

SAN ANTONIO SIGGRAPH #2002#

SAN ANTONIO SIGGRAPH #2002#

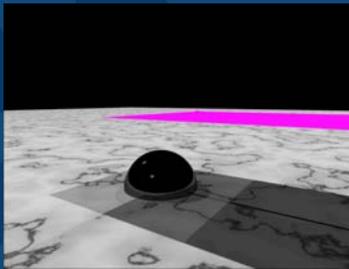
Soft Things

Robust Treatment of Collisions, Contact
and Friction for Cloth Animation

Robert Bridson (presenting),
Ronald Fedkiw and John Anderson

An example of our algorithm

- 1.2 million triangles
- Thousands of contacts per timestep
- ~1 day of computation on a laptop



First problem: sheer size

- **Every node is on the surface, surface folds easily**
 - 10,000+ collisions per time step easily possible when cloth folds over onto itself
 - Too expensive to resolve in time order

Second problem: low tolerance

- **Cloth is very thin – once it interpenetrates, it pops out the other side**
 - In most interesting folding, effect is too severe to try to untangle after the fact
- **Need to stop all interpenetration**

One solution: repulsion forces

- **Put a repelling force-field around cloth**
 - e.g. Terzopoulos et al, Moore & Wilhelms, Carignan et al, Laffeur et al, Baraff & Witkin



- Good for automatically handling contact
- If set correctly, models cloth thickness and compressibility (e.g. the fuzz on a towel)
- **When resolved**, smooth and accurate

Problems with repulsion forces

- **Not robust**
 - Can miss multiple or fast collisions
 - Once on wrong side, pushes the wrong way
- **Partial fix: increase size and strength**
 - Makes cloth “float”
 - Numerical difficulties

Another solution: geometric collisions

- **Consider trajectories over timestep, find all collisions, apply impulses**
 - e.g. Provot '97
 - If rounding error properly handled, never misses a collision



Problems with geometric collisions

- **Difficult to resolve multiple collisions simultaneously**
 - Fixing one may cause others...
 - Expensive to iterate too long
- **Triangles resist sliding over each other**
 - Catastrophic error: “chainmail” friction inconsistent with physics



How can do we do better?

- **Combine the two approaches!**
 - First apply repulsion forces – quickly and accurately handles almost everything
 - Check new trajectories geometrically, eliminate all remaining intersections
- **Well-conditioned and bullet-proof, almost as cheap as repulsions alone**

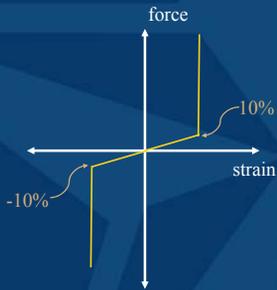
Time stepping

- **Advance x and v (internal cloth dynamics)**
 - $(x^n, v^n) \rightarrow (x^{n+1}, v^{n+1})$
- **Get average v over step**
 - $v^{n+1/2} = (x^{n+1} - x^n) / \Delta t$
- **Adjust $v^{n+1/2}$ for repulsions/friction**
- **Adjust $v^{n+1/2}$ to resolve all geometric collisions**
- **Get new x from modified $v^{n+1/2}$**
 - $x^{n+1} = x^n + \Delta t v^{n+1/2}$
- **Advance modified v (internal cloth dynamics)**
 - $v^{n+1} = v^{n+1/2} + \frac{1}{2} \Delta t a(x^{n+1}, v^{n+1})$

Internal cloth dynamics

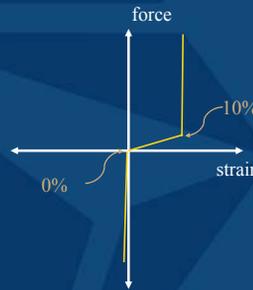
- **Could be anything! (Use existing code)**
 - One implicit step, many Runge-Kutta steps, masses and springs, finite elements, ...
- **We use masses and springs, Provot '95**
 - Additional impulses limit excess strain rate: helps keep cloth together after collisions

Limiting strain



- Like Provot '95, apply impulses to limit strain
- Implicit integration (Gauss-Seidel) of **biphasic** springs

Limiting strain



- Like Provot '95, apply impulses to limit strain
- Implicit integration (Gauss-Seidel) of **biphasic** springs
- **Zero compression**
 - Causes buckling
 - See next talk...

Repulsion forces

- Check for triangle/point, edge/edge at old positions
- Limit repulsion to a fraction of cloth thickness – eliminate “kicks”
- Normal force gives Coulombic friction
 - If v_T is tangential velocity before friction, Δv_N is normal repulsion impulse, then
$$v_T^{\text{friction}} = \max(|v_T| - \mu \Delta v_N, 0) v_T / |v_T|$$

Resolving geometric collisions

- Use Provot '97:
 - Apply inelastic collision impulses
 - Check for additional collisions
 - After 3 rounds of impulses, solidify inter-colliding patches into rigid “impact zones”
- To prevent cloth creeping through with round-off error, enforce minimum separation

Subdivision

- Sharp folds barely resolved in simulation
 - Unacceptable for rendering
- Can subdivide mesh in each frame
 - We use Loop
- Convex-hull property helps, but... **self-intersections** and **object-intersections** may be introduced

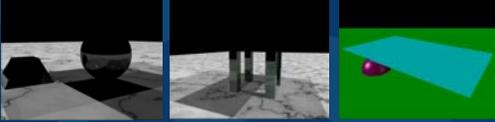
Collision-aware subdivision

- Modify subdivision to avoid collisions
 - Start with refined mesh (linear rule)
 - Move to smooth subdivision positions
 - Check “motion” for collisions, scale down “velocities”



Results

- Minutes per frame on a laptop, 15k-40k simulation nodes, subdivided twice



Thanks!

- Igor Neverov, Neil Molino, Joey Teran, Henrik Wann Jensen
– rendering examples
- Sebastian Marino, Cliff Plumer, Andy Hendrickson, and Lucasfilm
– Yoda