Introduction to Computer Graphics

– Animation (2) –

May 26, 2016

Kenshi Takayama

Physically-based deformations

Simple example: single mass & spring in 1D

- Mass m, position x, spring coefficient k, rest length l, gravity g :

Equation of motion

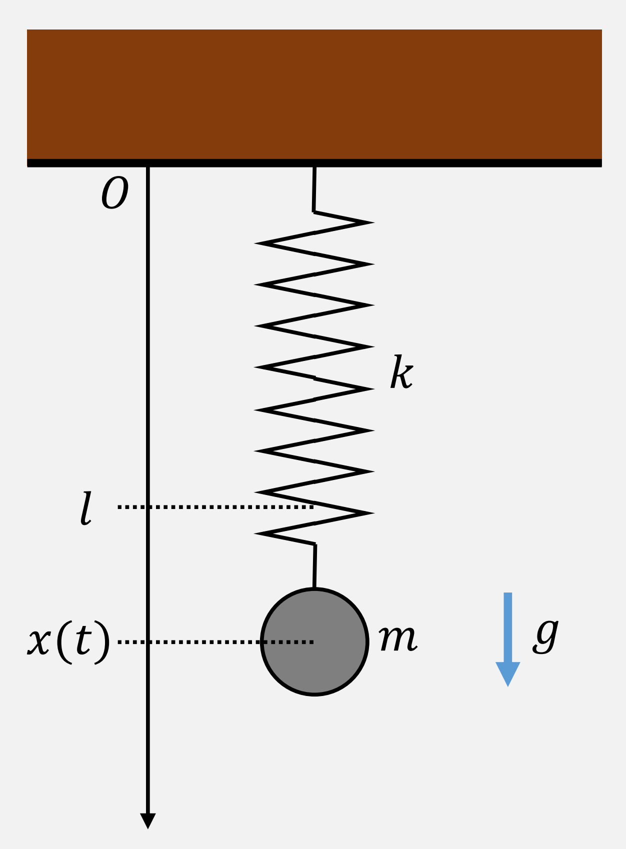
$$m\frac{d^2x}{dt^2} = -k\,(x - l) + g$$
$$= f_{\text{int}}(x) + f_{\text{ext}}$$

- f_{ext} : External force (gravity, collision, user interaction)
- $f_{\text{int}}(x)$: Internal force (pulling the system back to original)
  - Spring's internal energy (potential):

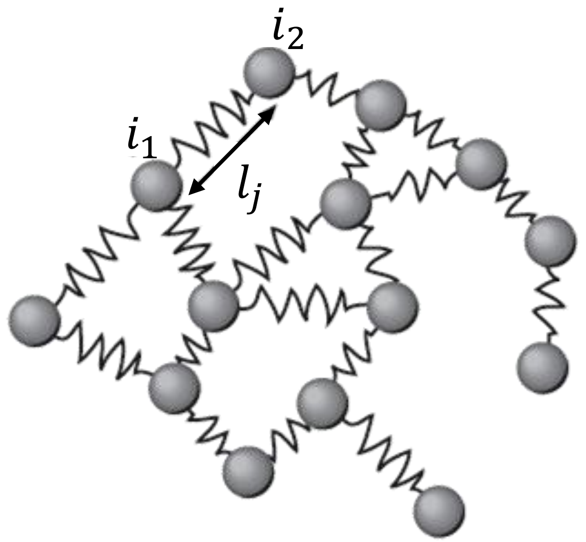
$$E(x) \coloneqq \frac{k}{2}\,(x - l)^2$$

  - Internal force is the opposite of potential gradient:

$$f_{\text{int}}(x) \coloneqq -\frac{dE}{dx} = -k(x - l)$$



Mass-spring system in 3D



- N masses: i-th mass m_i, position $x_i \in \mathbb{R}^3$
- M springs: j-th spring $e_j = (i_1, i_2)$
  - Coefficient k_j, rest length l_j

- System's potential energy for a state $\mathbf{x} = (x_1, \dots, x_N) \in \mathbb{R}^{3N}$:

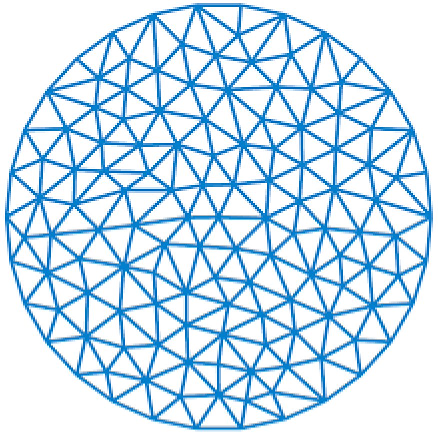
$$E(\mathbf{x}) \coloneqq \sum_{e_j=(i_1,i_2)} \frac{k_j}{2} \left(\| x_{i_1} - x_{i_2} \| - l_j \right)^2$$

- Equation of motion:

$$\mathbf{M} \frac{d^2\mathbf{x}}{dt^2} = -\boldsymbol{\nabla} E(\mathbf{x}) + \mathbf{f}_{\text{ext}}$$

  - $\mathbf{M} \in \mathbb{R}^{3N \times 3N}$: Diagonal matrix made of m_i (mass matrix)

Continuous elastic model in 2D (**F**inite **E**lement **M**ethod)

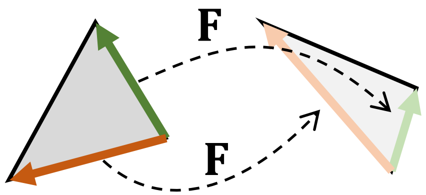


Tessellate the domain into triangular mesh

- N vertices: i-th position $x_i \in \mathbb{R}^2$
- M triangles: j-th triangle $t_j = (i_1, i_2, i_3)$

- Undeformed state: $\mathbf{X} = (X_1, \dots, X_N) \in \mathbb{R}^{2N}$
- Deformed state: $\mathbf{x} = (x_1, \dots, x_N) \in \mathbb{R}^{2N}$
- Deformation gradient:

$$\mathbf{F}_j(\mathbf{x}) \coloneqq \begin{pmatrix} x_{i_2} - x_{i_1} & x_{i_3} - x_{i_1} \end{pmatrix} \begin{pmatrix} X_{i_2} - X_{i_1} & X_{i_3} - X_{i_1} \end{pmatrix}^{-1} \in \mathbb{R}^{2\times 2}$$



Linear transformation which maps edges

- System's potential:

Area of t_j

$$E(\mathbf{x}) \coloneqq \sum_{t_j = (i_1, i_2, i_3)} \frac{A_j}{2} \left\| \mathbf{F}_j(\mathbf{x})^\top \mathbf{F}_j(\mathbf{x}) - \mathbf{I} \right\|_{\mathcal{F}}^2$$

Green's strain energy

- Equation of motion:

$$\mathbf{M} \frac{d^2\mathbf{x}}{dt^2} = -\nabla E(\mathbf{x}) + \mathbf{f}_{\text{ext}}$$

  - $\mathbf{M} \in \mathbb{R}^{2N\times 2N}$: Diagonal matrix made of vertices' Voronoi areas

Computing dynamics

- Problem: Given initial value of position $\mathbf{x}(t)$ and velocity $\mathbf{v}(t) \coloneqq \frac{d\mathbf{x}}{dt}$ as

$$\mathbf{x}(0) = \mathbf{x}_0 \quad \text{and} \quad \mathbf{v}(0) = \mathbf{v}_0 \ ,$$

compute $\mathbf{x}(t)$ and $\mathbf{v}(t)$ for $t > 0$. (Initial Value Problem)

- Simple case of single mass & spring:

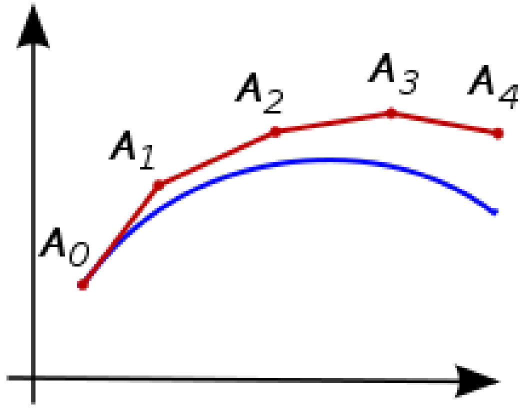
$$m\frac{d^2x}{dt^2} = -k(x - l) + g$$

➔ analytic solution exists (sine curve)

- General problems don't have analytic solution

➔ From state $(\mathbf{x}_n, \mathbf{v}_n)$ at time t, compute next state $(\mathbf{x}_{n+1}, \mathbf{v}_{n+1})$ at time $t + h$. (time integration)

  - h: time step



Simplest method: Explicit Euler

Discretize acceleration using finite difference:

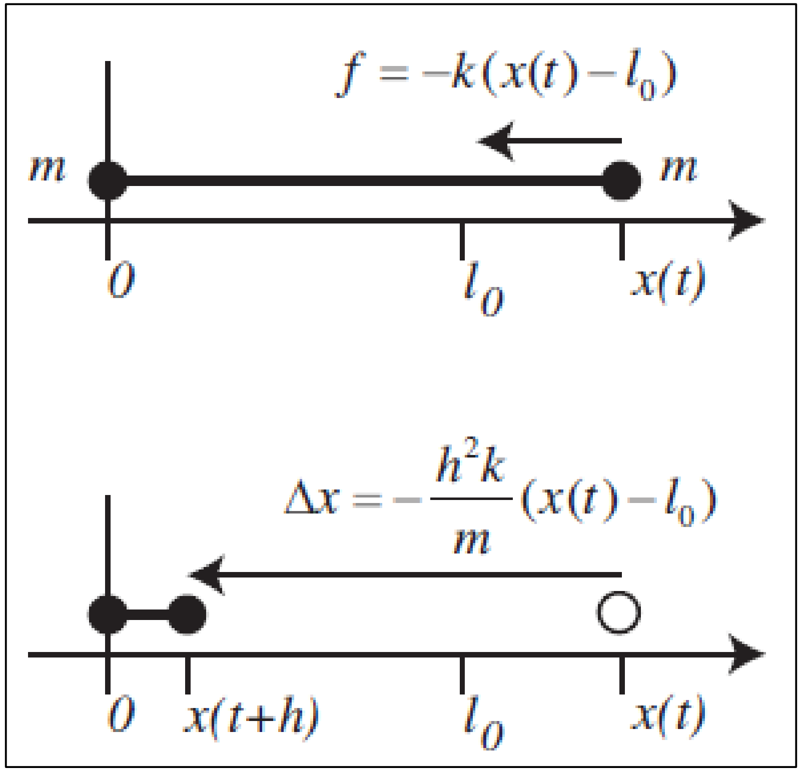
$$\mathbf{M}\frac{\mathbf{v}_{n+1}-\mathbf{v}_n}{h} = \mathbf{f}_{\text{int}}(\mathbf{x}_n) + \mathbf{f}_{\text{ext}}$$

Update velocity $\quad \mathbf{v}_{n+1} \leftarrow \mathbf{v}_n + h\,\mathbf{M}^{-1}\,(\mathbf{f}_{\text{int}}(\mathbf{x}_n) + \mathbf{f}_{\text{ext}})$

Update position $\quad \mathbf{x}_{n+1} \leftarrow \mathbf{x}_n + h\,\mathbf{v}_{n+1}$

- Pro: easy to compute

- Con: overshooting
  - With larger time steps, mass can easily go beyond the initial amplitude
    - ➔ System energy explodes over time



Method to be chosen: Implicit Euler

Find $(\mathbf{x}_{n+1}, \mathbf{v}_{n+1})$ such that:

$$\begin{cases} \mathbf{v}_{n+1} = \mathbf{v}_n \ + \ h\, \mathbf{M}^{-1} \left(\mathbf{f}_{\text{int}}(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}} \right) \\ \mathbf{x}_{n+1} = \mathbf{x}_n \ + \ h\, \mathbf{v}_{n+1} \end{cases}$$

- Represent $\mathbf{v}_{n+1}$ using unknown position $\mathbf{x}_{n+1}$

- Pros: can avoid overshoot

- Cons: expensive to compute (i.e. solve equation)

Inside of Implicit Euler

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\,\mathbf{M}^{-1}\left(\mathbf{f}_{\text{int}}(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}}\right)$$

$$\begin{aligned}
\mathbf{x}_{n+1} &= \mathbf{x}_n + h\,\mathbf{v}_{n+1} \\
&= \mathbf{x}_n + h\,\mathbf{v}_n + h^2\mathbf{M}^{-1}\left(\mathbf{f}_{\text{int}}(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}}\right) \\
&= \mathbf{x}_n + h\,\mathbf{v}_n + h^2\mathbf{M}^{-1}\left(-\nabla E(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}}\right)
\end{aligned}$$

Denote unknown $\mathbf{x}_{n+1}$ as $\mathbf{y}$

$$\mathbf{F}(\mathbf{y}) \coloneqq h^2\nabla E(\mathbf{y}) + \mathbf{M}\,\mathbf{y} - \mathbf{M}(\mathbf{x}_n + h\,\mathbf{v}_n) - h^2\mathbf{f}_{\text{ext}} = \mathbf{0}$$

- Reduce to root-finding problem of function $\mathbf{F}: \mathbb{R}^{3N} \mapsto \mathbb{R}^{3N}$
  ➔ Newton's method:

$$\begin{aligned}
\mathbf{y}^{(i+1)} &\leftarrow \mathbf{y}^{(i)} - \left(\frac{d\mathbf{F}}{d\mathbf{y}}\right)^{-1} \mathbf{F}(\mathbf{y}^{(i)}) \\
&= \mathbf{y}^{(i)} - \left(h^2\mathcal{H}_E\left(\mathbf{y}^{(i)}\right) + \mathbf{M}\right)^{-1} \mathbf{F}(\mathbf{y}^{(i)})
\end{aligned}$$

2nd derivative of potential E (Hessian matrix)

- Coefficient matrix of large linear system changes at every iteration
  ➔ high computational cost!

Mass-spring model vs continuous model (FEM)

- Both:
  - System potential defined as the sum of deformation energy of small elements
  - Implicit Euler needed for both

- Mass-spring:
  - Inappropriate for modeling objects occupying continuous 2D/3D domains
  - Effective for modeling web-/mesh-like materials

- FEM:
  - Higher computational cost in general
    - Good domain tessellation
    - Complex potential energy
  - Can handle various (nonlinear) materials

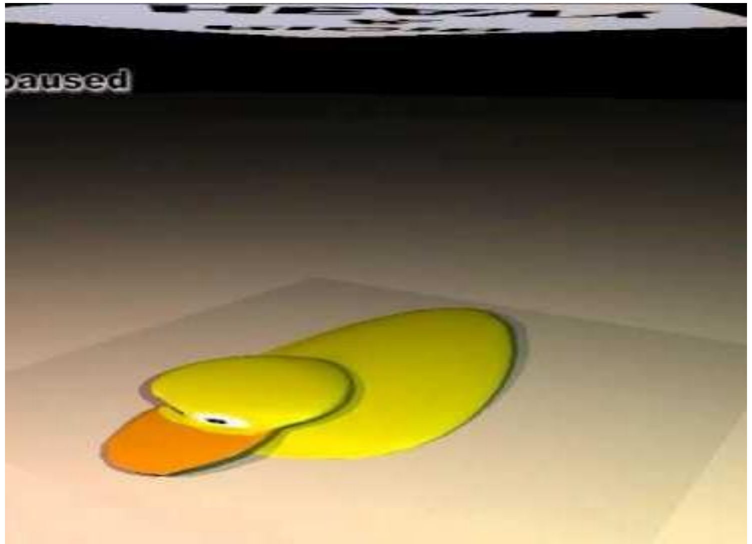
	Mass-spring	FEM
Physical accuracy	△	○
Impl. / comput. cost	○	△

Position-**B**ased **D**ynamics

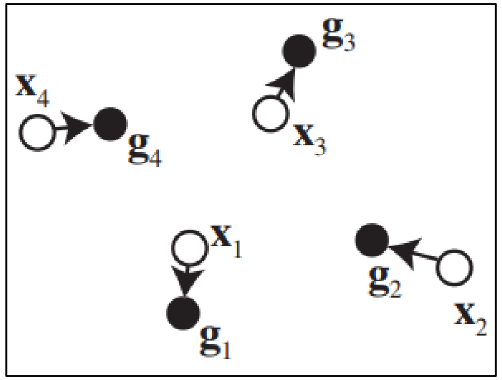
PBD: Physics-based animation framework specialized for CG

- First PBD papers:
  - Meshless deformations based on shape matching
    [Müller et al., SIGGRAPH 2005]
  - Position Based Dynamics
    [Müller et al.,VRIPhys 2006]
- Basic idea

  Compute positions making potential zero (goal position), then pull particles toward them

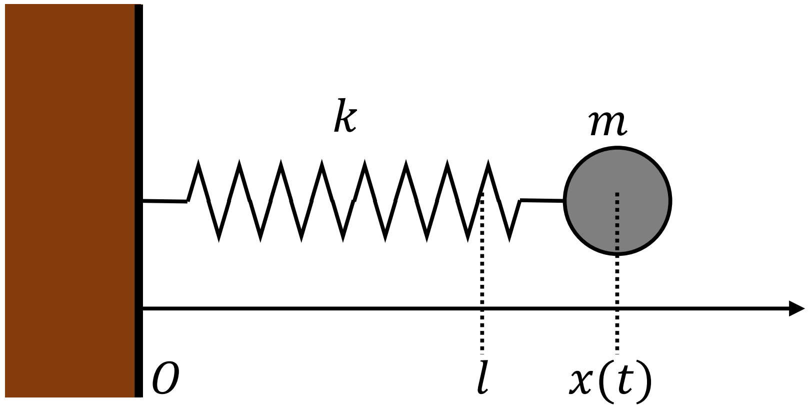


https://www.youtube.com/watch?v=CCIwiC37kks



- System energy always decreases (never explodes)
- Easy to compute ➔ perfect for games!
- Not physically meaningful computation (e.g. FEM)
  - OK for CG purposes

Case of single mass & spring (no ext. force)



Explicit Euler

$$v_{n+1} \leftarrow v_n + \frac{h\,k}{m}(l - x_n)$$
$$x_{n+1} \leftarrow x_n + h\, v_{n+1}$$

Position-Based Dynamics

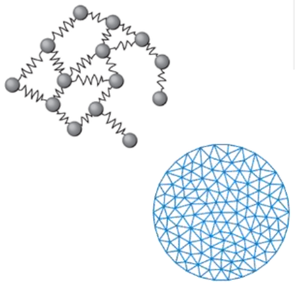
$$v_{n+1} \leftarrow v_n + \frac{\alpha}{h}(l - x_n)$$
$$x_{n+1} \leftarrow x_n + h\, v_{n+1}$$

- $0 \leq \alpha \leq 1$ is "stiffness" parameter unique to PBD
  - $\alpha = 0$ ➔ No update of velocity (spring is infinitely soft)
  - $\alpha = 1$ ➔ Spring is infinitely stiff (?)

  ➔ Energy never explodes in any case ☺

- Note: Unit of α/h is $(\text{time})^{-1}$ ➔ α has no physical meaning!
  - Reason why PBD is called non physics-based but geometry-based

Case of general deforming shape (no ext. force)



Explicit Euler

$$\mathbf{v}_{n+1} \leftarrow \mathbf{v}_n - h\,\mathbf{M}^{-1}\nabla E(\mathbf{x}_n)$$

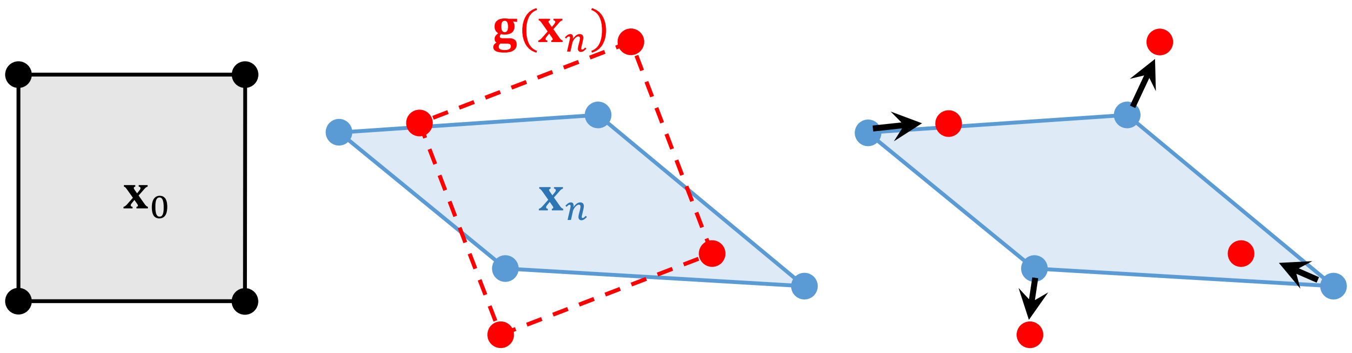
$$\mathbf{x}_{n+1} \leftarrow \mathbf{x}_n + h\,\mathbf{v}_{n+1}$$

Position-Based Dynamics

$$\mathbf{v}_{n+1} \leftarrow \mathbf{v}_n + \frac{\alpha}{h}\left(\mathbf{g}(\mathbf{x}_n) - \mathbf{x}_n\right)$$

$$\mathbf{x}_{n+1} \leftarrow \mathbf{x}_n + h\,\mathbf{v}_{n+1}$$

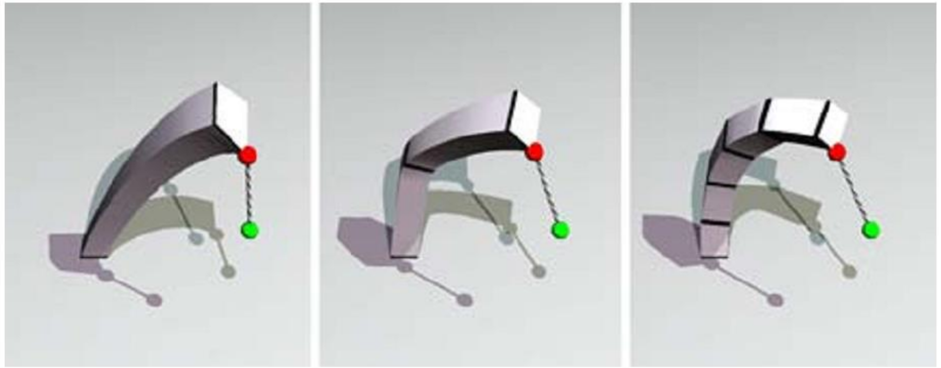
- Goal position $\mathbf{g}$
  - Rest shape rigidly transformed such that it best matches the current deformed state
    - (SVD of moment matrix)

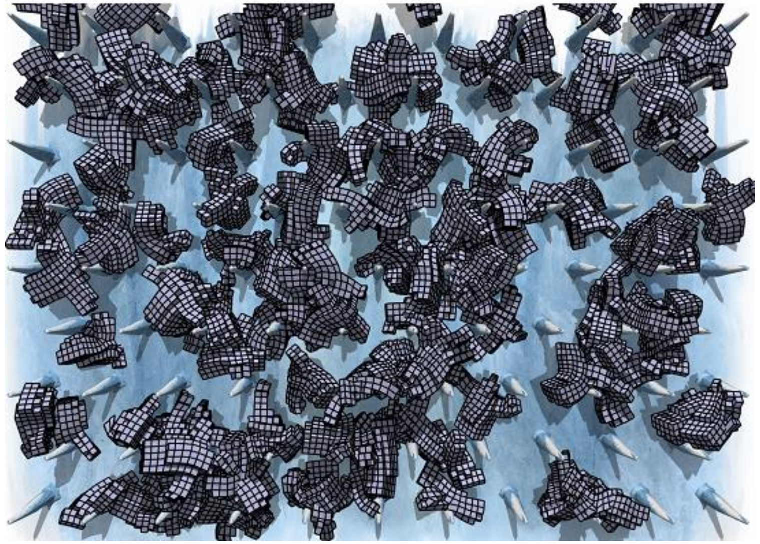


- Called “Shape Matching”
  - One technique within the PBD framework
  - Connectivity info (spring/mesh) not needed ➔ *meshless*

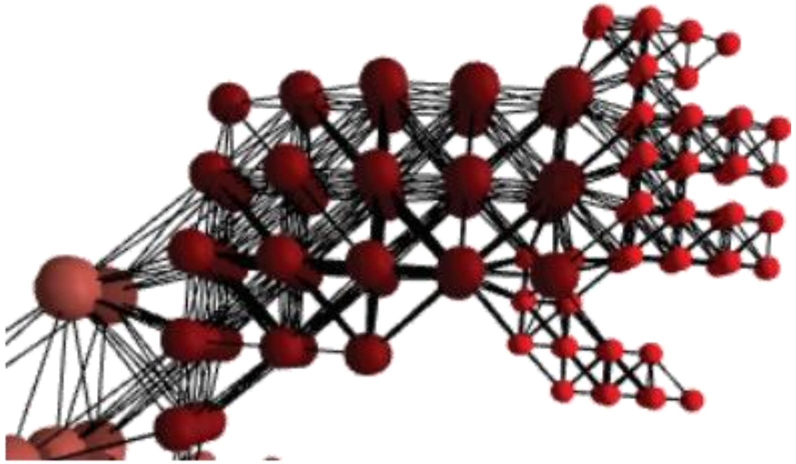
Shape Matching per (overlapping) local region

- More complex deformations
- Acceleration techniques

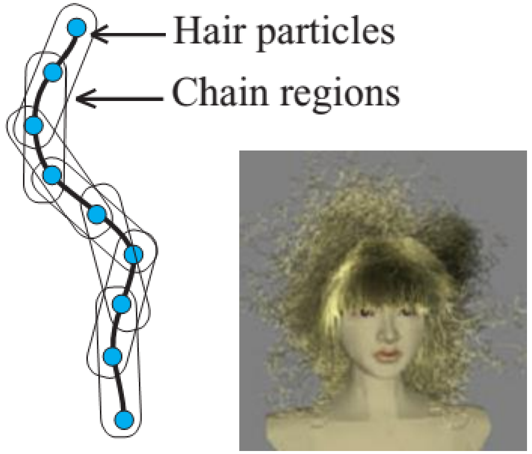




Local regions from voxel lattice



Local regions from octree

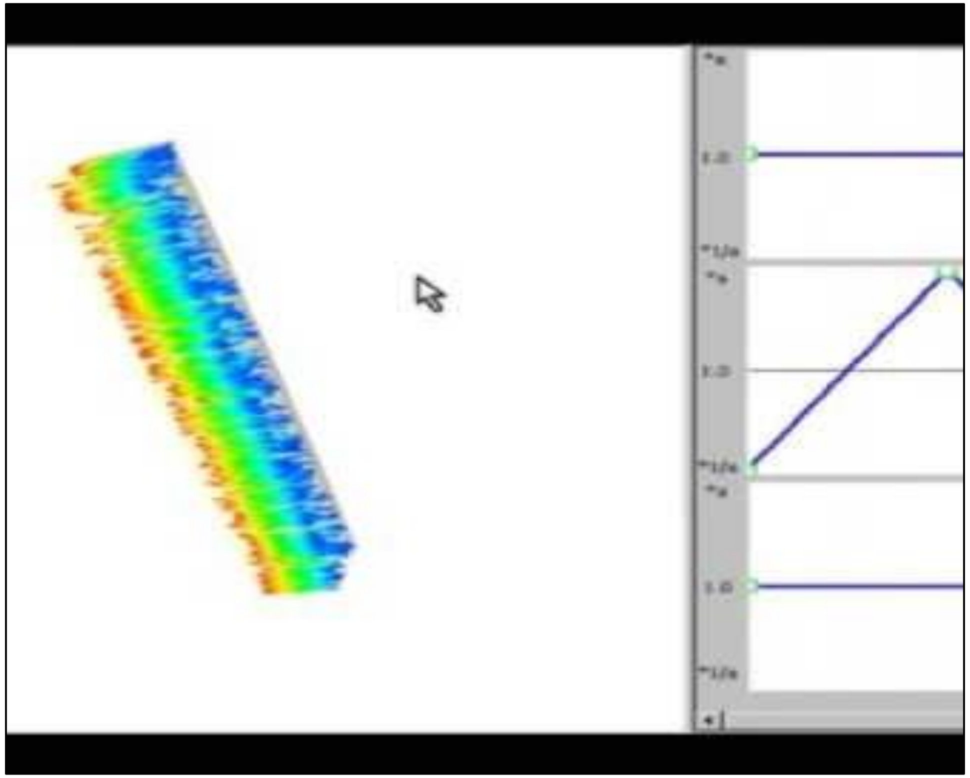


Animating hair using
1D chain structure

FastLSM; fast lattice shape matching for robust real-time deformation [Rivers SIGGRAPH07]
Fast adaptive shape matching deformations [Steinemann SCA08]
Chain Shape Matching for Simulating Complex Hairstyles [Rungjiratananon CGF10]

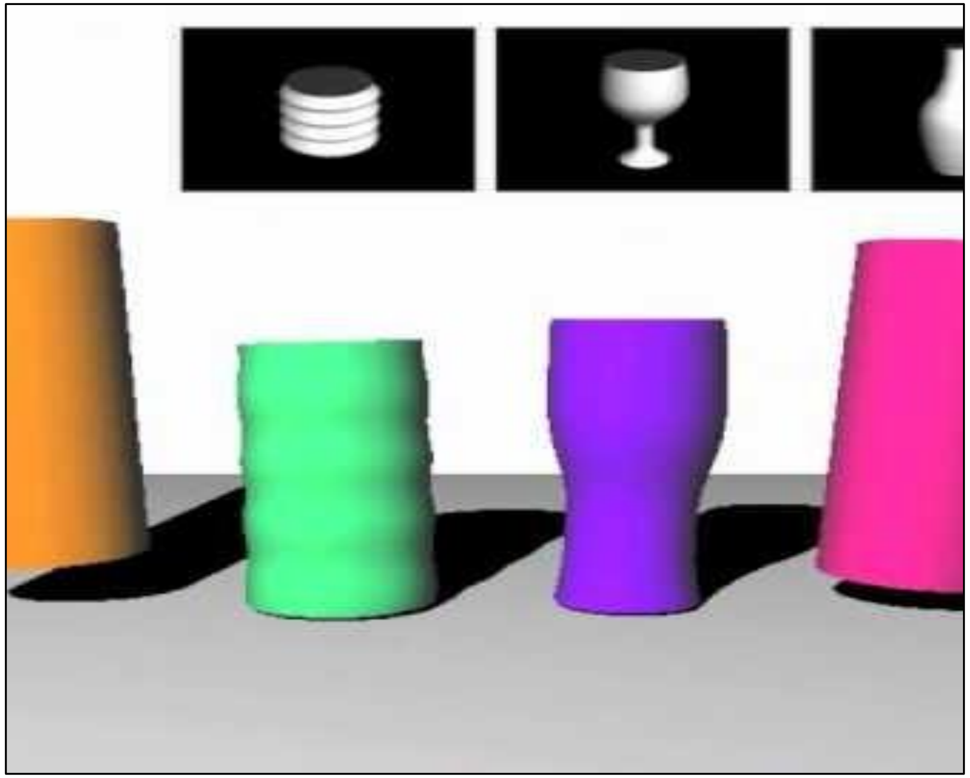
Extension: Deform rest shapes of local regions

Autonomous motion of soft bodies



https://www.youtube.com/watch?v=0AWtQbVBi3s

Example-based deformations



https://www.youtube.com/watch?v=45QjojWiOEc

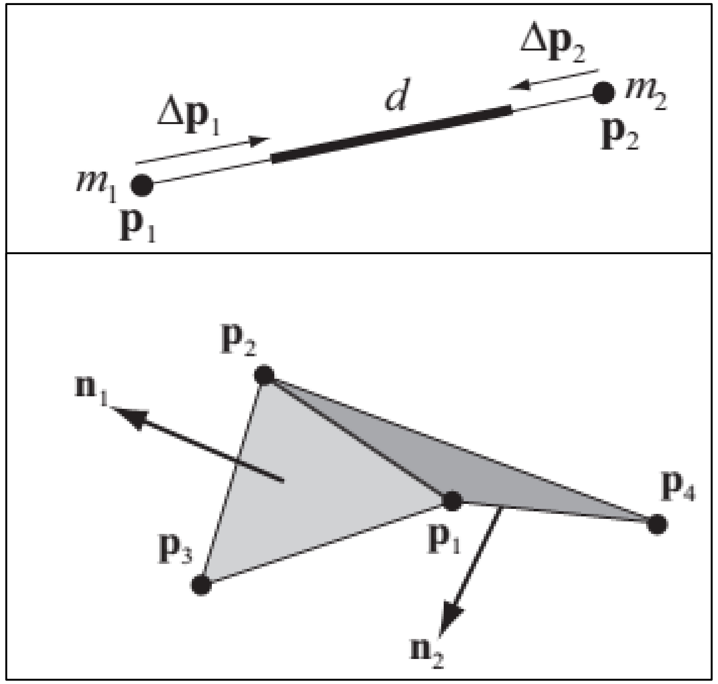
ProcDef; local-to-global deformation for skeleton-free character animation [Ijiri PG09]
Real-Time Example-Based Elastic Deformation [Koyama SCA12]

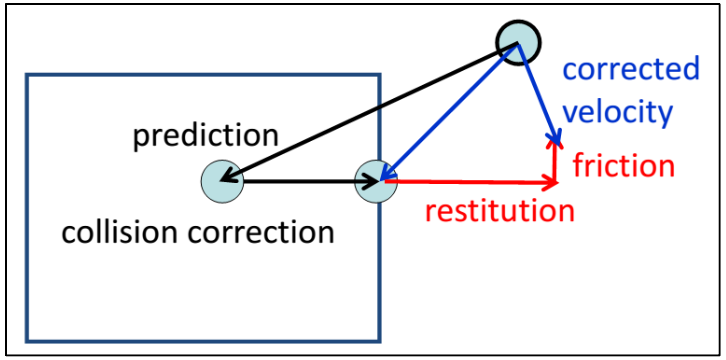
General procedure of PBD

- Input: initial position $\mathbf{x}_0$ & velocity $\mathbf{v}_0$

- At every frame:

$\mathbf{p}$	$= \mathbf{x}_n + h\,\mathbf{v}_n$	prediction
$\mathbf{x}_{n+1}$	$= \mathrm{modify}(\mathbf{p})$	position correction
$\mathbf{u}$	$= (\mathbf{x}_{n+1} - \mathbf{x}_n)/h$	velocity update
$\mathbf{v}_{n+1}$	$= \mathrm{modify}(\mathbf{u})$	velocity correction

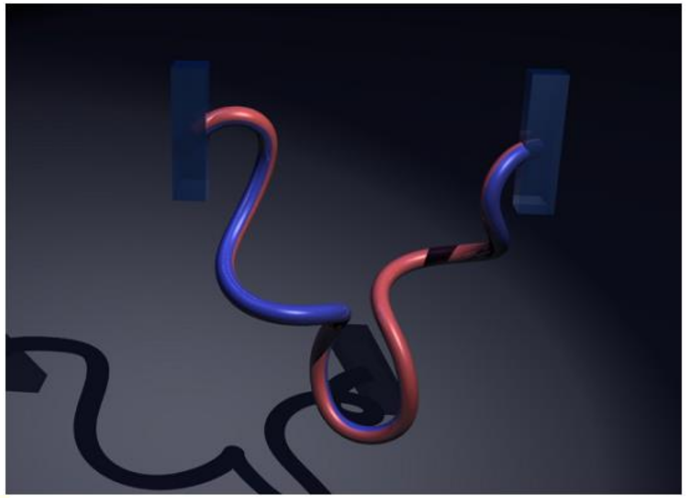




(My understanding is still weak)

Position Based Dynamics [Müller et al., VRIPhys 2006]
http://www.csee.umbc.edu/csee/research/vangogh/I3D2015/matthias_muller_slides.pdf

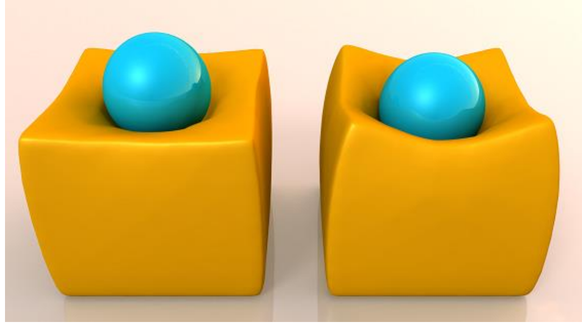
Various geometric constraints available in PBD (other than Shape Matching)



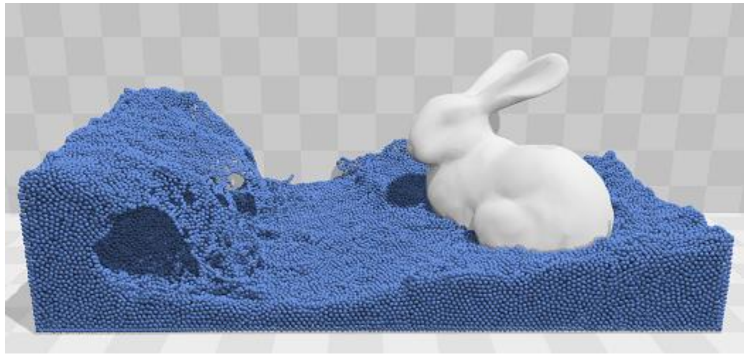
Twist constraint

Volume constraint

Stretch constraint



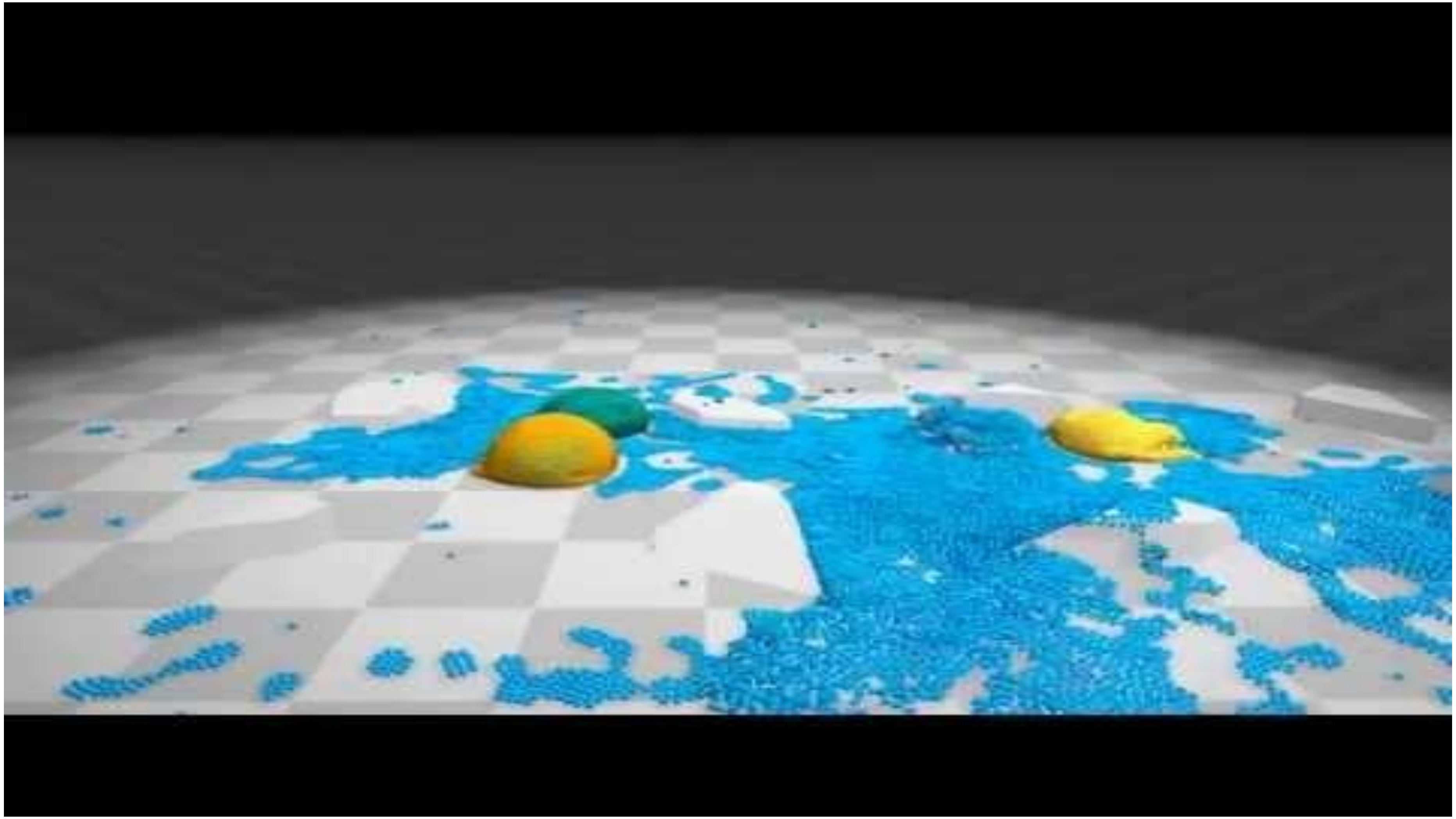
Strain constraint



Density constraint

Robust Real-Time Deformation of Incompressible Surface Meshes [Diziol SCA11]
Long Range Attachments - A Method to Simulate Inextensible Clothing in Computer Games [Kim SCA12]
Position Based Fluids [Macklin SIGGRAPH13]
Position-based Elastic Rods [Umetani SCA14]
Position-Based Simulation of Continuous Materials [Bender Comput&Graph14]

Putting everything together: FLEX in PhysX



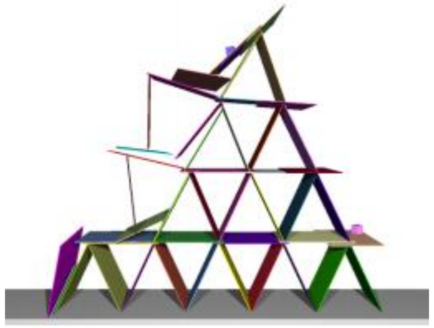
https://www.youtube.com/watch?v=z6dAahLUbZg

- SDK released by NVIDIA!

Unified Particle Physics for Real-Time Applications [Macklin SIGGRAPH14]

Collisions

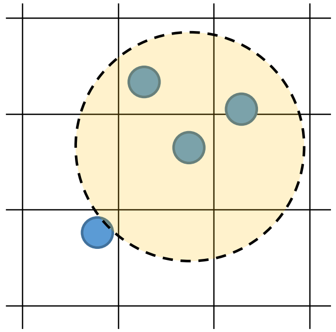
- Another tricky issue
- Popular methods in PBD:
  - For each voxel grid, record which particles it contains
  - Test collisions only among nearby particles
- Recent method specialized for PBD



[Kaufman08]

[Harmon09]

[Zheng12]



[Muller15]

Collision detection for deformable objects [Teschner CGF05]
Staggered Projections for Frictional Contact in Multibody Systems [Kaufman SIGGRAPHAsia08]
Asynchronous Contact Mechanics [Harmon SIGGRAPH09]
Energy-based Self-Collision Culling for Arbitrary Mesh Deformations [Zheng SIGGRAPH12]
Air Meshes for Robust Collision Handling [Muller SIGGRAPH15]

Pointers

- Surveys, tutorials
  - A Survey on Position-Based Simulation Methods in Computer Graphics [Bender CGF14]
  - http://www.csee.umbc.edu/csee/research/vangogh/I3D2015/matthias_muller_slides.pdf
  - Position-Based Simulation Methods in Computer Graphics [Bender EG15Tutorial]

- Libraries, implementations
  - https://code.google.com/p/opencloth/
  - http://shapeop.org/
  - http://matthias-mueller-fischer.ch/demos/matching2dSource.zip
  - https://bitbucket.org/yukikoyama
  - https://developer.nvidia.com/physx-flex
  - https://github.com/janbender/PositionBasedDynamics