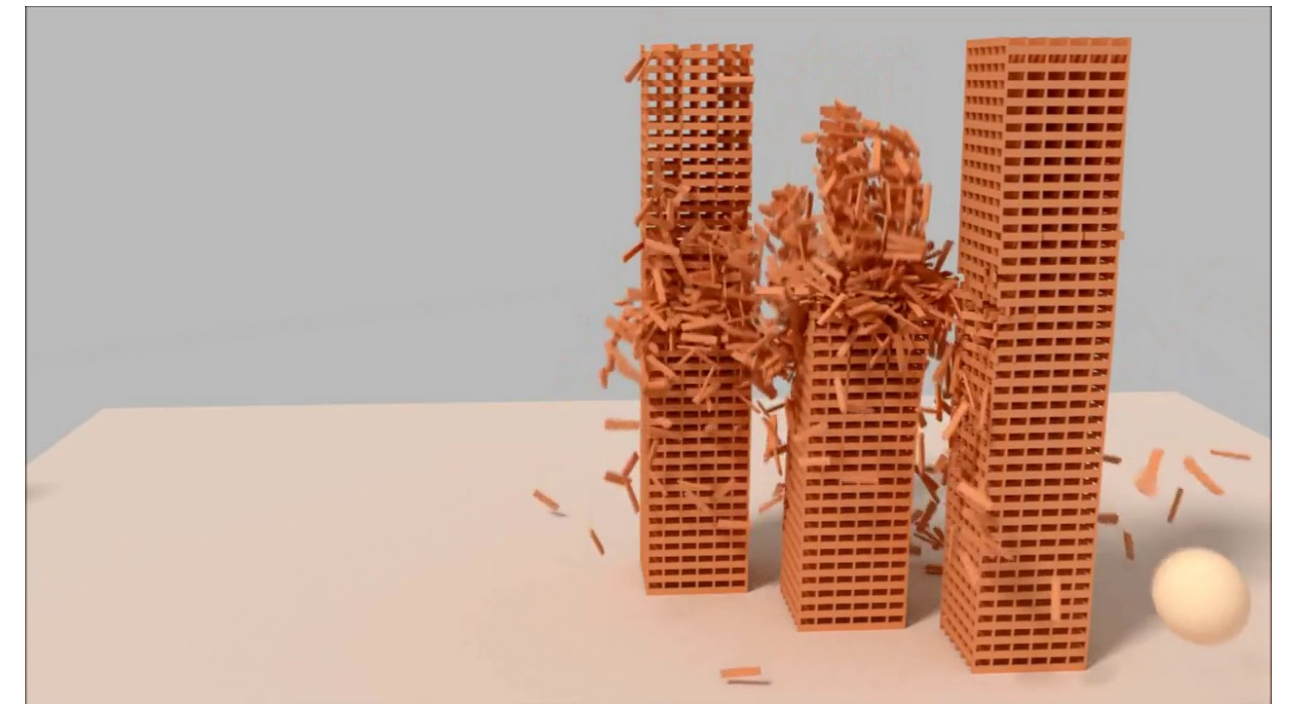
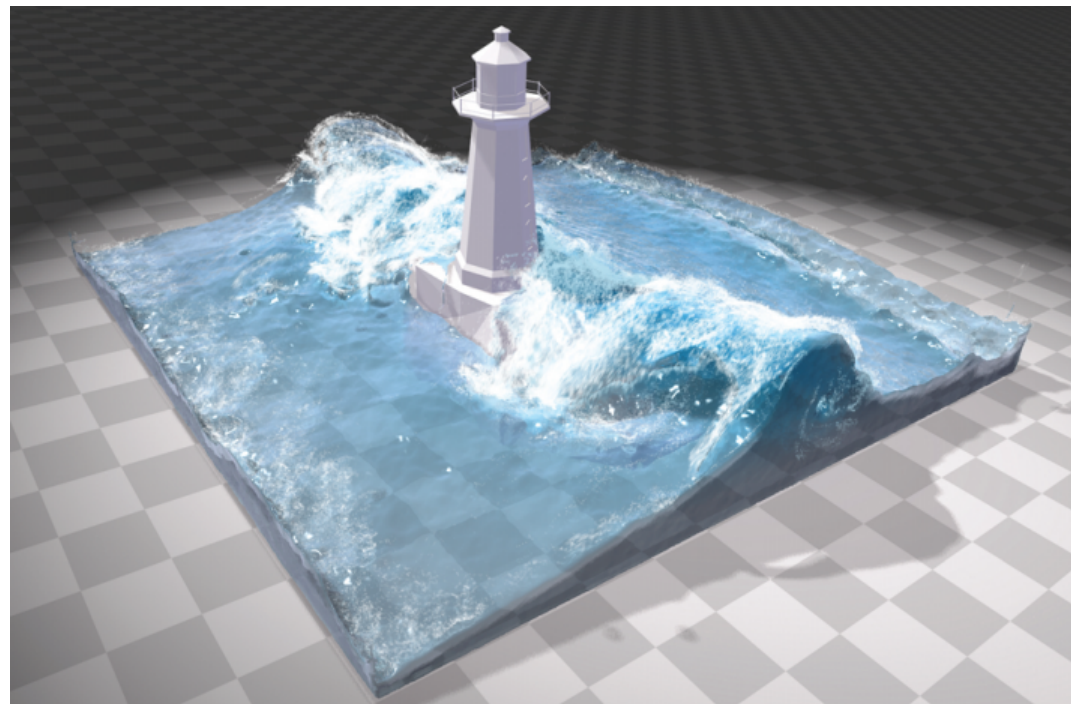


Physically based simulation - Models

When physically based simulation is needed

- Accurate dynamics
- Tedious 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 controlling 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

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

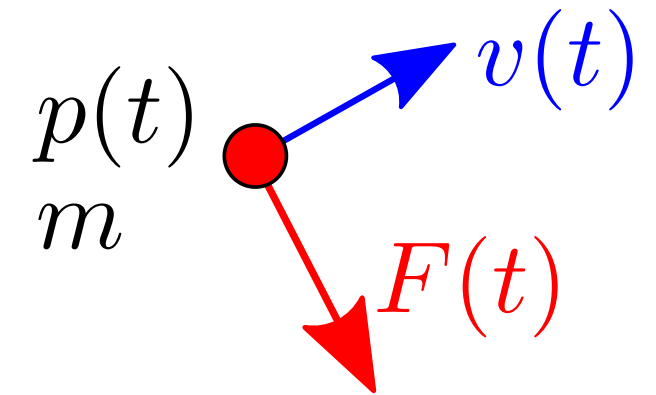
- Fundamental principle of dynamics

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.)

- Lagrangian, or Hamiltonian (reduced coordinates)



Physically-based particle system

3. Numerical Solution

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

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

- Discretize in time $t^k = k h$, $h = \Delta t = \text{time step}$.
 $\Rightarrow$ 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).

Fundamental models

- 1- Particles
- 2- Rigid bodies**
- 3- Continuum models

Rigid body description

- Solid defined within a domain $\Omega \subset \mathbb{R}^3$
- With a density of mass $\rho(p_i)$ at each point $p_i \in \Omega$

- Total mass of the solid m

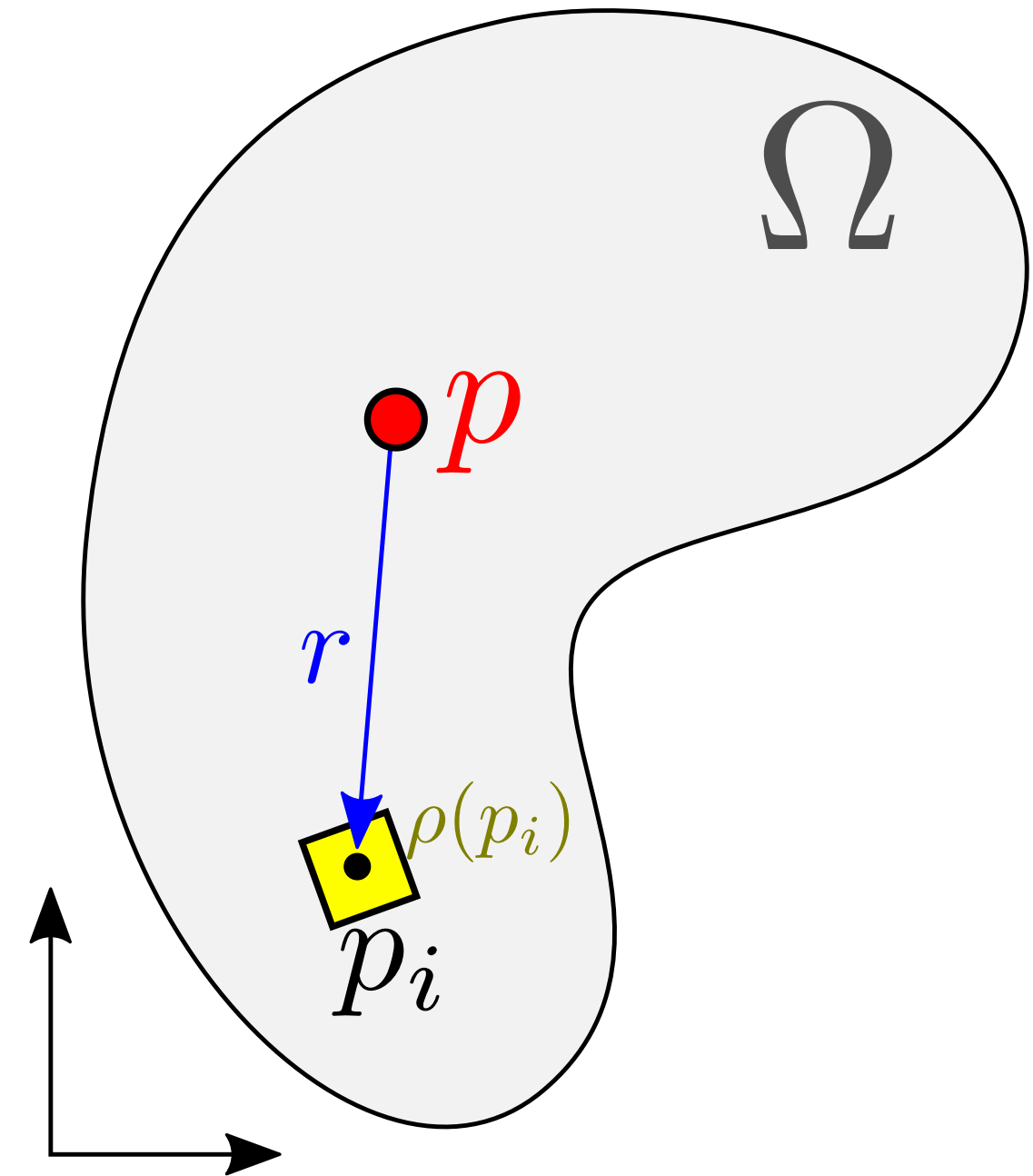
$$m = \int_{p_i \in \Omega} \rho(p_i) d\Omega$$

- Position of the center of mass (com) p

$$p = \frac{1}{m} \int_{p_i \in \Omega} \rho(p_i) p_i d\Omega$$

- Relative position of a position p_i with respect to com

$$r = p_i - p$$



Position and speed of a point on the rigid body

The center of mass has, at time t ,

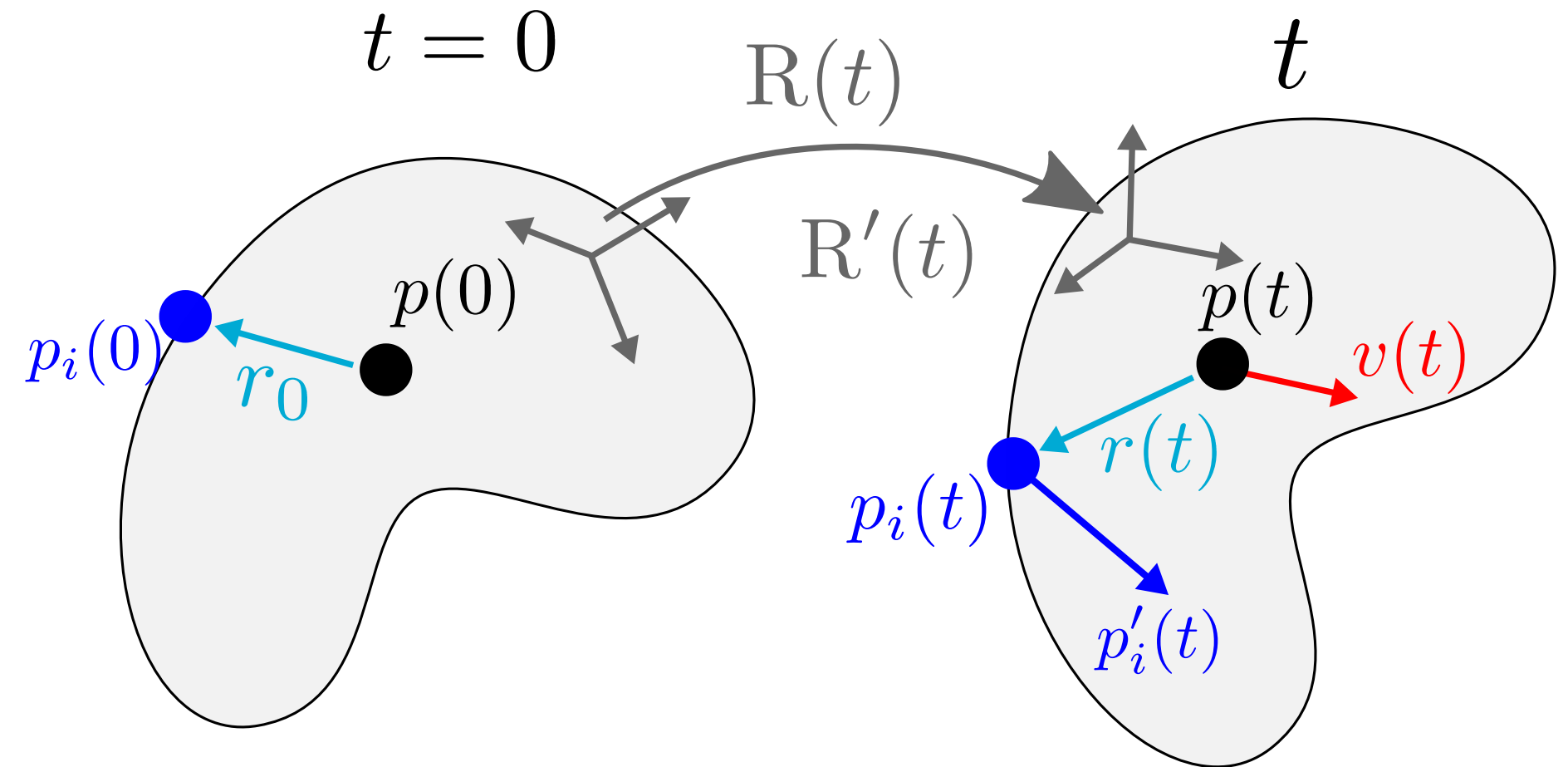
- a position $p(t)$
- a velocity $p'(t) = v(t)$

The body has an orientation $R(t)$

A point of the rigid body has

- a position $p_i(t) = p(t) + R(t) r_0$
with $r_0 = p_i(0) - p(0)$

- a speed $p'_i(t) = v(t) + R'(t) r_0$



Angular speed

Speed of p_i : $p'_i(t) = v(t) + R'(t) r_0$

Introduce angular speed $\omega \in \mathbb{R}^3$ such that

$$p'_i(t) = v(t) + \omega(t) \times r(t)$$

$\omega \simeq$ vector expressing the instantaneous rotation of $r(t)$

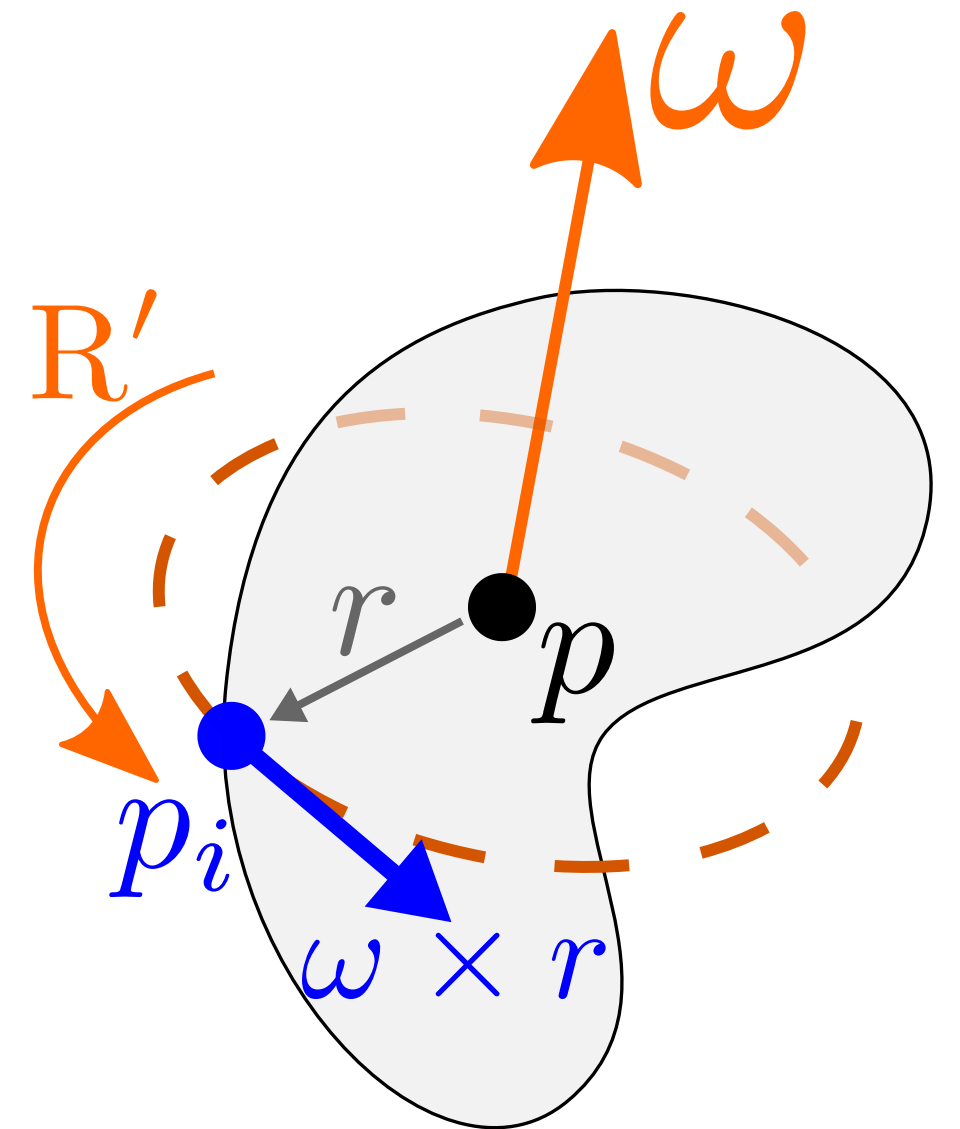
By identification $R'(t) r_0 = \omega(t) \times r(t)$

$$\Rightarrow R'(t) r_0 = \omega(t) \times (R(t) r_0)$$

Matrix expression of $\omega = (\omega_x, \omega_y, \omega_z)$

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

$$\Rightarrow R'(t) = \hat{\omega}(t) R(t)$$



Rigid body kinematics

Similarly to particles

- Position of the com: $p(t)$
- Velocity of the com: $v(t) = p'(t)$

Linear Momentum: $P(t) = m v(t)$ $\left(= \int_{\Omega} \rho v_i(t) d\Omega \right)$

Specific to rigid body

- Orientation of the body: $R(t) \in \mathbb{R}^{3 \times 3}$
- Angular velocity of the body: $\omega(t)$ such that $\hat{\omega} = R'(t) R^T(t)$

Angular Momentum: $L(t) = I(t) \omega(t)$

with $I(t)$: Inertia tensor

$$I(t) = R(t) I_0 R^T(t)$$

$$I_0 = \int_{r \in \Omega} \rho(r) (r^T r I_d - r r^T) d\Omega$$

Mass: resistance to change of speed (of the com)

Inertia: resistance to change of angular speed

Inertia tensor

Defining inertia tensor formulation from angular momentum definition

Angular momentum expressed with respect to an arbitrary point p_0 : $r(p_i) = p_i - p_0$

$$L = \int_{\Omega} r \times (\rho r') d\Omega = \int_{\Omega} \rho r \times (p' + \omega \times r) d\Omega \quad (\text{first part sum to } 0)$$

$$\Rightarrow L = \int_{\Omega} \rho r \times \omega \times r d\Omega = \int_{\Omega} \rho r \times (-r \times \omega) d\Omega$$

$$\Rightarrow L = \underbrace{\left(\int_{\Omega} \rho \hat{r} \hat{r}^T d\Omega \right)}_I \omega = I \omega \quad \text{with } \hat{r} = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix}$$

Changing the reference frame

$$\Rightarrow L = \left(\int_{\Omega} \rho (R\hat{r}_0) (R\hat{r}_0)^T d\Omega \right) \omega = R \underbrace{\left(\int_{\Omega} \rho(r) \hat{r}_0 \hat{r}_0^T d\Omega \right)}_{I_0} R^T \omega = R I_0 R^T \omega$$

Inertia tensor properties

$$I = \int_{r \in \Omega} \rho(r) \begin{pmatrix} r_y^2 + r_z^2 & -r_x r_y & -r_x r_z \\ -r_x r_y & r_x^2 + r_z^2 & -r_y r_z \\ -r_x r_z & -r_y r_z & r_x^2 + r_y^2 \end{pmatrix} d\Omega = \int_{r \in \Omega} \rho(r) (r^T r \text{ Id} - r r^T) d\Omega$$

- I is usually expressed at the center of mass p
- I depends on the body orientation. Given a rotation R : $I = R I_0 R^T$
 $\Rightarrow$ compute once I_0 in a rest position, then update it using R
- There exist a frame in which I is diagonal (principle axes of inertia).
Corresponds to eigenvectors of matrix I .

Dynamics: Forces and torques on a solid

- Given a local force $f(p_i)$ acting on a position $p_i \in \Omega$
- Contribute to 2 global components applied on the body:

- Total **net force** F applied on the shape

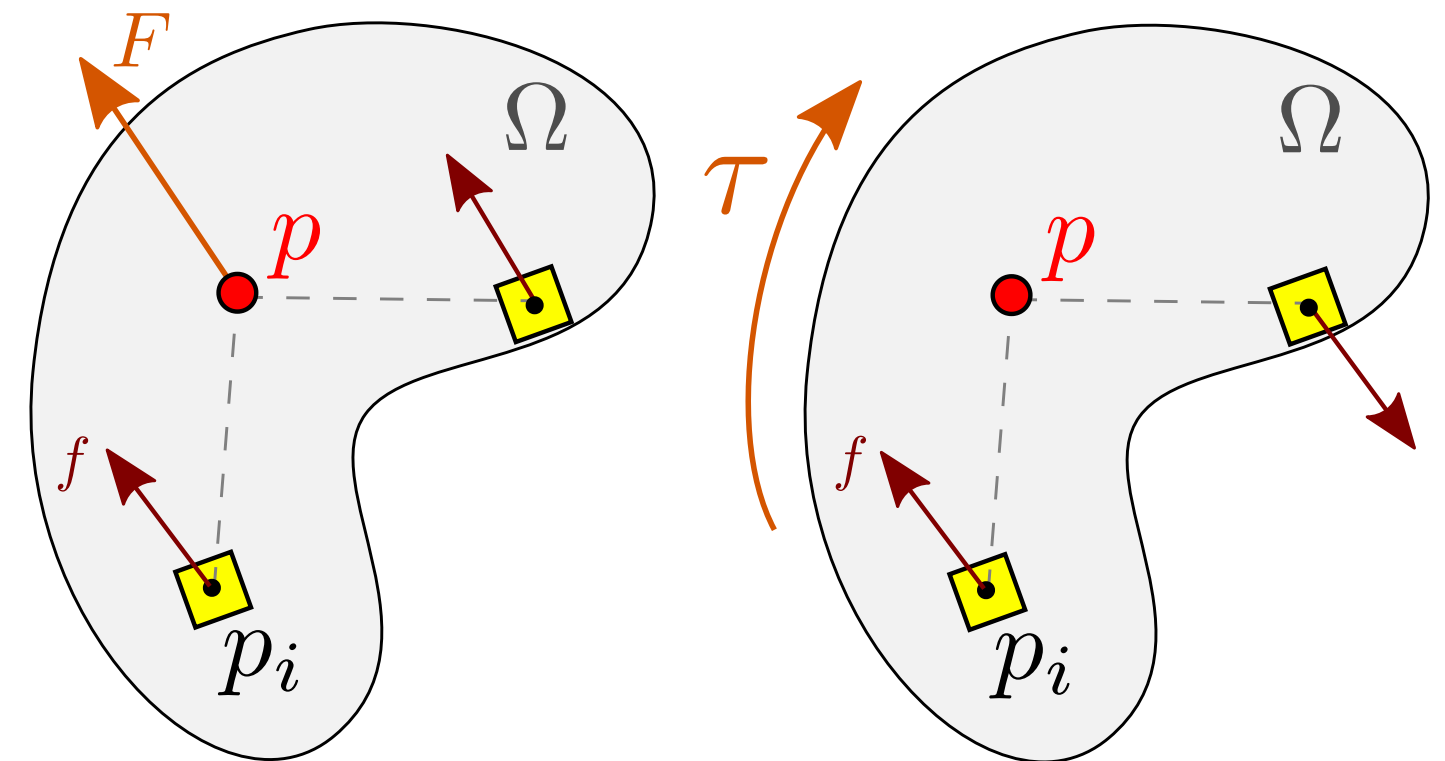
Induce change of linear momentum (a displacement of COM)

$$F = \int_{p_i \in \Omega} f(p_i) d\Omega$$

- **Torque** τ applied on the body

Induce change of angular momentum (spin of the solid)

$$\tau = \int_{p_i \in \Omega} (p_i - p) \times f(p_i) d\Omega$$



Dynamics

Similarly to particles

Force F is related to the change of linear momentum $F(t) = P'(t) = (m v)'(t)$

Specific to rigid bodies

Torque τ is related to the change of angular momentum $\tau(t) = L'(t) = (I \omega)'(t)$

Equation of Motion

Fundamental principle of dynamics for rigid body

$$\frac{d}{dt} \begin{pmatrix} p \\ P \\ R \\ L \end{pmatrix} = \begin{pmatrix} v \\ F \\ \hat{\omega} R \\ \tau \end{pmatrix}$$

Rigid solid dynamics in practice

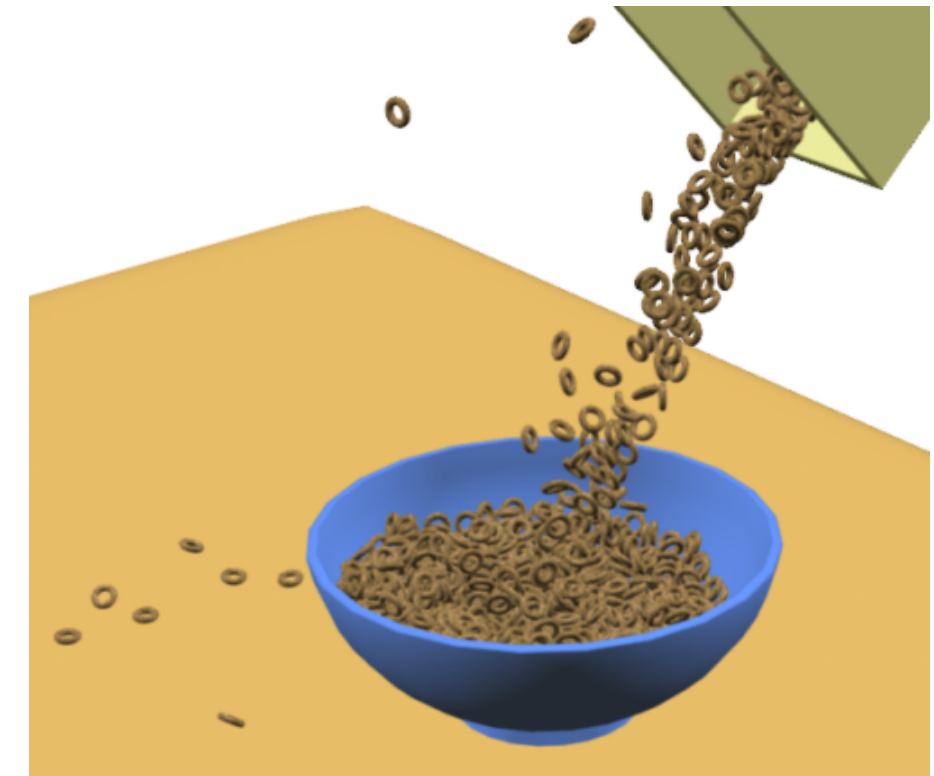
1. Initial condition

- $p(t_0), v(t_0), \mathbf{R}(t_0), \omega(t_0)$ given as initial condition
- Precompute $I_0 = I(t_0)$
- Compute $L(t_0) = I_0 \omega(t_0)$

2. Temporal Evolution

Iterate over time t_k

- Compute total force $F(t_k)$ and torque $\tau(t_k)$
- Compute $I(t_k) = \mathbf{R}(t_k) I_0 \mathbf{R}^T(t_k)$
- Compute $\omega(t_k) = I(t_k)^{-1} L(t_k)$
- Numerical integration updating state vector
 $\rightarrow (p(t_{k+1}), P(t_{k+1}), \mathbf{R}(t_{k+1}), L(t_{k+1}))$
- *Handle collision*



$$\begin{pmatrix} p'(t) \\ P'(t) \\ \mathbf{R}'(t) \\ L'(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ F(t) \\ \hat{\omega}(t) \mathbf{R}(t) \\ \tau(t) \end{pmatrix}$$

$$I(t) = \mathbf{R}(t) I_0 \mathbf{R}^T(t)$$

$$L(t) = I(t) \omega(t)$$

$$P(t) = m v(t)$$

Side note: Use of quaternion

Relation $\mathbf{R}'(t) = \hat{\omega}(t) \mathbf{R}(t)$ may lead to numerical drift from rotation matrix
ex. $\mathbf{R}^{k+1} = (\text{Id} + h \hat{\omega}^k) \mathbf{R}^k$ (explicit scheme)

Using quaternion leads to more robust behavior

- Quaternion expression: $q'(t) = \frac{1}{2} q_{\omega}(t) q(t)$, with $q_{\omega}(t) = (\omega(t), 0)$
- Quaternion is forced to keep a unit norm

$$\text{ex. } \begin{cases} q^{k+1} = q^k + \frac{1}{2} q_{\omega}^k q^k \\ q^{k+1} = q^{k+1} / \|q^{k+1}\| \end{cases}$$

Collisions in rigid bodies

Collisions at position p change both linear and angular velocity

Use of impulse (sudden change of velocity) J .

$$J = \int_{t_0}^{t_0 + \Delta t} F(t) dt$$

Impulse split into

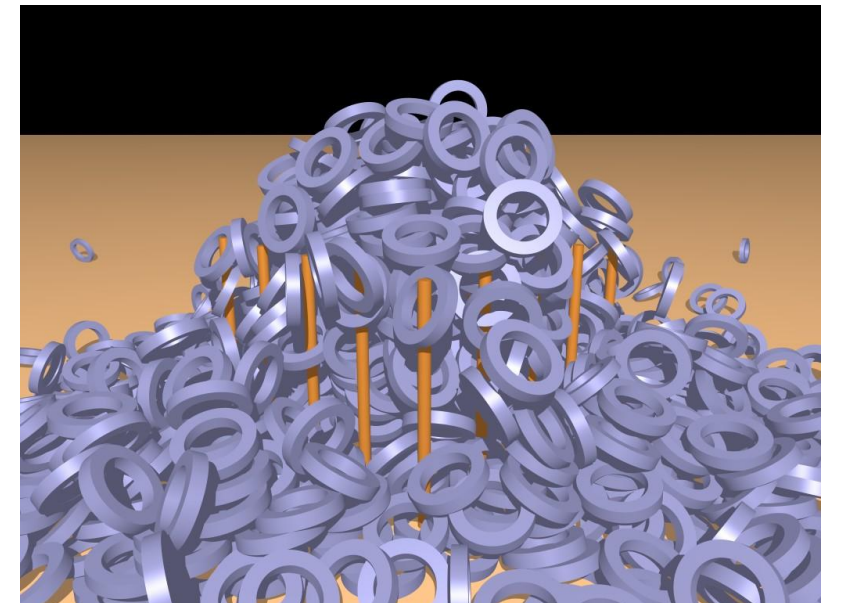
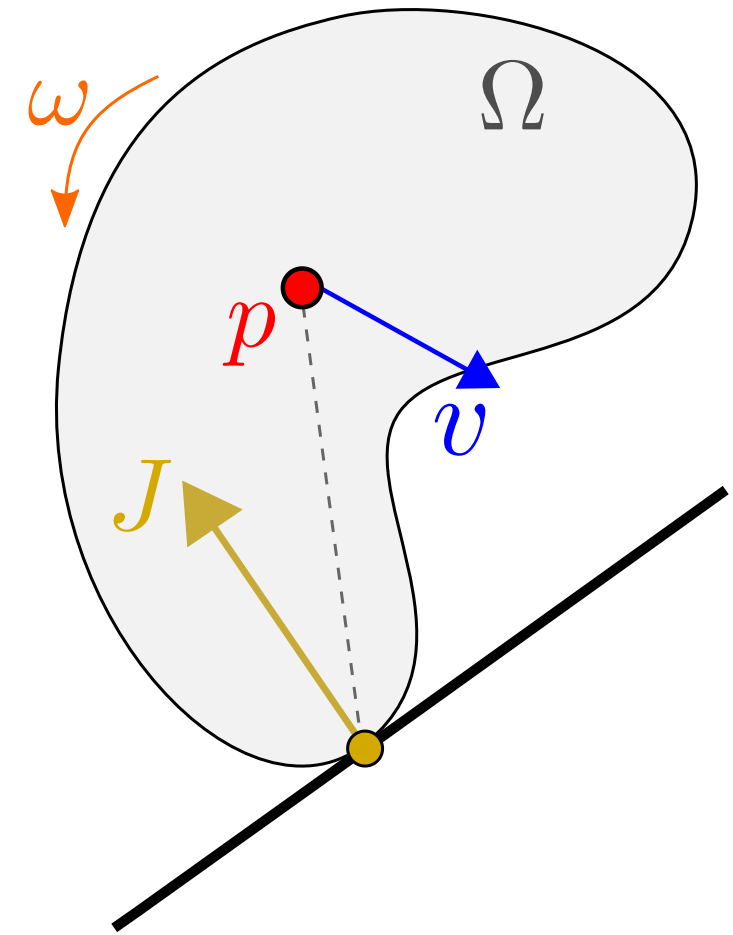
- Force impulse $\Delta P = m \Delta v = J$
- Torque impulse $\Delta L = I \Delta \omega = (p - p_{com}) \times J$

Elastic collision between two solids at position p_i :

$J = j n$, n : normal of the separating planes

$$j = (v_1(p_i) - v_2(p_i)) \cdot n / K$$

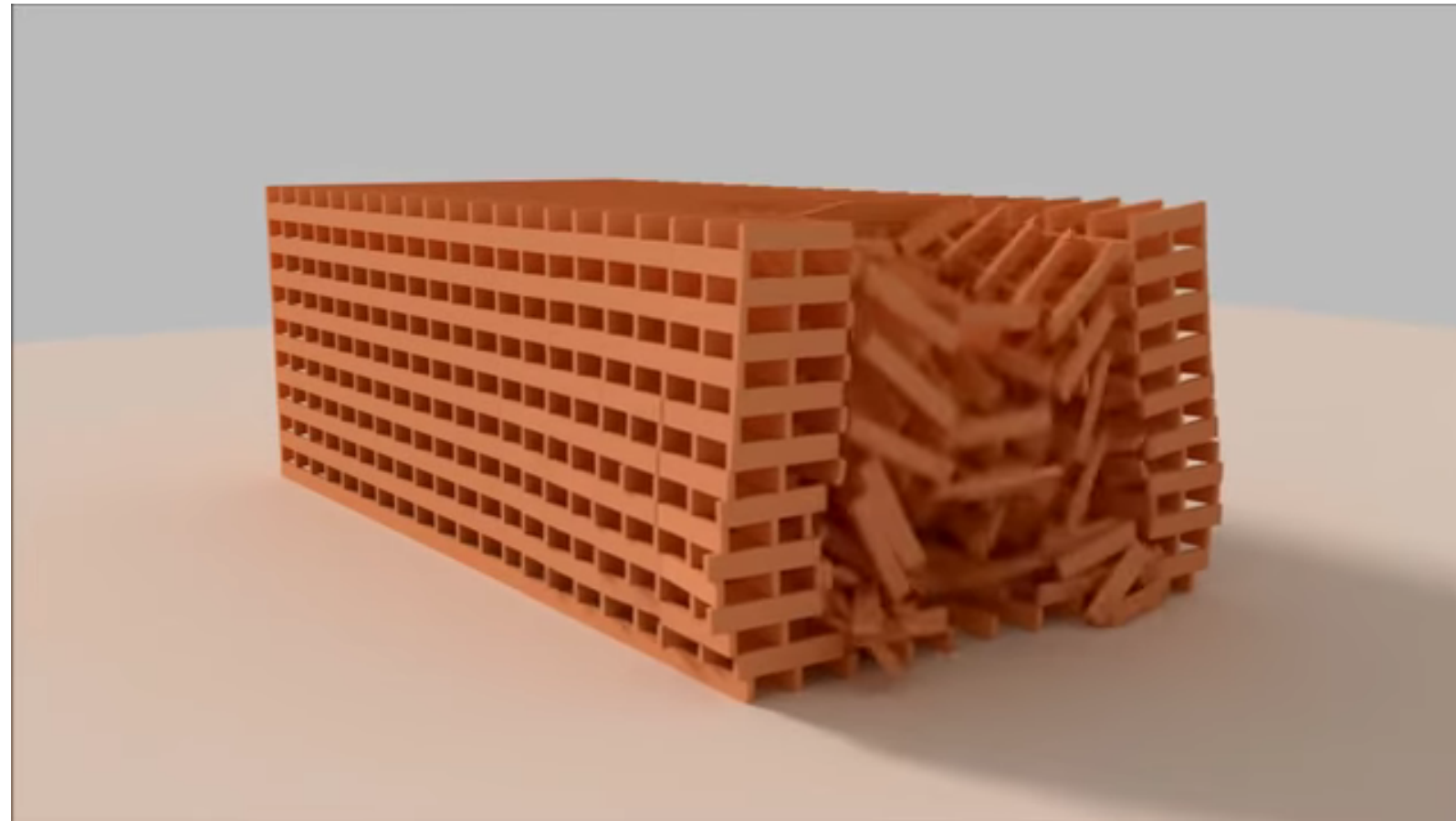
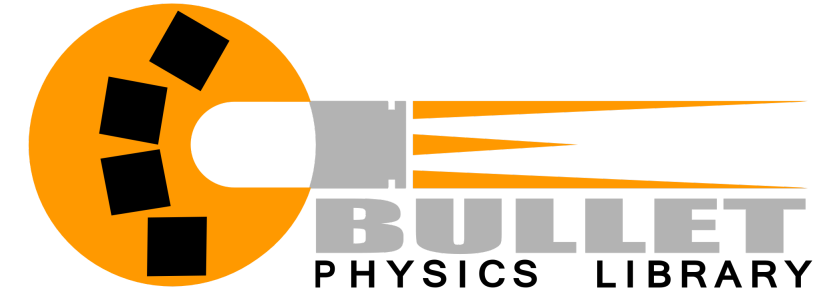
$$K = 1/m_1 + 1/m_2 + n \cdot (I_1^{-1}(r_1 \times n) \times r_1 + I_2^{-1}(r_2 \times n) \times r_2)$$



More details [\[D. Barff. Physically Based Modeling. SIGGRAPH Course Notes 1999\]](#)

Rigid bodies usage

- Standard usage for rigid bodies motions
 - Limited to non-deformable shapes
- Common in VFX (explosions), and *simulation games* (cars, airplanes, etc).
- Standard library: [Bullet physics](#) (ex. used in Blender).



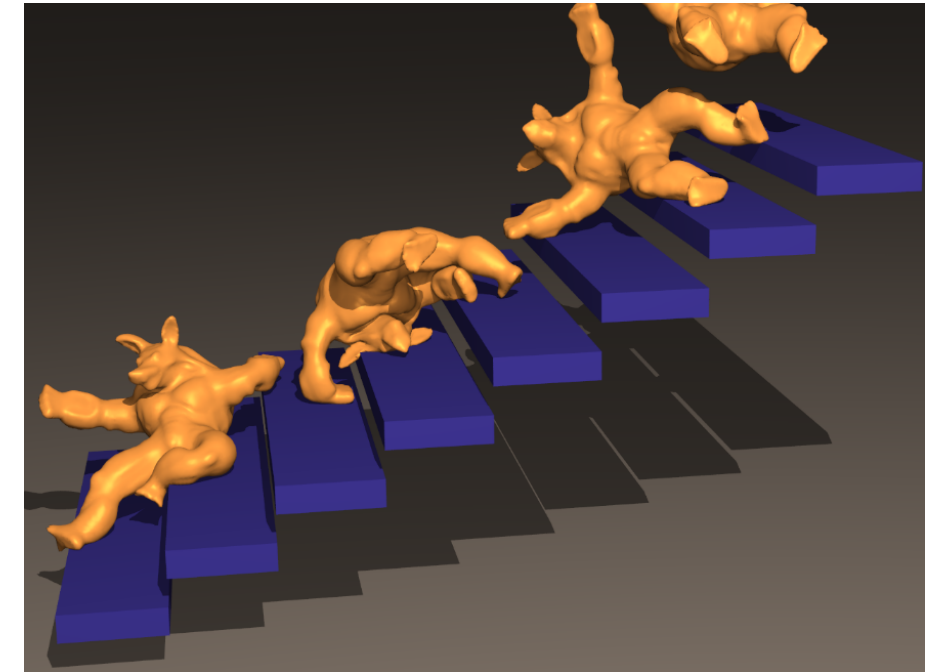
Fundamental models

- 1- Particles
- 2- Rigid bodies
- 3- Continuum material**

Deformation of a continuous shape

Every part of the shape can be deformed

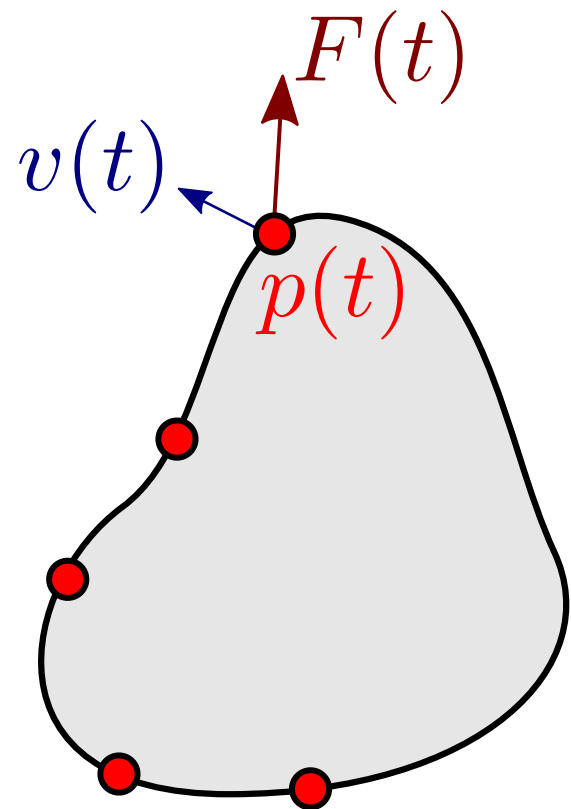
ex. Describing elastic shapes, visco-elastic shapes, fluids, etc.



Two ways to describe the deforming object

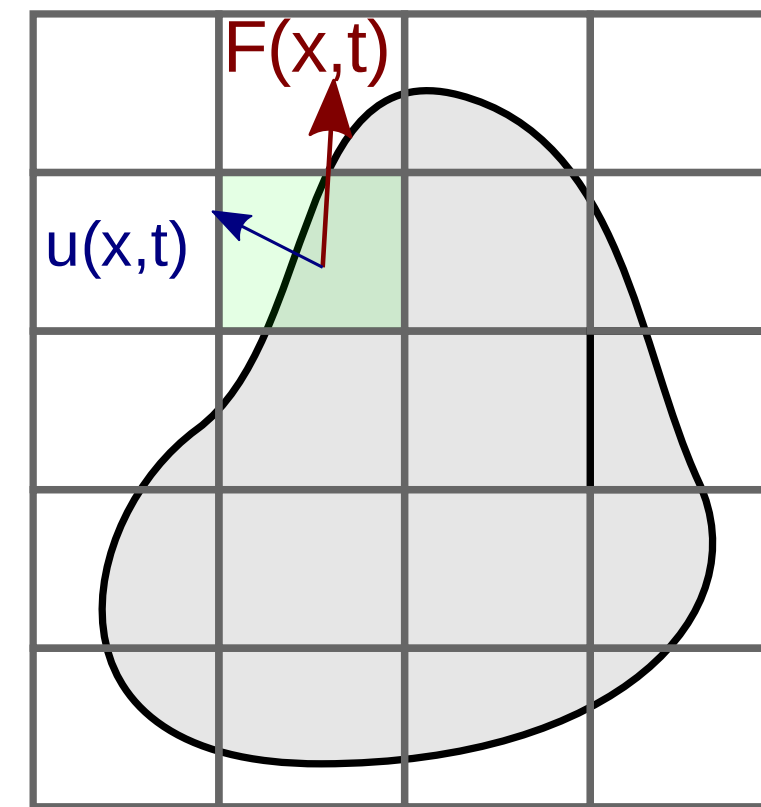
1. Lagrange representation

Positions follow the object deformation



2. Euler representation

Positions are fixed in 3D space



Deformation the Lagrangian description

Deformation map $\varphi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ such that $p = \varphi(P)$

P position in the reference undeformed shape

p position in the deformed configuration.

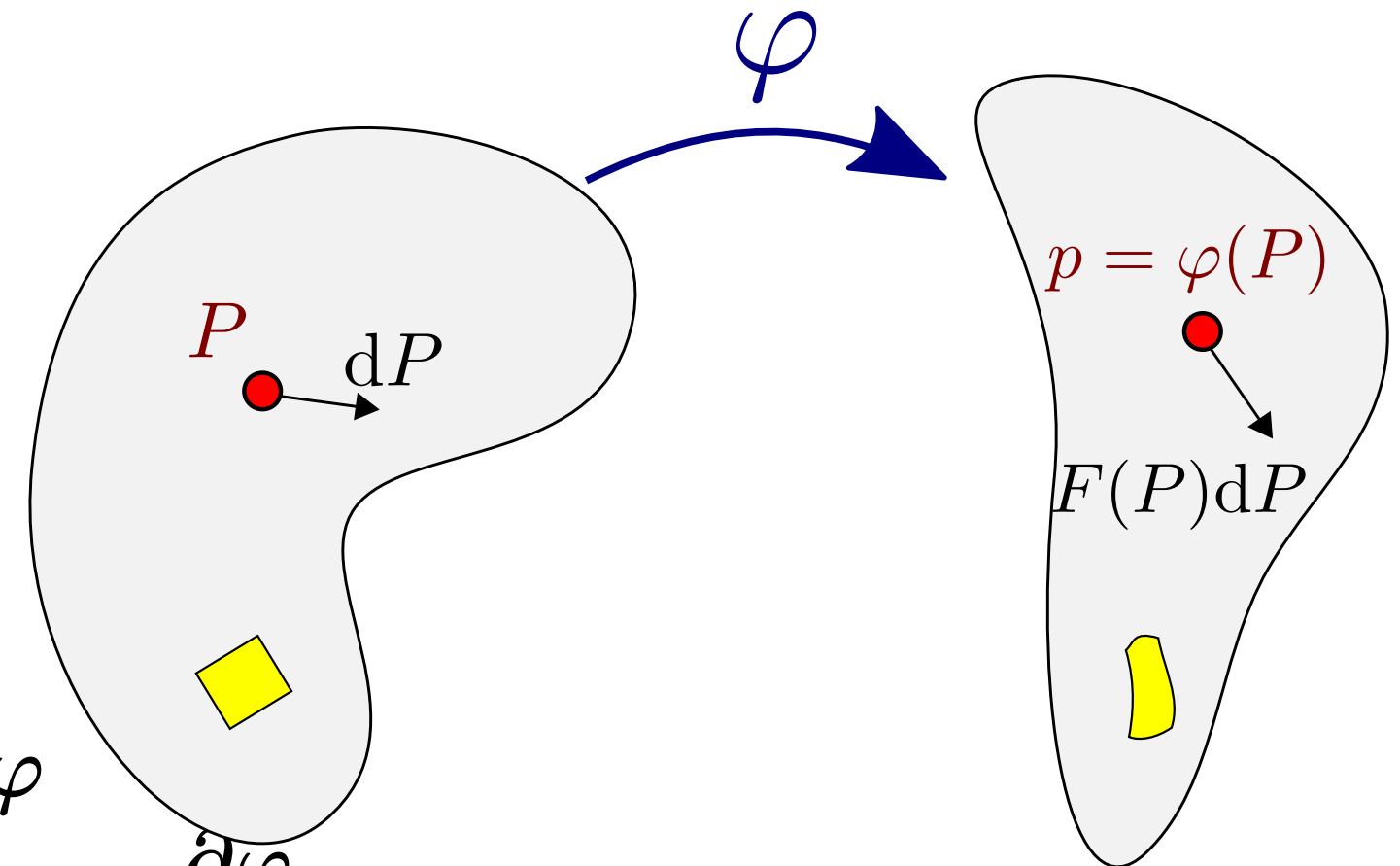
Deformation Gradient F

- $F(P) = \frac{\partial \varphi}{\partial P}(P) = \frac{\partial p}{\partial P} \in \mathbb{R}^{3 \times 3}$

- Characterizes the local deformation associated to φ

Position $P + dP$ is mapped into $\varphi(P + dP) \simeq p + \frac{\partial \varphi}{\partial P} dP$

- $F(P) = \begin{pmatrix} \frac{\partial \varphi_x}{\partial X} & \frac{\partial \varphi_x}{\partial Y} & \frac{\partial \varphi_x}{\partial Z} \\ \frac{\partial \varphi_y}{\partial X} & \frac{\partial \varphi_y}{\partial Y} & \frac{\partial \varphi_y}{\partial Z} \\ \frac{\partial \varphi_z}{\partial X} & \frac{\partial \varphi_z}{\partial Y} & \frac{\partial \varphi_z}{\partial Z} \end{pmatrix}$



Strain

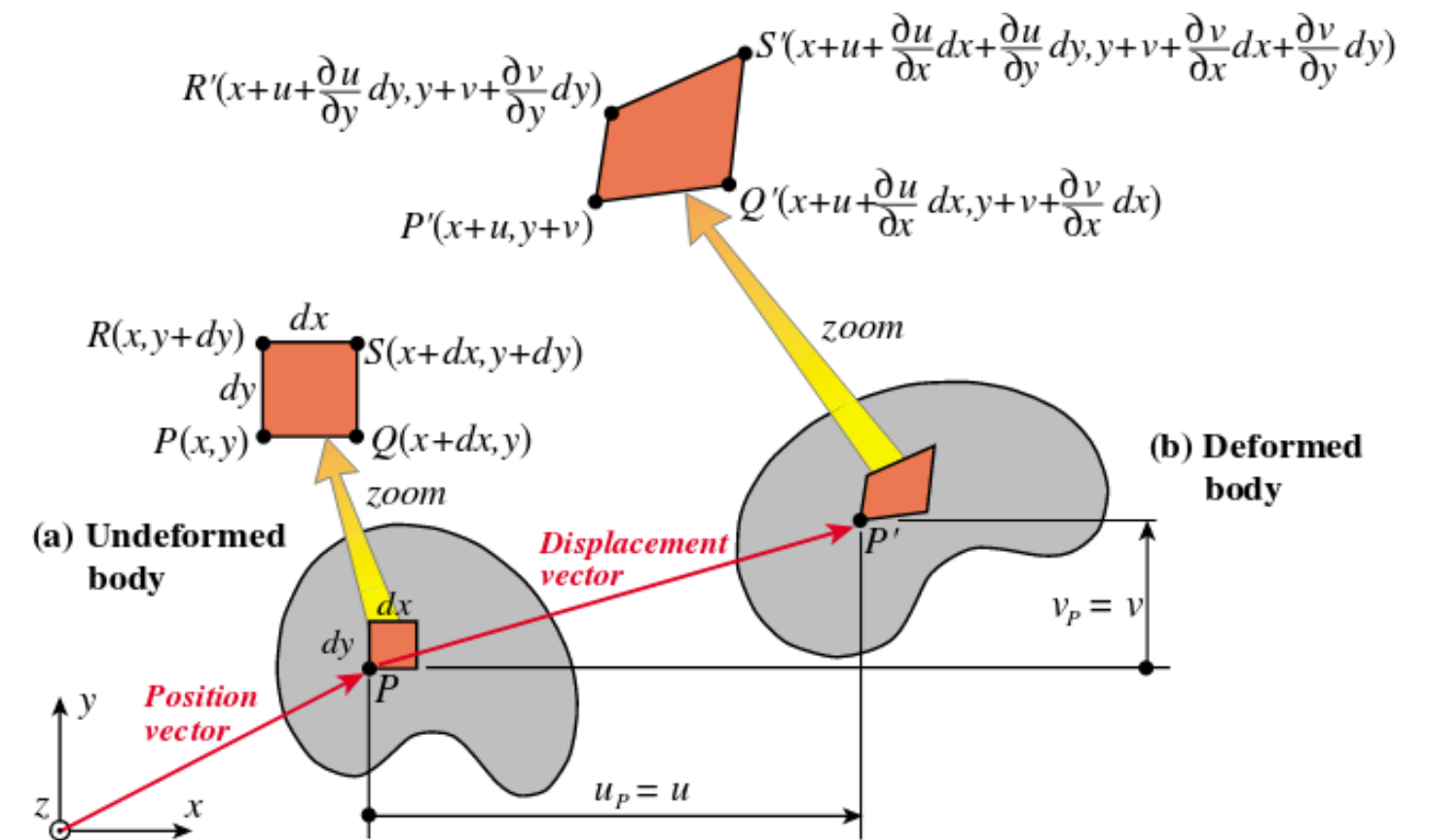
Deformation gradient F describe both

- Rigid transformation (rotation) - not related to material effort
- Any other deformation inducing local length change - related to material effort

Strain ϵ is a measure of deformation ignoring rigid transformation.

Several possible measure of strain

- Green strain tensor $\epsilon = \frac{1}{2} (F F^T - \text{Id})$
 - (+) If φ is a rotation $F = R \Rightarrow \epsilon = 0$
 - (-) Non linear in p
- Linearized Cauchy strain $\epsilon = \frac{1}{2} (F^T + F) - \text{Id}$
Used for small deformations



Stress

Stress $\sigma \in \mathbb{R}^{3 \times 3}$ describes internal forces (per area unit) induced by the local deformation (strain) in any direction

Constitutive Relation: Relation between stress and strain, characterize a type of material.

For linear constitutive relation:

$$\sigma_{ij} = \sum_{k,l} C_{ijkl} \epsilon_{kl} \quad , \quad C: \text{stiffness tensor (81 coefficients)}$$

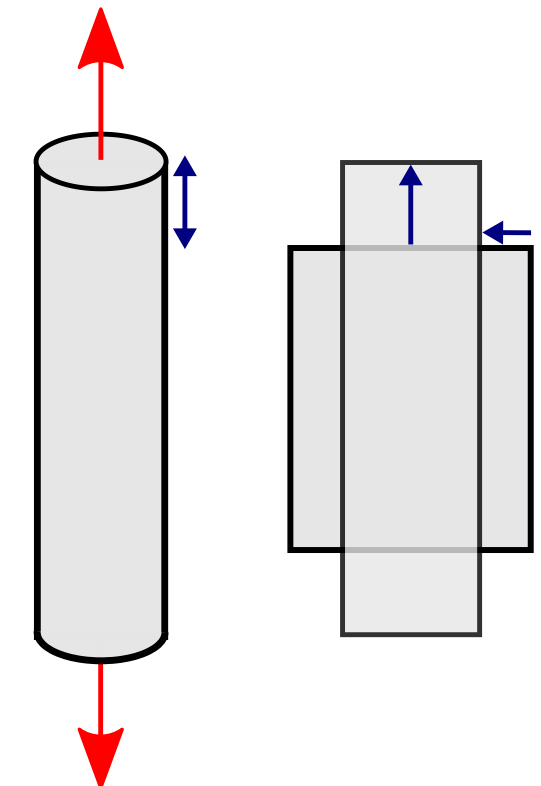
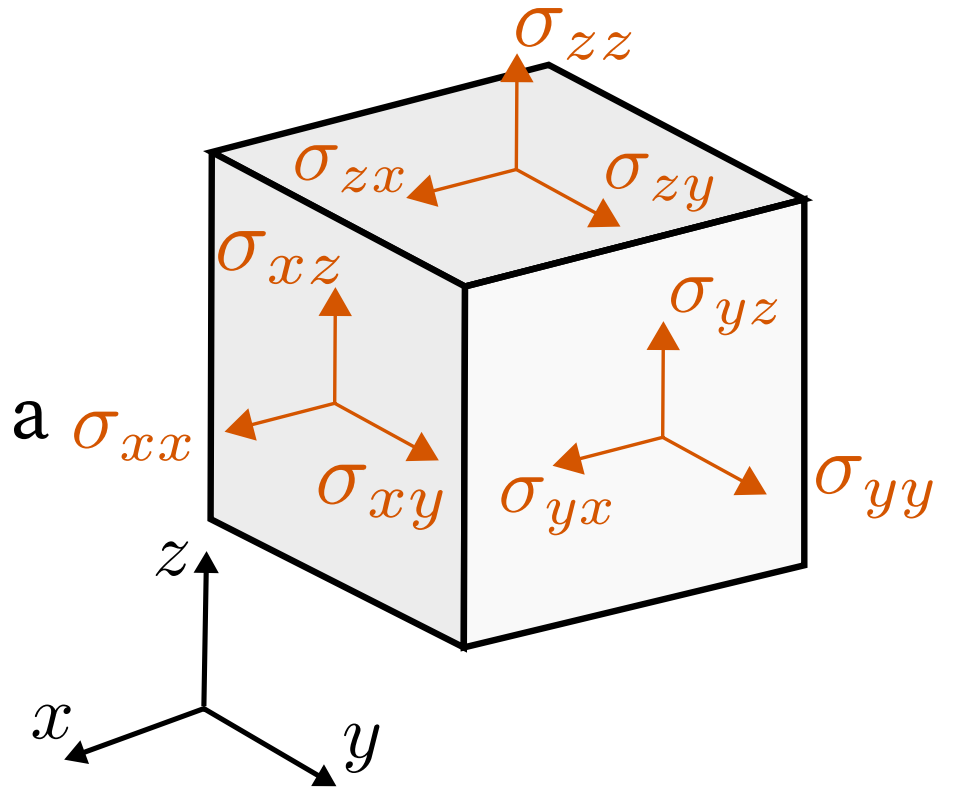
Strain energy/elastic potential energy:
$$U = \frac{1}{2} \sum_{i,j,k,l} \sigma_{ij}(\epsilon) \epsilon_{kl} = \frac{1}{2} \sum_{i,j,k,l} C_{ijkl} \epsilon_{ij} \epsilon_{kl}$$

For homogeneous isotropic elastic material, constitutive relation simplifies to

$$\sigma = 2\mu \epsilon + \lambda \text{tr}(\epsilon) \text{Id}, \quad (\mu, \lambda): \text{Lamé parameters}$$

Related to common mechanical modulus : Young' modulus Y and Poisson's ratio ν

$$\mu = \frac{Y}{2(1+\nu)}, \quad \lambda = \frac{Y\nu}{(1+\nu)(1-2\nu)}$$



Evolution equation

Fundamental principle of dynamics in the entire volume Ω

Change of momentum = External forces (in volume) + Traction (stress applied on exterior surface normals)

$$\Rightarrow \underbrace{\int_{\Omega} \rho p''(t) d\Omega}_{\text{Change of momentum}} = \underbrace{\int_{\Omega} F(t) d\Omega}_{\text{External forces}} + \underbrace{\int_{\partial\Omega} \sigma n dS}_{\text{Traction force on the boundary}}$$

Using divergence theorem $\int_{\partial\Omega} \sigma n dS = \int_{\Omega} \text{div}(\sigma) d\Omega$

Equation in volume satisfied at each position $p \in \Omega$

$$\boxed{\rho p''(t) = F(t) + \text{div}(\sigma(t))} \quad \sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \quad \text{div}(\sigma) = \begin{pmatrix} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} \\ \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z} \\ \frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} \end{pmatrix}$$

Euler formulation

In Euler formulation quantities are expressed at fixed position in 3D space.

Deformation described by velocity $u(p, t)$ at a given 3D fixed point $p = (x, y, z)$ at time t .

- Do not require anymore a reference shape
- Usefull for heavily deforming shapes (ex. fluids, gaz).

- Change of speed during dt

$$\frac{du}{dt}(p, t) = \frac{\partial u}{\partial t} + \sum_i \frac{\partial u}{\partial p_i} \underbrace{\frac{dp_i}{dt}}_{u_i} = \frac{\partial u}{\partial t} + (u \cdot \nabla)u$$

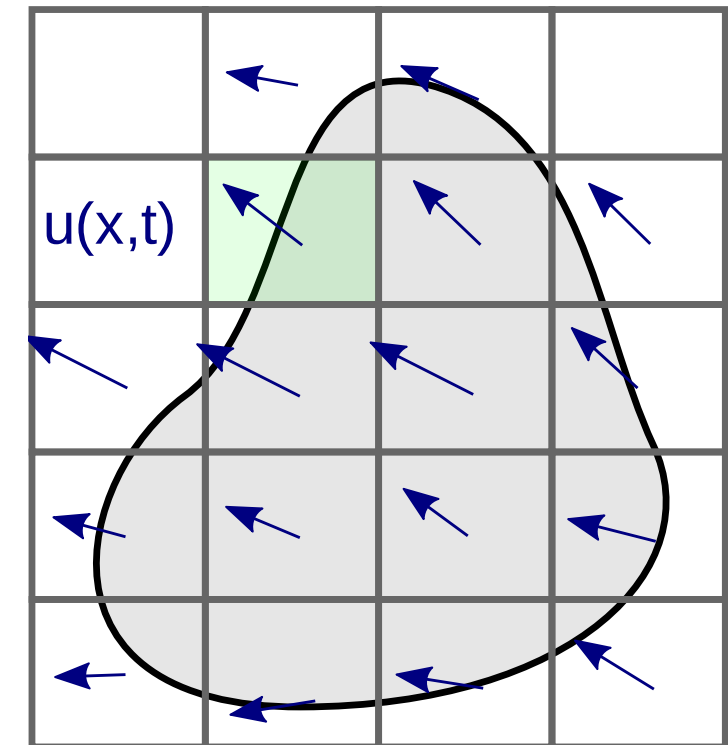
Called *material derivative*.

- Similarly to Lagrangian derivation:

- Strain-rate tensor ϵ (rate of change of deformation in a neighborhood of a point)

expressed with respect to u : $\epsilon = \frac{1}{2} (\nabla u + \nabla u^T)$

- Stress-rate tensor σ (rate of change of direction force per area in a neighborhood of a point).



Equation of motion for a fluid

- Fundamental principle of dynamics on linear momentum

$$\rho \frac{du}{dt} = F + \operatorname{div}(\sigma)$$

$$\Rightarrow \rho \frac{\partial u}{\partial t} = F + \operatorname{div}(\sigma) - \rho (u \cdot \nabla)u. \text{ The term } (u \cdot \nabla)u \text{ is called } \textit{advection}.$$

- External force: weight $F = \rho g$

- Stress decomposed into

$$\sigma = \sigma_{viscous} + \sigma_{pressure}$$

$$\sigma_{pressure} = -p \operatorname{Id} \text{ (pressure acts along normal of surface elements)}$$

$$\rho \frac{\partial u}{\partial t} = \rho g - \rho u \cdot \nabla u + \operatorname{div}(\sigma_{viscous} - p \operatorname{Id})$$

$$\Rightarrow \rho \frac{\partial u}{\partial t} = \rho g - \rho u \cdot \nabla u - \nabla p + \operatorname{div}(\sigma_{viscous})$$

Navier-Stokes equation

- Isotropic Newtonian fluid $\Rightarrow$ Linear (scalar) relation between strain-rate ϵ and stress-rate

$\sigma_{viscous}$

- $\sigma_{viscous} = 2\mu \epsilon = \mu (\nabla u + \nabla u^T)$, μ constant viscosity parameter
- Incompressible fluid $\Rightarrow \text{div}(u) = 0$

Equation of motion

$$\Rightarrow \rho \frac{\partial u}{\partial t} = \rho g - \rho u \cdot \nabla u - \nabla p + \text{div} (\mu (\nabla u + \nabla u^T))$$

- Noting that $\text{div}(\nabla u^T) = \nabla \text{div}(u) = 0$
- And $\text{div}(\nabla u) = \Delta u$
- Set $\nu = \mu/\rho$

$$\Rightarrow \frac{\partial u}{\partial t} = g - (u \cdot \nabla)u - \frac{1}{\rho} \nabla p + \nu \Delta u$$

Navier-Stokes equation for incompressible Newtonian fluid.

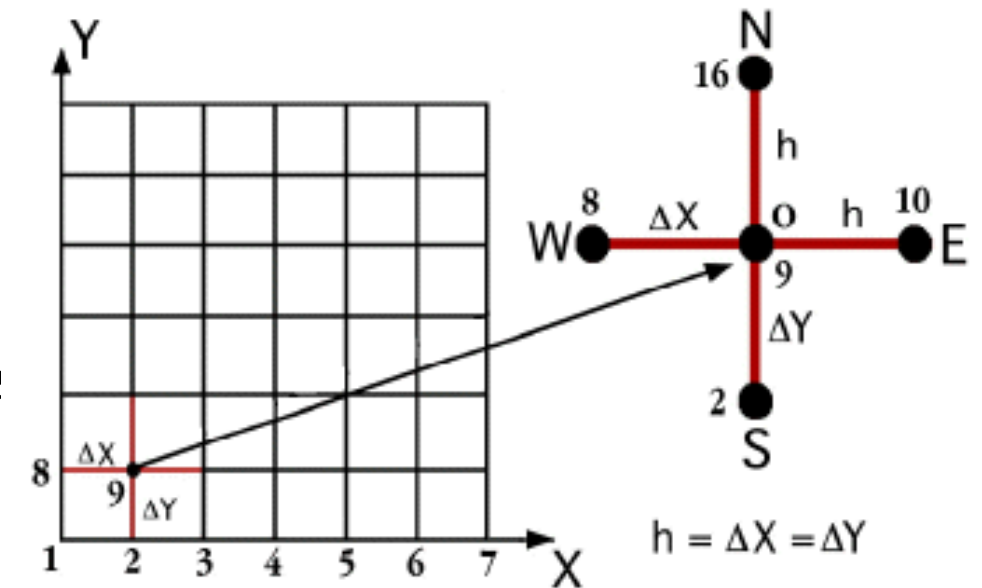
Numerical solutions

Lagrangian and Euler description leads to PDE (Partial Differential Equations)

In general: no explicit solutions $\Rightarrow$ approximate numerical solution

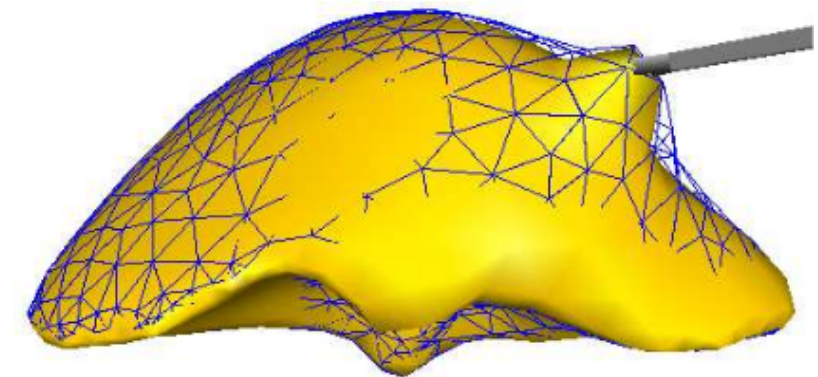
Finite Differences (FD)

- Discretize in space on a grid Δx and time Δt
- Use numerical approximation of derivatives using masks in space and time
 - (+) Very general, simple to setup
 - (+) Works well with rectangular grid (ex. Euler description)
 - (-) Difficult to handle shape boundaries
 - (-) Instabilities

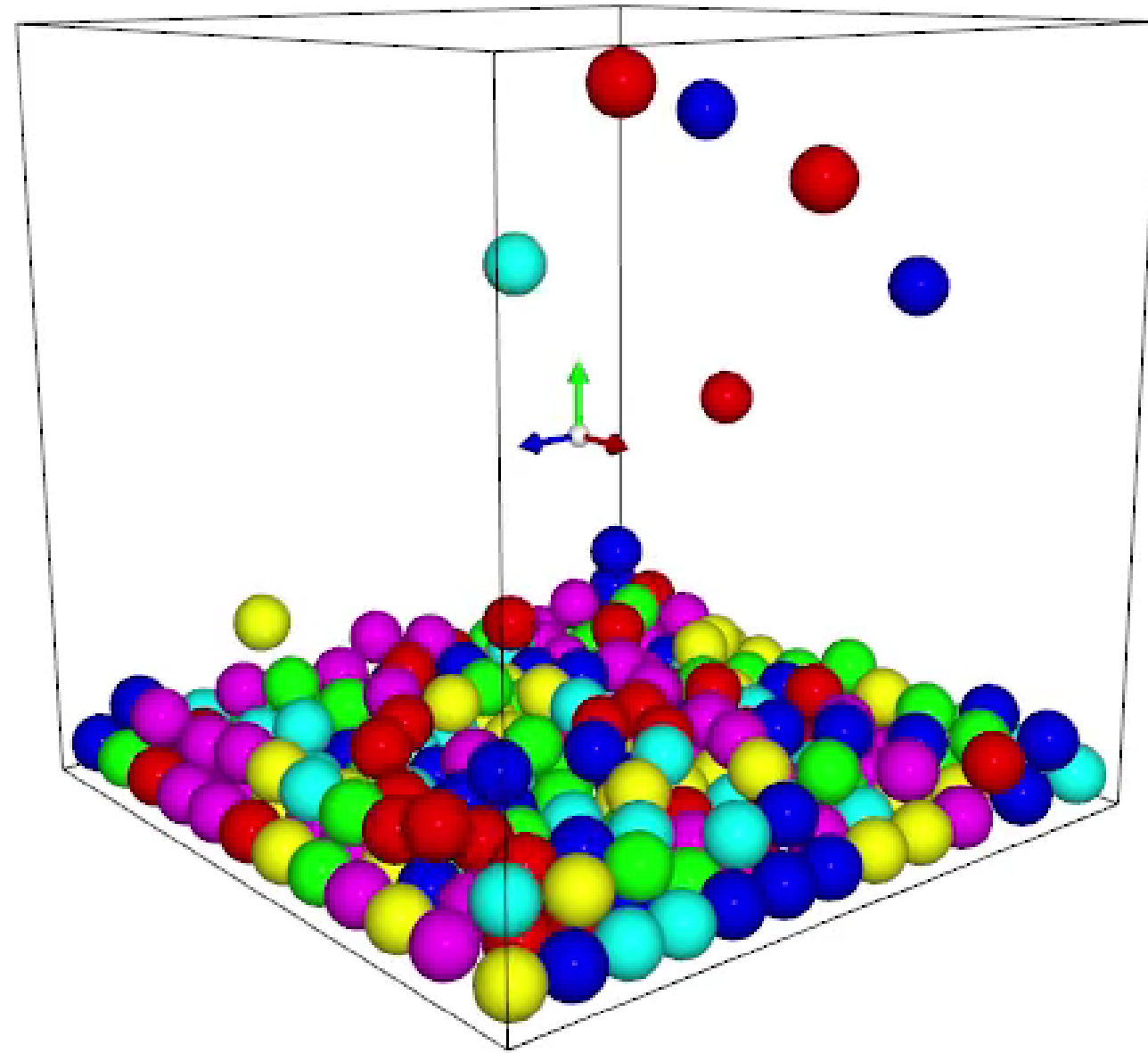


Finite elements method (FEM)

- Discretize the shape into simple elements. Build continuous function on each element.
 - In CG: Elements are tetrahedron (in volume). Continuous function are barycentric coordinates (linear interpolation functions).
- Integrate PDE over each element (weak formulation), leads to a linear system
 - (+) Handle boundaries (ex. Deforming solid)
 - (+) Guarantee on accuracy
 - (-) Complex to set up, and computationally heavy
 - (-) Requires good quality meshing



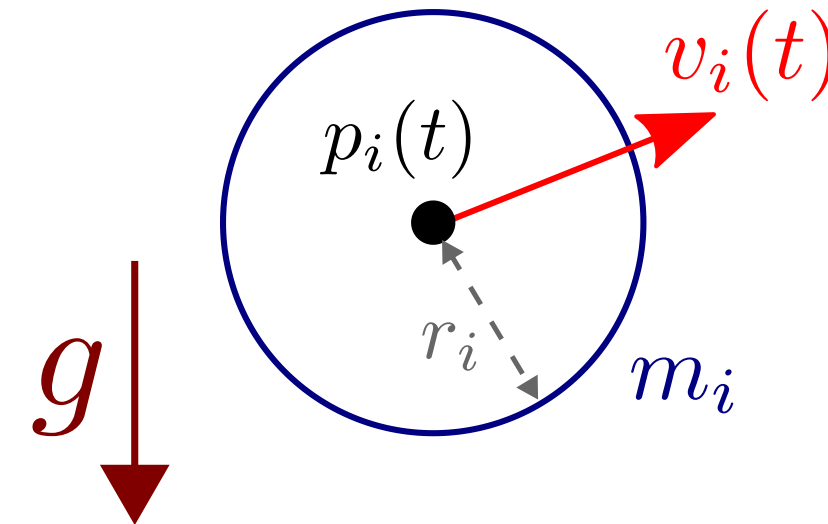
Rigid spheres



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 .

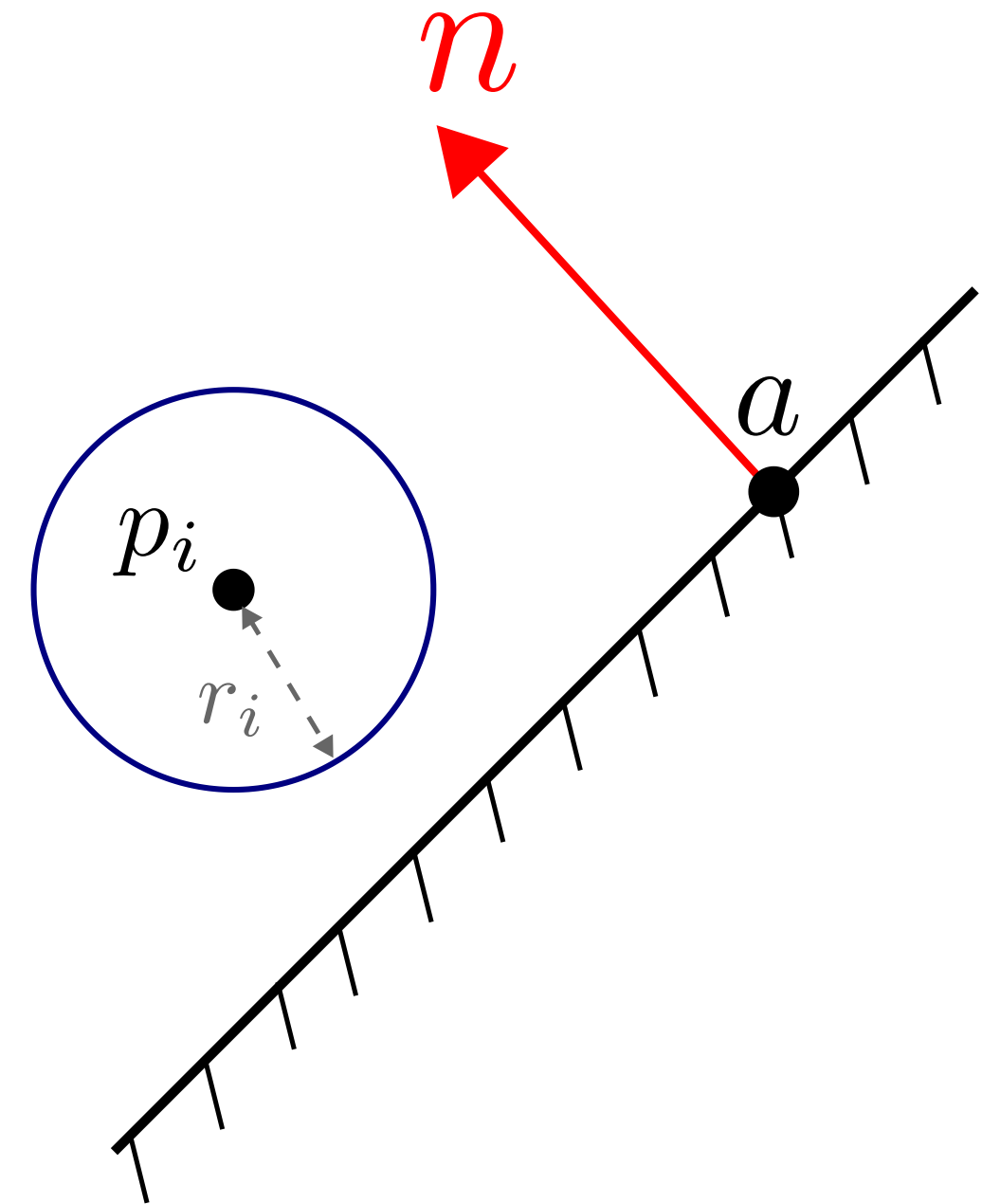
$$\{p \in \mathbb{R}^3 \in \mathcal{P} \Rightarrow (p - a) \cdot n = 0\}$$

- 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 $v = v_{//} + v_{\perp}$

$$-v_{\perp} = (v \cdot n) n$$

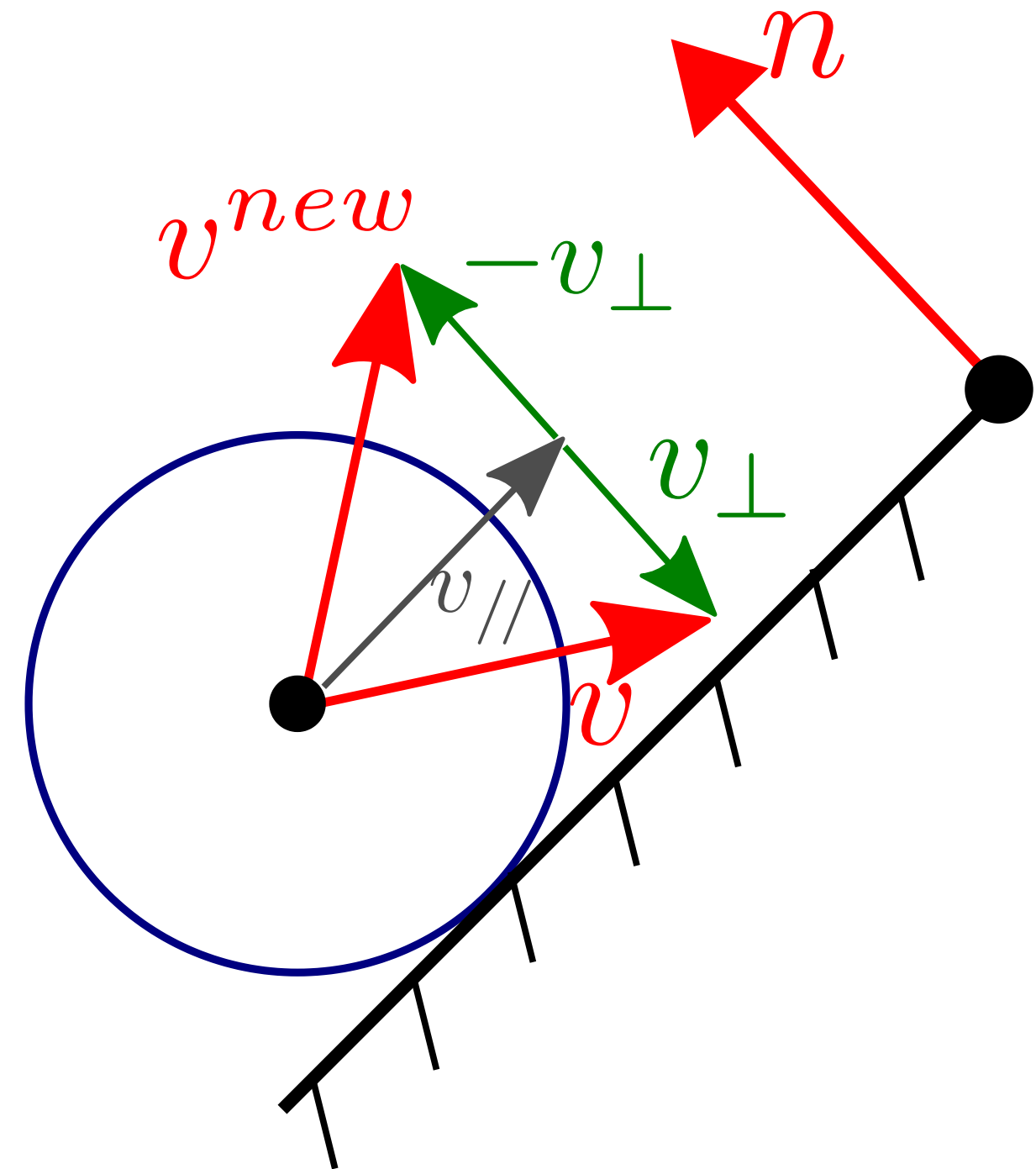
$$-v_{//} = v - (v \cdot n)n$$

New velocity

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

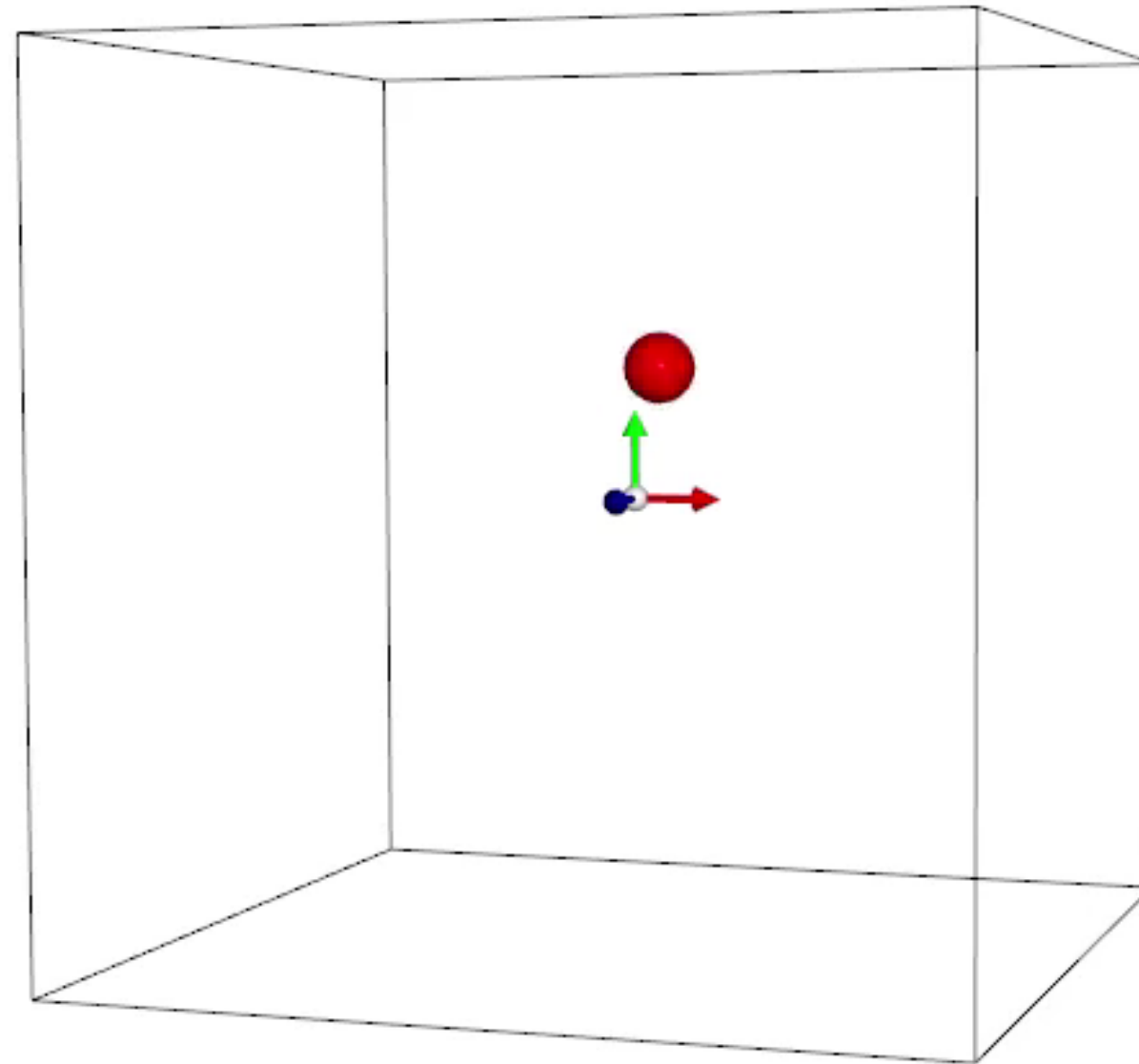
$\alpha \in [0, 1]$ Restitution coefficient in $//$ direction (friction)

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



Result: Collision response

Applying collision response on speed only

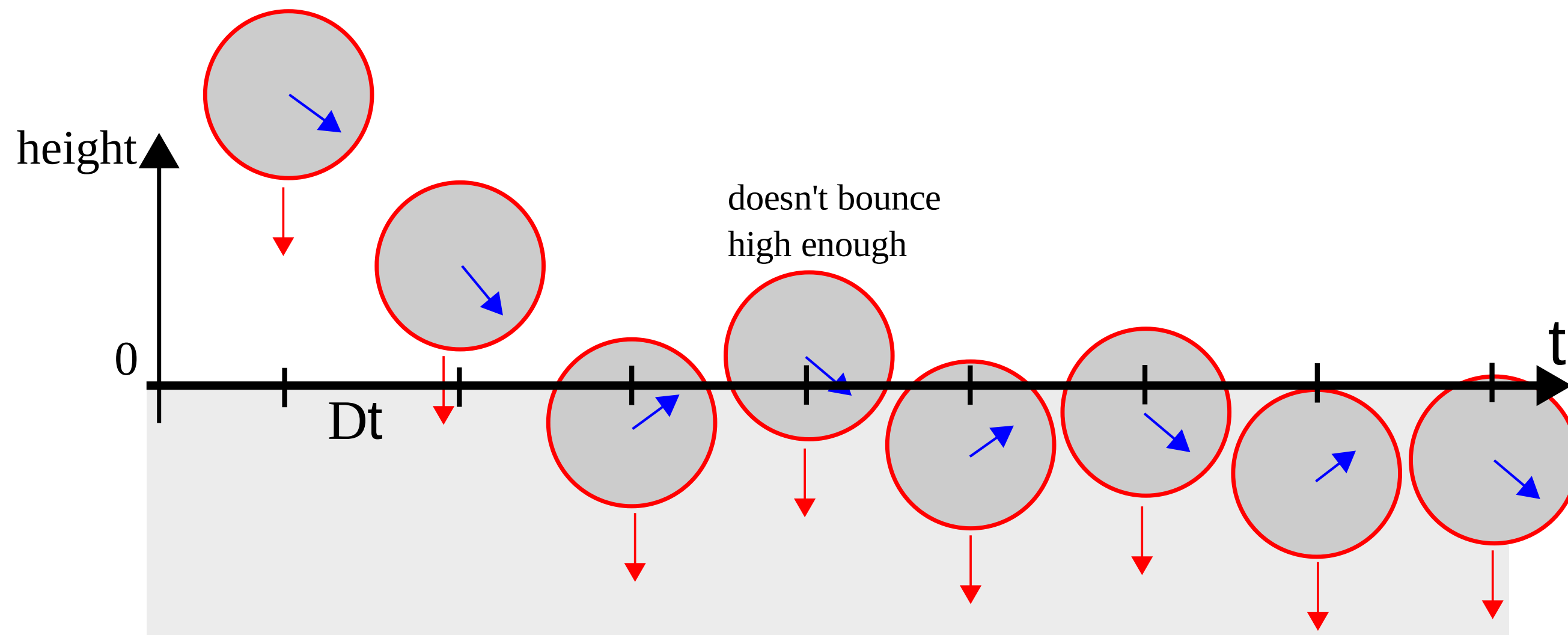


Result: Collision response - issue with discrete time

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) Correct position in projecting on the
constraint

(+) *Simple to implement*

(-) *Physically incorrect position*

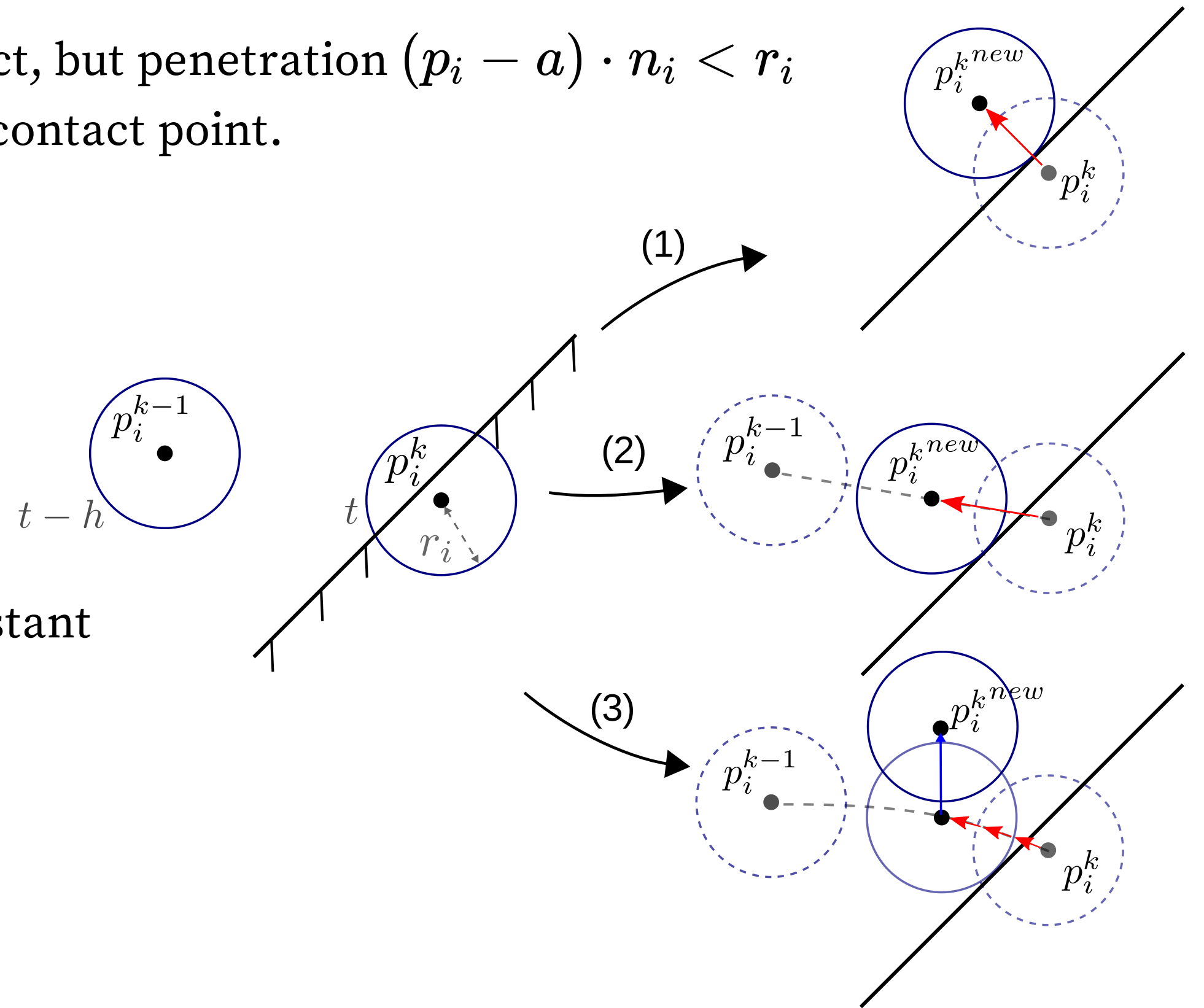
(2) Approximate the correct position

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

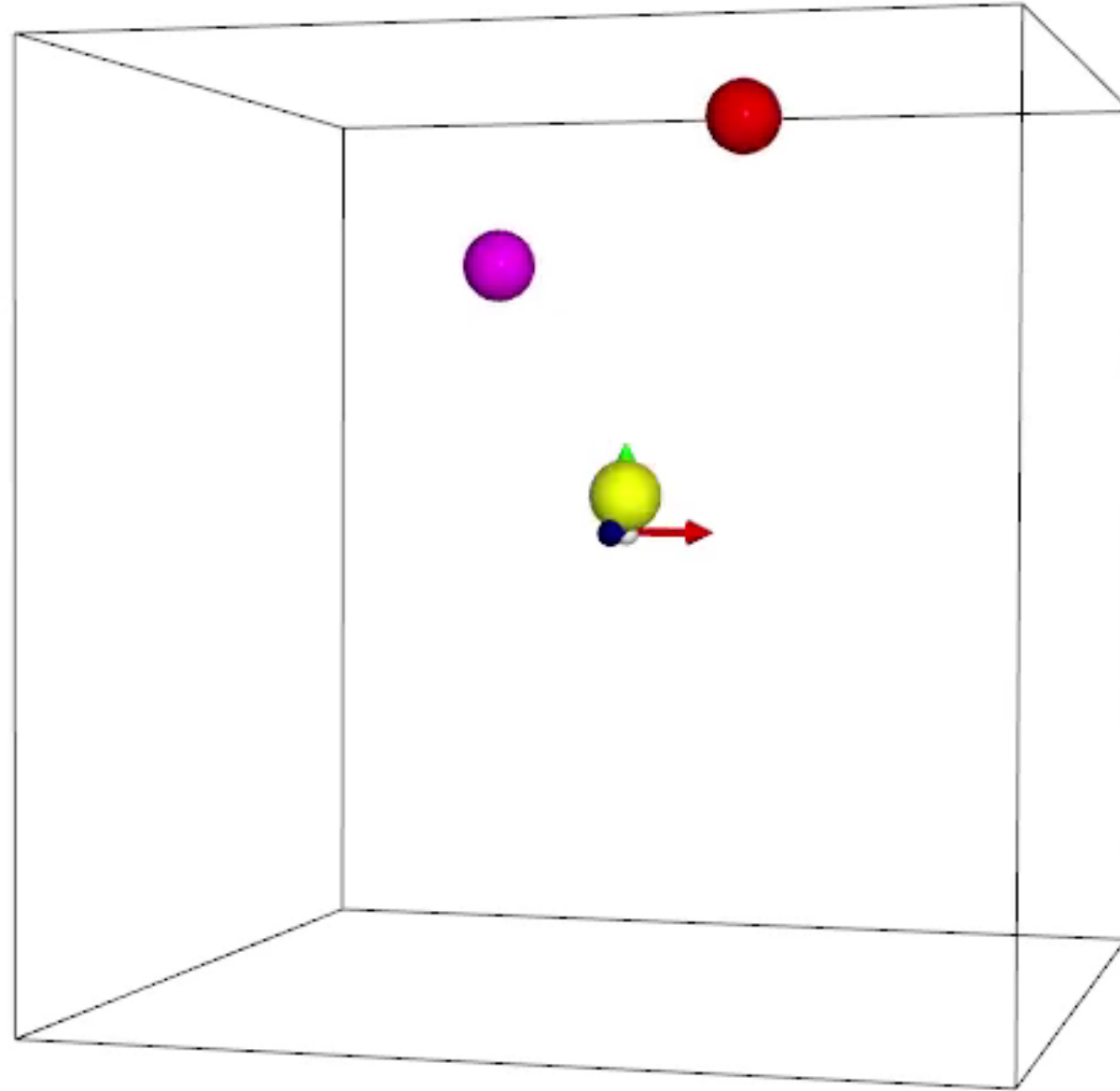
Continuous Collision Detection

(+) *Physically correct*

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



Result: Projecting position on plane



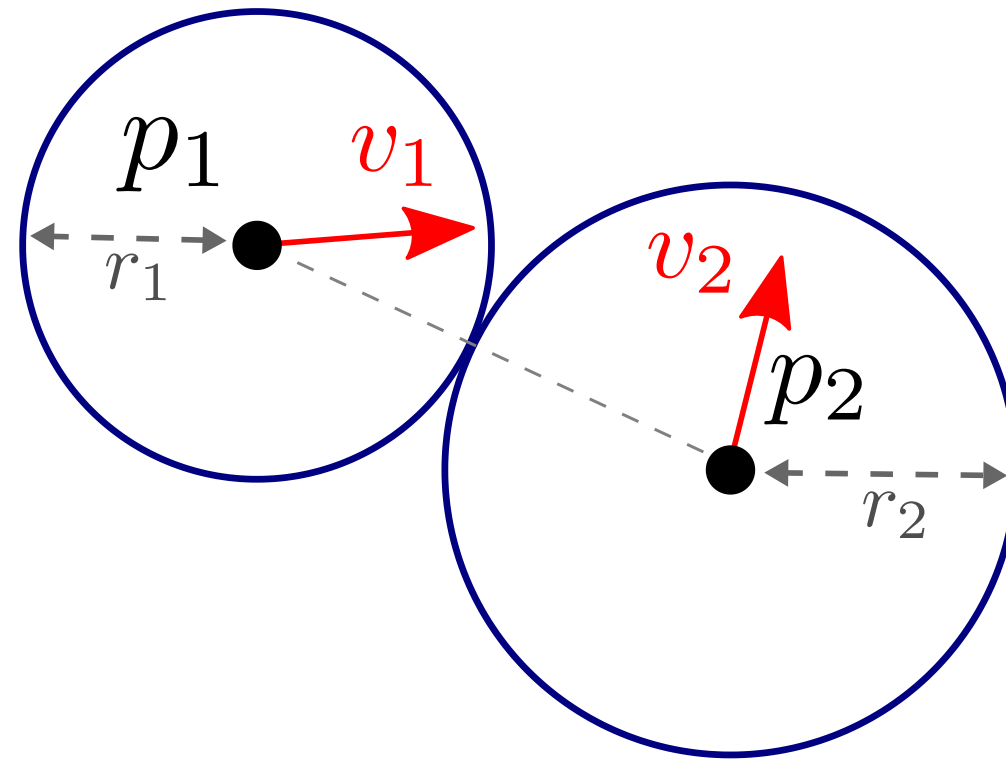
$$p_i^{new} = p_i + d n$$

$$d = r_i - (p_i - a) \cdot n_i : \text{distance of penetration}$$

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 $J = \int_{t_1}^{t_2} F(t) \, dt$

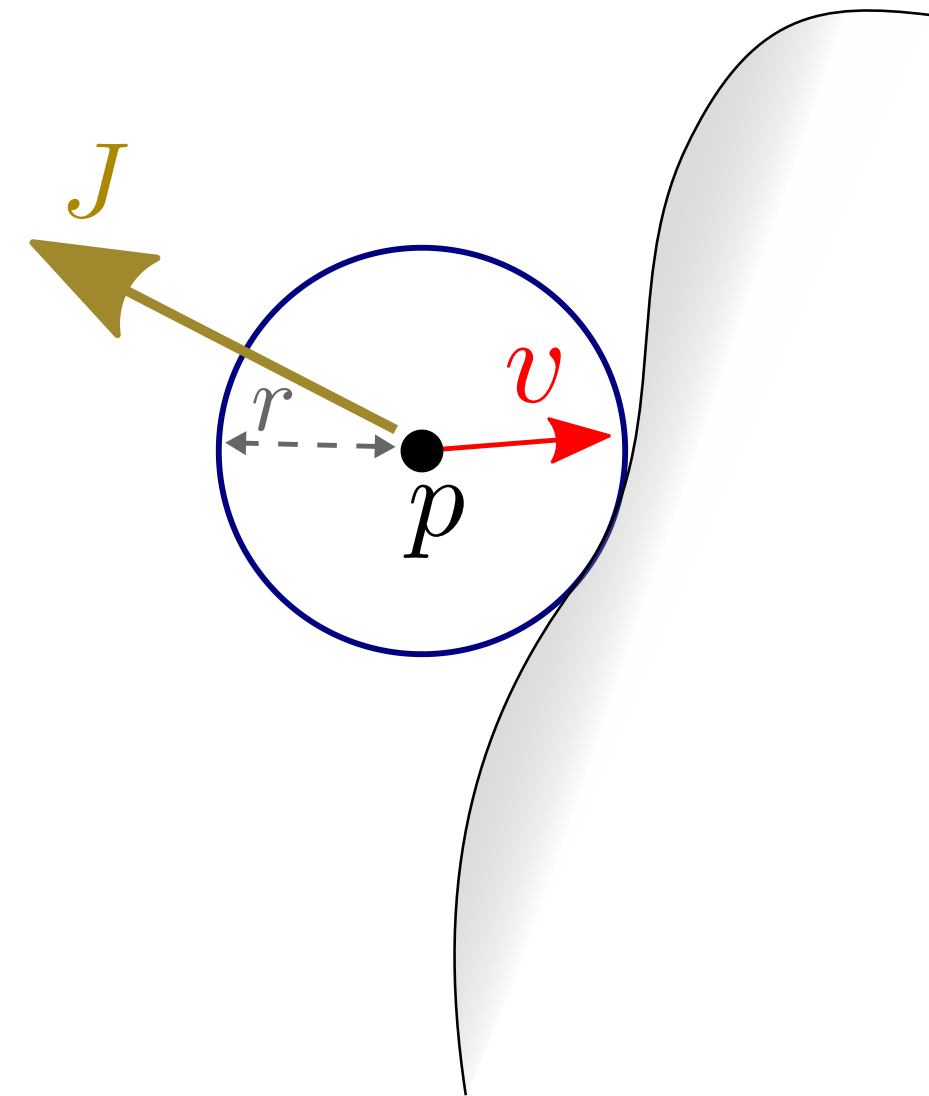
→ results in a sudden change of speed (/momentum) in a discrete case

For a particle with constant mass

$$\int_{t_1}^{t_2} F(t) \, dt = \int_{t_1}^{t_2} m a(t) \, dt$$
$$\Rightarrow J = m (v(t_2) - v(t_1))$$

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

⇒ 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

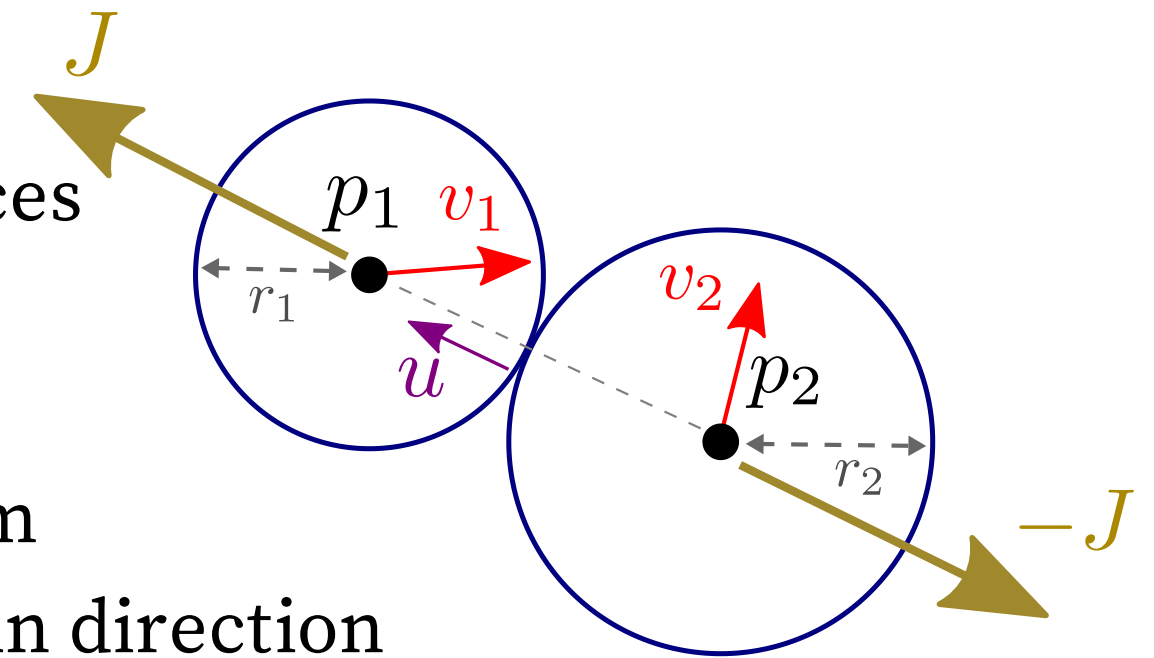
$$\Rightarrow j = 2 \frac{m_1 m_2}{m_1 + m_2} (v_2 - v_1) \cdot u$$

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

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

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

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



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

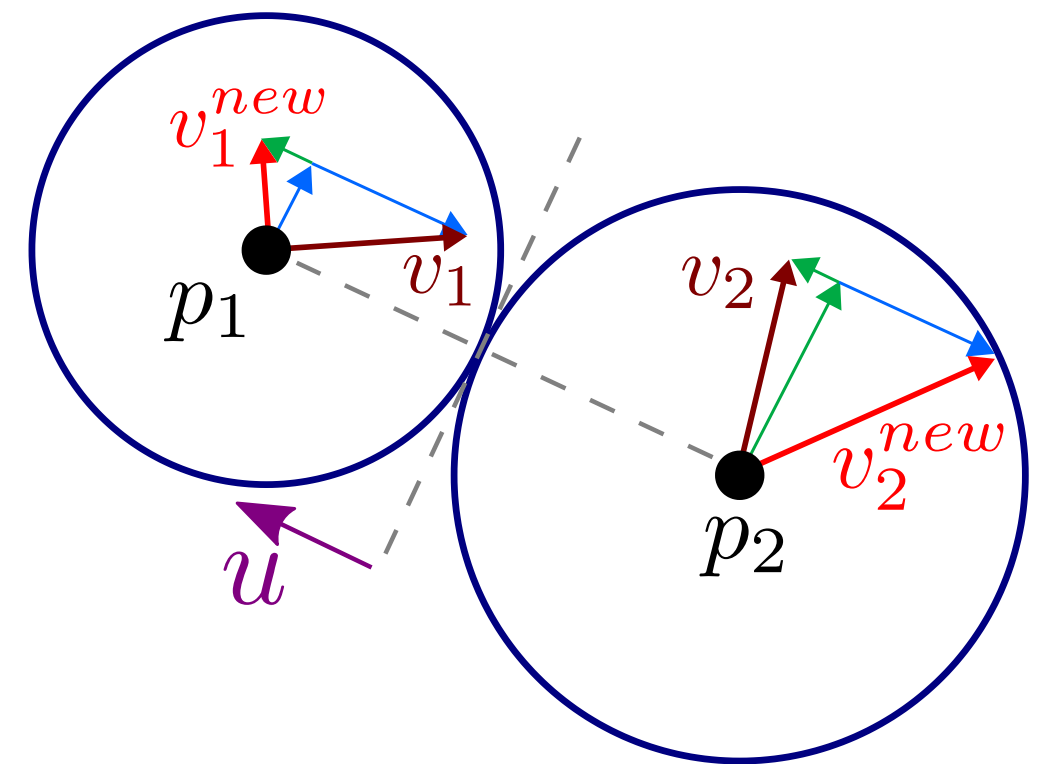
$$v_1^{new} = v_1 + ((v_2 - v_1) \cdot u) u = v_{1//} + v_{2\perp}$$

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

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

$$v_1^{new} = \alpha v_{1//} + \beta v_{2\perp}$$

$$v_2^{new} = \alpha v_{2//} + \beta v_{1\perp}$$



Summary

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

2-a. If collision (relative speed $> \epsilon$)

Elastic collision (/bouncing) $v_{1/2} = \alpha v_{1/2} \pm \beta J / m_{1/2}$

2-b. If *static* contact (relative speed $\leq \epsilon$)

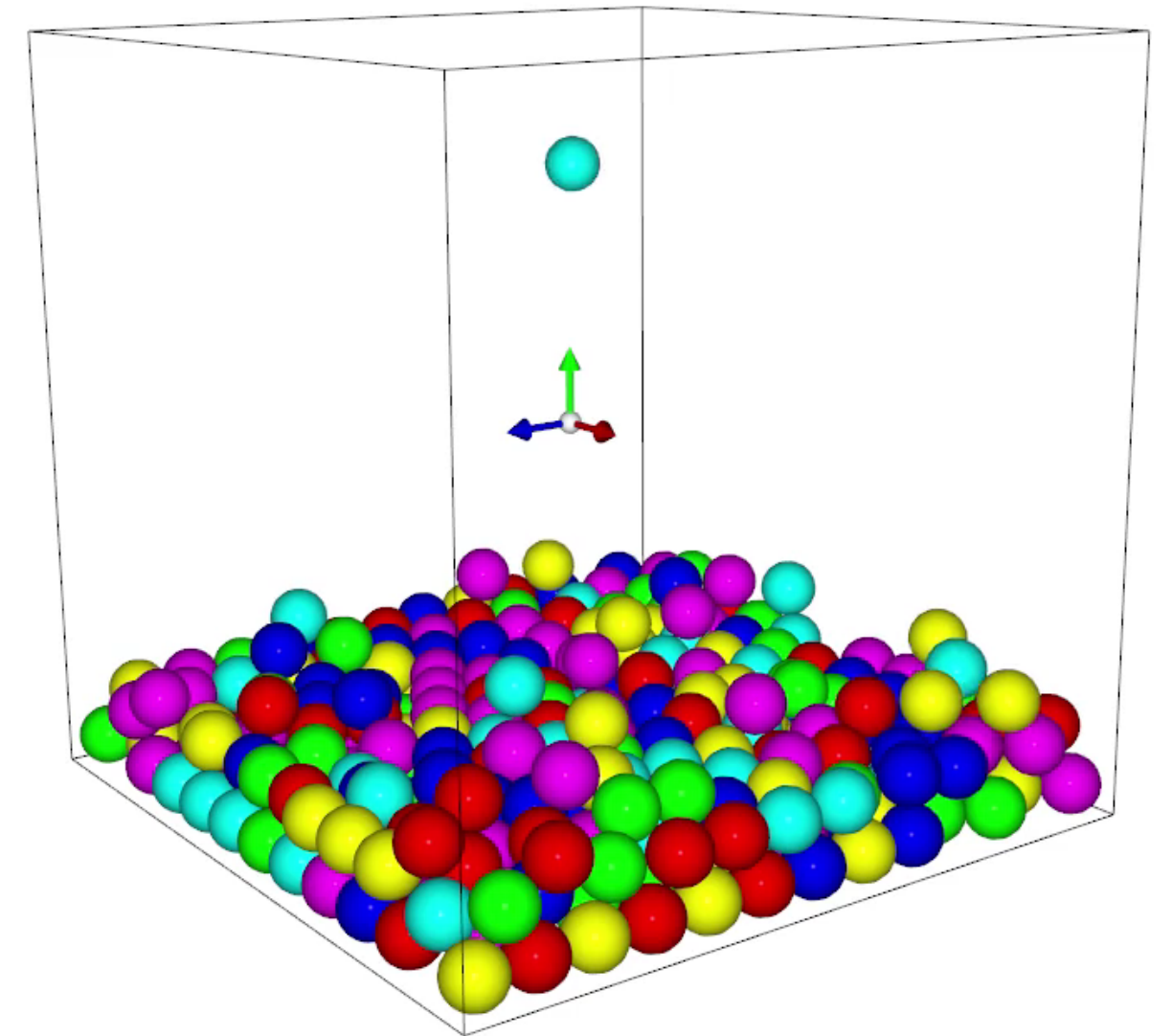
Friction $v_{1/2} = \mu v_{1/2}$, $\mu \in [0, 1]$

Avoids jittering

3. Correct position (project on contact surface)

$$p = p \pm d/2 u$$

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



Multiple collisions

Pairwise collisions $\Rightarrow$ no global collision free state

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

Reducing time-step help, Iterating over multiple pass help

But correct solution in all cases is complex $\rightarrow$ global approach

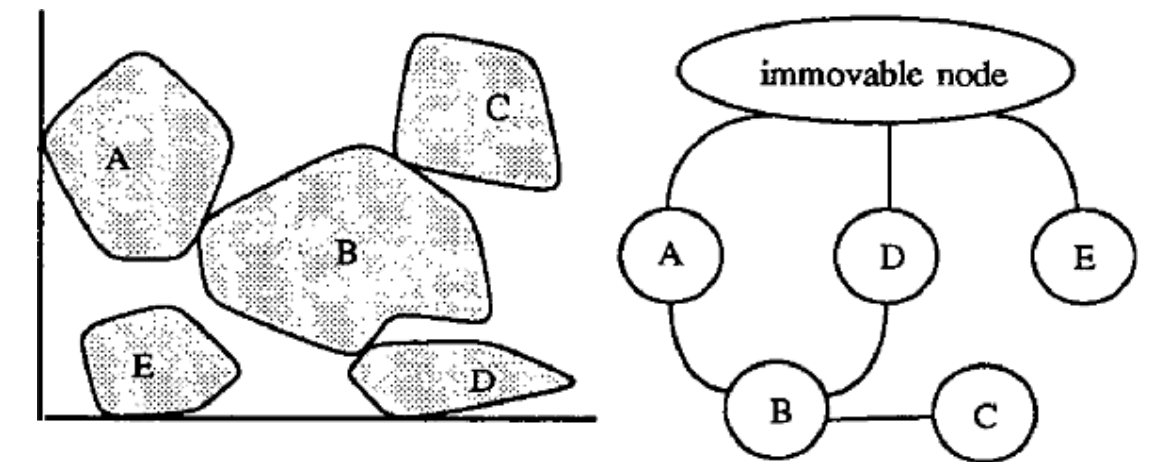
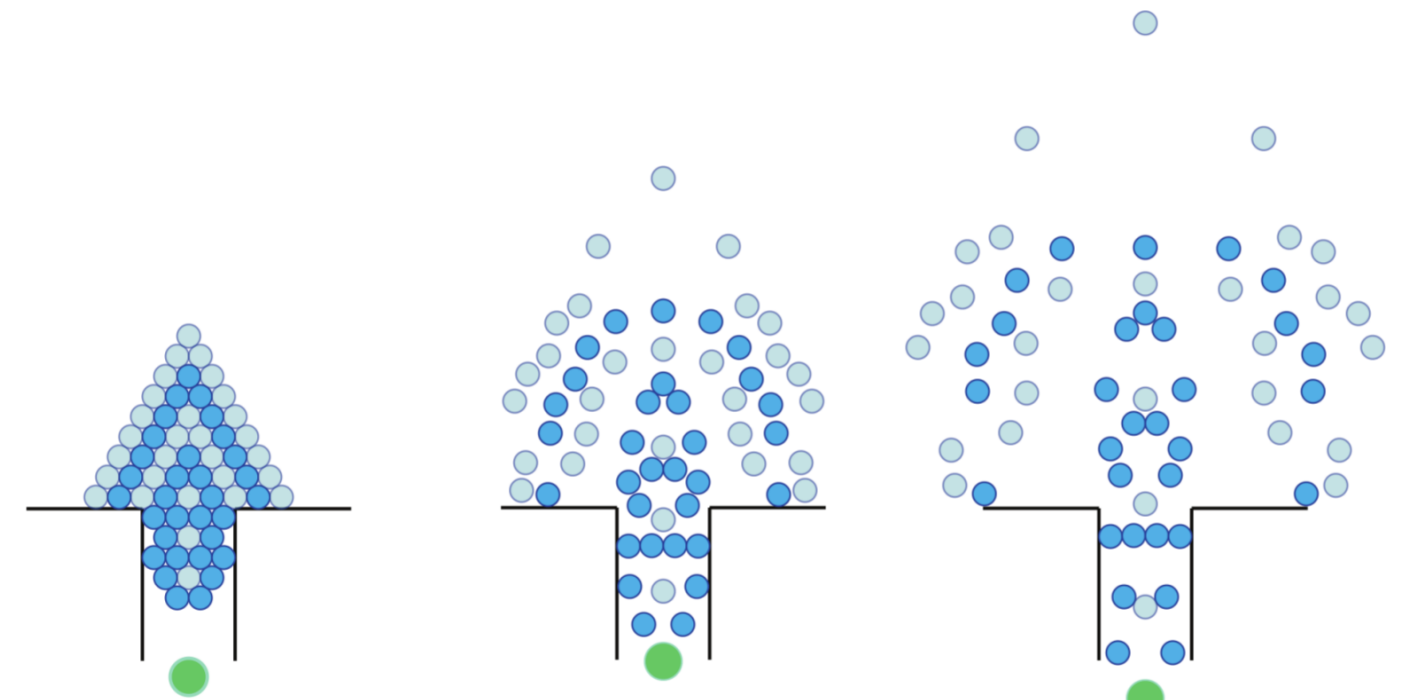
- Precompute contact graph
explicit shock propagation management
- Global constraint-based method

Impulse: $n_i \cdot (v_i - v_j) \geq 0$

Momentum preservation: $m_i v_i - m_j v_j = 0$

Energy preservation/dissipation

$\Rightarrow$ 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]

Elastic models

Spring structure

Numerical solution of ODE

Cloth simulation

Material model

Elasticity: Shape goes back toward its original rest position when external forces are removed.

- Purely elastic models don't lose energy when deformed (potential $\leftrightarrow$ kinetic)



Plasticity: Opposite of elasticity. Plastic material doesn't come back to their original shape (/change their rest position during deformation).

- Ductile material - can allow large amount of plastic deformation without breaking (plastic)
- Brittle - Opposite (glass, ceramics)

Viscosity: Resistance to flow (usually for fluid, ex. honey)

In reality

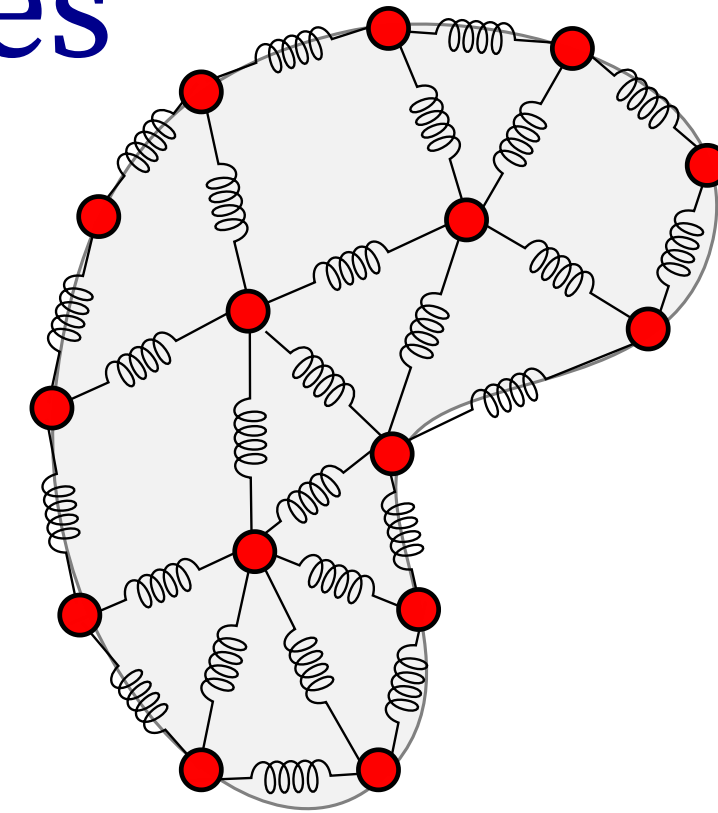
- *Elasto-plastic materials:* Allow elastic behavior for small deformation, and plastic at larger one.
- *Visco-elastic materials:* Elastic properties with delay.



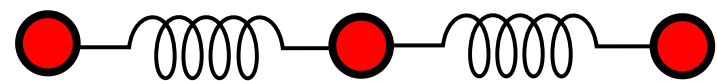
Modeling elastic shapes with particles

Spring mass systems

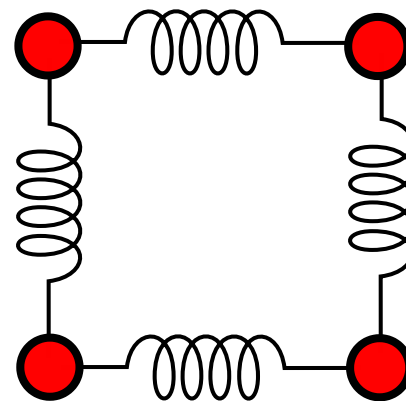
- Particles (position, velocity, mass): samples on shape
- Springs : link closed-by particles in the reference shape



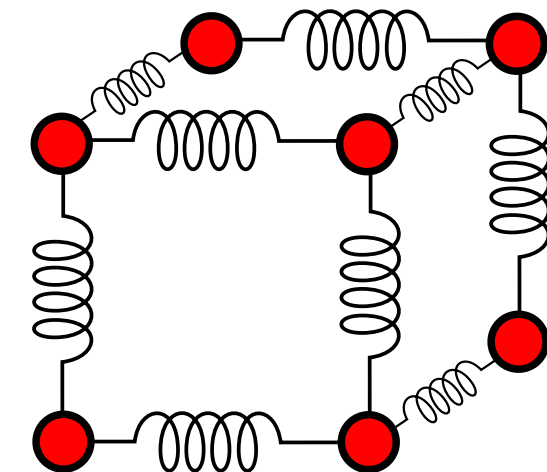
1D curve structure



2D surface structure



3D volume structure

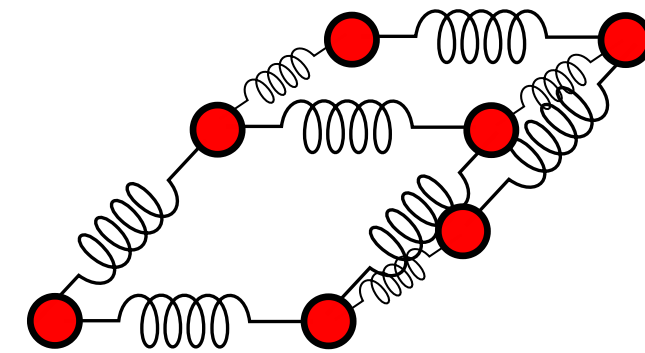
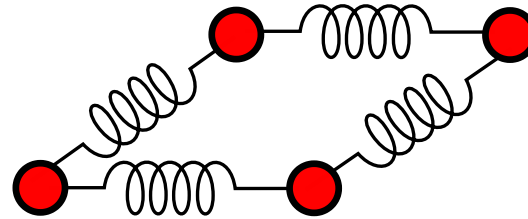
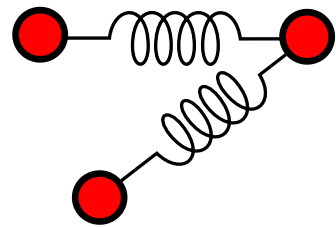


Spring structure

How to model spring connectivity ?

- **Structural springs:** 1-ring neighbors springs ($\simeq$ mesh edges)

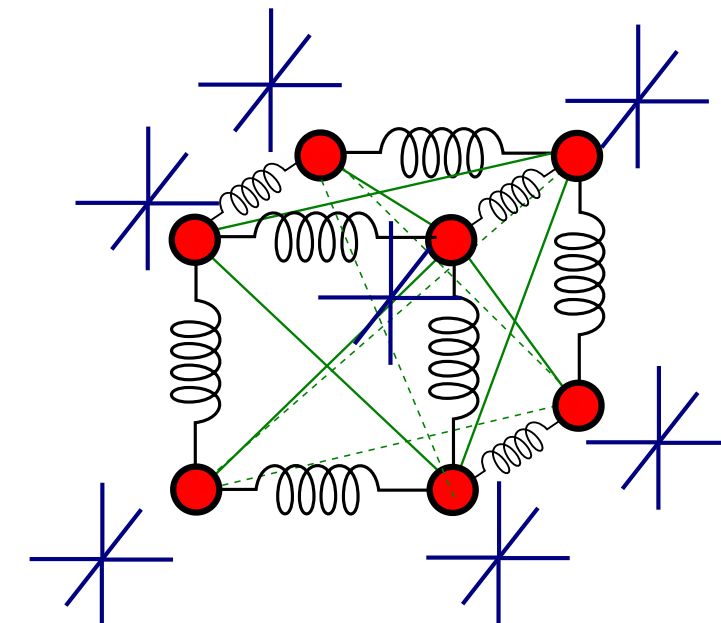
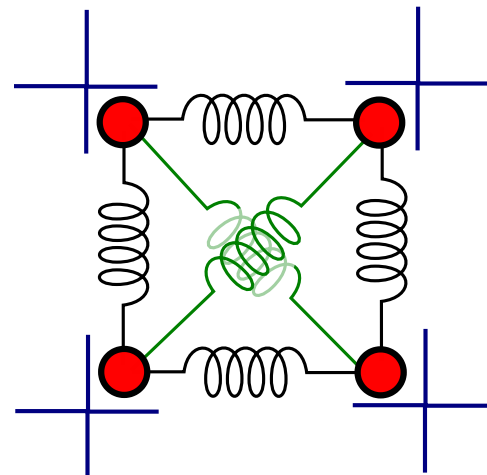
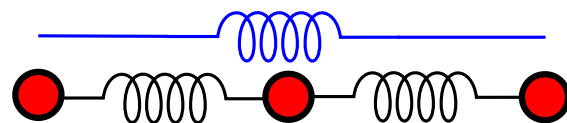
(+) Limit elongation/contraction, (-) Allows shearing, and bending



$\Rightarrow$ Add extra springs connectivity

- **Shearing springs:** Diagonal links

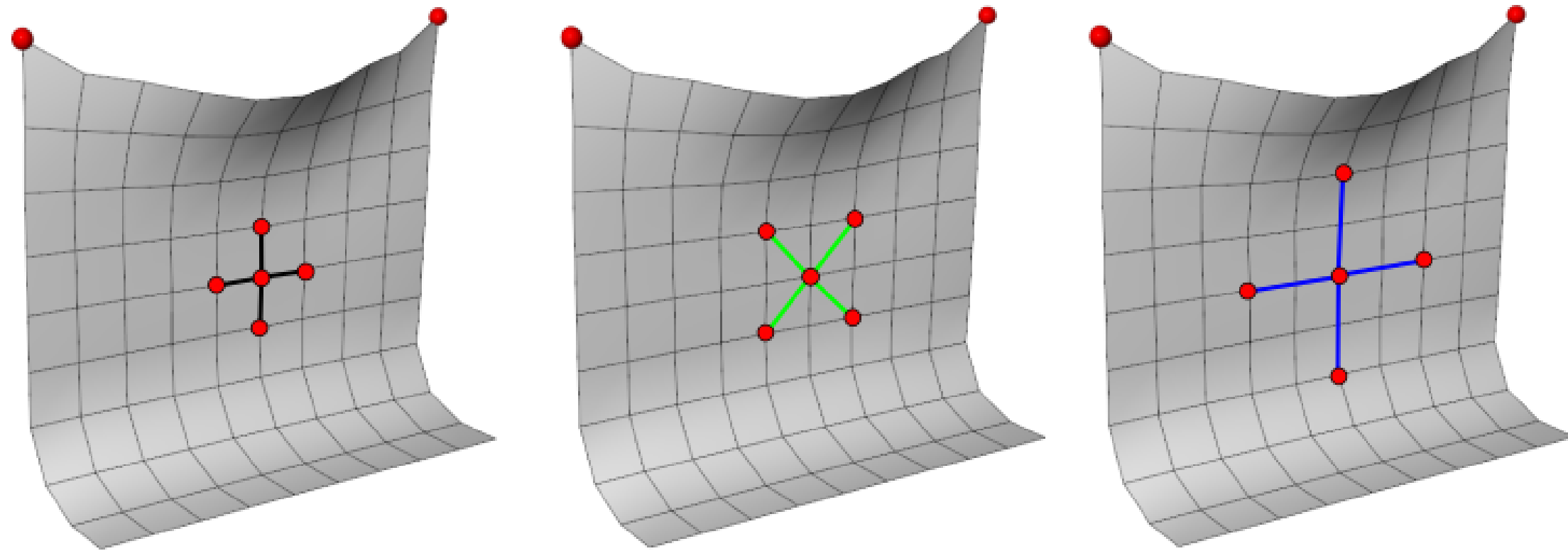
- **Bending springs:** 2-ring neighborhood



Cloth Simulation

Mass-spring cloth simulation

- Particles are sampled on a $N \times N$ grid.
 - Each particle has a mass m ($m_{cloth} = N^2 m$)
- Set structural, shearing and bending springs.



Forces

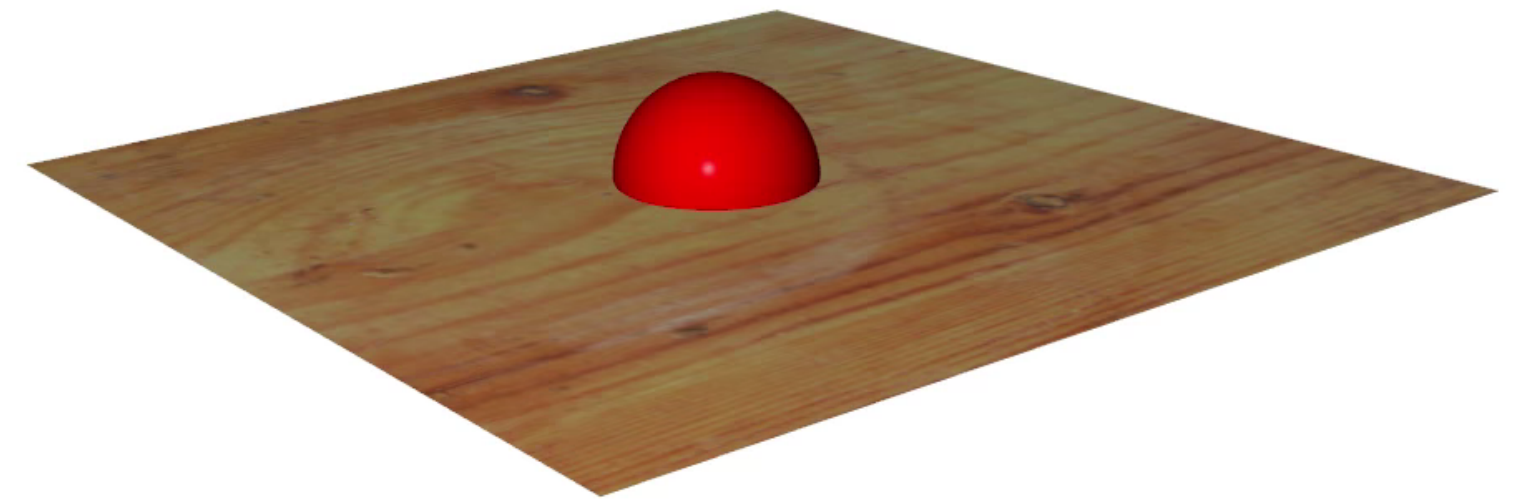
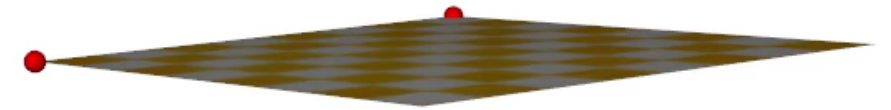
- On each particle: gravity + drag + spring forces

$$- F_i(p, v, t) = m_i g - \mu v_i(t) + \sum_{j \in \mathcal{V}_i} K_{ij} \left(\|p_j(t) - p_i(t)\| - L_{ij}^0 \right) \frac{p_j(t) - p_i(t)}{\|p_j(t) - p_i(t)\|}$$

- $\mathcal{V}_i$: neighborhood of particle i

- L_{ij}^0 : rest length of spring ij

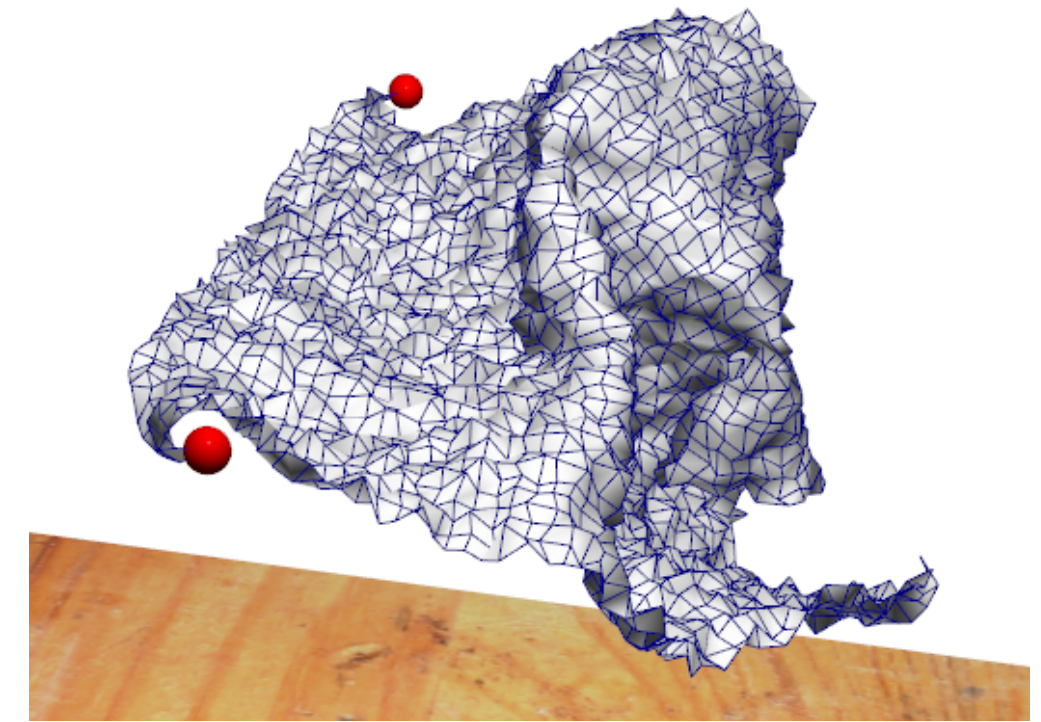
Associated ODE $\forall i, \begin{cases} p_i'(t) = v_i(t) \\ v_i'(t) = F_i(p, v, t)/m_i \end{cases}$



Q. How can we model the effect of the wind ?

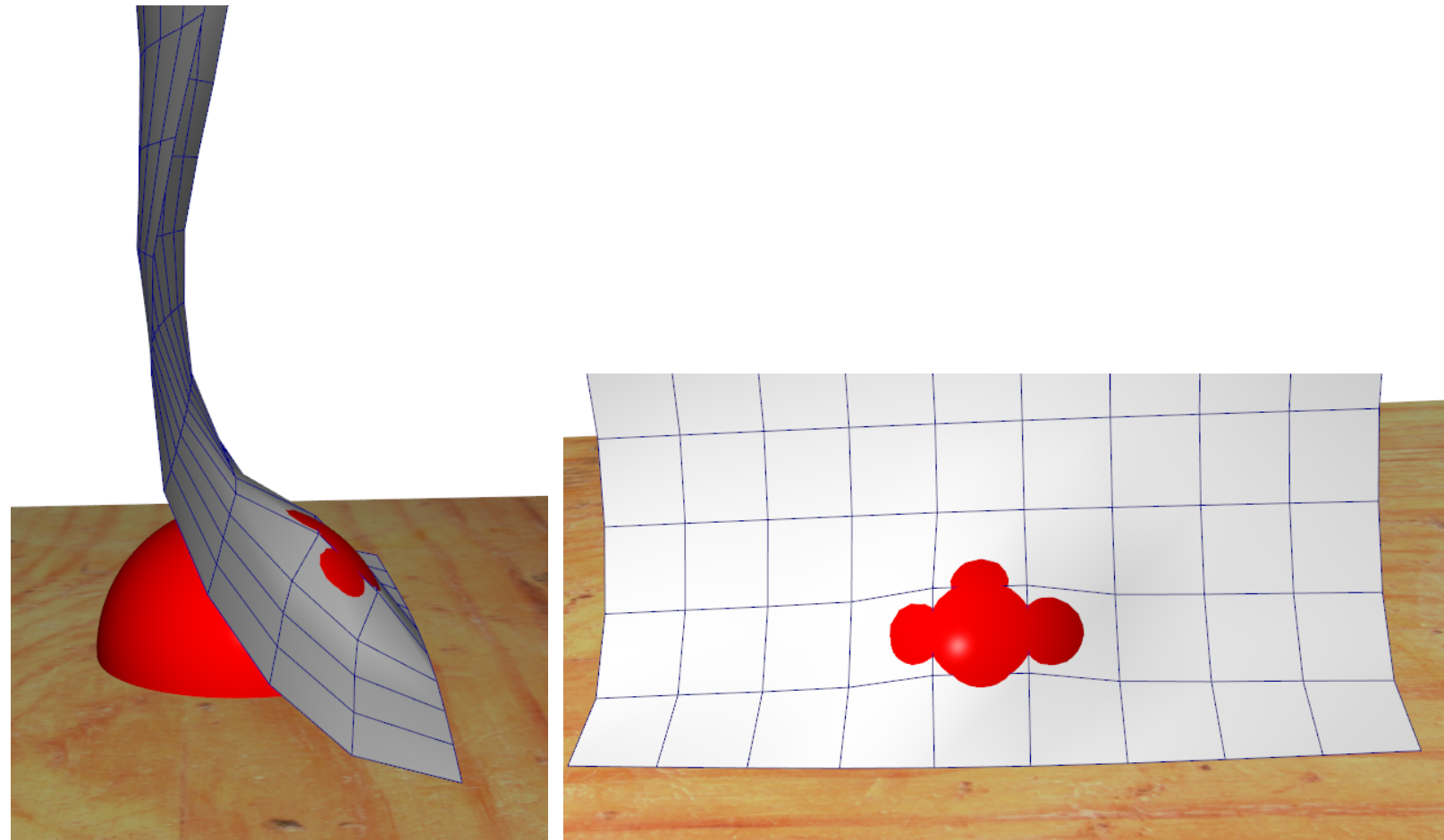
Note on Mass-Spring numerical solution

- Non-linear ODE
- Large K_{ij} : good length preservation, but stiff ODE
⇒ divergence of explicit schemes.
- Avoid explicit Euler (divergence)
- Semi-implicit Euler/Verlet works fine for low K_{ij}
Semi-implicit Euler + PBD allows simple integration + stable stiff springs
[Muller et al. [PBD](#) , [Inextensible clothing in Computer Games](#)]
- RK4 more accurate (but higher complexity than Verlet)
- Implicit Euler : requires linearization, but very stable



Collisions

- Simple approach : Handled as collision between particles and shapes
 - (+) Simple and efficient
 - (-) Collision may still appears within a triangle
 - ⇒ Exhaustive approach: edges + faces



Limitation of mass spring model and continuous model

- Does mass-spring system converge toward a unique solution when sampling increase ?

⇒ No :(

Depends on the connectivity → bad for physical accuracy

Corollary

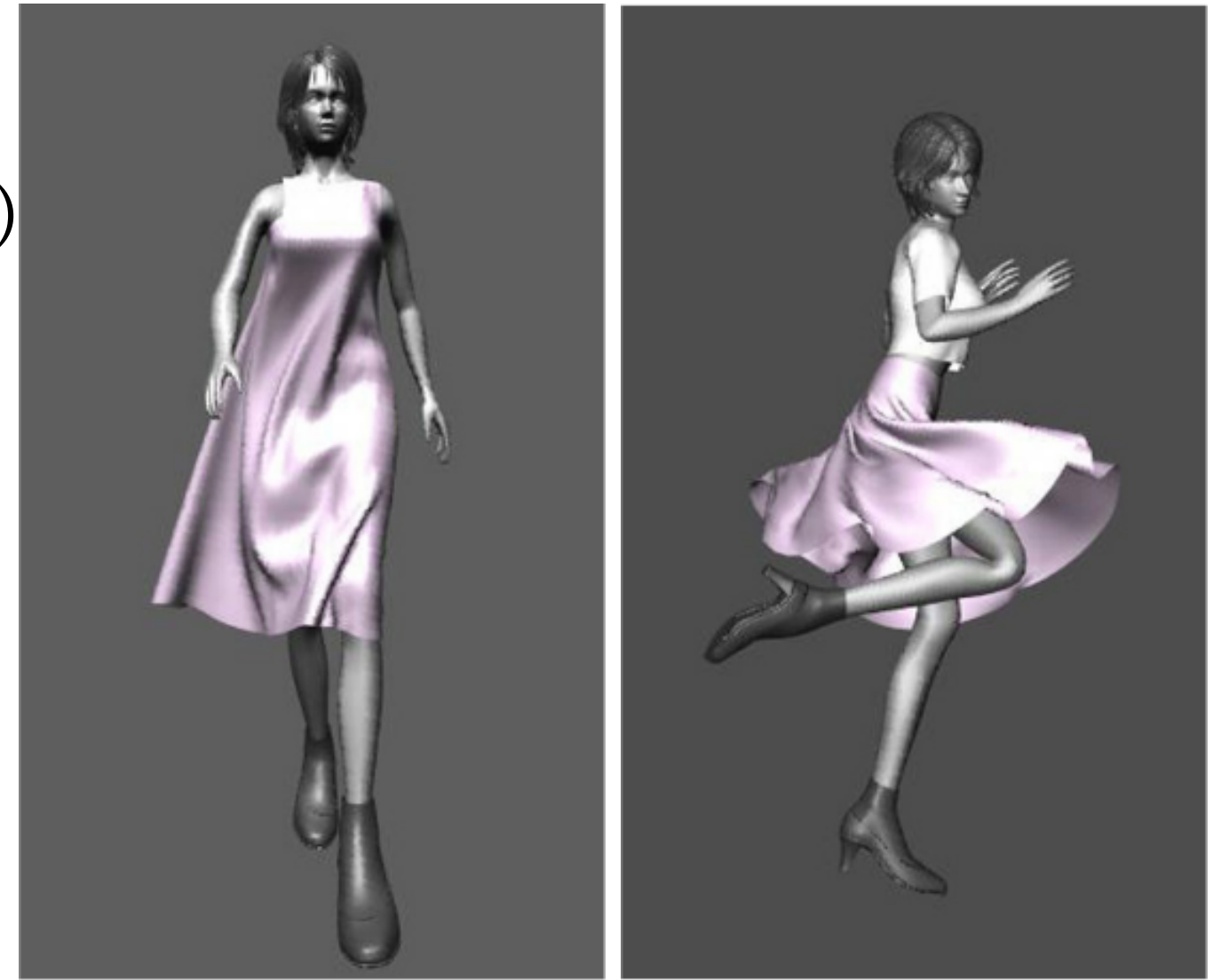
- Mass-springs work well for grid-mesh structure (draping)
- Less for arbitrary triangular meshes

1st improvment: Change toward energy formulation for bending springs (limits locking effect)

$$F = \frac{\partial E}{\partial p}$$

$$E = \frac{1}{2} K L \kappa^2, \kappa: \text{curvature}$$

[Cho et al, Stable but Responsive Cloth, ACM SIGGRAPH 2002]



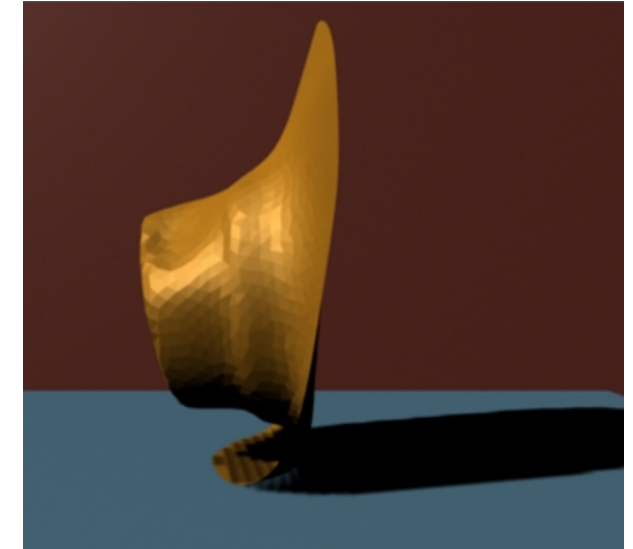
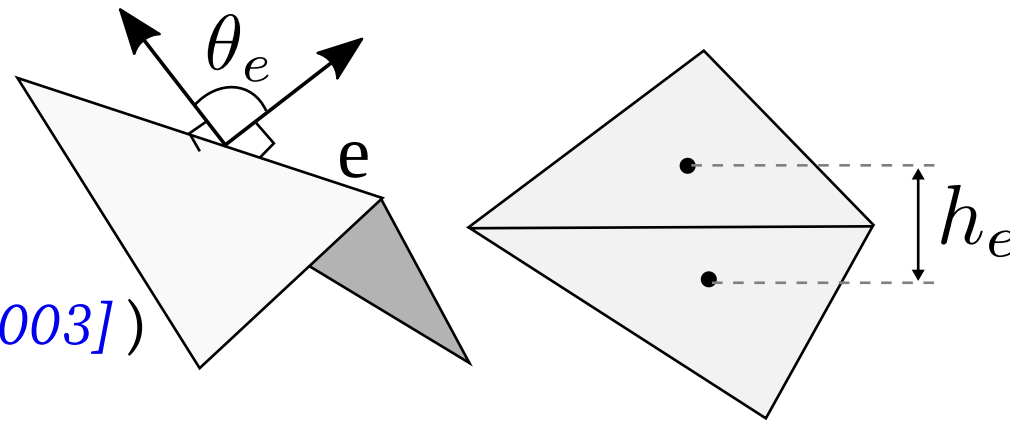
Triangle as continuous elements

- Defining Bending Energy between triangles

$$- W_B(x) = \sum_{\text{edges } e} (\theta_e - \theta_e^0) \frac{\|e^0\|}{h_e^0}$$

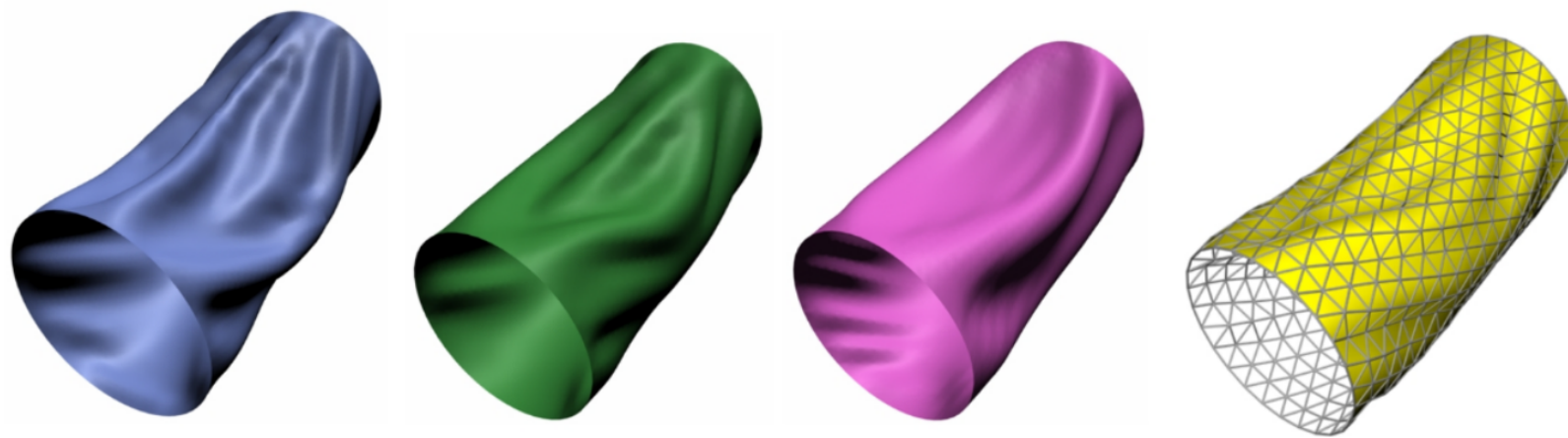
[E. Grinspun et al., Discrete Shells, SCA 2003]

(or expressed using forces in [R. Bridson et al., SCA 2003])



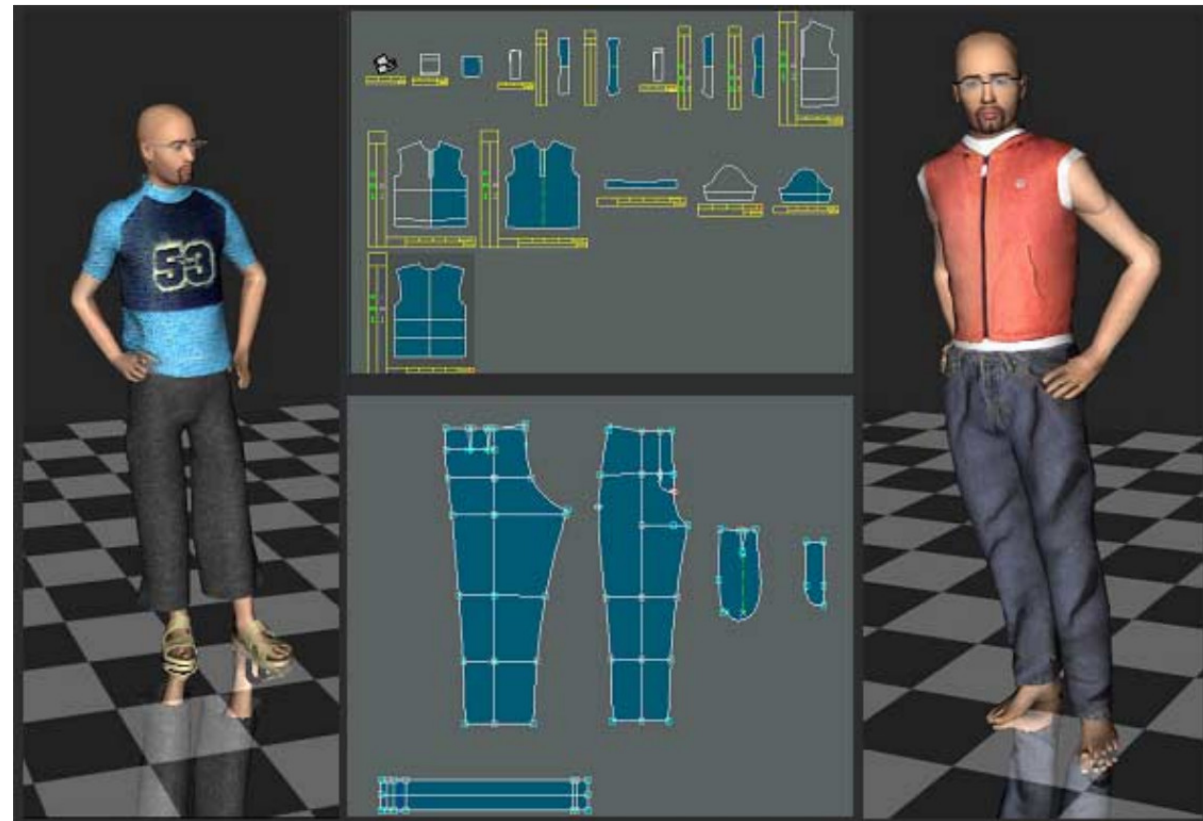
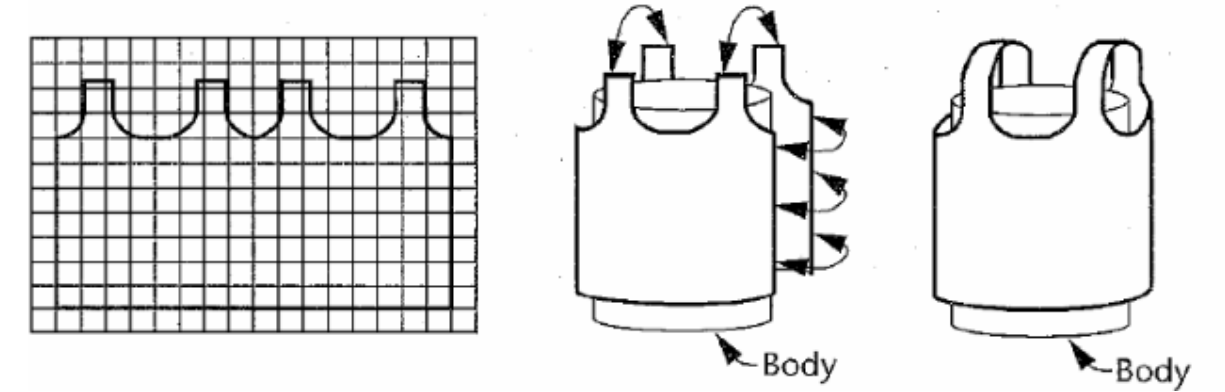
- Going toward full FEM numerical resolution

- B. Thomaszewski et al. [SCA 2006], [VRIPHYS 2008], [EG 2009].



Clothing

- Stitch 2D patterns together to generate full cloth
- Cloths are developable material (preserve length w/r their 2D patterns)

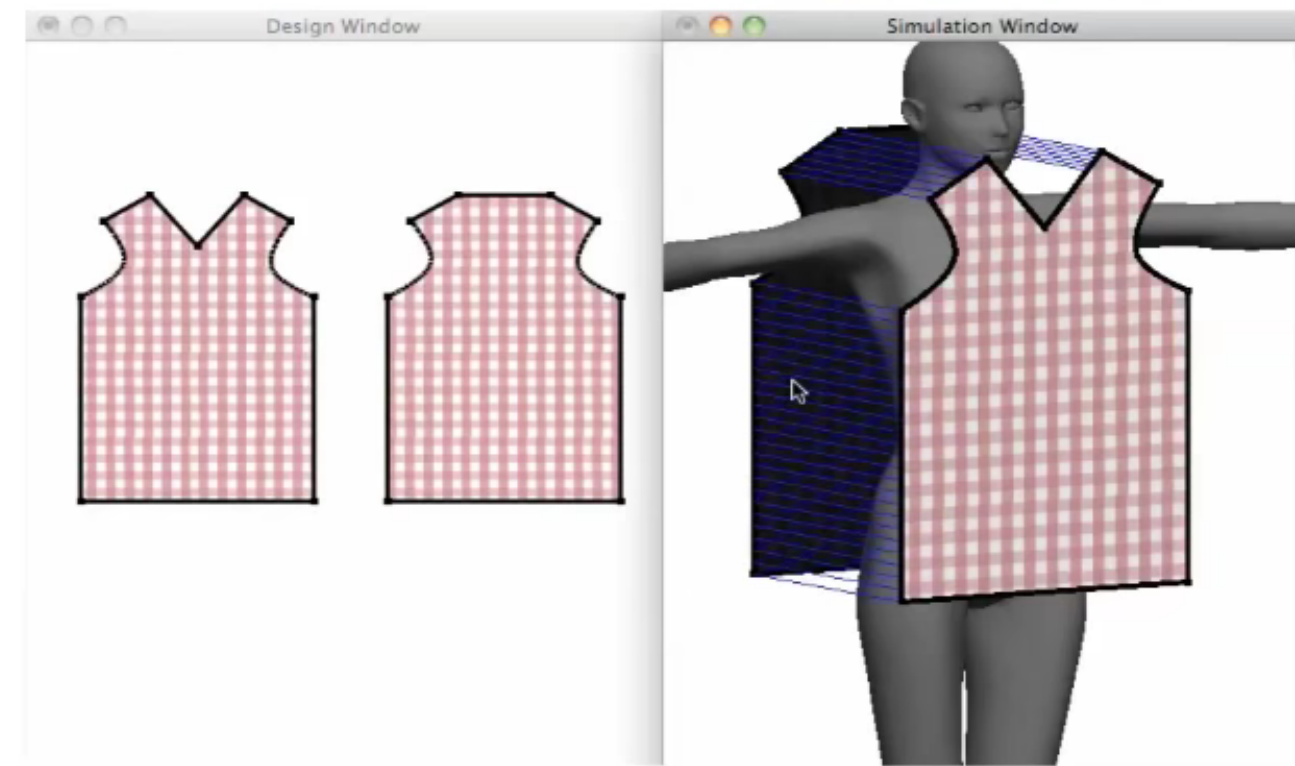


[Thalman et al. 2002]

Overview

Cloth Pattern

Cloth Simulation



[Umetani et al., 2011]

Detecting self collision

Handled as moving point in collision with moving triangle

Inputs

- Triangle $P_1(t)P_2(t)P_3(t)$, a point $P(t)$
- Each position $P_k(t) = P_k(0) + t v_{P_k}$

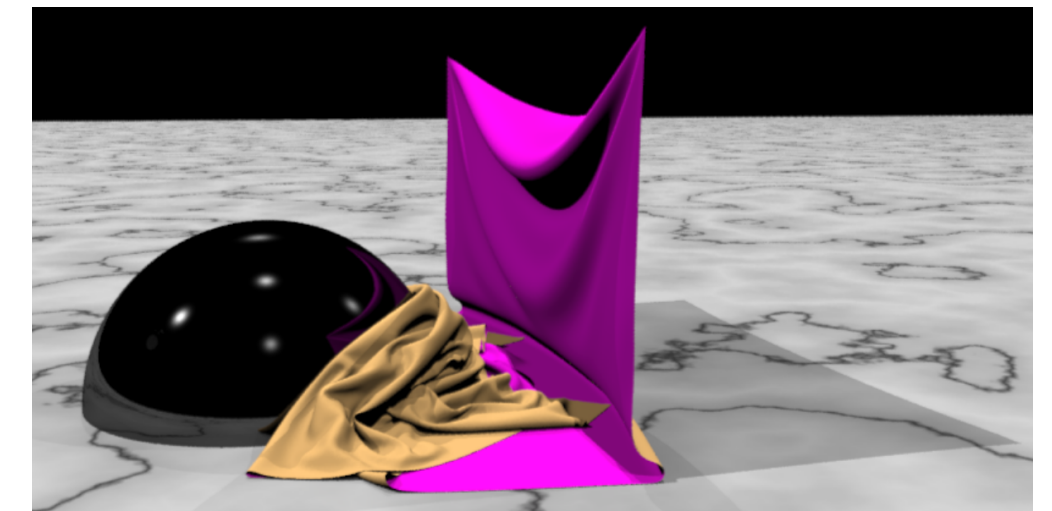
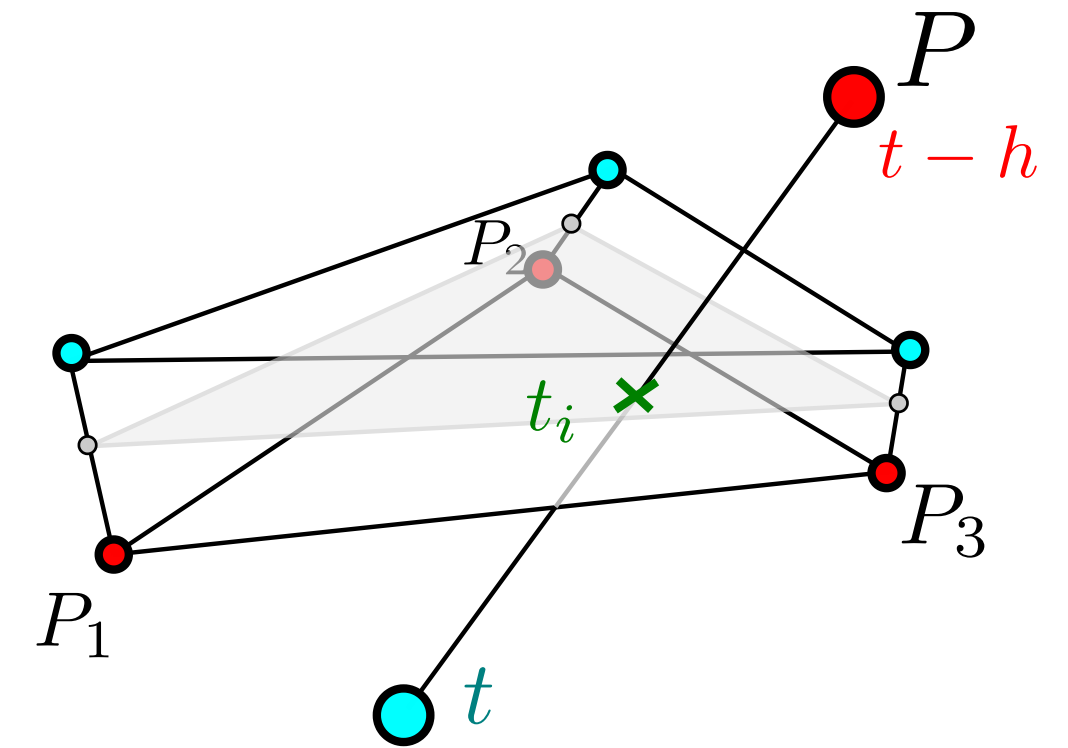
Computing intersection

Necessary condition

- Find $t_i \in [0, h]$ such that $P(t_i)$ is in triangle plane
 $(P(t_i) - P_1(t_i)) \times n(t_i) = 0$
 $n(t_i)$: normal of the triangle at time t_i

Sufficient condition

- Check $P(t_i)$ is inside the triangle
 $P(t_i) = \alpha P_1(t_i) + \beta P_2(t_i) + \gamma P_3(t_i)$
 $(\alpha, \beta, \gamma) \in [0, 1]^3, \alpha + \beta + \gamma = 1$



[X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. Graphics Interface 1997.]

[R. Bridson et al. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. ACM SIGGRAPH 2002]

Animating fluids (I)

Stable Fluid

Solving Navier-Stokes on grid

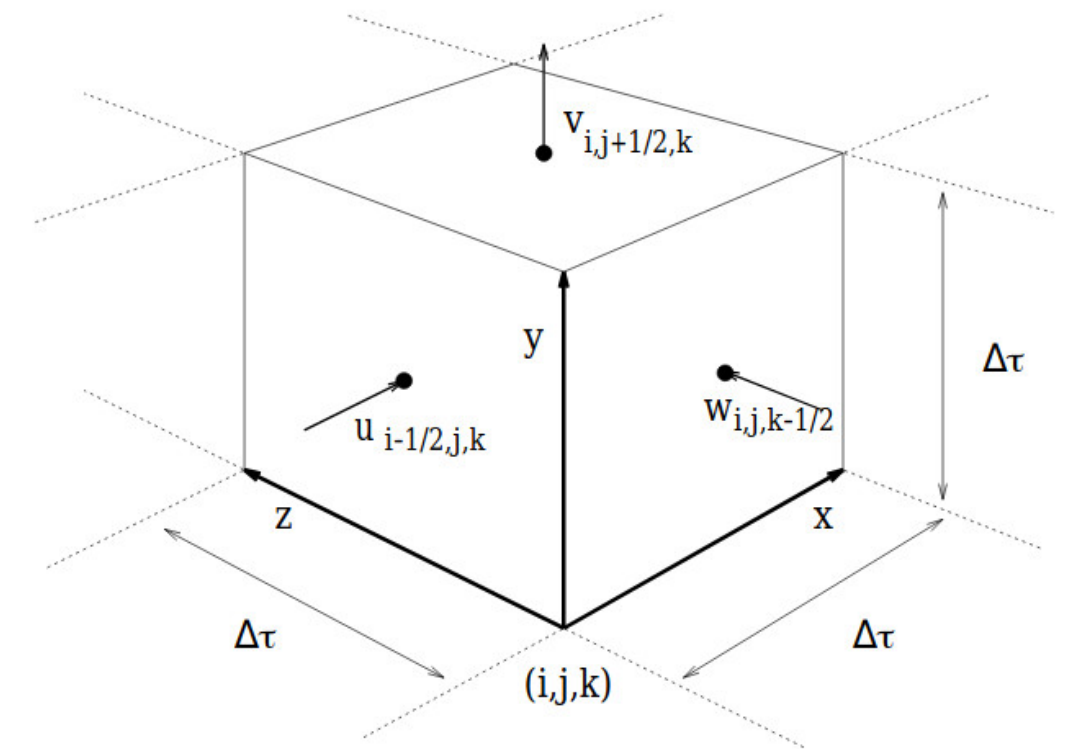
"Brute force" approach

- Rectangular grid filled with fluid
- Use finite differences on the grid for Navier-Stokes equation

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho} \nabla p + f - (u \cdot \nabla)u + \nu \Delta u$$
$$\text{div}(u) = 0$$

(-) Stability conditions

(-) Loose advection details on the grid



[Modeling the Motion of a Hot, Turbulent Gas. N Foster and D. Metaxas. SIGGRAPH 1997]

Stable Fluids - Idea

Well known improvement: **Jos Stam, Stable Fluids, ACM SIGGRAPH 1999**

$$\frac{\partial u}{\partial t} = f - (u \cdot \nabla)u + \nu \Delta u - \frac{1}{\rho} \nabla p$$

- $1/\rho \nabla p$: Pressure term only used to ensure divergence free
- Similar to Lagrange multiplier for constraints

1st Idea

Remove pressure term

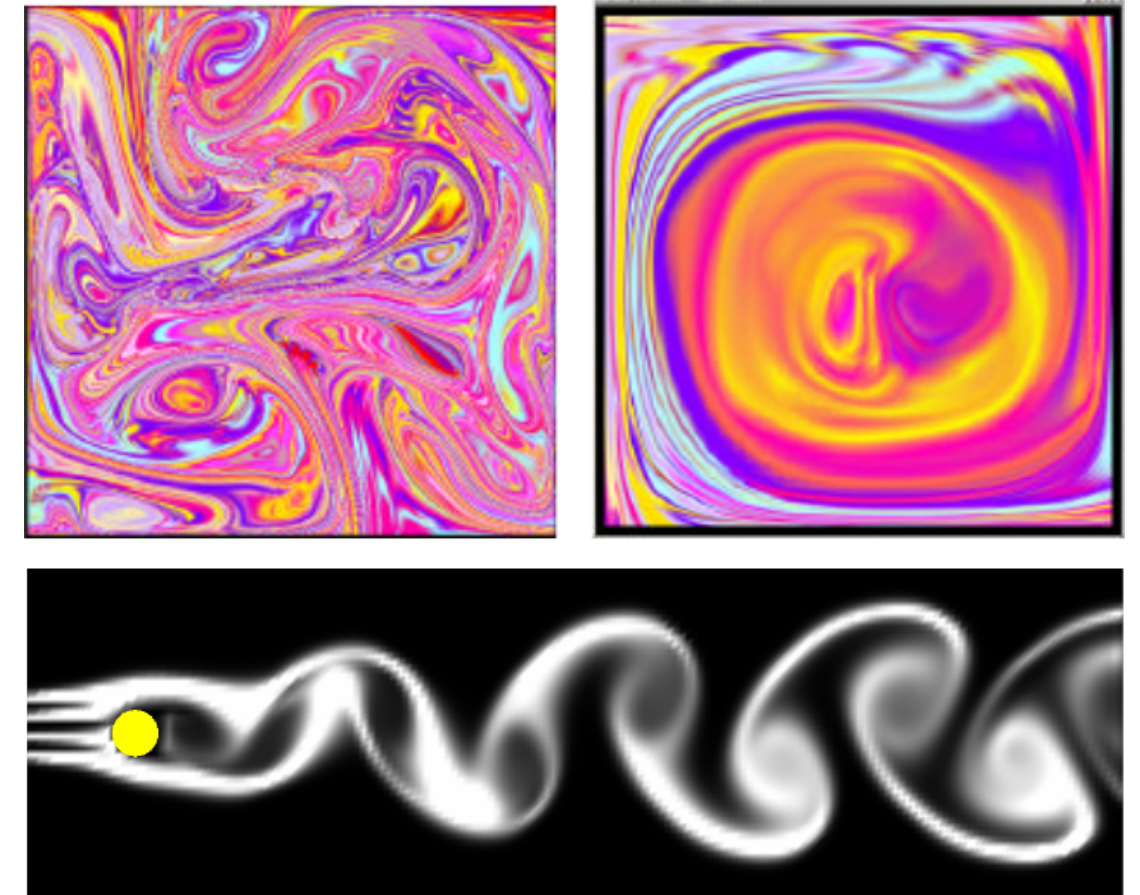
Replace by explicit projection on divergence free vector field P

$$\Rightarrow \frac{\partial u}{\partial t} = P(f - (u \cdot \nabla)u + \nu \Delta u)$$

2nd Idea

Compute each terms one after the other

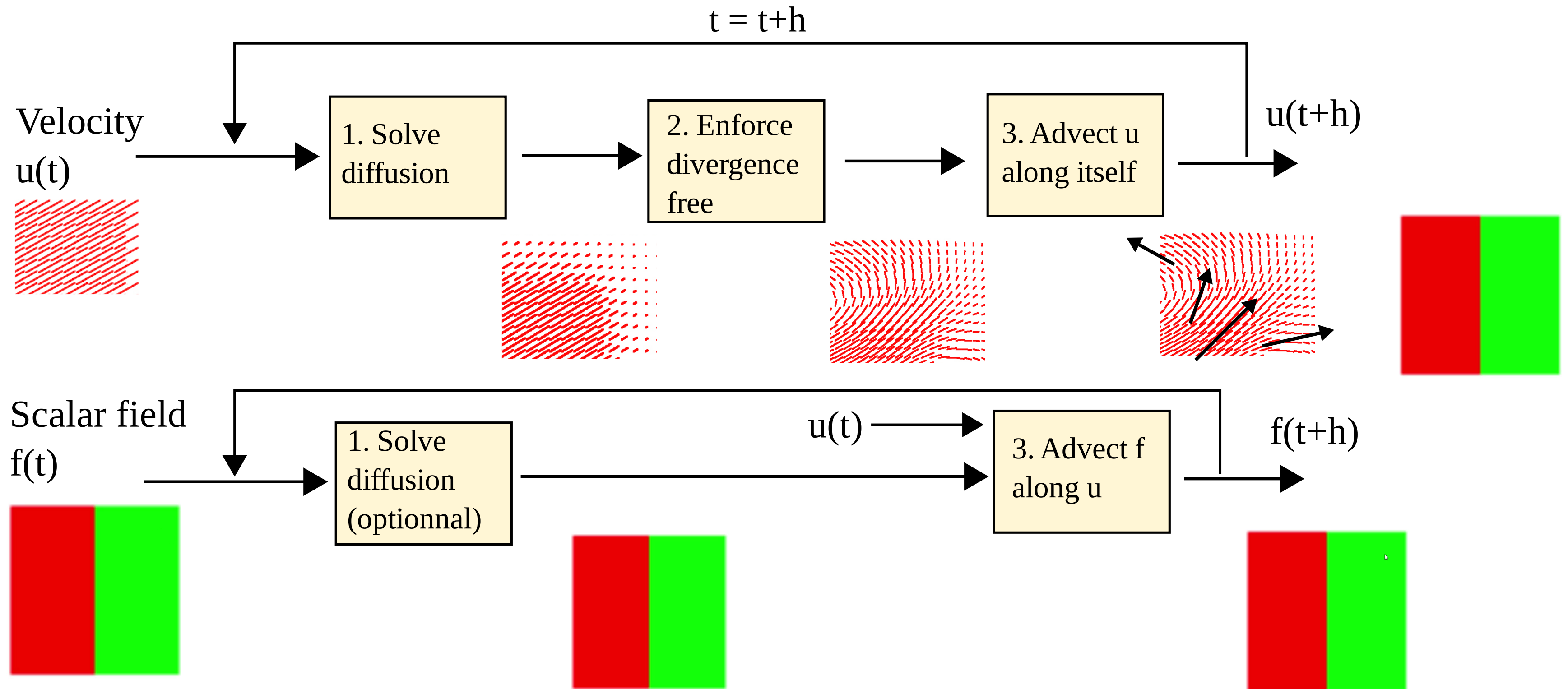
$$u^k \xrightarrow[\text{add forces } f]{\quad} u_1^k \xrightarrow[\text{diffuse } \nu \Delta u]{\quad} u_2^k \xrightarrow[\text{project } P]{\quad} u_3^k \xrightarrow[\text{advect } (u \cdot \nabla)u]{\quad} u^{k+1}$$



[*Stable Fluids*. J. Stam. SIGGRAPH 1999]

[*Real Time Fluid Dynamics for Games*. J. Stam. Game Dev. Conf. 2003]

Stable Fluids - General Algorithm



1 - Diffusion

Use finite difference on $\frac{\partial f}{\partial t} = \nu \Delta f$

Notation: $f_{x,y}^t = f(k_x \Delta x, k_y \Delta y, k_t \Delta t)$

Explicit schemes may oscillates/diverge for large time steps

⇒ Use implicit scheme for unconditional stability

$$\frac{f_{x,y}^{k+1} - f_{x,y}^k}{\Delta t} = \nu \left(\frac{f_{x+1,y}^{k+1} - 2f_{x,y}^{k+1} + f_{x-1,y}^{k+1}}{(\Delta x)^2} + \frac{f_{x,y+1}^{k+1} - 2f_{x,y}^{k+1} + f_{x,y-1}^{k+1}}{(\Delta y)^2} \right)$$

Assuming $\Delta x = \Delta y = 1$

$$(1 + 4\nu\Delta t) f_{x,y}^{k+1} - \nu\Delta t(f_{x+1,y}^{k+1} + f_{x-1,y}^{k+1} + f_{x,y+1}^{k+1} + f_{x,y-1}^{k+1}) = f_{x,y}^k$$

Use Gauss-Seidel iterative method to solve the sparse linear system

Initialize $f^{k+1} = f^k$

for $i = 1..N_{\max}$

$$f_{x,y}^{k+1} = \frac{1}{1+4a} (f_{x,y}^k + a(f_{x-1,y}^{k+1} + f_{x+1,y}^{k+1} + f_{x,y-1}^{k+1} + f_{x,y+1}^{k+1})) , \quad a = \nu \Delta t$$



2 - Advection

Advection = move some function along given velocity u .

- Advecting a scalar field f along u

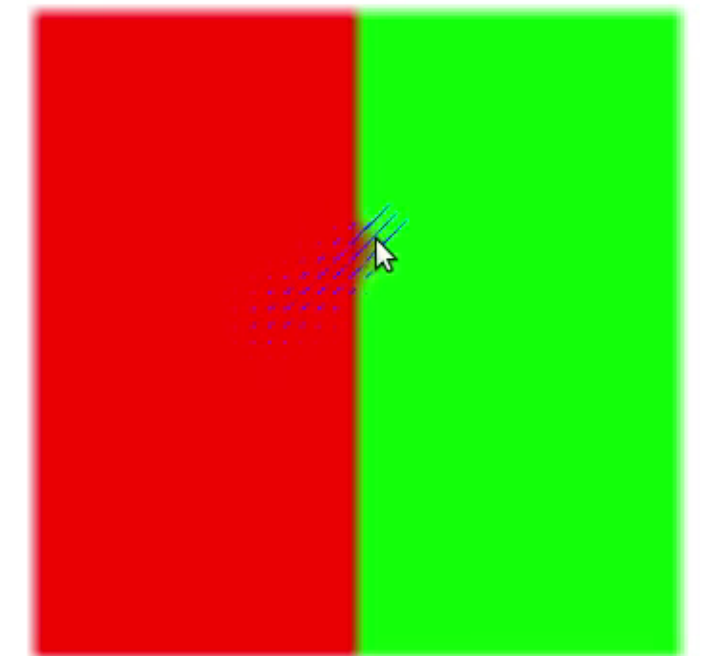
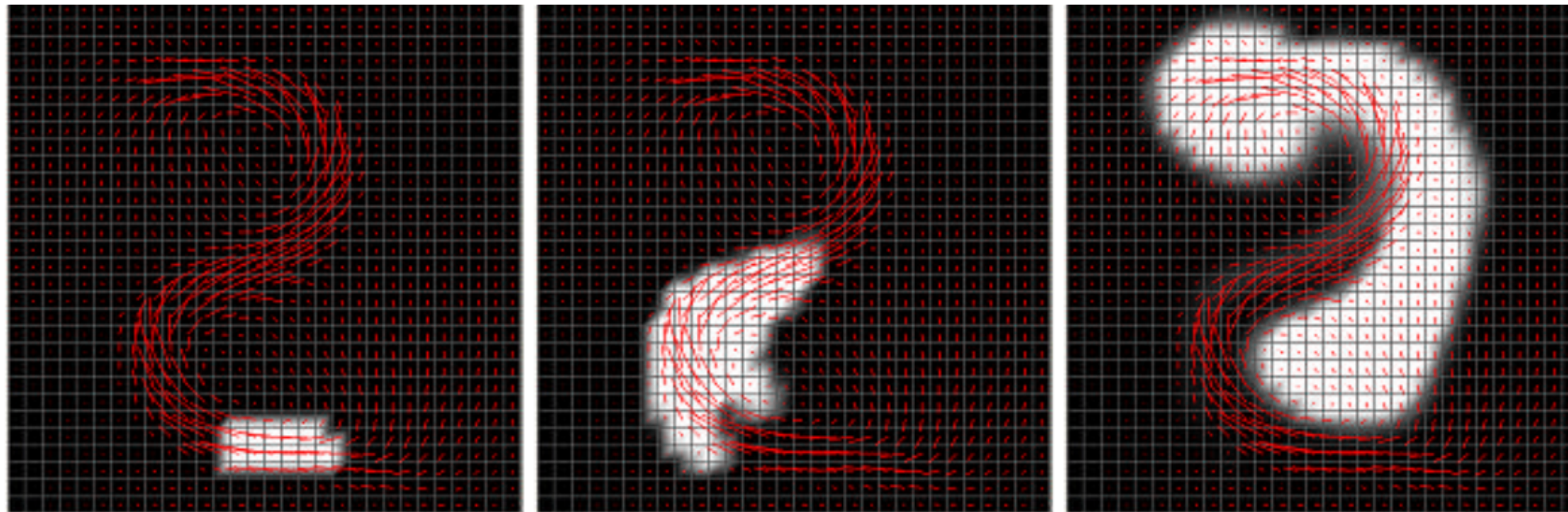
$$\frac{\partial f}{\partial t}(p, t) + u(p, t) \cdot \nabla f = 0$$

- Advecting a vector field f along u

$$\frac{\partial f}{\partial t}(p, t) + (u(p, t) \cdot \nabla) f = 0$$

- In Navier-Stokes advect the velocity itself $f = u$

- Can also advect density, color, texture coordinates, etc. to visualize the motion.



2 - Computing advection

Advecting generic value f along u

Idea Compute value of f at time t at fixed position grid p in moving back at $t - \Delta t$.

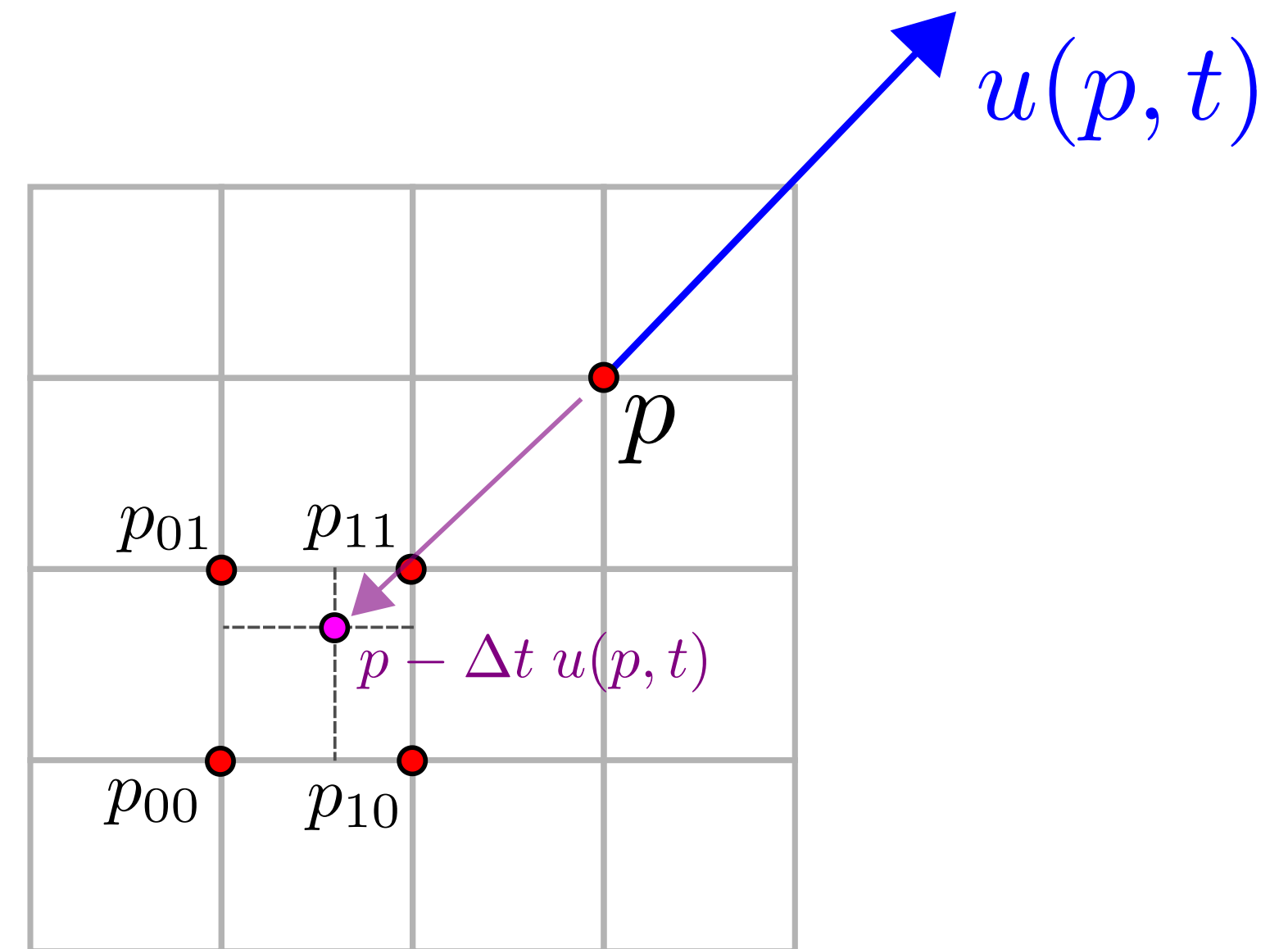
Value of f advected at point p at time t was at position $p_{prev} = p - \Delta t v(p, t)$ at time $t - \Delta t$.

$$\Rightarrow f(p, t) = f(p_{prev}, t - \Delta t)$$

p_{prev} is not a grid point coordinates: Use interpolation

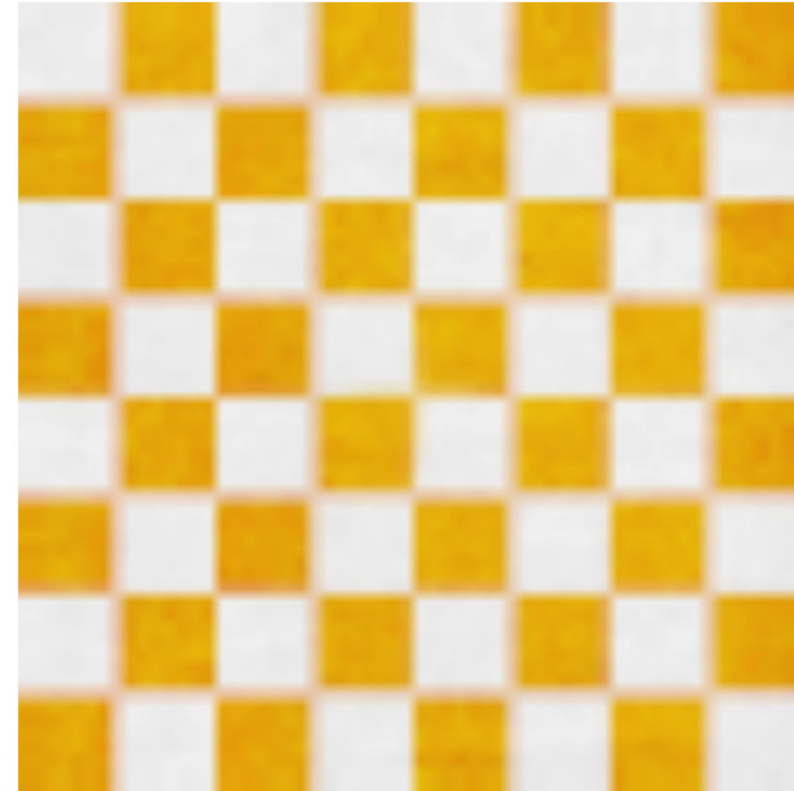
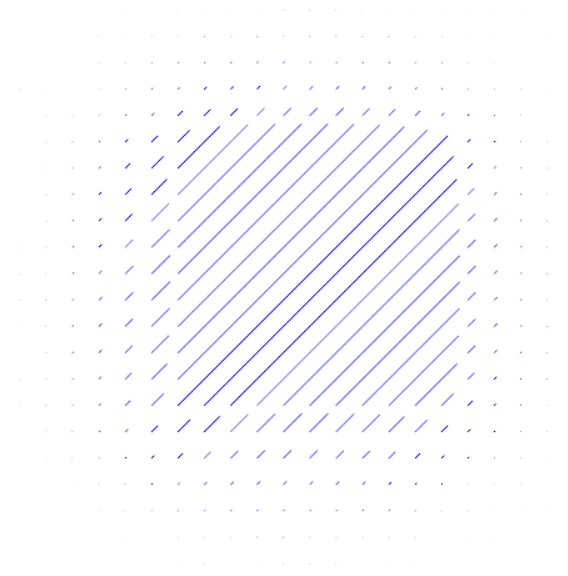
Can use Bilinear interpolation

$$f(p_{prev}) = (1 - \alpha)(1 - \beta)p_{00} + (1 - \alpha)\beta p_{01} + \alpha(1 - \beta)p_{10} + \alpha\beta p_{11}$$

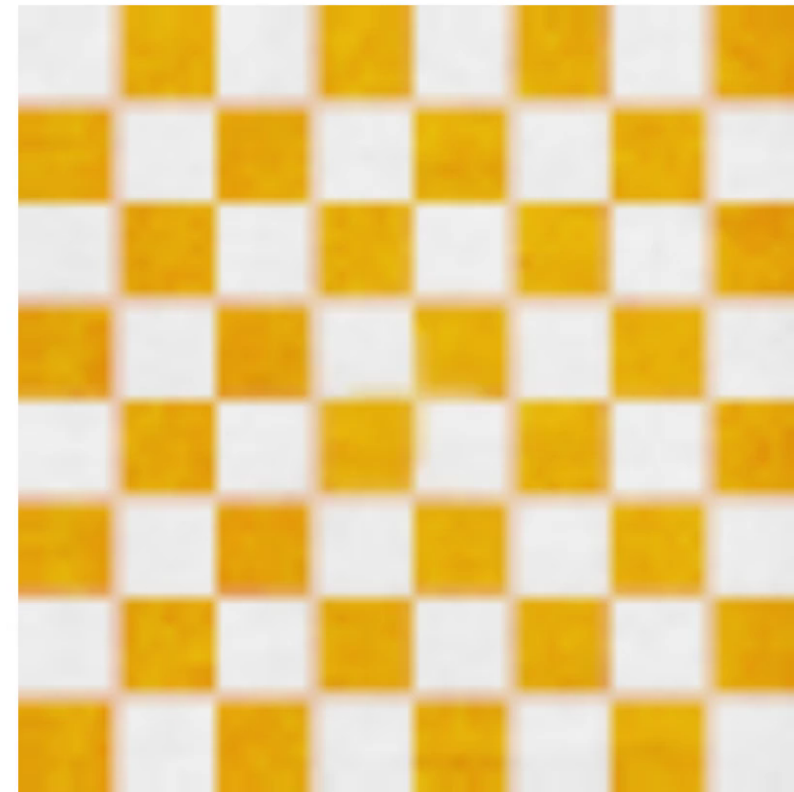
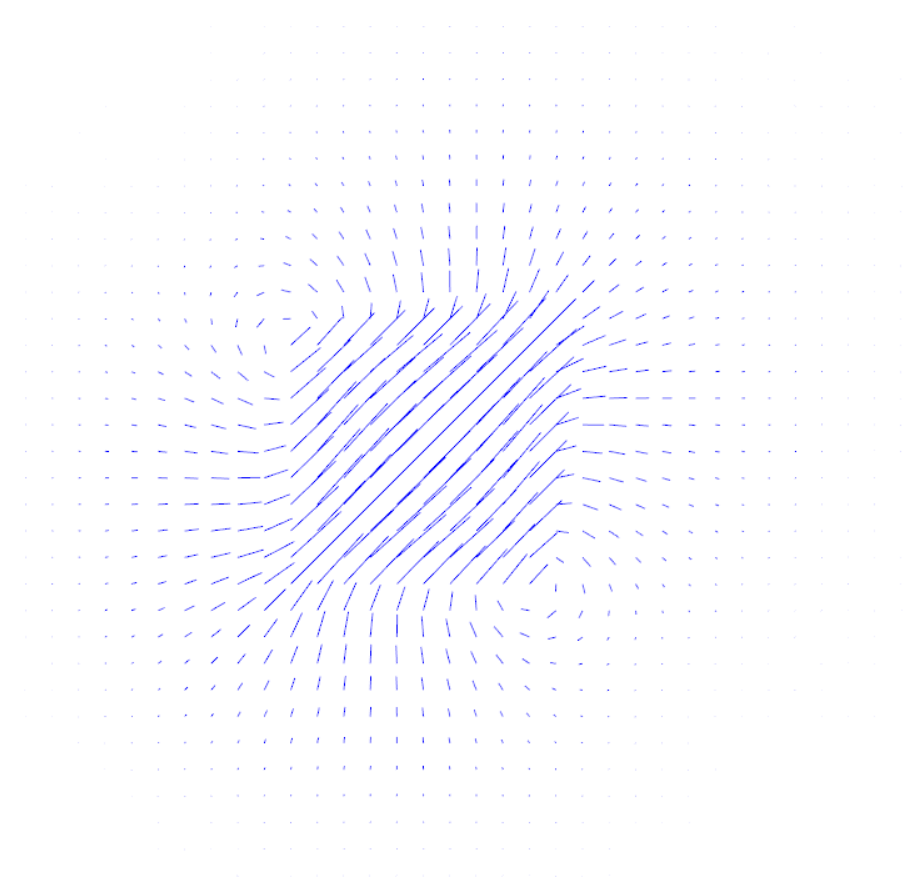


3 - Divergence Free Vector Field

Before projection:



After projection:



$$\operatorname{div}(v) = 0$$

$$v_{x+1,y}^x + v_{x-1,y}^x + v_{x,y+1}^y + v_{x,y-1}^y = 0$$

3 - Projection to divergence free vector field

Consider a general vector field w

Helmoltz decomposition: $w = u + v$

- u : Divergence free vector field such that $\text{div}(u) = 0$
- v : Gradient field $v = \nabla q$, q scalar field.

q satisfies a Poisson equation

$$\text{div}(w) = \underbrace{\text{div}(u)}_{=0} + \text{div}(v) \Rightarrow \text{div}(w) = \underbrace{\text{div}(\nabla q)}_{\Delta q}$$

Method- Given an input field w

1. Compute q as solution of $\Delta q = \text{div}(w)$
2. Compute $u = w - \nabla q$



3 - Projection to divergence free vector field (Algo)

Input vector field $w = (w^x, w^y)$

Note: we assume in the following $\Delta x = \Delta y = 1$

1 - Compute $d = \text{div}(w)$

$$d_{x,y} = (w_{x+1,y}^x - w_{x-1,y}^x + w_{x,y+1}^y - w_{x,y-1}^y) / 2$$

2 - Compute q in solving $\Delta q = b$

$$(q_{x+1,y} + q_{x-1,y} - 2q_{x,y}) + (q_{x,y+1} + q_{x,y-1} - 2q_{x,y}) = d_{x,y}$$
$$\Rightarrow 4q_{x,y} = q_{x+1,y} + q_{x-1,y} + q_{x,y+1} + q_{x,y-1} - d_{x,y}$$

ex. Numerical iterations using Gauss Seidel

Initialize $q = 0$

For $i = [1..N_{\max}]$

$$q_{x,y} = 1/4 (q_{x+1,y} + q_{x-1,y} + q_{x,y+1} + q_{x,y-1} - d_{x,y})$$

3 - Compute $u = w - \nabla q$

$$u_{x,y} = w_{x,y} - (q_{x+1,y} - q_{x-1,y}, q_{x,y+1} - q_{x,y-1}) / 2$$

Handling boundaries

Boundaries $x = 0, x = N_x - 1, y = 0, y = N_y - 1$
need special care

- For density

Assume value C^0 continuity on the boundary

Row/Column $f_{x,0} = f_{x,1}, f_{0,y} = f_{1,y}$ etc.

- For velocity: $f = (f^x, f^y)$

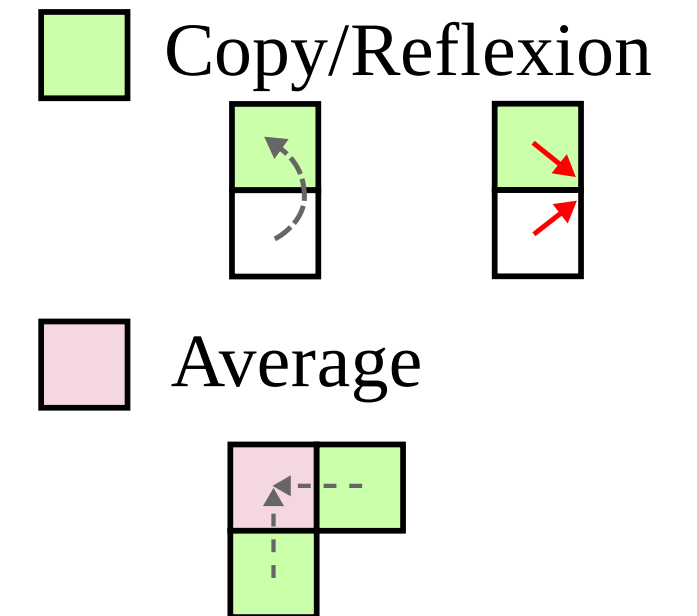
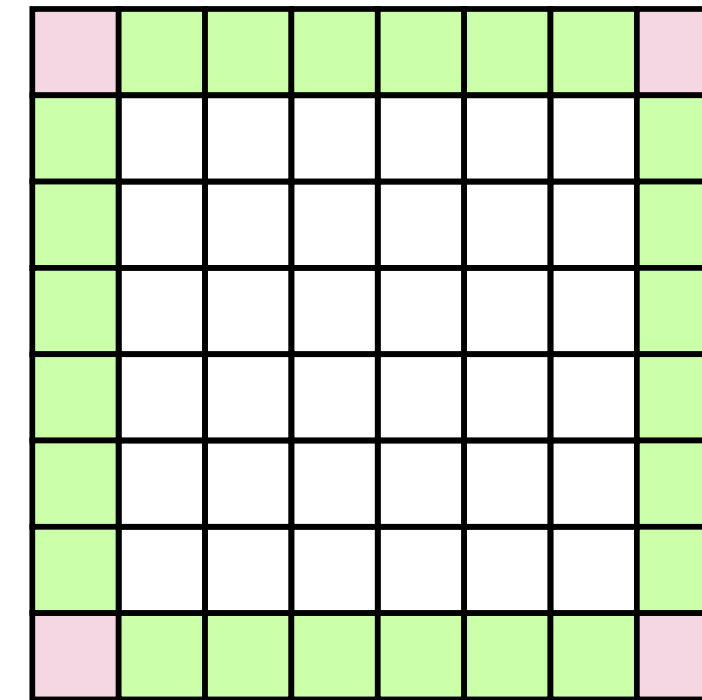
Assume reflexion on walls

Row: $f_{x,0} = (f_{x,1}^x, -f_{x,1}^y)$

Column: $f_{0,y} = (-f_{1,y}^x, f_{1,y}^y)$

- In all cases: Average value for corners

$f_{0,0} = (f_{1,0} + f_{0,1})/2$, etc.



Stable fluids example

Animating fluids (II)

- Procedural
- Level Set
- PIC/FLIP
- SPH

Procedural models for free surface water

Often based on **Shallow water equation**

neglect depth velocity component

[A. Fournier, W. T. Reeves. *A simple Model of Ocean Waves*, ACM SIGGRAPH 1986]

[D. Hinsiger et al. *Interactive Animation of Ocean Waves*. SCA 2002]

[Jerry Tessendorf, *Simulating Ocean Water*, ACM SIGGRAPH Course Notes 2004]



ex. *Trochoid/Gerstner models*

Particles following circular trajectories, waves propagates

$$\mathbf{x} - \mathbf{x}_0 = \sum_i a_i \mathbf{k}_i / \|\mathbf{k}_i\| \sin(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t + \phi_i)$$

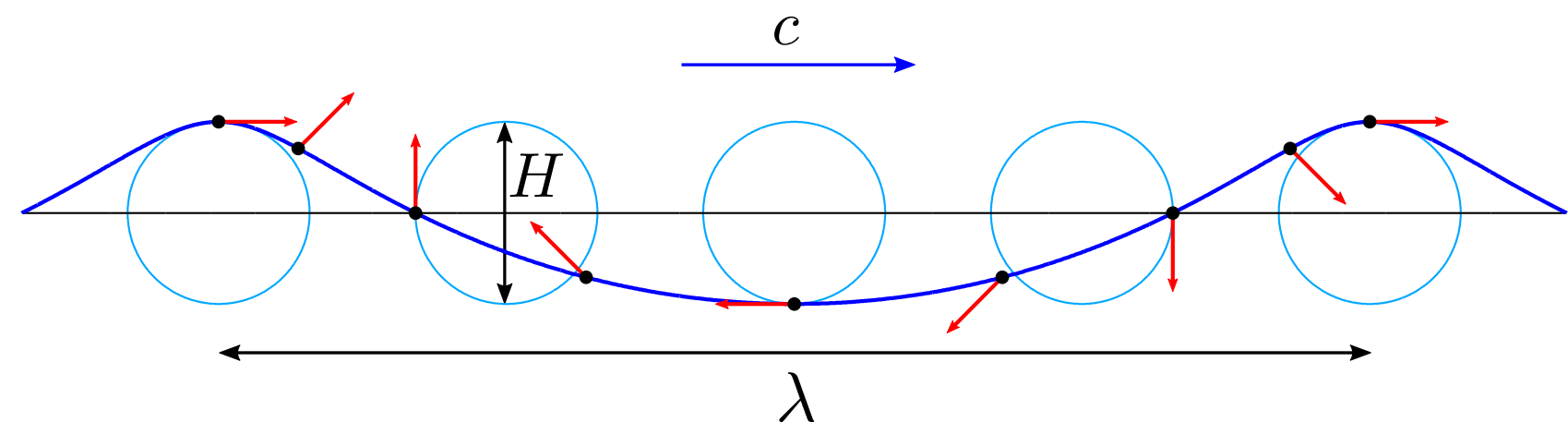
$$z - z_0 = \sum_i a_i \cos(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t + \phi_i)$$

$$\omega_i^2 = g k_i - \text{dispersion}$$

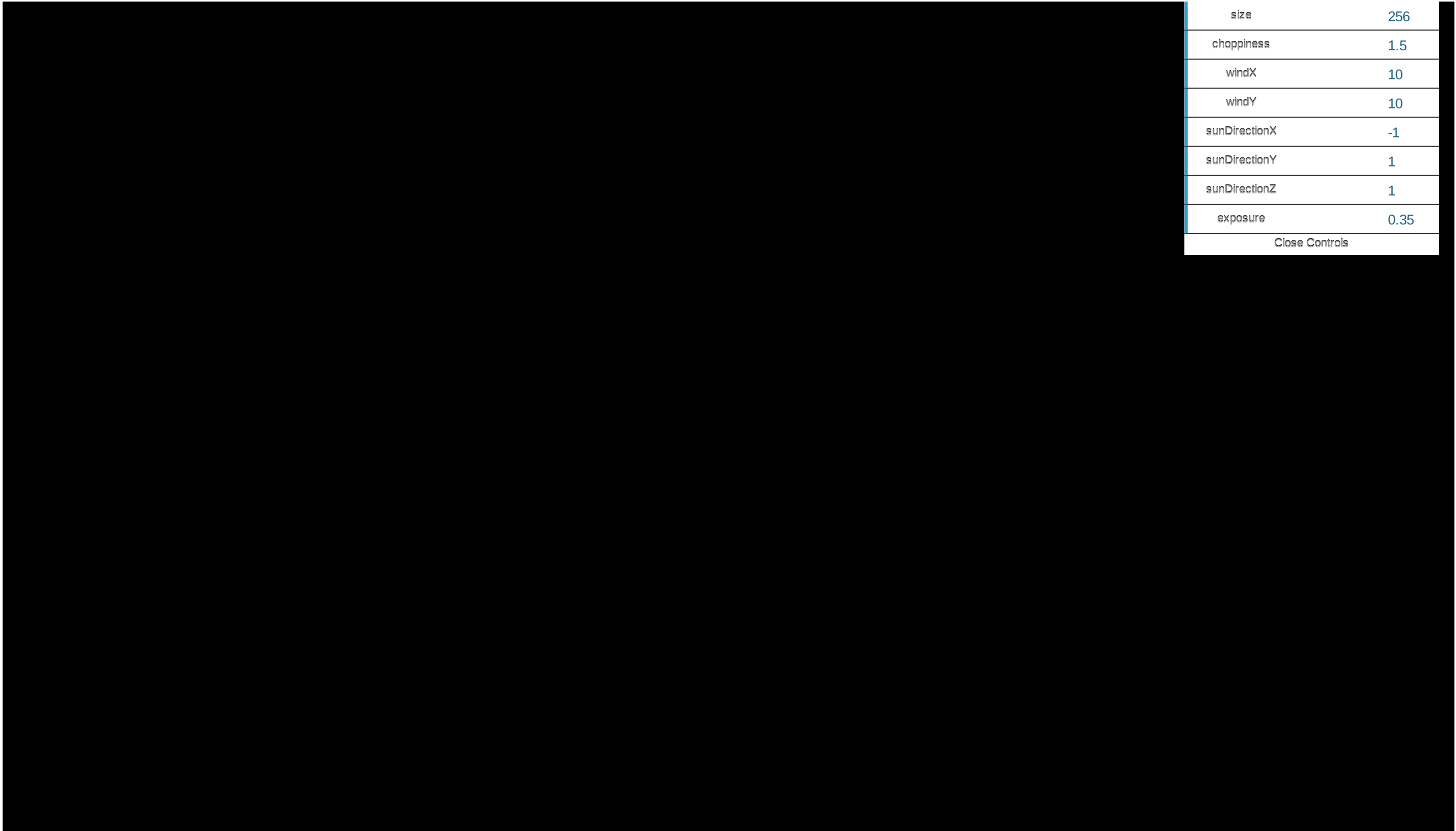
Used for procedural ocean modeling

(+) Simple and scalable

(-) Interaction with other objects



Free surface water - Example



size	256
choppiness	1.5
windX	10
windY	10
sunDirectionX	-1
sunDirectionY	1
sunDirectionZ	1
exposure	0.35
Close Controls	

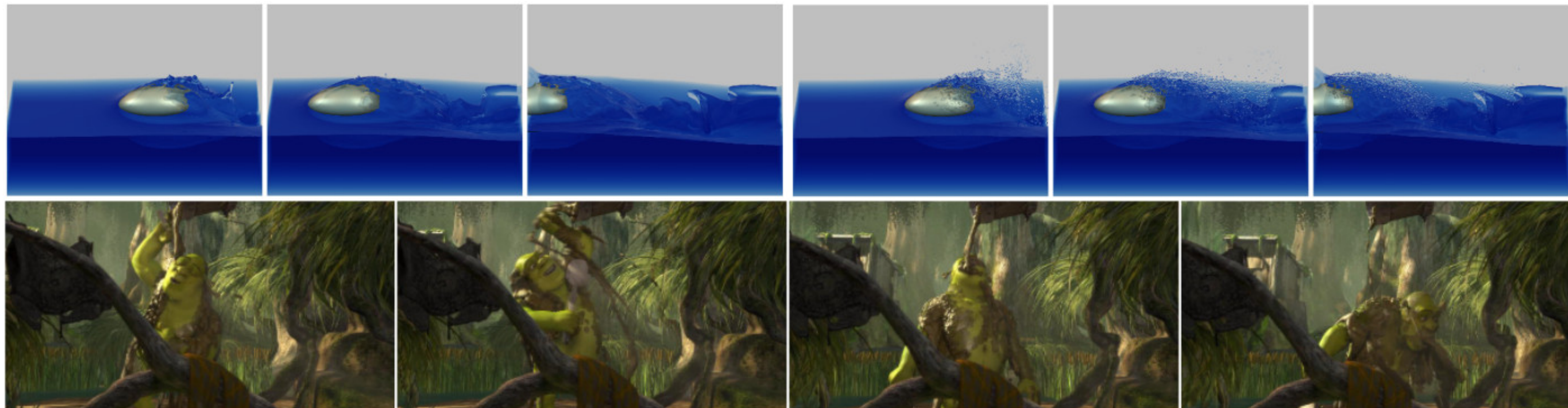
Level Set methods introduction

Eulerian models (ex. stable fluids) do not handle natively free boundaries such as fluid/air.

Idea: Track surface boundary using implicit surfaces

- Encode fluid volume by $\Omega = \{p \in \mathbb{R}^3, \varphi(p) = 0\}$, φ stored within a 3D grid.
- Solve Navier-Stokes equation within Ω
- Deform fluid volume using implicit surface deformation $\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0$

Introduced by Ronald Fedkiw, Nick Foster, Stanley Osher (ex. [\[Practical Animation of Liquids, ACM SIGGRAPH 2001\]](#))



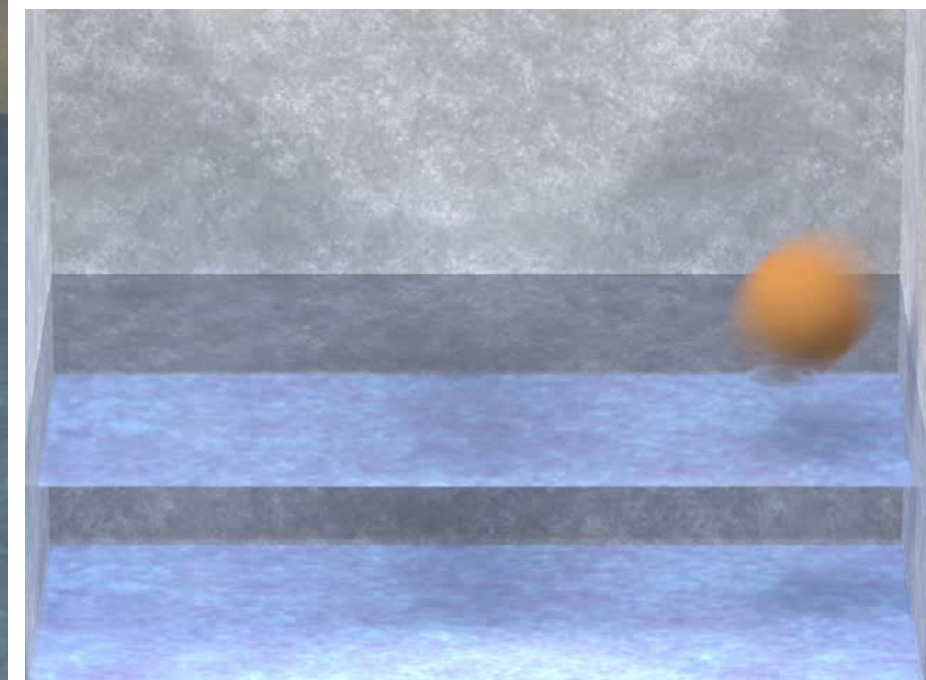
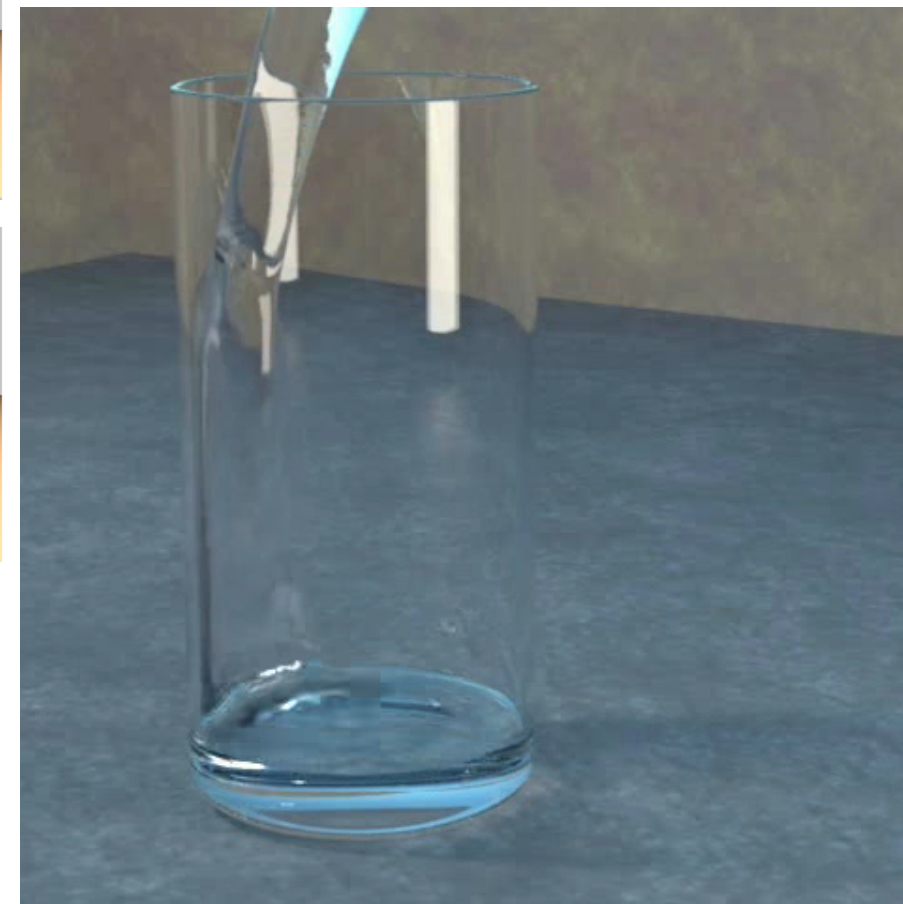
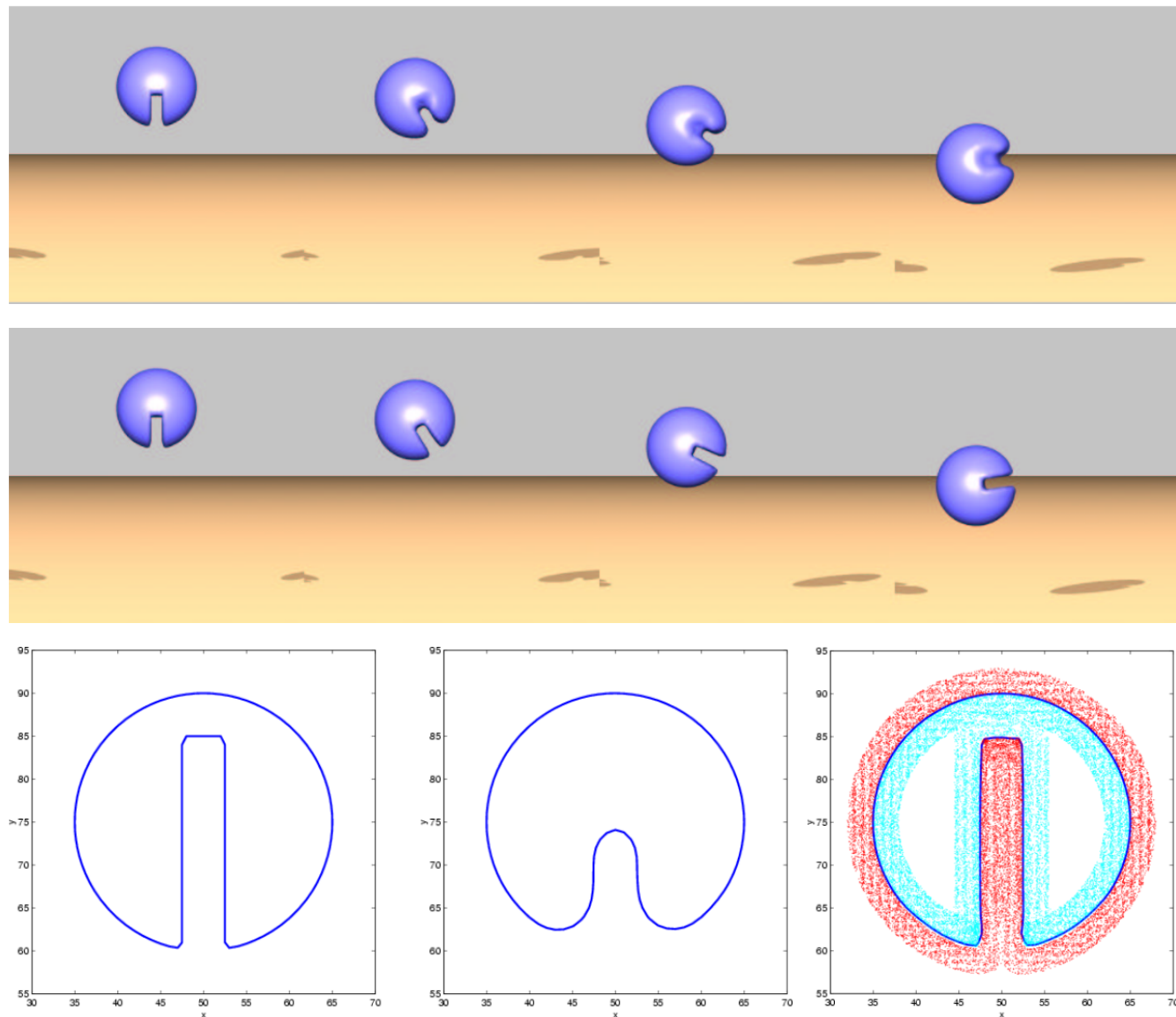
Particles Level Set

Limitation of level sets: smoothing and loss of volume from grid interpolation

⇒ Use of **Particle Level Set Method**

[D. Enright et al. Hybrid Particle Level Set Method for Improved Interface Capturing. J. Comp. Physics 2001]

[D. Enright et al. Animation and Rendering of Complex Water Surfaces, ACM SIGGRAPH 2002]



D. Enright et al.

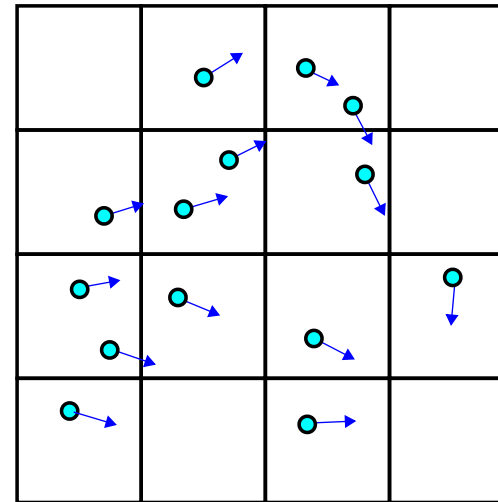
PIC/FLIP (Material Point Method)

Mix between particles and grid based approach.

- Particles: good for advection
- Grid: forces, pressure, viscosity

u_p : velocity on particle

u_g : velocity on grid



- **PIC** approach - Transfert velocity from grid to particles

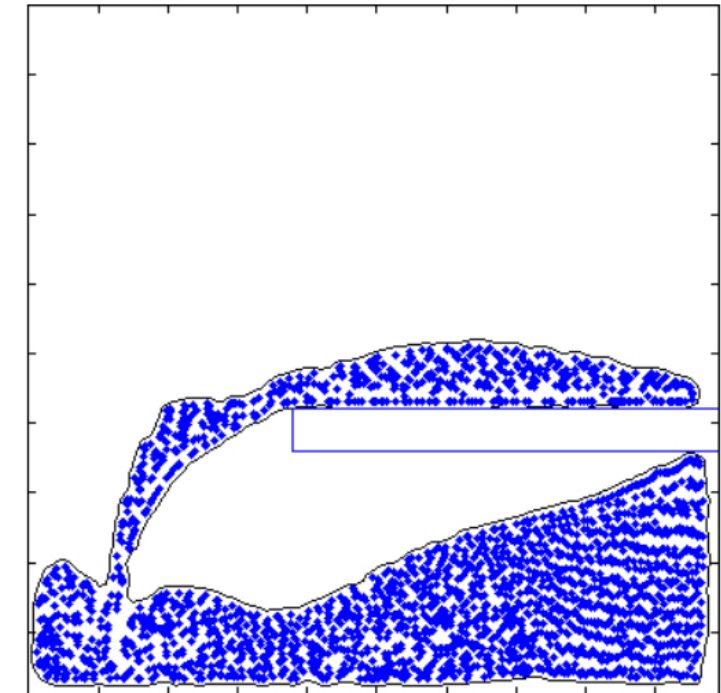
$$u_p^{k+1} = \text{interp}(u_g^{k+1}, p^{k+1})$$

- **FLIP** approach - Add velocity difference from grid to particles.

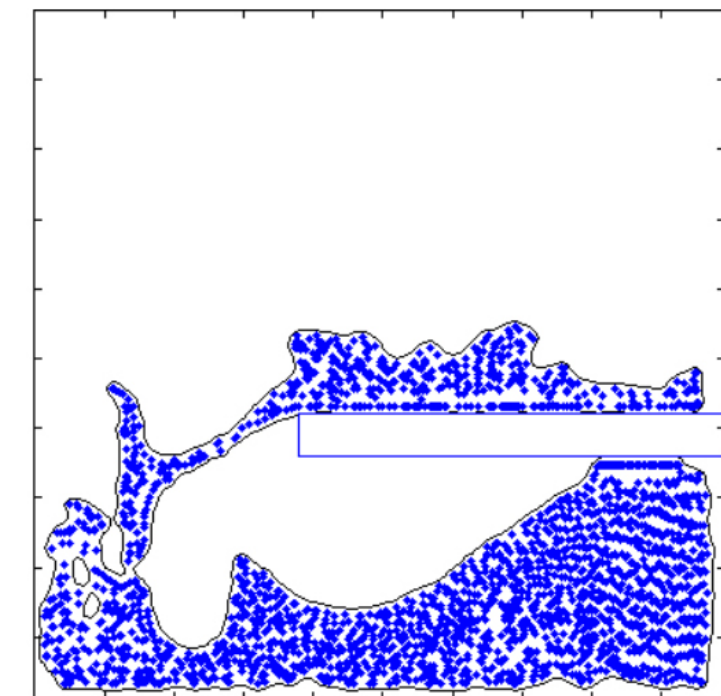
$$u_p^{k+1} = v_p^k + (\text{interp}(u_g^{k+1}, p^{k+1}) - \text{interp}(u_g^k, p^k))$$

- **PIC/FLIP** : blending b/w two approaches

[Y. Zhu and R. Bridson, Animating Sand as a Fluid, ACM SIGGRAPH 2005]



PIC: Stable, smoothed-out



FLIP: Details, few dissipation

MAC grid

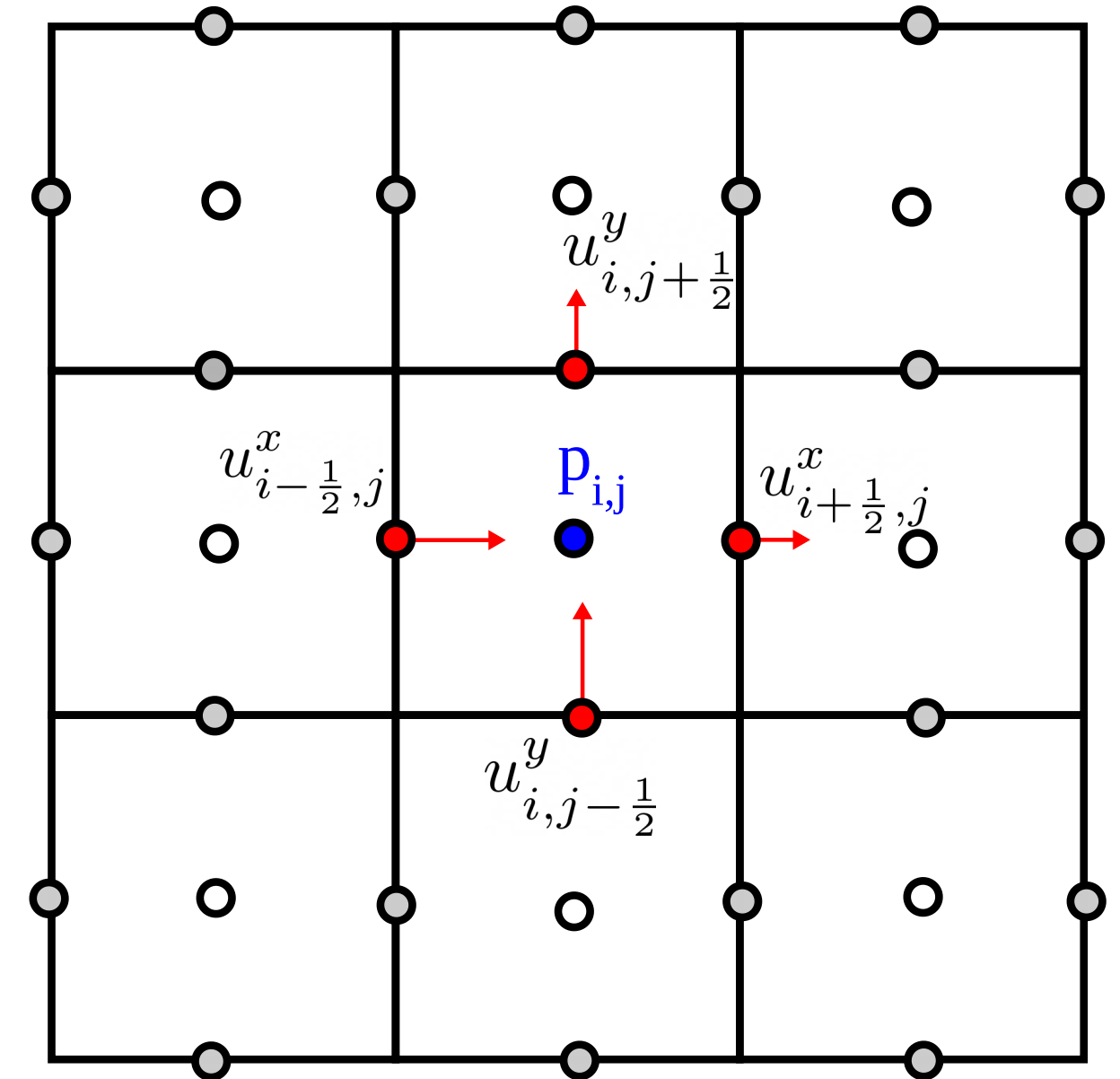
MAC = Marker And Cell

Staggered grid b/w scalar and velocity

Widely used grid storage to handle velocity and scalar values.

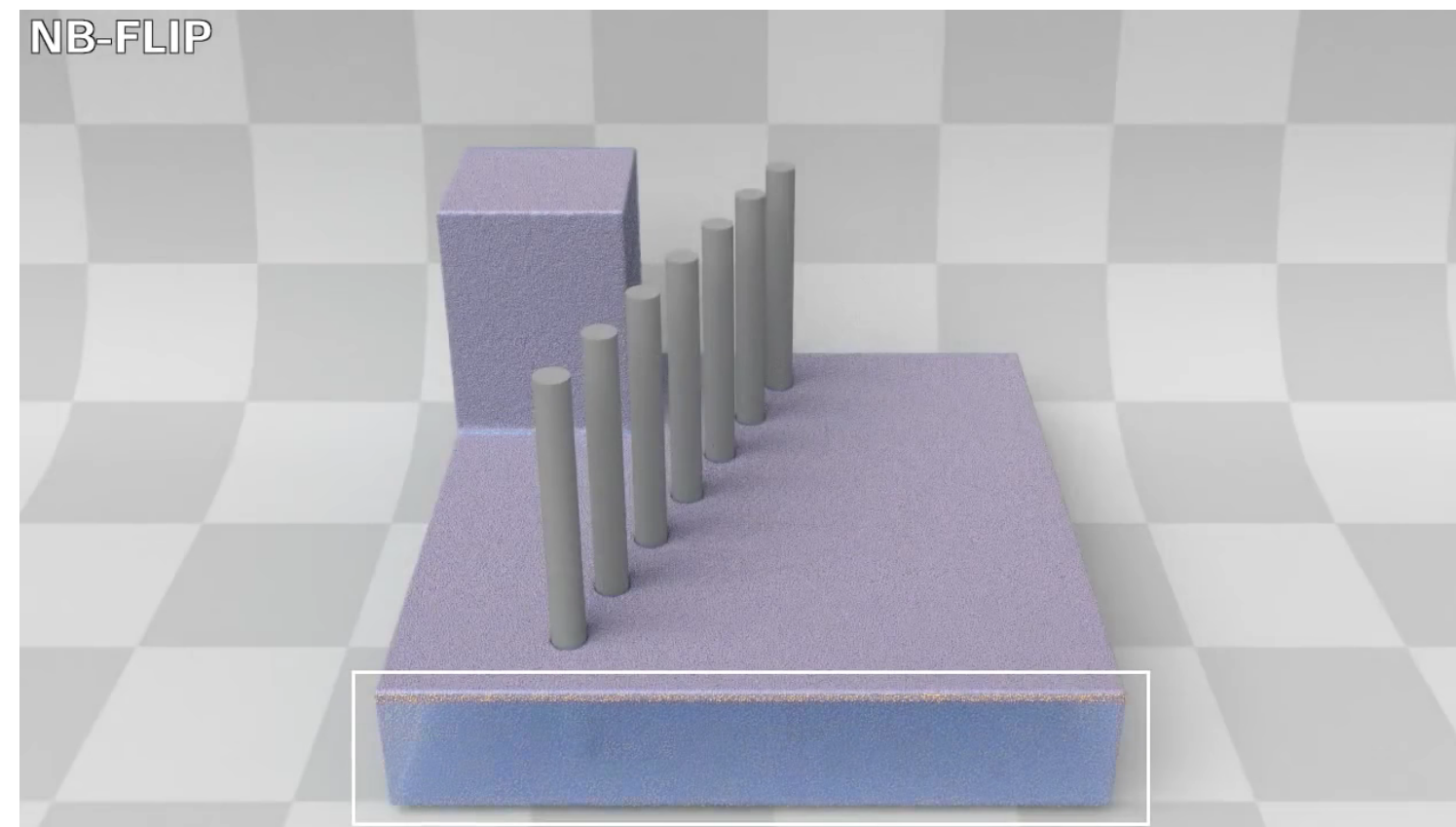
- Store scalar (pressure, density), in the center of the cell
- Store velocity components (u^x, u^y) on the cell edges

Improves accuracy and stability



PIC/FLIP Method

- Transfert particle velocity to the MAC grid (Store velocity u^k on grid)
- Evolve velocity on grid (pressure, forces, viscosity) excepted advection to u^{k+1}
- Add velocity difference $\Delta u = u^{k+1} - u^k$ to particles using interpolation (FLIP approach)
- Blend particle velocity with interpolated grid velocity (PIC/FLIP)
- Advect particles along their new velocity



[F. Ferstl et al., EUROGRAPHICS 2016]

Animating fluids

SPH

SPH - Smoothed Particle Hydrodynamics

Pure Lagrangian approach.

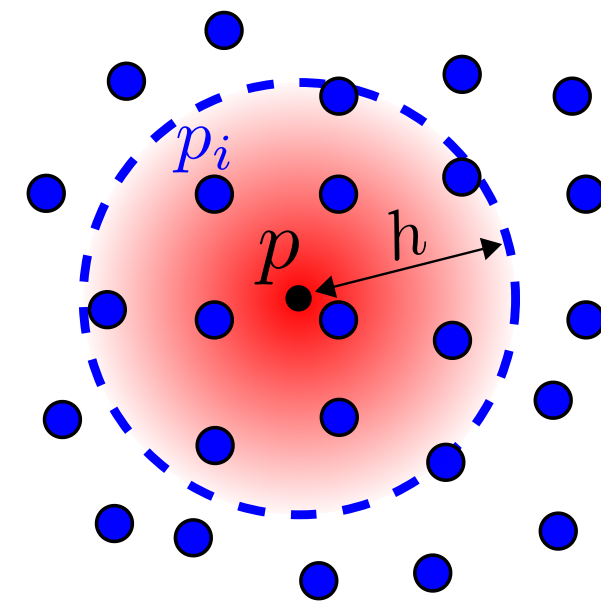
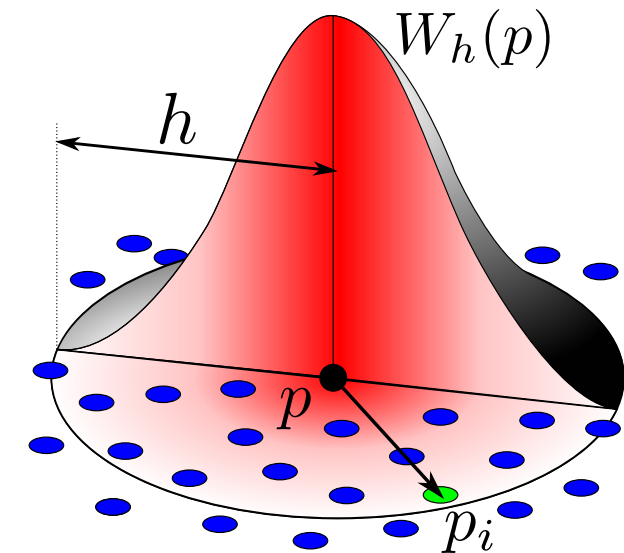
- Sample the fluid volume with particles
- Build a continuous field from local averaging around samples
Use some local weighting kernel W
- Express derivatives/Navier-Stokes on the continuous field

Advantages

- (+) Particle based - can interact with other models
- (+) Scalable

Initial proposed in Astronomy field

[L. Lucy, A numerical approach to the testing of the fission hypothesis. The Astronomical Journal, 1977.]



Sampling and density

How-to build a continuous field from arbitrary sampled particles ?

Consider arbitrary continuous field $A(p)$

Def. of convolution: $A(p) = (A \star \delta)(p) = \int_{\Omega} A(q) \delta(p - q) dq$

1. Consider W_h a smooth kernel with $\int_{\Omega} W_h(p) dp = 1$

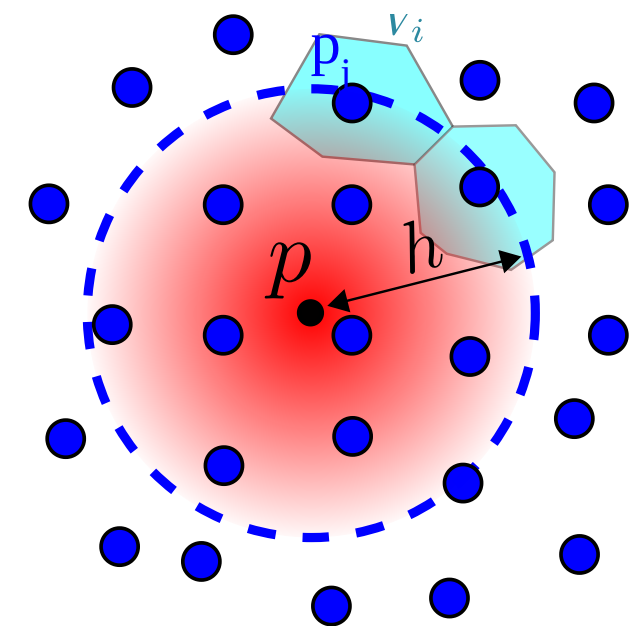
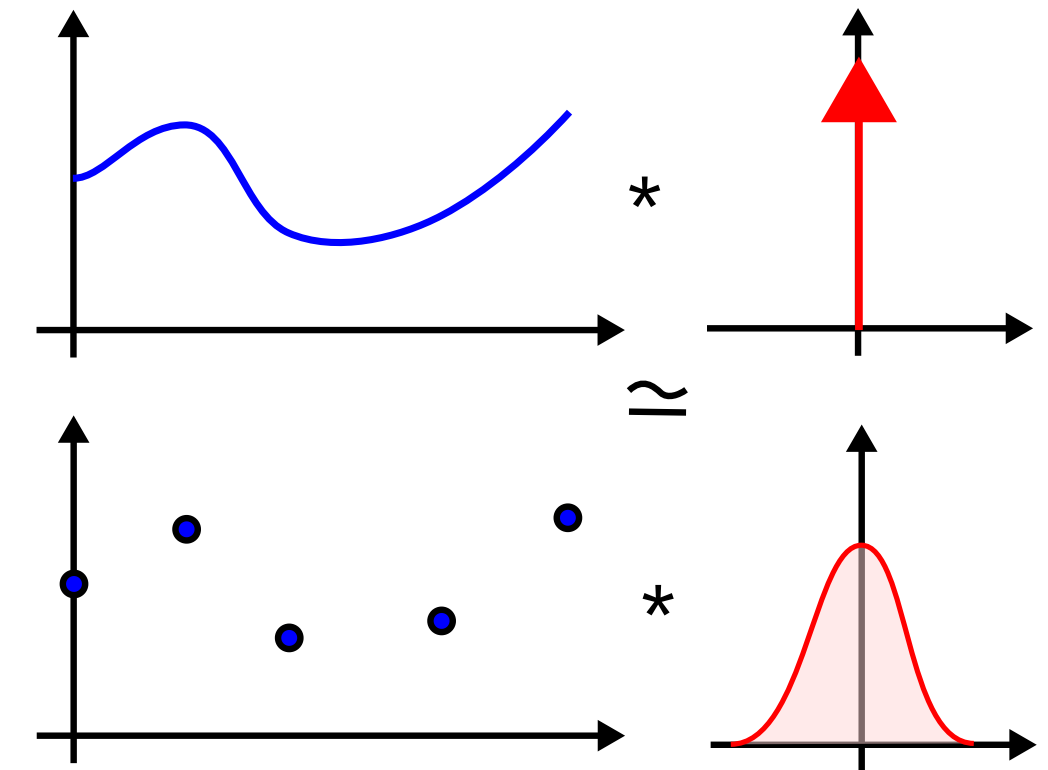
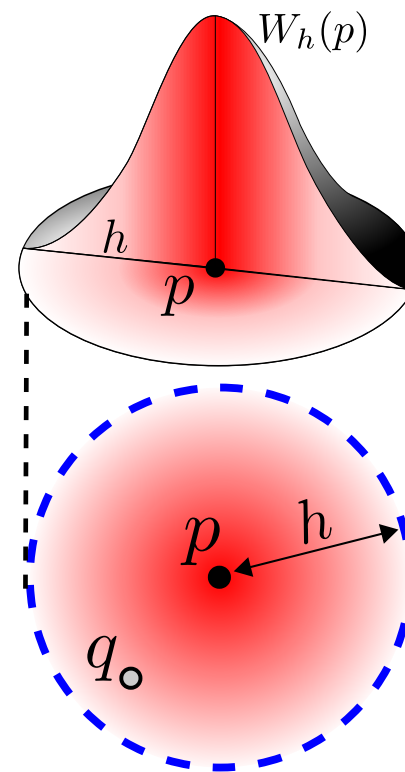
$$A(p) \simeq (A \star W_h)(p) = \int_{\Omega} A(q) W_h(p - q) dq$$

Low pass filter applied to A

2. Discrete sampling on p_j

$$A(p) = \sum_j A(p_j) W_h(p - p_j) V_j$$

V_j : small volume associated to p_j



SPH for Navier Stokes

[Desbrun and Cani, Smoothed Particles: A new paradigm for animating highly deformable bodies, EGCAS 1996]

[M. Muller et al., Particle-Based Fluid Simulation for Interactive Applications, SCA 2003]

[M. Ihmsen et al., SPH Fluids in Computer Graphics, EG STAR 2014]

Lagrangian representation on particle i

$$m_i \frac{dv_i}{dt} = \underbrace{m_i g}_{F_{weight}} - \underbrace{\frac{m_i}{\rho_i} \nabla p_i}_{F_{pressure}} + \underbrace{m_i \nu \Delta v_i}_{F_{viscosity}}$$

Objective:

1. Express $\rho_i, \nabla p_i, \Delta v_i$ using SPH formulation
2. Then integrate: ex. $v_i^{k+1} = v_i^k + \Delta t (F_{weight} + F_{pressure} + F_{viscosity}) / m_i$

Generic SPH representation:

Arbitrary field A at position p_i : $A(p_i) = \sum_j A(p_j) W_h(p_i - p_j) V_j$

For a particle of total mass m_i in the volume V_i : $\rho_i V_i = m_i \Rightarrow A(p_i) = \sum_j A(p_j) m_j / \rho_j W_h(p_i - p_j)$

Usually W_h are distance function :

$$A(p_i) = \sum_j A(p_j) m_j / \rho_j W_h(\|p_i - p_j\|)$$

Density

ρ_i : Replace $A(p)$ as ρ

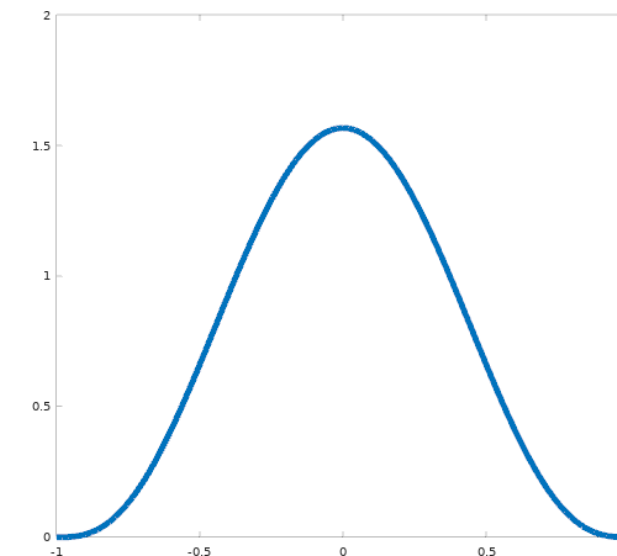
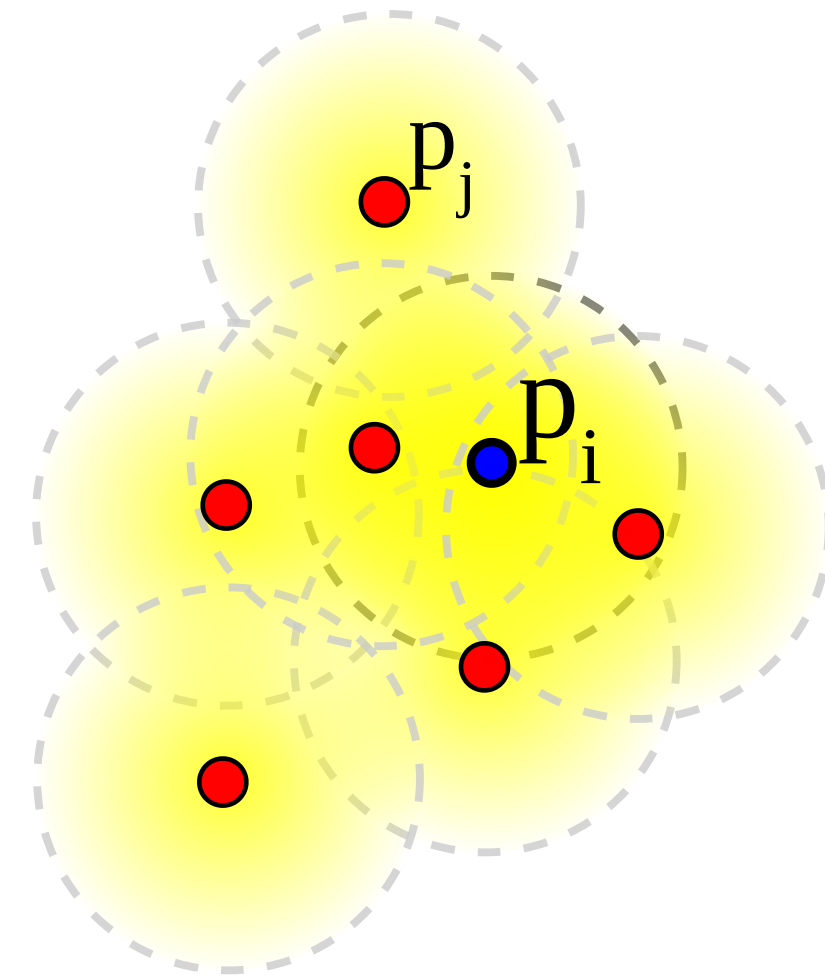
$$\rho(p_i) = \sum_j \rho(p_j) m_j / \rho_j W_h(\|p_i - p_j\|)$$

$$\Rightarrow \rho_i = \sum_j m_j W_h(\|p_i - p_j\|)$$

Choice of weight functions

Use a smooth polynomial:

$$\text{ex. } W_h^{\text{poly6}}(d) = \frac{315}{64 \pi h^9} (h^2 - d^2)^3 \quad 0 \leq d \leq h$$



Pressure

$$F_{pressure} = -\frac{m_i}{\rho_i} \nabla p_i$$

1. Use symmetric gradient b/w (i,j) $F_{pressure} = -\frac{m_i}{\rho_i} \nabla (p_i + p_j)/2$

$$F_{pressure} = -\frac{m_i}{\rho_i} \sum_j m_j \frac{p_i + p_j}{2 \rho_j} \nabla W_h(\|p_i - p_j\|)$$

2. Express the pressure as a function of the density ρ

Simple approximation: $p_i = s (\rho_i - \rho_0)$

- s : Stiffness property

- ρ_0 : Rest density of the fluid

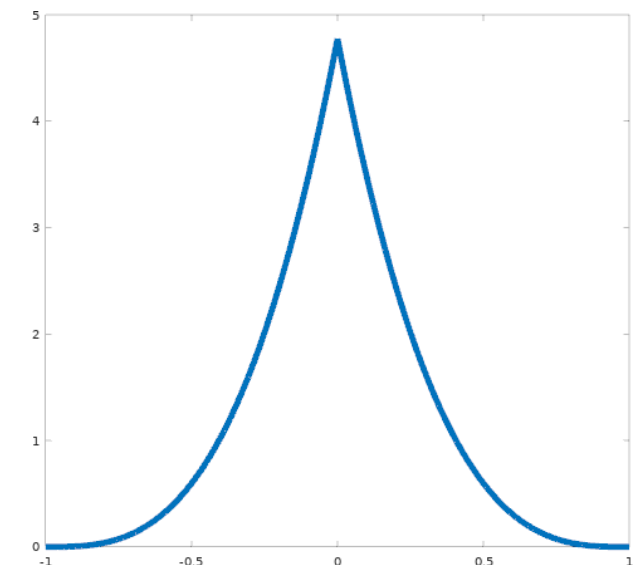
3. Weight function

Pressure is used to avoid particles to group together

Avoid local maxima $\Rightarrow$ non smooth "spiky" function at 0

$$W_h^{spiky}(d) = \frac{15}{\pi h^6} (h - d)^3 \quad 0 \leq d \leq h$$

$$\nabla W_h^{spiky}(p_i - p_j) = -\frac{45}{\pi h^6} (h - \|p_i - p_j\|)^2 \frac{p_i - p_j}{\|p_i - p_j\|} \quad 0 \leq \|p_i - p_j\| \leq h$$



Viscosity

$$F_{viscosity} = m_i \nu \Delta v_i$$

1. Use symmetric laplacian b/w (i,j)

$$F_{viscosity} = m_i \nu \Delta(\mathbf{v}_j - \mathbf{v}_i) \quad - \text{ viscosity depends on velocity differences}$$

$$F_{viscosity} = m_i \nu \sum_j m_j \frac{(\mathbf{v}_j - \mathbf{v}_i)}{\rho_j} \Delta W_h(\|p_i - p_j\|)$$

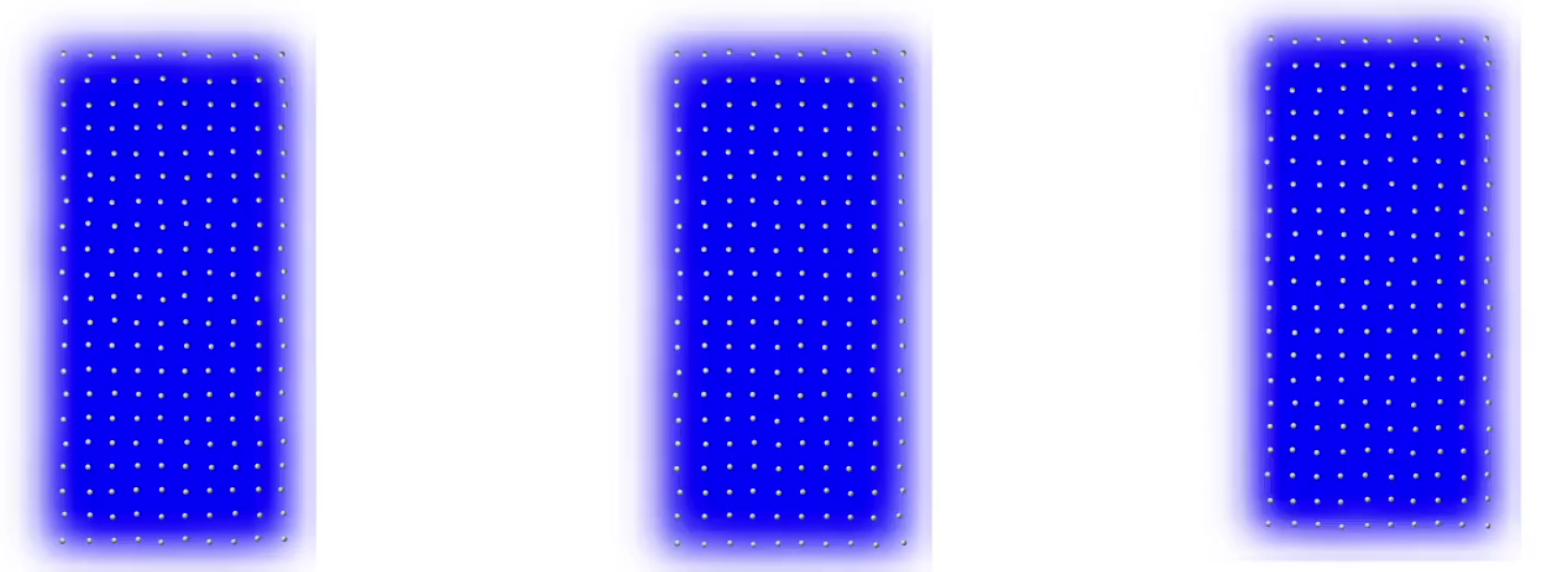
2. Weight function

Second derivative should remain positive

Can use the spiky kernel

$$W_h^{spiky}(d) = \frac{15}{2\pi h^6} (h - d)^3 \quad 0 \leq d \leq h$$

$$\Delta W_h^{spiky}(d) = \frac{45}{\pi h^6} (h - d) \quad 0 \leq d \leq h$$



Increasing viscosity ν

SPH Summary

Set initial conditions v_i

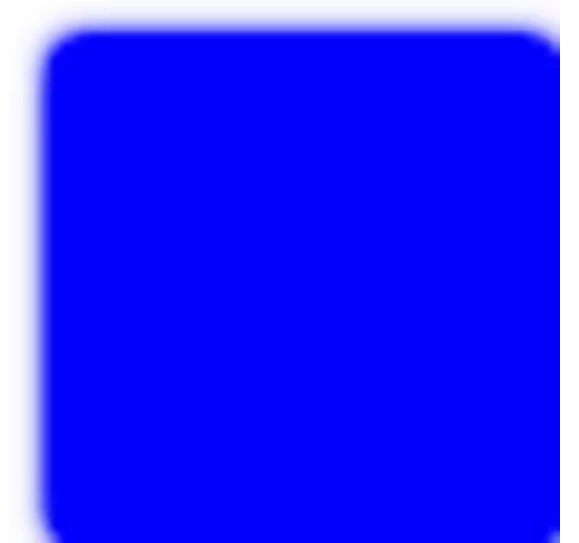
Compute values

- Density: $\rho_i = \sum_j m_j W_h^{poly6}(\|p_i - p_j\|)$
- Pressure: $p_i = s(\rho_i - \rho_0)$

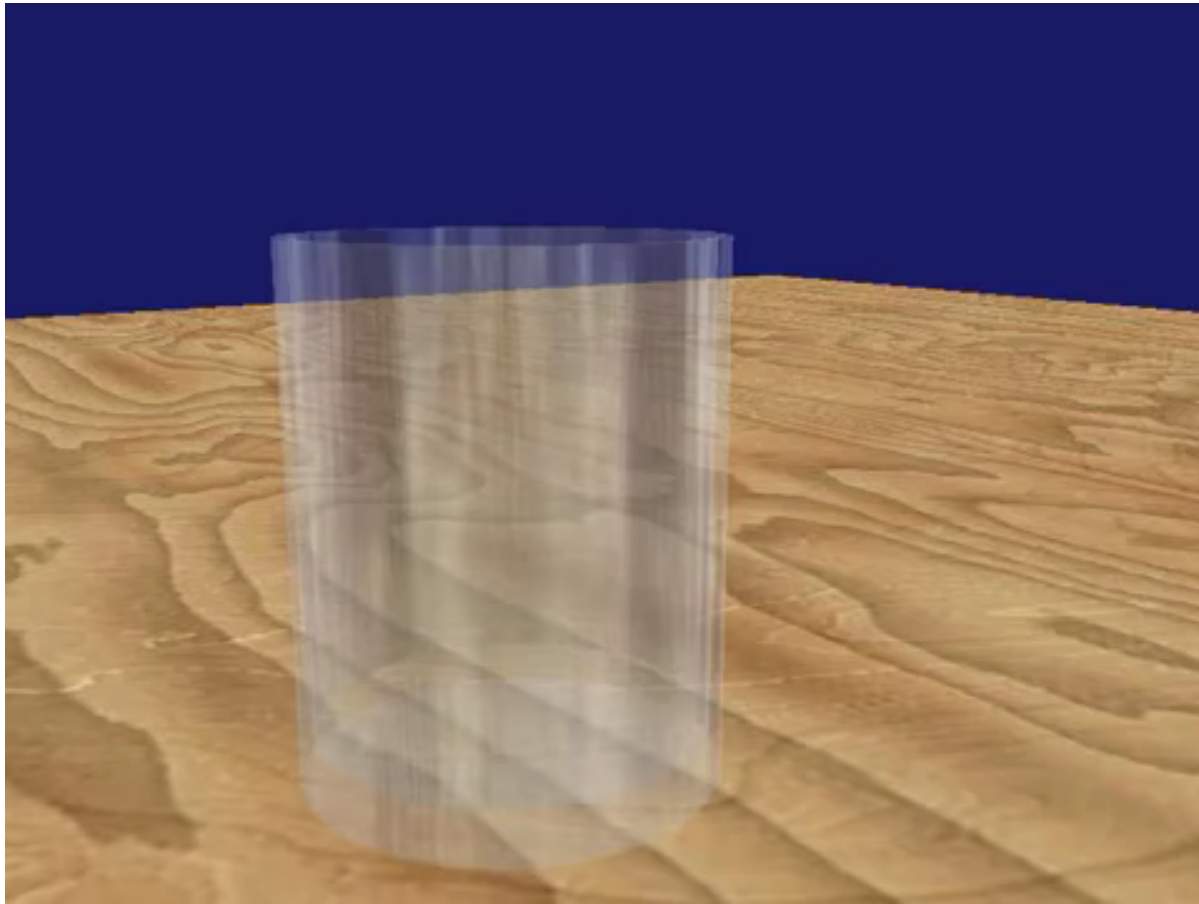
Compute forces

- $F_{weight} = m_i g$
- $F_{pressure} = -\frac{m_i}{\rho_i} \sum_j^{j \neq i} m_j \frac{p_i + p_j}{2 \rho_j} \nabla W_h^{spiky}(\|p_i - p_j\|)$
- $F_{viscosity} = m_i \nu \sum_j^{j \neq i} m_j \frac{(v_j - v_i)}{\rho_j} \Delta W_h^{spiky}(\|p_i - p_j\|)$

Time integration: $v_i^{k+1} = v_i^k + \Delta t (F_{weight} + F_{pressure} + F_{viscosity}) / m_i$



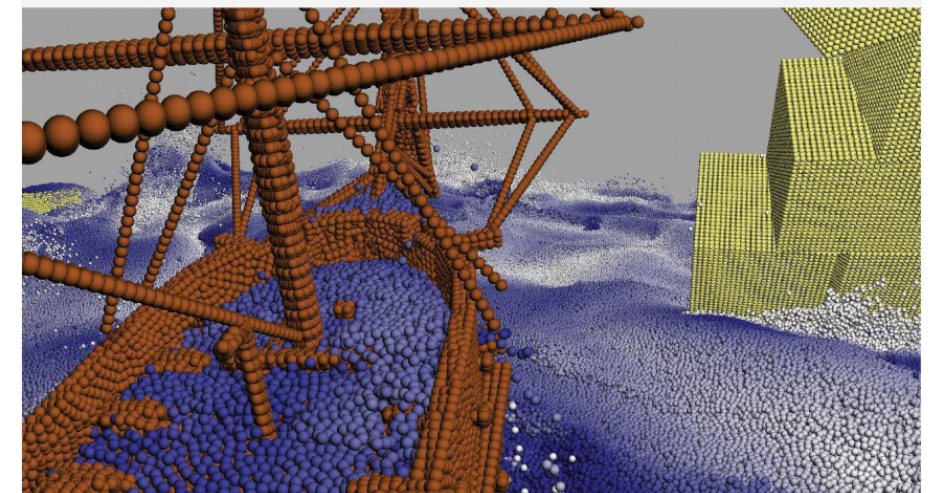
SPH examples



Muller 2003



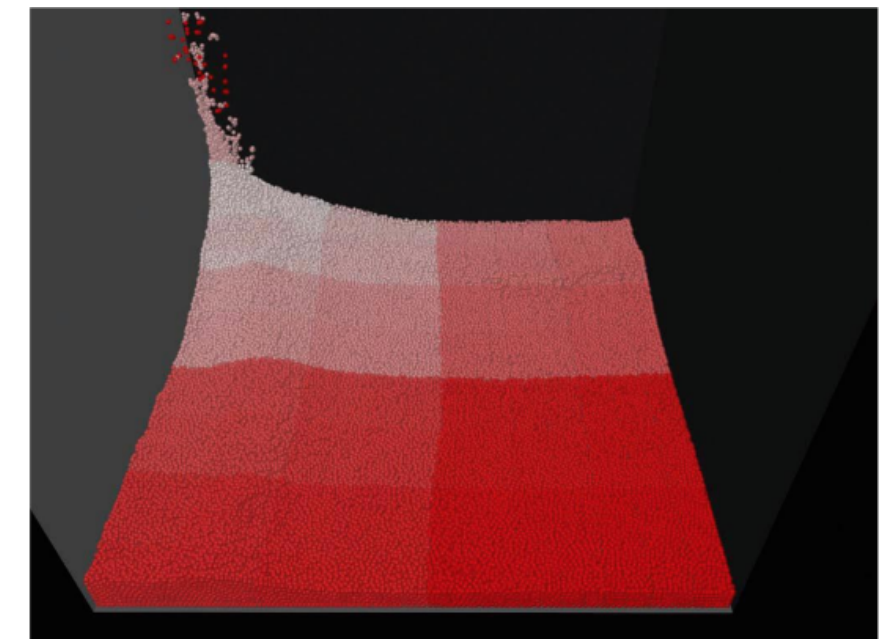
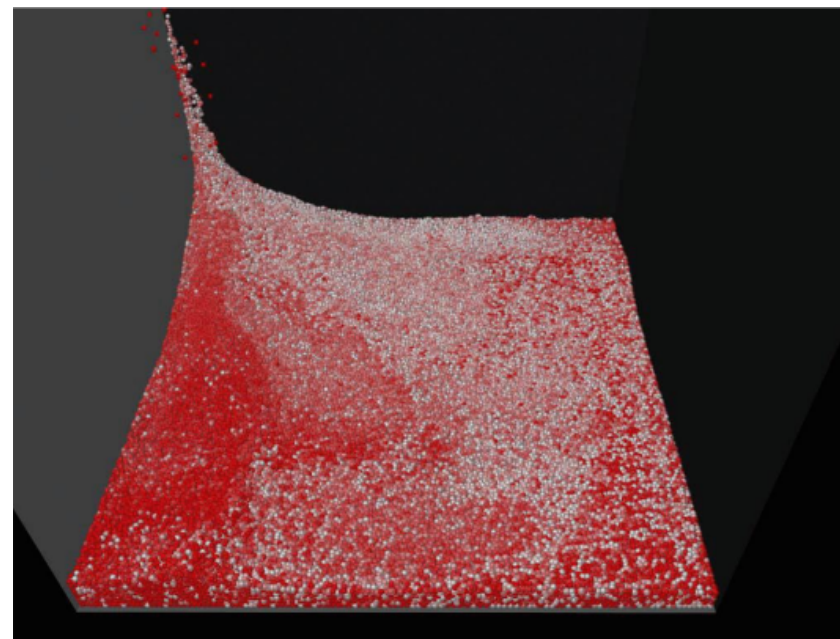
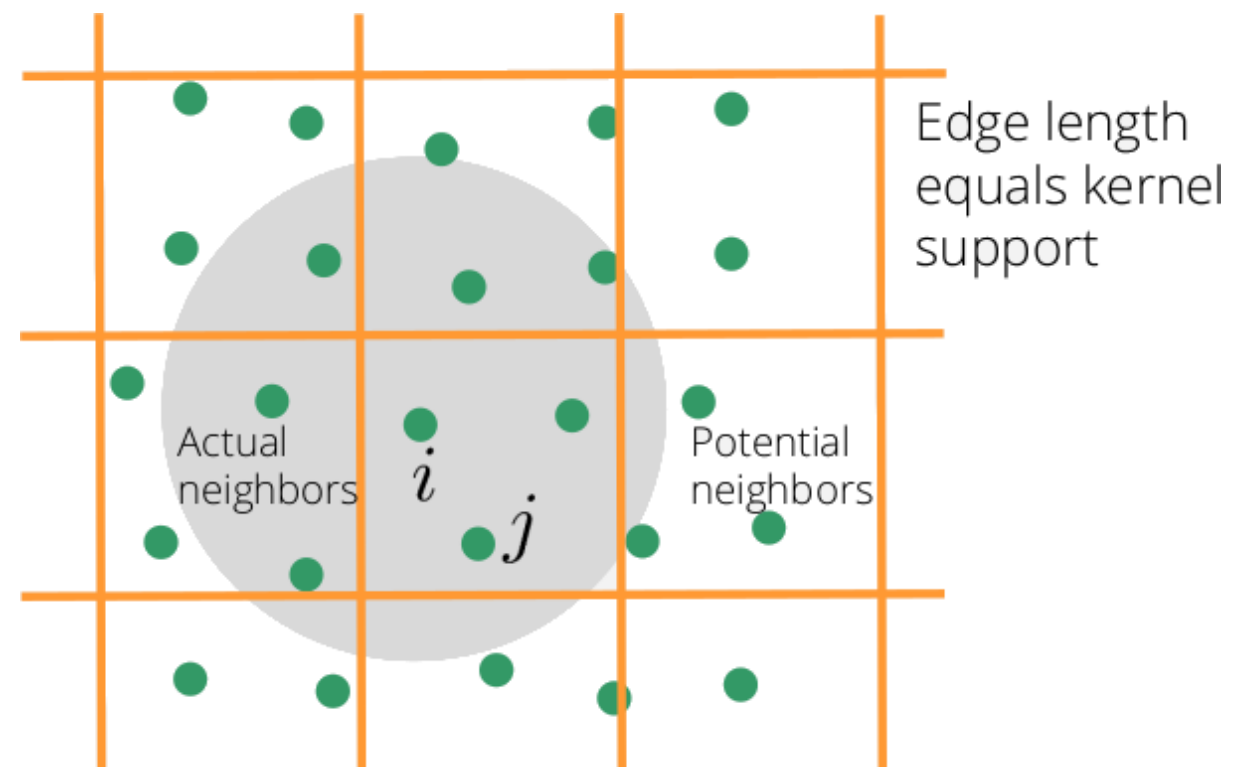
M. Teschner 2012 - 20M particles



Acceleration structure

SPH based on pair-wise interaction $\Rightarrow$ spatial sorting acceleration structure

- Uniform grid: simple and efficient.
- Verlet lists (wider neighborhood, updated every n steps only)
- List of vertices per cell, hash table for cell storage
- Spatial sorting for cache efficiency



M. Teschner

SPH extensions

(+) Very versatile (interaction between any deforming shapes)

Not only fluids

(-) Not well understood accuracy

(-) Compressible

[Solenthaler et al., Predictive-Corrective Incompressible SPH, ACM SIGGRAPH 2009]

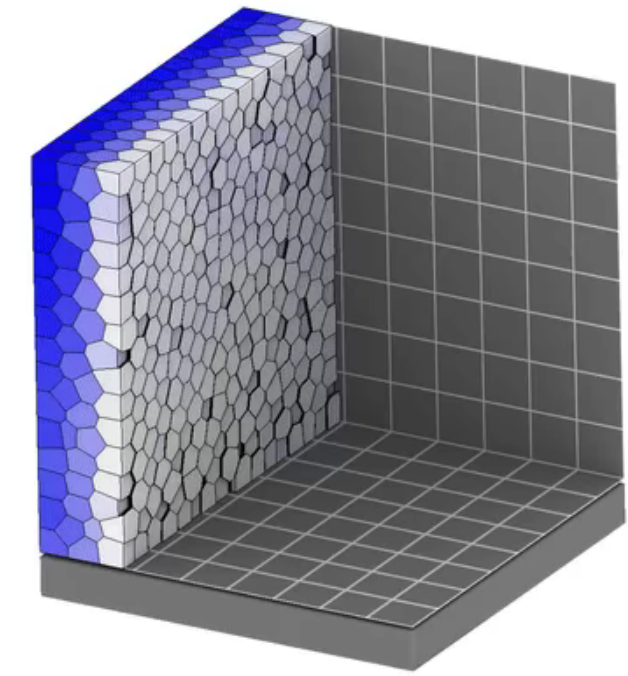
[Ihmsen et al, Implicit Incompressible SPH, IEEE TVCG 2013]

(-) Limited time step

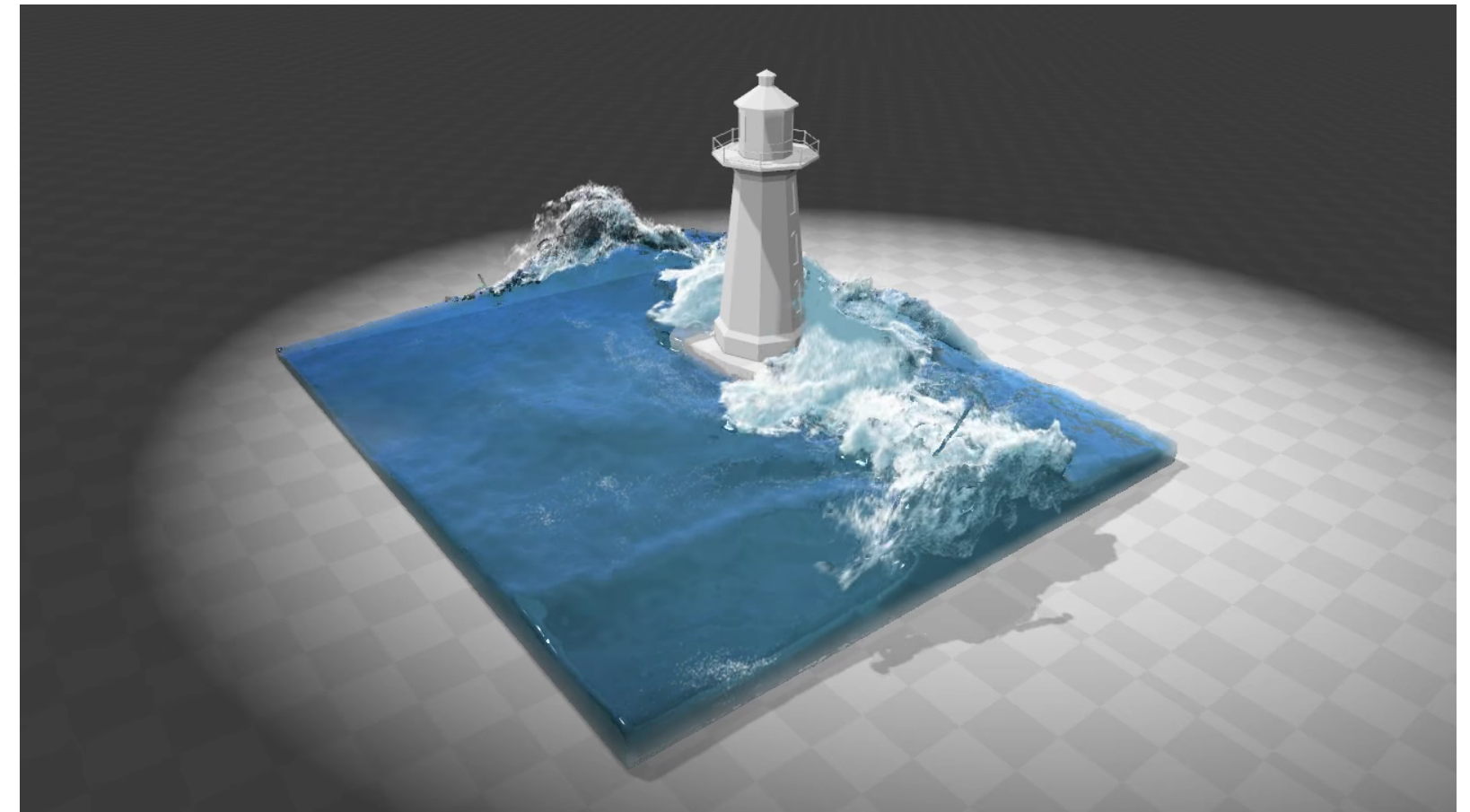
[Macklin and Muller, Position based Fluids, ACM SIGGRAPH 2013]

(-) Boundaries are hard to handle

[Brand et al., Pressure Boundaries for Implicit Incompressible SPH, ACM TOG 2018]



Bruno Levy



[Macklin and Muller 2013] , [Yu and Turk 2009]