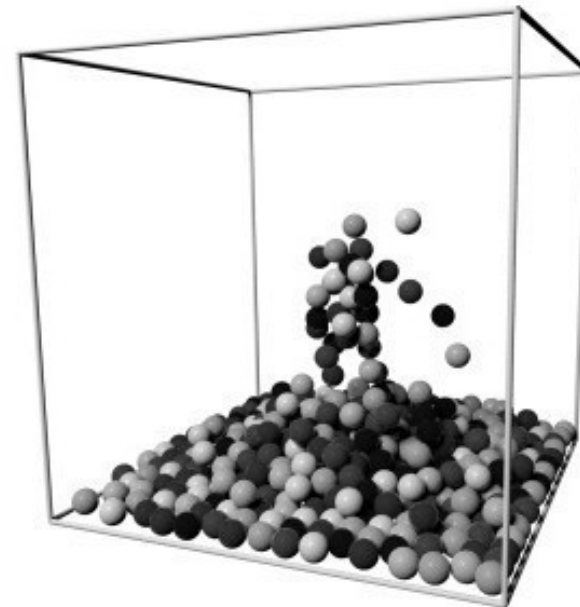
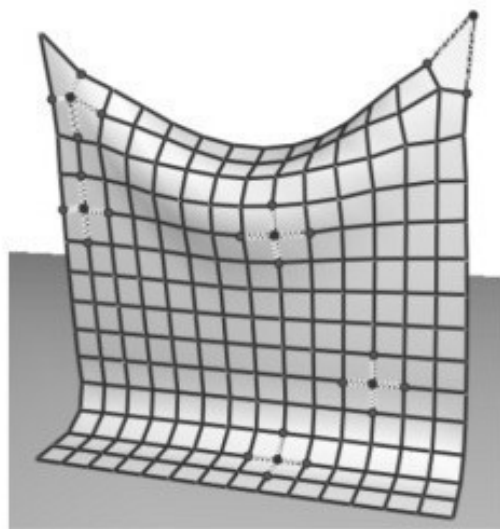


$$\frac{u^{k+1} - u^k}{\Delta t} = A u^{k+1} + b$$

Physically based animation



000

Physically based animation

When using physically based animation?

For complex deformation we cannot model "by hand"
To model physical phenomenon



[Fedkiw SIGGRAPH 06], [Grinspun SIGGRAPH 07]

001

Principle

1. Define a system S_0 at $t = 0$
positions, speed, forces
2. Model the temporal evolution of the system using
ODE / PDE

$$\mathcal{F} \left(S(t), \frac{\partial^n S}{\partial t^n}, \frac{\partial^m S}{\partial x^m} \right) = 0$$

We usually considers $\frac{\partial S}{\partial t} = AS + \mathcal{H} \left(S(t), \frac{\partial^m S}{\partial x^m} \right)$

3. We numerically solve in moving forward in
time by Δt

002

Modeling

Level of precisions

1. Particles systems

Everything is reduced to a point (no rotation, EDO)

+Simple
- Accuracy

2. Solid mecanics (rigid bodies)

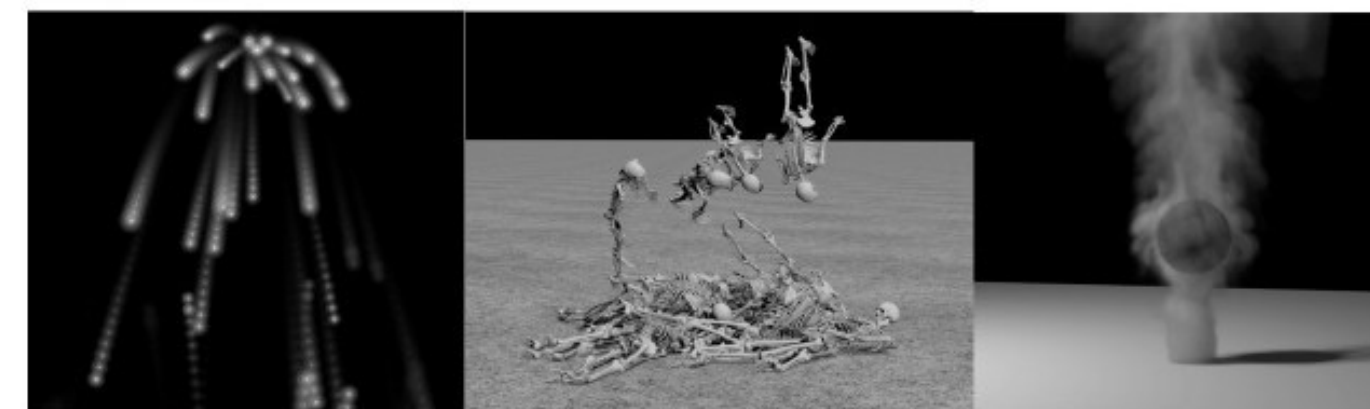
Unique set of parameters for the entire object

+Inertia
- Collision

3. Continuum mechanics

Varying parameters within the shape (PDE: FEM, FV, ...)

+ Accurate
- Heavy



003

Modeling

Level of precisions

1. Particles systems

$$ma = \sum_i f_i \Rightarrow x_i'' = F(x, x', t)$$

2. Solid mechanics (rigid bodies)

$x(t)$: position

$R(t)$: rotation

$\omega(t)$: angular speed

$L(t)$: angular momentum

$\tau(t)$: torque

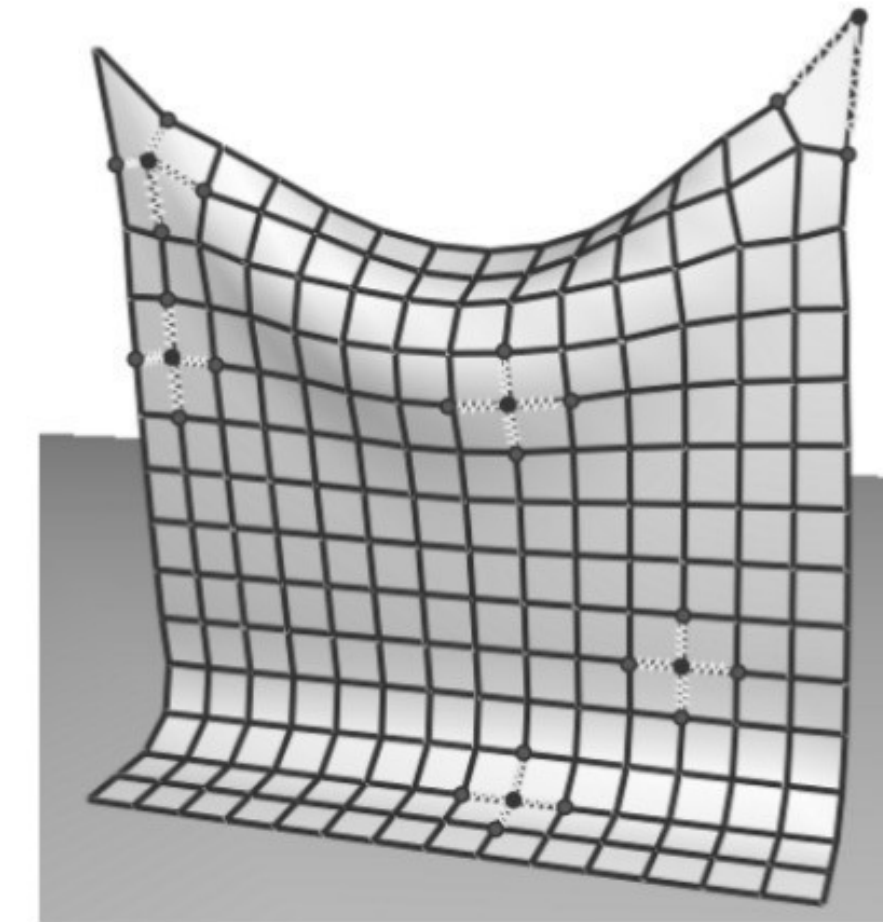
$$\frac{d}{dt} \begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t)^* R(t) \\ F(t) \\ \tau(t) \end{pmatrix}$$

3. Continuum mechanics

$$\begin{aligned} \text{stress} \quad \nabla \cdot \sigma + \mathbf{F}_{\text{ext}} &= \eta \frac{d^2 \mathbf{u}}{dt^2} \quad \text{deformation} \\ \sigma &= E \epsilon_{\text{strain}} \\ \epsilon &= \frac{1}{2} (\nabla \mathbf{u} + [\nabla \mathbf{u}]^T + [\nabla \mathbf{u}]^T \nabla \mathbf{u}) \end{aligned}$$

004

Cloth Simulation

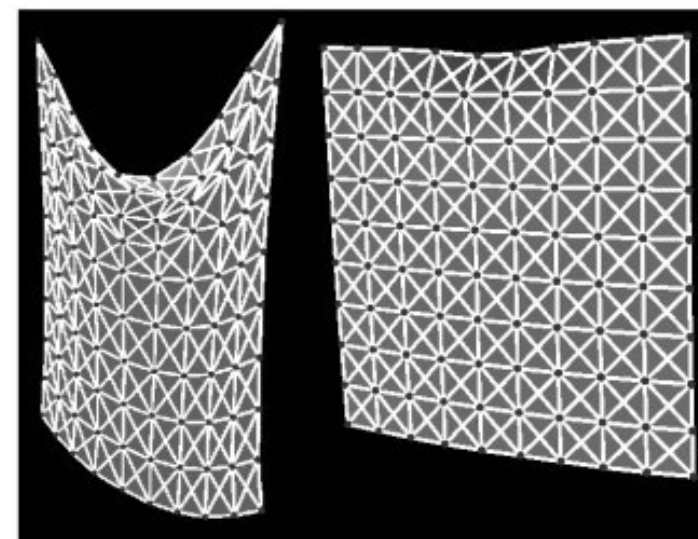
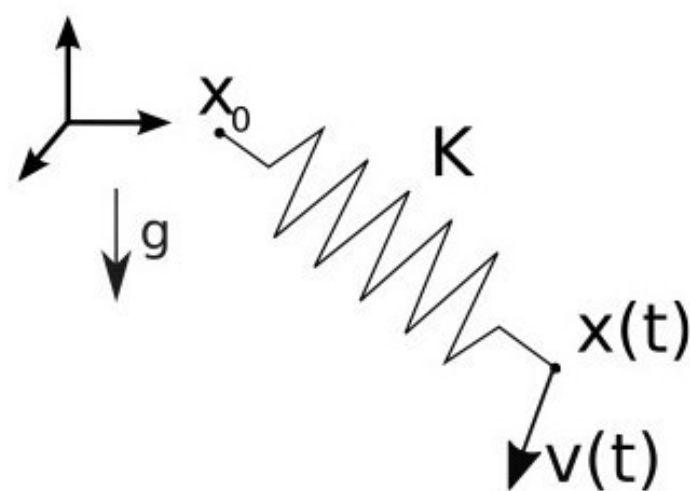


005

Cloth Simulation

In 3D:

$$F(t) = K (L_0 - \|\mathbf{p} - \mathbf{p}_0\|) \frac{\mathbf{p} - \mathbf{p}_0}{\|\mathbf{p} - \mathbf{p}_0\|}$$



006

Cloth Simulation: mass springs

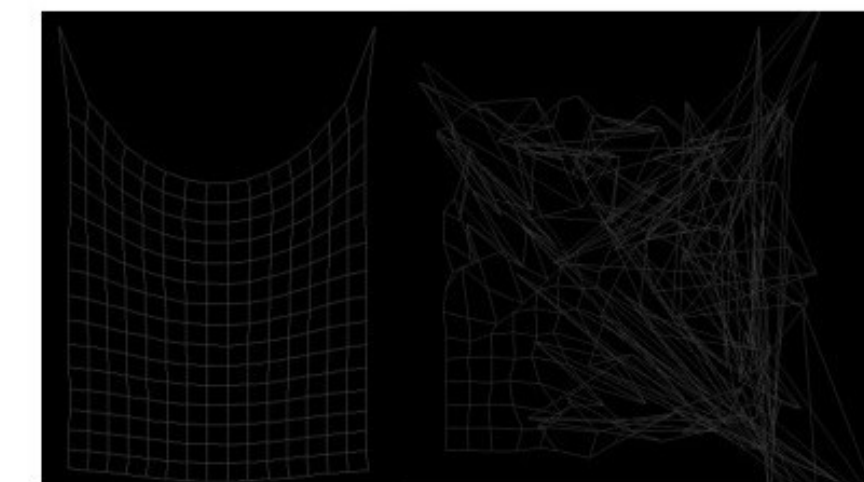
Model cloth using coupled springs

Force on a vertex i with neighbors $\mathcal{V}_i$

$$F(x_i, t) = \sum_{j \in \mathcal{V}_i} K^{ij} (L_0^{ij} - \|x_i - x_j\|) \frac{x_i - x_j}{\|x_i - x_j\|} + g$$

$$\forall i, \begin{cases} x_i'(t) = v_i(t) \\ v_i'(t) = \frac{1}{m_i} \sum_j K^{ij} (L_0^{ij} - \|x_i - x_j\|) \frac{x_i - x_j}{\|x_i - x_j\|} + g \end{cases}$$

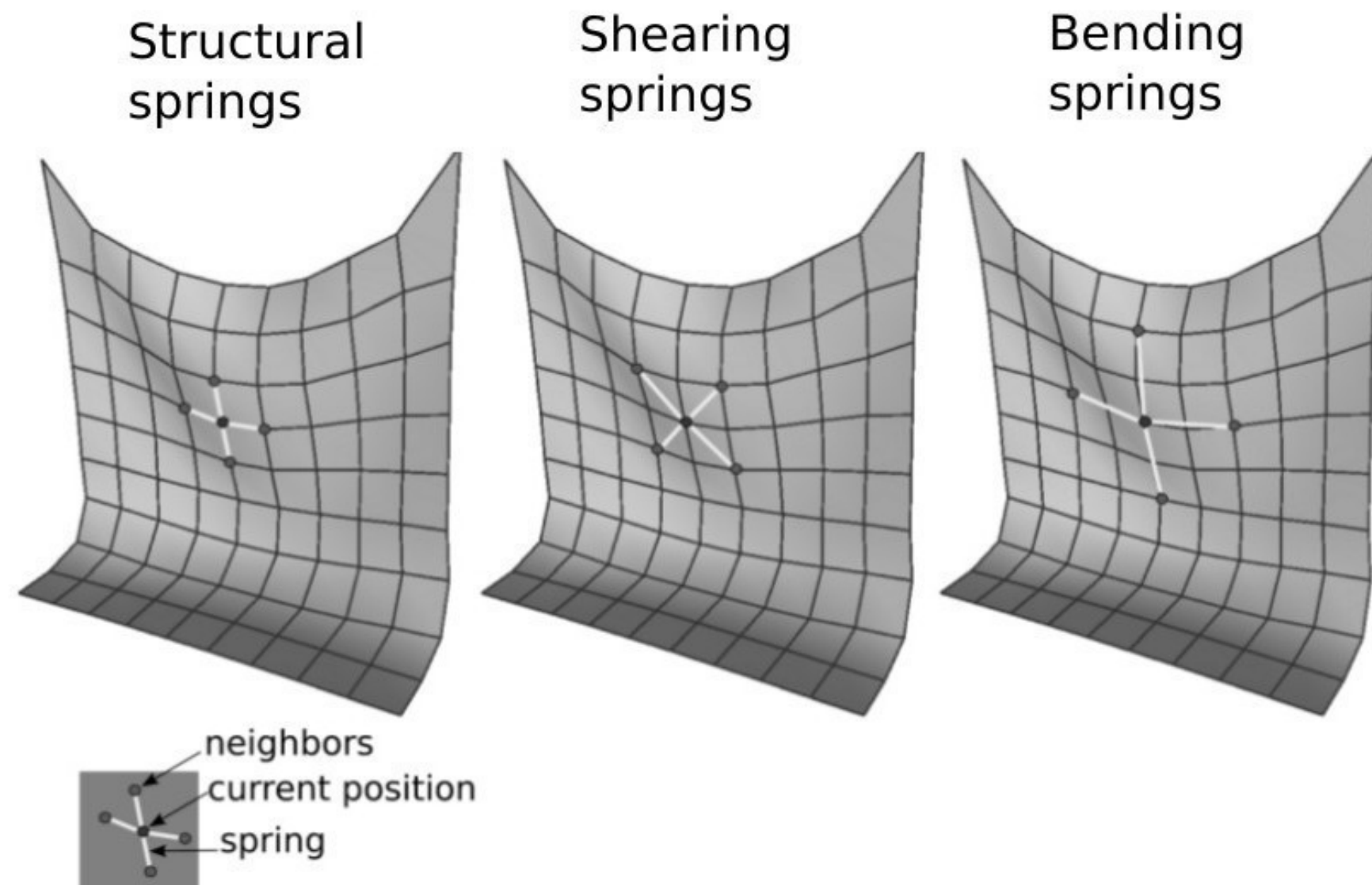
Use you best interpolation scheme



[P. Jacobs]

007

Cloth Simulation: spring types



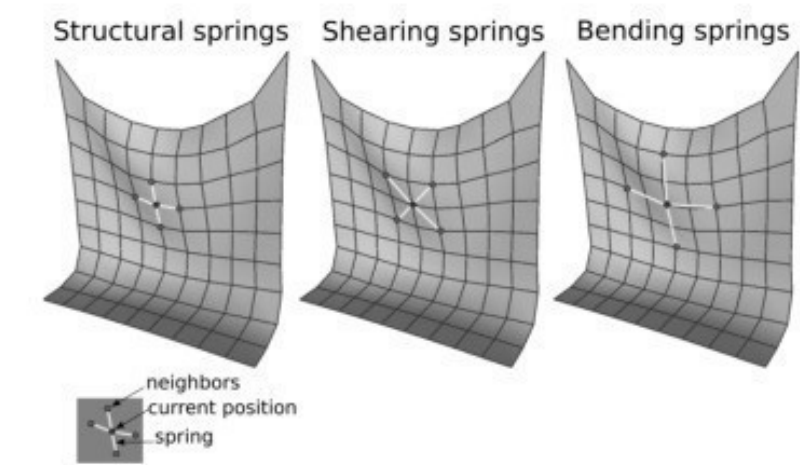
008

Cloth Simulation: Resolution

Explicit Euler

```
//Compute forces
For all i
  For j: 4 direct neighbors (structural): K=K1
         4 diagonal neighbors (shear) : K=K2
         8 neighbors (bend) : K=K3
  u=p[i]-p[j]
  F[i] += K (L0-norm(u)) *u/norm(u)

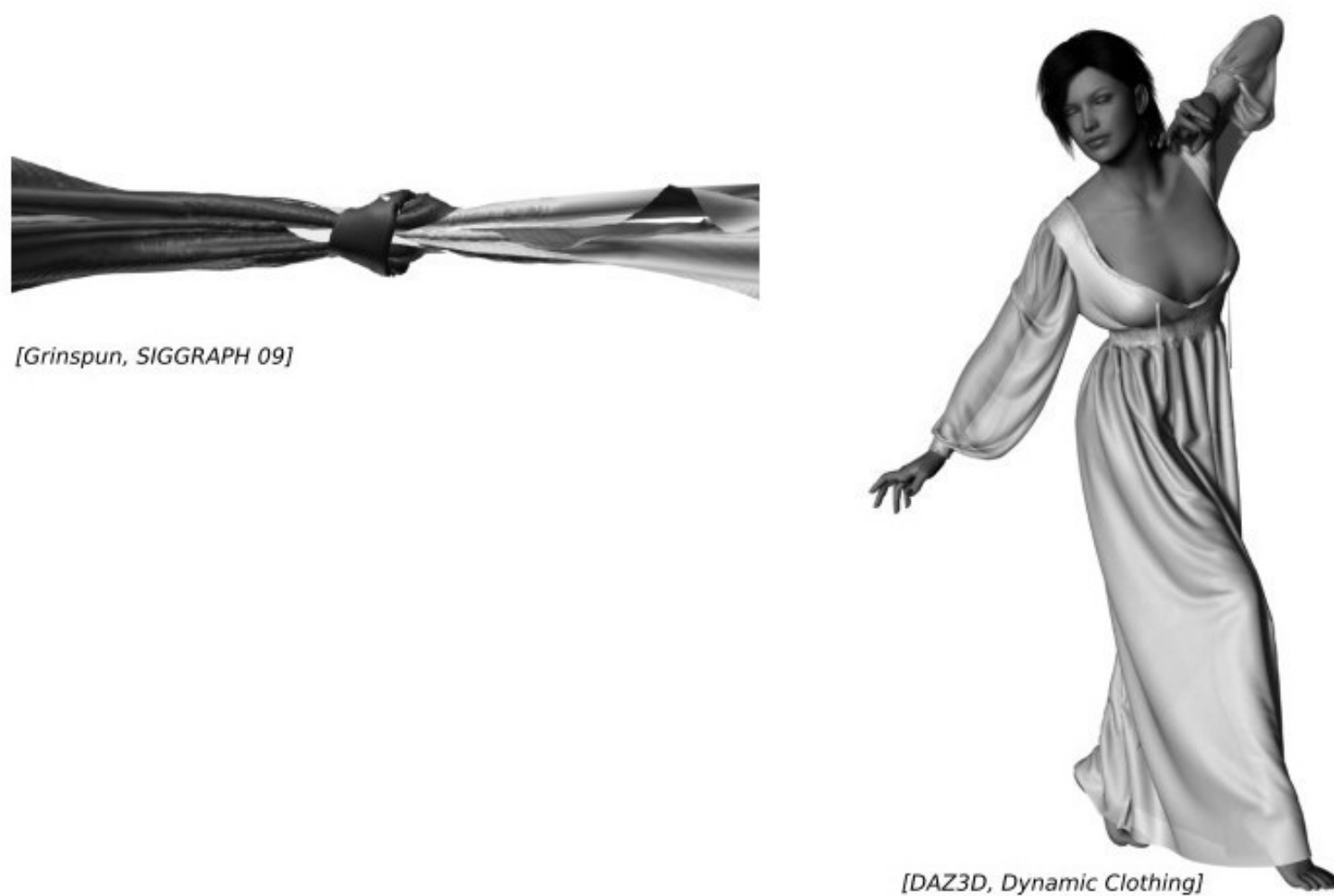
For all i
  v[i] += dt *F[i]
  p[i] += dt *v[i]
```



009

Complex cloth simulation

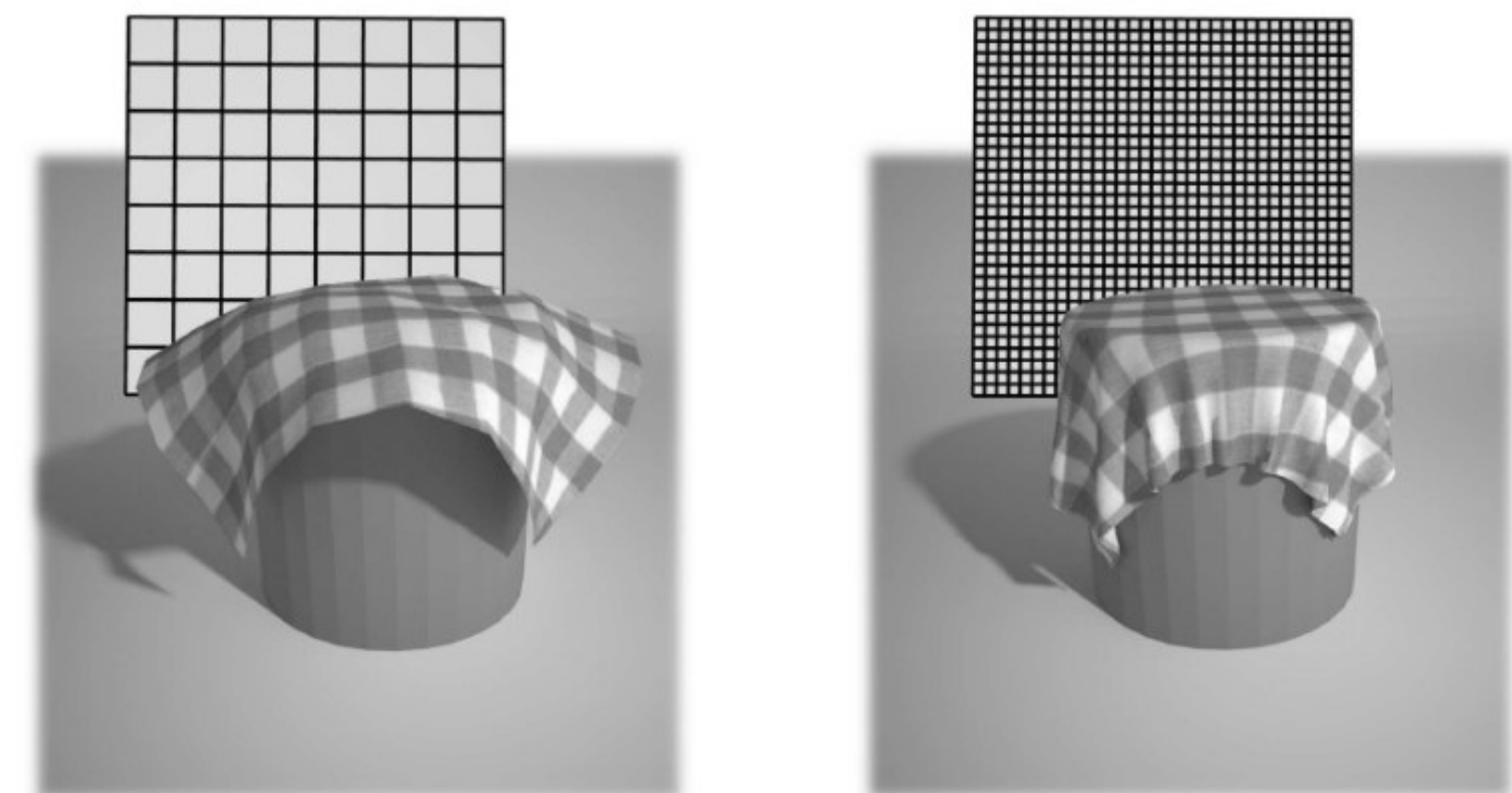
Full simulation = Cloth + complex collisions



010

Cloth simulation: Limitations

Mesh influence



011

Implicit scheme

Write the system in a vectorial form

$$u(t) = (p_0(t), p_1(t), \dots, p_{N-1}(t), v_0(t), \dots, v_{N-1}(t))$$

$$u'(t) = \mathcal{F}(u(t))$$

Non linear system with N unknown: impossible to invert
Linearize the system in $p(t) - p_0 = \Delta L v(t) + \mathcal{O}((\Delta t)^2)$

Get a linear system.

Loss of the unconditional stability:

In practice, very stable

012

Implicit scheme

Linearization:

$$\mathbf{f}(\mathbf{p}_n + \Delta p, \mathbf{v}_n + \Delta v) = \mathbf{f}_n + \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \Delta p + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \Delta v$$

Solve a linear system at each time step

$$\mathbf{A} \Delta v = \mathbf{b}$$

$$\mathbf{A} = \mathbf{I} - \Delta t \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{v}} - \Delta t^2 \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{p}}$$

$$\mathbf{b} = \Delta t \mathbf{M}^{-1} \left(\mathbf{f}_n + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \mathbf{v}_n \right)$$

013

Collisions

014

Particles collisions

Particles can enter in collision with a plane:

$$\mathcal{P} : \langle \mathbf{x} - \mathbf{p}_0, \mathbf{n} \rangle = 0$$

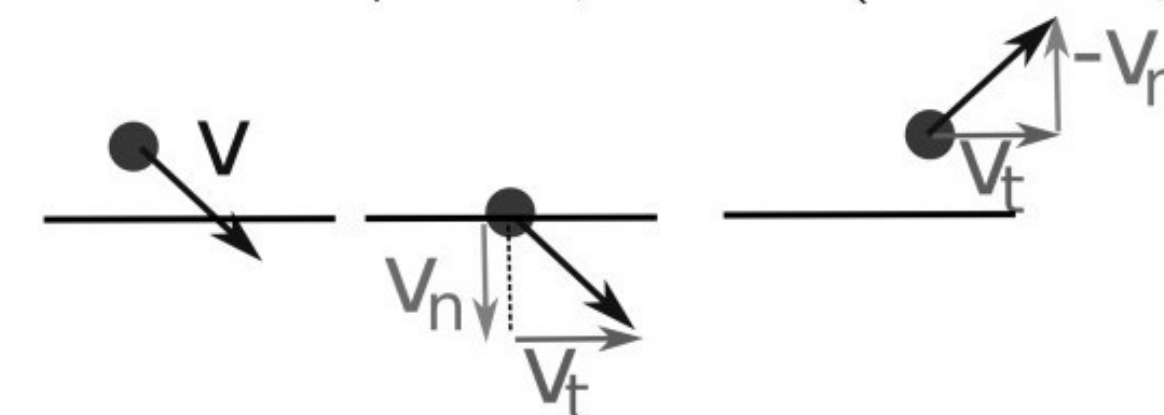
$$\text{Detection } \langle \mathbf{x} - \mathbf{p}_0, \mathbf{n} \rangle < 0$$

Separate tangential speed v_t and normal speed v_n

$$\begin{cases} v_n = \langle \mathbf{v}, \mathbf{n} \rangle \\ v_t = \mathbf{v} - \langle \mathbf{v}, \mathbf{n} \rangle \mathbf{n} \end{cases}$$

After collision, loss of energy: dumping

$$v^{k+1} = -\mu \langle v^k, \mathbf{n} \rangle \mathbf{n} + (v^k - \langle v^k, \mathbf{n} \rangle \mathbf{n})$$



015

Particles collisions

Need to take care of the temporal discretization

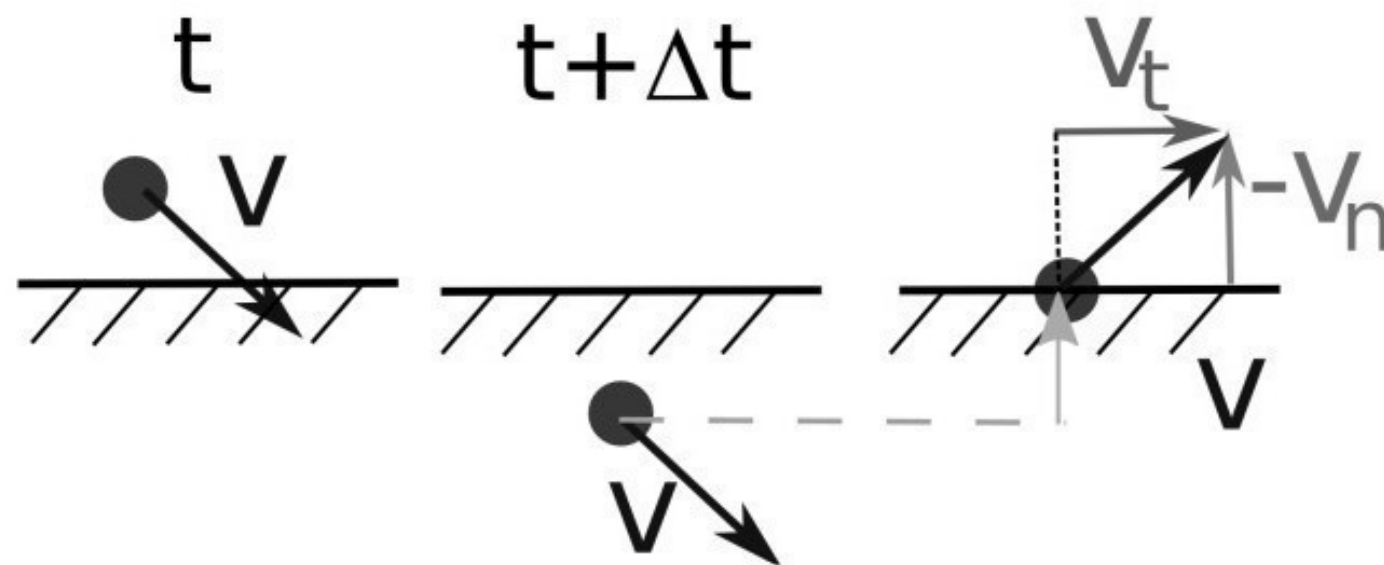
1. Projection of the particle

Easy to compute, physically biased (instabilities)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \langle \mathbf{x}^k - \mathbf{p}_0, \mathbf{n} \rangle \mathbf{n}$$

2. Backward search

Less easy, less wrong



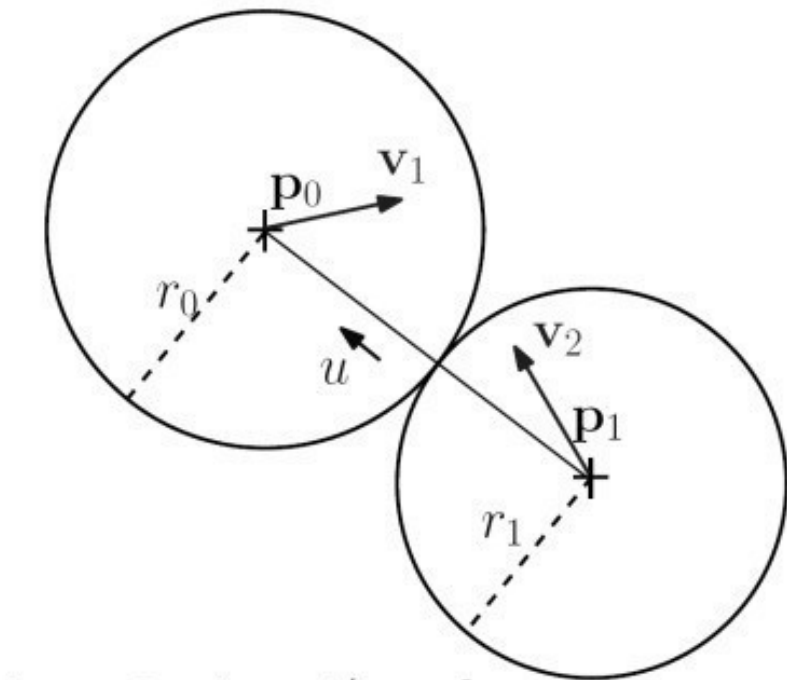
016

Model of hard spheres

Particles => Sphere of mass m, center x, radius r

Collision if

$$\|\mathbf{x}_1 - \mathbf{x}_2\| < r_1 + r_2$$



New speed after collision

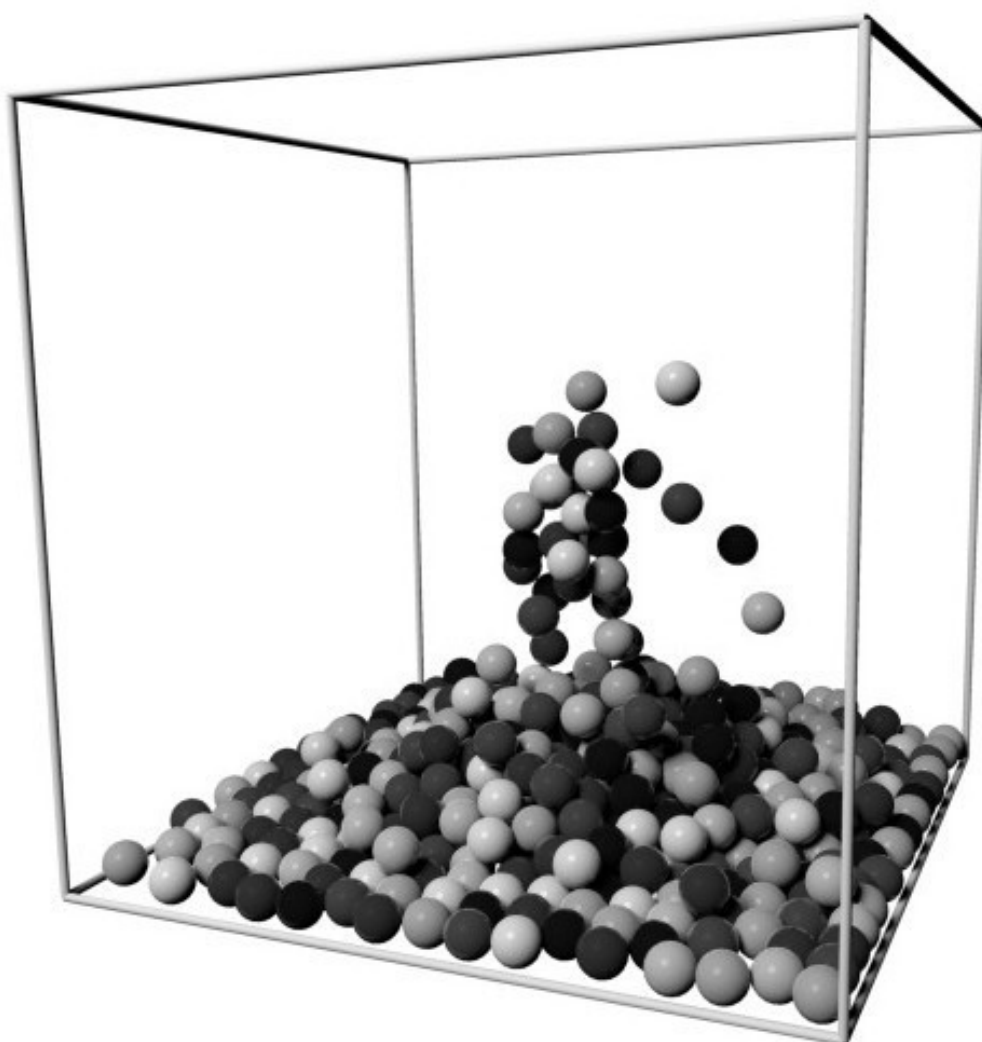
$$\begin{cases} v_1^{k+1} = v_1^k + \frac{1}{m_1+m_2} [m_2 \langle v_2^k, u \rangle - \frac{1}{2}(m_1+3m_2) \langle v_1^k, u \rangle] u \\ v_2^{k+1} = v_2^k + \frac{1}{m_1+m_2} [m_1 \langle v_1^k, u \rangle - \frac{1}{2}(m_2+3m_1) \langle v_2^k, u \rangle] u \\ u = \frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|} \end{cases}$$

Don't forget to project back to the collision surface

017

Model of hard spheres

Generate a lot of spheres



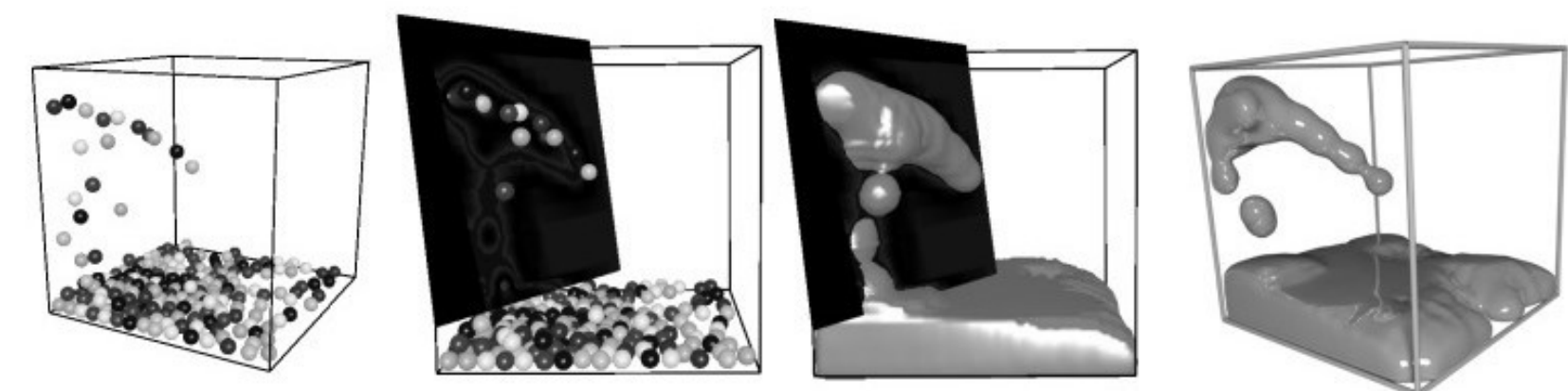
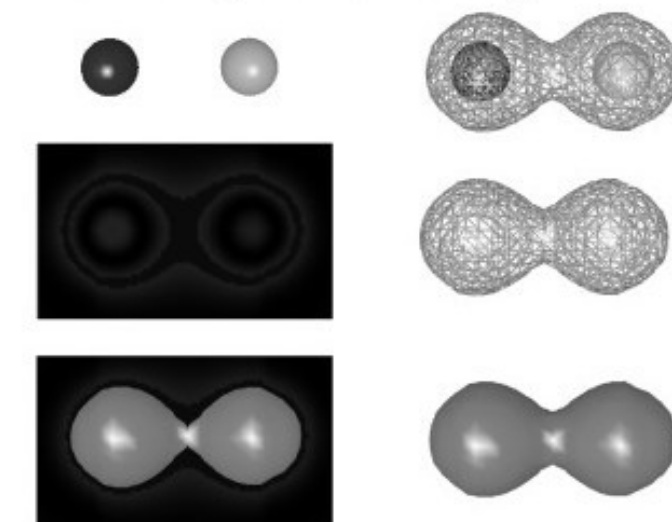
018

Sphere: implicit functions

Set a field function to each particle (ex. blobs)

$$\psi_i(\mathbf{p}) = e^{-\frac{\|\mathbf{p} - \mathbf{p}_i\|^2}{\sigma^2}}$$

=> Fluid simulation



019

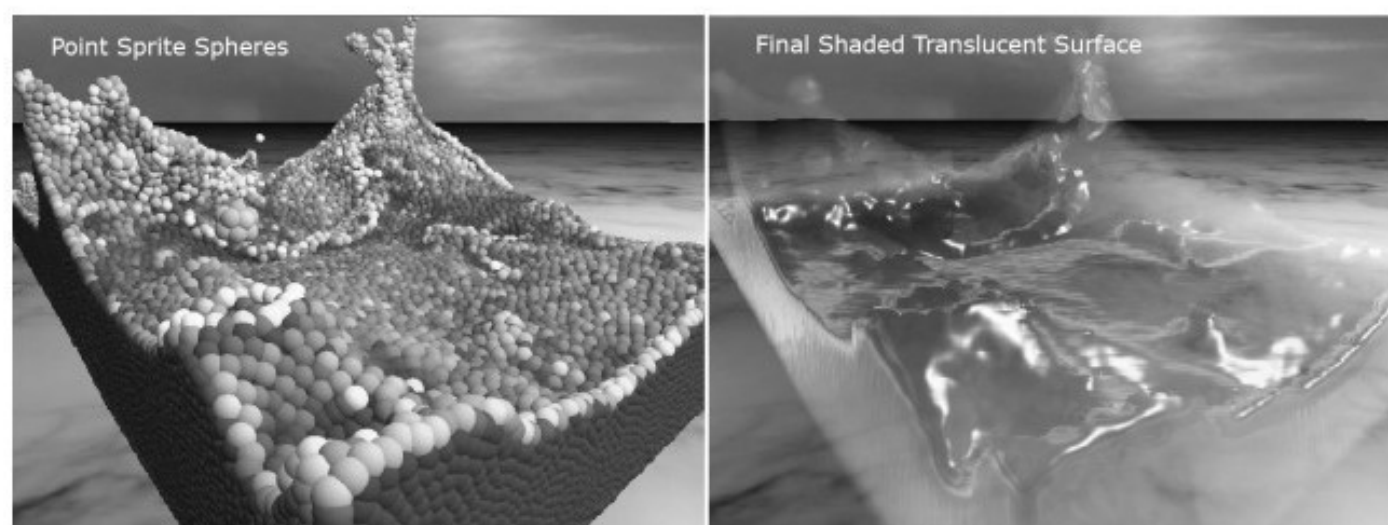
Efficient collision detection

Brute force algorithm

```
for (i=0; i<N; ++i)
  for (j=i+1; j<N; ++j)
    if (norm(xi-xj)<r1+r2)
      CollisionResponse(i, j)
```

Complexity in $O(N^2)$

Impossible to simulation 10^{3-6} particles in real time



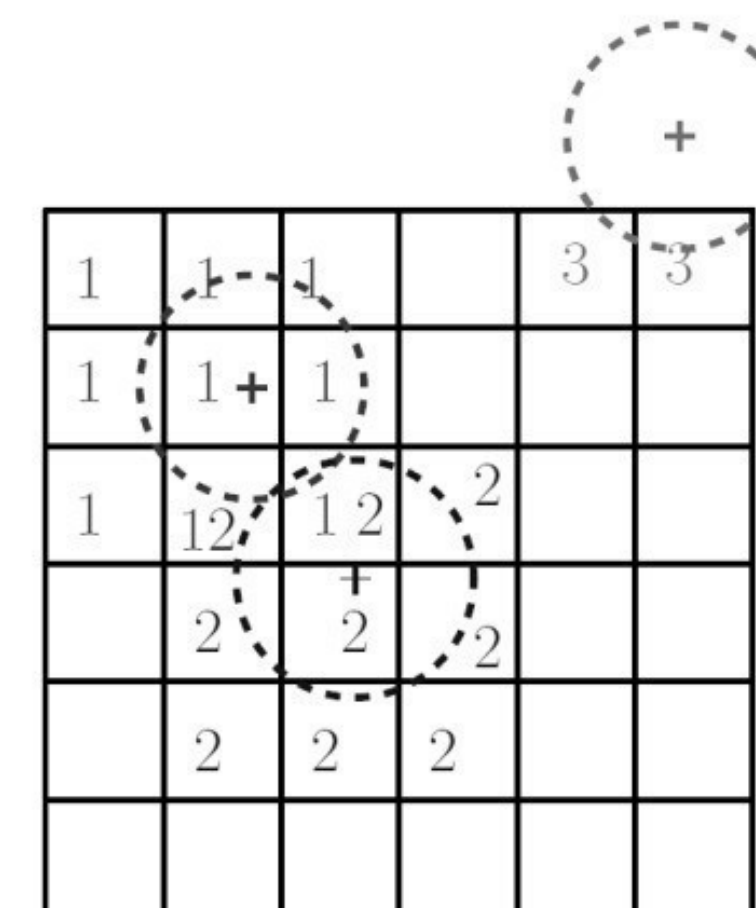
[NVIDIA, Green SIGGRAPH 10]

020

Acceleration structure

Regular grid
Research in $O(1)$

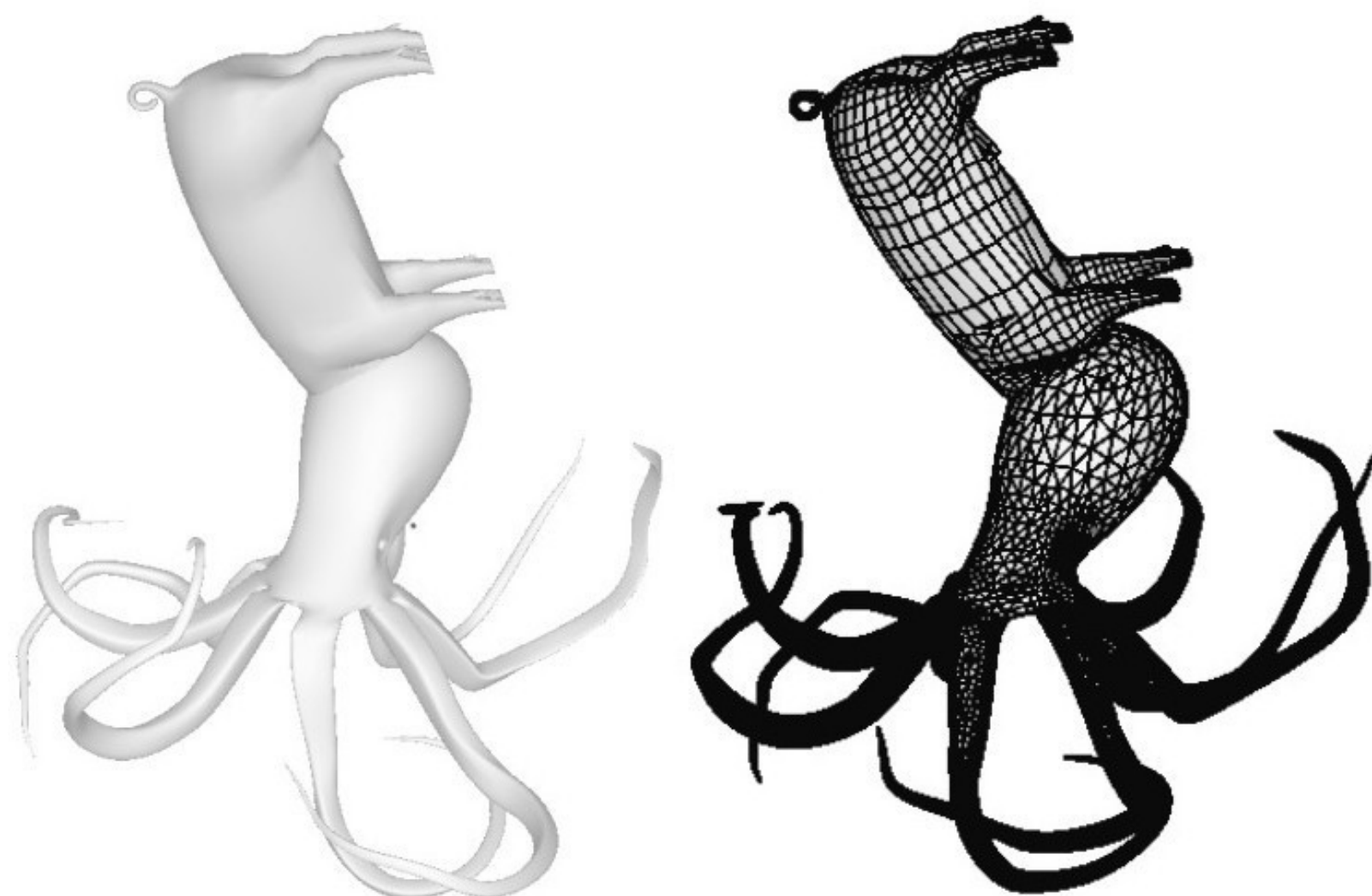
- + Simple
- + Efficient research
- + Good for uniform samples
- Memory consumption for empty areas



021

Collision detection between objects

Collision bw triangles/triangles in $O(N_{t1} N_{t2})$



022

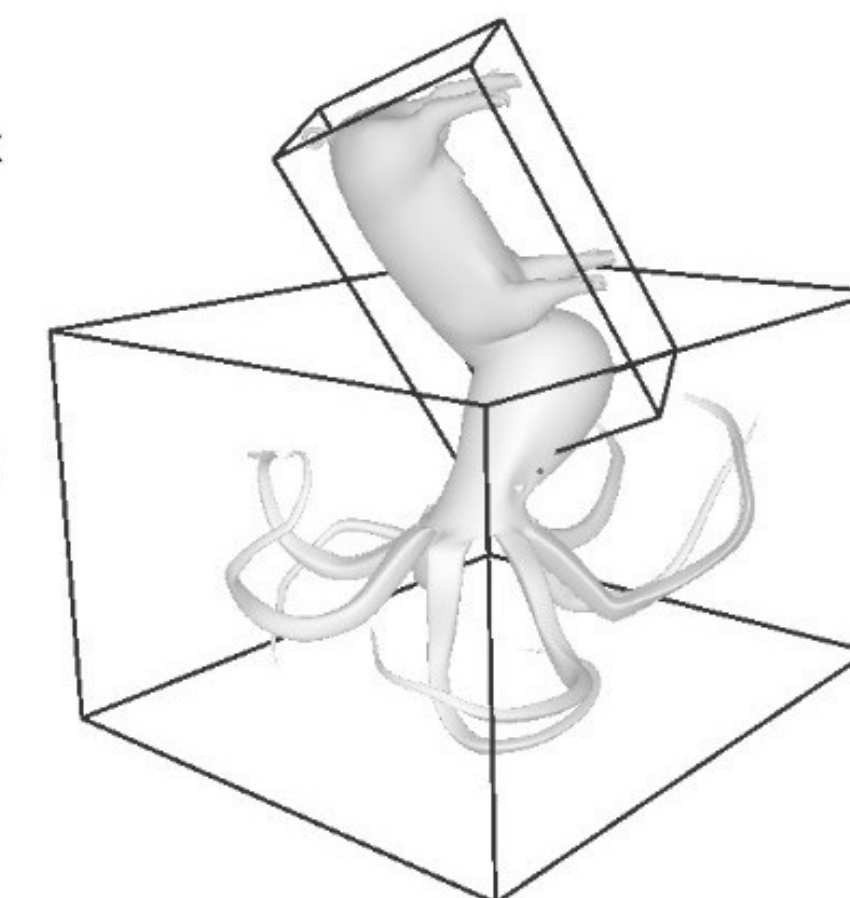
Bounding boxes

Bounding Box (BB)

The simplest: **AABB**
Axis **A**ligned **B**ounding **B**ox

Bounding Spheres

- + Detection of non-collision in $O(N_{obj}^2)$

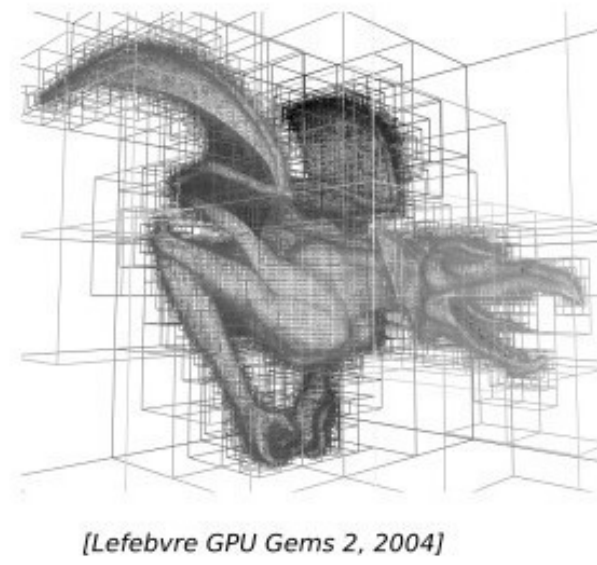
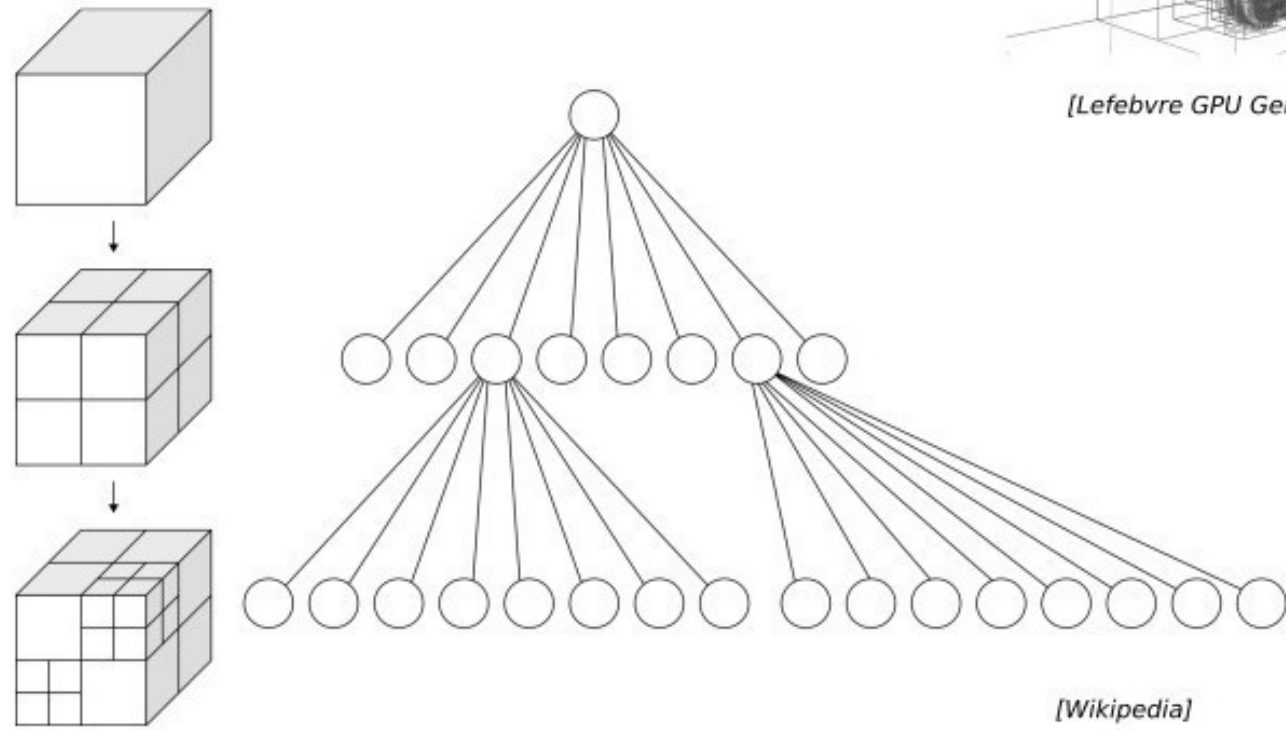


023

Octree

Self-adaptable grid structure

- + Can adapt to complex geometry
- + Research in $O(\log(M))$, M =tree depth



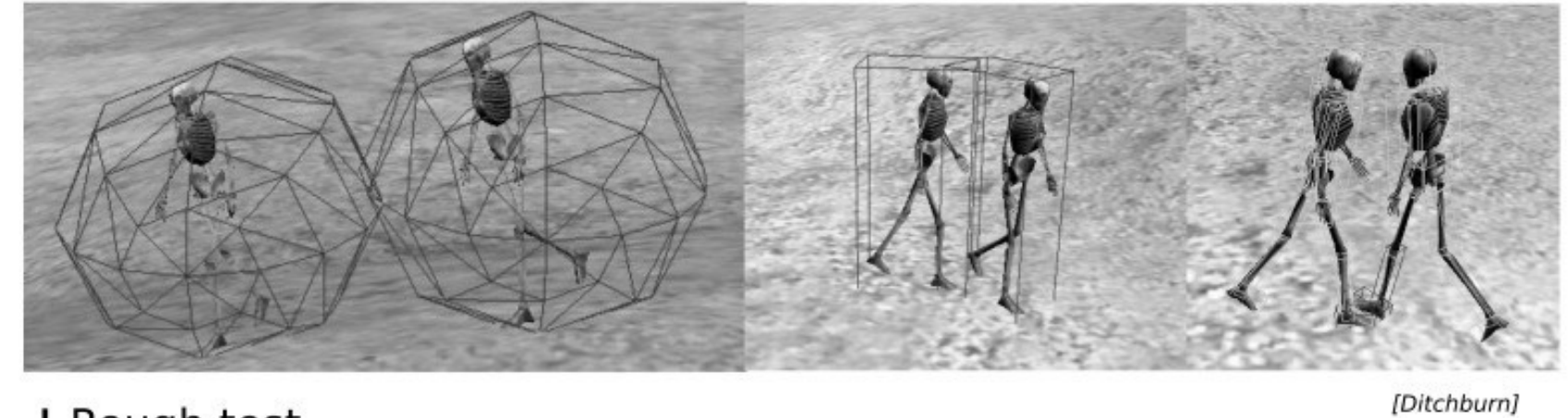
024

Collisions

In practice: Several levels of details

ex.

- Spheres within BB within Octree
- OBB within AABB within Spheres



Rough test

Less rough test

...

Fine test

...

Eventually: Compute the real intersection (rare)

=> Non collision tests much more efficient than collision

025