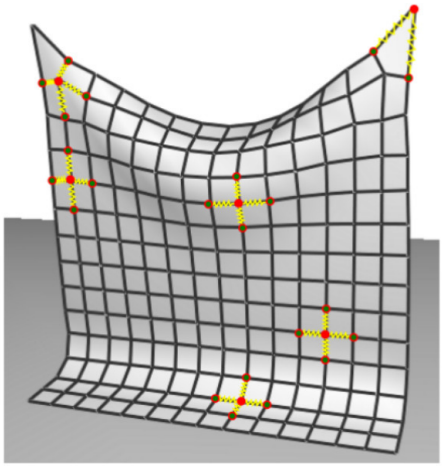
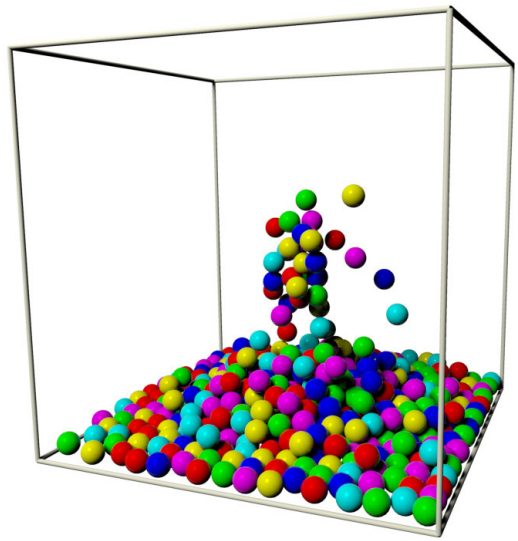
$$\frac{u^{k+1} - u^k}{\Delta t} = \mathrm{A}\, u^{k+1} + \mathrm{b}$$

Physically based animation

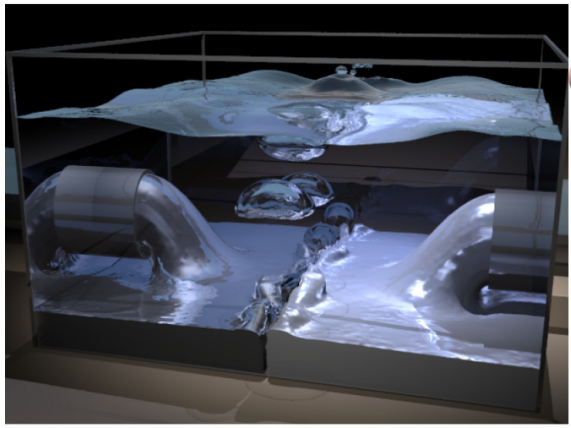


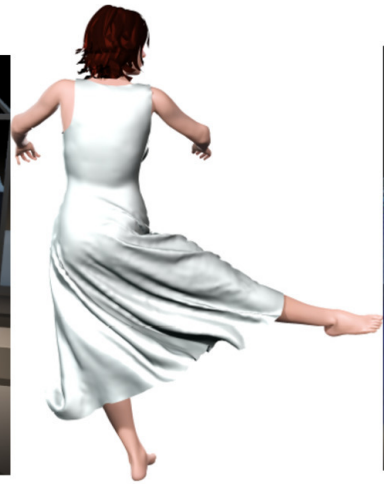


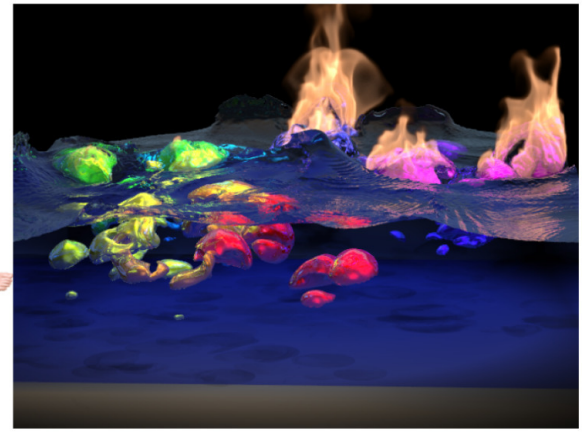
Physically based animation

When using physically based animation?

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







[Fedkiw SIGGRAPH 06], [Grinspun SIGGRAPH 07]

Principle

1. Define a system $S_0 \text{ at } t = 0$

   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 $\frac{\partial S}{\partial t} = \mathrm{A}S + \mathcal{H}\left(S(t), \frac{\partial^m S}{\partial x^m}\right)$

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

Modeling

Level of precisions

1. Particles systems
Everything is reduced to a point (no rotation, EDO)

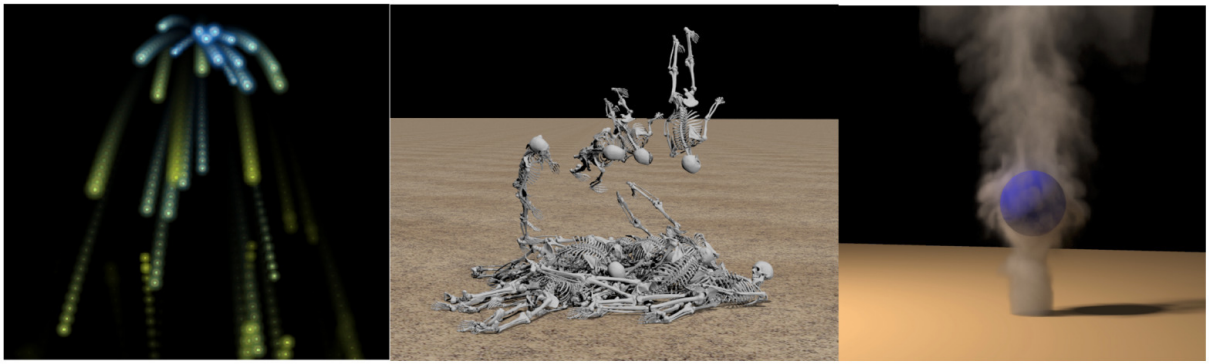
+Simple
- Accuracy

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



Modeling

Level of precisions

1. Particles systems

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

2. Solid mecanics (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

$$\nabla \cdot \sigma + \mathbf{F}_{\text{ext}} = \eta \frac{\mathrm{d}^2 \mathbf{u}}{\mathrm{d}t^2}$$

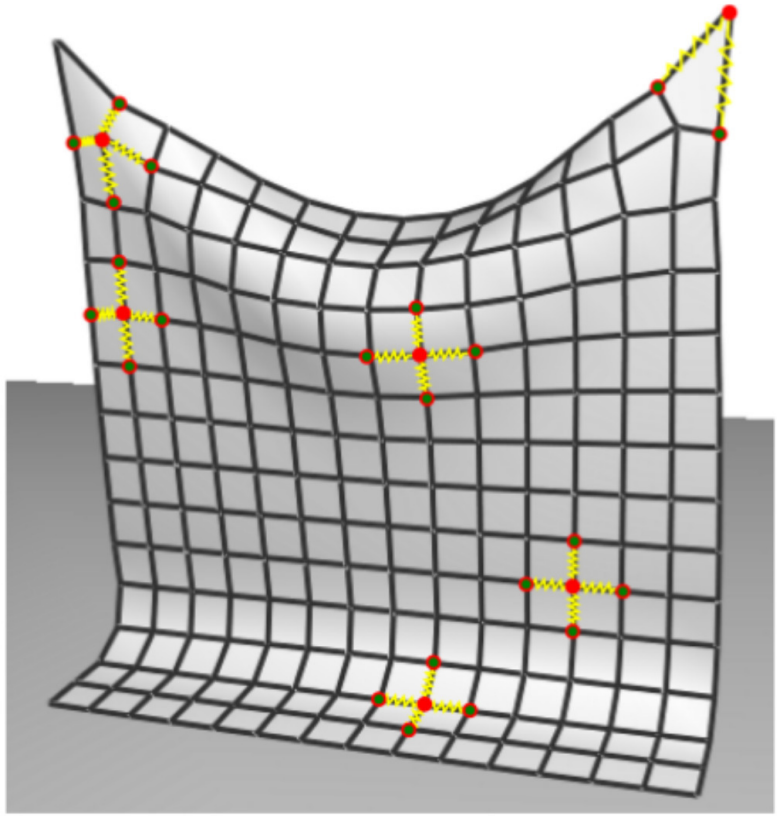
stress, deformation

$$\sigma = E\epsilon$$

strain

$$\varepsilon = \frac{1}{2}(\nabla \mathbf{u} + [\nabla \mathbf{u}]^T + [\nabla \mathbf{u}]^T \nabla \mathbf{u})$$

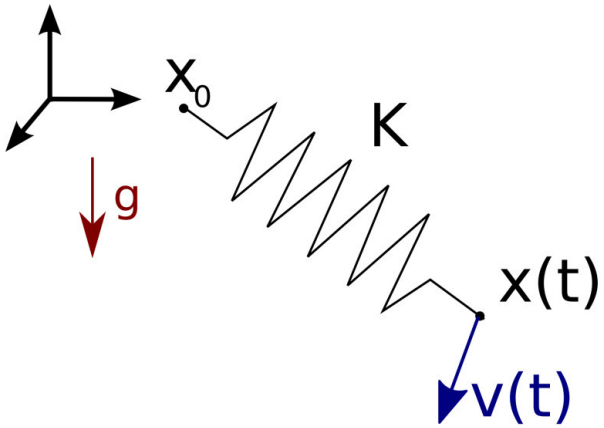
Cloth Simulation

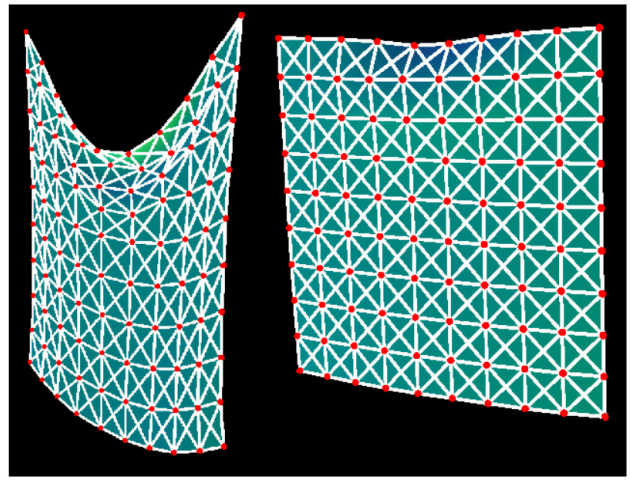


Cloth Simulation

In 3D:

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





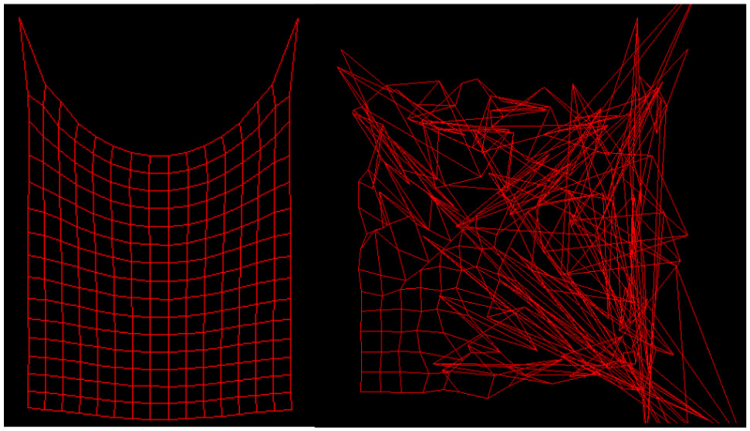
Cloth Simulation: mass springs

Model cloth using coupled springs
Force on a vertex i with neighbors $\mathcal{V}_i$

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

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

Use you best interpolation scheme



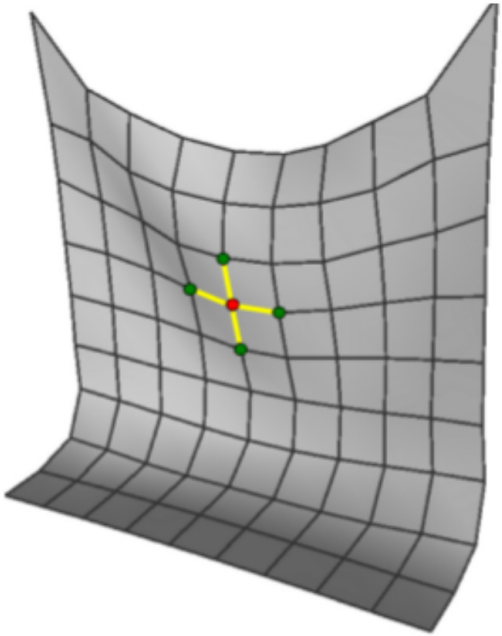
[P. Jacobs]

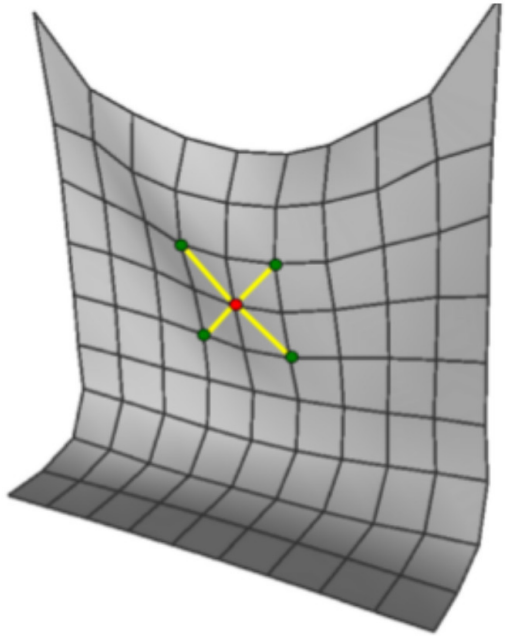
Cloth Simulation: spring types

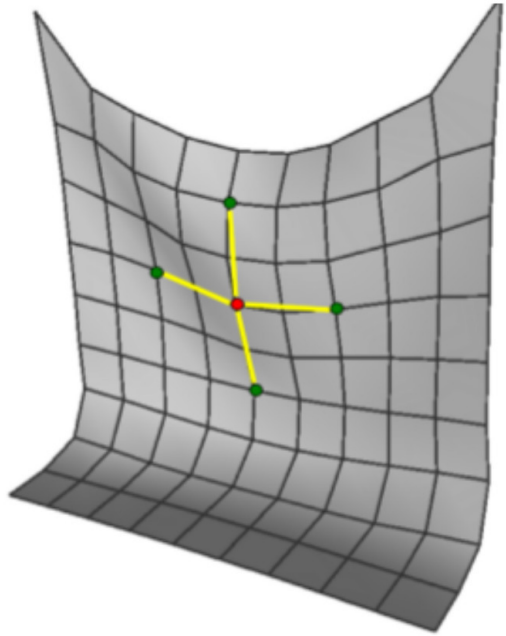
Structural springs

Shearing springs

Bending springs





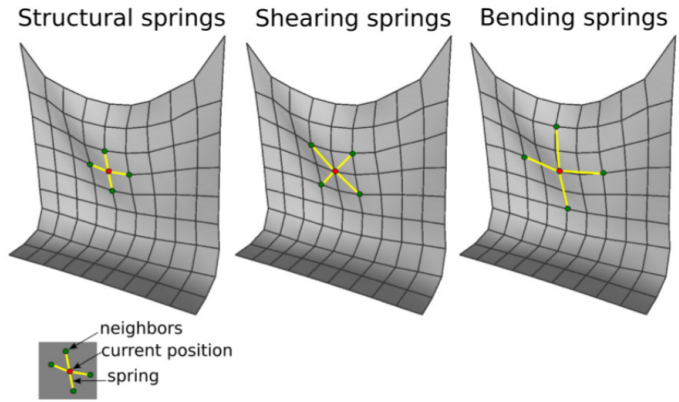


Cloth Simulation: Resolution

Explicit Euler

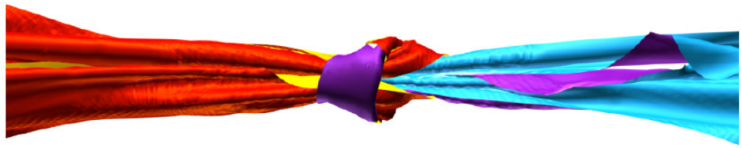
```
//Compute forces
For all i
   For j:  4 direct neighbors (structural): K=K1
           4 diagonal neighbors (shear)   : K=K2
           8 neighbors (bend)             : K=K3
      u=p[i]-p[j]
      F[i] += K (L0-norm(u))*u/norm(u)

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



Complex cloth simulation

Full simulation = Cloth + complex collisions

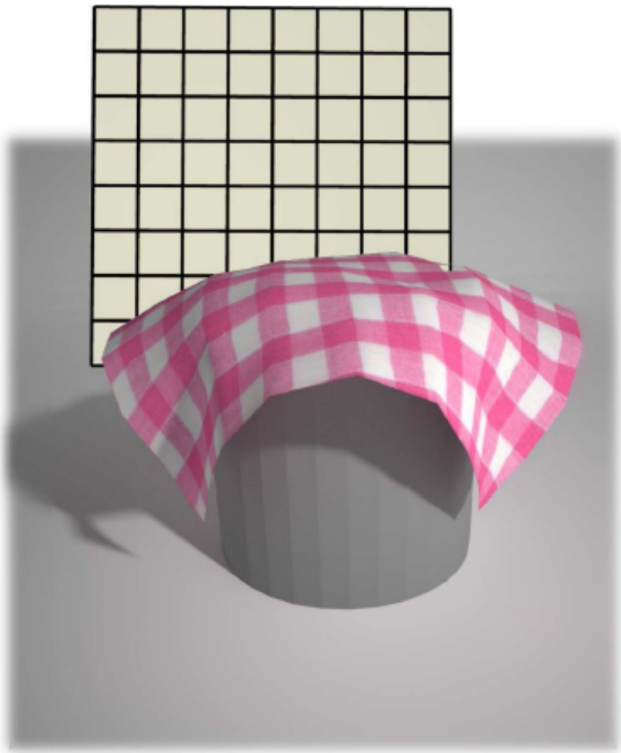


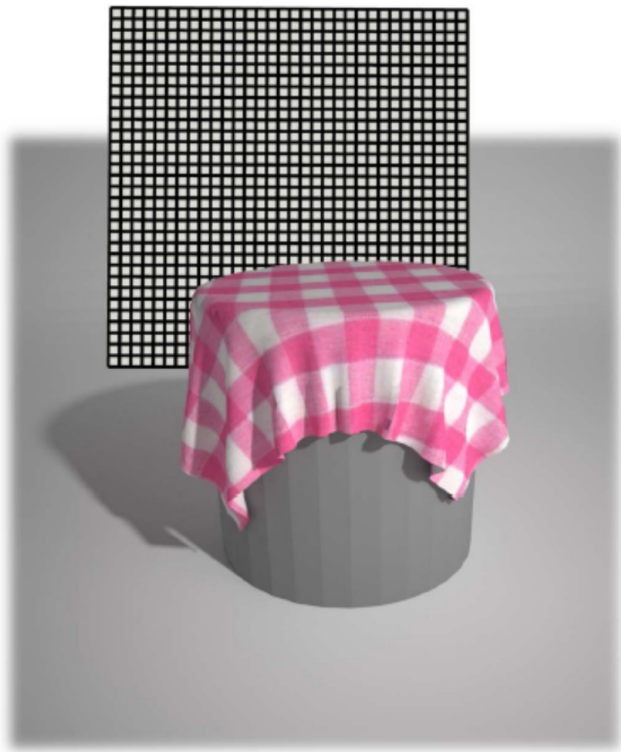
[Grinspun, SIGGRAPH 09]

[DAZ3D, Dynamic Clothing]

Cloth simulation: Limitations

Mesh influence





Implicit scheme

Write the system in a vectorial form

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

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

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

Get a linear system.
Loss of the inconditional stability:
In practice, very stable

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

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

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

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

Collisions

Particles collisions

Particles can enter in collision with a plane:

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

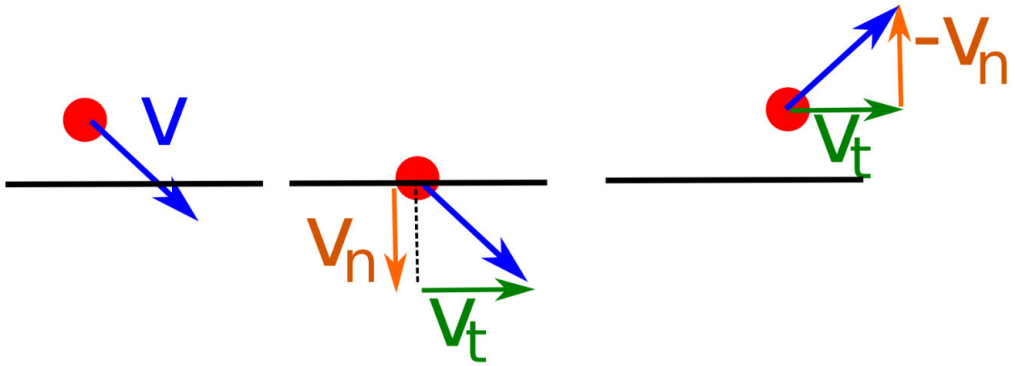
Detection $< \mathbf{x} - \mathbf{p}_0, \mathbf{n} > < 0$

Separate tangential speed v_t and and normal speed v_n

$$\begin{cases} v_n =< v, n > n \\ v_t = v - < v, n > n \end{cases}$$

After collision, loss of energy: dumping

$$v^{k+1} = -\mu < v^k, n > n + (v^k - < v^k, n > n)$$



Particles collisions

Need to take care of the temporal discretization

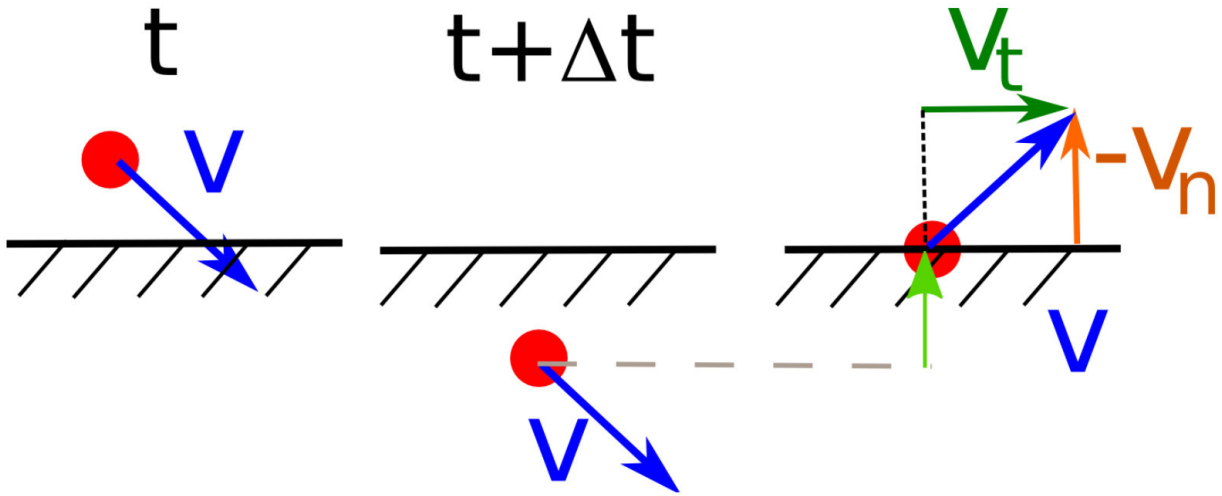
1. Projection of the particle

Easy to compute, physically biased (instabilities)

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

2. Backward search

Less easy, less wrong

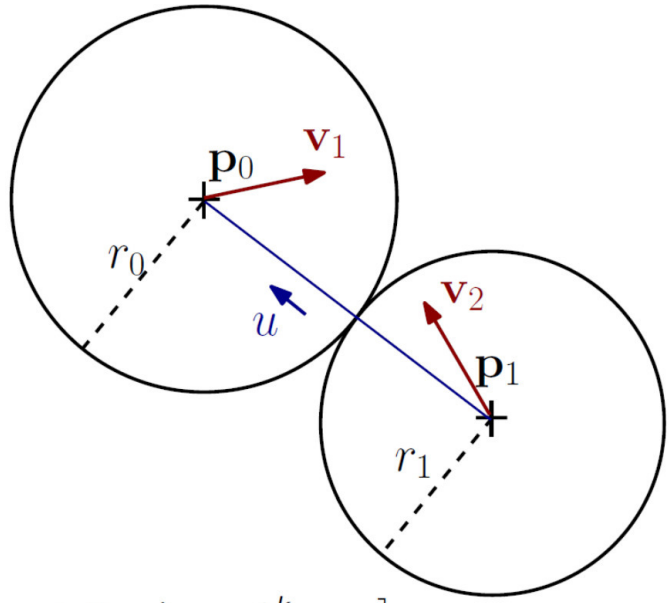


Model of hard spheres

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

Collision if

$$\|x_1 - x_2\| < r_1 + r_2$$



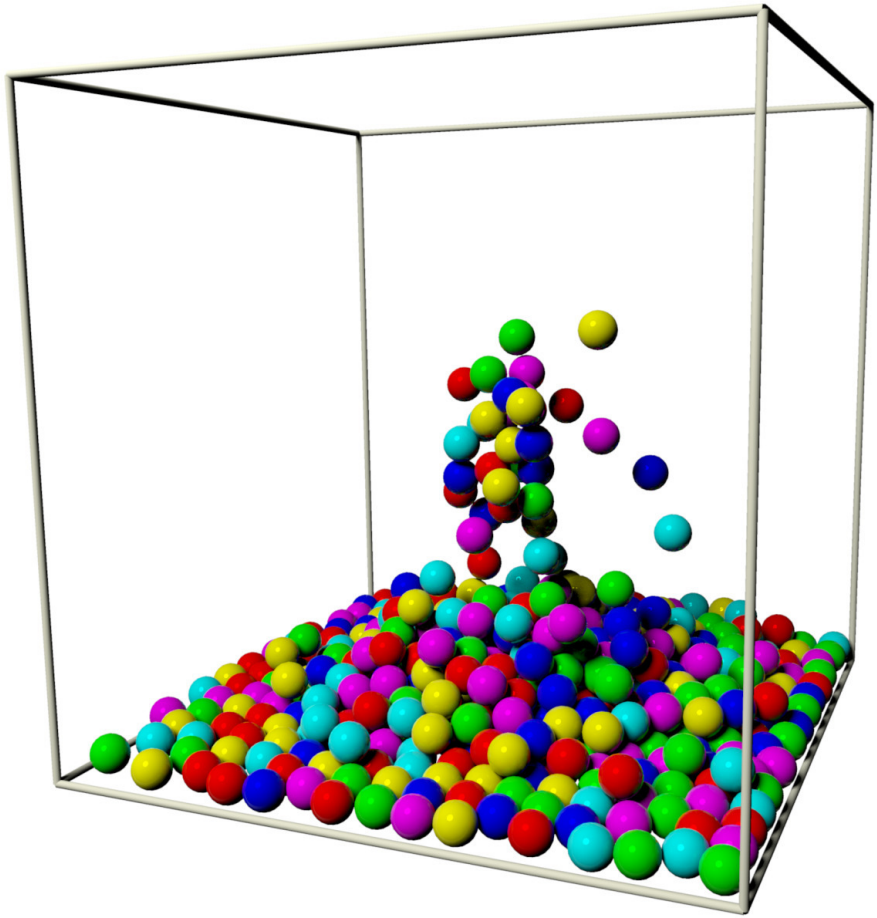
New speed after collision

$$\begin{cases} v_1^{k+1} = v_1^k + \frac{1}{m_1+m_2}\left[m2 < v2^k, u > -\frac{1}{2}(m_1 + 3m_2) < v1^k, u >\right] u \\ v_2^{k+1} = v_2^k + \frac{1}{m_1+m_2}\left[m1 < v1^k, u > -\frac{1}{2}(m_2 + 3m_1) < v2^k, u >\right] u \\ u = x1 - x0 \end{cases}$$

Don't forget to project back to the collision surface

Model of hard spheres

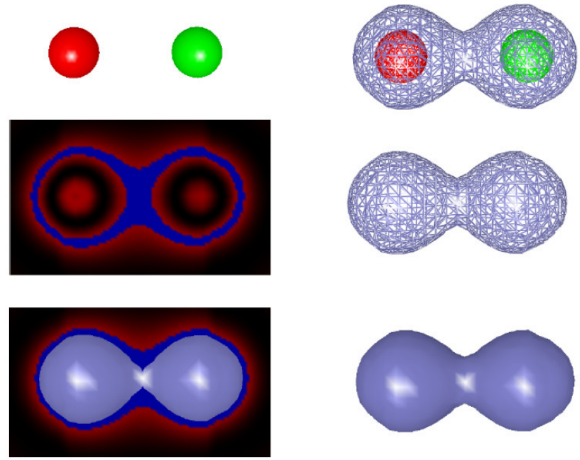
Generate a lot of spheres



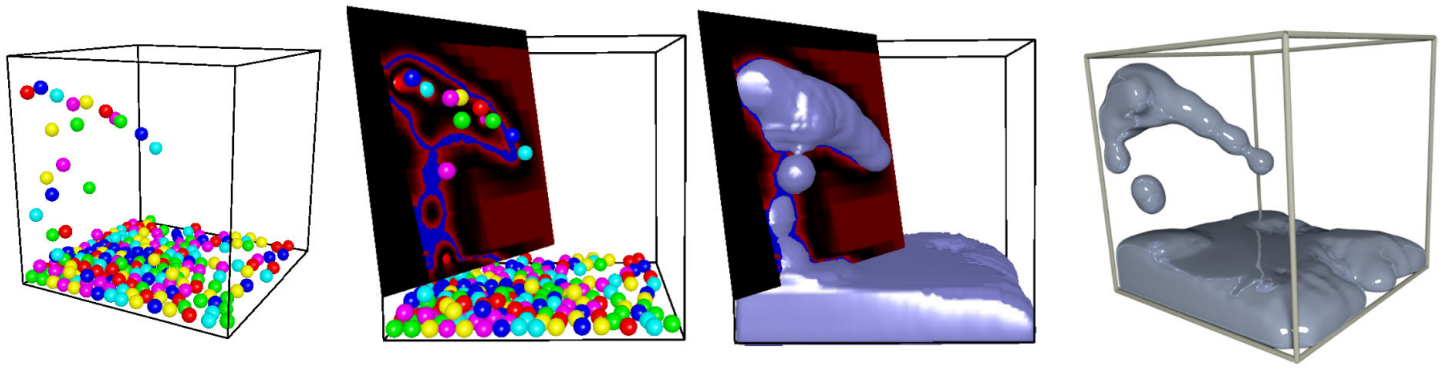
Sphere: implicit functions

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



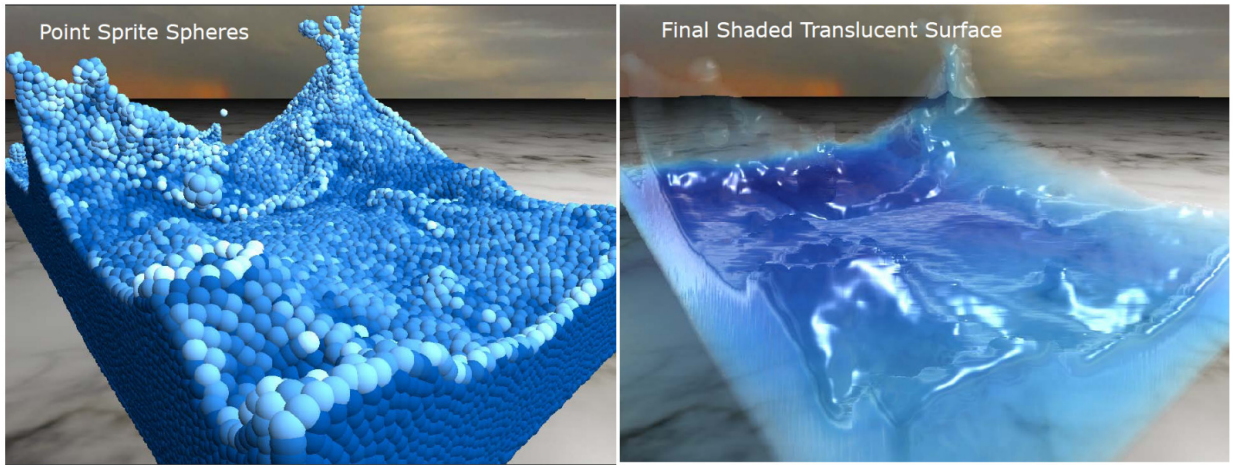
Efficient collision detection

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 $10^{3\text{-}6}$ particules in real time



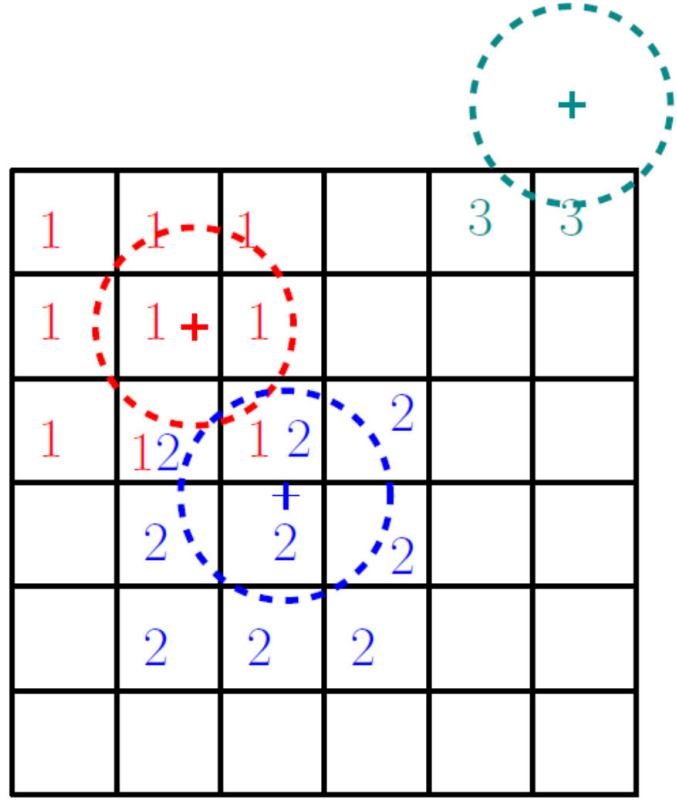
[NVIDIA, Green SIGGRAPH 10]

Acceleration structure

Regular grid
Research in O(1)

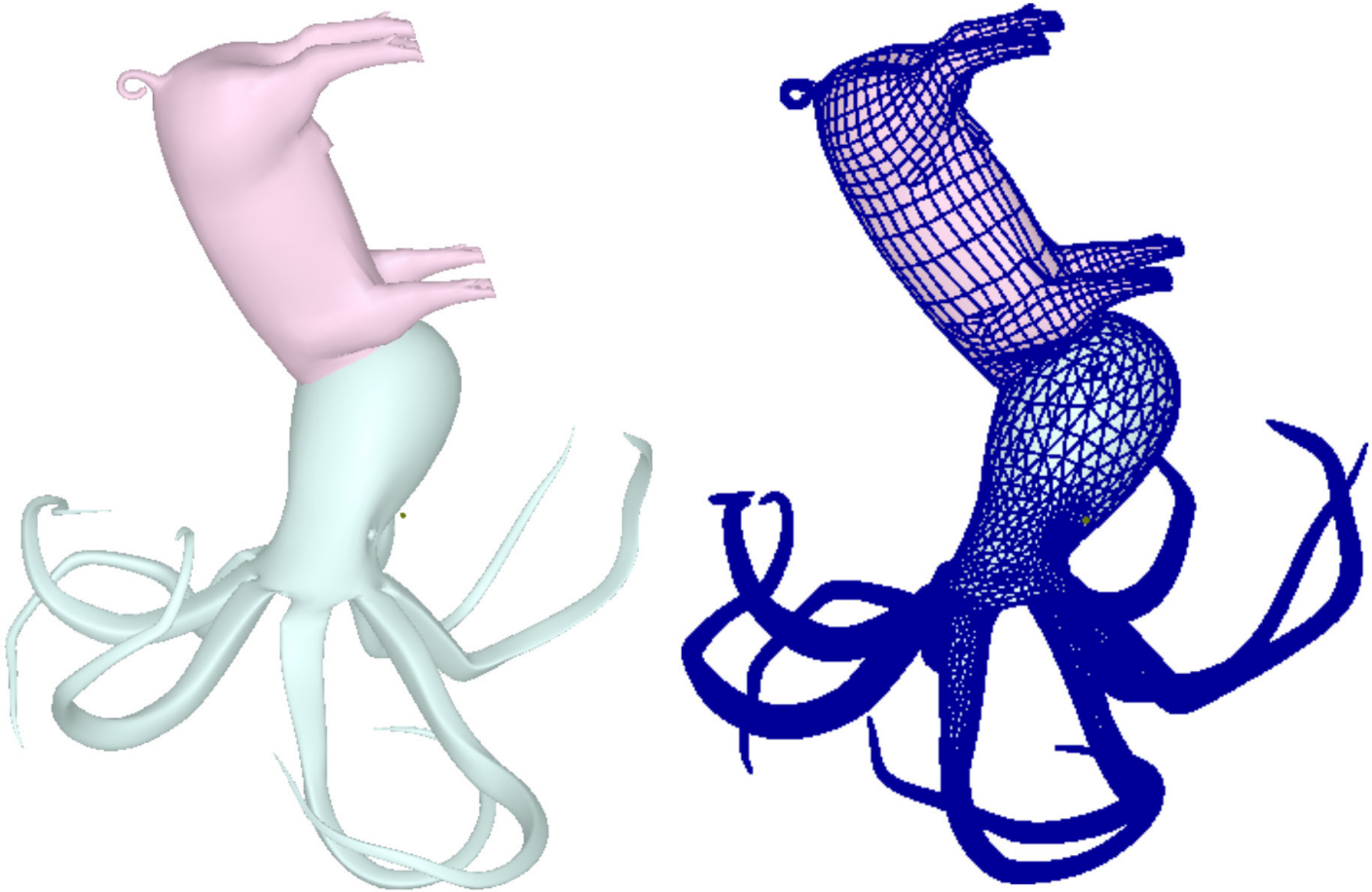
+ Simple
+ Efficient research
+ Good for uniform samples

- Memory consumption
  for empty areas



Collision detecion between objects

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



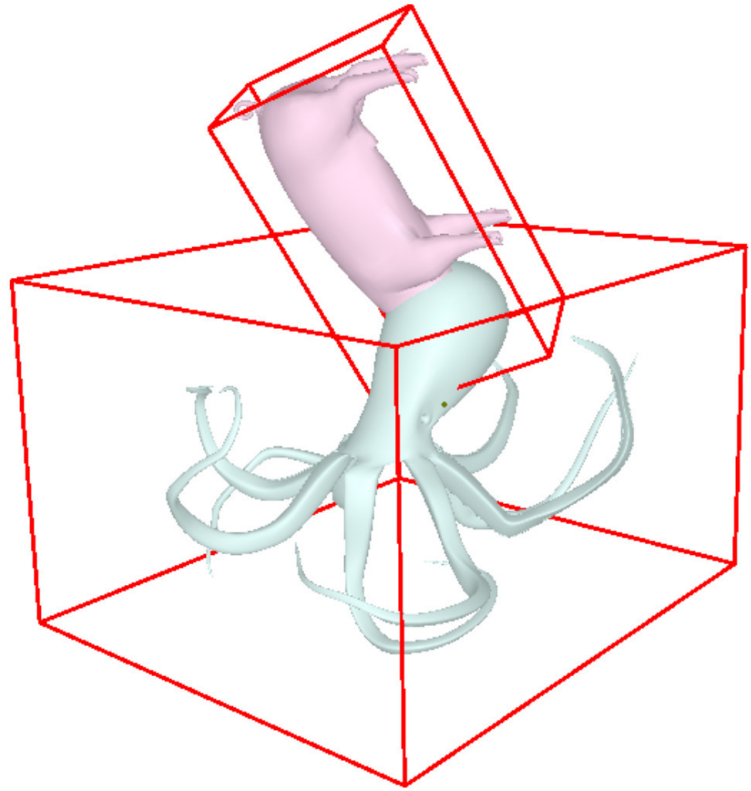
Bounding boxes

Bounding Box (BB)

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

Bounding Spheres

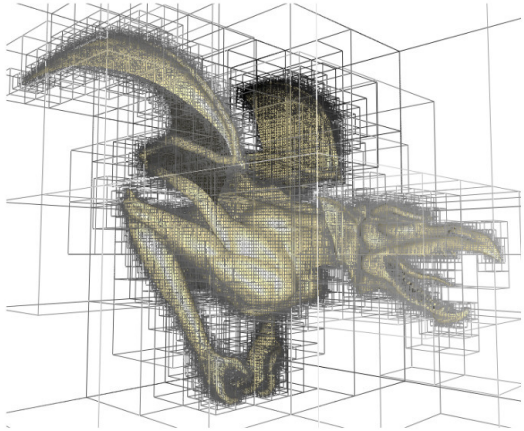
+ Detection of non-collision in $O(N^2_{obj})$



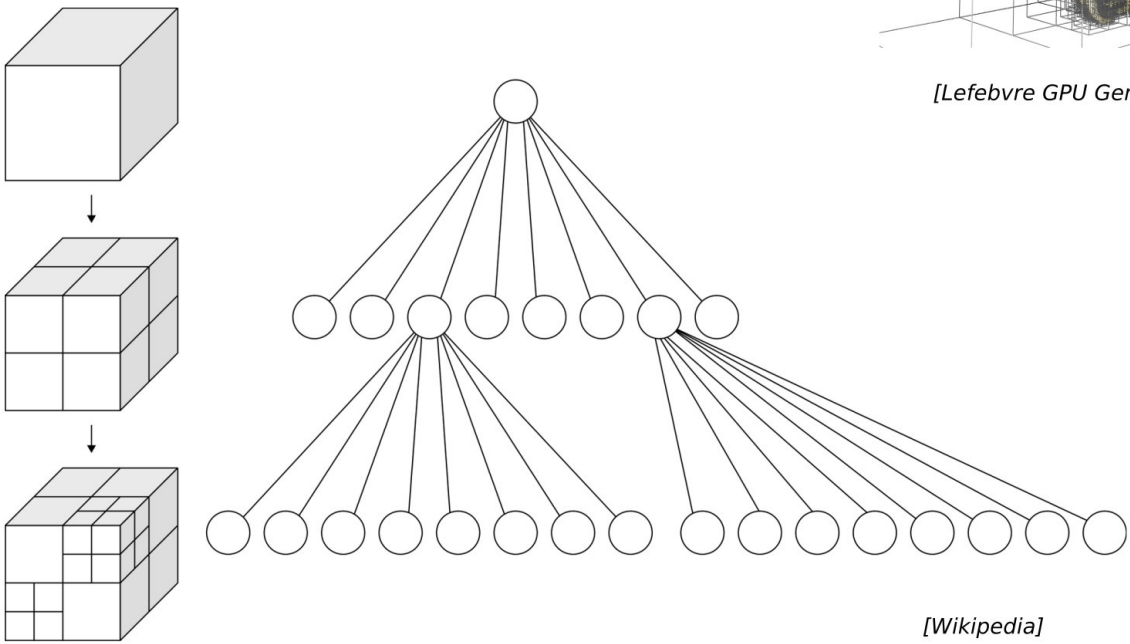
Octree

Self-adaptable grid structure

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



[Lefebvre GPU Gems 2, 2004]

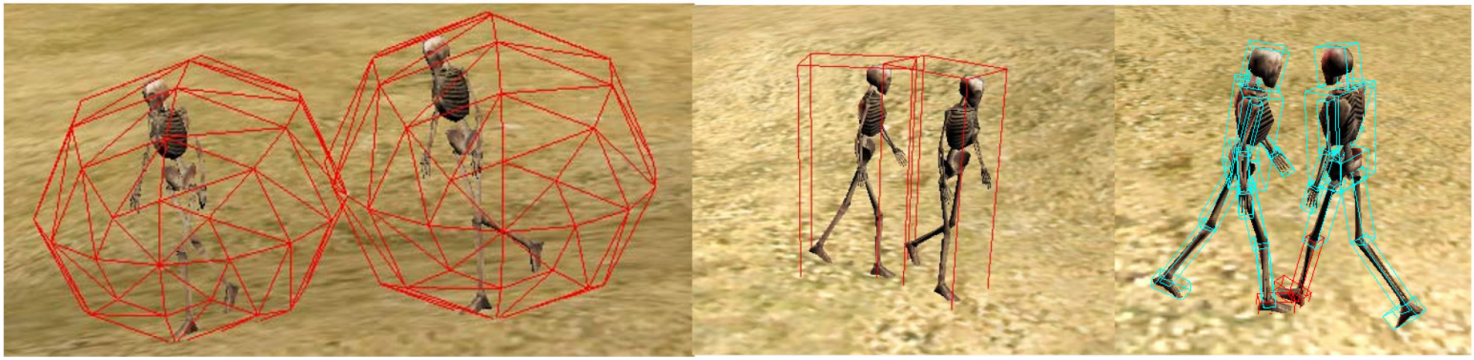


[Wikipedia]

Collisions

In practice: Several levels of details

ex.
- Spheres within BB within Octree
- OBB within AABB within Spheres



[Ditchburn]

Rough test
Less rough test
...
Fine test
...
Eventually: Compute the real intersection (rare)

=> Non collision tests much more efficient than collision