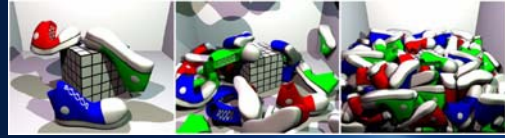


# Meshless Deformations Based on Shape Matching

Matthias Müller Bruno Heidelberger  
Matthias Teschner Markus Gross

Presented by Mike Caloud

## Motivation and Goals



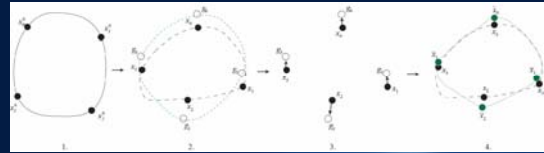
- Simulate object deformation
  - Simulate in real time, for interactive applications
    - Simple, fast computation
  - Always remains stable, despite extreme deformations
  - Require no pre-processing

## Motivation and Goals



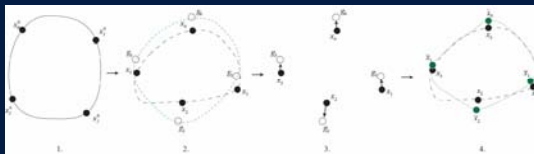
- Not physically accurate; doesn't strive for correct material representation
  - Inappropriate for engineering simulation
- Fast and visually believable
  - Appropriate for virtual environments and video games

## High-Level Overview



1. Undeformed mesh
2. Given the current deformed mesh, find goal positions that transform the undeformed mesh to the deformed mesh
3. Given goal positions, try to move the current deformed mesh towards the goal positions, subject to internal (spring) and external (gravity, collision) forces.
4. Transform deformed mesh

## Algorithm



- A. Use shape matching from  $\mathcal{V}_U$  to  $\mathcal{V}_D$  find goal positions  $\mathcal{G}$
- B. Calculate forces and integrate to find new deformed mesh  $\mathcal{V}_D'$
- C. Go to A

## Overview

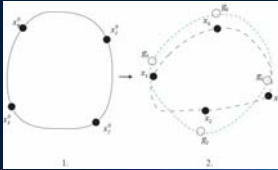
- Shape matching
- Integration
- Spring model
- Extensions

## Questions?

## Overview

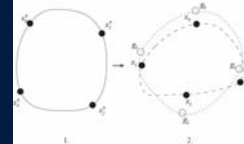
- Shape matching
- Integration
- Spring model
- Extensions

## Shape Matching



- Given the undeformed and deformed meshes, we want to find the transformation that moves the undeformed mesh to the deformed mesh

## Shape Matching



- At each time step, two sets of particles
  - $\mathbf{x}_i^0$  particle positions
  - $\mathbf{x}_i$  particles that move according external forces
  - $m_i$  masses
- Find rotation and translation that move  $\mathbf{x}_i^0$  as close as possible to  $\mathbf{x}_i$
- Store results as goals  $\mathbf{t}$

## Shape Matching

- Find rotation matrix  $\mathbf{R}$  and translation vectors  $\mathbf{t}$  and  $\mathbf{t}_0$  that minimize

$$\sum_i w_i (\mathbf{R}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} - \mathbf{x}_i)^2$$

- Known as problem of absolute orientation
- No derivation of results

## Shape Matching – find $\mathbf{t}_0$ and $\mathbf{t}$

- Translations are given by centers of mass of  $\mathbf{x}_i^0$  and  $\mathbf{x}_i$

$$\mathbf{t}_0 = \mathbf{x}_{\text{cm}}^0 = \frac{\sum_i m_i \mathbf{x}_i^0}{\sum_i m_i}, \quad \mathbf{t} = \mathbf{x}_{\text{cm}} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i}$$

## Shape Matching – find $\mathbf{R}$

- With translation vectors  $\mathbf{t}$  and  $\mathbf{t}_0$ , we can solve for  $\mathbf{R}$ .

$$\mathbf{A} = \left( \sum_i m_i \mathbf{p}_i \mathbf{q}_i^T \right) \left( \sum_i m_i \mathbf{q}_i \mathbf{q}_i^T \right)^{-1} = \mathbf{A}_{pq} \mathbf{A}_{qq}^{-1}$$

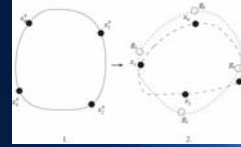
where  $\mathbf{p}_i = \mathbf{x}_i - \mathbf{t}$      $\mathbf{q}_i = \mathbf{x}_i^0 - \mathbf{t}_0$

Polar decomposition of  $\mathbf{A}_{pq}$  yields

$$\mathbf{S} = \sqrt{\mathbf{A}_{pq}^T \mathbf{A}_{pq}}$$

$$\mathbf{R} = \mathbf{A}_{pq} \mathbf{S}^{-1}$$

## Shape Matching – find $\mathbf{g}_i$



- With  $\mathbf{R}$  and translation vectors  $\mathbf{t}$  and  $\mathbf{t}_0$  we can calculate goal positions  $\mathbf{g}_i$

$$\mathbf{g}_i = \mathbf{R}(\mathbf{x}_i^0 - \mathbf{x}_{\text{cm}}^0) + \mathbf{x}_{\text{cm}}$$

## Questions?

## Overview

- Shape matching
- Integration
- Spring model
- Extensions

## Newton's Second Law of Motion

$$\mathbf{F} = m\mathbf{a}$$

F=force, m=mass, a=acceleration; rate of change of velocity

Note that we need to integrate this equation over time to get a particle's new position  $\mathbf{p}$

$$\mathbf{a} = \mathbf{F} / m$$

$$\mathbf{v} = \int \mathbf{a} dt$$

$$\mathbf{p} = \int \mathbf{v} dt$$

## Explicit Euler Integration

- The authors use Euler integration for the two integrations because of its speed

$$\begin{aligned} v(t+h) &= v(t) + h \frac{-k(x(t) - l_0)}{m} \\ x(t+h) &= x(t) + hv(t+h), \end{aligned}$$

- Note that although a fast and simple integration scheme, the integration can possibly blow up if time step  $h$  is too large

## Overview

- Shape matching
- Integration
- Spring model
- Extensions

## Hooke's Law

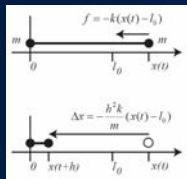
$$\mathbf{F} = -k\mathbf{x}$$

Hooke's Law is used to simulate spring deformation.

$\mathbf{F}$  = restoring spring force  
 $k$  = spring constant  
 $X$  = rest length of the spring

## Mass-Spring System

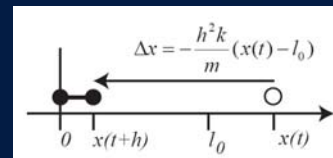
- Authors use a mass-spring system (which uses Newton's Second Law and Hooke's Law) with explicit Euler integration to enable deformation



### 1D Spring Example

- $m$  = mass,  $f$  = force,  $k$  = spring constant,  $t$  = time,  $x(t)$  = position of dot in  $x$  at  $t$ ,  $l_0$  = rest length of the spring,  $h$  = time step
- Change in  $x$ 
  - Acceleration = force divided by mass
  - Change in velocity = acceleration \*  $h$
  - Change in position = velocity \*  $h$

## Explicit Euler Integration Problem



- Note that although a fast and simple integration scheme, the integration can possibly blow up if time step  $h$  is too large

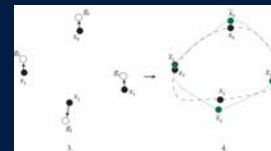
## Meshless Deformation Integration

- Nonetheless, given goal positions  $\mathbf{g}_i$  we can now transform the particle system to new positions now that we can integrate

$$\begin{aligned} \mathbf{v}_i(t+h) &= \mathbf{v}_i(t) + \alpha \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h f_{\text{ext}}(t) / m_i \\ \mathbf{x}_i(t+h) &= \mathbf{x}_i(t) + h \mathbf{v}_i(t+h), \end{aligned}$$

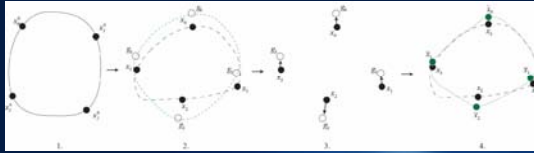
- Modified Euler integration – contains both internal and external force calculations
- Alpha – stiffness factor in  $[0, 1]$ . For alpha = 1 the particles move exactly to the goal. For alpha < 1 the particles act like springs, oscillating around the goals.

## Meshless Deformation Integration



$$\begin{aligned} \mathbf{v}_i(t+h) &= \mathbf{v}_i(t) + \alpha \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h f_{\text{ext}}(t) / m_i \\ \mathbf{x}_i(t+h) &= \mathbf{x}_i(t) + h \mathbf{v}_i(t+h), \end{aligned}$$

## Algorithm



- A. Use shape matching from  $\mathbf{x}_i^0$  to  $\mathbf{x}_i^1$  find goal positions  $\mathbf{g}_i$
- B. Calculate forces and integrate to find new deformed mesh  $\mathbf{x}_i^2$
- C. Go to A

## Questions?

## Overview

- Shape matching
- Integration
- Spring model
- Extensions

## Extensions

- Can imitate rigid body dynamics (a non-deforming mesh still subject to forces)

$$\mathbf{v}_i(t+h) = \mathbf{v}_i(t) + \alpha \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h f_{\text{ext}}(t)/m_i$$

- Setting alpha to 1 forces deformed mesh to snap exactly to the goals
- An optimization – particles not subject to external forces can just be transformed directly to the goal, skipping integration

## Extensions

- Can also allow cluster-based deformations, allowing for more arbitrary types of deformations
- Subdivide mesh surface into cubes. Treat each cube as a particle.
- Requires a modified spring term, where each cube moves towards a particle which is subjected to force

$$\Delta \mathbf{v}_i = \alpha \frac{\mathbf{g}_i^c(t) - \mathbf{x}_i(t)}{h}$$

## Results





## Results



## Demo

## Discussion