EUROGRAPHICS 2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

T6: Position-Based Simulation Methods in Computer Graphics

Jan Bender Miles Macklin Matthias Müller

RWTH AACHEN UNIVERSITY

NVIDIA

Jan Bender

- Organizer
- Professor at the Visual Computing Institute at Aachen University
- Research topics
  - Rigid bodies, deformable solids, fluids
  - Collision detection, fracture, real-time visualization
  - Position based methods
- Maintains open source PBD code base
  - github.com/InteractiveComputerGraphics/PositionBasedDynamics

Miles Macklin

- Principal engineer at NVIDIA
- Inventor and author of FLEX
  - Unified, particle based, position based solver, GPU accelerated
  - UE4 integration
  - developer.nvidia.com/flex
- Research
  - Position based fluids
  - Inventor of XPBD, making PBD truly physical with a simple trick!

Matthias Müller

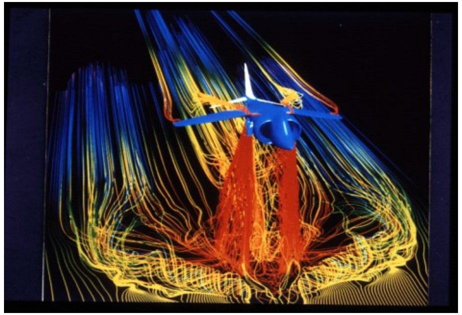
- Leader of physics research group at NVIDIA
- Co-initiator of PBD (with Thomas Jakobsen)
- Co-founder of NovodeX which became physics group at NVIDIA
- Research
  - Co-rotational FEM, SPH
  - Position based methods: cloth, soft bodies, shape matching, oriented particles, air meshes
- www.matthiasmueller.info

Tutorial Outline

- Matthias
  - Motivation, Basic Idea
  - The solver
  - Constraint examples for solids
  - Solver accelerations
- Miles
  - Fluids
  - XPBD
  - Continuous materials
  - Rigid bodies

Motivation

Physical Simulations



- Well studied problem
  in the computational sciences (since 1940s)
- Complement / replace real experiments
- Extreme conditions, spatial scale, time scale
- Accuracy most important factor
- Low accuracy – useless result

Computer Graphics

- Early 1980s
- Adopted methods: FEM, SPH, grid based fluids, ..
- Applications
  - Special effects in movies and commercials
  - Computer games
  - VR
- Requirements
  - Speed, stability, controllability
  - Only visual plausibility
- New methods needed: e.g. PBD

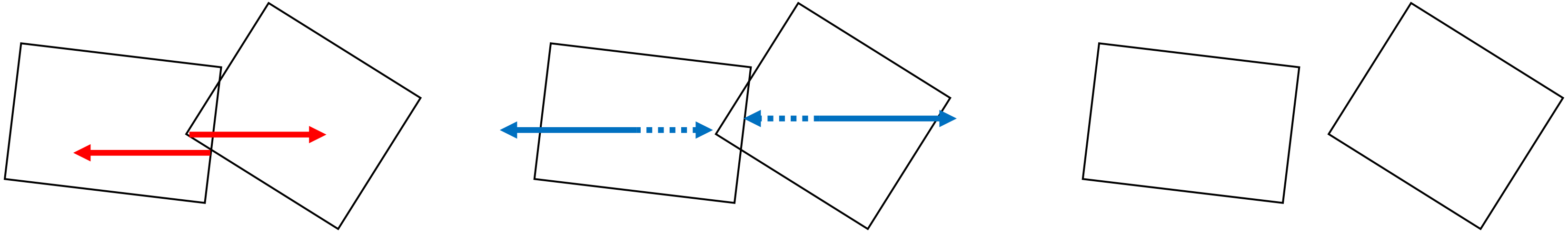
Funhouse

Traditional Methods

- Typically force based
- Explicit integration
  - Simple and fast
  - Only conditionally stable (bad for real time apps)
- Implicit integration
  - Expensive (multiple linearizations and solves per time step)
  - Numerical damping

Basic Idea

Force Based Update



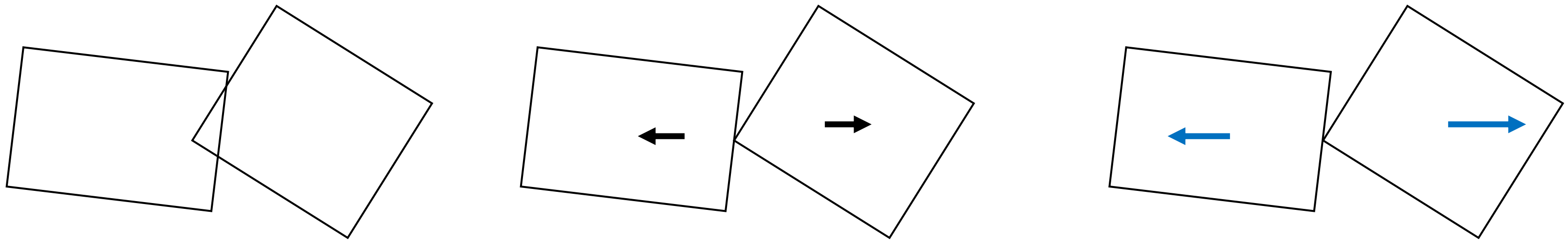
penetration causes forces

forces change velocities

velocities change positions

- Reaction lag
- Small spring stiffness → squashy system
- Large spring stiffness → stiff system, overshooting

Position Based Update



penetration detection only

move objects so that they do not penetrate

update velocities!

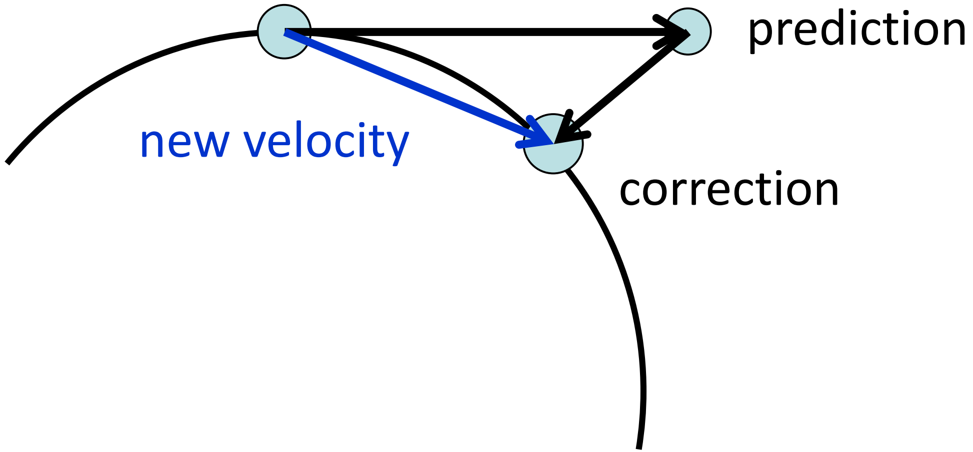
- Controlled position change
- Only as much as needed → no overshooting
- Velocity update needed to get 2nd order system!

Position Based Integration

init $\mathbf{x}_0, \mathbf{v}_0$			$\mathbf{x}_n, \mathbf{v}_n, \mathbf{p}, \mathbf{u} \in \mathbb{R}^{3N}$
loop			
$\mathbf{v}_n$	$\leftarrow \mathbf{v}_n + \Delta t \cdot \mathbf{f}_{ext}(\mathbf{x}_n)$		velocity update
$\mathbf{p}$	$\leftarrow \mathbf{x}_n + \Delta t \cdot \mathbf{v}_n$		prediction
$\mathbf{x}_{n+1}$	$\leftarrow$ modify $\mathbf{p}$		position correction
$\mathbf{u}$	$\leftarrow (\mathbf{x}_{n+1} - \mathbf{x}_n)/\Delta t$		velocity update
$\mathbf{v}_{n+1}$	$\leftarrow$ modify $\mathbf{u}$		velocity correction
end loop			

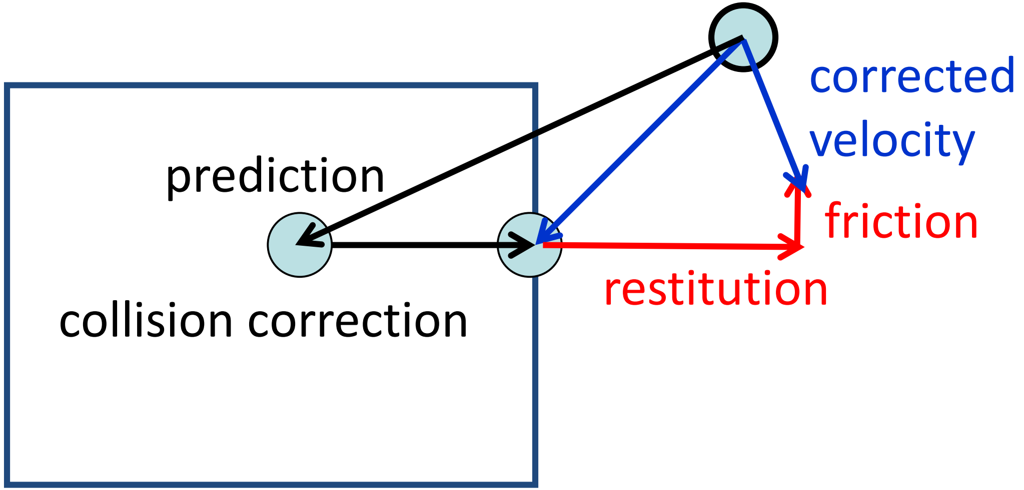
Position Correction

- Example: Particle on circle

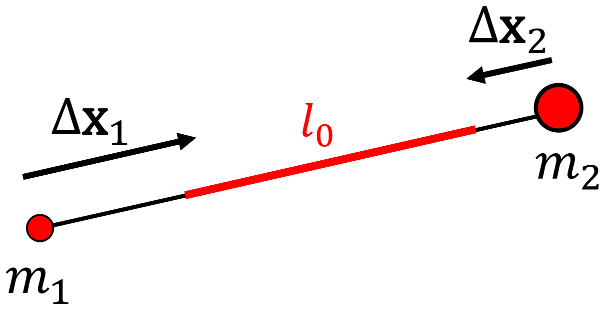


Velocity Correction

- External forces: $\mathbf{v}_{n+1} = \mathbf{u} + \Delta t\ \frac{\mathbf{g}}{m}$
- Internal damping
- Friction
- Restitution



Distance Constraint



$$\Delta\mathbf{x}_1 = -\frac{w_1}{w_1 + w_2}(|\mathbf{x}_1 - \mathbf{x}_2| - l_0)\frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$$

$$\Delta\mathbf{x}_2 = +\frac{w_2}{w_1 + w_2}(|\mathbf{x}_1 - \mathbf{x}_2| - l_0)\frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$$

$$w_i = \frac{1}{m_i}$$

- Conservation of momentum
- Stiffness: scale corrections by $k \in [0,1]$
  - Easy to tune
  - Effect dependent on time step size and iteration count
  - Fixed! See XPBD

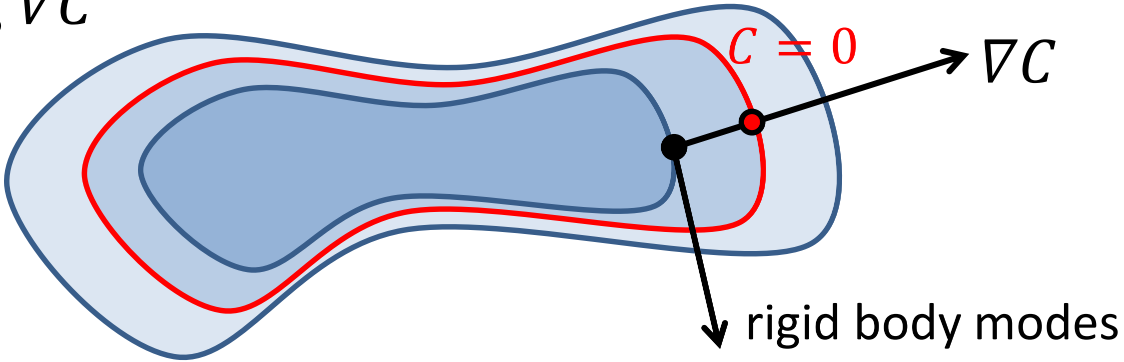
General Internal Constraint

- Define constraint via scalar function:

$$C_{stretch}(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_1 - \mathbf{x}_2| - l_0$$

$$C_{volume}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = [(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)] \cdot (\mathbf{x}_4 - \mathbf{x}_1) - 6v_0$$

- Find configuration for which $C = 0$
- Search along ∇C



Constraint Projection

$$C(\mathbf{x} + \Delta\mathbf{x}) = 0$$

- Linearization (equal for distance constraint)

$$C(\mathbf{x} + \Delta\mathbf{x}) \approx C(\mathbf{x}) + \nabla C(\mathbf{x})^T \Delta\mathbf{x} = 0$$

- Correction vectors

$$\Delta\mathbf{x} = \lambda \, \nabla C(\mathbf{x})$$

$$\lambda = -\frac{C(\mathbf{x})}{\nabla C(\mathbf{x})^T \nabla C(\mathbf{x})}$$

$$\Delta\mathbf{x} = \lambda \, \mathrm{M}^{-1} \nabla C(\mathbf{x})$$

$$\lambda = -\frac{C(\mathbf{x})}{\nabla C(\boldsymbol{x})^T \mathrm{M}^{-1} \, \nabla C(\mathbf{x})}$$

$$\mathrm{M} = diag(m_1, m_2, .., m_n)$$

The Solver

Constraint Solver

- Gauss-Seidel
  - Iterate through all constraints and apply projection
  - Perform multiple iterations
  - Simple to implement
- Modified Jacobi
  - Process all constraints in parallel
  - Accumulate corrections
  - After each iteration, average corrections [Bridson et al., 2002]
- Both known for slow convergence

Global Solver

[Goldenthal et al., 2007]

- Constraint vector

$$\mathrm{C}(\mathbf{x}) = \begin{bmatrix} C_1(\mathbf{x}) \\ \cdots \\ C_M(\mathbf{x}) \end{bmatrix} \qquad \nabla\mathrm{C}(\mathbf{x}) = \begin{bmatrix} \nabla C_1(\mathbf{x})^T \\ \cdots \\ \nabla C_M(\mathbf{x})^T \end{bmatrix} \qquad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \cdots \\ \lambda_M \end{bmatrix}$$

$$\Delta\mathbf{x} = \mathrm{M}^{-1}\nabla C(\mathbf{x})\,\lambda \qquad \lambda = -\frac{C(\mathbf{x})}{\nabla C(\boldsymbol{x})^T \mathrm{M}^{-1}\,\nabla C(\mathbf{x})}$$

⇩

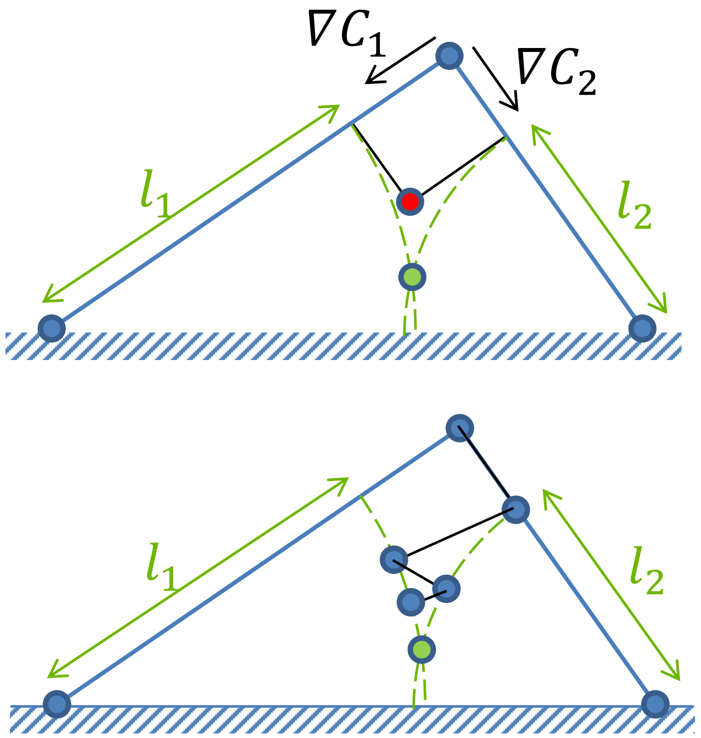
$$\Delta\mathbf{x} = \mathrm{M}^{-1}\nabla\mathrm{C}(\mathbf{x})^T\boldsymbol{\lambda}$$

$$[\nabla C(\mathbf{x})\mathrm{M}^{-1}\nabla\mathrm{C}(\mathbf{x})^T]\,\boldsymbol{\lambda} = -\mathrm{C}(\mathbf{x})$$

Global vs. Gauss-Seidel

- Gradients fixed
- Linear solution ≠ true solution
- Multiple Newton steps necessary

- Current gradients at each constraint projection
- Solver converges to the true solution



Other Speedup Tricks

- Use as smoother in a multi-grid method
- Long range distance constraints (LRA)
- Hierarchy of meshes
- Shape matching

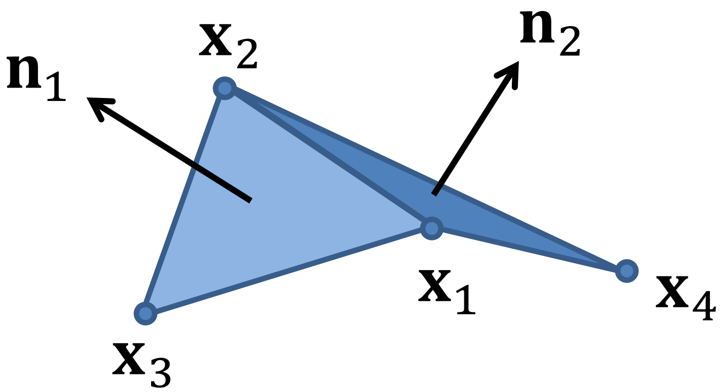
→ more details later

Powerful Gauss-Seidel

- Can handle inequality constraints trivially (LCPs, QPs)!
  - Fluids: separating boundary conditions [Chentanez at al., 2012]
  - Rigid bodies: LCP solver [Tonge et al., 2012]
  - Deformable objects: Long range attachments [Kim et al., 2012]
- Works on non-linear problem directly
- Handles under and over-constrained problems
- GS + PBD: garbage in, simulation out (almost ☺)
- Fine grained interleaved solver trivial
- Easy to implement and parallelize

Constraint Examples

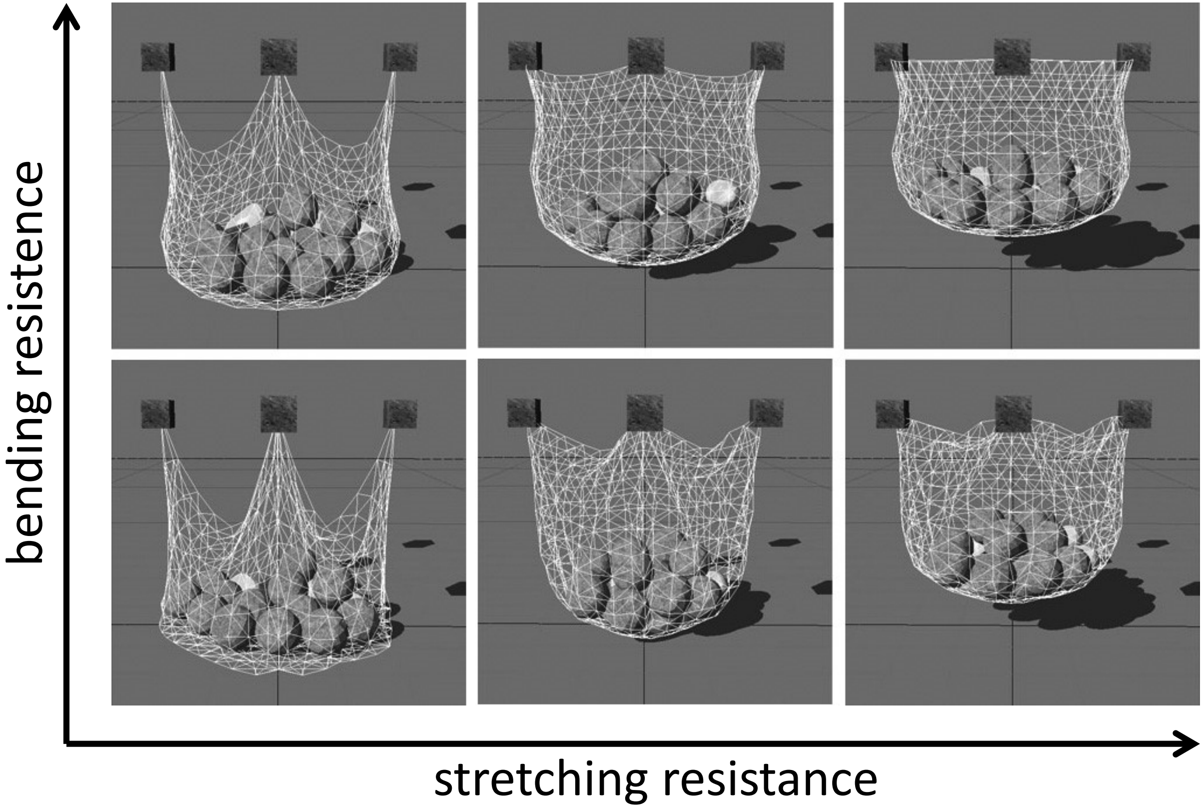
Bending



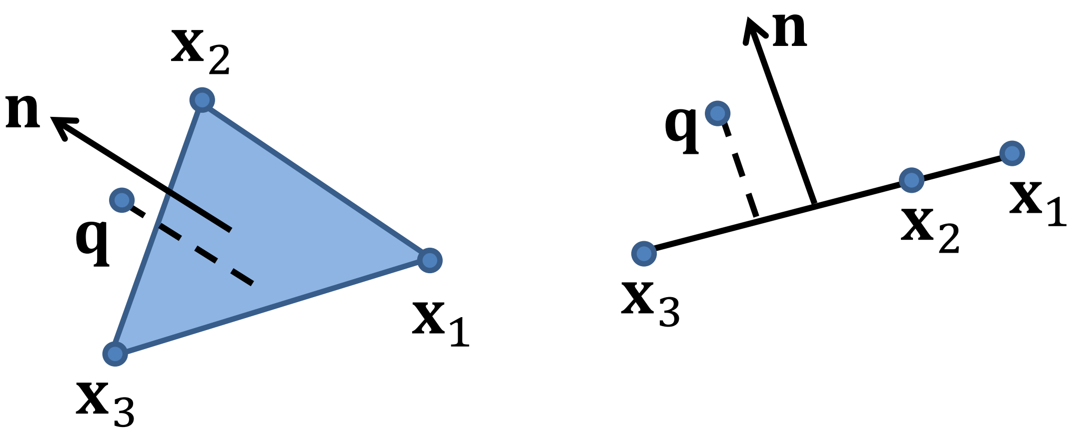
$$C_{bending}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \, acos\left(\frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)}{|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)|} \cdot \frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_1)}{|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_1)|}\right) - \varphi_0$$

- More expensive than constraint $C_{stretch}(\mathbf{x}_3, \mathbf{x}_4)$
- But: Orthogonal to stretching

Stretching – Bending Independence

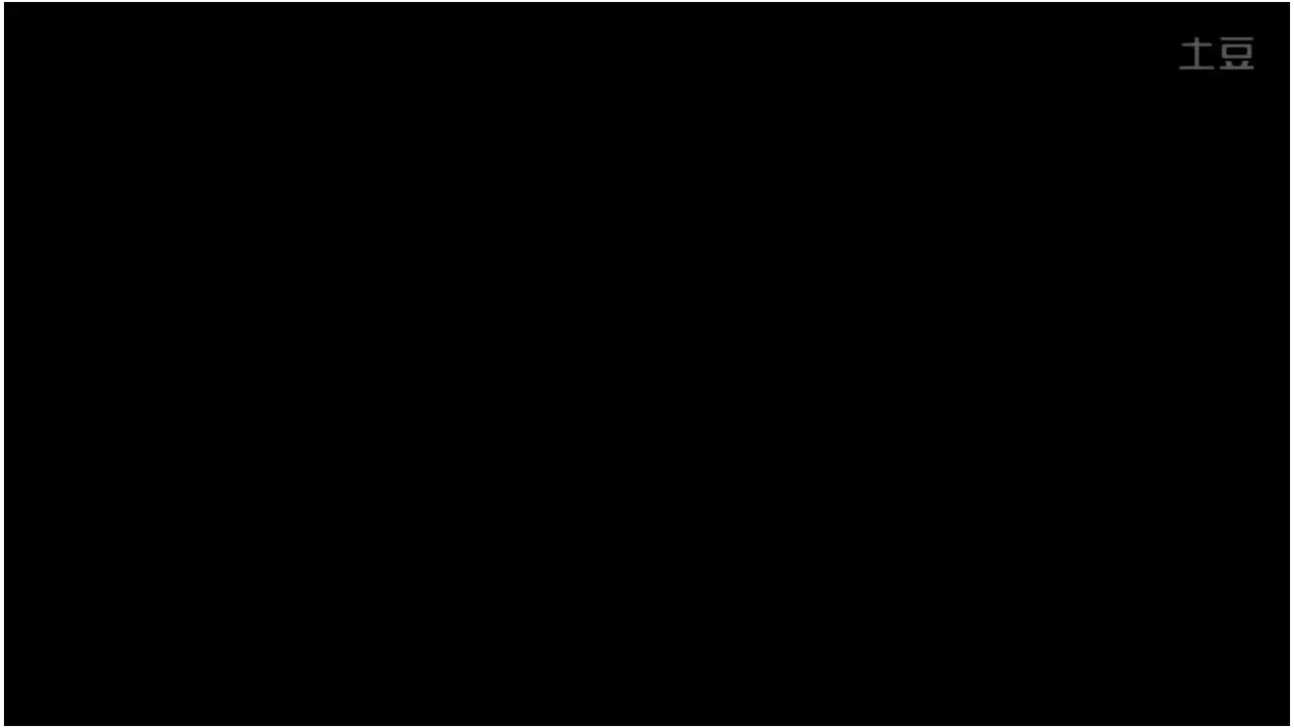


Triangle Collision



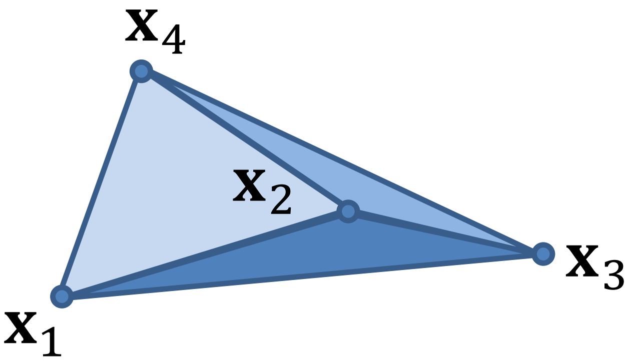
$$C_{coll}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \quad (\mathbf{q} - \mathbf{x}_1) \cdot \frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)}{|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)|} - h$$

Cloth Example



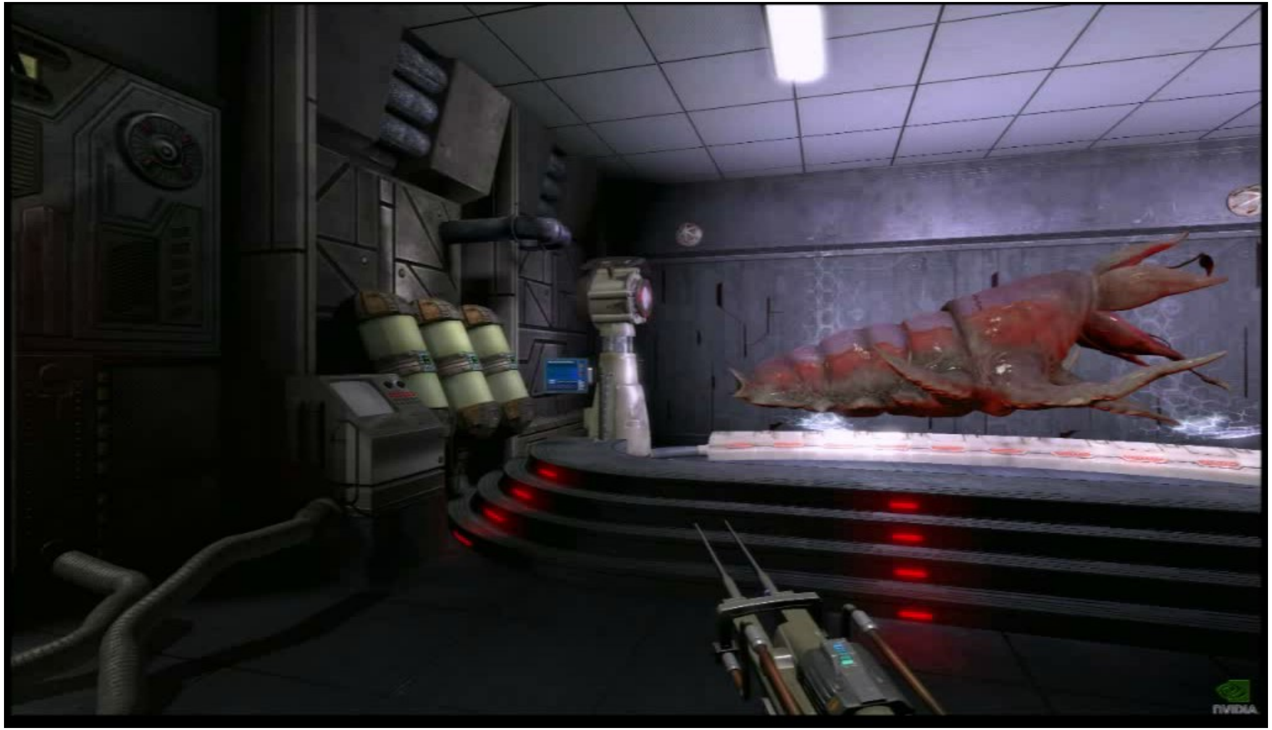
King of Wushu

Tetra Volume

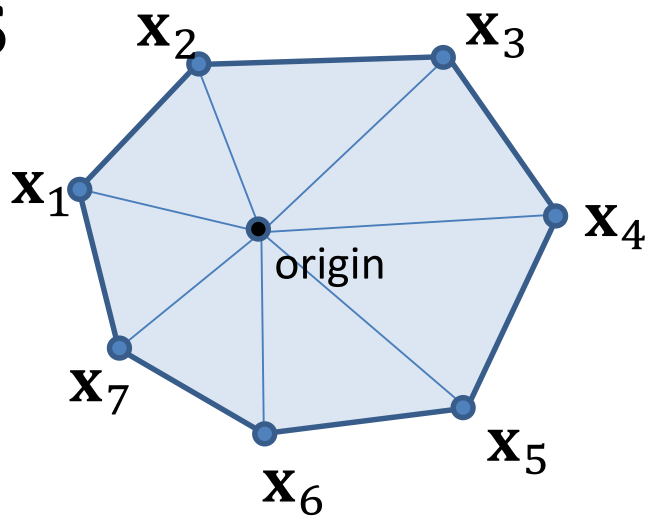


$$C_{air}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = det[\mathbf{x}_2 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_1, \mathbf{x}_4 - \mathbf{x}_1] - 6V_0$$

Soft Body Example



Global Volume - Balloons

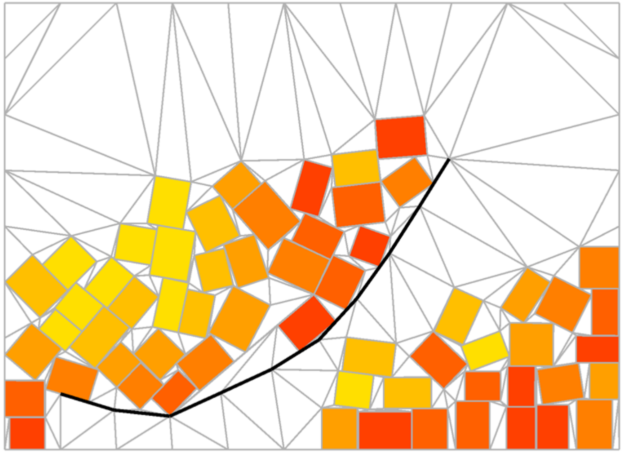


$$C_{balloon}(\mathbf{x}_1, \ldots, \mathbf{x}_N) =$$

$$\frac{1}{6}\left(\sum_{i=1}^{n_{triangles}} \left(\mathbf{x}_{t_1^i} \times \mathbf{x}_{t_2^i}\right) \cdot \mathbf{x}_{t_3^i}\right) - k_{pressure} V_0$$

Air Meshes

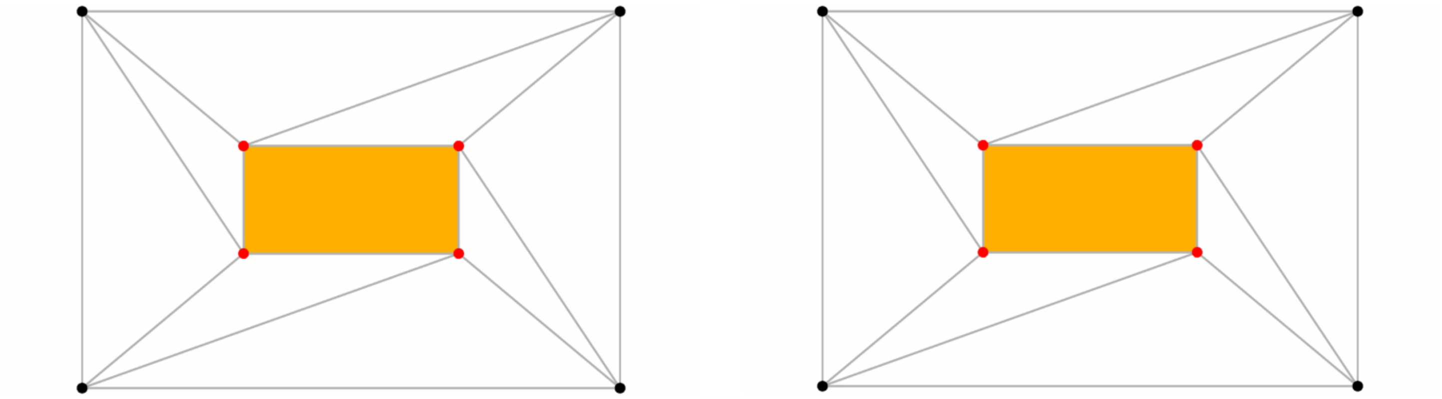
- Triangulate air
- Prevent volume from inverting



- Add one unilateral constraint per cell:

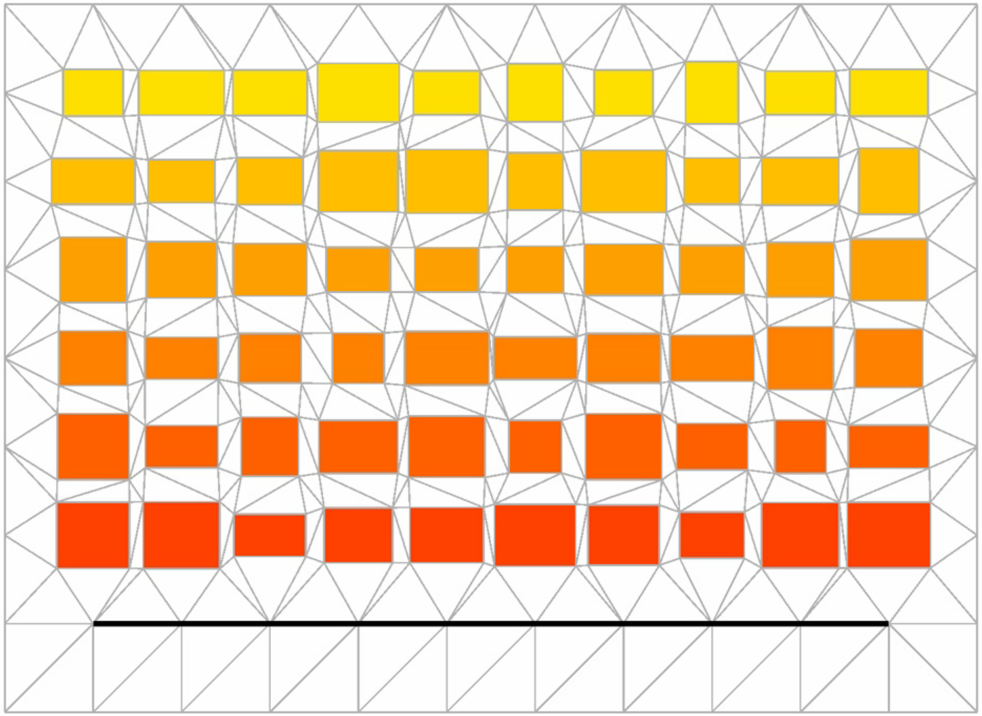
$$C_{air}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = |(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)| \geq 0$$

Locking

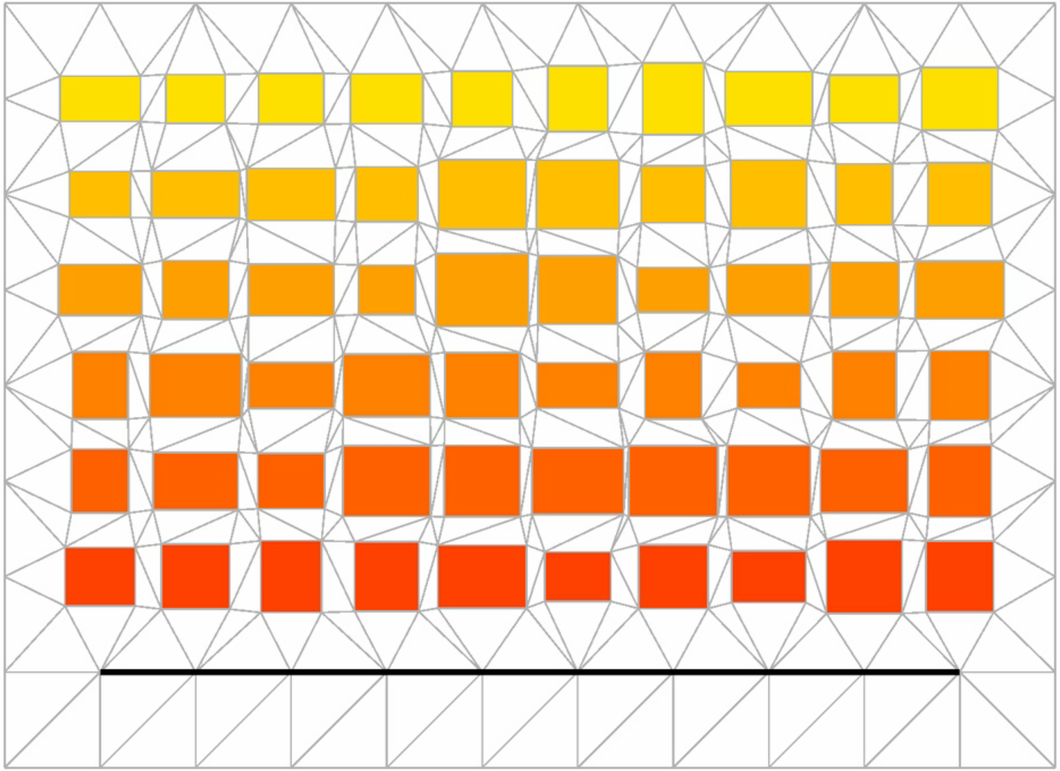


- Elements can invert without collisions
- Solution: Mesh optimization (edge flips)

2D Boxes



Boxes Recovery

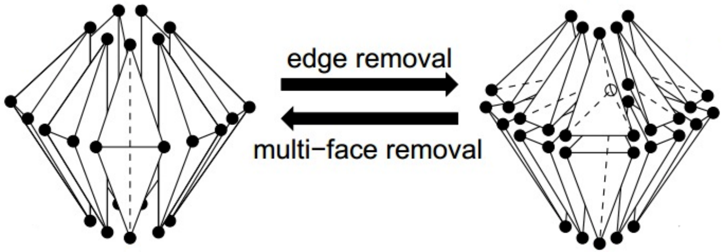


3D Air Meshes

- Per tetra unilateral constraint:

$$C_{air}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = det[\mathbf{x}_2 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_1, \mathbf{x}_4 - \mathbf{x}_1] \geq 0$$

- Mesh optimization more expensive!



3D Air Meshes

- Two cases that work well without optimization

- Multi-layered clothing
- Tissue collision
- No large relative translations / rotations

Multi-Layered Clothing

Untangling

High Resolution Air Mesh

Tissue Collision Handling

Position Based Fluids

[Macklin et al. 2013]

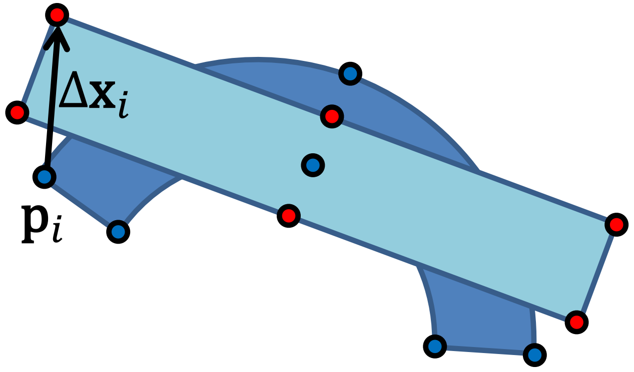
- Particle based
- Pair-wise lower distance constraints
$\rightarrow$ granular behavior
- Move particles in local neighborhood
such that density = rest density
- Density constraint

$$C(\mathbf{x}_1, .., \mathbf{x}_n) = \rho_{SPH}(\mathbf{x}_1, .., \mathbf{x}_n) - \rho_0$$

Position Based Fluids

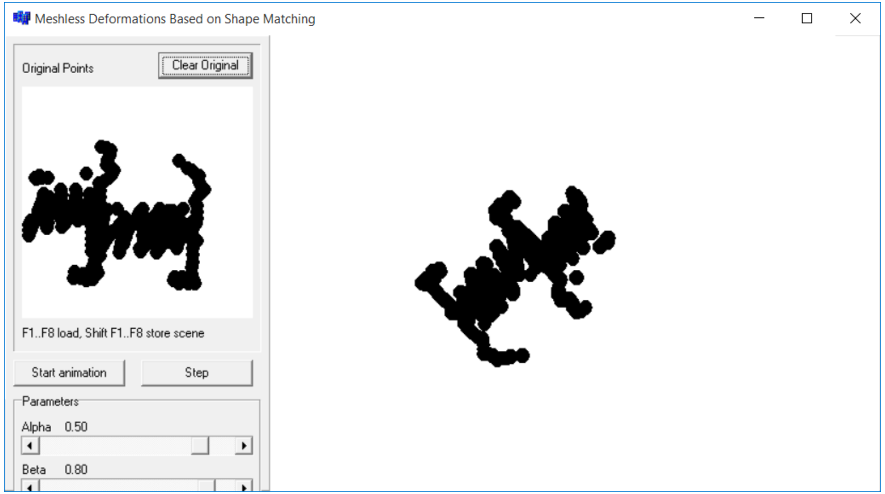
Shape Matching

- Optimally match rest with deformed shape
- Only allow translation and rotation



- Global correction, no propagation needed
- No mesh needed!

2d Demo



Meshless Deformations Based on Shape Matching
Original Points
Clear Original
F1..F8 load, Shift F1..F8 store scene
Start animation
Step
Parameters
Alpha 0.50
Beta 0.80

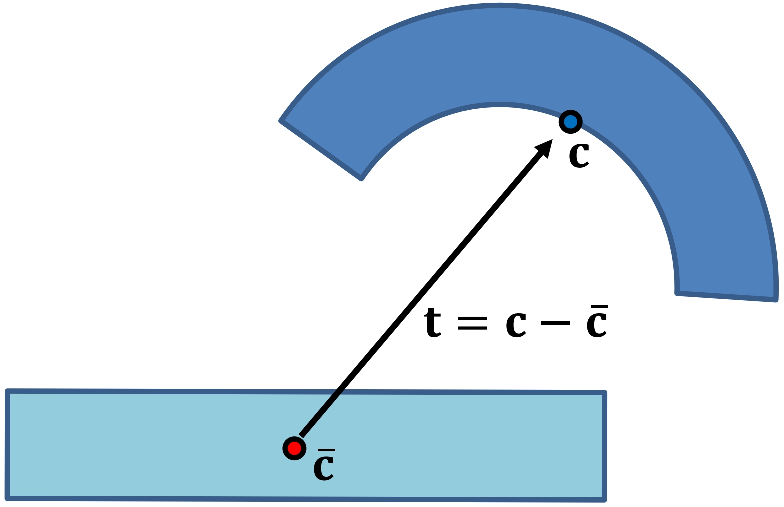
Optimal Translation

- Given rest positions $\bar{\mathbf{x}}_i$, current positions $\mathbf{x}_i$ and masses m_i
- Compute

$$\bar{\mathbf{c}} = \frac{1}{M}\sum_i m_i \bar{\mathbf{x}}_i \qquad M = \sum_i m_i$$

$$\mathbf{c} = \frac{1}{M}\sum_i m_i \mathbf{x}_i$$

$$\mathbf{t} = \mathbf{c} - \bar{\mathbf{c}}$$

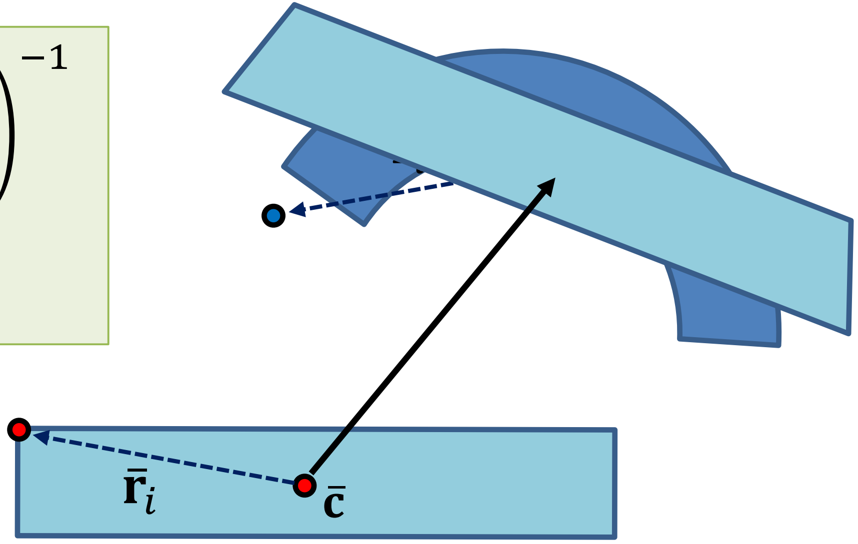


Optimal Transformation

- The optimal linear transformation is:

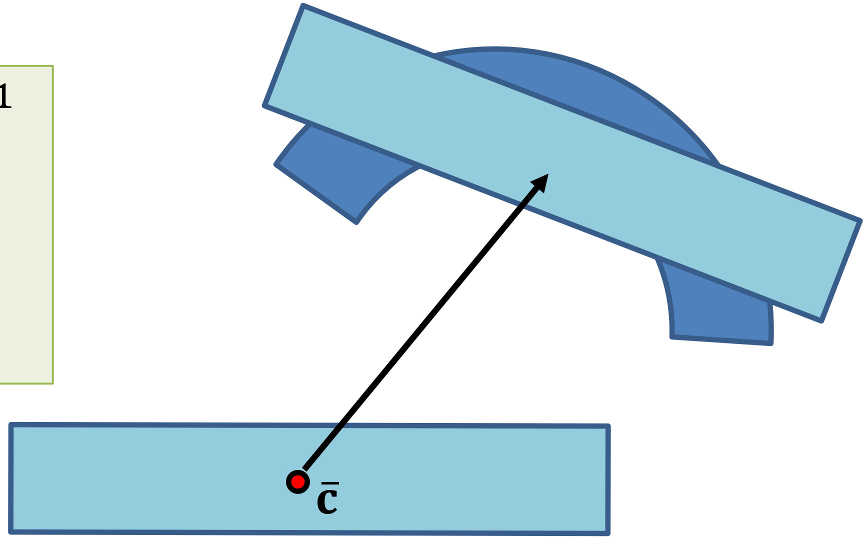
$$\mathbf{A} = \left(\sum_i m_i \mathbf{r}_i \bar{\mathbf{r}}_i^{\,T}\right)\left(\sum_i m_i \bar{\mathbf{r}}_i \bar{\mathbf{r}}_i^{\,T}\right)^{-1}$$
$$= \mathbf{A}_r \mathbf{A}_s$$

$$\bar{\mathbf{r}}_i = \bar{\mathbf{x}}_i - \bar{\mathbf{c}}$$
$$\mathbf{r}_i = \mathbf{x}_i - \mathbf{c}$$



Optimal Rotation

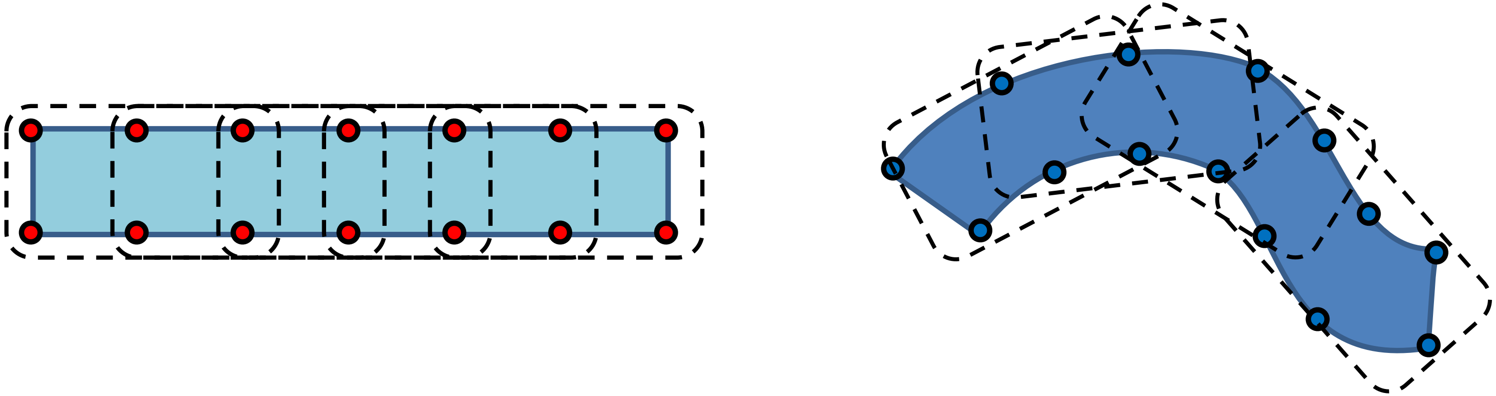
$$\mathbf{A} = \left(\sum_i m_i \mathbf{r}_i \bar{\mathbf{r}}_i{}^T\right)\left(\sum_i m_i \bar{\mathbf{r}}_i \bar{\mathbf{r}}_i{}^T\right)^{-1}$$
$$= \mathbf{A}_r \mathbf{A}_s$$



- $\mathbf{A}_s$ is symmetric →contains no rotation
- Extract rotational part of $\mathbf{A}_r$
- Polar decomposition

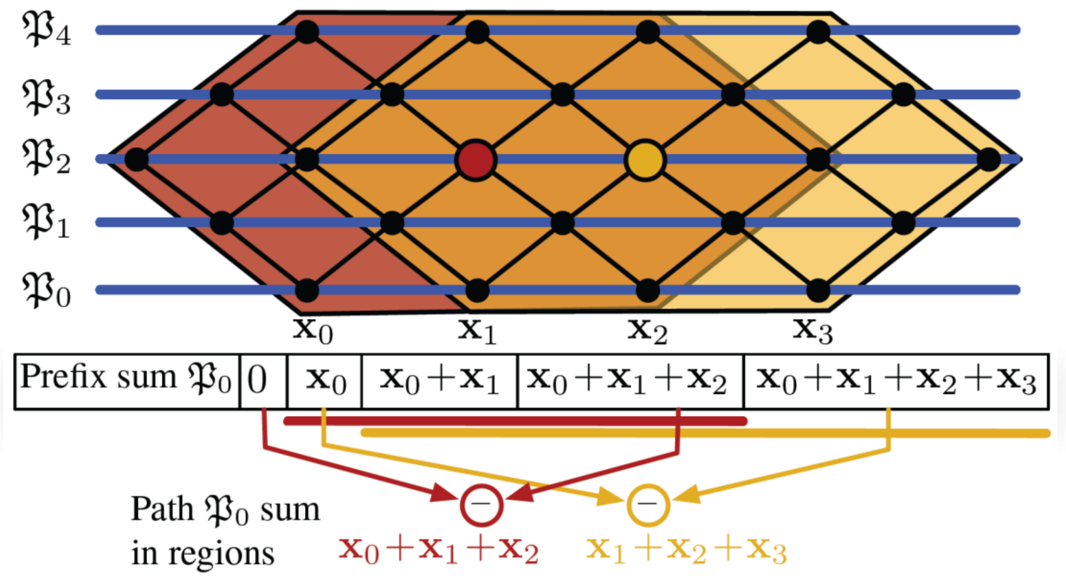
Region Based Shape Matching

- Shape matching allows only small deviations from the rest shape.
- Performing shape matching on several overlapping regions.
- Each particle is part of multiple regions.

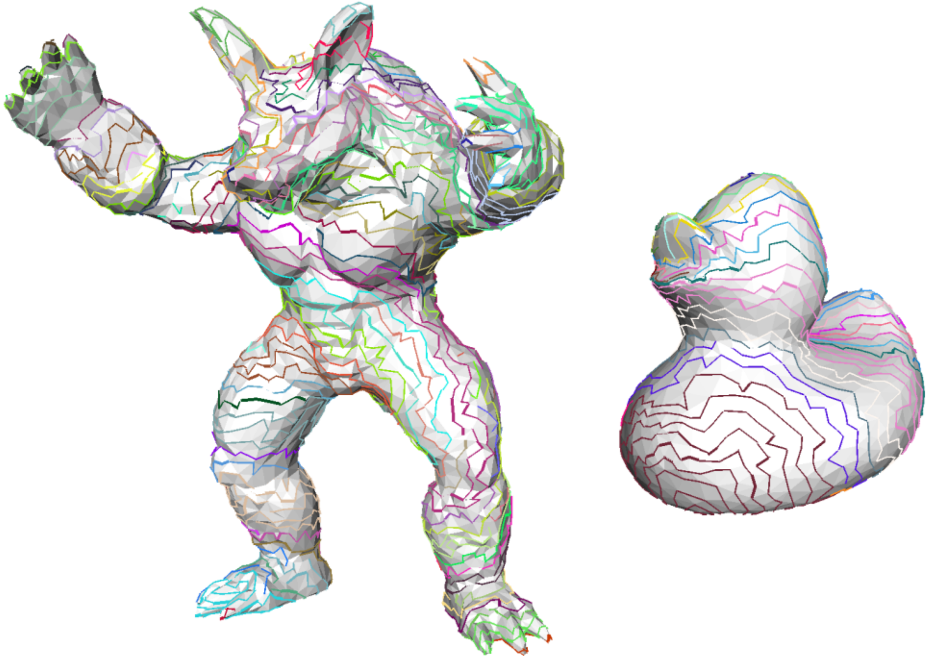


Fast Summation

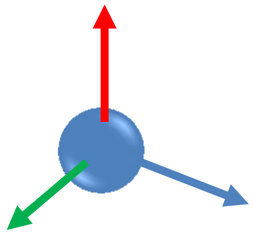
- Compute prefix sum



On Irregular Mesh



Oriented Particles

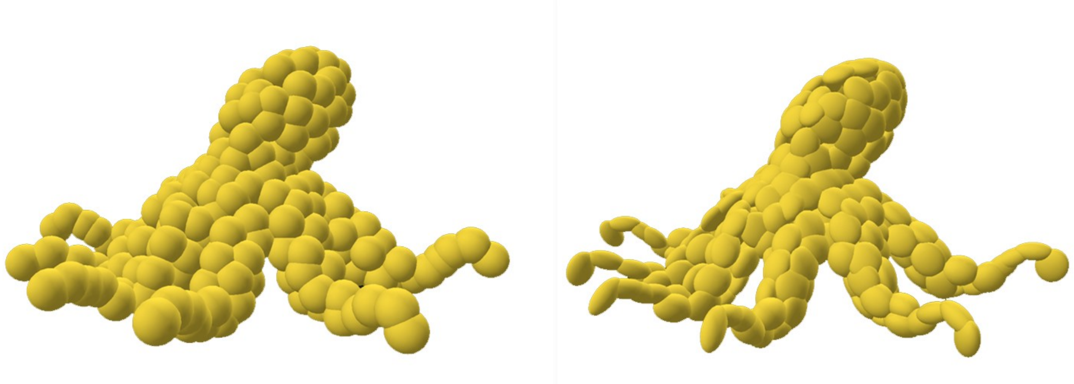


- For co-linear, co-planar or isolated particles optimal transformation is not unique
  $\rightarrow$ Numerical instabilities

- Add orientation information to particles!

Oriented Particles

- Orientation information can be used
  - to stabilize simulation
  - to position anisotropic collision shapes
  - for robust skinning of visual mesh



Generalized Shape Matching

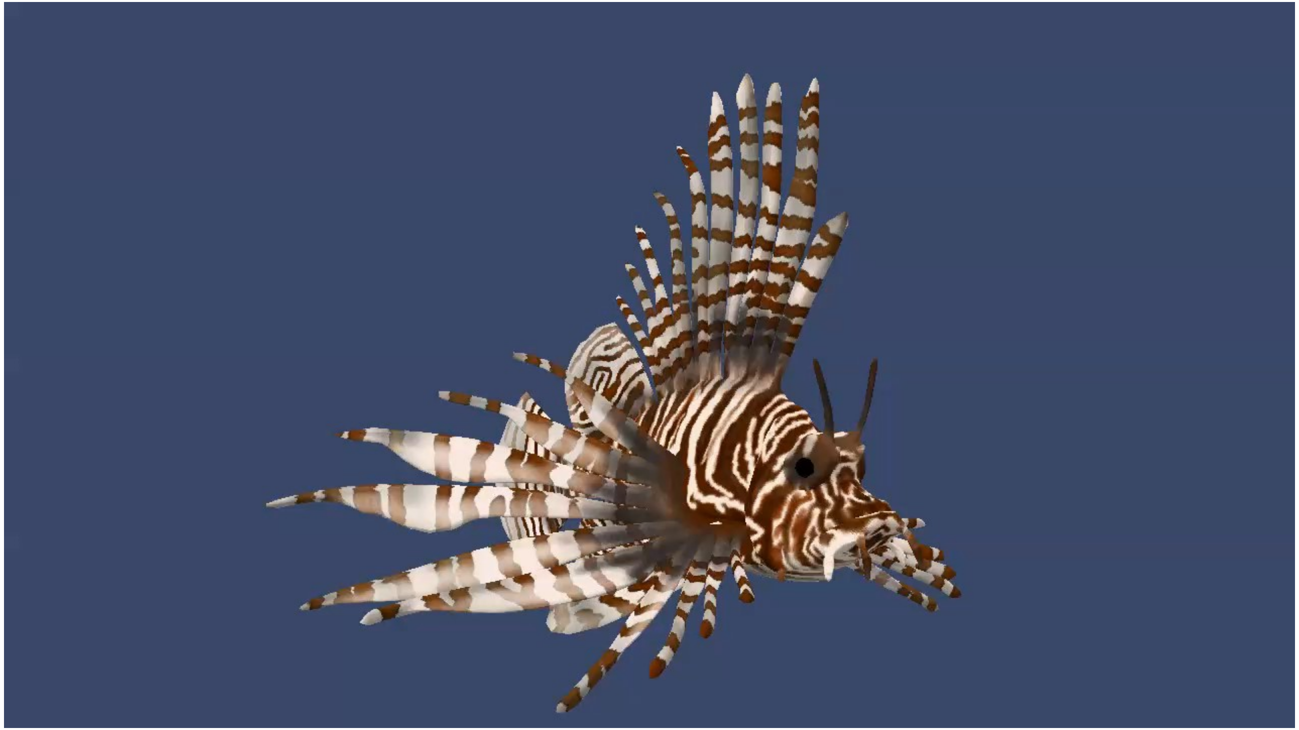
- Optimal translation is still $\mathbf{t} = \bar{\mathbf{c}} - \mathbf{c}$

- Small modification in the calculation of $\mathbf{A}_r$

$$\mathbf{A}_r = \left(\sum_i m_i \mathbf{r}_i \bar{\mathbf{r}}_i{}^T + \mathbf{A}_i \right)$$

where $\mathbf{A}_i^{\text{sphere}} = \frac{1}{5} m r^2 \mathbf{R}$ and $\mathbf{R}$ the particle's rotation matrix

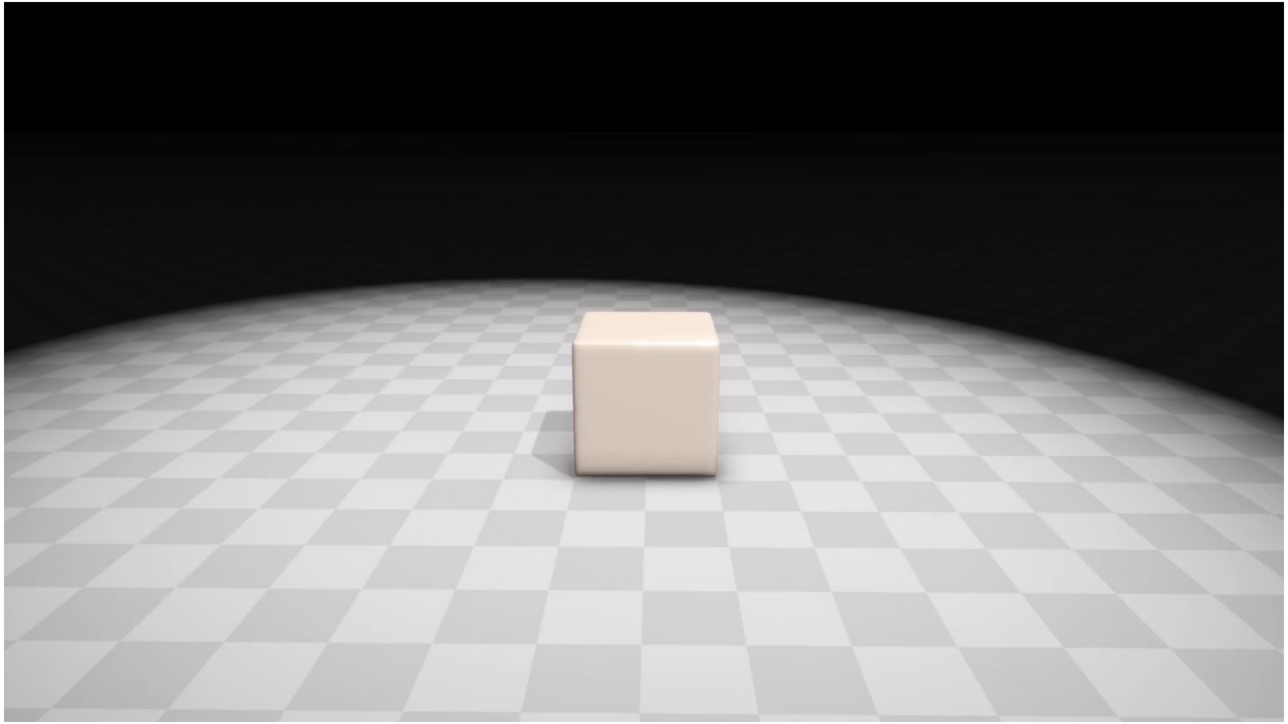
Oriented Particles Demo



Large Elasto-Plastic Deformation

- Handle splits, merges, large deformations
- Use explicit surface mesh to define object
  - Explicit surface tracking for merges and splits
  - Move with particles using linear blend skinning
- Dynamically add and remove particles
  - Remove particles outside surface, resample under-sampled regions
- Dynamically update clusters
  - Control cluster sizes

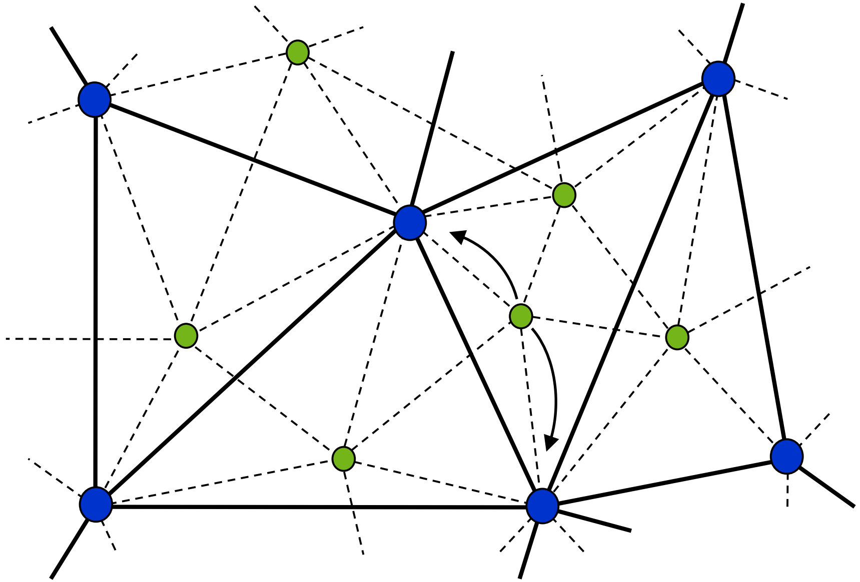
Doug Simulation



Solver Accelerations

Hierarchical PBD

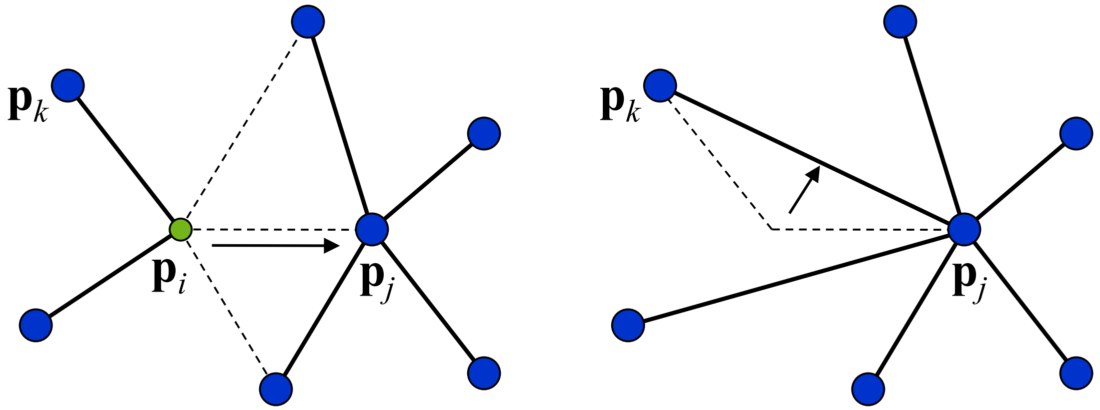
- Create hierarchical mesh



- Next coarser mesh:
  - Subset of vertices
  - Each fine vertex is connected to at least k coarse vertices

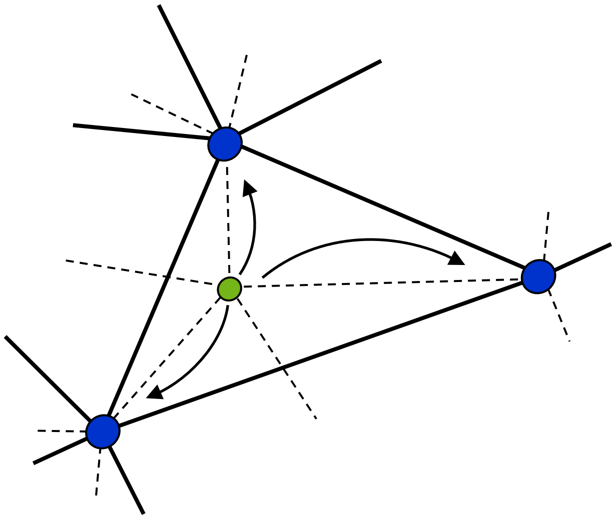
Hierarchical Constraints

- Constraints on coarse meshes



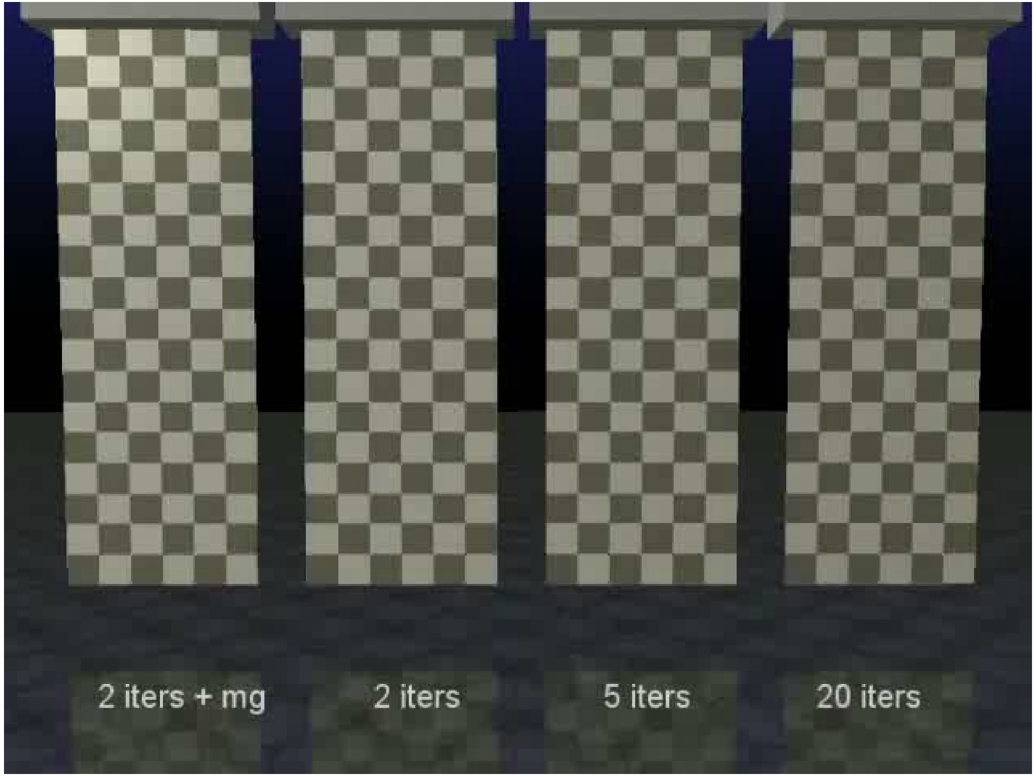
- Unilateral, upper bounds!

Hierarchical Solver

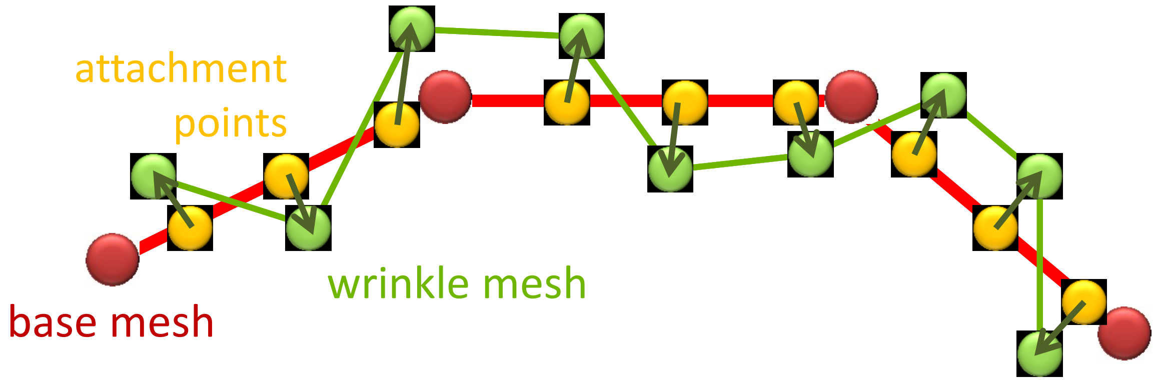


- Solve coarse → fine
- Interpolate displacements from next coarser level

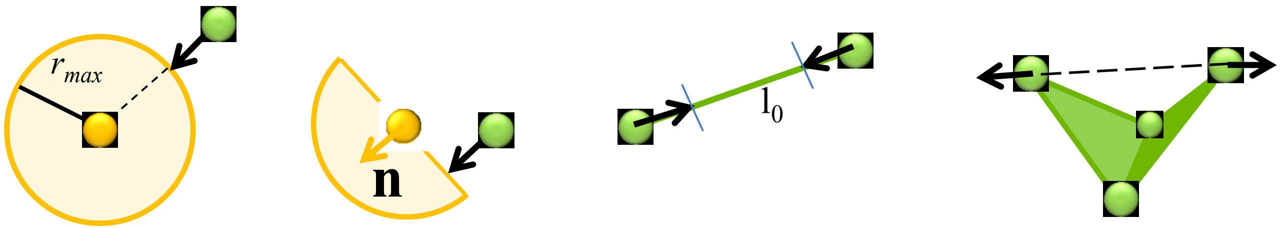
Hierarchical PBD



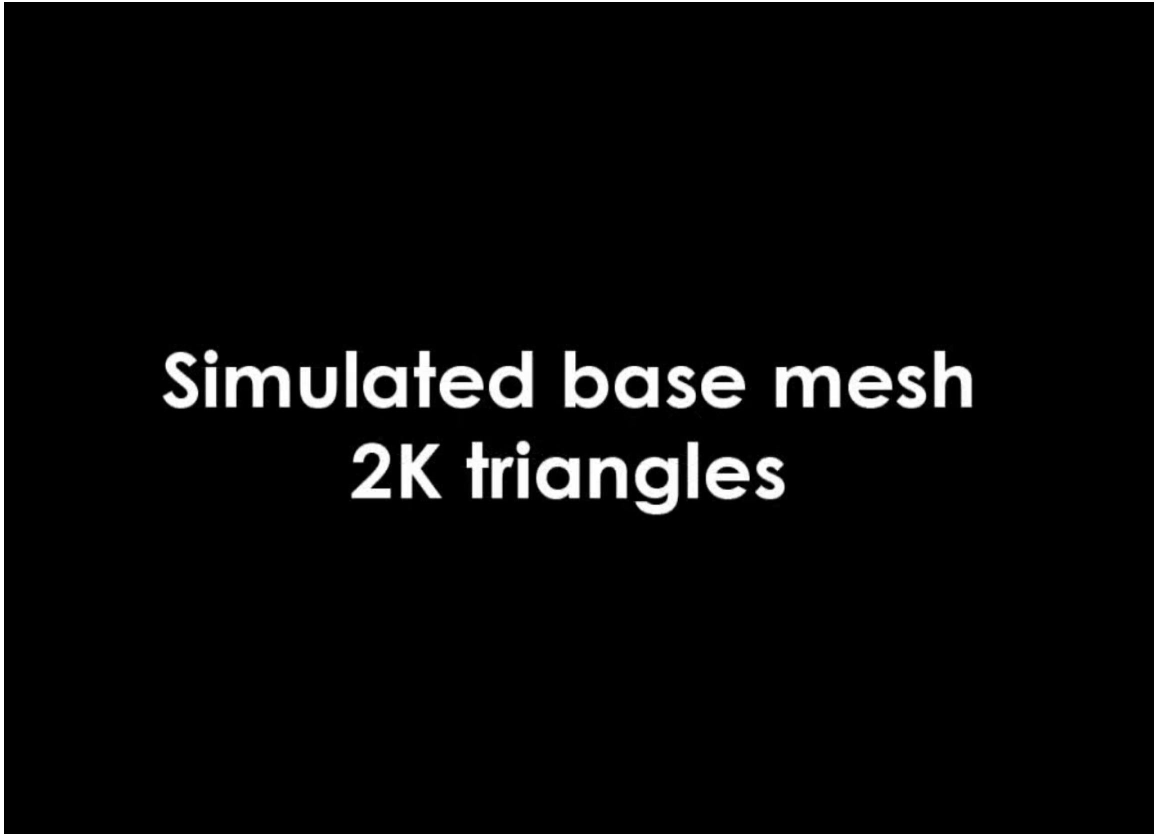
Wrinkle Meshes



- 4 constraint types, geometric projection

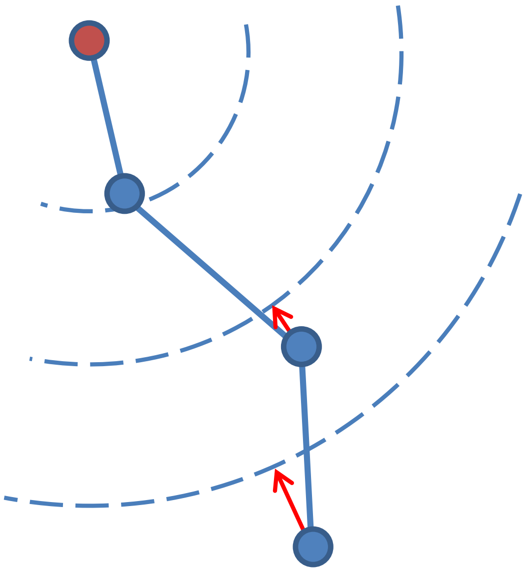


Wrinkle Meshes



Long Range Attachments (LRA)

- Very often cloth is attached (curtain, flags, clothing)
- Upper distance constraint to closest attachment point
- Only radial stretch resistance

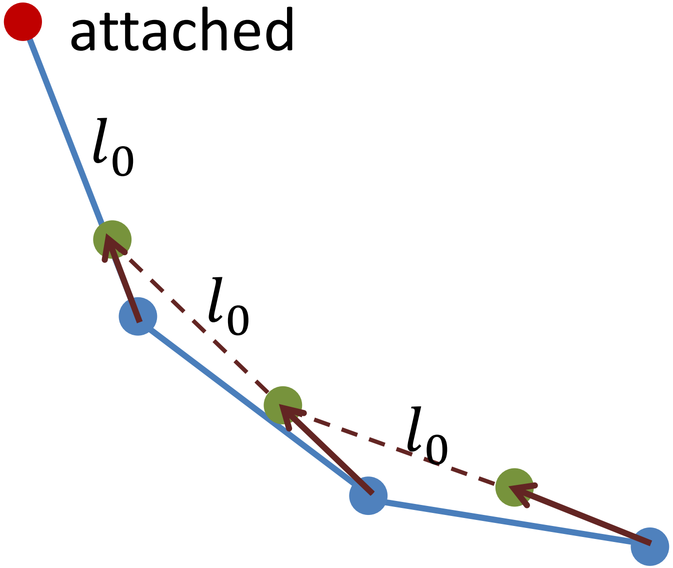


Long Range Attachments (LRA)

[Kim et al., 2012], 90k particles

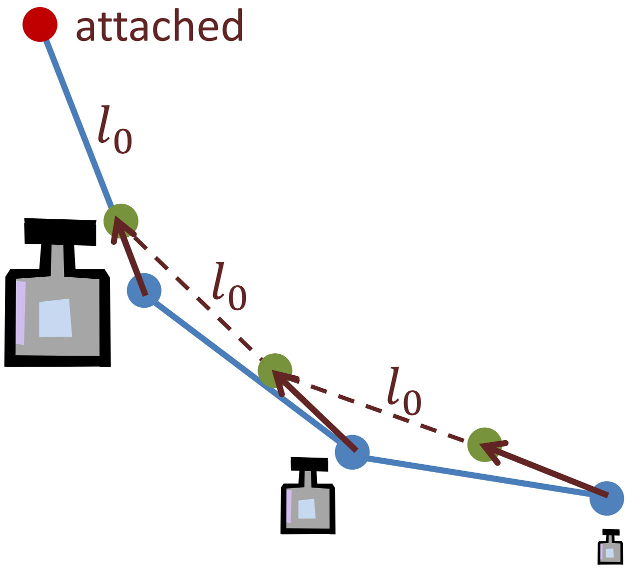
Follow The Leader (FTL)

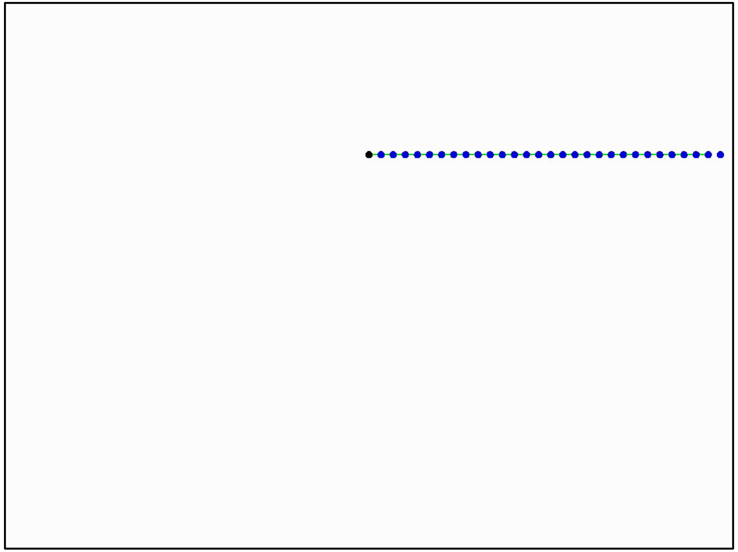
- From top to bottom
- Only move lower particle
- All constraints satisfied!



Follow The Leader (FTL)

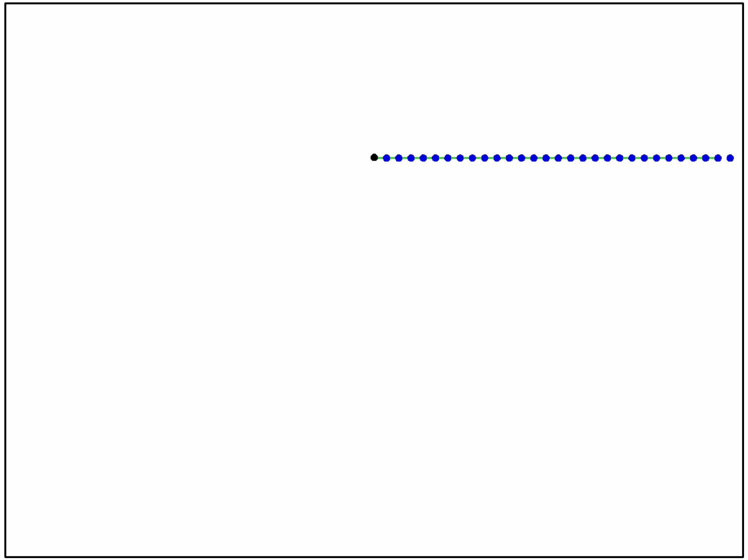
- Momentum not conserved!





Dynamic Follow The Leader (DFTL)

- Update positions one-sided
- Update velocities symmetrically



Fur Demo

630k particles, 100k hairs, 27.29 fps
Left Mouse – Blow
Right Mouse – Rotate Camera
3 Lights
8xMSAA
Supersampling
Shadow Opacity
Shadow Ambient
Per Hair Shadow Variation
Hair Opacity
Hair Base Thickness
Hair Tip Thickness
Hair Diffuse Color Scale
Hair Specular Color Scale
Hair Specularity
Fluffiness
Wind X
Wind Y
Wind Z
Blower Speed
Blower Radius
Animate (A)
Simulate (S)
Slim Windy (1)
Normal Windy (2)
Fluffy Windy (3)
Slim No Wind (4)
Normal No Wind (5)
Fluffy No Wind (6)