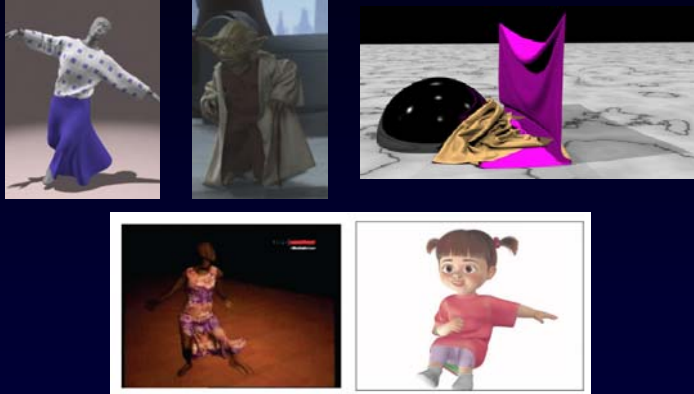


Cloth Simulation

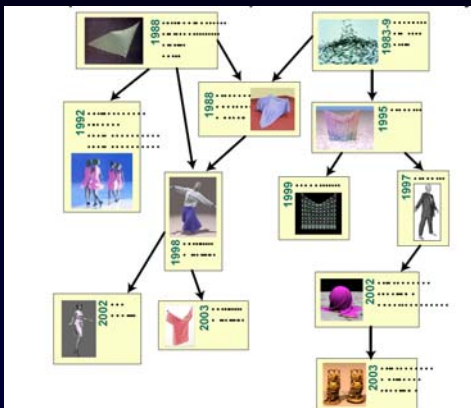


used slides are from Baraff and Witkin course notes
used slides from Chris Twigg presentation about cloth

Read

- Large Steps in Cloth Simulation (by Baraff and Witkin)
- Robust Treatment of Collisions, Contact and Friction for Cloth Animation (Bridson, Fedkiw, Anderson)

continuum
models



particle
systems

Equation for cloth

Specific details may vary but in all cases cloth simulation is formulated as time-varying partial differential equation

Cloth Internal Energy

Other Forces (air-drag,
contact and constraints....)

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1} \left(-\frac{\partial E}{\partial \mathbf{x}} + \mathbf{F} \right)$$

Cloth Simulation: main challenges

- Model
- Integration Scheme
- Constraint Handling
- Collision Handling

Model

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1} \left(-\frac{\partial E}{\partial \mathbf{x}} + \mathbf{F} \right)$$

Integration Scheme:

- Explicit
 - Euler
 - Midpoint
 - Runge-Kutta
- Implicit

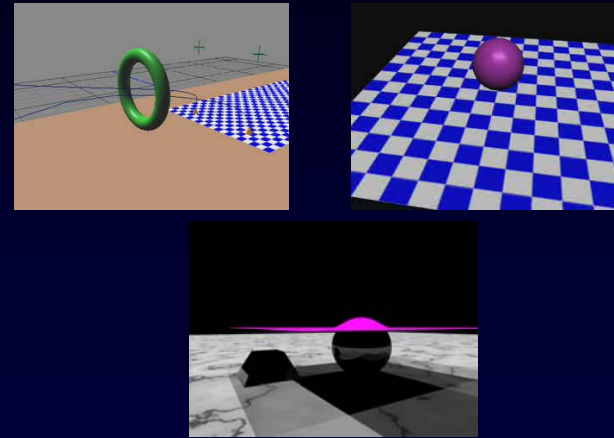
Constraints:

- Attach particle to fixed or moving point
- Constraint particle to a curve/surface
- Contact constraints between cloth and solid
 - attach to surface if a lot of friction
 - constrained to remain on surface with sliding allowed

Collision Handling:

- Cloth Object
- Self Collisions

Cloth Simulation: main challenges

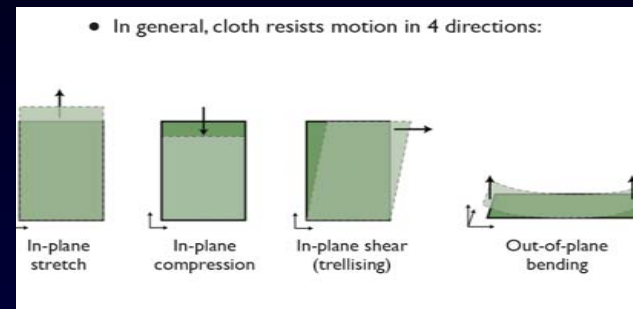


Cloth Simulation: main challenges

- Model
- Integration Scheme
- Constraint Handling
- Collision Handling

Model

- The most important forces → internal cloth forces



Model

Baraf / Witkin

Energy functions are defined in terms of a (heuristic) "soft" constraint function

Stretch: $C(\mathbf{x}) = a \left(\frac{\|w_u(\mathbf{x})\| - b_u}{\|w_v(\mathbf{x})\| - b_v} \right)$

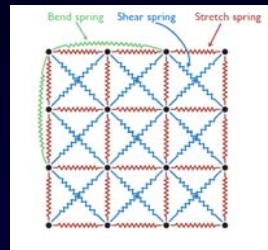
Shear: $C(\mathbf{x}) = a w_u(\mathbf{x})^T w_v(\mathbf{x})$

Bend: $C(\mathbf{x}) = \theta$

Now, energy and force are defined as $E_c(\mathbf{x}) = \frac{k}{2} C(\mathbf{x})^T C(\mathbf{x})$ $\mathbf{f}(\mathbf{x}) = -\frac{\partial E_c}{\partial \mathbf{x}}$

Bridson/Fedkiw/Anderson

Spring-based model



Cloth Simulation: main challenges

- Model
- Integration Scheme
- Constraint Handling
- Collision Handling
- Implementation Details

Integration Scheme

Baraf / Witkin

Implicit

Bridson/Fedkiw/Anderson

Any

Cloth Simulation: main challenges

- Model
- Integration Scheme
- Constraint Handling
- Collision Handling
- Implementation Details

- Constraints:
- Attach particle to fixed or moving point
 - Constraint particle to a curve/surface
 - Contact constraints between cloth and solid
 - attach to surface if a lot of friction
 - constrained to remain on surface with sliding allowed

Constraints Handling

- Reduced Coordinates
 - complicates the system
- Penalty Methods
 - create stiff system
 - can handle stiff system
 - do not enforce constraints exactly
- Lagrange Multipliers
 - turns positive-definite system into an indefinite
 - work well with explicit methods

Constraints Handling

- Mass Modification

$$\ddot{\mathbf{x}}_i = \frac{1}{m_i} \mathbf{f}_i$$

- To keep particle i velocity from changing \rightarrow set mass to infinity
- $1/m_i = 0$

$$\ddot{\mathbf{x}}_i = \begin{pmatrix} 1/m_i & 0 & 0 \\ 0 & 1/m_i & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{f}_i \quad \text{which constraint?}$$

Constraints Handling

- Does not have to be coordinate-aligned
- A particle is prevented from accelerating along p

$$(\mathbf{I} - \mathbf{p}\mathbf{p}^T)\mathbf{p} = \mathbf{0}$$

$$\frac{1}{m_i}(\mathbf{I} - \mathbf{p}\mathbf{p}^T)$$

$$\frac{1}{m_i}(\mathbf{I} - \mathbf{p}\mathbf{p}^T - \mathbf{q}\mathbf{q}^T)$$

- Build block diagonal matrix $\mathbf{W} \rightarrow \mathbf{W}_{ii} = \frac{1}{m_i} \mathbf{S}_i$

$$\mathbf{S}_i = \begin{cases} \mathbf{I} & \text{if ndof}(i) = 3 \\ (\mathbf{I} - \mathbf{p}_i\mathbf{p}_i^T) & \text{if ndof}(i) = 2 \\ (\mathbf{I} - \mathbf{p}_i\mathbf{p}_i^T - \mathbf{q}_i\mathbf{q}_i^T) & \text{if ndof}(i) = 1 \\ \mathbf{0} & \text{if ndof}(i) = 0. \end{cases}$$

Constraints Handling

- Rewrite to include constraints

$$\left(\mathbf{I} - h\mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2 \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \Delta \mathbf{v} = h\mathbf{M}^{-1} \left(\mathbf{f}_0 + h \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v}_0 \right)$$



$$\left(\mathbf{I} - h\mathbf{W} \frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2 \mathbf{W} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \Delta \mathbf{v} = h\mathbf{W} \left(\mathbf{f}_0 + h \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v}_0 \right)$$

Constraints Handling

- Rewrite to include constraints

$$\left(\mathbf{I} - h\mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2 \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \Delta \mathbf{v} = h\mathbf{M}^{-1} \left(\mathbf{f}_0 + h \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v}_0 \right)$$

- Velocity constraints

$z_i \rightarrow$ the change in velocity we wish to enforce in constrained direction

$$\left(\mathbf{I} - h\mathbf{W} \frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2 \mathbf{W} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \Delta \mathbf{v} = h\mathbf{W} \left(\mathbf{f}_0 + h \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v}_0 \right) + \mathbf{z}$$

Cloth Simulation: main challenges

- Model
- Integration Scheme
- Constraint Handling
- Collision Handling
- Implementation Details

Collision Handling:

- Cloth Object
- Self Collisions

Collision Handling

Baraf / Witkin

- Cloth Object \rightarrow constraints
- Self Collisions \rightarrow penalty forces

Bridson/Fedkiw/Anderson

The whole paper is about robust collision handling

Collision Handling

Robust Treatment of Collisions, Contact and Friction for Cloth Animation (Bridson, Fedkiw, Anderson)

Collisions are a major bottleneck in cloth simulation

Tens of thousands of particles or more

Cloth is very thin, even small interpenetrations can lead to cloth protruding from the wrong side

Large number of collisions

resting contact
high speed impact

Works with any method for simulating internal dynamics

Collision – common approach

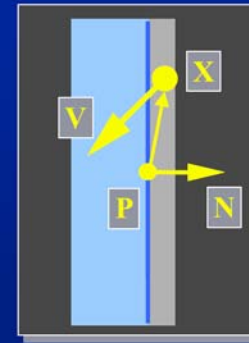
Collision detection (plane, sphere, cylinder, triangle)

Collision response

high velocity impact → impulse-based

contact → repulsion forces

Collision Detection



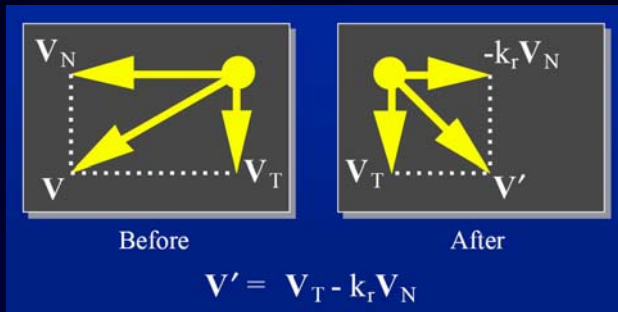
$$(\mathbf{X} - \mathbf{P}) \cdot \mathbf{N} < \epsilon$$

$$\mathbf{N} \cdot \mathbf{V} < 0$$

- Within ϵ of the wall.
- Heading in.

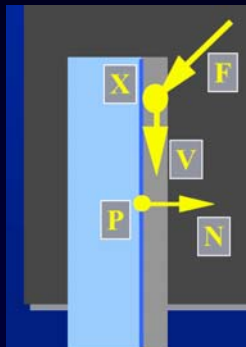
High velocity collision response

- Impulse-based



Contacts

- Repulsion Force



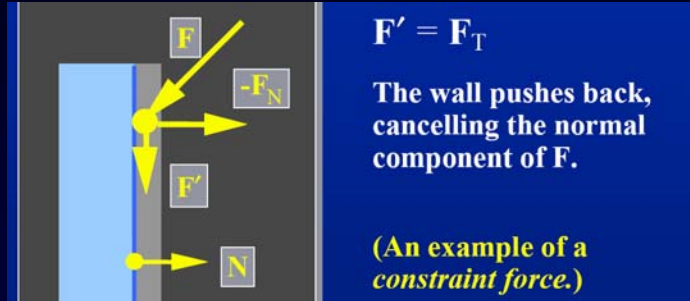
$$|(\mathbf{X} - \mathbf{P}) \cdot \mathbf{N}| < \epsilon$$

$$|\mathbf{N} \cdot \mathbf{V}| < \epsilon$$

- On the wall
- Moving along the wall
- Pushing against the wall

Collision

- Repulsion Force



Outline

Robust Treatment of Collisions, Contact and Friction for Cloth Animation (Bridson, Fedkiw, Anderson)

- Use repulsive forces to deal with majority of collisions
 - must be fast
 - use non-stiff spring model
- Use more expensive and robust methods to stop few remaining collisions
- First method to guarantee no dynamic self-interference of cloth
- Independent of cloth internal dynamics

Algorithm

- Collision time step size Δt $t^{n+1} = t^n + \Delta t$
- Integrate cloth equations $\bar{x}^{n+1} \quad \bar{v}^{n+1}$
- Compute average velocity $\bar{v}^{n+1/2} = (\bar{x}^{n+1} - x^n) / \Delta t$
- Check for proximity x^n
- Apply repulsion impulses and friction to average velocity $\tilde{v}^{n+1/2}$
- Check linear trajectory from x^n with $\tilde{v}^{n+1/2}$ for collisions and resolve them $v^{n+1/2}$
- Compute final position $x^{n+1} = x^n + \Delta t v^{n+1/2}$
- Compute final velocity $\begin{cases} v^{n+1} = \bar{v}^{n+1} & \text{if no repulsions or collisions} \\ v^{n+1} = v^{n+1/2} & \text{otherwise} \end{cases}$

Algorithm

- Collision time step size Δt
- Integrate cloth equations
- Check for proximity
- Apply repulsion impulses and friction
- Check linear trajectory from previous step for collisions and resolve them

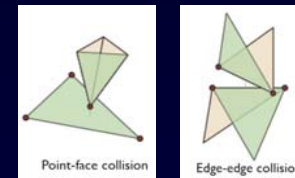
Algorithm

- Collision time step size Δt
- Integrate cloth equations
- Check for proximity
- Apply repulsion impulses and friction
- Check linear trajectory from previous step for collisions and resolve them

Check for proximity

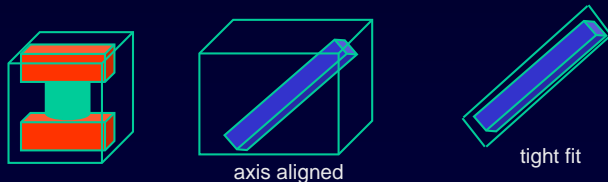
- Point is within small dist from triangle
- Edge is within small dist from other edge
- Build axis-aligned bounding box hierarchy

Review Hierarchical Bounding Volumes



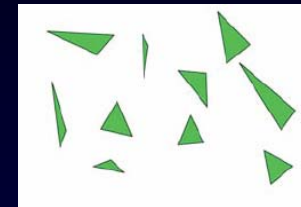
Hierarchical Bounding Volumes

- Wrap things that are hard to check for intersection in things that are easy to check
 - Example: wrap a complicated polygonal mesh in a box
 - Point can not be inside real object unless it is inside the box
 - Adds some overhead, but generally pays for itself.
- Most common bounding volume types: sphere and box
 - box can be axis-aligned or not



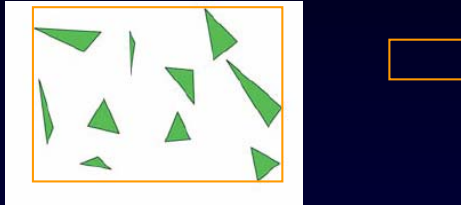
Hierarchical Bounding Volumes

- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones



Hierarchical Bounding Volumes

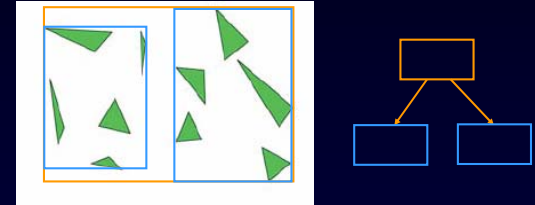
- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones



33

Hierarchical Bounding Volumes

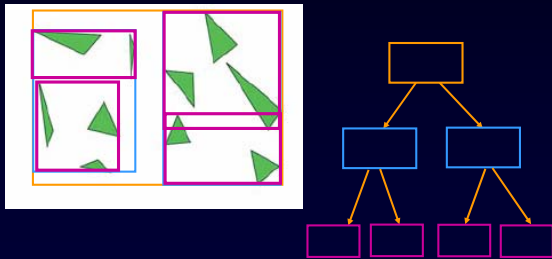
- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones



34

Hierarchical Bounding Volumes

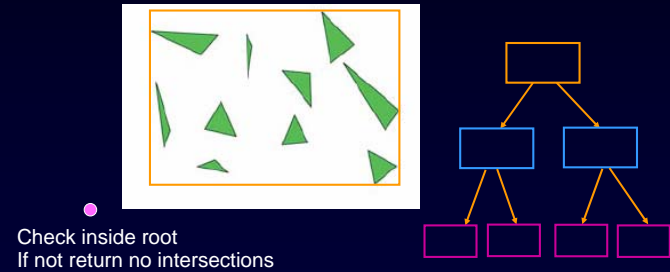
- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones



35

Hierarchical Bounding Volumes

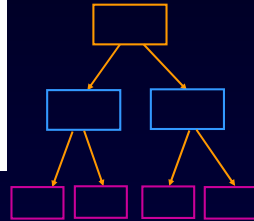
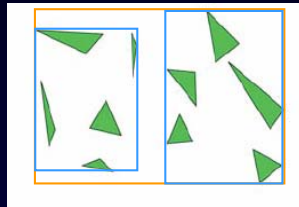
- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones



36

Hierarchical Bounding Volumes

- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones

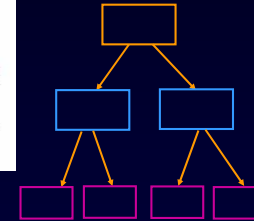
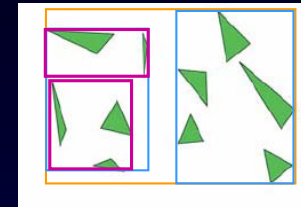


Check inside root
If inside
check inside left sub-tree
check inside right sub-tree

37

Hierarchical Bounding Volumes

- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones

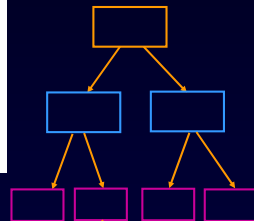
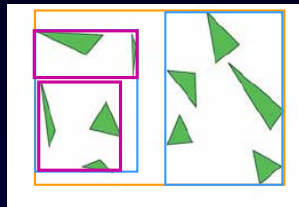


Check inside root
If inside
check inside left sub-tree
check inside right sub-tree

38

Hierarchical Bounding Volumes

- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones

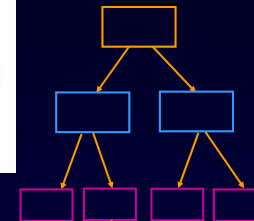
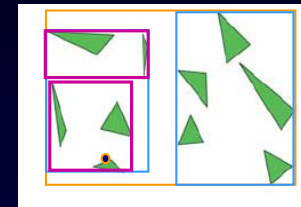


Check inside root
If inside
check inside left sub-tree
check inside right sub-tree

39

Hierarchical Bounding Volumes

- Still need to check point against every object --- $O(n)$
- Use tree data structure
 - Larger bounding volumes contain smaller ones

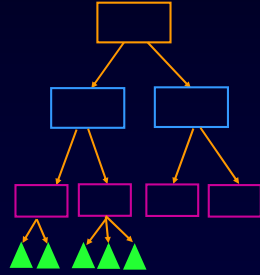
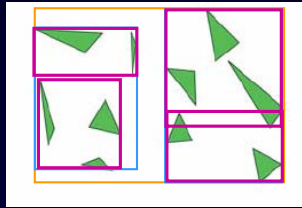


Check inside root
If inside
check inside left sub-tree
check inside right sub-tree

40

Hierarchical Bounding Volumes

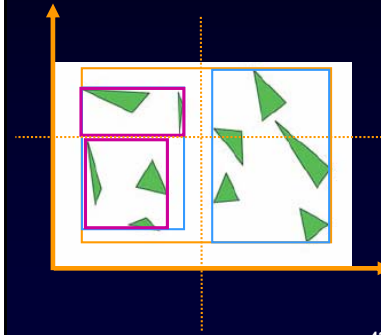
- Many ways to build a tree for the hierarchy
- Works well:
 - Binary
 - Roughly balanced
 - Boxes of sibling trees not overlap too much



41

Hierarchical Bounding Volumes

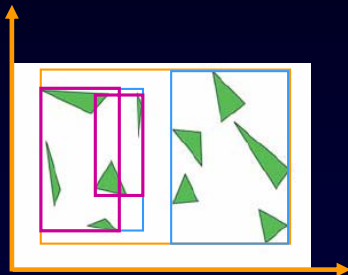
- Sort the surfaces along the axis before dividing into two boxes
- Carefully choose axis each time
- Choose axis that minimizes sum of volumes



42

Hierarchical Bounding Volumes

- Sort the surfaces along the axis before dividing into two boxes
- Carefully choose axis each time
- Choose axis that minimizes sum of volumes



43

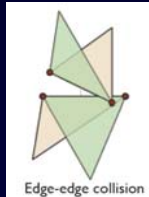
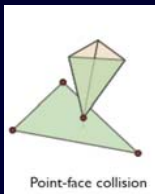
Hierarchical Bounding Volumes

- Works well if you use good (appropriate) bounding volumes and hierarchy
- Should give $O(\log n)$ rather than $O(n)$ complexity ($n = \#$ of objects)
- Can have multiple classes of bounding volumes and pick the best for each enclosed object

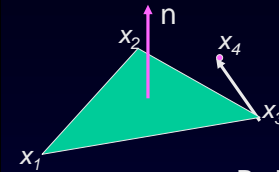
44

Axis-aligned bounding box hierarchy

- Once in the beginning
- Use bottom up approach
 - greedily pair adjacent triangles
 - pair these pairs in the next sweep
 - and so on
- Re-compute at each collision step
 - compute a box around triangle and its candidate position at the end of the step

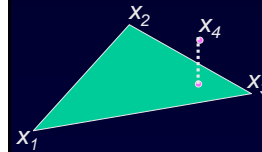


Check point close to triangle



$$|x_{43} \circ n| < h$$

Project x_4 onto triangle
 closest point in barycentric coordinates
 $x_1 w_1 + x_2 w_2 + x_3 w_3$

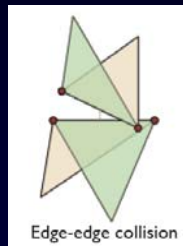


$$\begin{bmatrix} x_{13} \cdot x_{13} & x_{13} \cdot x_{23} \\ x_{13} \cdot x_{23} & x_{23} \cdot x_{23} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} x_{13} \cdot x_{43} \\ x_{23} \cdot x_{43} \end{bmatrix}$$

$$w_1 + w_2 + w_3 = 1$$

Check edge/edge collision

Find a pair of closest points on the edge
 Check their distance



Algorithm

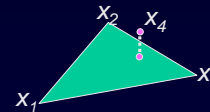
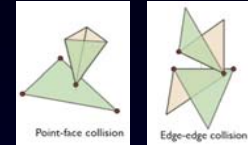
- Collision time step size Δt
- Integrate cloth equations
- Check for proximity
- Apply repulsion impulses and friction
- Check linear trajectory from previous step for collisions and resolve them

Repulsion forces

Resolving all self-collisions in the cloth is expensive
 Repulsive forces are mandatory
 Dramatically reduce number of collisions
 In many cases eliminate them

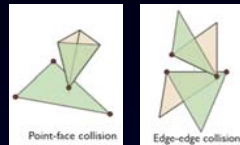
Repulsion forces

- Apply proximity detection:
 - point-triangle pair
 - edge-edge pair
- If pair is close enough apply repulsive force
 - inelastic impulse $I = mv_n/2$
 - spring based force (compression of cloth fibers)
 - proportional to overlap beyond cloth thickness



Friction

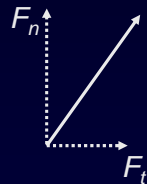
- Repulsive force is negative of normal force F_n that presses cloth together



- Coulomb's friction model

Static friction models the fact that it requires a larger force to get something moving than it does to keep it moving.

- Friction force at most $\rightarrow \mu F_n$



Algorithm

- Collision time step size Δt
- Integrate cloth equations
- Check for proximity
- Apply repulsion impulses and friction
- Check linear trajectory from previous step for collisions and resolve them

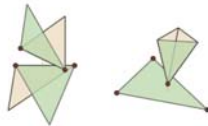
Geometric Collisions

Repulsive forces should remove most collisions
 Repulsive forces along can not insure no interpenetrations

If triangles are moving too fast, they may pass through each other in a single timestep.

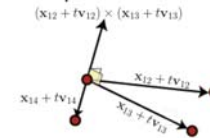
We can prevent this by checking for *any* collisions during the timestep (Provot [1997])

Note first that both point-face and edge-edge collisions occur when the appropriate 4 points are *coplanar*



Geometric Collisions

Detecting time of coplanarity - assume linear velocity throughout timestep:



So the problem reduces to finding roots of the cubic equation

$$((x_{12} + tV_{12}) \times (x_{13} + tV_{13})) \cdot (x_{14} + tV_{14})$$

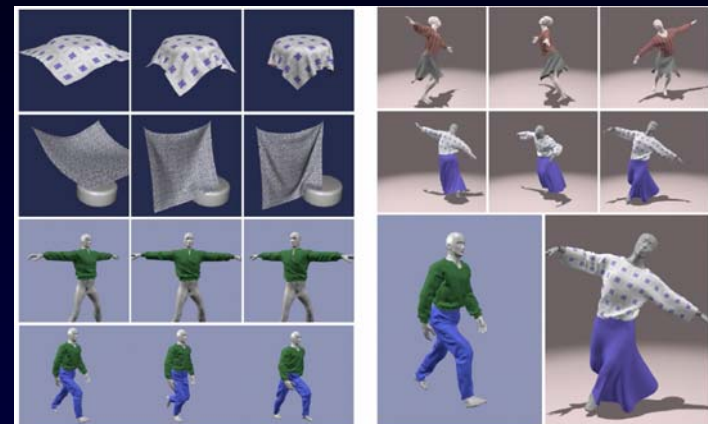
Once we have these roots, we can plug back in and test for triangle adjacency.

If intersection is detected, apply inelastic impulse to prevent it

Adaptive time-step

- Algorithm works for any Δt
- Start with Δt^{\max}
- Halve the time step when actual collisions occur
- Only do the full collision processing at Δt^{\min}
- Double the time step after three successful time steps

Results

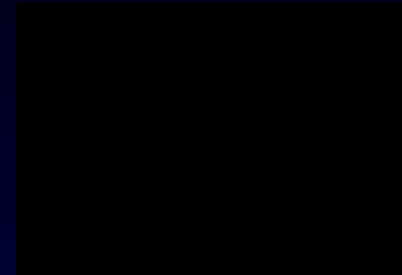


Results

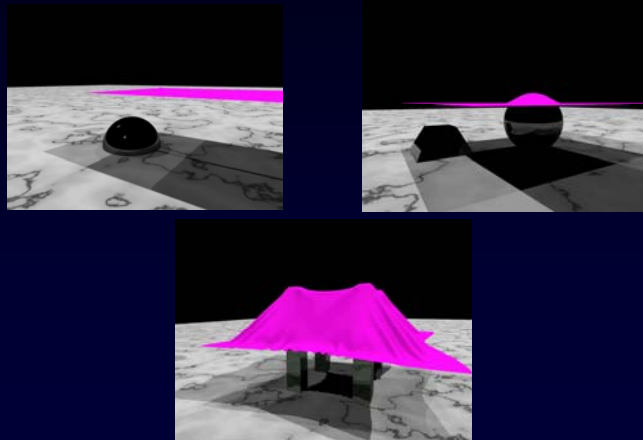
- Used by Alias|Wavefront in Maya Cloth
- Something similar used by Pixar



Large Steps in Cloth Simulation (by Baraff and Witkin)



Robust Treatment of Collisions, Contact and Friction for Cloth Animation (Bridson, Fedkiw, Anderson)



Stable but Responsive Cloth Kwang-Jin Choi, Hyeong-Seok Ko

Animation #1(b)

Stable but Responsive Cloth
Kwang-Jin Choi, Hyeong-Seok Ko

Animation #3

Stable but Responsive Cloth
Kwang-Jin Choi, Hyeong-Seok Ko

Animation #4

Read

- Large Steps in Cloth Simulation (by Baraff and Witkin)
- Robust Treatment of Collisions, Contact and Friction for Cloth Animation (Bridson, Fedkiw, Anderson)

Near Future

- Fluids (Next)
- Rigid Bodies
- Elastic Models
- Finite Element methods