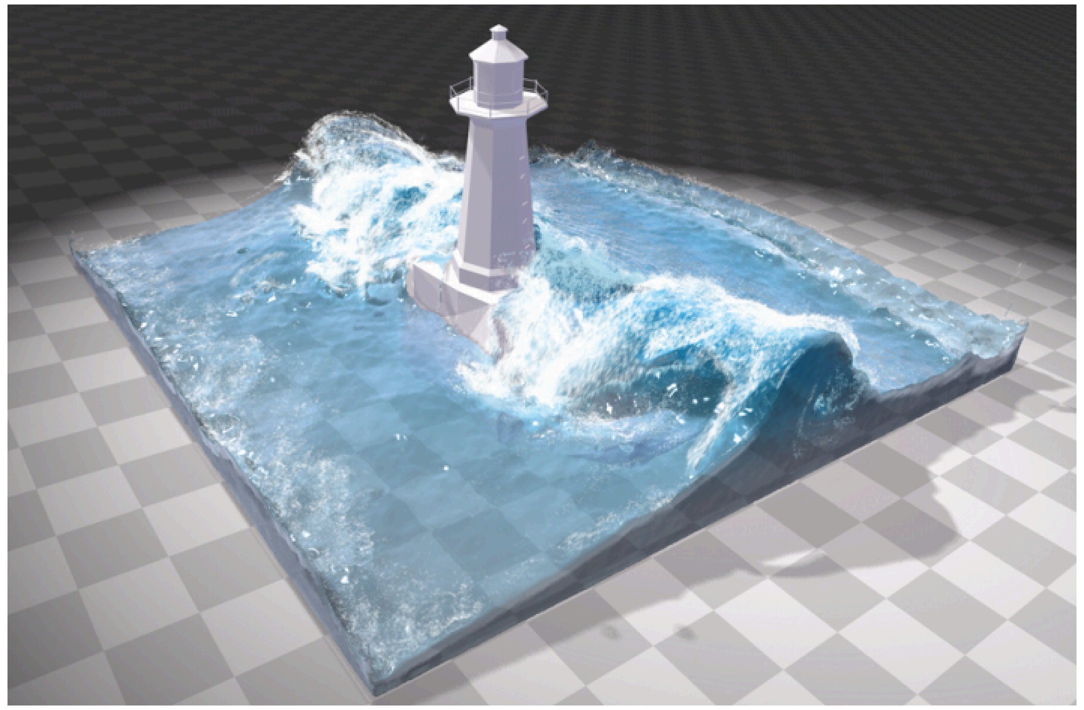
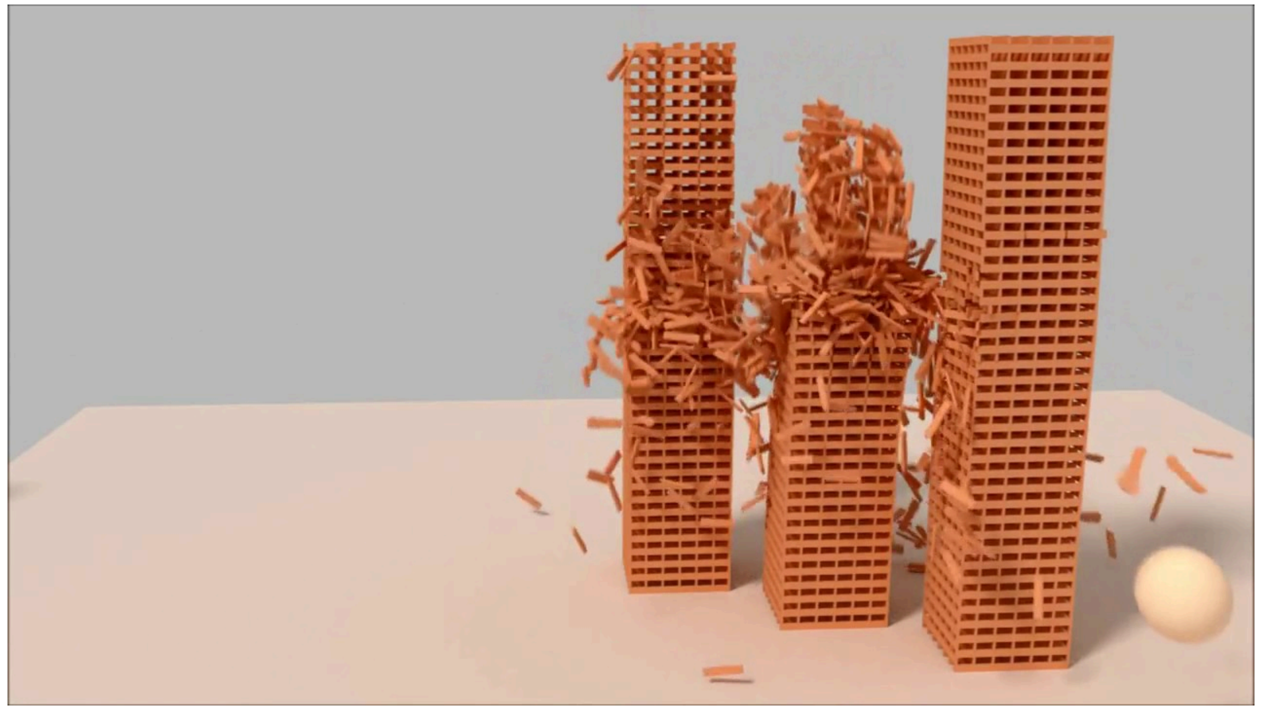
Physically based simulation - Models

When physically based simulation is needed

- Accurate dynamics
- Teadious to model by hand or procedurally
    - Multiple interacting elements: ex. Multiple collisions: rigid bodies, hairs, etc.
    - Complex animated geometry: Cloths, fluids





General methodology

1. Description of the system

Describe system by some parameters (positions, speed, orientation, etc).

- State of the system is known at time $t = 0$ - *Initial value problem in time*
- State of the system may be constrained in space - *Boundary value problem in space*

2. Evolution

Link the evolution of the system to forces or constraints using dynamic principles and conservation laws

$\Rightarrow$ Differential equation

3. Numerical Solution

Solve the differential equation using numerical iterative approaches.

Note: Fundamentally different that direct approach controling the trajectories at key-frames

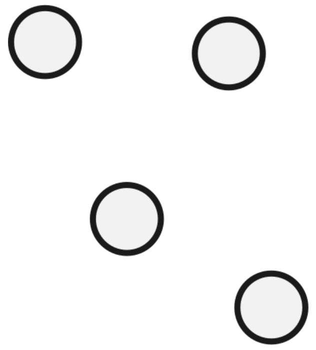
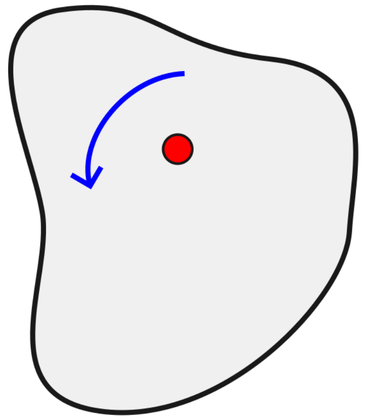
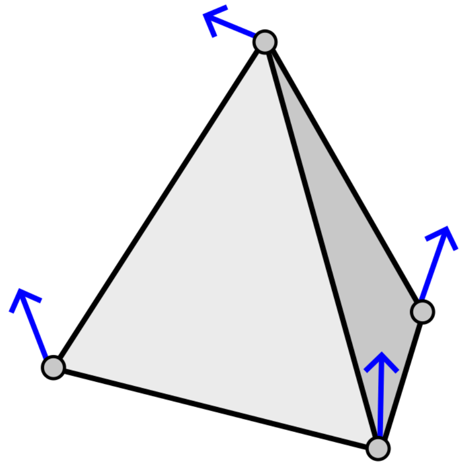
- The system is set at an initial step
- We let the numerical solution build the space-time trajectory for us

(+) Allows to model complex behavior

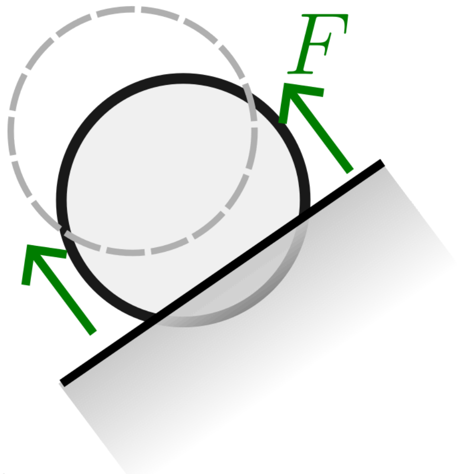
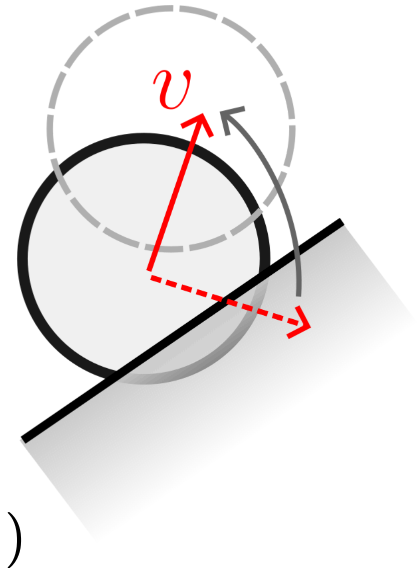
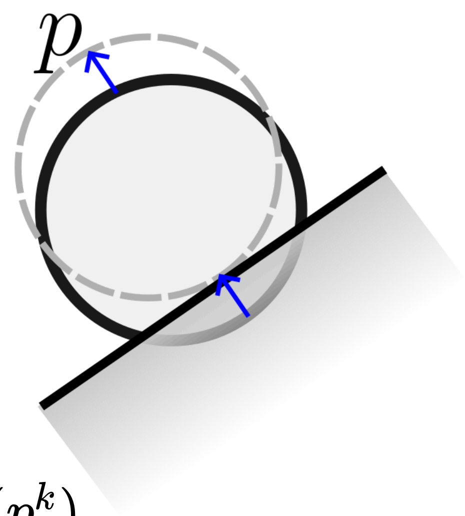
(-) Lack of control on the result

Types of Simulation Models

Deformable model

Particle	Solid	Deformable
		
(p_i, v_i)	(p_i, v_i, R_i, L_i)	$\sigma = C\,\epsilon$

Collision/Constraint Handling

Force based	Impulse based	Position based
		
$v_i^{k+1} = v_i^k + F_i/m_i\,\Delta t$	$v_i^{k+1} \rightarrow \mathcal{F}(v^k)$	$p_i^{k+1} \rightarrow \mathcal{F}(p^k)$

Fundamental models

1- Particles
2- Rigid bodies
3- Continuum models

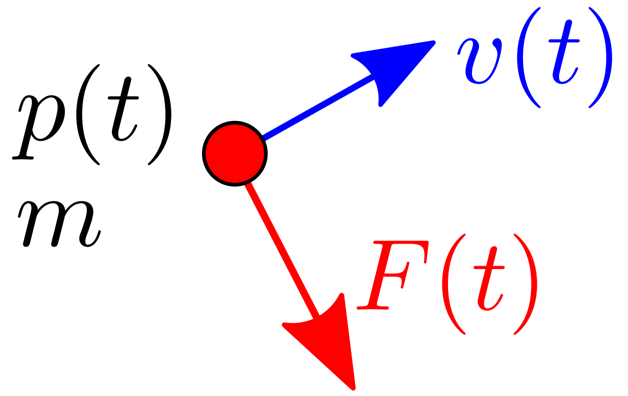
Physically-based particle system

1. Description

Particle is fully described by: Position p, Velocity v, Mass m

Fundamental quantities: position and linear momentum $P = m\,v$

Linear Momentum preserved in isolated system



2. Evolution

- Fundamental principle of dynamics

Force applied on particle $F(p, v, t)$

$$\begin{cases} p'(t) = v(t) \\ P'(t) = m\,v'(t) = F(p, v, t) \end{cases}$$

- Conservation of energy (ex. kinetic energy ($1/2\,m\,v^2$)+potential energy = const, etc.)

- Lagragian, or Hamiltonian (reduced coordinates)

Physically-based particle system

3. Numerical Solution

ODE (Ordinary Differential Equation) formulated as an **Initial Value Problem**

ex. $$\begin{cases} p'(t) = v(t) \\ mv'(t) = F(p, v, t) \end{cases} \quad , \quad \text{with } v(0) = v_0,\ p(0) = p_0$$

- Discretize in time $t^k = k\,h$, $\quad h = \Delta t =$ time step.

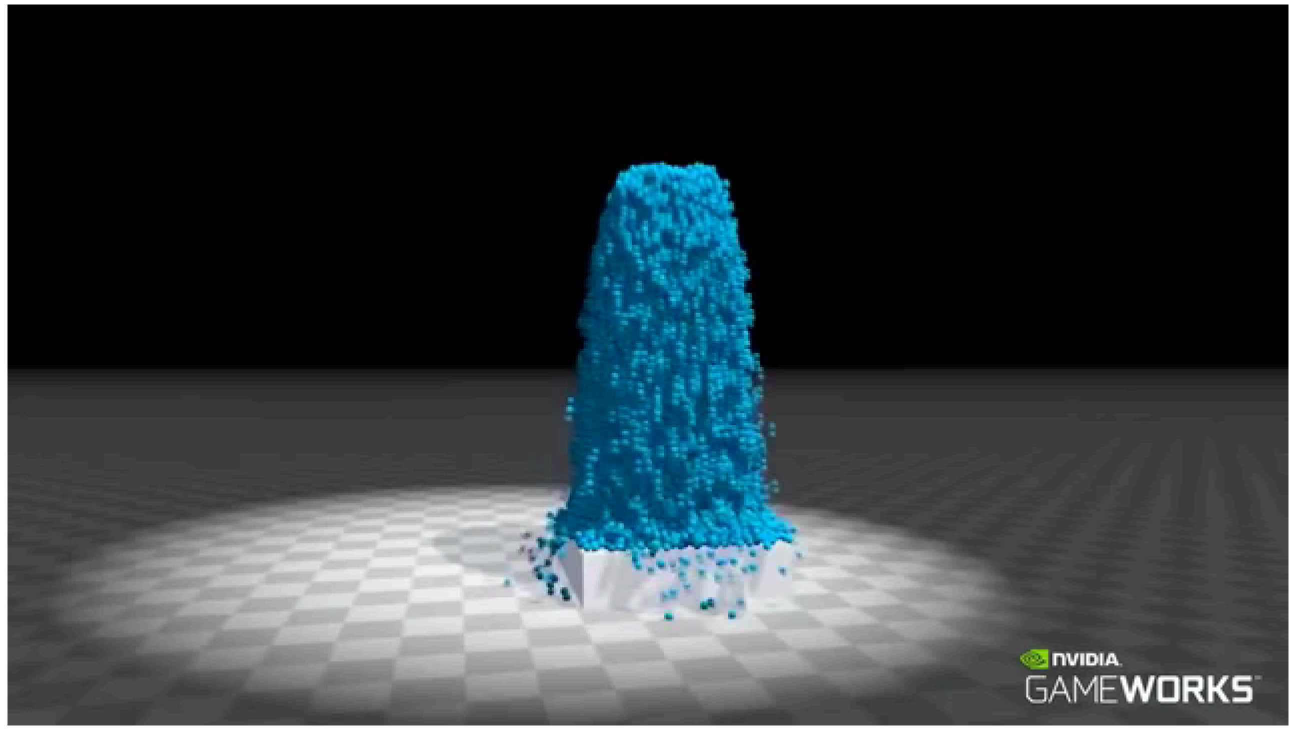
$\Rightarrow$ Build a discrete numerical solution $p^k = p(t^k), v^k = v(t^k)$.

- We can consider initially the following iterative scheme

$$\begin{cases} v^{k+1} = v^k + h\,F(p^k, v^k, t^k) \\ p^{k+1} = p^k + h\,v^{k+1} \end{cases}$$

Simple to implement, reasonably OK for simple examples (more details later).

Physically-based particle system - pro/cons



Pro

(+) Simple to implement, and control.

(+) Efficient to compute, scalable

(+) Highly adaptable from simple particle to rigid and deformable models

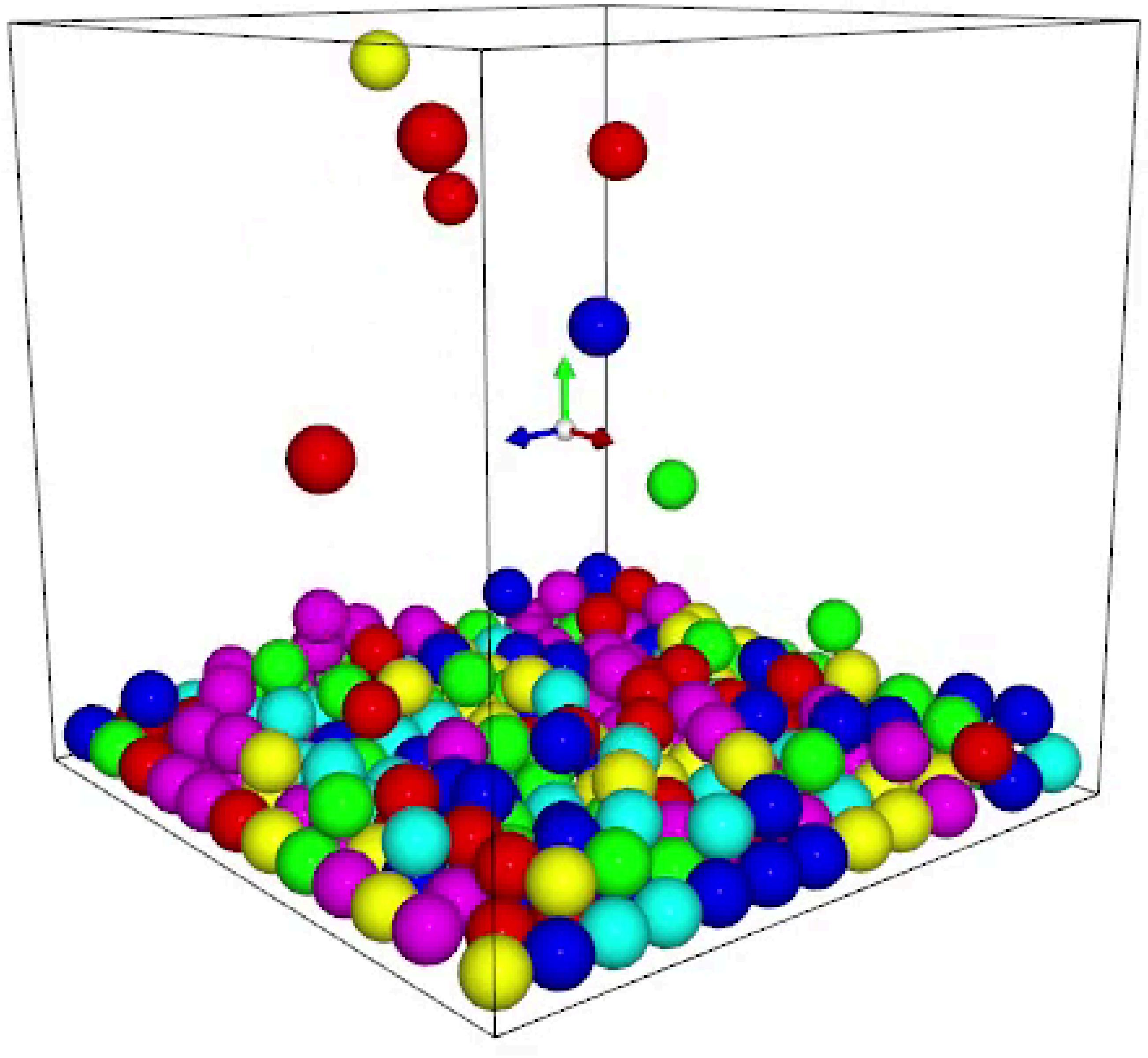
Cons

(-) Limited accuracy - highly simplified model from physical point of view.

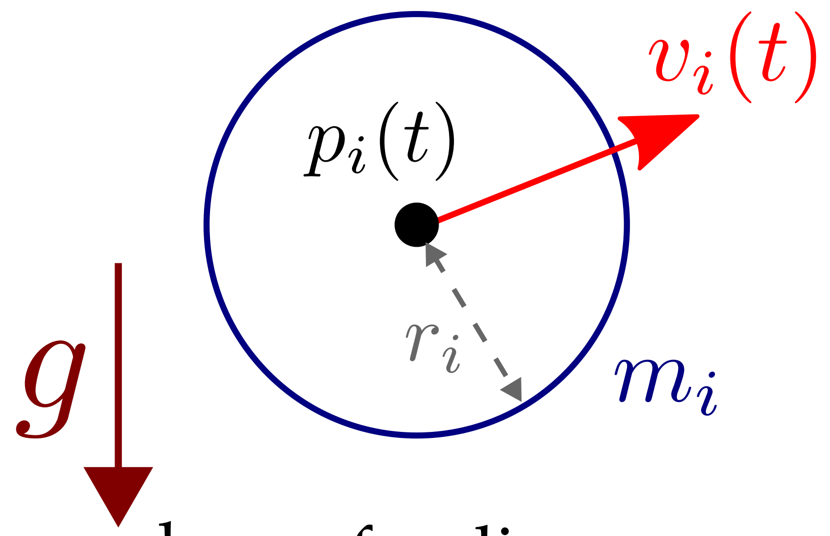
⇒ Dominant model in CG production for general purpose deformable model animation.

Common use: Lots of sparsely interacting particles

Rigid spheres



System modeling



Particles modeling the center of hard spheres.

- Spheres can collide with surrounding obstacles
- Spheres can collide with each others

- *System*: N particles with position p_i, velocity v_i, mass m_i, modeling a sphere of radius r_i.
  - Initial conditions $p_i(0) = p_i^0$, $v_i(0) = v_i^0$

- *Forces*: Single gravity forces $F_i = m_i\, g$. Collisions handled by *impulses*.

- *Temporal evolution*: Fundamental principle of dynamics $v_i(t) = p_i'(t)$, $v_i'(t) = g$

- *Numerical solution*

$$\begin{cases} v^{k+1} = v^k + h\, g \\ p^{k+1} = p^k + h\, v^{k+1} \end{cases}$$

Collision with a plane

Plane $\mathcal{P}$: parameterized using a point a and its normal n.

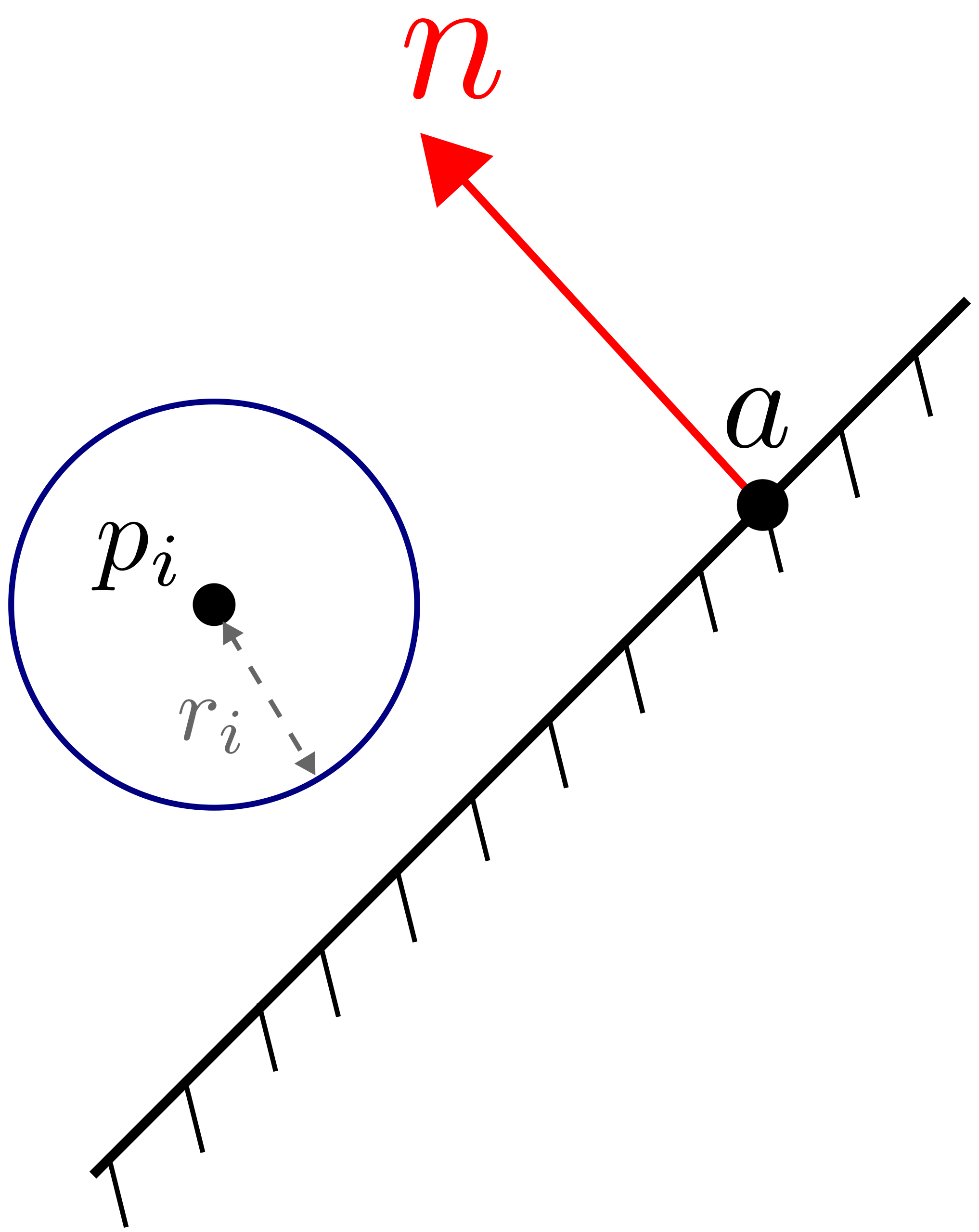
$\{p \in \mathbb{R}^3 \in \mathcal{P} \Rightarrow (p-a)\cdot n = 0\}$

- Sphere above plane : $(p_i - a)\cdot n > r_i$
- Sphere in collision: $(p_i - a)\cdot n \leq r_i$

- Collision detection algorithm

```
for(int i=0; i<N; ++i)
{
    float detection = dot(p[i]-a, n);
    if (detection <= r[i])
    {
        // ... collision response
    }
}
```

What should we do when a collision is detected



Collision response with plane

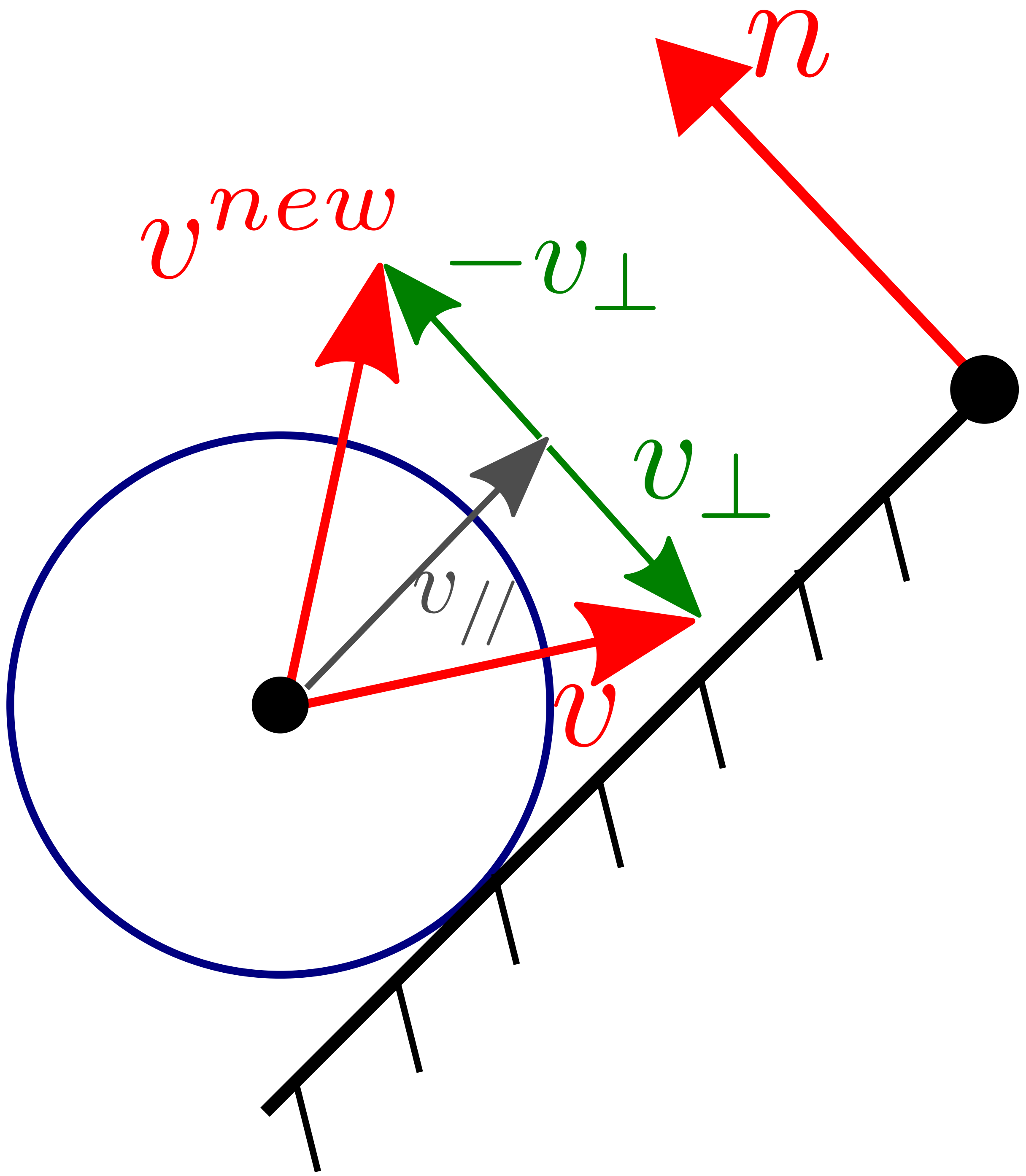
Suppose exact contact: $(p_i - a) \cdot n = r_i$
Collision response = **Update velocity**

Split $v = v_{//} + v_\perp$
- $v_\perp = (v \cdot n)\, n$
- $v_{//} = v - (v \cdot n) n$

New velocity

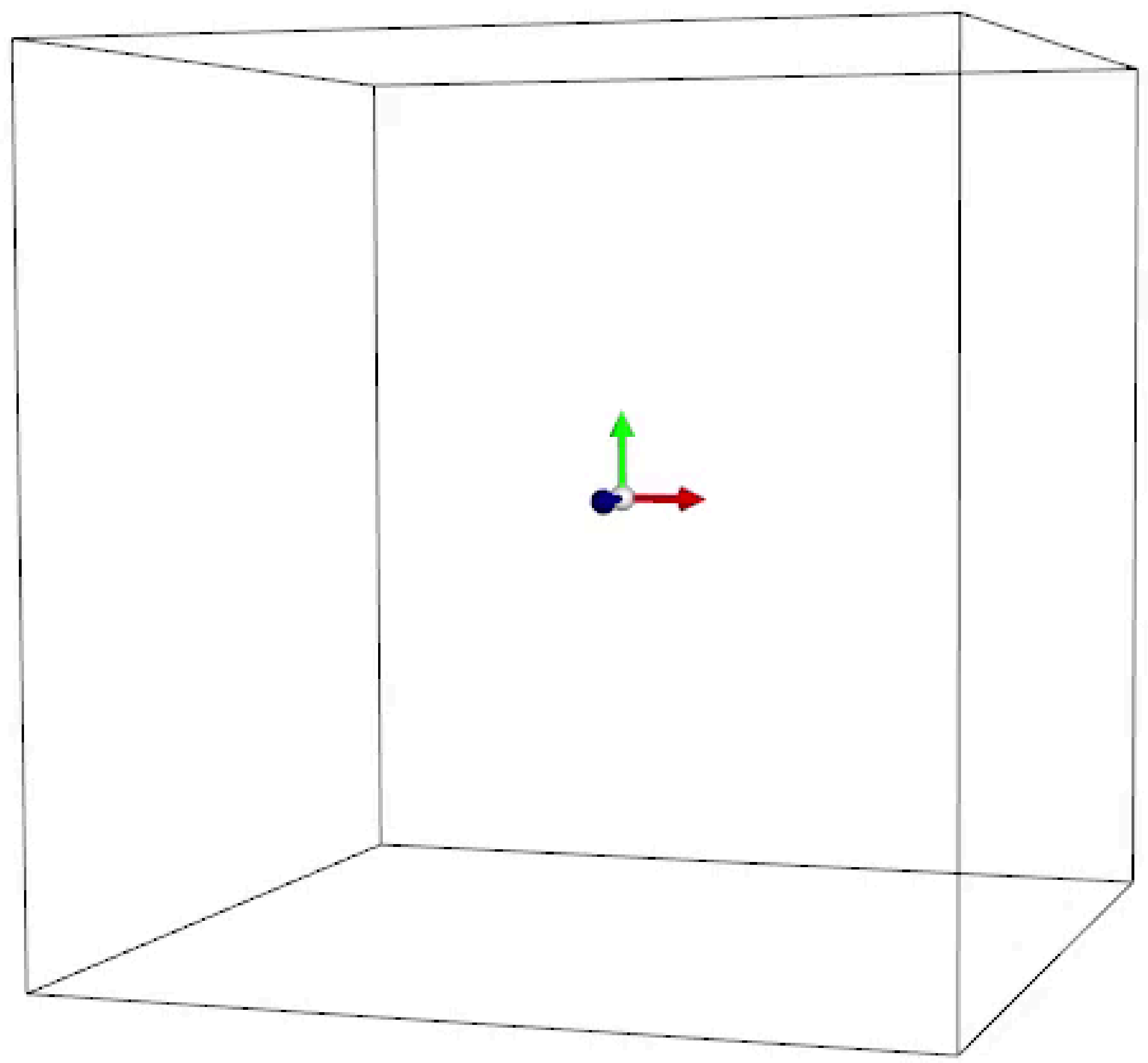
$v^{new} = \alpha\, v_{//} - \beta\, v_\perp$

$\alpha \in [0, 1]$ Restitution coefficient in // direction (friction)
$\beta \in [0, 1]$ Restitution coefficient in $\perp$ direction (impact)



Result: Collision response

Applying collision response on speed only

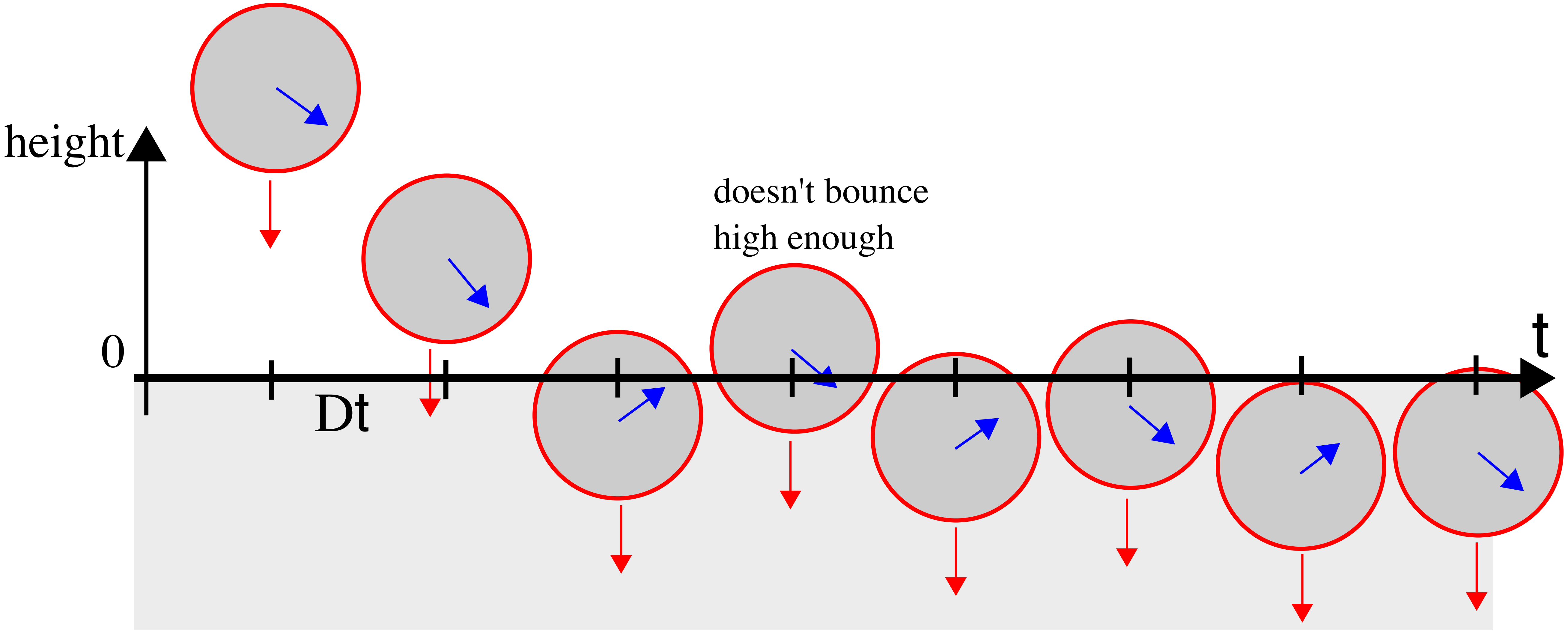


Result: Collision response - issue with discrete time

We assumed contact b/w sphere and plane

But: Exact contact never happens in discrete time

- *When collision is detected* $\rightarrow$ *already inside the wall*
- *Weight is still acting*



Collision response with plane : position

In real case (discrete time) no exact contact, but penetration $(p_i - a) \cdot n_i < r_i$

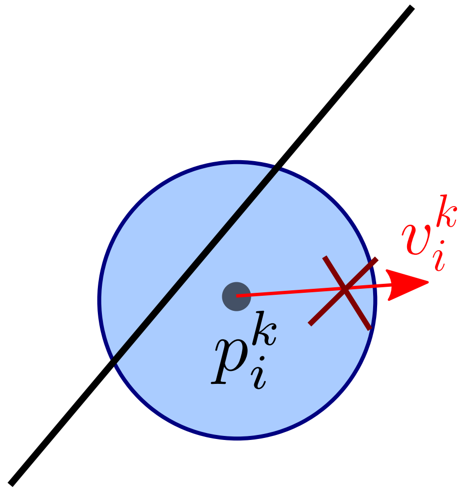
$\Rightarrow$ Need to compute collision response at contact point.

Three possibilities

(1) Update velocity to remove penetration

(+) Simple for well defined volumes

(-) Keep collision state

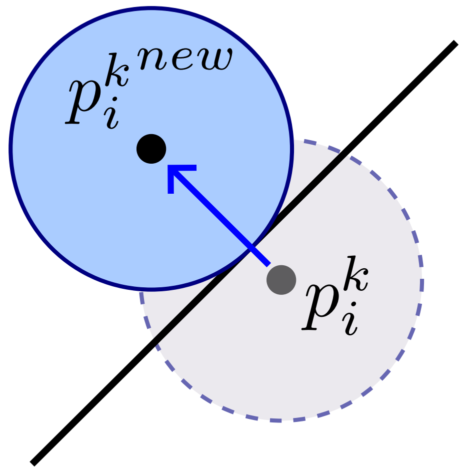


(2) Correct positions in projecting on the contact plane

Position Based Dynamics (PPD)

(+) Simple to implement

(-) Physically inaccurate

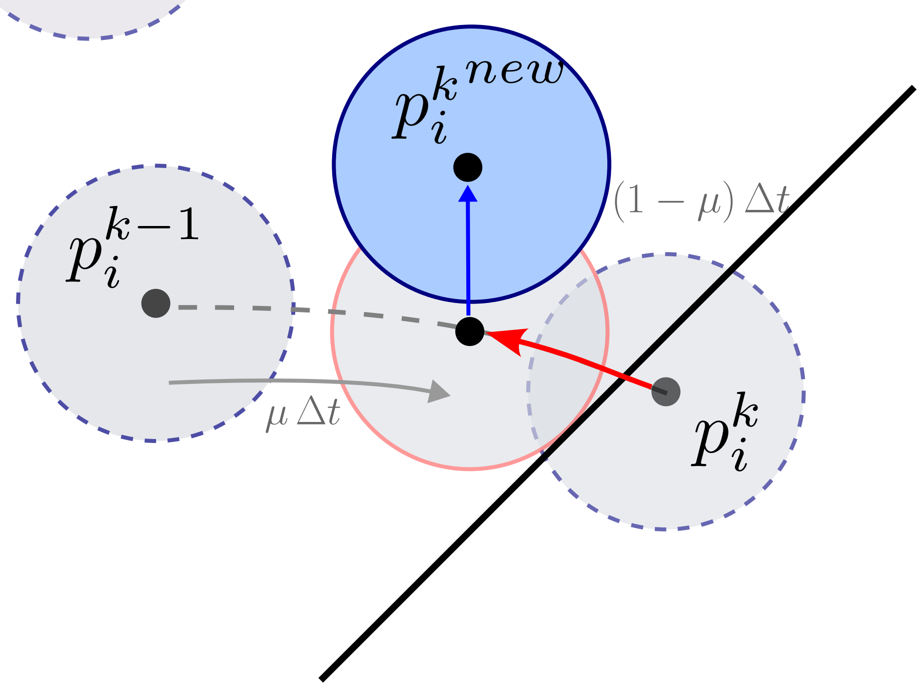


(3) Go backward in time to find exact instant of collision

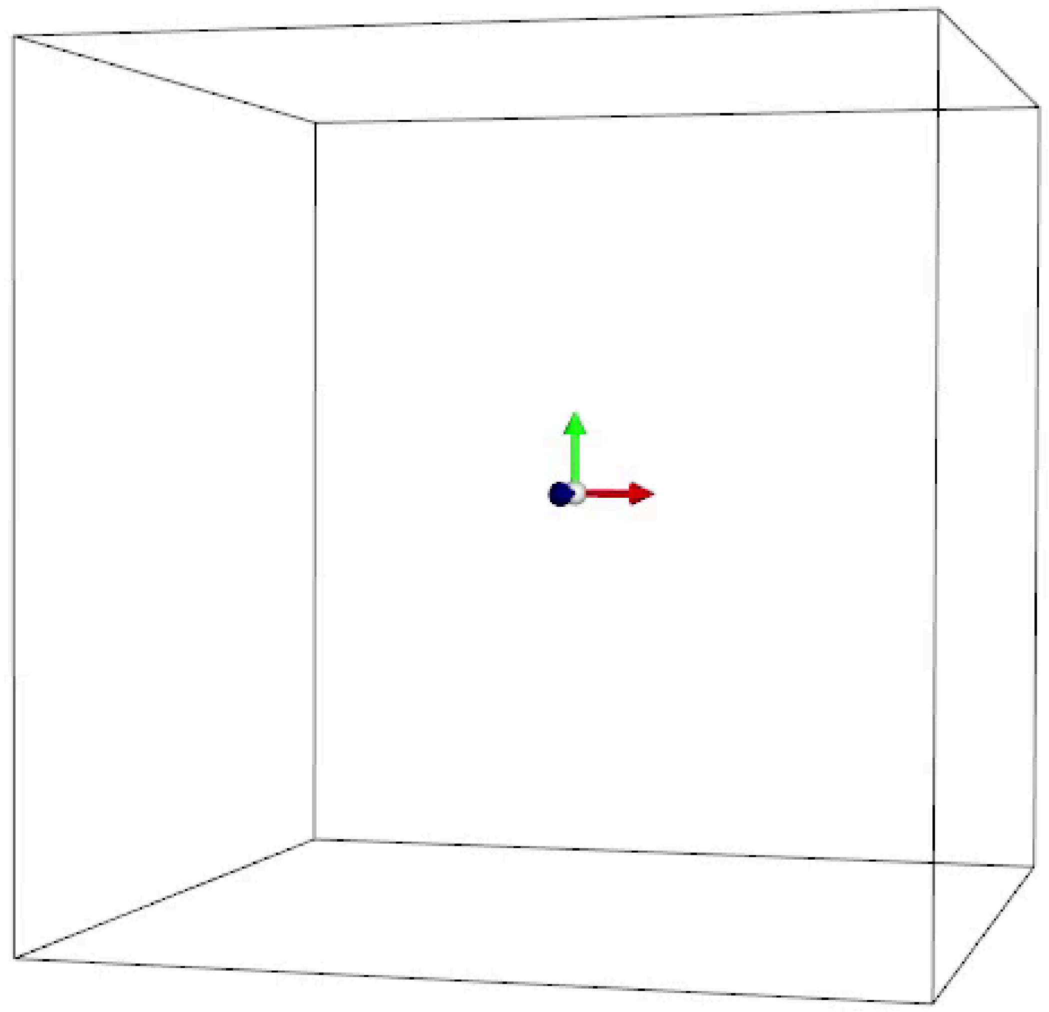
Continuous Collision Detection (CCD)

(+) Physically accurate

(-) Computationally heavy (binary search, etc.)



Result: After correction



Either avoiding negative oriented velocity

Velocity bounce if $(p_i - a) \cdot n < 0$ and $v_i \cdot n < 0$

Either position projection on surface contact

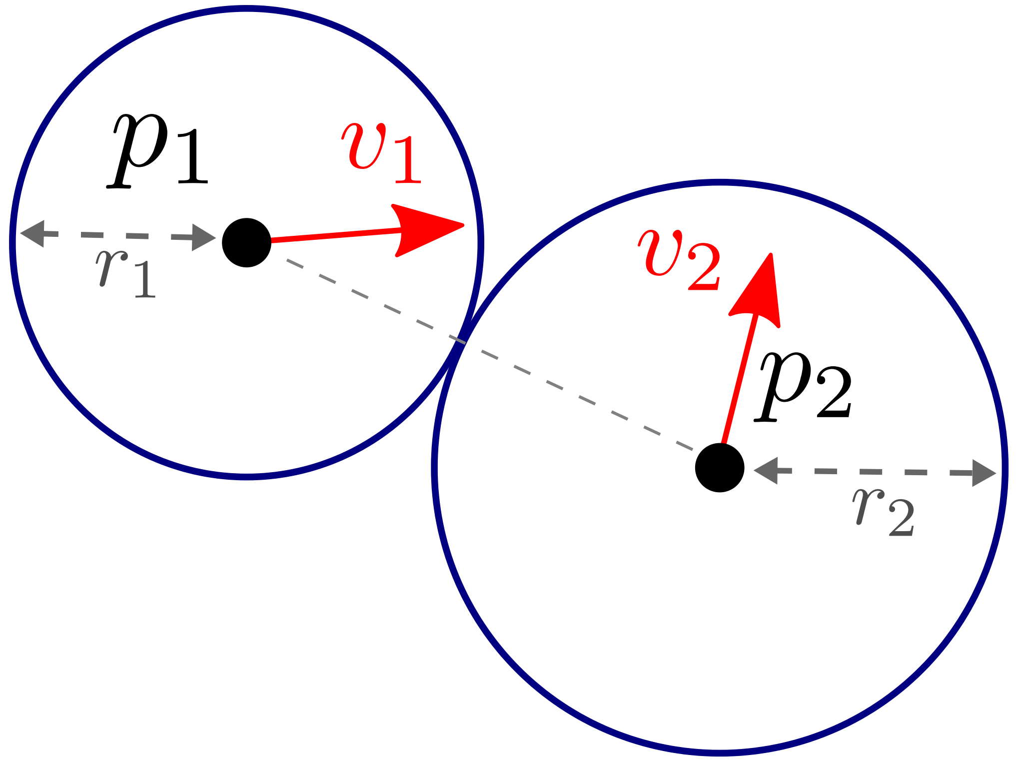
$p_i^{new} = p_i + d\,n$

$d = r_i - (p_i - a) \cdot n$: distance of penetration

Collision between spheres

Given 2 spheres (p_1, v_1, r_1, m_1), (p_2, v_2, r_2, m_2).

Collision when $\|p_1 - p_2\| \leq r_1 + r_2$



What happen with their velocities ?

$v_1 \rightarrow v_1^{new}$, $v_2 \rightarrow v_2^{new}$

Notion of impulse

An impulse J is the integrated force over time $J = \int_{t_1}^{t_2} F(t)\,\mathrm{d}t$

$\rightarrow$ results in a sudden change of speed (/momentum) in a discrete case

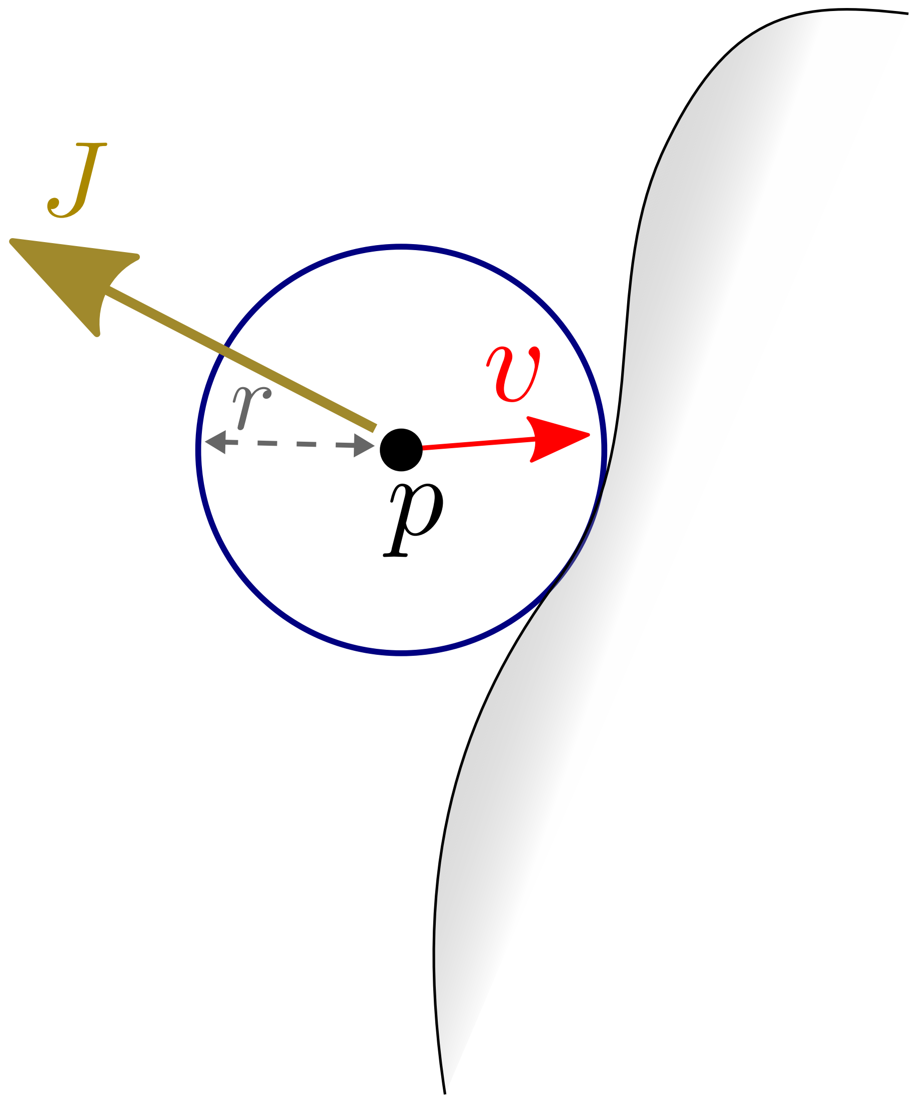
For a particle with constant mass

$$\int_{t_1}^{t_2} F(t)\,\mathrm{d}t = \int_{t_1}^{t_2} m\,a(t)\;\mathrm{d}t$$

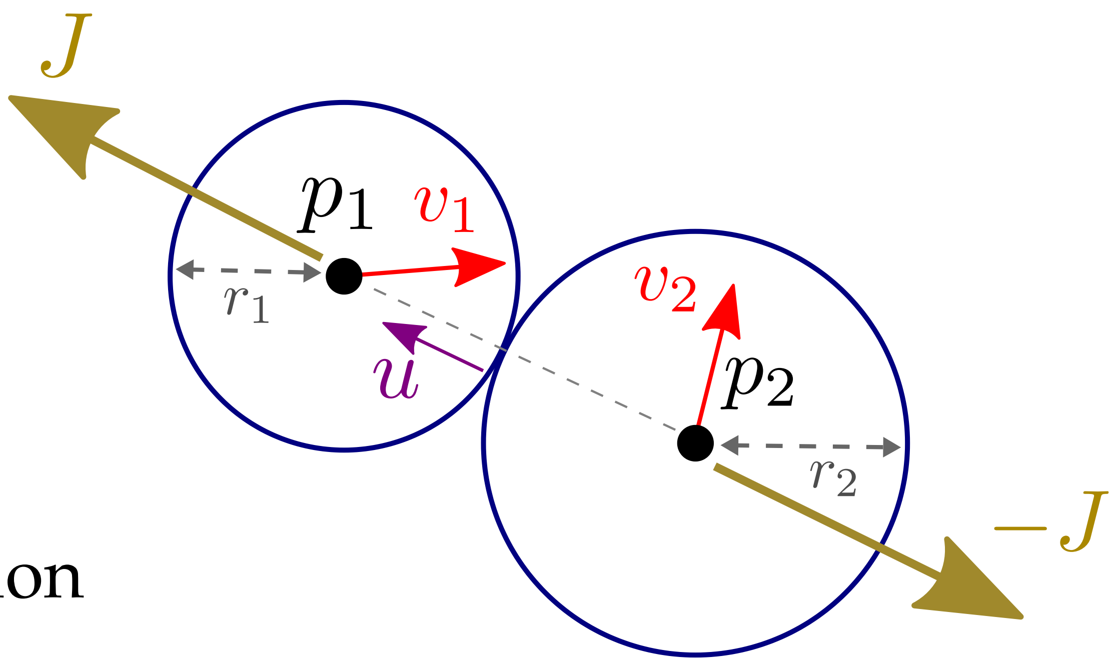
$$\Rightarrow J = m\,(v(t_2) - v(t_1))$$

For an impact $v \rightarrow v^{new}$

$$v^{new} = v + J/m$$



Two spheres in collision



Impulse orthogonal to the separating plane between the two surfaces

$J = j\,u, \quad u = (p_1 - p_2)/\|p_1 - p_2\|$

The system with the two spheres is preserving its linear momentum

$\Rightarrow$ Respective impulses j are equals in magnitude, and opposed in direction

$m_1 v_1 + m_2 v_2 = m_1 v_1^{new} + m_2 v_2^{new} \Rightarrow m_1(v_1^{new} - v_1) = -m_2(v_2^{new} - v_2) \Rightarrow J_1 = -J_2$

Assume collision of "hard spheres" = "Elastic collision"

= No loss of energy, conservation of kinetic energy of the system

$$\Rightarrow j = 2\frac{m_1\,m_2}{m_1 + m_2}\,(v_2 - v_1)\cdot u$$

$1/2\,m_1\,v_1^2 + 1/2\,m_2\,v_2^2 = 1/2\,m_1\,(v_1^{new})^2 + 1/2\,m_2\,(v_2^{new})^2$

$\Rightarrow m_1 v_1^2 + m_2 v_2^2 = m_1\left(v_1 + \frac{j}{m_1}u\right)^2 + m_2\left(v_2 - \frac{j}{m_2}u\right)^2$

$\Rightarrow 0 = 2\,j\;v_1 \cdot u + \frac{j^2}{m_1} - 2\,j\;v_2 \cdot u + \frac{j^2}{m_2}$

$\Rightarrow j = \frac{2}{1/m_1 + 1/m_2}\,(v_2 - v_1)\cdot u$

Two spheres in collision

$$v_1^{new} = v_1 + j/m_1\, u = v_1 + 2\frac{m_2}{m_1+m_2}((v_2 - v_1)\cdot u)\, u$$
$$v_2^{new} = v_2 - j/m_2\, u = v_2 - 2\frac{m_1}{m_1+m_2}((v_2 - v_1)\cdot u)\, u$$

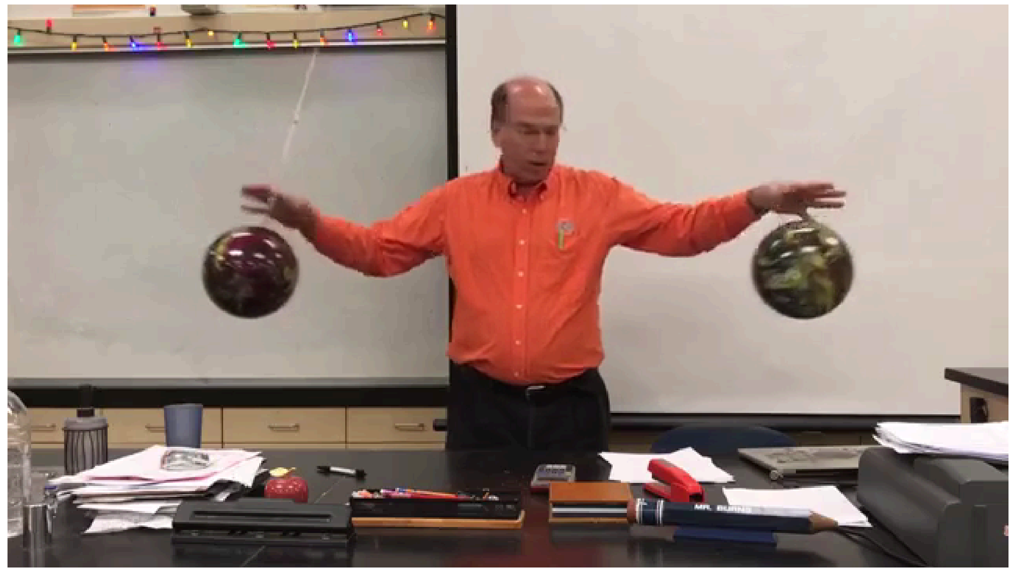
Rem. If $m_1 = m_2$: Switch their $\perp$ speeds

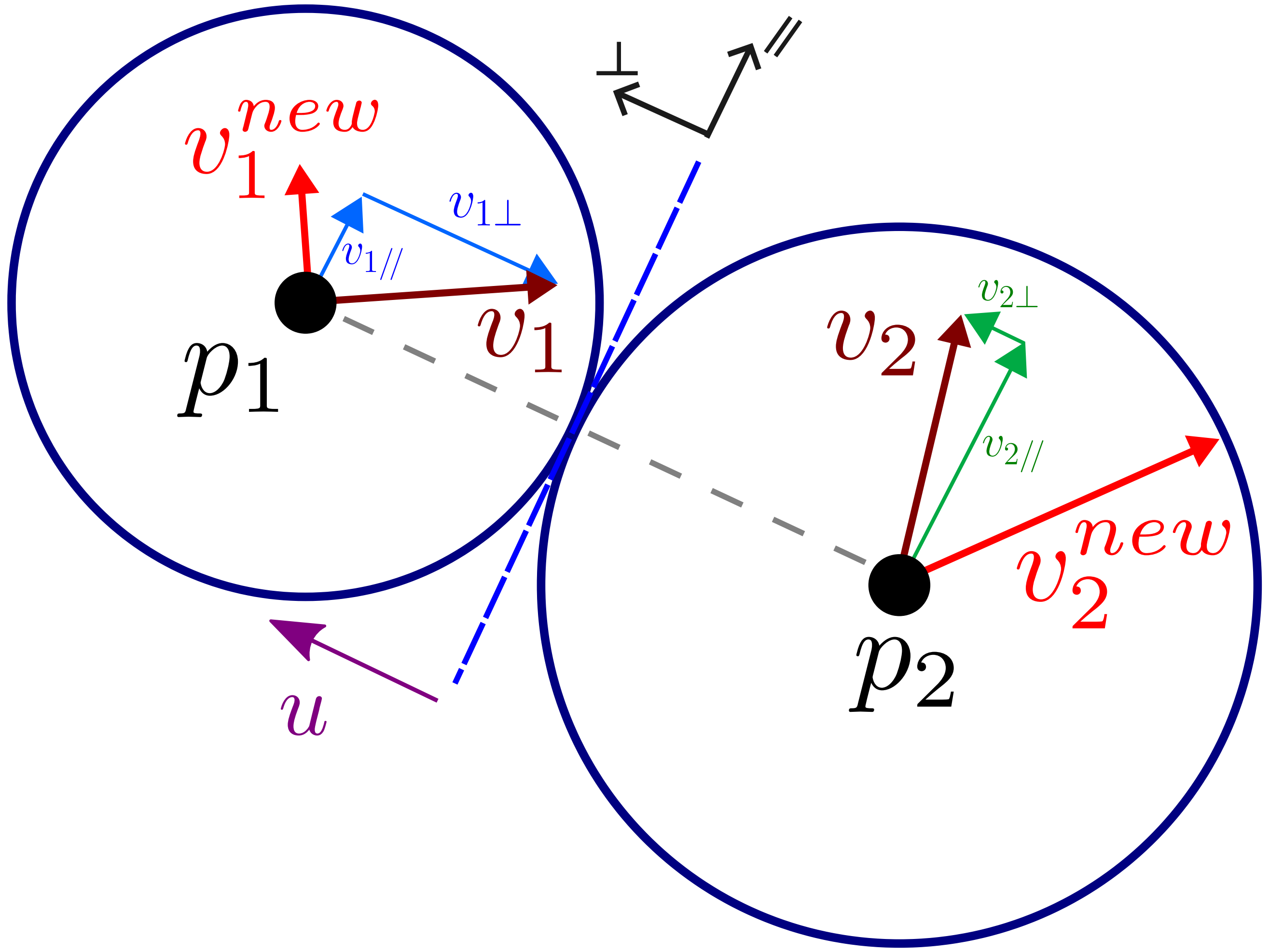
$$v_1^{new} = v_1 + ((v_2 - v_1)\cdot u)\, u = v_{1//} + v_{2\perp}$$
$$v_2^{new} = v_2 - ((v_2 - v_1)\cdot u)\, u = v_{2//} + v_{1\perp}$$

with $v_\perp = (v\cdot u)u$ and $v_{//} = v - (v\cdot u)u$

Can use restitution coefficient and attenuation $\alpha \in [0,1]$

$$v_1^{new} = \alpha\,(v_{1//} + v_{2\perp})$$
$$v_2^{new} = \alpha\,(v_{2//} + v_{1\perp})$$





Handling collision between two spheres

Position Based

1. Detect collision $\|p_1 - p_2\| \leq r_1 + r_2$

If collision then:

2a. Update Velocity

Elastic collision (/bouncing)

$v_1 = \alpha\,(v_1 + j/m_1\,u)$

$v_2 = \alpha\,(v_2 - j/m_2\,u)$

2b. Correct position (project on contact surface)

$p_1 = p_1 + d/2\,u$

$p_2 = p_2 - d/2\,u$

$d = r_1 + r_2 - \|p_1 - p_2\|$: Collision depth

Velocity Based

1. Detect collision $\|p_1 - p_2\| \leq r_1 + r_2$

If collision then:

2. Update Velocity

Elastic collision (/bouncing)

If $v \cdot n < 0$

$v_1 = \alpha\,(v_1 + j/m_1\,u)$

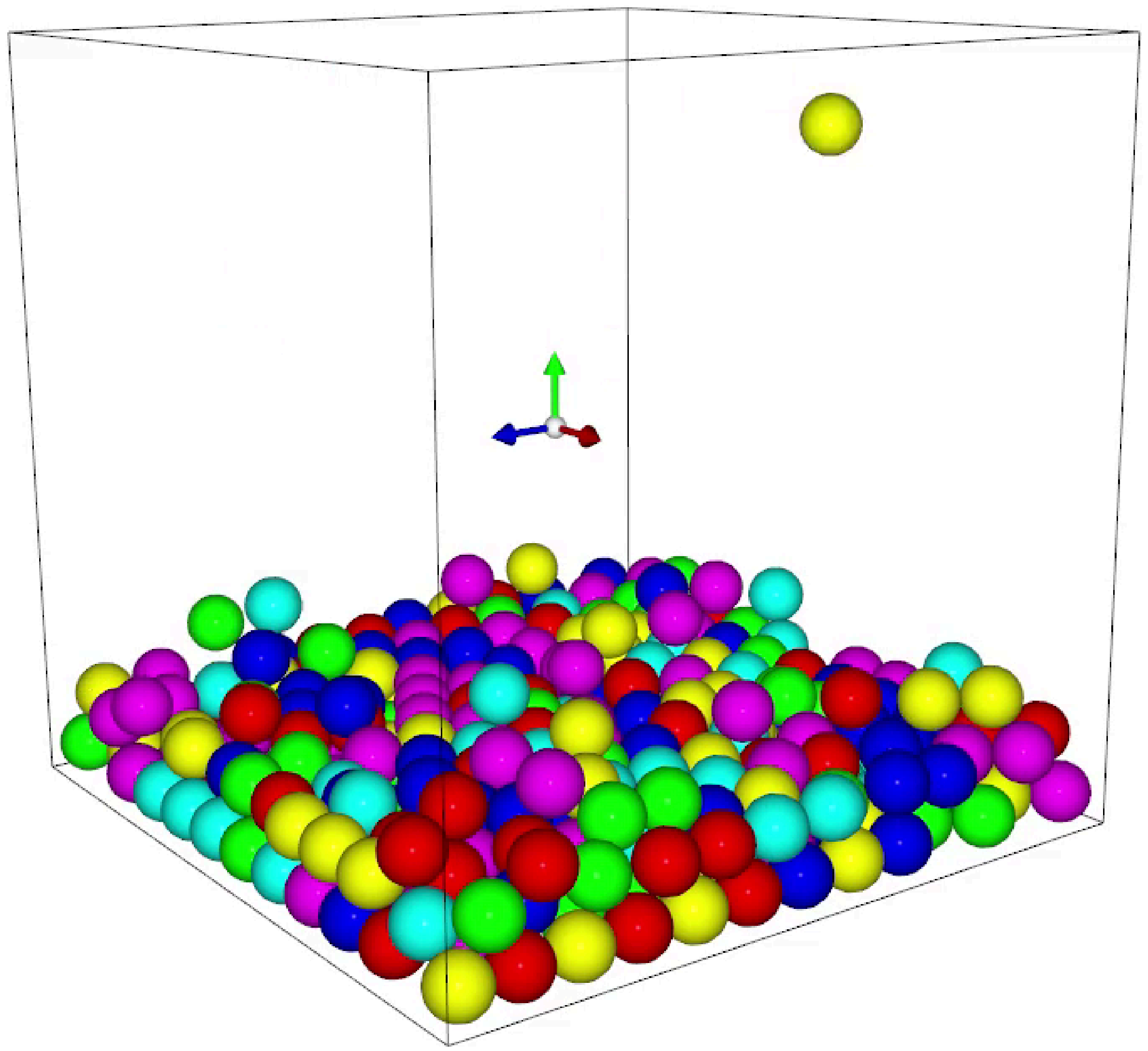
$v_2 = \alpha\,(v_2 - j/m_2\,u)$

Summary with multiple particles

Position Based

A. Update position and velocity from field forces (gravity, friction)
B. Handle collision (velocity+position) between particles
C. Handle collision (velocity+position) with walls

(+) Good collision avoindance for the last constraints
(-) Jittering appears in stacked spheres



Velocity Based

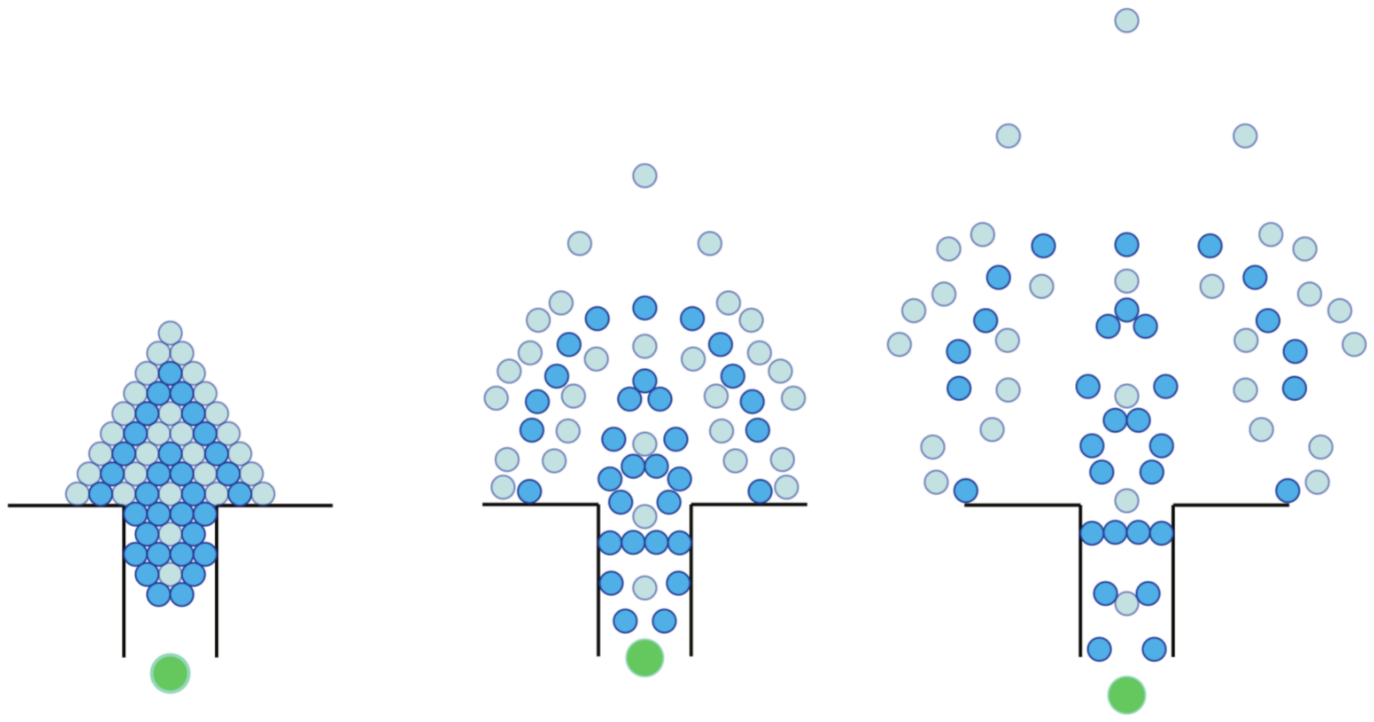
A. Update velocity from field forces (gravity, friction)
B. Handle collision (with velocity) between particles, and walls
C. Cancel velocity component contributing to penetration
D. Update position from current velocity

(+) Smooth and stable motion
(-) Existing collision persists

Multiple collisions

Pairwise collisions ⇒ no global collision free state

- Correcting one collision may induce new collisions.
- Order of correction does matter



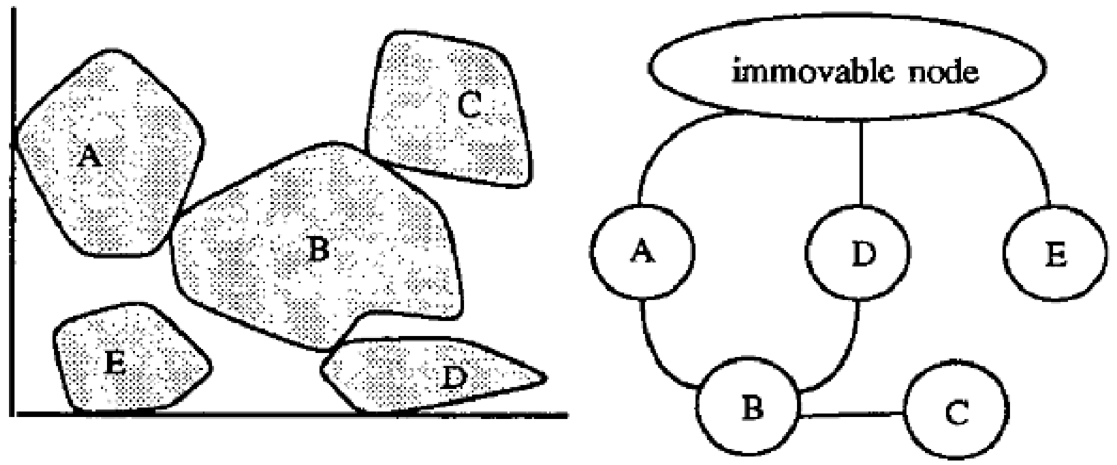
Reducing time-step help, Iterating over mutliple pass help

But correct solution in all cases is complex → global approach

- Precompute contact graph

  explicit shock propagation management
- Global constraint-based method

  Impulse: $n_i \cdot (v_i - v_j) \geq 0$
  Momentum preservation: $m_i v_i - m_j v_j = 0$
  Energy preservation/dissipation
  ⇒ Linear Complementarity Program, Gauss Seidel, etc.



[Realistic Animation of Rigid Bodies. J. Hahn. SIGGRAPH 1988.]

[Collision Detection and Response for Computer Animation. M. Moore and J. Wilhelms. Computer Graphics 1988.]

[Reflections on Simultaneous Impact. B. Smith et al. SIGGRAPH 2012]

[Guaranteed Resolution of Simultaneous Rigid Body Impact. E. Vouga. ACM SIGGRAPH 2017]