Real Time Physics

www.matthiasmueller.info/realtimephysics

Matthias Müller  Doug James  Jos Stam  Nils Thuerey
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
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
1999

• Among my literature search:
  – D. James et al., *ArtDefo, Accurate Real Time Deformable Objects*, Siggraph 1999

• They brought physics brought to life!

• My assignment: make this real-time:
ArtDefo

- Boundary element method
- Haptic interaction
Doug James

• CV
  – 2001: PhD in applied mathematics, University of British Columbia
  – 2002: Assistant prof, Carnegie Mellon University
  – 2006: Associate prof, Cornell University
  – National Science Foundation CAREER award

• Research interests
  – Physically based animation
  – Haptic force feedback rendering
  – Reduced-order modeling
Stable Fluids

- Semi-Lagrangian advection
- Equation splitting
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
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
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
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
Dimensionality

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

- **Rope: chain**
  - Additional springs for *bending* and *torsional* resistance needed

- **Cloth: triangle mesh**
  - Additional springs for *bending* resistance needed

- **Soft body: tetrahedral mesh**
Mass Spring Physics

• Mass point: mass $m$, position $x$, velocity $v$

\[ \begin{bmatrix} x_m \end{bmatrix} \begin{bmatrix} v \end{bmatrix} \]

• Springs: $f \leftrightarrow -f$

\[ f = \frac{x_j - x_i}{|x_j - x_i|} \left[ k_s \left( |x_j - x_i| - l_0 \right) + k_d (v_j - v_i) \cdot \frac{x_j - x_i}{|x_j - x_i|} \right] \]

• Scalars $k_s$, $k_d$, stretching, damping coefficients
Time Integration

• Newton:

\[ \ddot{v} = \frac{f}{m} \]

\[ \dot{x} = v \]

• Explicit Euler:

\[ v_{i+1} = v_i + \Delta t \frac{1}{m_i} \sum_j f(x_i, v_i, x_j, v_j) \]

\[ x_{i+1} = x_i + \Delta t v_{i+1} \]

• Assumes velocity and force constant within \( \Delta t \)

• Correct would be:

\[ x(t + \Delta t) = x(t) + \int_{t}^{t+\Delta t} v(t) \, dt \]
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!

\[ \Delta t^2 f/m \]
Implicit Integration

• Use values of next time step on the right

\[ v_{i}^{t+1} = v_{i}^{t} + \Delta t \frac{1}{m_{i}} \sum_{j} f(x_{i}^{t+1}, v_{i}^{t+1}, x_{j}^{t+1}, v_{j}^{t+1}) \]

\[ x_{i}^{t+1} = x_{i}^{t} + \Delta t v_{i}^{t+1} \]

• Intuitively
  – Don’t extrapolate blindly
  – Arrive at a physical configuration
Implicit Integration Issues

• Unconditionally stable (for any $\Delta t$)!
• 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
• Temporal details disappear, numerical damping
Position Based Dynamics
Force Based Update

- Reaction lag
- Small $k_s \rightarrow$ squaschy 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 2\textsuperscript{nd} order system!
Position Based Integration

Init all \( x_i^0, v_i^0 \)

Loop

\[
\begin{align*}
    p_i^{t+1} &= x_i^t + \Delta t \cdot v_i^t \quad \text{// prediction} \\
    x_i^{t+1} &= \text{modify } p_i \quad \text{// position correction} \\
    u_i &= [x_i^{t+1} - x_i^t] / \Delta t \quad \text{// velocity update} \\
    v_i^{t+1} &= \text{modify } u_i \quad \text{// velocity correction}
\end{align*}
\]

End loop

- Explicit, Verlet related
- If correction done by a solver \( \rightarrow \) semi implicit
Position Correction

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

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

\[ \Delta x_1 = -\frac{w_1}{w_1 + w_2} \left( |x_1 - x_2| - l_0 \right) \frac{x_1 - x_2}{|x_1 - x_2|} \]

\[ \Delta x_2 = +\frac{w_2}{w_1 + w_2} \left( |x_1 - x_2| - l_0 \right) \frac{x_1 - x_2}{|x_1 - x_2|} \]

\[ w_i = 1/m_i \]

- 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:

\[ C_{\text{stretch}}(x_1, x_2) = |x_1 - x_2| - l_0 \]
\[ C_{\text{volume}}(x_1, x_2, x_3, x_4) = [(x_2 - x_1) \times (x_3 - x_1)] \cdot (x_4 - x_1) - 6v_0 \]
General Position Correction

- Correction along gradient
  \[ \Delta x_i = -sw_i \nabla_x C(x_1, \ldots, x_n) \]

- Scalar \( s \) tells us how far to go
  \[ s = \frac{C(x_1, \ldots, x_n)}{\sum_j w_j |\nabla_x C(x_1, \ldots, x_n)|^2} \]
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 $x_i$ be the undeformed vertex positions.
- The optimal translation is
  \[ t = p_{cm} - x_{cm} \]
  where
  \[ p_{cm} = \frac{\sum_i m_i p_i}{\sum_i m_i} \]
  and
  \[ x_{cm} = \frac{\sum_i m_i x_i}{\sum_i m_i} \]

- The optimal linear transformation is
  \[
  A = \left( \sum_i m_i (p_i - p_{cm})(x_i - x_{cm})^T \right)
  \left( \sum_i m_i (x_i - x_{cm})(x_i - x_{cm})^T \right)^{-1}
  \]

- The optimal rotation \( R \) is the rotational part of \( A \)
  (use polar decomposition)
2d Shape Matching Demo
Working with Points and Edges

• No notion of volume or area
  – Spring stiffness (N/m) not related to 3d stiffness (N/m²)

• Volumetric behavior dependent on
  – Tessellation of volume
  – Hand tune spring stiffnesses

• Often OK in real time environments
  – Evenly tesselated physics meshes
  – Fixed time step
Co-Rotated Finite Elements
Continuum Mechanics on one Slide

- Body as continuous set of points
- Deformation continuous function $p(x)$
- Elasticity theory yields $f_{elast}(x)$ from $p(x)$
- PDE of motion (Newton):
  \[
  \rho \frac{\partial^2 p}{\partial t^2}(x, t) = f_{elast}(x, t) + f_{ext}(x, t)
  \]
- Solve for $p(x,t)$
- Analytical solution only for very simple problems
Finite Element Method on one Slide

- Represent body by set of finite elements (tetrahedra)
- Represent continuous $p(x)$ by vectors $p_i$ on vertices

- $p_i$ induce simple continuous $p(x)$ within each element
- Continuous elasticity theory yields forces at vertices
Hyper Spring

• Vertex forces depend on displacements of all 4 vertices

\[ [f_0, f_1, f_2, f_3] = F_{\text{tetra}}(p_0, p_1, p_2, p_3, x_0, x_1, x_2, x_3) \]

• Tetrahedron acts like a hyper spring

• Compare to: \([f_0, f_1] = F_{\text{spring}}(p_0, p_1, l_0)\)

• Given \(F_{\text{tetra}}()\) -blackbox, simulate as mass spring system

• \(F_{\text{tetra}}()\) is non linear, expensive
Linearization

• Linearization $F_{tetra}$ of yields

$$
\begin{bmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3
\end{bmatrix} = K
\begin{bmatrix}
p_0 - x_0 \\
p_1 - x_1 \\
p_2 - x_2 \\
p_3 - x_3
\end{bmatrix}, \quad K \in \mathbb{R}^{12 \times 12}
$$

• $K$ depends on $x_0, x_1, x_2, x_3$ and can be pre-computed (see class notes for how to compute)

• Much faster to evaluate
Linearization Artifact

- Linearization only valid close to the point of linearization.
Corotational Formulation

- **Only rotations** problematic, translations OK
- **Extract rotation:**

\[
\begin{align*}
\text{p-x} & \quad \Rightarrow \quad \text{K}(R^T\text{p-x}) \\
R^T\text{p-x} & \quad \Rightarrow \quad \text{RK}(R^T\text{p-x})
\end{align*}
\]
Rotational Part

- Modified force computation

\[
\begin{bmatrix}
  f_0 \\
  f_1 \\
  f_2 \\
  f_3
\end{bmatrix} = \begin{bmatrix}
  0 & 0 & 0 & 0 \\
  0 & R & 0 & 0 \\
  0 & 0 & R & 0 \\
  0 & 0 & 0 & R
\end{bmatrix} \begin{bmatrix}
  R^T p_0 \\
  R^T p_1 \\
  R^T p_2 \\
  R^T p_3
\end{bmatrix} - \begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  x_3
\end{bmatrix}
\]

- Transformation matrix

\[
A = [p_1 - p_0, p_2 - p_0, p_3 - p_0][x_1 - x_0, x_2 - x_0, x_3 - x_0]^{-1}
\]

- Rotation via polar decomposition of \( A \)
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
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
Cloth in Games
Mesh Generation

• Input
  – Graphical triangle surface mesh
  – Extreme case: Triangle soup

• Output
  – Input independent tessellation
  – 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
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**