Real-time Cloth Simulation

Sean Oxley

SUNYIT

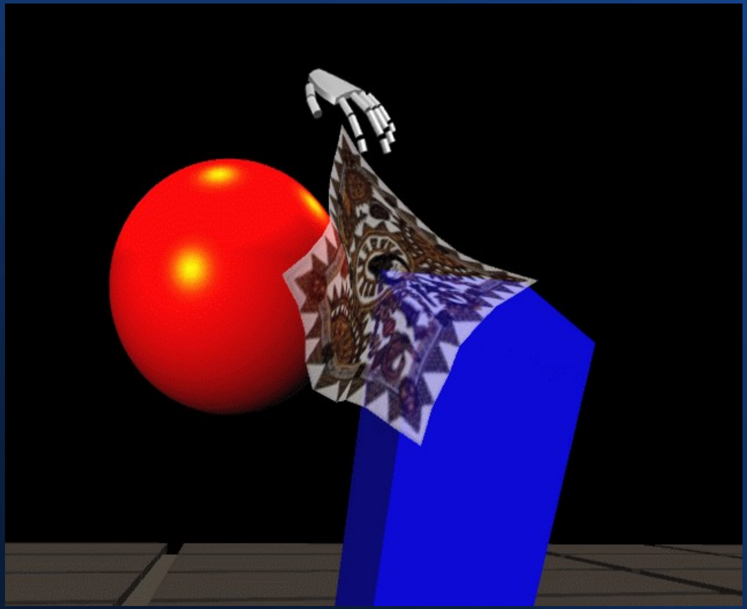
Hudson River Undergraduate Mathematics Conference 2013

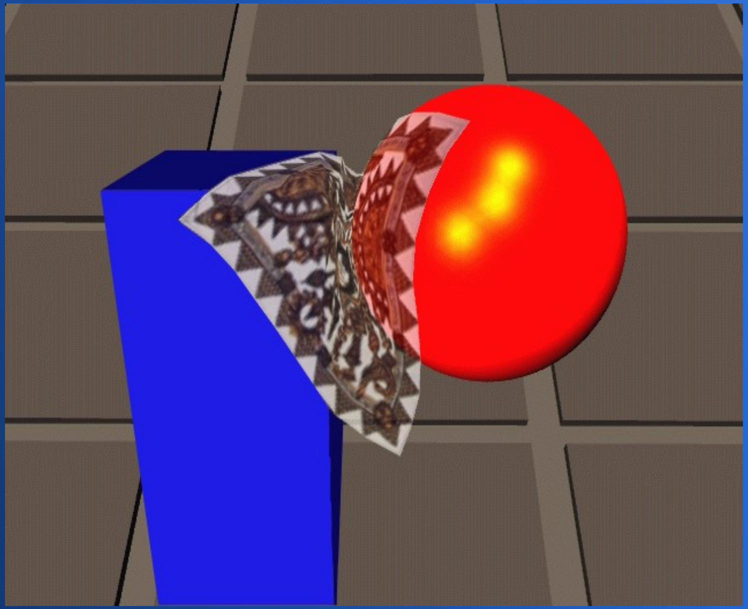
April 6, 2013

Referenced papers

- *Interactive animation of structured deformable objects.* Mathieu Desbrun, Peter Schroder, Alan Barr. Caltech.
- *An Introduction to Physically Based Modeling: An Introduction to Continuum Dynamics for Computer Graphics.* Michael Kass. Pixar.

Cloth simulation in action





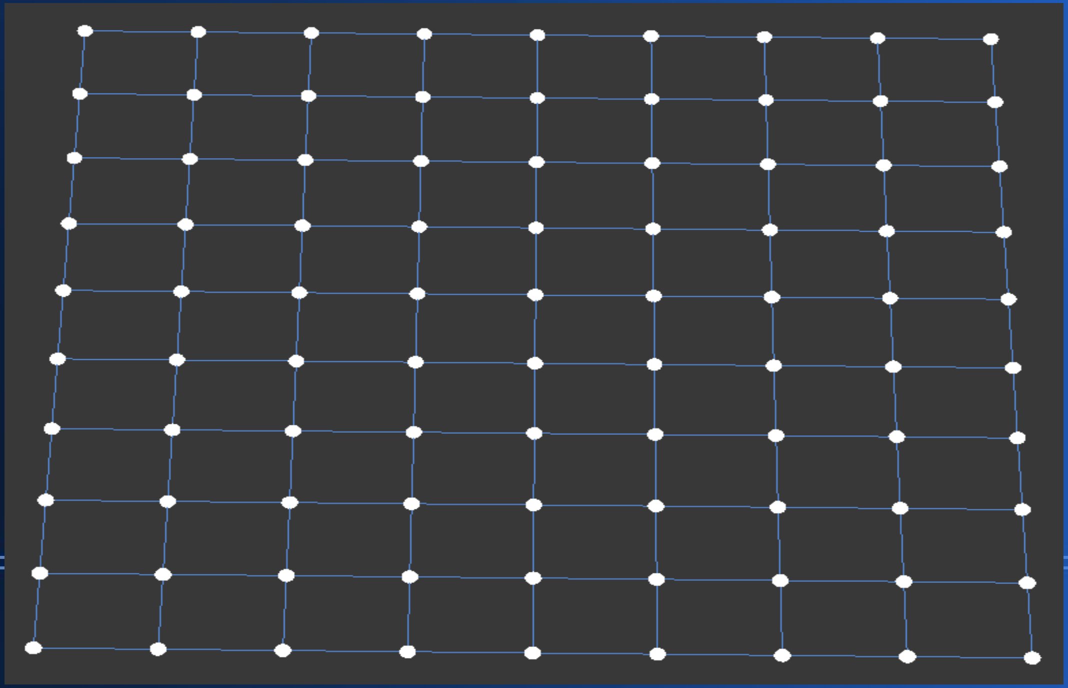
Pictures from *Interactive Animation of Structured Deformable Objects*

The objective

- Algorithm for simulating cloth
  - Preferably extendable to all soft bodies
- Real-time performance (absolute requirement)
- Visual plausibility (accuracy not so much)
- Stability

Mass-spring model

- Break a cloth up into many evenly spaced, distinct points (called mass points)
- Often these are the vertices of the polygons that are used to render the cloth



Mass-spring model (cont.)

- Then, connect adjacent points with springs.
- But why masses and springs?
  - Allows each mass point to deform individually while maintaining connection with others
  - Simple, fast, well-suited to multi-core computers

So how do we move this thing?

- This is the hard (and mathematical) part
- In short, we want to simulate two laws:
  - Newton's Second Law: $F = ma$
  - Hooke's law: $F = -kx$
- We use these two laws to compute the positions of the mass points in the system

Approximating the integral

- Let's say we want to get positions from F = ma
- The formula is F = mx'', we need x
- We want to integrate, but can't without some assumptions
- Remember the Riemann Integral?
- We simulate in distinct time steps with constant force

Simulating Newton's Second Law

- Explicit integration:
  - $x^{n+1} = x^n + v^{n+1}\, dt$
  - $v^{n+1} = v^n + F^n\, (dt/m)$
- Implicit integration:
  - $x^{n+1} = x^n + v^{n+1}\, dt$
  - $v^{n+1} = v^n + F^{n+1}\, (dt/m)$
- Implicit is what we want, but how do we compute F^{n+1}?

Calculating F^{n+1}

- The formula is $F^{n+1} = F^{n} + (\partial F/\partial x)(x^{n+1} - x^{n})$
- $(\partial F/\partial x)$ is actually Hooke's law in action
- Use a matrix H to act as $(\partial F/\partial x)$
- First consider 1D case:

$$H = k\begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Filtering

- We now have:
  - $F^{n+1} = F^n + H(x^{n+1} – x^n)$
- We use this to calculate the change in velocity
- We need one more: $x^{n+1} – x^n = v^{n+1}\, dt$
- $v^{n+1} – v^n = W(F^n + dtHv^n)(dt/m)$
- Where $W = (I – dt^2/m\ H)^{-1}$
- This W matrix is interesting: if we apply a force to a single point, for example, W takes care of moving its neighboring points appropriately. This is called filtering.

Extension to 2D and 3D

- Problem: H (and therefore W) is no longer constant
- We could recompute every time step, but slow
- Instead, we take *predictor-corrector* approach
- While predicting, H and W are constant again

Correcting angular momentum

- The liberties we took resulted in a loss of angular momentum (rotation)
- Loss of torque across mass points
- Too slow to fix them all individually
- Compute a global torque and apply to all mass points

Inverse dynamics

- At this point, we have an isolated mass-spring system modeled
- But we're modeling a cloth that's part of a bigger world
- We must compensate for constraints (collisions, desired material properties, etc.)

Inverse dynamics (cont.)

- Collisions
  - Ensuring cloth is not embedded in a surface
  - Friction, restitution
- Avoiding sagging
  - Enforcing a max spring length
  - Bring all offending points closer together

Summary

- Mass-spring system
- For each mass point
  - Predict its change in velocity
  - Correct it to preserve Nature's laws
  - Use this new velocity to move it
- Inverse dynamics