Slide Credit: Don Fussel

CS354 Computer Graphics
Particle Systems

Qixing Huang

April 18th 2018

Reading

- Required:
  - Witkin, *Particle System Dynamics*, SIGGRAPH '97 course notes on Physically Based Modeling.
  - Witkin and Baraff, *Differential Equation Basics*, SIGGRAPH '01 course notes on Physically Based Modeling.
- 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?

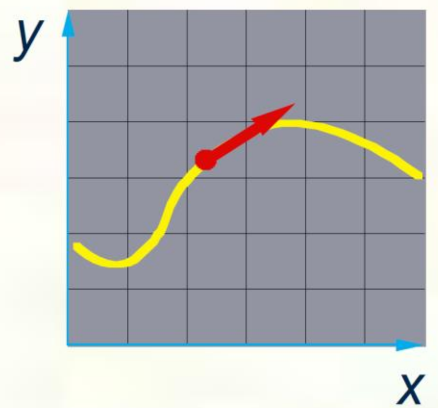
- A **particle system** is a collection of point masses that obeys some physical laws (e.g, gravity, heat convection, spring behaviors, …).

- Particle systems can be used to simulate all sorts of physical phenomena:

Particle in a flow field

- We begin with a single particle with:

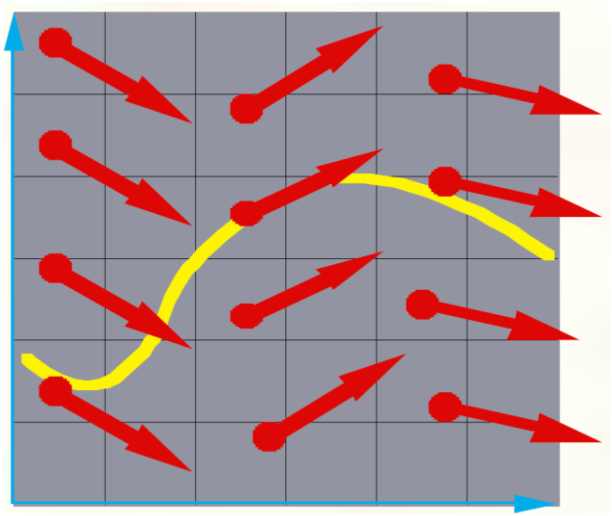
  - Position, $\vec{\mathbf{x}} = \begin{bmatrix} x \\ y \end{bmatrix}$
  - Velocity, $\vec{\mathbf{v}} = \dot{\mathbf{x}} = \frac{d\vec{\mathbf{x}}}{dt} = \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix}$



- Suppose the velocity is actually dictated by some driving function **g**: $\dot{\mathbf{x}} = \mathrm{g}(\vec{\mathbf{x}}, t)$

Vector fields

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

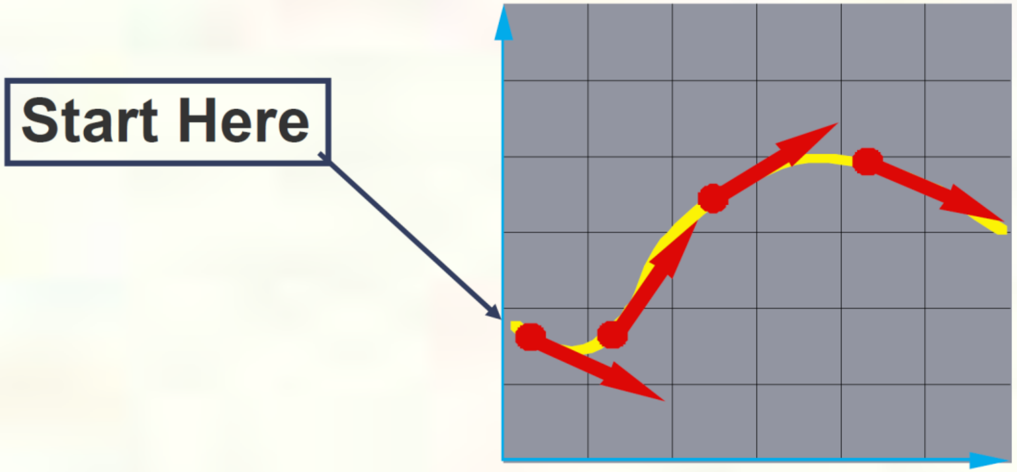


- How does our particle move through the vector field?

Diff eqs and integral curves

- The equation $\dot{\mathbf{x}} = g(\vec{\mathbf{x}}, t)$

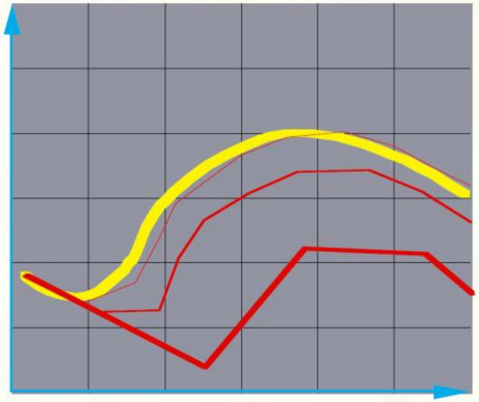
  is actually a **first order differential equation**.
- We can solve for **x** through time by starting at an initial point and stepping along the vector field:



- This is called an **initial 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: $$\vec{\mathbf{x}}(t+\Delta t) = \vec{\mathbf{x}}(t) + \Delta t \cdot \dot{\mathbf{x}}(t) = \vec{\mathbf{x}}(t) + \Delta t \cdot g(\vec{\mathbf{x}}, t)$$
- Writing as a time iteration: $$\vec{\mathbf{x}}^{i+1} = \vec{x}^{i} + \Delta t \cdot \vec{\mathbf{v}}^{i}$$
- This approach is called **Euler's method** and looks like:



- Properties:
  - Simplest numerical method
  - Bigger steps, bigger errors. Error ~ $O(\Delta t^2)$.
- Need to take pretty small steps, so not very efficient. Better (more complicated) methods exist, e.g., "Runge-Kutta" and "implicit integration."

Particle in a force field

- Now consider a particle in a force field **f**.
- In this case, the particle has:
  - Mass, m
  - Acceleration, $\vec{\mathbf{a}} \equiv \ddot{\mathbf{x}} = \dfrac{d\vec{\mathbf{v}}}{dt} = \dfrac{d^2\vec{\mathbf{x}}}{dt^2}$
- The particle obeys Newton's law: $\vec{\mathbf{f}} = m\vec{\mathbf{a}} = m\ddot{\mathbf{x}}$
- 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:

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

Second order equations

This equation:

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

is a **second order differential equation**.

Our solution method, though, worked on first order differential equations.

We can rewrite this as:

$$\begin{bmatrix} \dot{\mathbf{x}} = \vec{\mathbf{v}} \\ \dot{\mathbf{v}} = \dfrac{\vec{\mathbf{f}}(\vec{\mathbf{x}}, \vec{\mathbf{v}}, t)}{m} \end{bmatrix}$$

where we have added a new variable **v** to get a pair of coupled first order equations.

Phase space

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

- 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.

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

- A vanilla 1st-order differential equation.

Differential equation solver

Starting with:

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

Applying Euler' s method:

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

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

And making substitutions:

$$\vec{\mathbf{x}}(t+\Delta t) = \vec{\mathbf{x}}(t) + \Delta t \cdot \vec{\mathbf{v}}(t)$$

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

Writing this as an iteration, we have:

$$\vec{\mathbf{x}}^{i+1} = \vec{x}^{i} + \Delta t \cdot \vec{\mathbf{v}}^{i}$$

$$\vec{\mathbf{v}}^{i+1} = \vec{\mathbf{v}}^{i} + \Delta t \cdot \frac{\vec{\mathbf{f}}^{i}}{m}$$

Again, performs poorly for large Δt.

Verlet Integration

- Also called Størmer's Method
  - Invented by Delambre (1791), Størmer (1907), Cowell and Crommelin (1909), Verlet (1960) and probably others

- More stable than Euler's method (time reversible as well)

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)
  - Combinations (Damped springs)
- How do we compute the net force on a particle?

Gravity and viscous drag

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

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

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

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

$$\vec{\mathbf{f}}_{drag} = -k\vec{\mathbf{v}}$$

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

Damped spring

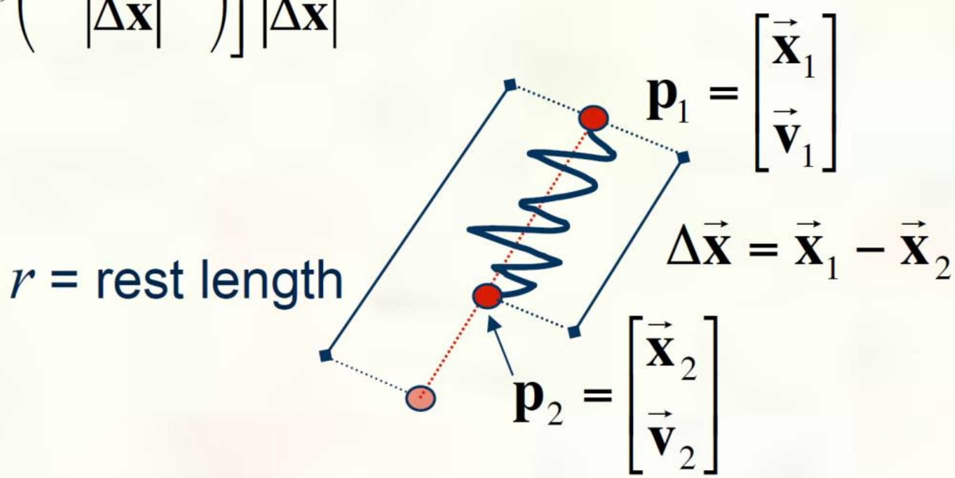
Recall the equation for the force due to a spring: $f = -k_{spring}(|\Delta\vec{\mathbf{x}}| - r)$

We can augment this with damping: $f = -\left[k_{spring}(|\Delta\vec{\mathbf{x}}| - r) + k_{damp}|\vec{\mathbf{v}}|\right]$

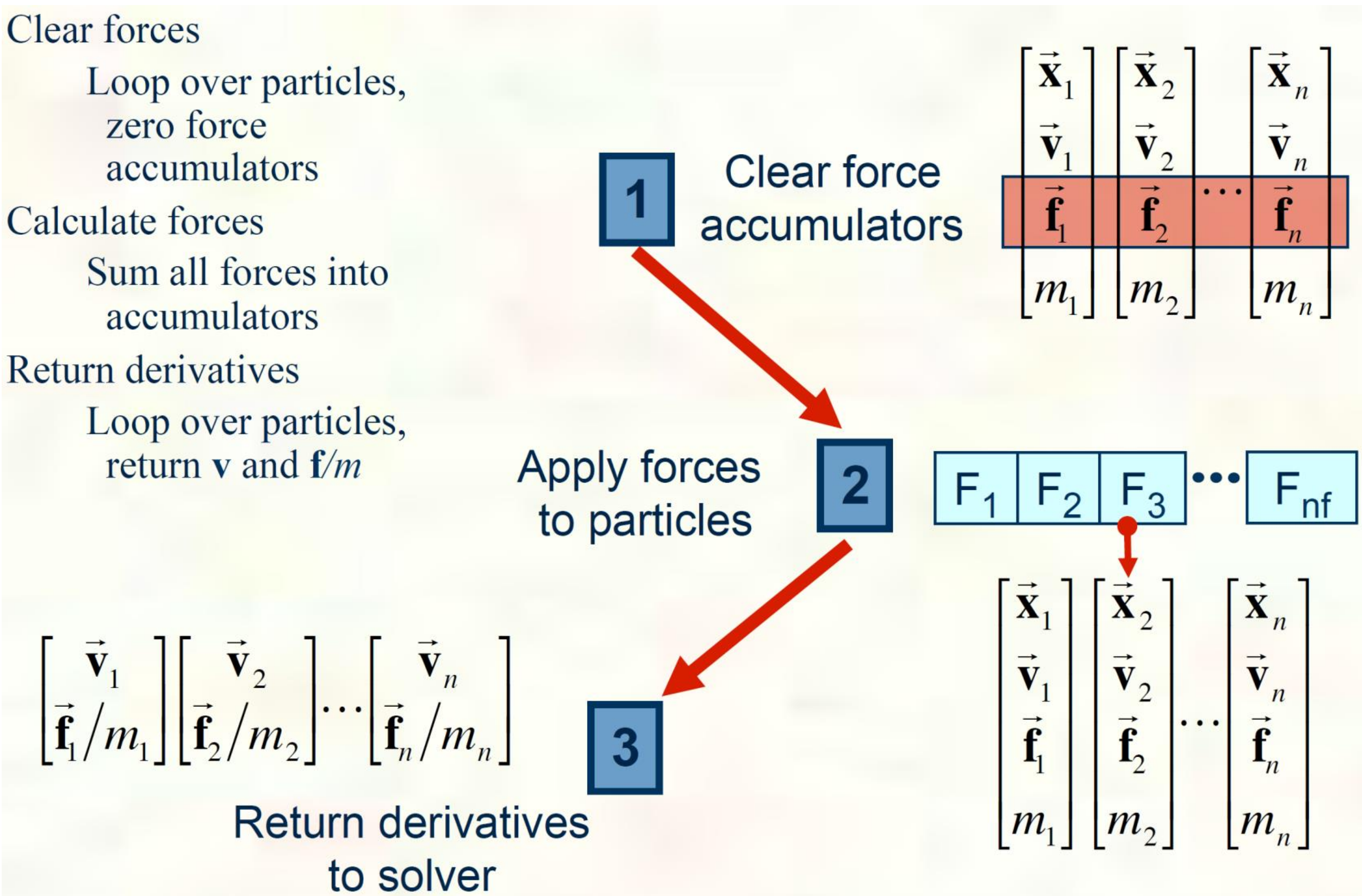
The resulting force equations for a spring between two particles become:

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

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

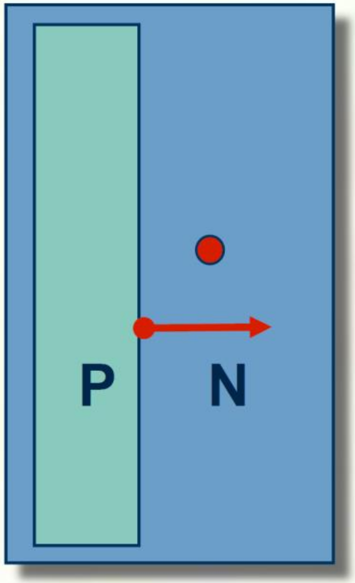


derivEval



Clear forces
Loop over particles, zero force accumulators
Calculate forces
Sum all forces into accumulators
Return derivatives
Loop over particles, return v and f/m
1
Clear force accumulators
2
Apply forces to particles
F1 F2 F3 ••• Fnf
3
Return derivatives to solver

Bouncing off the walls

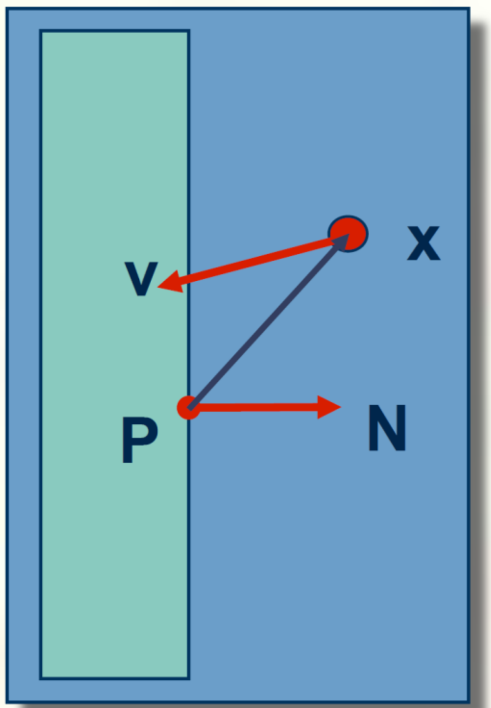


- Add-on for a particle simulator
- For now, just simple point-plane collisions

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

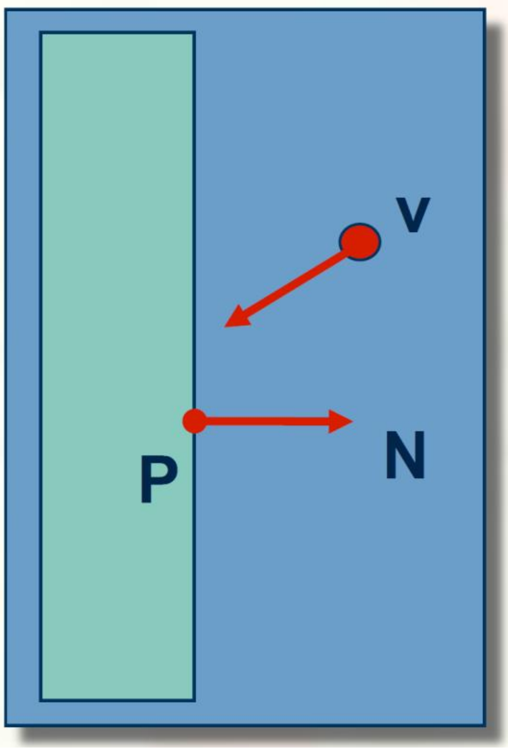
Collision Detection

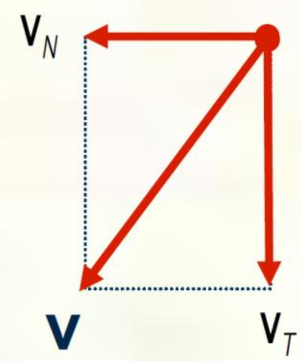
How do you decide when you' ve crossed a plane?



Normal and tangential velocity

To compute the collision response, we need to consider the normal and tangential components of a particle's velocity.

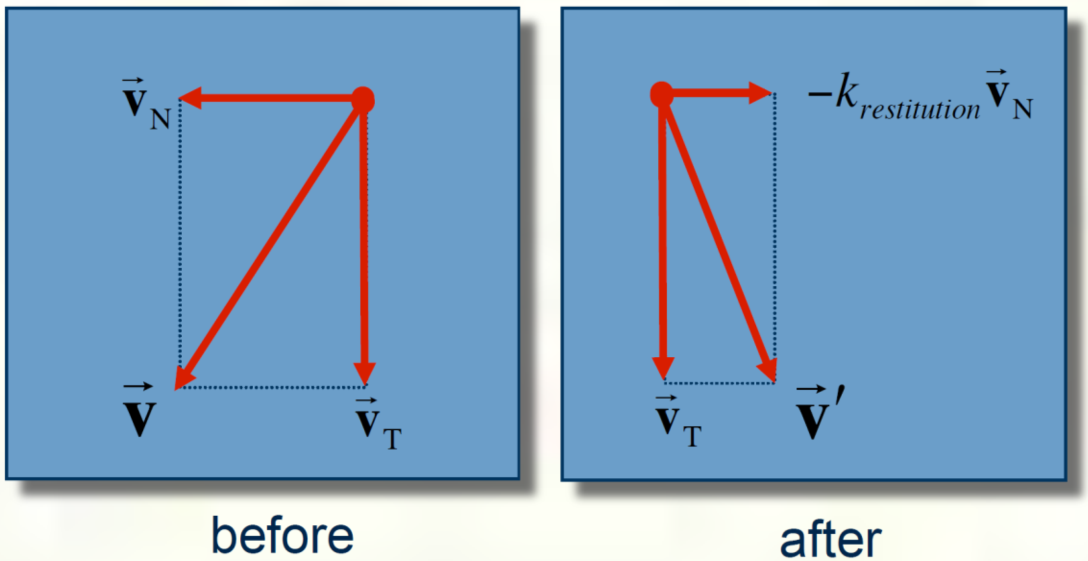




$$\vec{\mathbf{v}}_N = \left(\vec{\mathbf{N}} \bullet \vec{\mathbf{v}}\right)\vec{\mathbf{N}}$$

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

Collison Response



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

Without backtracking, the response may not be enough to bring a particle to the other side of a wall.
In that case, detection should include a velocity check:

Discussion