Physically-based simulation

Deformable models

Material model

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

-Purely elastic models don't loose energy when deformed (potential ↔ kinetic)

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

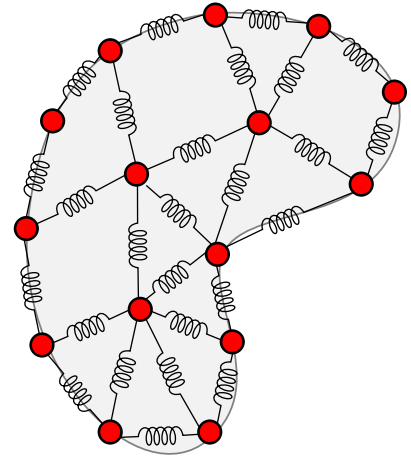
- Ductile material - can allow large amount of plastic deformation without breaking (plastic)
- Brittle - Opposite (glass, ceramics)

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

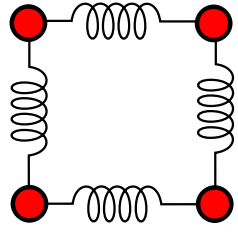


Spring mass systems

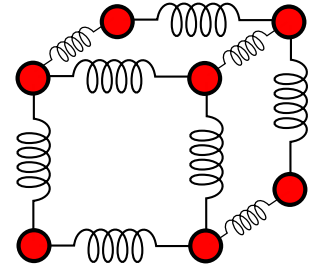
- Particles (position, speed, mass): samples on shape
- Springs : link closed-by particles in the reference shape

1D curve structure

2D surface structure



3D volume structure

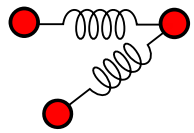
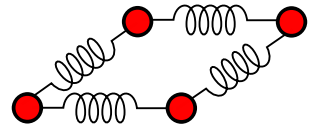
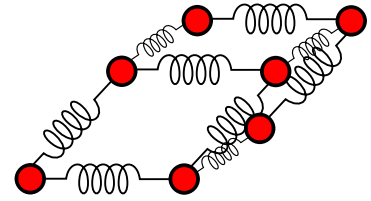


Spring structure

How to model spring connectivity ?

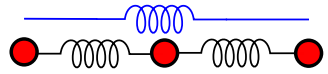
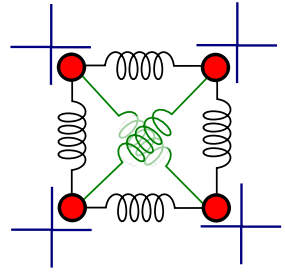
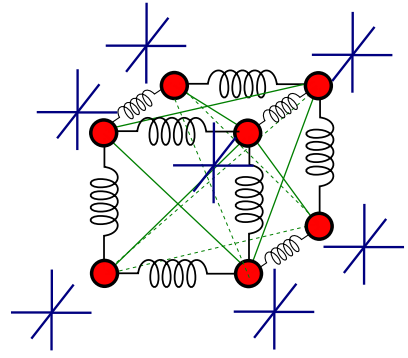
- **Structural springs**: 1-ring neighbors springs ($\simeq$ mesh edges)

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

$\Rightarrow$ Add extra springs connectivity

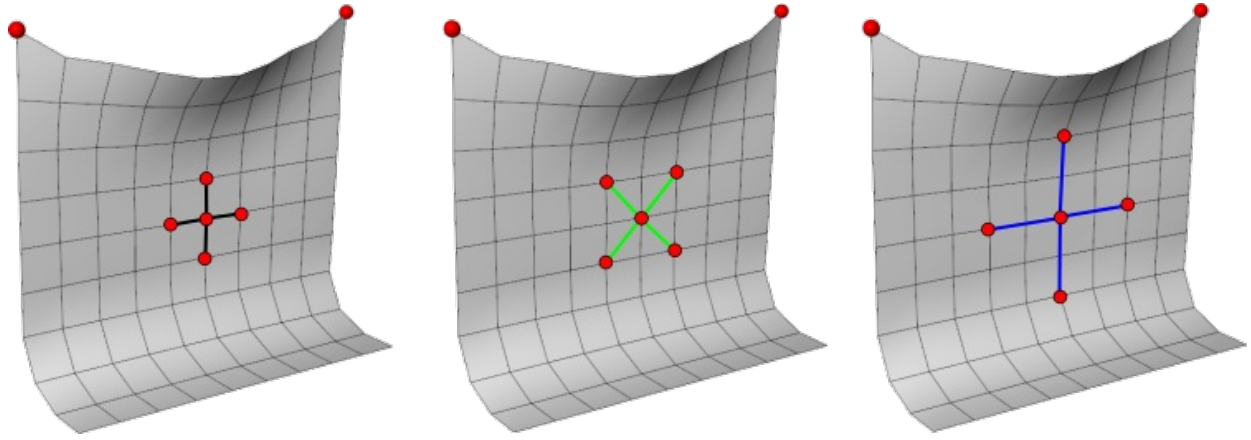
- **Shearing springs**: Diagonal links
- **Bending springs**: 2-ring neighborhood

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.



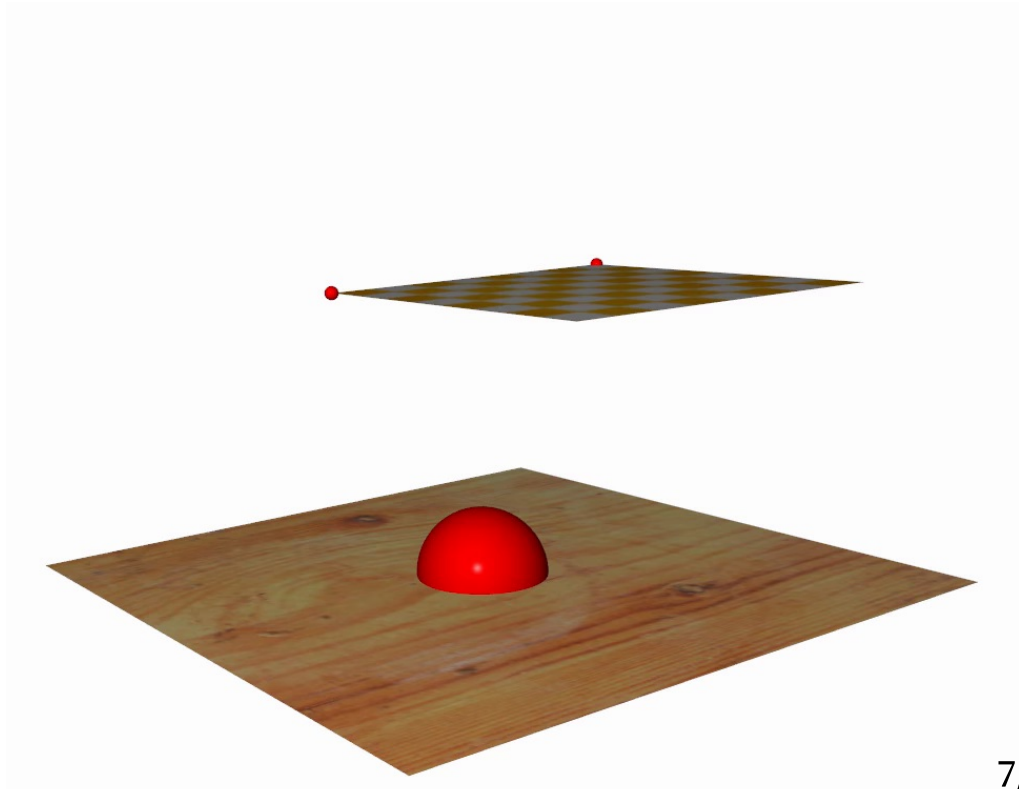
Forces

- On each particle: gravity + drag + spring forces

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

  - $\mathcal{V}_i$: neighborhood of particle i
  - L^0_{ij}: rest length of spring ij

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



Note on Mass-Spring numerical solution

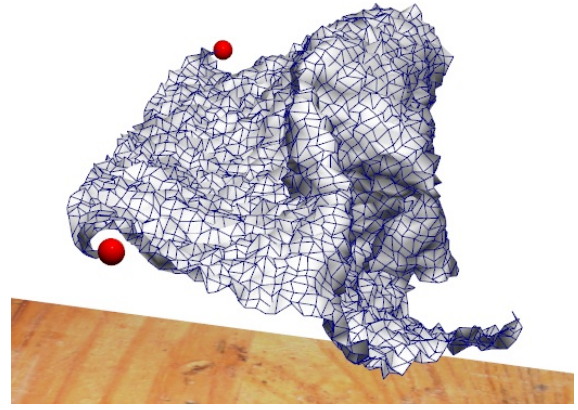
- Non-linear ODE

- Large K_{ij} : good length preservation, but stiff ODE
  $\Rightarrow$ 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



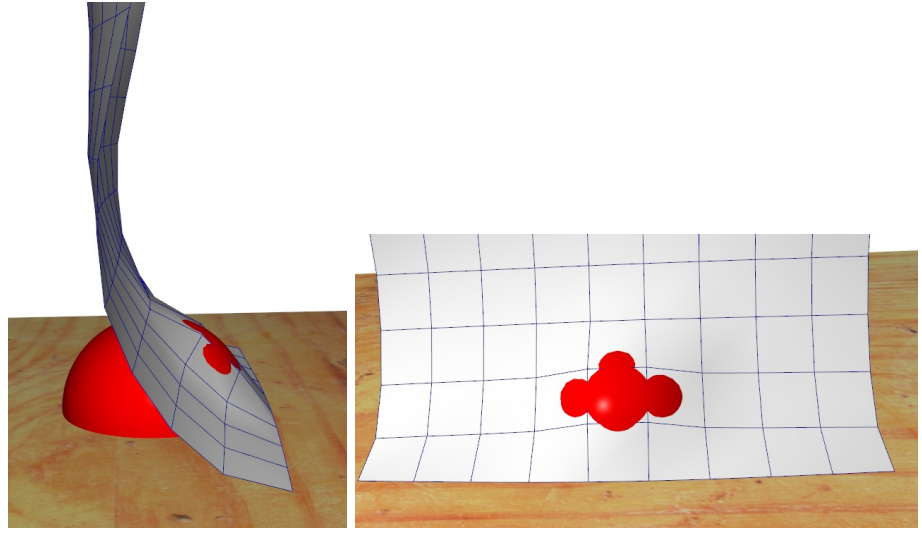
Collisions

- Simple approach : Handled as collision between particles and shapes

  (+) Simple and efficient

  (-) Collision may still appears within a triangle

  $\Rightarrow$ Exaustive approach: point-face + edge-edge



Limitation of mass spring model and continuous model

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

  $\Rightarrow$ No :(

  Depends on the connectivity $\rightarrow$ 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]

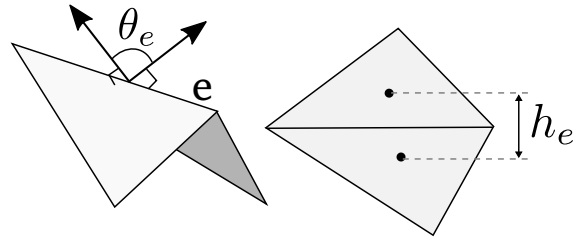
Triangle as continuous elements

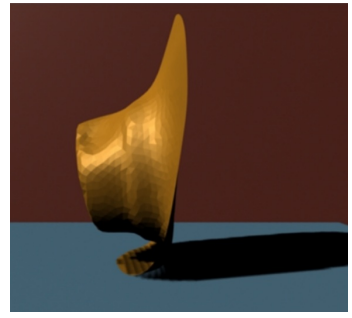
- Defining Bending Energy between triangles

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

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

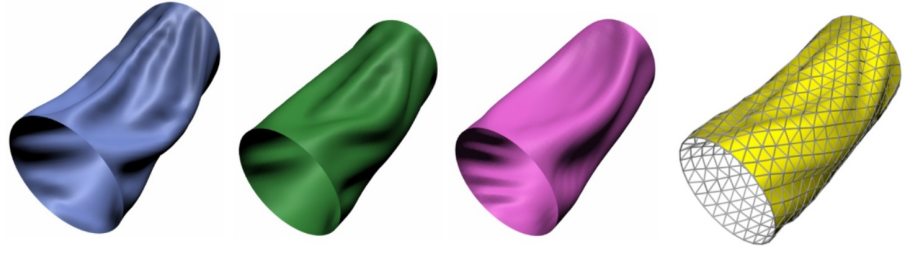
    (or expressed using forces in [R. Bridson et al., SCA 2003])





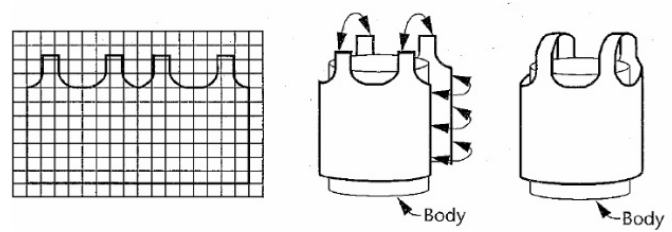
- Going toward full FEM numerical resolution

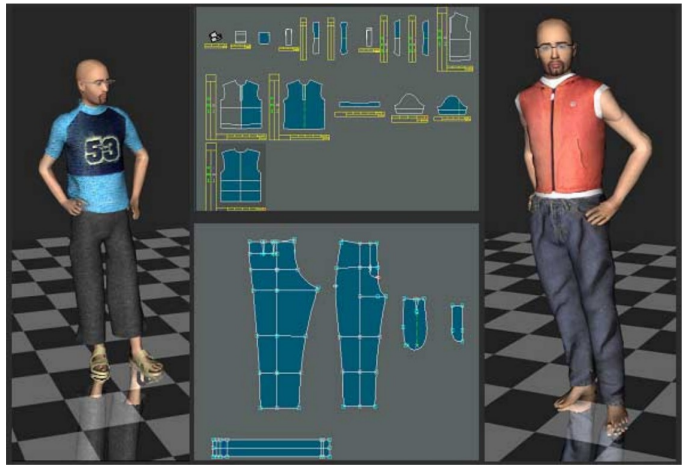
  - B. Thomaszewski et al. [SCA 2006], [VRIPHYS 2008], [EG 2009].



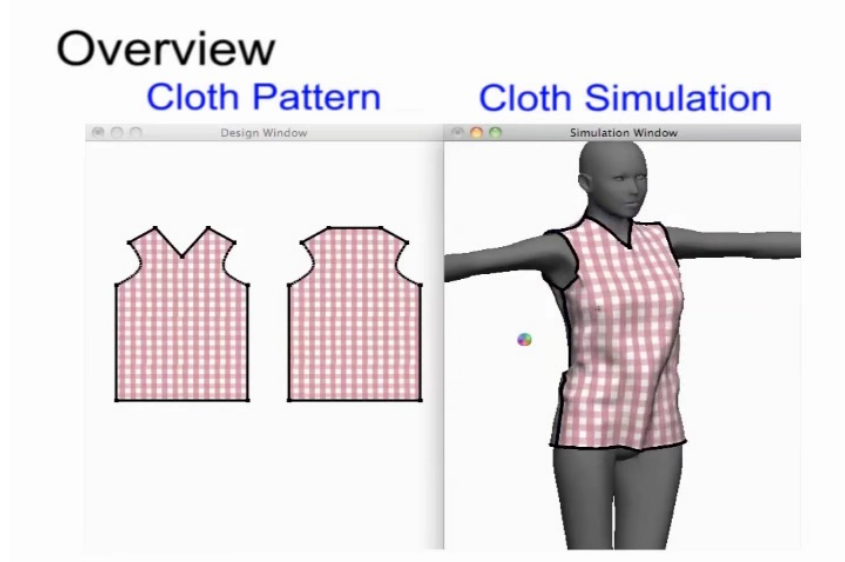
Clothing

- Stich 2D patterns together to generate full cloth
  - Cloths are developable material (preserve isometry to their 2D patterns)





[Thalman et al. 2002]



[Umetani et al., 2011]

Advanced collision

Self collision

- Handled as moving point in collision with moving triangle

- Given a triangle $P_1(t)P_2(t)P_3(t)$ and a point $P(t)$, with

$$P_k(t) = P_k(t_0) + v_{P_k}(t)$$

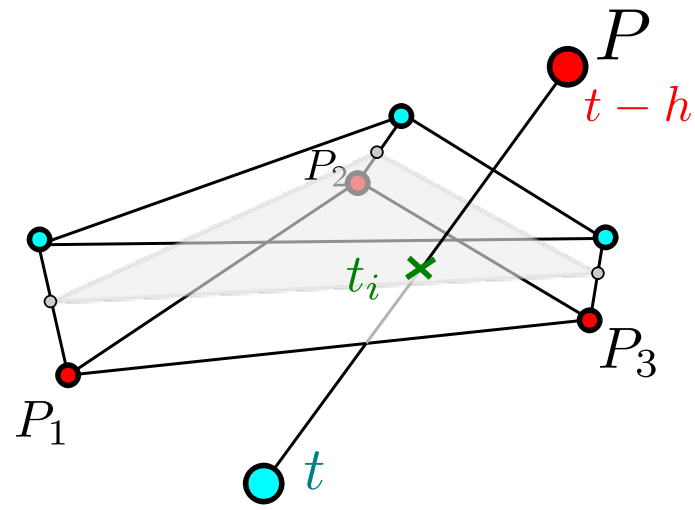
- $P(t_i)$ in collision with triangle $P_1(t_i)P_2(t_i)P_3(t_i)$ at time t_i if

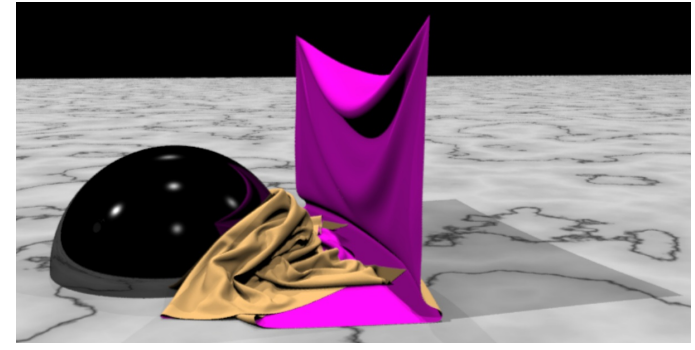
$$\exists(\alpha,\beta,\gamma) \in [0,1]^3,\ P(t_i) = \alpha\, P_1(t_i) + \beta\, P_2(t_i) + \gamma\, P_3(t_i)$$

$$\alpha + \beta + \gamma = 1$$

$$(P(t_i) - P_1(t_i)) \times n(t_i) = 0,\ n(t_i) \text{ normal to the triangle}$$

(necessary condition to help computation)





[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]

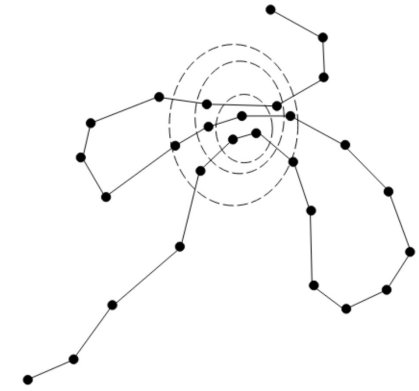
Advanced collision

Multiple collisions

- *Solving each collisions separately may lead to new ones without converging to collisions free state*

⇒ Rigid Impact Zone - Incremental approach

1. Detected collisions are grouped by proximity (same face, edge, etc)
2. Each group is handled as collision between rigid body
3. Treated groups are merged together.
4. Iterate at step 1 with larger groups



[Provot et al., 97]

- Recently handled on the GPU

  [Tang et al., I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation, ACM SIGGRAPH Asia 2018]

- Robust generic approach using tetrahedrization of air volume

  [Muller et al., Air Meshes for Robust Collision Handling, ACM SIGGRAPH 2015]

Untangling cloths

Specific problem of collision when no previous state is known.

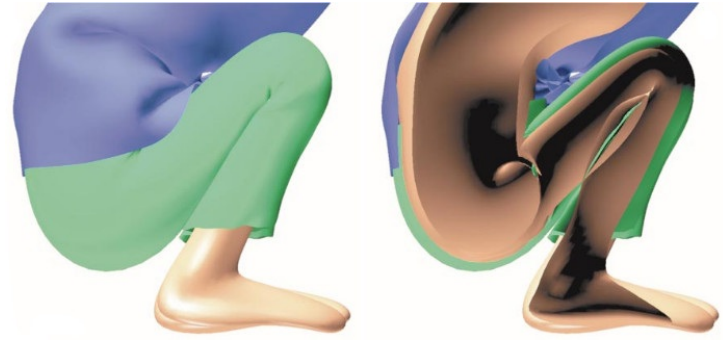
How to solve collision ?

- Geometrical displacement + penalty based forces

  ex. Optimize intersection contour minimization

  [Baraf et al. 2002], [Volino et al 2006]

- Implicit surfaces



Wrinkles in cloth modeling

Obtaining cloths wrinkles requires high resolution simulations

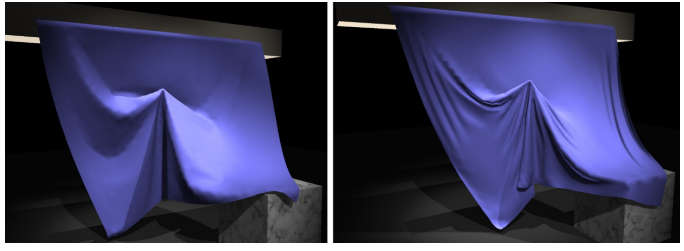
(-) Computationally costly, difficult to parameterized, lack of control

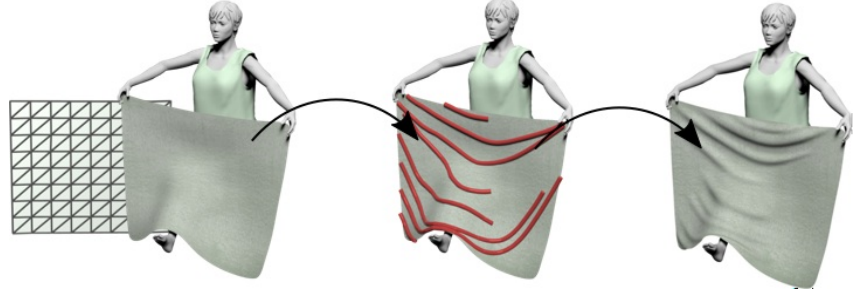
Idea: Separate low resolution cloth simulation from local wrinkles appearance.

- Cloth is a developable material (cannot compress/stretch w/r 2D pattern)
- Detect compression on numerical surface $\rightarrow$ add wrinkles

Model wrinkles using

- Texture deformation [Hadap et al., IEEE Proc. on Viz. 1999]
- Subdivision mesh with PBD length constraints [Muller and Chentanez, SCA 2010]
- Example data and ML [Wang et al. ACM SIGGRAPH 2010]
- Strain to generate geometrical wrinkles [Rohmer et al., ACM SIGGRAPH Asia 2011]



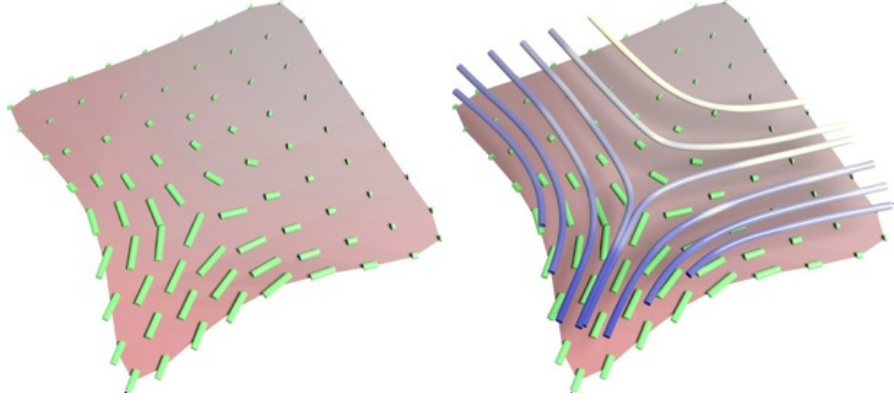


Geometrical wrinkles from strain

Compute strain on low resolution mesh

$$\mathbf{F} = [e_1' - e_1 \;\; e_2' - e_2], \quad \sigma = \mathbf{F}\,\mathbf{F}^T, \quad (\lambda_1, \lambda_2) = \text{Spectrum}(\sigma)$$

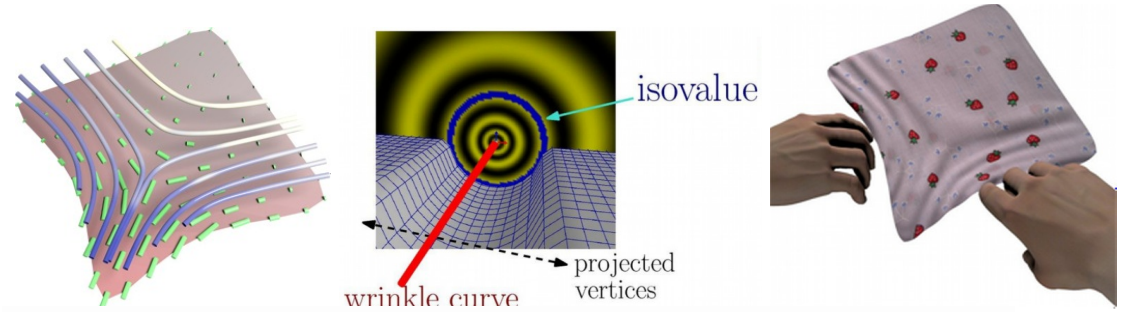
- Analyse main direction of compression $(0 < \min(\lambda_1, \lambda_2) < 1)$ as a vector field
- Generate stream lines $\rightarrow$ *wrinkle curves*

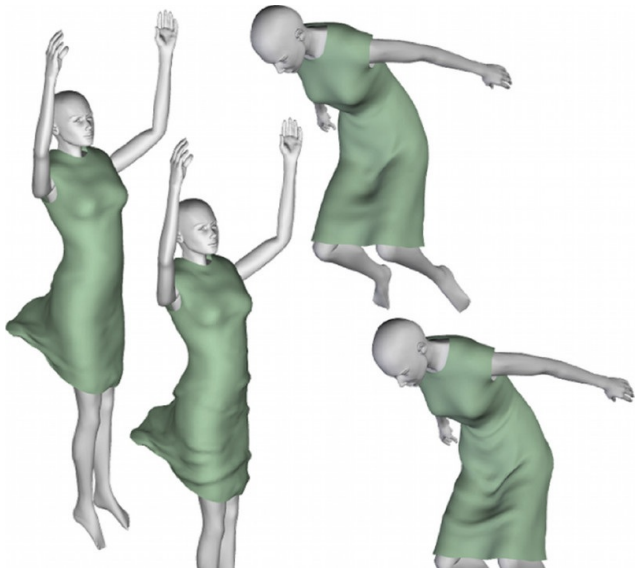


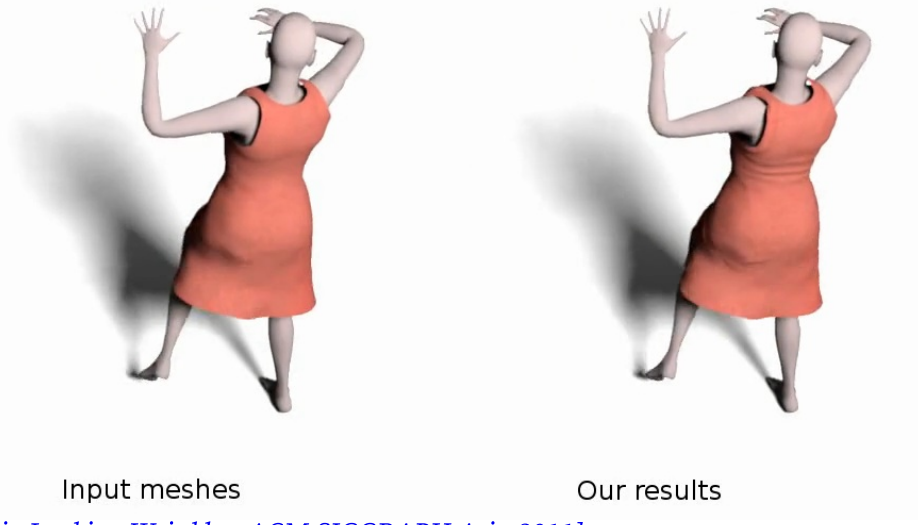
Geometrical wrinkles from strain

Create geometrical deformation using implicit surfaces

Allow smooth blending between wrinkles







[Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles, ACM SIGGRAPH Asia 2011]

Animating fluids

Fluid in grid: Stable Fluids

Consider a rectangular grid filled with fluid

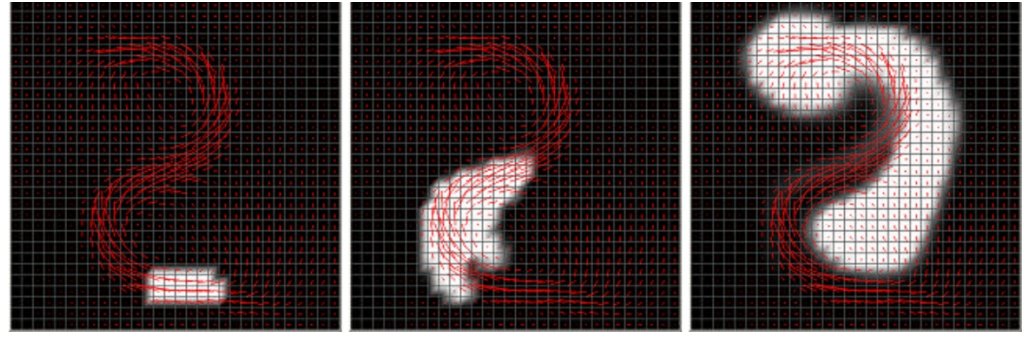
Direct approach: Solve Navier-Stokes equation using finite differences on the grid.

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho}\nabla \mathrm{p} + f - u \cdot \nabla u + \nu \triangle u$$

(-) Stability hard to achieve, loose advection details on the grid. (ex. [Foster and Metaxas, 97])

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

- First idea: get rid of explicit pression expression
  - Pression p models incompressible behavior.
  $\rightarrow$ Project velocity u into divergence free field at each step instead of tracking pression values.



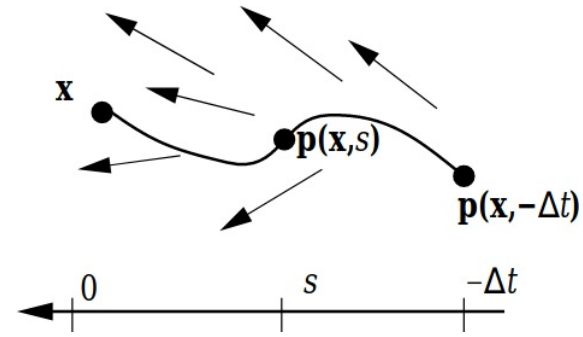
$$\Rightarrow \frac{\partial u}{\partial t} = \mathrm{P}(f - u \cdot \nabla u + \nu \triangle u)$$

P projection operator such that $\mathrm{div}(u) = 0$

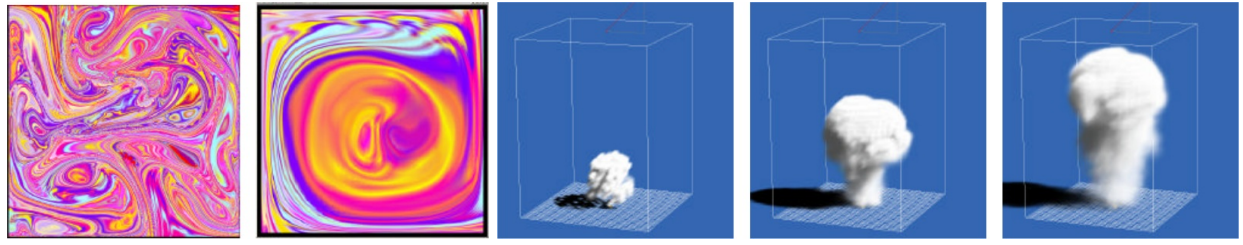
Stable fluids method

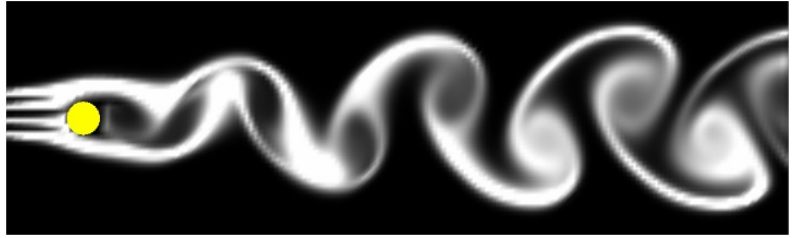
General approach: split each component

$$u^k \underbrace{\rightarrow}_{\text{add forces}} u_1^k \underbrace{\rightarrow}_{\text{advect}} u_2^k \underbrace{\rightarrow}_{\text{diffuse}} u_3^k \underbrace{\rightarrow}_{\text{project}} u^{k+1}$$



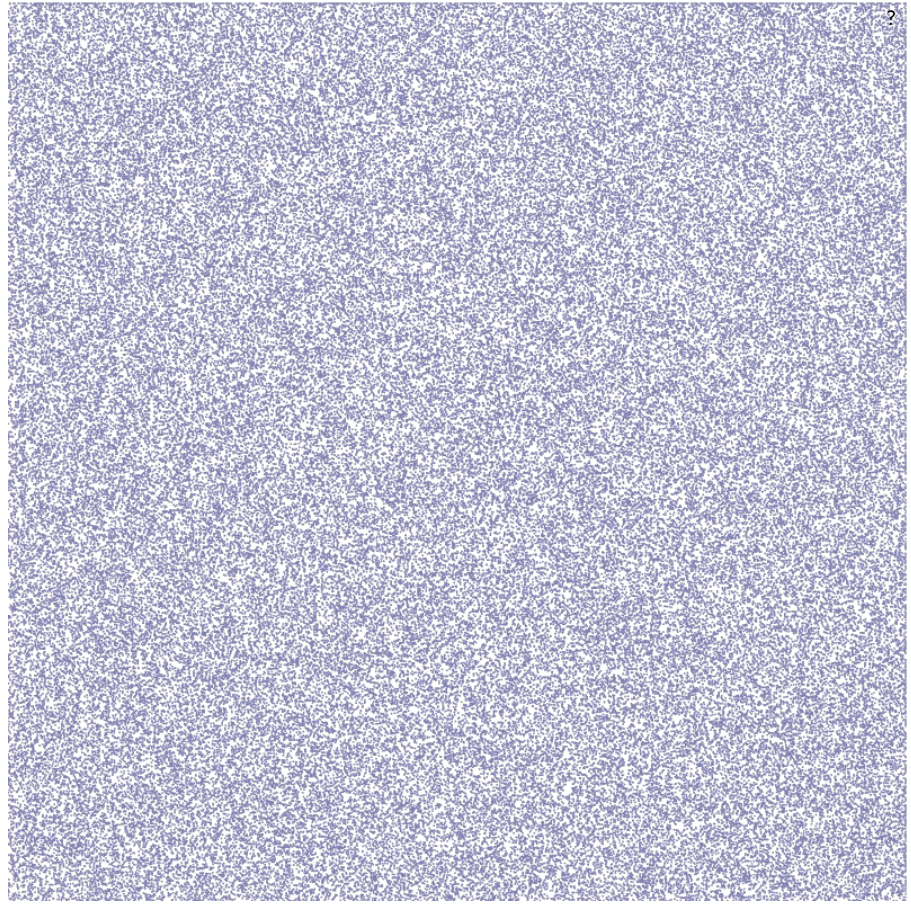
- Advection: Method of characteristics (use explicit position, trace back their trajectory in time)
- Diffusion: Use implicit finite differences
- Project: Poisson solver





Jos Stam, Real-Timer Fluid Dynamics for Games, 2003

Stable fluids example



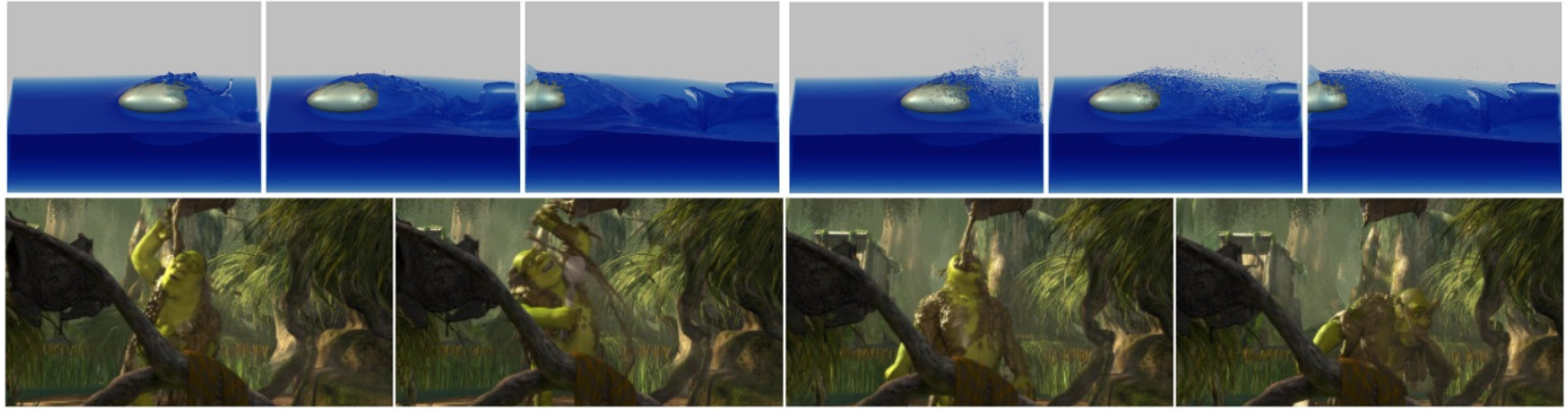
Level Set methods introduction

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 $\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0$ (advection)

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

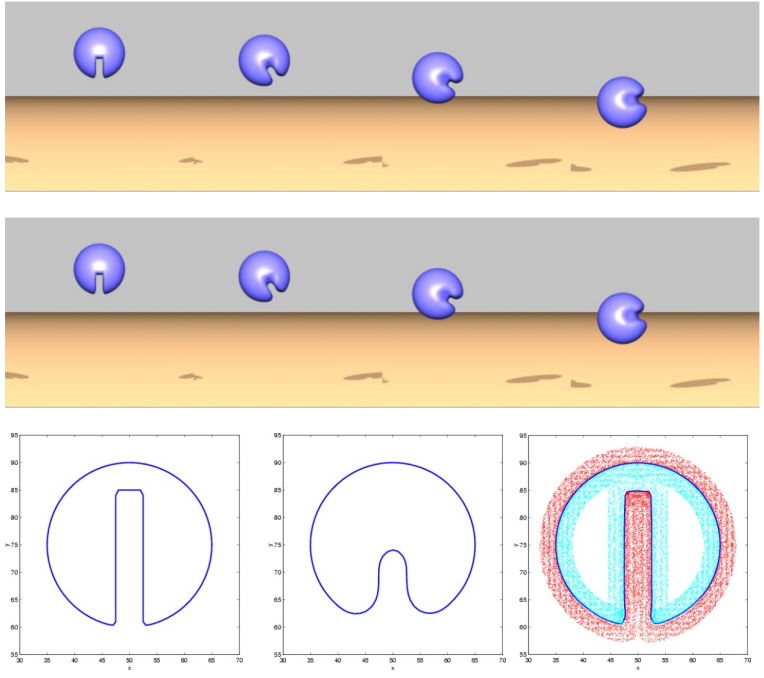


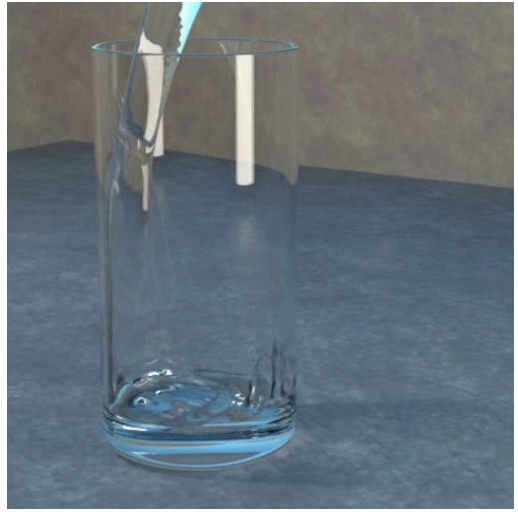
Particles Level Set

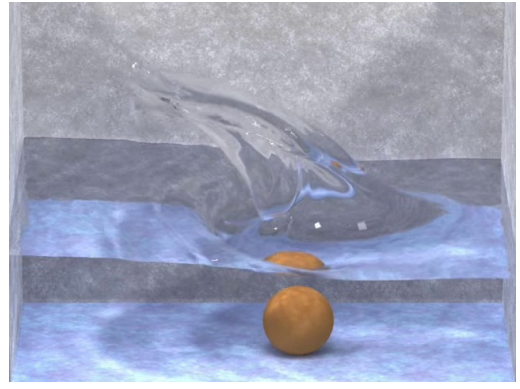
Limitation of level sets: smoothing and loss of volume from grid interpolation

⇒ 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. *(8 min / frames)*

Free surface water

Specific lightweight models for free surface of ocean-like 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]

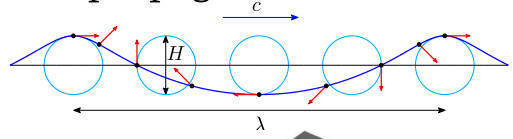
[D. Hinsiger et al. Interactive Animation of Ocean Waves. SCA 2002]

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

ex. *Trochoid models*

Particles following circular trajectories, waves propagates

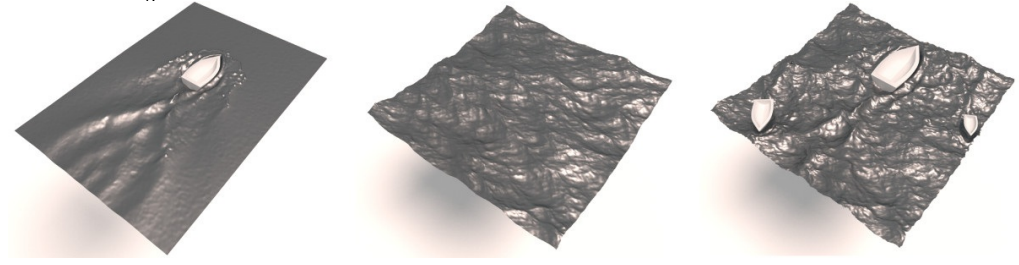
$$x(u,v,t) = u + e^{k\,v}\ \sin(k(u+c\,t))/k$$
$$x(u,v,t) = v - e^{k\,v}\ \cos(k(u+c\,t))/k$$
$$k = 2\pi\,\lambda$$



Used for procedural ocean modeling

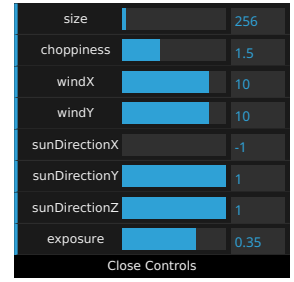
(+) Simple and scalable

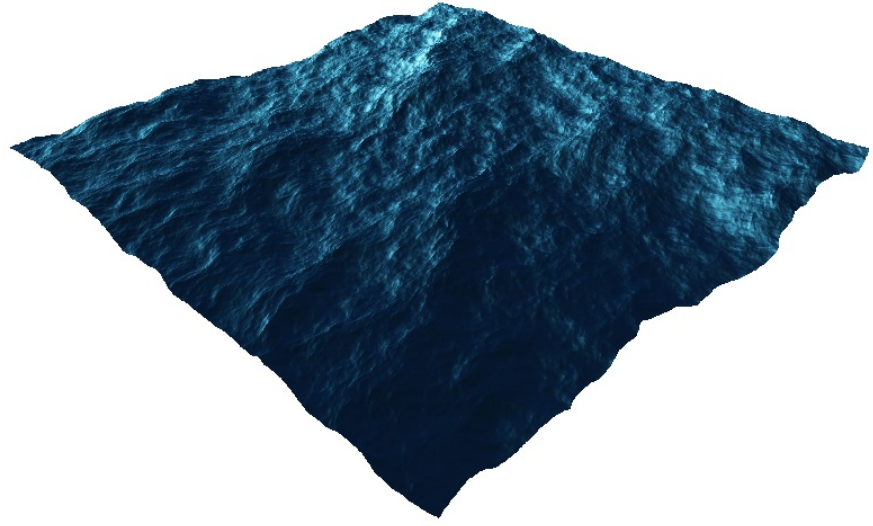
(-) Interaction with other objects



[Manteaux et al. Space-time sculpting of liquid animation, MIG 2016]

Free surface water - Example





Link Three.js

Animating fluids

SPH

General SPH idea

Pure Lagrangian: particles can follow surface deformation precisely

But how to evaluate values (density, speed, etc) between particles ?

Pure Eulerian: Values are known at fixed point in space

But how to follow deformation of surface boundaries ?

⇒ SPH Idea:

- Reconstruct surface from arbitrary samples in space using convolution
- Allow samples to be advected with the fluid

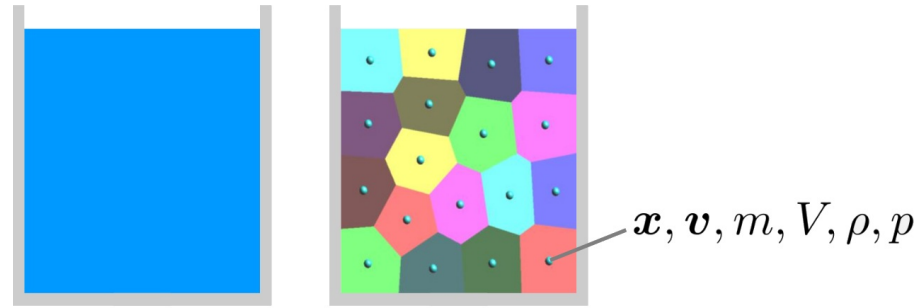
(+) Particle based model: can interact with other shapes

(+) Handle naturaly arbitrary boundaries

(+) Scalable

Initial used in Astronomy

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



Sampling and density

Arbitrary quantity A at position p can be expressed as a convolution

$$A(p) = (A \star \delta)(p) = \int_\Omega A(q)\,\delta(p-q)\,\mathrm{d}q$$

First approximation: $\delta \to W_h$

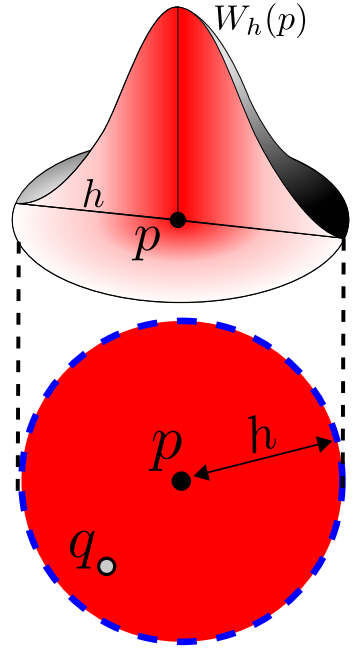
$$A(p) \simeq \int_\Omega A(q)\,W_h(p-q)\,\mathrm{d}q$$

W_h smooth kernel with limited support (/test functions)

W_h converge to δ when $h \to 0$

$$\Rightarrow \forall h, \int_\Omega W_h(p)\mathrm{d}p = 1$$

Classical *low pass* reconstruction (/average) for functions around p



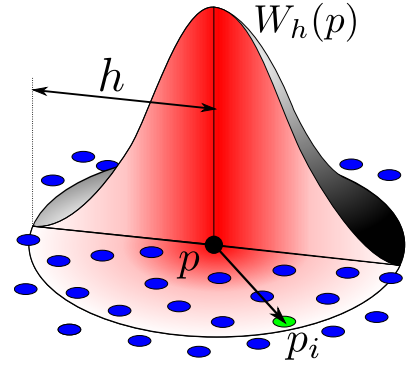
Model

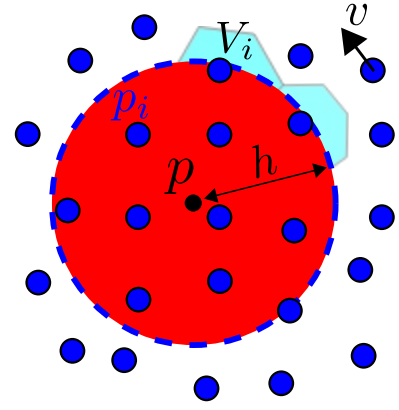
Second approximation : discrete sampling of A at position p_i.

$$A(p) = \int_\Omega A(q)\, W_h(p-q)\, \mathrm{d}q$$

$$A(p) = \sum_i A(p_i)\, W_h(p-p_i)\, V_i$$

V_i is the small volume (/area in 2D) associated to the ith samples.

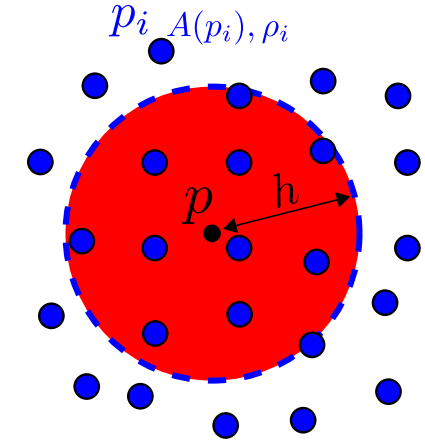




SPH Model for fluid

For a fluid: $m_i = \rho_i \, V(p_i)$

- m_i total mass associated to the volume at sample i
- ρ_i local density (supposed constant over the volume)



- General SPH formulation for a quantity A at position p

$$A(p) = \sum_i \frac{m_i}{\rho_i} A(p_i) \, W_h(p - p_i)$$

- Special case of the density

$$\rho(p) = \sum_i m_i \, W_h(p - p_i)$$

SPH Relations

- Local value at position p

$$A(p) = \sum_i \frac{m_i}{\rho_i} A(p_i)\, W_h(p - p_i)$$

- Gradient (symmetric) at position p

$$\nabla A(p) = \rho(p) \sum_i m_i \left(\frac{A(p)}{\rho^2} + \frac{A(p_i)}{\rho_i^2} \right) \nabla W_h(p - p_i)$$

- Laplacien (symmetric) at position p

$$\triangle A(p) = 2 \sum_i \frac{m_i}{\rho_i} \frac{(p - p_i) \cdot (A(p) - A(p_i))}{\|p - p_i\|^2 + \epsilon\, h^2} \nabla W_h(p - p_i),\ \epsilon = 10^{-2}$$

Note: derivatives are symetrized such that $\nabla A(p_i)/p_j = \nabla A(p_j)/p_i$

See for instance [Desbrun and Cani, EGCAS 96]

SPH for Navier Stokes

Continuous case

$$\frac{\mathrm{d}v}{\mathrm{d}t} = g - \frac{\nabla \mathrm{p}}{\rho} + \nu \triangle v$$

Using discrete SPH formulation at position p_i

$$v_i'(t) = g - \sum_j m_j \left(\frac{\mathrm{p}_i}{\rho_i^2} + \frac{\mathrm{p}_j}{\rho_j^2} \right) \nabla W_h(p_i - p_j) + 2\nu \sum_j \frac{m_j}{\rho_j} \frac{(p_i - p_j) \cdot (v_i - v_j)}{\|p_i - p_j\|^2 + \epsilon h^2} \nabla W_h(p_i - p_j)$$

- Pression given by the Tait equation of state

  $\mathrm{p}(p) = K \left(\frac{\rho(p)}{\rho_0} - 1 \right)^\alpha, \rho > \rho_0$

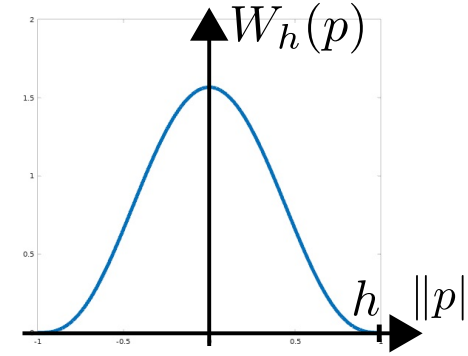
  ρ_0: rest density $1000\, Kg/m^3$, $\alpha \simeq 7$ for water, K stiffness constant.

- Density expressed using SPH evaluation $\rho(p_i) = \sum_j m_j W_h(p_i - p_j)$

SPH Kernel

- Smooth kernel

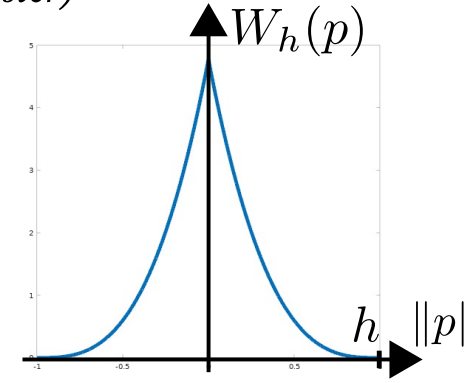
$$W_h(p) = \frac{315}{64\,\pi\,h^3}\left(1 - \left(\frac{\|p\|}{h}\right)^2\right)^3, \quad \|p\| \leq h$$



- Spiky kernel

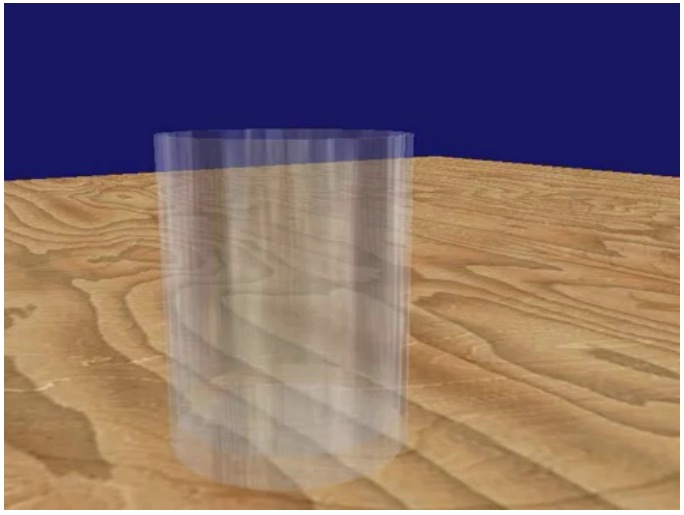
Can give better results when used to compute pression force (avoid particles cluster)

$$W_h(p) = \frac{15}{\pi\,h^3}\left(1 - \frac{\|p\|}{h}\right)^3, \quad \|p\| \leq h$$



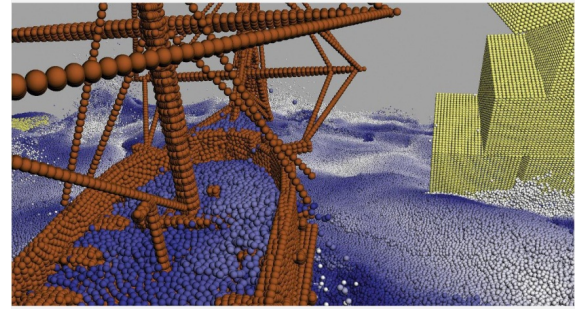
- [Desbrun and Cani, Smoothed Particles: A new paradigm for animating highly deformable bodies, EGCAS 1996]
- [M. Muller et al., Particle-Based Fluid Simulation for Interactive Applications, SCA 2003]
- [M. Ihmsen et al., SPH Fluids in Computer Graphics, EG STAR 2014]

SPH examples



Muller 2003

M. Teschner 2012 - 20M particles



SPH extensions

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

Not only fluids

(-) Not well understood accuracy

(-) Compressible

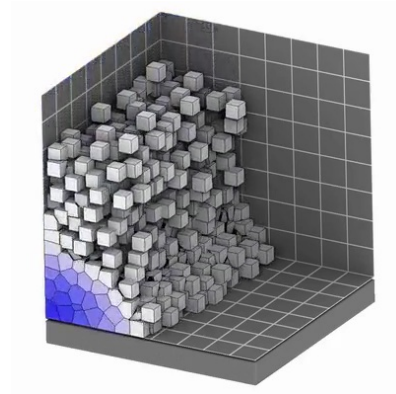
[Solenthaler et al., Predictive-Corrective Incompressible SPH, ACM SIGGRAPH 2009]
[Ihmsen et al, Implicit Incompressible SPH, IEEE TVCG 2013]

(-) Limited time step

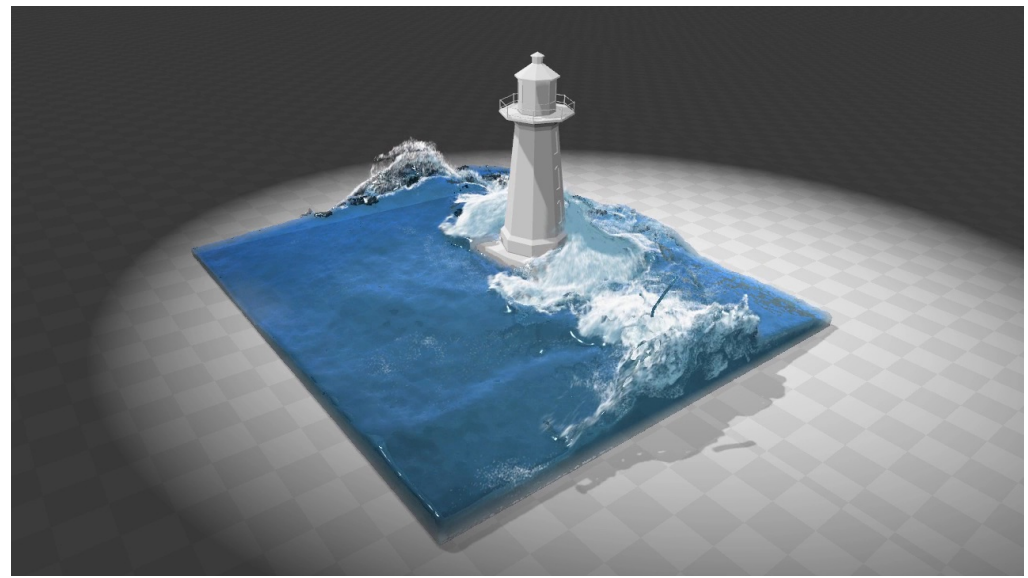
[Macklin and Muller, Position based Fluids, ACM SIGGRAPH 2013]

(-) Boundaries are hard to handle

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



Bruno Levy

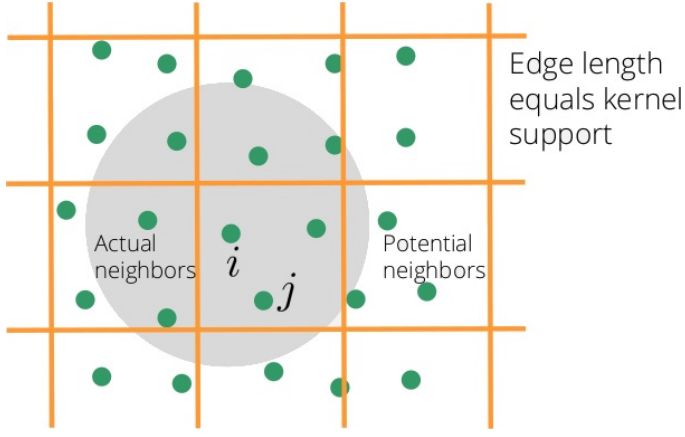


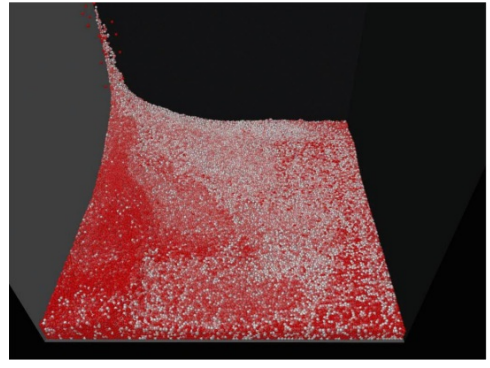
Macklin and Muller

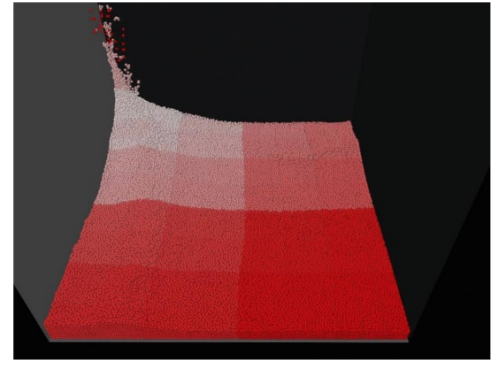
Acceleration structure

SPH based on pair-wise interation $\Rightarrow$ 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

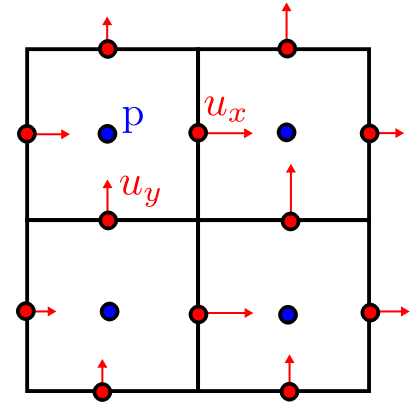
PIC/FLIP

Mix between particles and grid based approach.

Particles: advection, Grid: forces, pression, viscosity

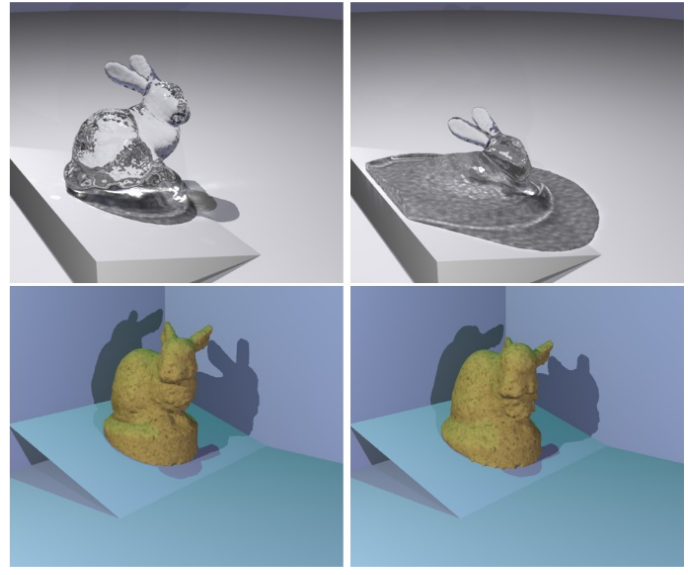
Use MAC grid (Marker-and-Cell)

Grid storing velocity on middle-edges, pressure on grid center.



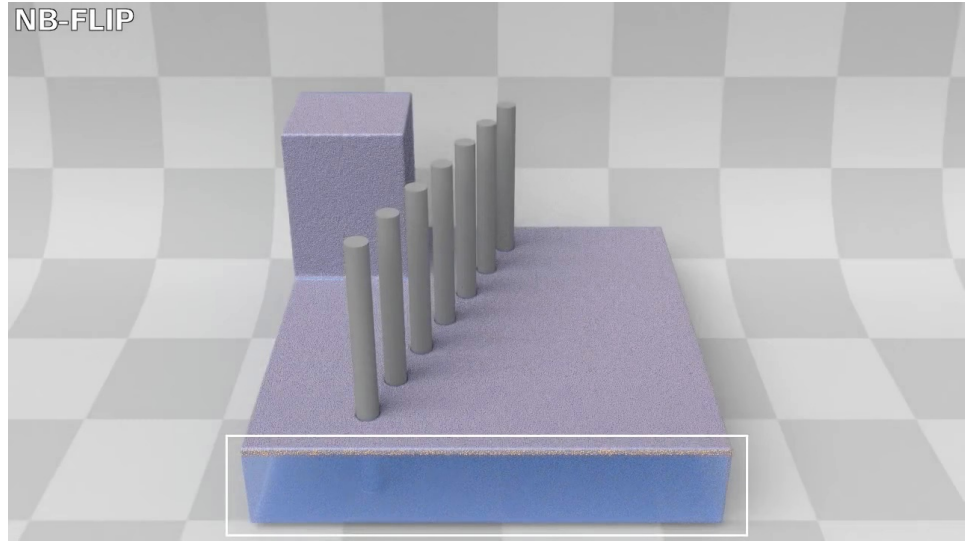
- **PIC** approach - Transfert velocity from grid to particles
- **FLIP** approach - Add velocity difference from grid to particles.
- **PIC/FLIP** : blending b/w two approaches

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



PIC/FLIP Method

- Transfert particle velocity to the MAC grid (Store velocity u^k on grid)
- Evolve velocity on grid (pression, 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 grid velocity field (solve ODE)



[F. Ferstl et al., EUROGRAPHICS 2016]