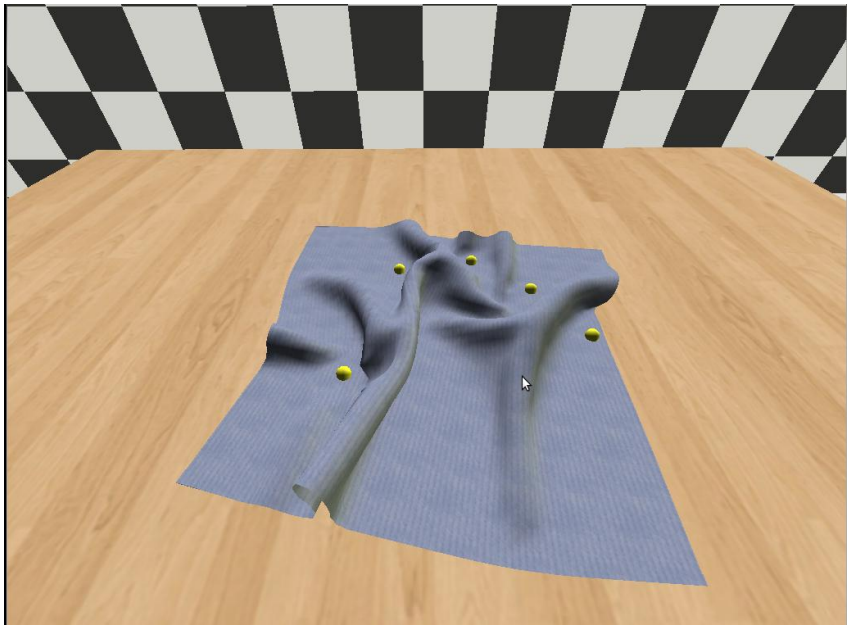
Cloth Simulation and Manipulation



Ken Toh

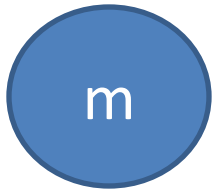
21 Feb 2011

Contents

- Physically-based Simulation: An Overview
- Cloth Representation
- Collision
- Manipulation
- Cloth Manipulation with Multi-Touch

Physically-based Simulation

- Model the phenomenon as a collection of particles (ie. particle system)
- Particle has the following physical quantities:
- ***x***: position
- ***v***: velocity
- ***a***: acceleration
- ***m***: mass
- **f**: force



Newton's 2nd Law: a = (1/**m**) **f**

Euler Integration

Simplest scheme but might not be stable for big time steps!

- The goal is always to find the new position and velocity of the particle after each new time step *dt*.

- First, sum up all the forces acting on the particle. Then use N2L to find the acceleration:
  - $\mathbf{a} = (1/\mathbf{m})\,\mathbf{f}$

- Using the acceleration, we can in turn find the new velocity and position by using Euler's:
  - $\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \boldsymbol{a}_t\,dt$
  - $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{v}_{t+1}\,dt$

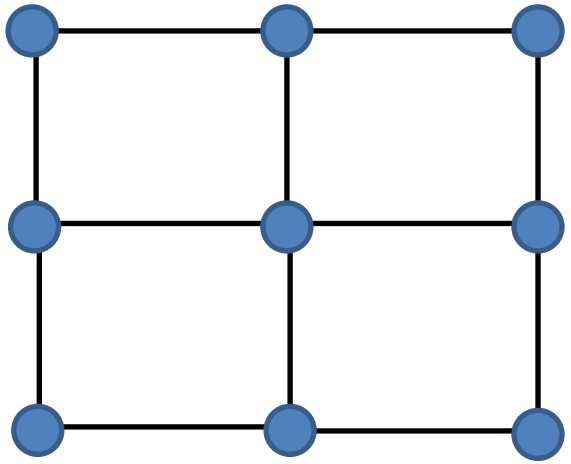
How about a whole particle system?

- General steps for simulating a particle system:
- For each particle:
  - Find and accumulate forces acting on particle
  - Compute acceleration $\boldsymbol{a}$ using Newton's 2nd Law
  - Integrate to get **x** and $\boldsymbol{v}$.
  - Update new **x** and **v**, and zero out forces

  Repeat again during next time step…

Cloth

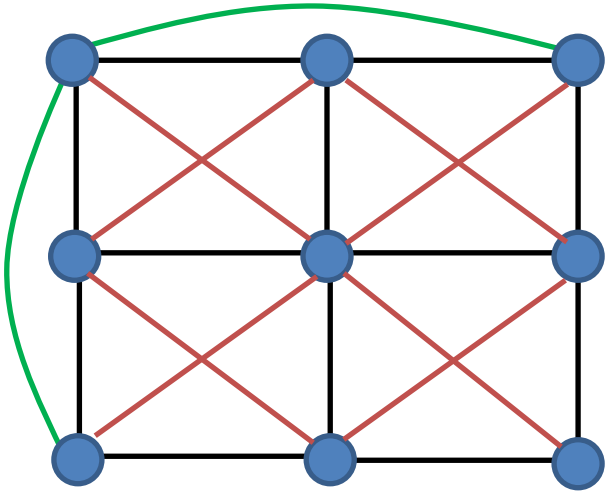
- Spring-based Cloth
- A structured lattice of particles inter-connected by spring dampers



What's wrong with this simple representation?

Cloth Properties

- Want to capture resistive properties of cloth:
- Resistance To:
- Stretching (Black)
- Bending (Green)
- Shearing (Red)



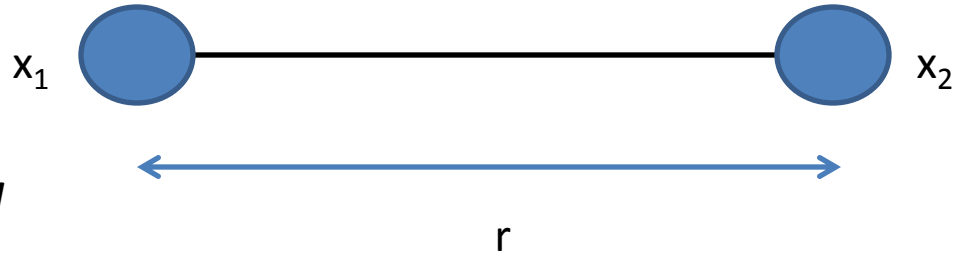
Cloth Simulation

At each time step, loop through each particle:

- Add forces acting on particle
  - Gravity force ($-9.81ms^{-2}$)
  - drag force, user-defined forces, etc
- Also compute the spring damper forces acting on each particle and add them
- Integrate to find new positions and velocities

Spring Damper Forces

- Each spring connects two particles.
- Important properties:
  - Spring Constant $\mathbf{k_s}$
  - Damping Constant $\boldsymbol{k_d}$
  - Rest Length $\boldsymbol{r}$



$$\mathbf{f}_1 = -\left[k_s(|\Delta\mathbf{x}| - r) + k_d\left(\frac{\Delta\mathbf{v}\cdot\Delta\mathbf{x}}{|\Delta\mathbf{x}|}\right)\right]\frac{\Delta\mathbf{x}}{|\Delta\mathbf{x}|}$$

$$\mathbf{f}_2 = -\mathbf{f}_1$$

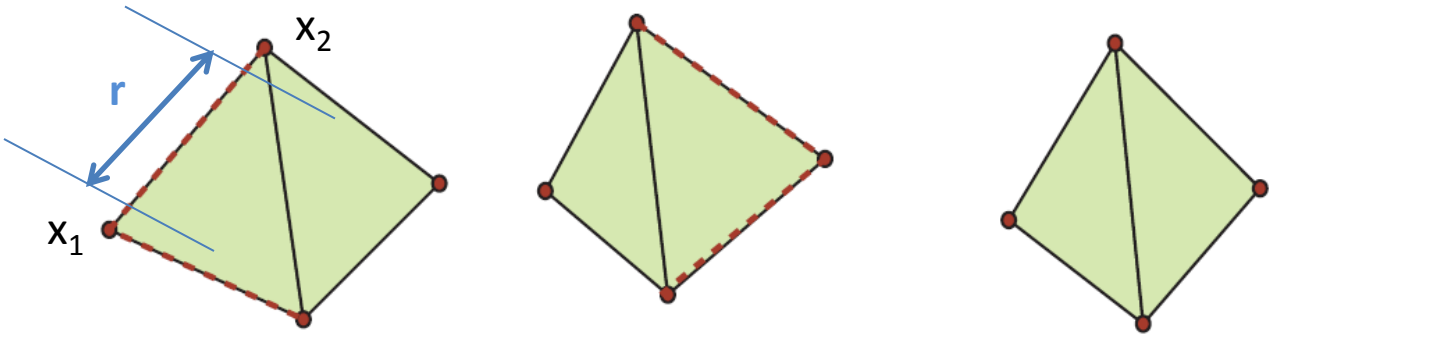
For each damper, compute the force it exerts on each particle in the pair.

Constraint-based (Inextensible) Cloth

- **Force** based spring dampers tend to lead to rather stretchy cloth. Stiff springs are problematic
- Use **position** constraints instead. After each time step, check for constraint violations and attempt to satisfy these constraints iteratively. (Provot [1995])
- Used this in my project!

Iterative Constraint Satisfaction

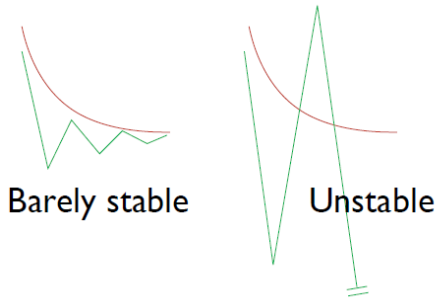
- Iteratively "fix" each constraint that is violated a few times after each time step.



$$\Delta \mathbf{x}_1 = \frac{1}{2}(|\mathbf{x}_2 - \mathbf{x}_1| - r)\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

$$\Delta \mathbf{x}_2 = -\Delta \mathbf{x}_1$$

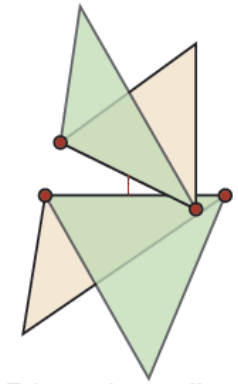
Remarks on stability



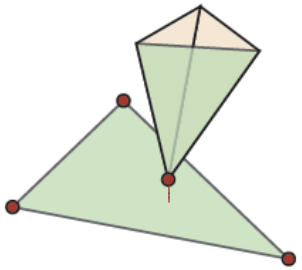
- Euler (Explicit) Integration can easily become unstable (overshooting!) unless we run it with very tiny time steps.
- Simulation can still blow up due to energy gains despite our attempts to introduce controlled damping
- Other methods of integration to improve stability: Verlet, Midpoint, Runge Kutta 2, Runge Kutta 4, etc.

Collision Overview

- **Detection**
- Typically, check for co-planarity of 4 points
- Use acceleration schemes

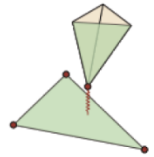


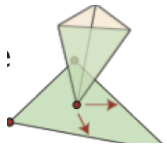
Edge-edge collision

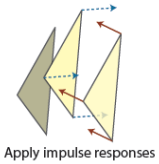


Point-face collision

- **Response**
- Penalty forces – use springs (hard to tune)
- Constraint-based response (only pt-face)
- Impulse based response (mult. Iterations)







Apply impulse responses

Interactive Cloth Manipulation

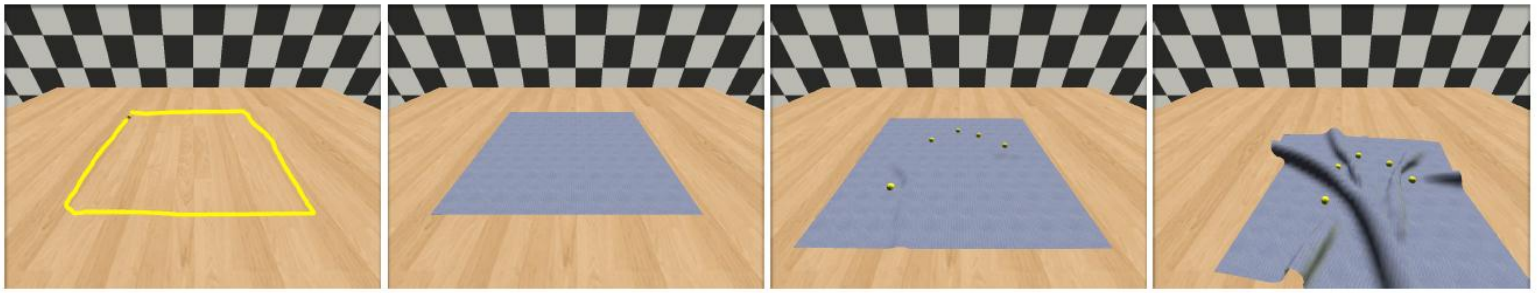


Figure 1: The user first creates a cloth by drawing its shape on the scene. The user then manipulates the cloth using multiple sticky fingers.

- **Verlet Integration and Position-based constraint satisfaction** (often used in games as well!)
- Supports pinching, folding, pinning, tearing, draping, lifting, crumpling, spreading, rotation, translation etc
- Note that these manipulations require multiple finger-control

Simulation Model

- Verlet Integration and Position-based Iterative Constraint Satisfaction (mentioned earlier)
- Verlet is velocity-less!
- Fast, much more stable than Euler and works directly with positions
- Keep an extra variable x_{t-1} : old position
- Velocity approximated by $x_t - x_{t-1}$

$$\boldsymbol{x}_{t+1} = 2\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\, dt^2$$

Cloth Manipulation With Sticky Fingers

- Direct manipulation of positions important
- It takes time for applied spring forces to act
- Solitary mouse cursor inadequate for manipulation
- Solution: Multiple “sticky” fingers
- User Interface: Multi-Touch

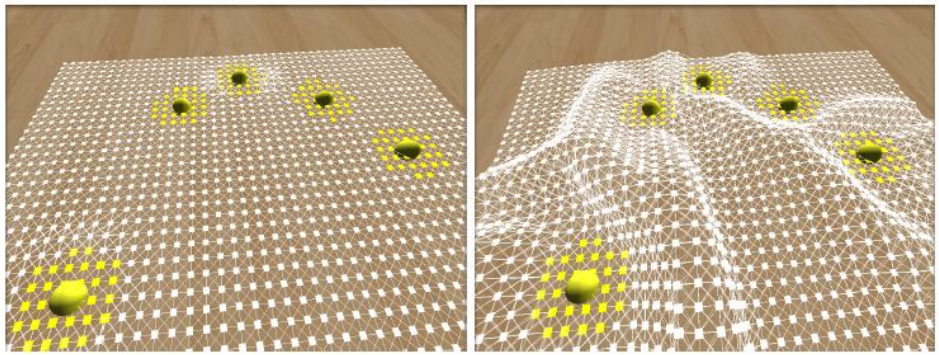


Figure 5: Sticky Fingers: Underlying cloth particles within a radius from the center of an active finger are stuck to that finger and move with it.

Tearing

- Easy way: break springs/remove constraints that are overstretched. But may lose “area”

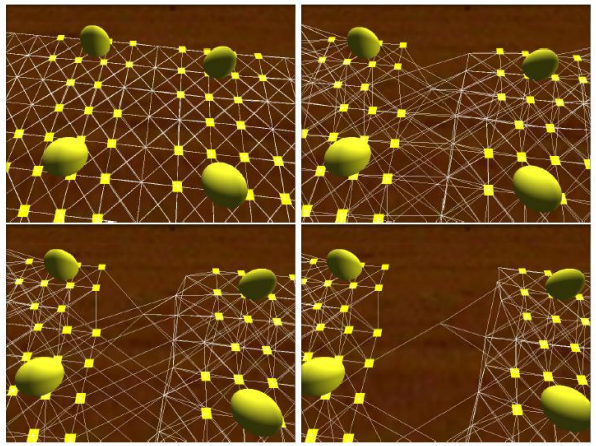
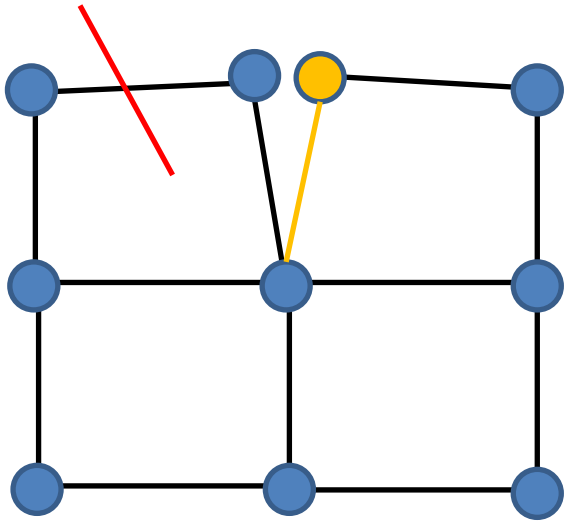


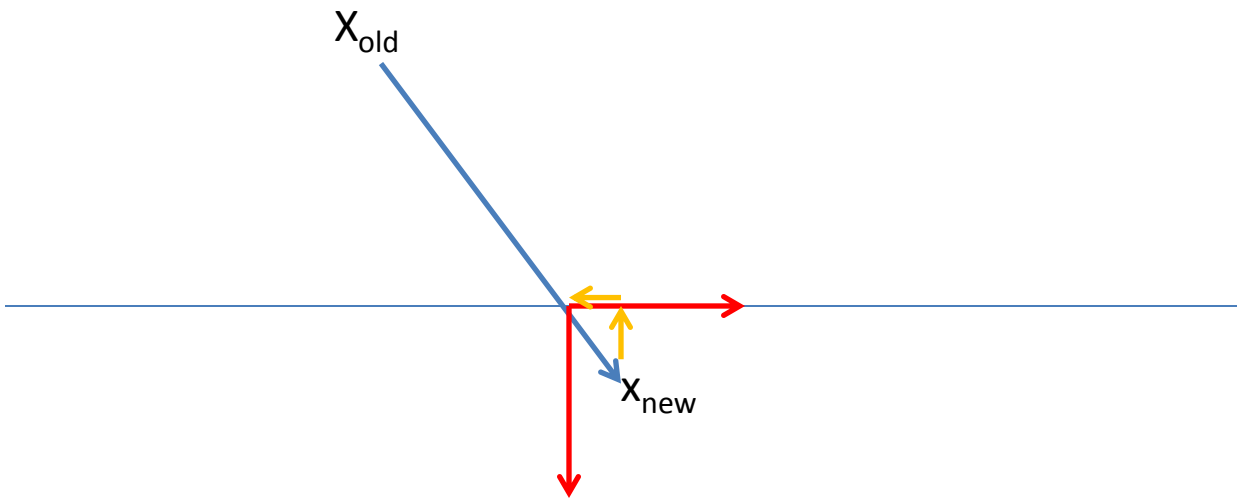
Figure 12: Simple mesh tearing is achieved by removing constraints that are ”over-stretched” beyond an allowed distance threshold

- More accurate way: node splitting.



Handling Collision

- Self collisions ignored
- Simple projection of particle to geometry surface upon collision
- Scale tangential component of response by a suitable coefficient of friction.
- Attenuate projected x_{new} with the scaled tangential component



Demos