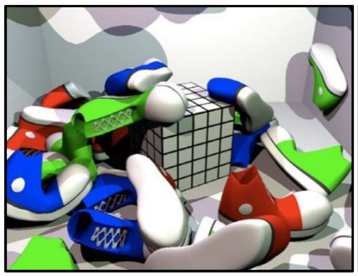
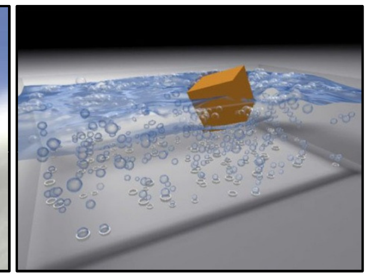
Real Time Physics

www.matthiasmueller.info/realtimephysics





Matthias Müller
NVIDIA

Doug James
Cornell University

Jos Stam
Autodesk

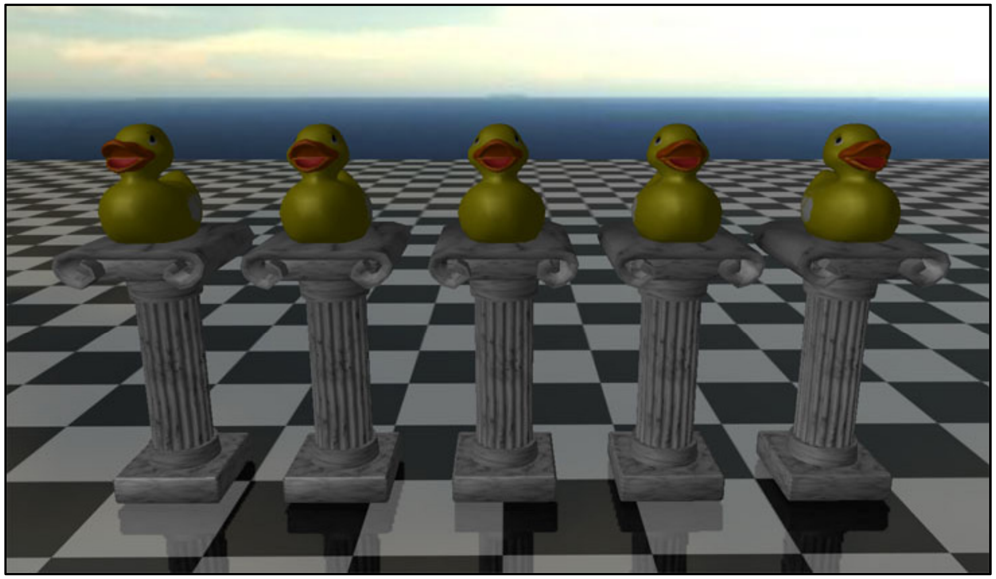
Nils Thuerey
ETH Zürich

SIGGRAPH2008

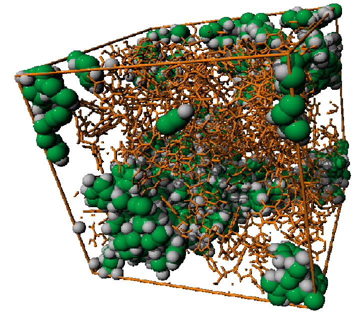
Schedule

- 8:30 **Introduction**, Matthias Müller
- 8:45 **Deformable Objects**, Matthias Müller
- 9:30 **Multimodal Physics and User Interaction**
  Doug James
- 10:15 **Break**
- 10:30 **Fluids**, Nils Thuerey
- 11:15 **Unified Solver**, Jos Stam
- 12:00 **Q & A**

Real Time Demos



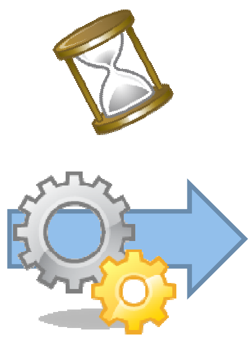
Before Real Time Physics



- My Ph.d. thesis:
  - Find 3d shape of dense polymer systems

```
!BIOSYM molecular_data 4
@molecule POLYCARB_B0

CARB_1:C        C cp     C1/1.5 C5/1.5 HC
CARB_1:HC       H hc     C
CARB_1:C1       C cp     C/1.5 C2/1.5 H1
CARB_1:H1       H hc     C1
CARB_1:C2       C cp     C1/1.5 C3/1.5 C7
CARB_1:C3       C cp     C2/1.5 C4/1.5 H3
CARB_1:H3       H hc     C3
CARB_1:C4       C cp     C3/1.5 C5/1.5 H4
CARB_1:H4       H hc     C4
...
...
```



```
!BIOSYM archive
PBC=OFF

C 3.313660622 -2.504962206 -11.698267937
HC 2.429594755 -2.005253792 -12.065854073
C1 4.420852184 -1.754677892 -11.284504890
H1 4.390906811 -0.676178515 -11.332902908
C2 5.566863537 -2.402448654 -10.808005333
C3 5.605681896 -3.800502777 -10.745265961
H3 6.489747524 -4.300211430 -10.377680779
C4 4.498489857 -4.550786972 -11.159029007
H4 4.528435707 -5.629286766 -11.110631943
...
...
```

Meeting Real Time Physics

- Post doc at MIT (1999-2001)
  - Plan: Parallelization of packing algorithms
  - Prof had left MIT before I arrived!
- Change of research focus
  - Computer graphics lab on same floor
  - Real-time physics needed for a virtual sculptor

B.Cutler et al.

1999

- Among my literature search:
  - D. James et al., *ArtDefo, Accurate Real Time Deformable Objects*, Siggraph 1999
  - J.Stam, *Stable Fluids*, Siggraph 1999
- They brought physics brought to life!
- My assignment: make this real-time:
  - J. O'Brien at al., *Graphical Modeling and Animation of Brittle Fracture,* Siggraph 1999

ArtDefo

- Boundary element method
- Haptic interaction

PhysX by NVIDIA

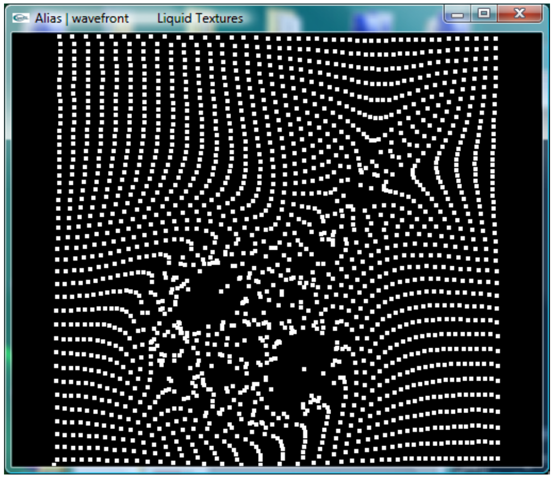
SIGGRAPH2008

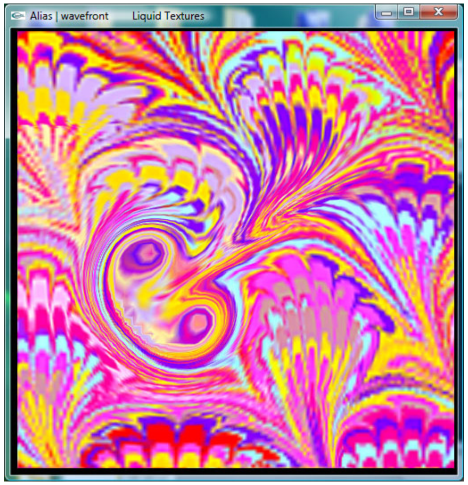
Doug James

- CV
  - 2001: PhD in applied mathematics, University of British Columbia
  - 2002: Assistant prof, Carnegie Mellon University
  - 2006: Associate prof, Cornell Universtiy
  - National Science Foundation CAREER award
- Research interests
  - Physically based animation
  - Haptic force feedback rendering
  - Reduced-order modeling

Stable Fluids

- Semi-Lagrangian advection
- Equation splitting





PhysX by NVIDIA

SIGGRAPH2008

Jos Stam

- CV
  - PhD in computer science, University of Toronto
  - Postdoc in Paris and Helsinki
  - Senior research scientist at Alias|Wavefront, now Autodesk
  - SIGGRAPH Technical Achievement Award
- Research interests
  - Natural phenomena
  - Physics based simulation
  - Rendering and surface modeling

PhysX by NVIDIA

SIGGRAPH2008

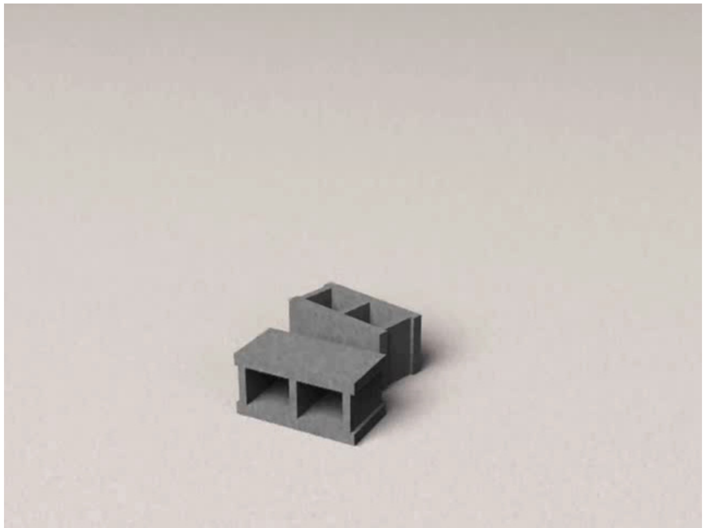
Animation of Brittle Fracture

- Finite elements, separation tensor
- Great results but 5-10 min/frame

J. O'Brien et al.

Real-Time Fracture of Stiff Materials

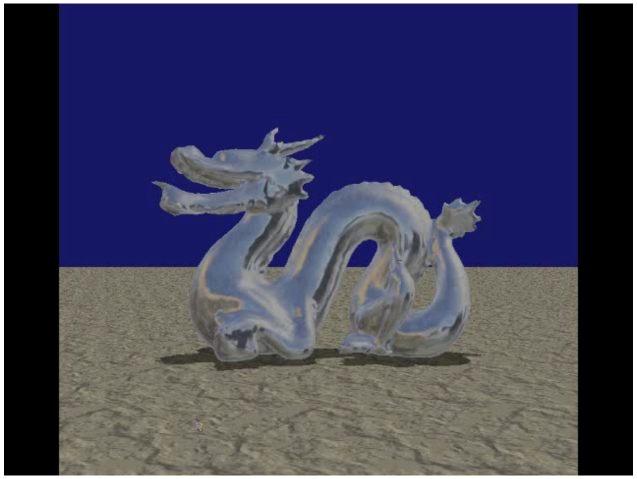
- Hybrid rigid body – static FEM
- Not quite as realistic but 30 fps



M.Müller et al. Eurographics CAS 2001

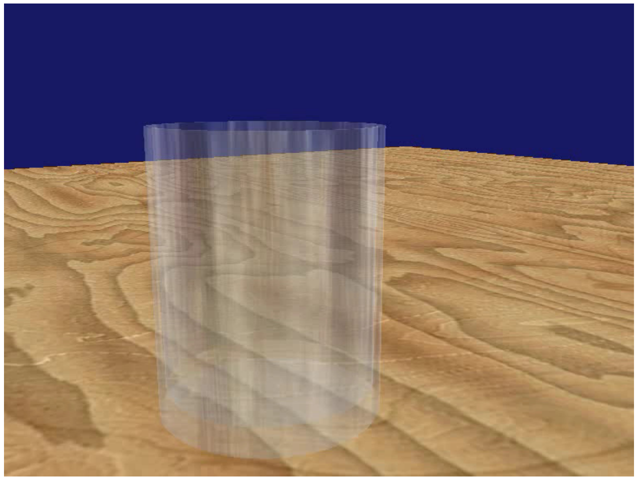
Deformables and Water

- Post doc with ETH computer graphics lab



2004

FEM base deformables



2003, Video by D.Charypar

SPH fluids

NovodeX - AGEIA

- 2003 NovodeX as ETH spin-off
- 2004 Acquisition by AGEIA
- 2007 Nils Thuerey AGEIA post doc

PhysX™ by NVIDIA

SIGGRAPH2008

Nils Thuerey

- CV
  - 2007: PhD in computer science from University Erlangen
  - 2007: Post doc with AGEIA
  - 2008: Post doc with ETH
- Research interests
  - Lattice-Boltzmann based fluid simulation
  - Real-time height field fluid simulation
  - Fluid Control

PhysX™ by NVIDIA

SIGGRAPH2008

Offline Physics

- Applications
  - Special effects in movies and commercials
- Typical setup
  - Millions of particles / triangles / tetrahedra / grid cells
  - Expensive photorealistic rendering
  - Impressive high quality results
  - Seconds up to hours per frame
- Characteristics
  - Predictable, re-run possible, no interaction

Real Time Physics

- Applications
  - Interactive systems
  - Virtual surgery simulators („respectable“, „scientific“)
  - Games (not so respectable but true in 99%)
- Requirements
  - Fast, 40-60 fps of which physics only gets a small fraction
  - Stable in any possible, non-predictable situation
- Challenge:
  - Approach offline results while meeting all requirements!

From Offline to Real Time

- Resolution reduction
  - Blobby and coarse look
  - Details disappear
- Use specialized real time techniques!
  - Physics low-res, appearance hi-res (shader effects)
  - Reduction of dimension from 3d to 2d (height field fluids, BEM)
  - Level of detail (LOD)
  - No equation solving, procedural animation for specific effects

Deformable Objects

Examples of Deformable Objects

- 1d: Ropes, hair

- 2d: Cloth, clothing

- 3d: Fat, tires, organs

PhysX by NVIDIA

SIGGRAPH2008

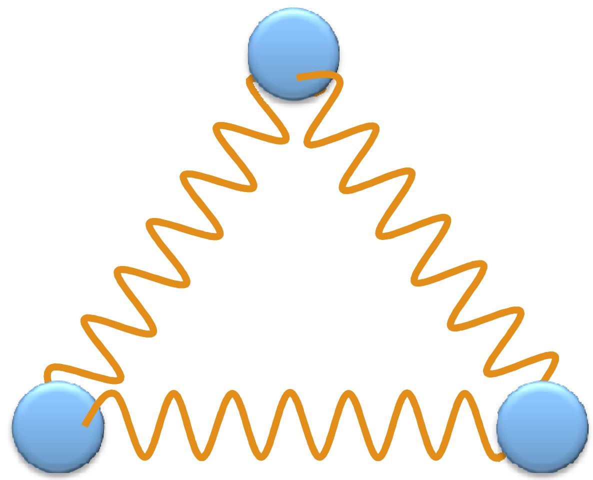
Dimensionality

- Every real object is 3d
- Approximated object with lower dimentional models if possible
- Dimension reduction substantially saves simulation time

PhysX by NVIDIA

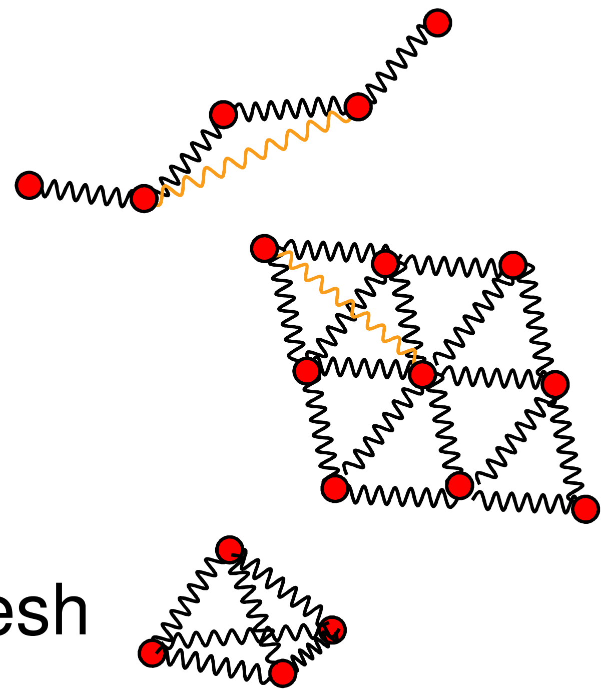
SIGGRAPH2008

Mass Spring Systems



Mass Spring Meshes

- Rope: chain
  - Additional springs for bending and torsional resistance needed
- Cloth: triangle mesh
  - Additional springs for bending restistance needed
- Soft body: tetrahedral mesh

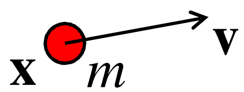


PhysX™ by NVIDIA

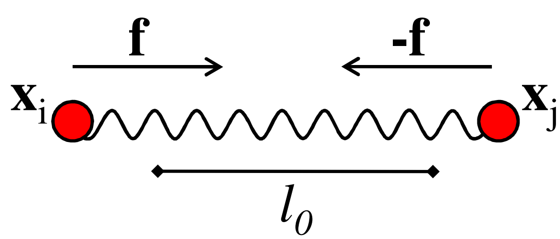
SIGGRAPH2008

Mass Spring Physics

- Mass point: mass m, position $\mathbf{x}$, velocity $\mathbf{v}$



- Springs:



$$\mathbf{f} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\left|\mathbf{x}_j - \mathbf{x}_i\right|}\left[k_s\left(\left|\mathbf{x}_j - \mathbf{x}_i\right| - l_o\right) + k_d\left(\mathbf{v}_j - \mathbf{v}_i\right)\cdot\frac{\mathbf{x}_j - \mathbf{x}_i}{\left|\mathbf{x}_j - \mathbf{x}_i\right|}\right]$$

- Scalars k_s, k_d, stretching, damping coefficients

Time Integration

- Newton:

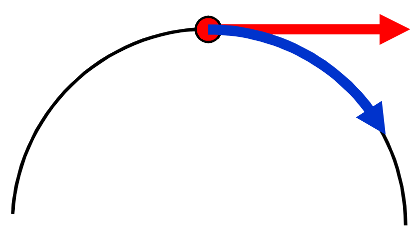
$$\dot{\mathbf{v}} = \mathbf{f} / m$$
$$\dot{\mathbf{x}} = \mathbf{v}$$

- Explicit Euler:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^{t} + \Delta t \frac{1}{m_i} \sum_j \mathbf{f}(\mathbf{x}_i^t, \mathbf{v}_i^t, \mathbf{x}_j^t, \mathbf{v}_j^t)$$
$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^{t} + \Delta t\, \mathbf{v}_i^{t+1}$$

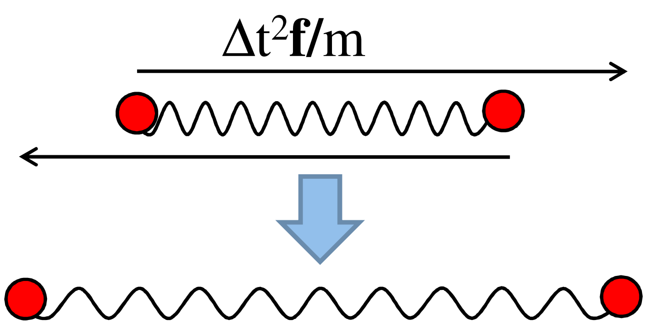
- Assumes velocity and force constant within Δt
- Correct would be:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \int_t^{t+\Delta t} \mathbf{v}(t) dt$$



Explicit Euler Issues

- Accuracy
  - Better with higher order schemes e.g. Runge Kutta
  - Not critical in real time environments
- Stability
  - Overshooting
  - Big issue in real time systems!



PhysX by NVIDIA

SIGGRAPH2008

Implicit Integration

- Use values of next time step on the right

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^{t} + \Delta t \frac{1}{m_i} \sum_j \mathbf{f}(\mathbf{x}_i^{t+1}, \mathbf{v}_i^{t+1}, \mathbf{x}_j^{t+1}, \mathbf{v}_j^{t+1})$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^{t} + \Delta t\, \mathbf{v}_i^{t+1}$$

- Intuitively
  - Don‘t extrapolate blindly
  - Arrive at a physical configuration

Implicit Integration Issues

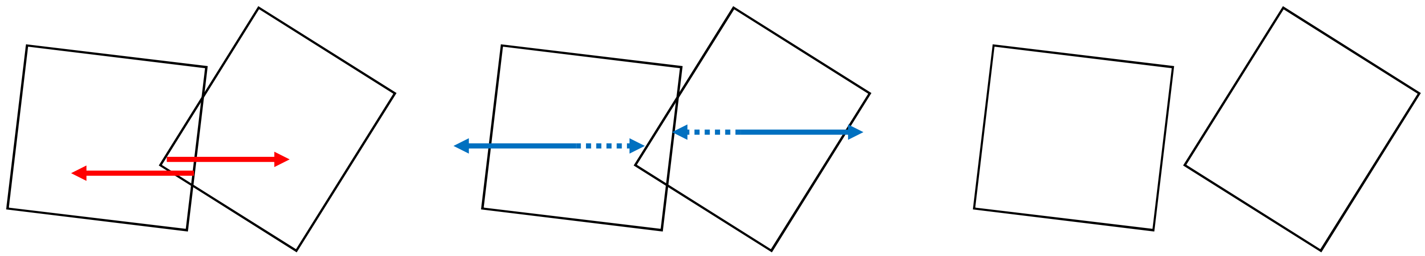
- Unconditionally stable (for any Δt)!
- Have to solve system of equations for velocities
  - n mass points, $3n$ unknowns
  - Non linear when the forces are non-linear in the positions as with springs
  - Linearize forces at each time step (Newton-Raphson)
- Slow $\rightarrow$ Take large time steps
- Temproal details disappear, numerical damping

PhysX™ by NVIDIA

SIGGRAPH2008

Position Based Dynamics

Force Based Update



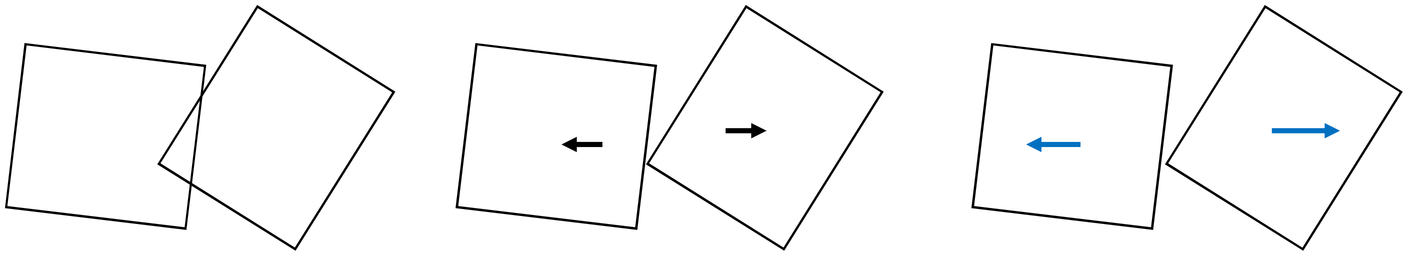
penetration causes forces

forces change velocities

velocities change positions

- Reaction lag
- Small $k_s \rightarrow$ squashy system
- Large $k_s \rightarrow$ stiff system, overshooting

Position Based Update



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

PhysX by NVIDIA

SIGGRAPH2008

Position Based Integration

Init all $\mathbf{x}_i^0$, $\mathbf{v}_i^0$

Loop

$\mathbf{p}_i$	$= \mathbf{x}_i^t + \Delta t \cdot \mathbf{v}_i^t$	// prediction
$\mathbf{x}_i^{t+1}$	= modify $\mathbf{p}_i$	// position correction
$\mathbf{u}_i$	$= [\mathbf{x}_i^{t+1} - \mathbf{x}_i^t] / \Delta t$	// velocity update
$\mathbf{v}_i^{t+1}$	= modify $\mathbf{u}_i$	// velocity correction

End loop

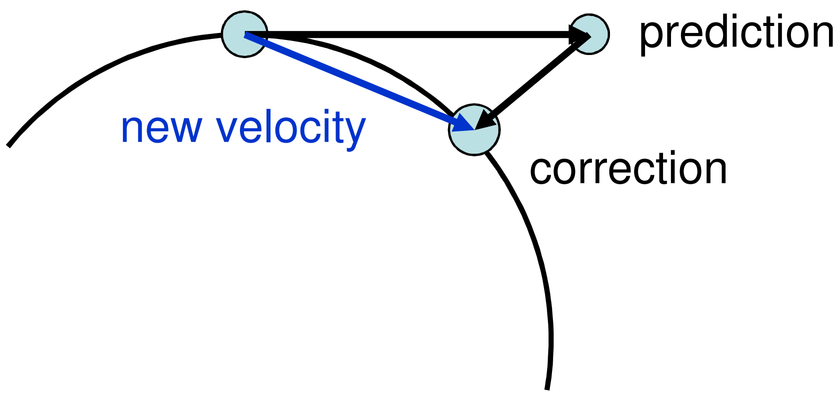
- Explicit, Verlet related
- If correction done by a solver → semi implicit

PhysX by NVIDIA

SIGGRAPH2008

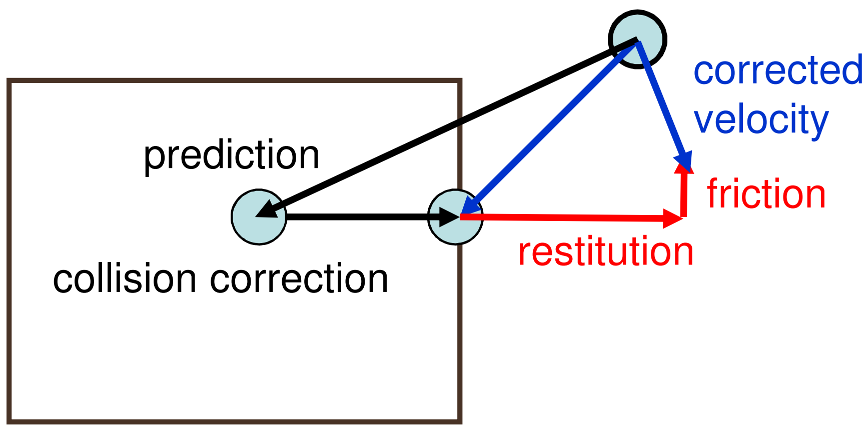
Position Correction

- Move vertices out of other objects
- Move vertices such that constraints are satisfied
- Example: Particle on circle

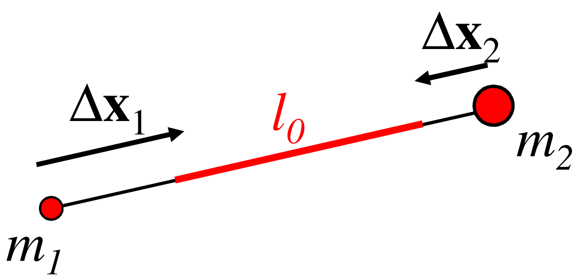


Velocity Correction

- External forces: $\mathbf{v}^t = \mathbf{u}^t + \Delta t \cdot \mathbf{g}/m$
- Internal damping
- Friction
- Restitution



Internal Distance Constraint



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

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

$$w_i = 1/m_i$$

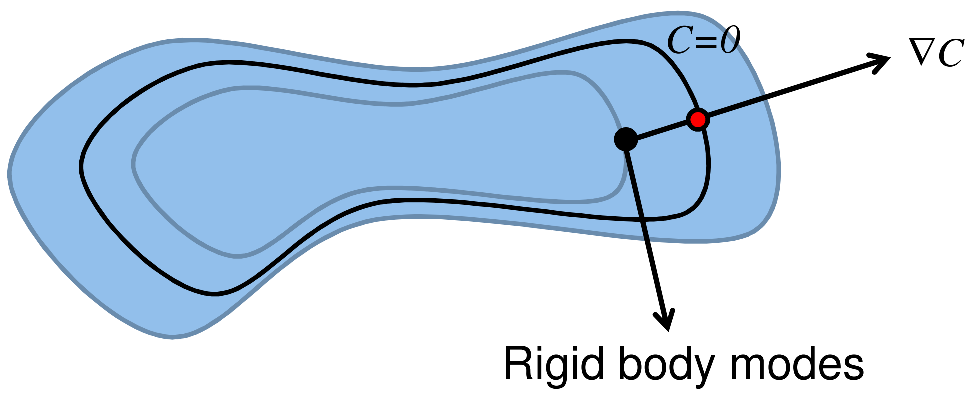
- Conservation of momentum
- Stiffness: scale corrections by $k \in [0..1]$
- Easy to tune but effect dependent on time step!

General Internal Constraint

- Define constraint via scalar function:

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

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



PhysX by NVIDIA

SIGGRAPH2008

General Position Correction

- Correction along gradient

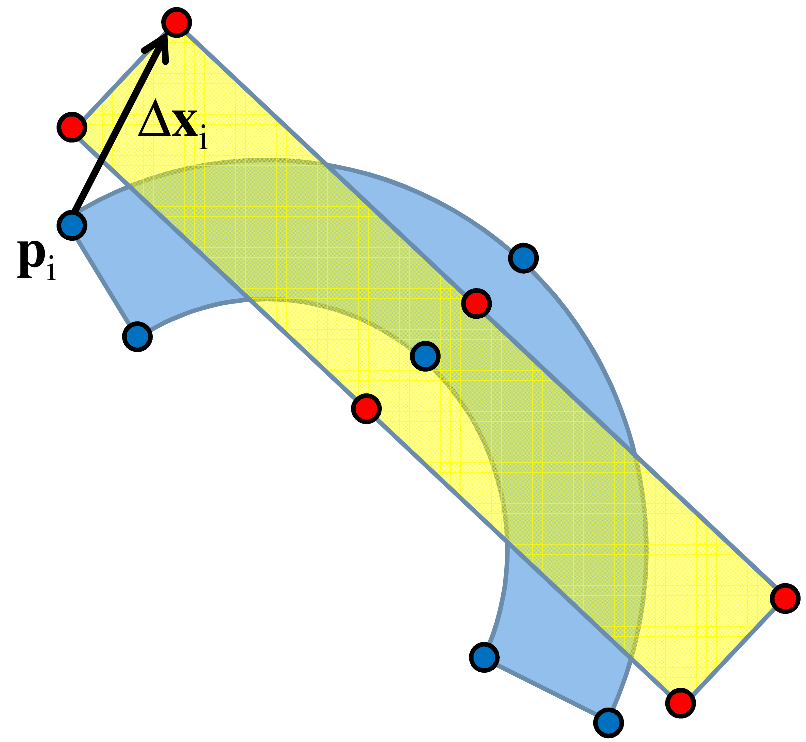
$$\Delta \mathbf{x}_i = -s w_i \nabla_{\mathbf{x}_i} C(\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

- Scalar s tells us how far to go

$$s = \frac{C(\mathbf{x}_1, \ldots, \mathbf{x}_n)}{\sum_j w_j \left| \nabla_{\mathbf{x}} C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \right|^2}$$

Shape Matching Idea

- Optimally match undeformed with deformed shape
- Only allow translation and rotation
- Global correction, no propagation needed
- No mesh needed!



Shape Matching

- Let $\mathbf{x}_i$ be the undeformed vertex positions
- The optimal translation is

$$\mathbf{t} = \mathbf{p}_{cm} - \mathbf{x}_{cm} \quad \text{where} \quad \mathbf{p}_{cm} = \sum_i m_i \mathbf{p}_i / \sum_i m_i \text{ and } \mathbf{x}_{cm} = \sum_i m_i \mathbf{x}_i / \sum_i m_i$$

- The optimal linear transformation is

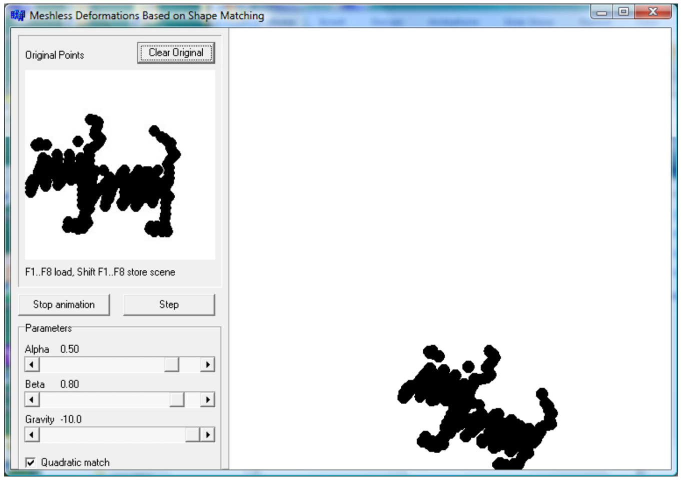
$$\mathbf{A} = \left(\sum_i m_i (\mathbf{p}_i - \mathbf{p}_{cm})(\mathbf{x}_i - \mathbf{x}_{cm})^T \right) \left(\sum_i m_i (\mathbf{x}_i - \mathbf{x}_{cm})(\mathbf{x}_i - \mathbf{x}_{cm})^T \right)^{-1}$$

- The optimal rotation $\mathbf{R}$ is the rotational part of $\mathbf{A}$ (use polar decomposition)

PhysX by NVIDIA

SIGGRAPH2008

2d Shape Matching Demo



PhysX™ by NVIDIA

SIGGRAPH2008

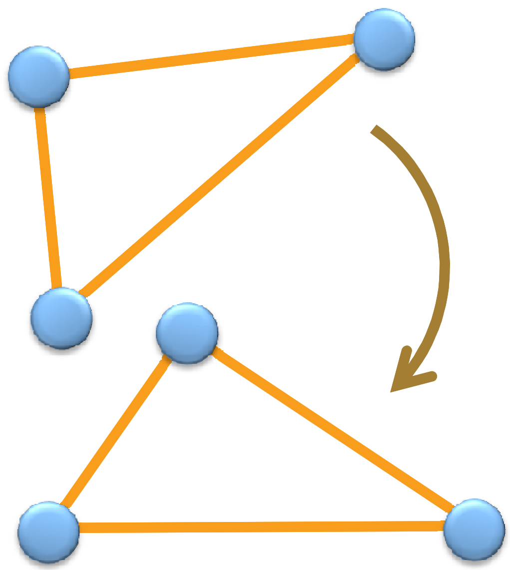
Working with Points and Edges

- No notion of volume or area
  - Spring stiffness (N/m) not related to 3d stiffness (N/m^2)
- Volumetric behavior dependent on
  - Tesselation of volume
  - Hand tune spring stiffnesses
- Often OK in real time environments
  - Evenly tesselated physics meshes
  - Fixed time step

PhysX™ by NVIDIA

SIGGRAPH2008

Co-Rotated Finite Elements

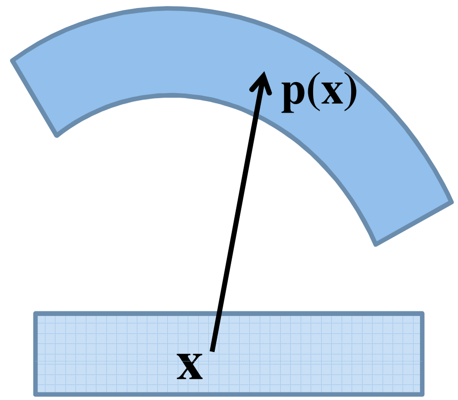


Continuum Mechanics on one Slide

- Body as continuous set of points
- Deformation continuous function $\mathbf{p}(\mathbf{x})$
- Elasticity theory yields $\mathbf{f}_{\mathrm{elast}}(\mathbf{x})$ from $\mathbf{p}(\mathbf{x})$
- PDE of motion (Newton):

$$\rho\,\mathbf{p}_{tt}(\mathbf{x},t) = \mathbf{f}_{elast}(\mathbf{x},t) + \mathbf{f}_{ext}(\mathbf{x},t)$$

- Solve for $\mathbf{p}(\mathbf{x},t)$
- Analytical solution only for very simple problems

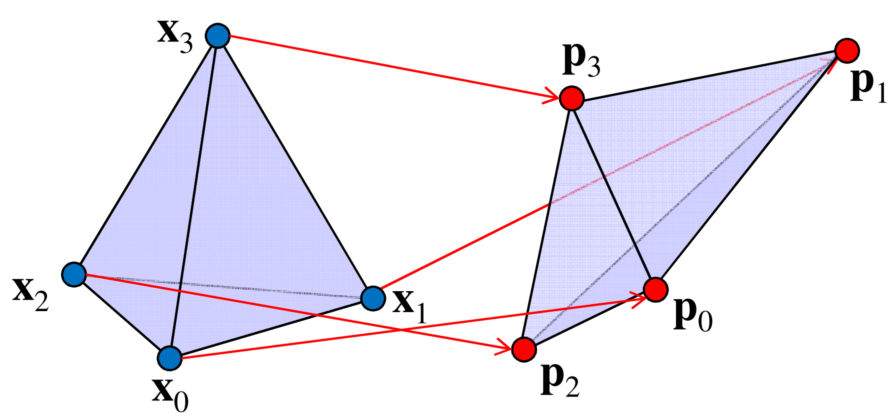


PhysX by NVIDIA

SIGGRAPH2008

Finite Element Method on one Slide

- Represent body by set of finite elements (tetrahedra)
- Represent continuous $\mathbf{p}(\mathbf{x})$ by vectors $\mathbf{p}_i$ on vertices



- $\mathbf{p}_i$ induce simple continuous $\mathbf{p}(\mathbf{x})$ within each element
- Continuous elasticity theory yields forces at vertices

PhysX by NVIDIA

SIGGRAPH2008

Hyper Spring

- Vertex forces depend on displacements of all 4 vertices

$$[\mathbf{f}_0,\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3] = F_{tetra}(\mathbf{p}_0,\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$$

- Tetrahedron acts like a hyper spring
- Compare to: $[\mathbf{f}_0, \mathbf{f}_1] = F_{\text{spring}}(\mathbf{p}_0,\mathbf{p}_1,l_0)$
- Given $F_{\text{tetra}}()$ -blackbox, simulate as mass spring system
- $F_{\text{tetra}}()$ is non linear, expensive

Linearization

- Linearization $F_{\text{tetra}}()$ of yields

$$\begin{bmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{p}_0 - \mathbf{x}_0 \\ \mathbf{p}_1 - \mathbf{x}_1 \\ \mathbf{p}_2 - \mathbf{x}_2 \\ \mathbf{p}_3 - \mathbf{x}_3 \end{bmatrix}, \quad \mathbf{K} \in \mathbf{R}^{12\times 12}$$

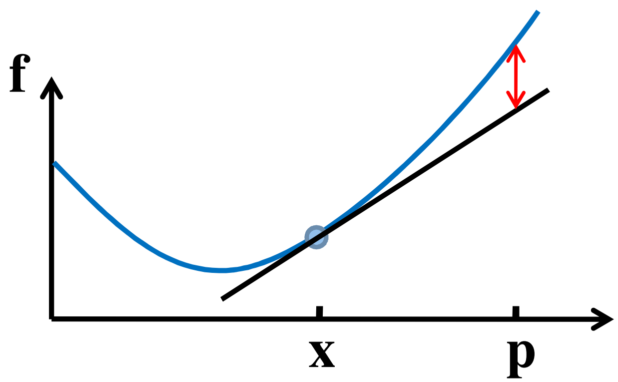
- $\mathbf{K}$ depends on $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and can be pre-computed
(see class notes for how to compute)
- Much faster to evaluate

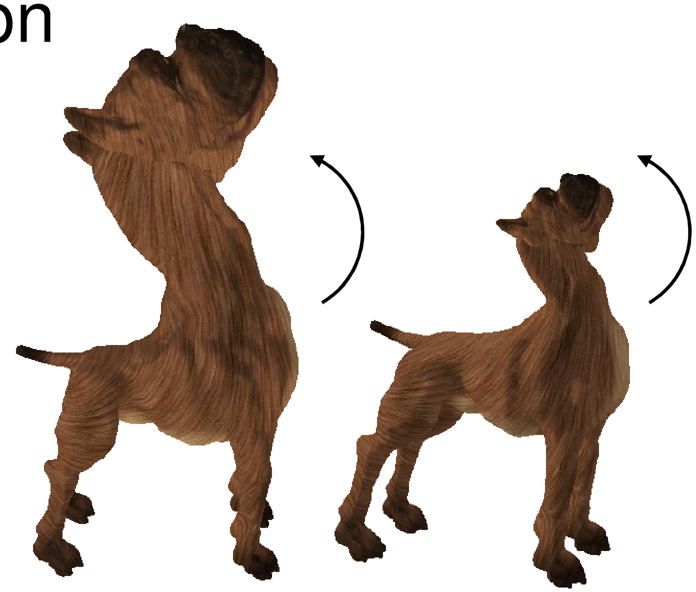
PhysX by NVIDIA

SIGGRAPH2008

Linearization Artifact

- Linearization only valid close to the point of linearization





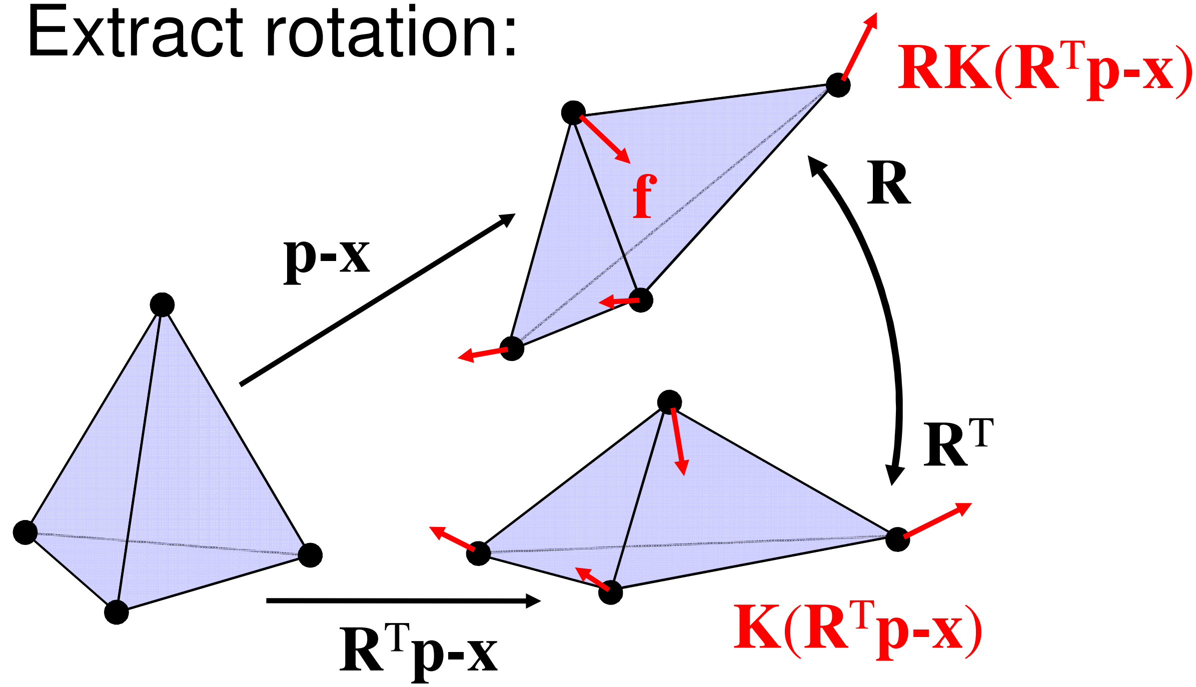
linearized non-linear

PhysX by NVIDIA

SIGGRAPH2008

Corotational Formulation

- Only rotations problematic, translations OK
- Extract rotation:



PhysX by NVIDIA

SIGGRAPH2008

Rotational Part

- Modified force computation

$$\begin{bmatrix}\mathbf{f}_0\\ \mathbf{f}_1\\ \mathbf{f}_2\\ \mathbf{f}_3\end{bmatrix} = \begin{bmatrix}\mathbf{R} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{R} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{R} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}\end{bmatrix}\mathbf{K}\left(\begin{bmatrix}\mathbf{R}^T\mathbf{p}_0\\ \mathbf{R}^T\mathbf{p}_1\\ \mathbf{R}^T\mathbf{p}_2\\ \mathbf{R}^T\mathbf{p}_3\end{bmatrix} - \begin{bmatrix}\mathbf{x}_0\\ \mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\end{bmatrix}\right)$$

- Transformation matrix

$$\mathbf{A} = [\mathbf{p}_1 - \mathbf{p}_0, \mathbf{p}_2 - \mathbf{p}_0, \mathbf{p}_3 - \mathbf{p}_0][\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0, \mathbf{x}_3 - \mathbf{x}_0]^{-1}$$

- Rotation via polar decomposition of $\mathbf{A}$

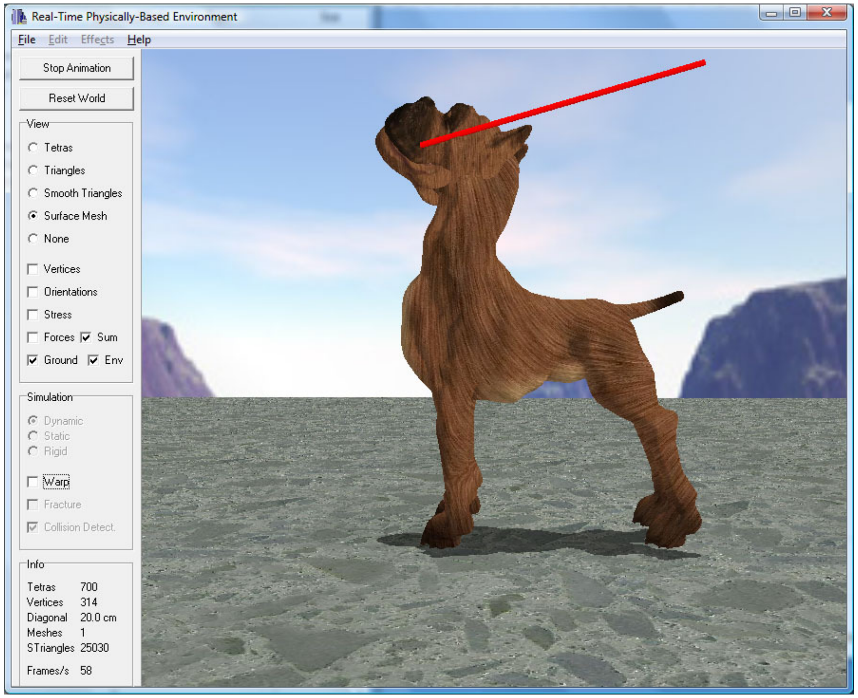
PhysX by NVIDIA

SIGGRAPH2008

Advantages

- Matrix $\mathbf{K}$ can still be precomputed
- Artifacts removed
- Faster force computation in explicit formulation
- Implicit time integration yields linear system $\rightarrow$ no Newton-Raphson solver needed

FEM Demo



PhysX by NVIDIA

SIGGRAPH2008

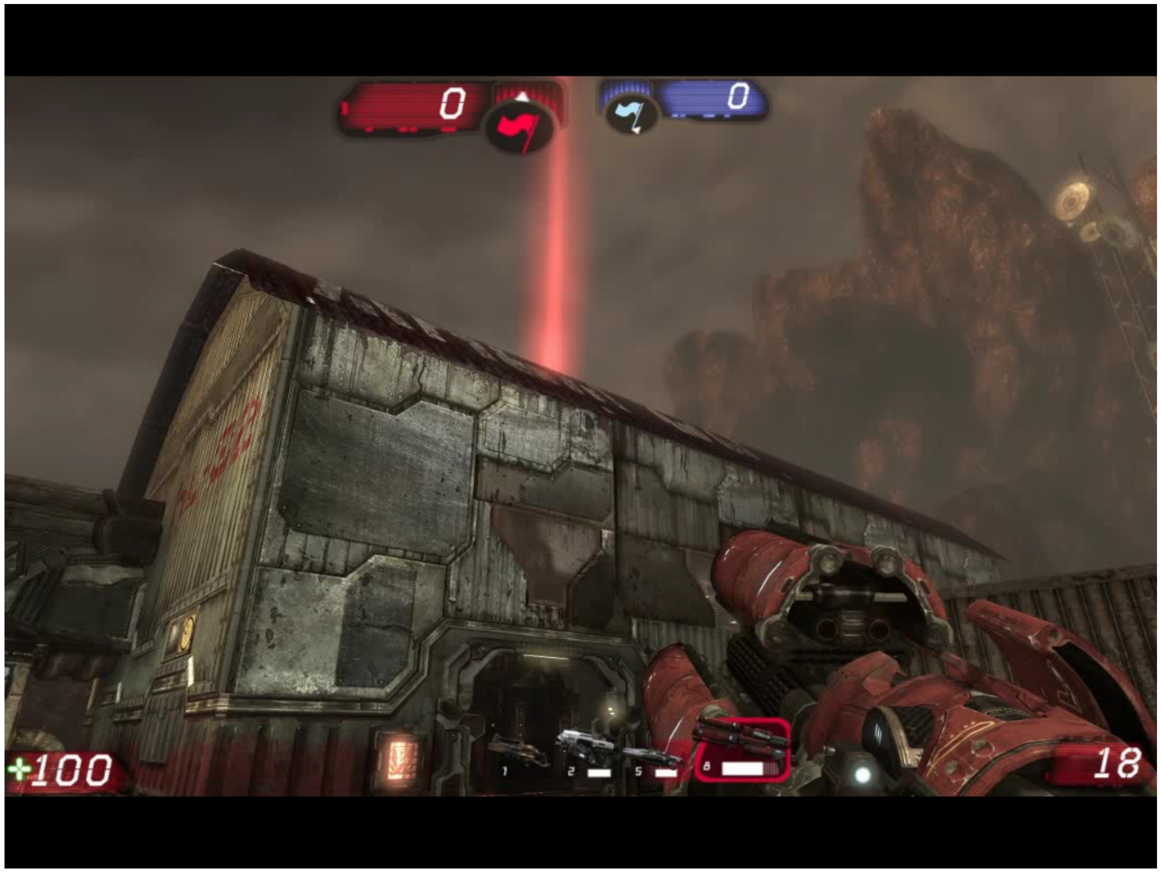
Conclusions

- Trade-off speed, accuracy, stability
- Choose method accordingly
- Stability most important in real time systems
  - Non predictable situations
  - No time step adaptions
  - No roll backs
- Remaining choice: accuracy vs. speed

PhysX™ by NVIDIA

SIGGRAPH2008

Cloth in Games



PhysX™ by NVIDIA

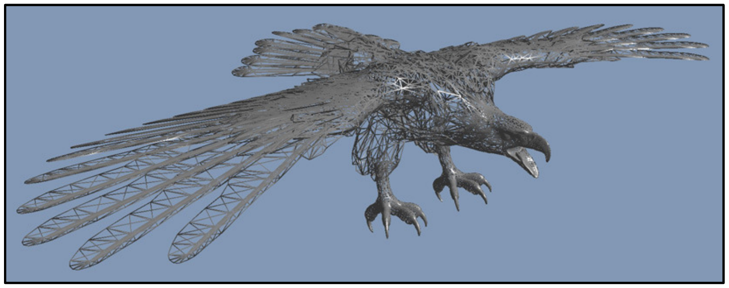
SIGGRAPH2008

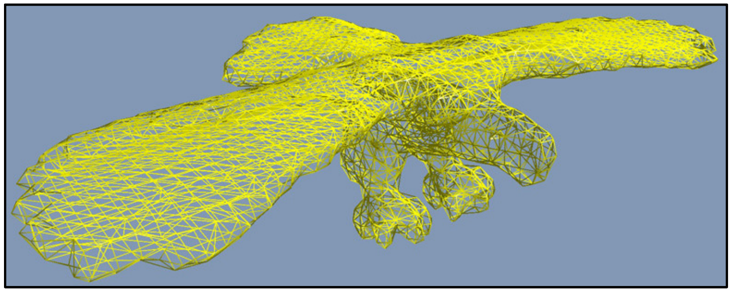
Mesh Generation

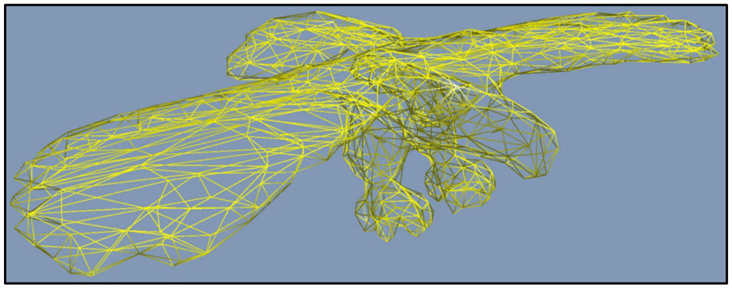
- Input
  - Graphical triangle surface mesh
  - Extreme case: Triangle soup
- Output
  - Input independent tesselation
  - User specify resolution (LOD)
  - Equally sized elements (stability, spatial hashing)

Surface Creation

- Input triangle mesh
- Each triangle adds density to a regular grid
- Extract iso surface using marching cubes
- Optional: Keep largest connected mesh only
- Quadric simplification





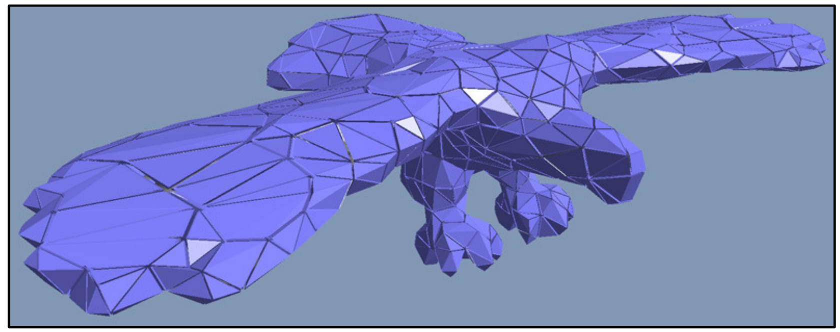


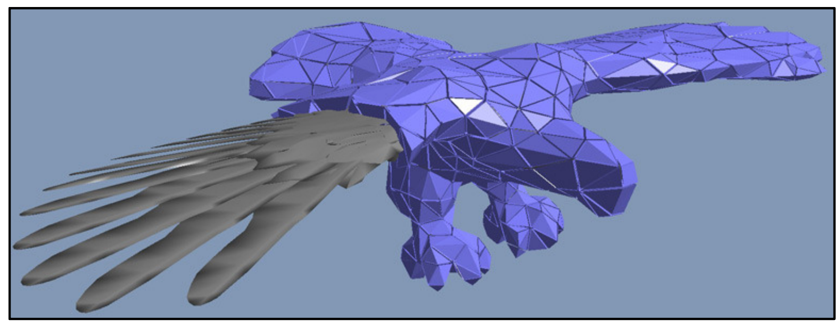
PhysX by NVIDIA

SIGGRAPH2008

Tetrahedra Creation

- Delaunay Tetrahedralization on vertices of surface mesh
- Triangles of surface mesh are used for clipping tetrahedra (if necessary)
- Graphical mesh is moved along with tetra mesh using barycentric coords





PhysX by NVIDIA

SIGGRAPH2008