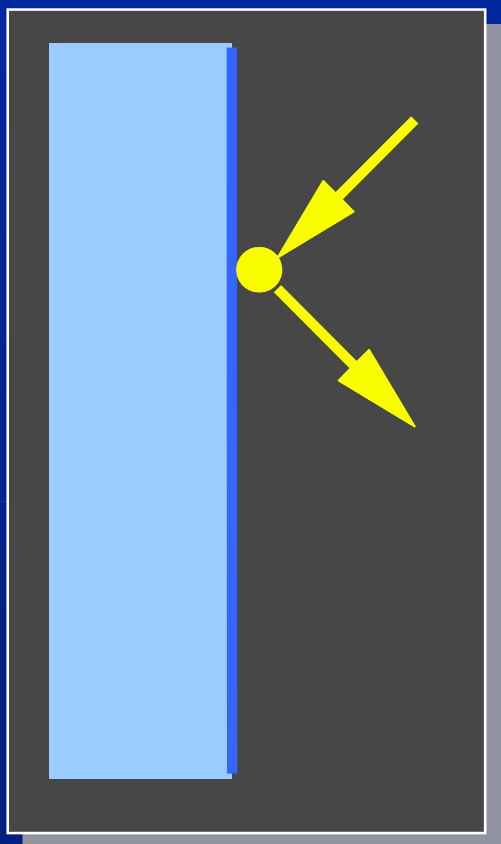
$$\mathbf{f}_2 = -\mathbf{f}_1$$




Deriv Eval Loop
x x x
v v v
f f • • • f
m m m
1
Clear Force
Accumulators
F F F • • • F
2
Invoke apply_force
functions
x x x
v v v
f f • • • f
m m m
3
Return [v, f/m,…]
to solver.

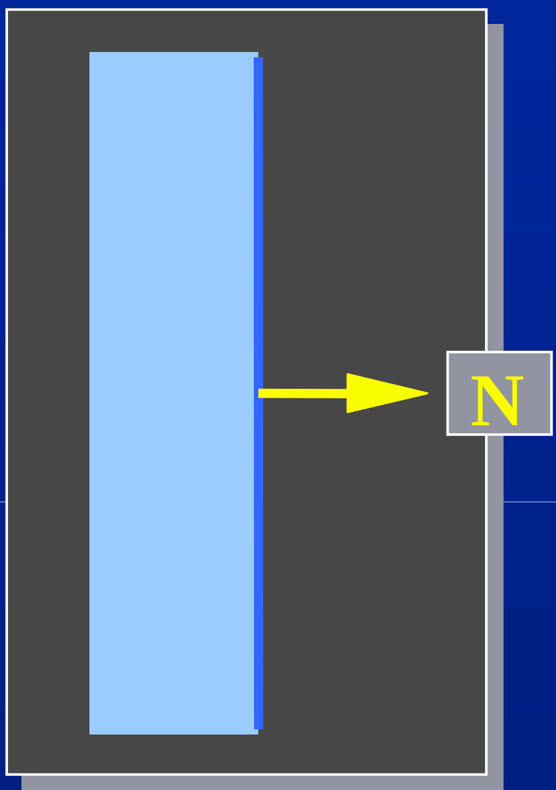

Solver Interface
System
Solver
Dim(state)
Get/Set State
Deriv Eval
You are Here

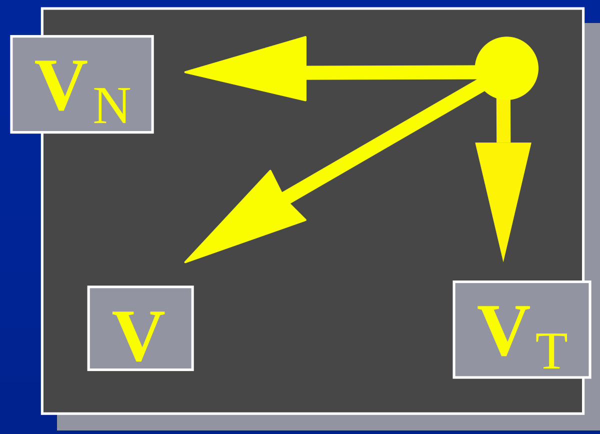
Bouncing off the Walls



- **Later: rigid body collision and contact.**
- **For now, just simple point-plane collisions.**
- **Add-ons for a particle simulator.**

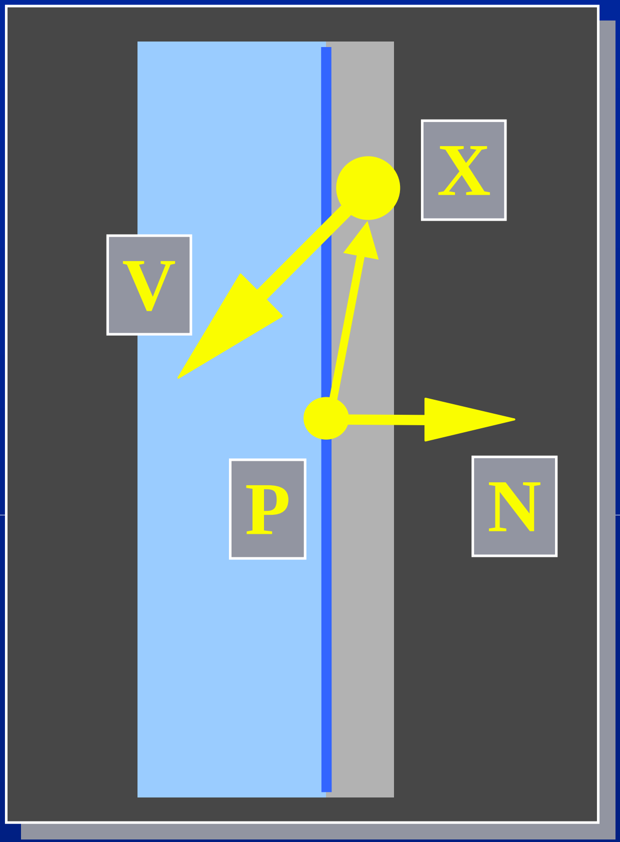
Normal and Tangential Components





$$V_N = (N \cdot V)N$$
$$V_T = V - V_N$$

Collision Detection

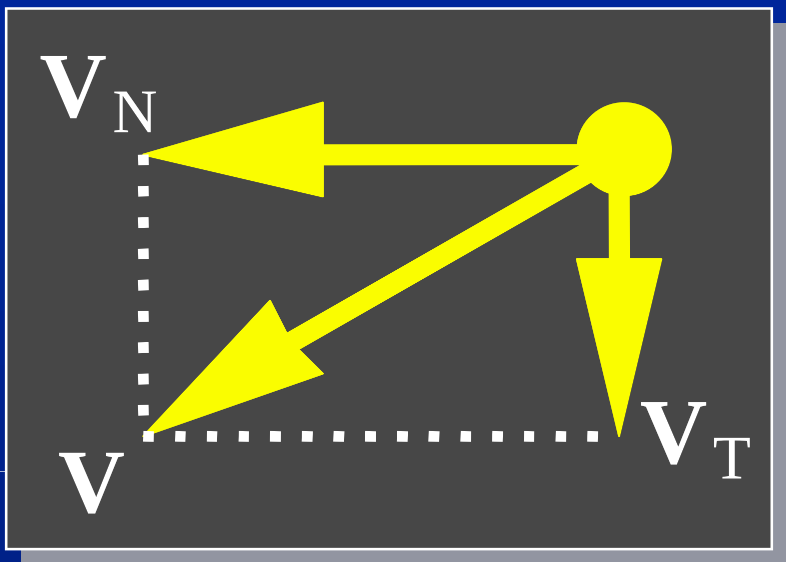


$$(\mathbf{X} - \mathbf{P}) \cdot \mathbf{N} < \varepsilon$$

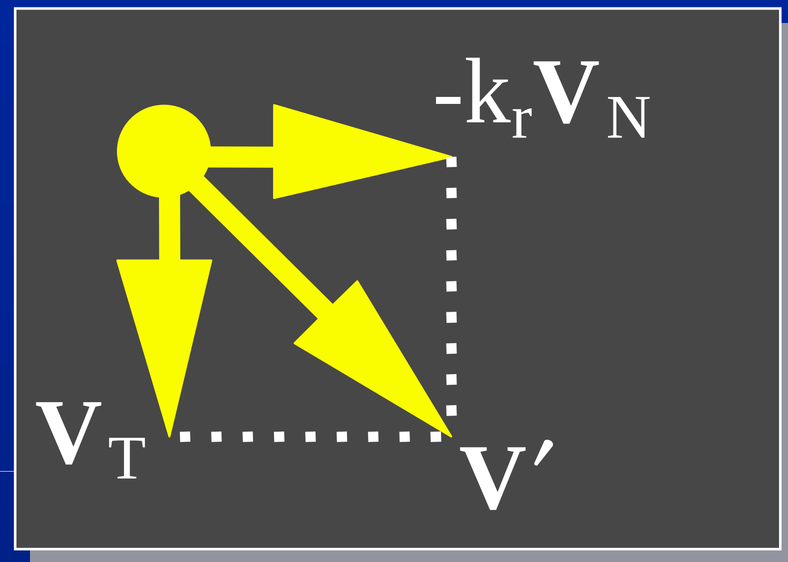
$$\mathbf{N} \cdot \mathbf{V} < 0$$

- Within ε of the wall.
- Heading in.

Collision Response



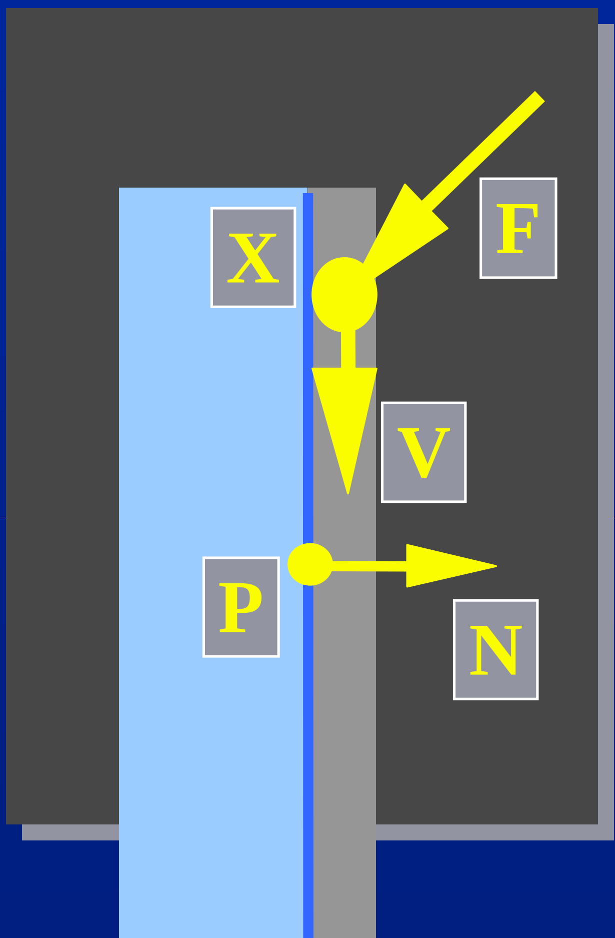
Before



After

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

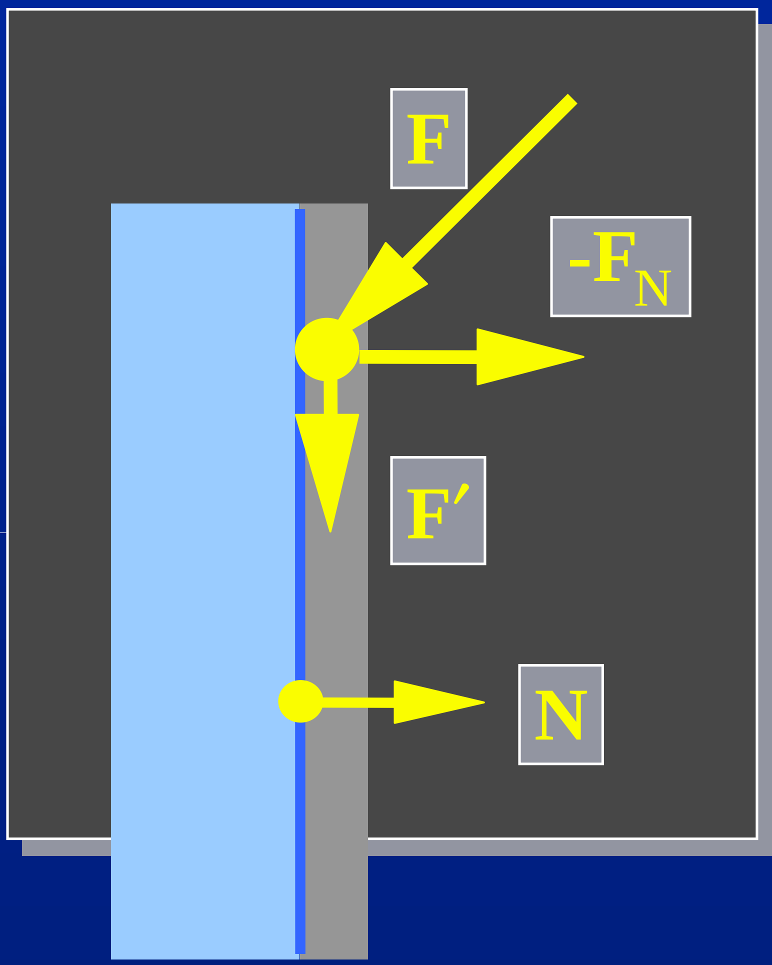
Conditions for Contact



$$|(\mathbf{X} - \mathbf{P}) \cdot \mathbf{N}| < \varepsilon$$

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

- On the wall
- Moving along the wall
- Pushing against the wall



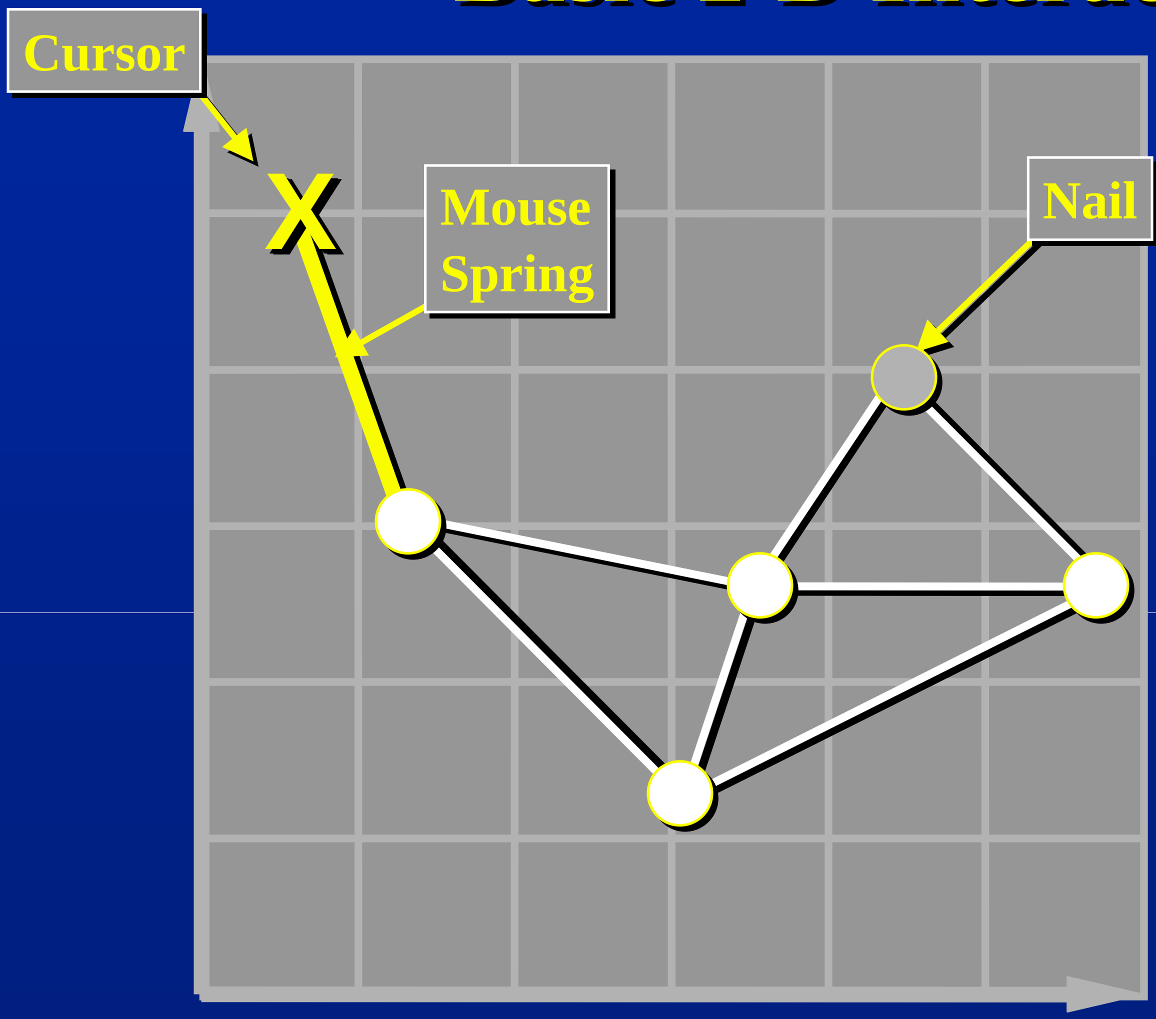
Contact Force

$$F' = F_T$$

The wall pushes back, cancelling the normal component of F.

(An example of a *constraint force.*)

Basic 2-D Interaction



Operations:

- Create
- Attach
- Drag
- Nail

Try this at home!

The notes give you everything you need to build a basic interactive mass/spring simulator—try it.