REAL-TIME FEM AND TRESSFX 4

ERIC LARSEN
KARL HILLESLAND
Simulates soft to nearly-rigid objects, with fracture

- Models object as mesh of tetrahedral elements
- Each element has material parameters:
  - Young’s Modulus: How stiff the material is
  - Poisson’s ratio: Effect of deformation on volume
  - Yield strength: Deformation limit before permanent shape change
  - Fracture strength: Stress limit before the material breaks
MOTIVATIONS FOR THIS METHOD

- Parameters give a lot of design control
- Can model many real-world materials
  - Rubber, metal, glass, wood, animal tissue
- Commonly used now for film effects
  - High-quality destruction
- Successful real-time use in Star Wars: The Force Unleashed 1 & 2
  - DMM middleware [Parker and O’Brien]
OUR PROJECT

• New implementation of real-time FEM for games

• Planned CPU library release
  – Heavy use of multithreading
  – Open-source with GPUOpen license

• Some highlights
  – Practical method for continuous collision detection (CCD)
  – Mix of CCD and intersection contact constraints
  – Efficient integrals for intersection constraint
STATUS

- Proof-of-concept prototype
- First pass at optimization
- Offering an early look for feedback
- Several generic components
Find time of impact between moving objects
- Impulses can prevent intersections [Otaduy et al.]
- Catches collisions with fast-moving objects

Our approach
- Conservative-advancement based
- Geometric solution for degeneracies
- Allows gap between primitives
- Can limit steps or accuracy
INTERSECTION-BASED CONTACT

- Preventing all intersection is expensive
  - High solver accuracy
  - Multiple iterations of CCD, solving
- Can handle intersection with volume constraint
  [Allard et al.]
  - Need integrals over intersection surface faces
- Our approach
  - Sum over edges and vertices of intersection
    - Found during BVH traversal
    - No explicit intersection surface
  - Dependent on topologically robust polyhedral intersection
    [Smith and Dodgson]
SIMULATION PIPELINE

Update Tet State:
- Rotation
- Stiffness
- Plasticity state
- Stress

Fracture Mesh:
- Split at tetrahedron faces
- Update mesh topology

Integration Solve:
- Implicit integration
- Assemble system matrix
- Solve with PCG

Build BVHs:
- Based on Radeon Rays
- Axis-aligned boxes with velocity bounds for CCD

Mesh Collision Detection:
- Compare mesh pair BVHs
- Sum integrals for intersection contacts
- CCD for distance contacts

Find contact islands:
- Group contacting bodies

Broad Phase:
- Find potentially colliding meshes
- Uses scene BVH

Contact Solve:
- Create Jacobian matrix from contacts
- Nested Gauss-Seidel solves
- Correct velocity and step
OPTIMIZATIONS

Multithreading optimizations
- Across meshes for tet state update, fracture, integration solve, BVH build
- Across mesh pairs for CCD and intersection contact generation
- Across contact island solves
- Some parallelism within contact island

Memory optimization
- Pre-allocated memory using bounds on maximum meshes, vertices, etc.
- Subdivision of memory for sub-meshes created by fracture
FUTURE WORK

- Optimization
  - Parallelism on large islands and meshes
  - SIMD

- Improvements to solver

- Integrating FEM with rigid-bodies or cloth
  - Solver can support other dynamics models
  - Transition between FEM and rigid

- Content pipeline

- Rendering
WOOD FRACTURE EXAMPLE
REFERENCES

- Baraff and Witkin, “Large Steps in Cloth Simulation” 1998
- Müller and Gross, “Interactive Virtual Materials” 2004
- Otaduy et al., “Implicit Contact Handling for Deformable Objects” 2009
- Miguel and Otaduy, “Efficient Simulation of Contact Between Rigid and Deformable Objects” 2011
- Allard et al., “Volume Contact Constraints at Arbitrary Resolution” 2012
- Smith and Dodgson, “A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic” 2006
- McAdams et al., “Computing the Singular Value Decomposition of 3x3 Matrices with Minimal Branching and Elementary Floating Point Operations” 2011
- Parker and O’Brien, “Real-Time Deformation and Fracture in a Game Environment” 2009
WHAT'S NEW IN TRESSFX 4.0 SIMULATION

- SDF (Signed Distance Field)
- SSH (Sudden Shock Handler)
- Bone-based skinning
- Improved art tool
- Code Improvements
**SIGNED DISTANCE FIELD**

Input: Triangles
Output: 3D Grid of values

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>0.7</td>
<td>0.6</td>
<td>0.5</td>
<td>0.5</td>
<td>1.8</td>
</tr>
<tr>
<td>1.1</td>
<td>0.1</td>
<td>-0.4</td>
<td>-0.5</td>
<td>-0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>1.8</td>
<td>0.8</td>
<td>0.1</td>
<td>-0.8</td>
<td>-0.8</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.3</td>
<td>-0.3</td>
<td>0.1</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>0.2</td>
<td>0.1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Negative = inside
Positive = outside
Empty = too far
Input: Hairs + Grid
Output: Moved Hairs

- Check distance vs margin
- Compute gradient
- Project
// Compute SDF Gradient using forward differencing

// Project hair vertex out of SDF
float3 normal = normalize(sdfGradient);

if (distanceAtVertex < g_CollisionMargin)
{
    float3 projectedVertex =
        hairVertex.xyz + normal * (g_CollisionMargin - distanceAtVertex);
    g_HairVertices[hairVertexIndex].xyz = projectedVertex;
    g_PrevHairVertices[hairVertexIndex].xyz = projectedVertex;
}
SDF CHARACTERISTICS

△ SDFs Independent of hair objects
  – Separate according to resolutions
  – Separate according to updates

△ Cost proportional to
  – SDFs / hair
  – Number of hairs

△ Cost of applying independent of mesh density
Three SDFs (body-130x162x120, hands-46x45x36)

Per frame generation from the skinned mesh (red) for moving character. ~2.0ms in our demo.

Collision checking and response takes ~0.5ms
SIGNED DISTANCE FIELD EXTRAS

- For static objects, no cost for generation.
- Can be used for other purposes such as cloth.
- Can be visualized using Marching Cubes (yellow) for debugging.
SIMULATION

- Update collision mesh
- Update bone matrices
- Simulate
- Update with SDFs
FAST MOTION PROBLEM

Teleportation

User Movements

“Fixed” using

- Increased iterations
- Parameter tweaks
- Hair “reset”
SSH (SUDDEN SHOCK HANDLER)

- Propagate acceleration
- Linear and Rotational
- Weight and Threshold
- Weight = 1
  - Skinning only
  - LOD
SKINNING CHANGES

Maya® Exporter
- Computes weights from mesh
- Exports bone names
- Exports weights

Can remap bone indices to match engine
- Optional (if Maya indices = bone indices, you’re done)
- Runtime or preprocess

Hair roots skinned directly
- No intermediate mesh

Bonus: Export skinned mesh (for SDF generation)
CODE IMPROVEMENTS

- Graphics through callbacks
  - GPU resource allocation, etc
  - “layouts” and “bindsets”
  - Transitions

- Shader code encapsulated in functions

- DX12 support
  - Async example
THANK YOU
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

©2016 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, [insert all other AMD trademarks used in the material here per AMD’s Checklist for Trademark Attribution] and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.