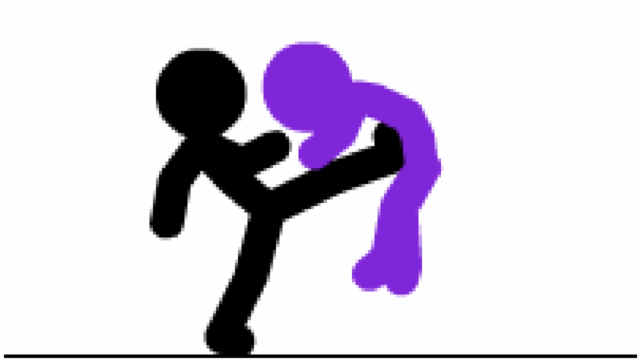
# Meshless Deformations Based on Shape Matching

M. Müller, B. Heidelberger, M. Teschner, M. Gross

Proceedings of SIGGRAPH'05, 2005

# The Goal

To simulate deformable objects in a computationally efficient and stable way



Oriented towards the gaming and animation industries

# Existing methods

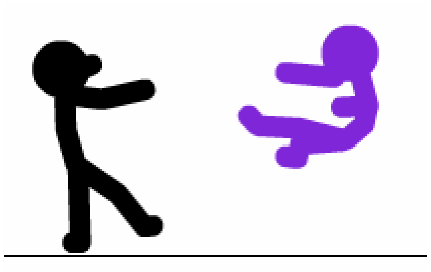
Not applicable because they are either:

- Difficult to code (FEM, BEM)
- Too slow (FEM)
- Not stable (explicit mass spring)
- Memory consuming (Eigenmodes, stiffness matrices, Green‘s functions, motion databases)

# Mickey Mouse physics

To simplify the problem, the authors:

- Relax the requirement of realism
- Do not model material properties
- Avoid using the connectivity information

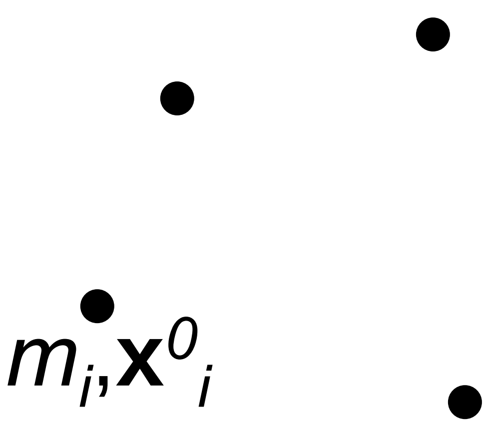


# Simple Model

- Solid represented by points with mass
  - Simple to generate
  - Animated as a particle system
- No connectivity needed
  - Volumetric meshes not available in games
  - Each point path is calculated independently
- Explicit integration
  - Simple to code
  - Efficient to compute
  - Unconditionally stable

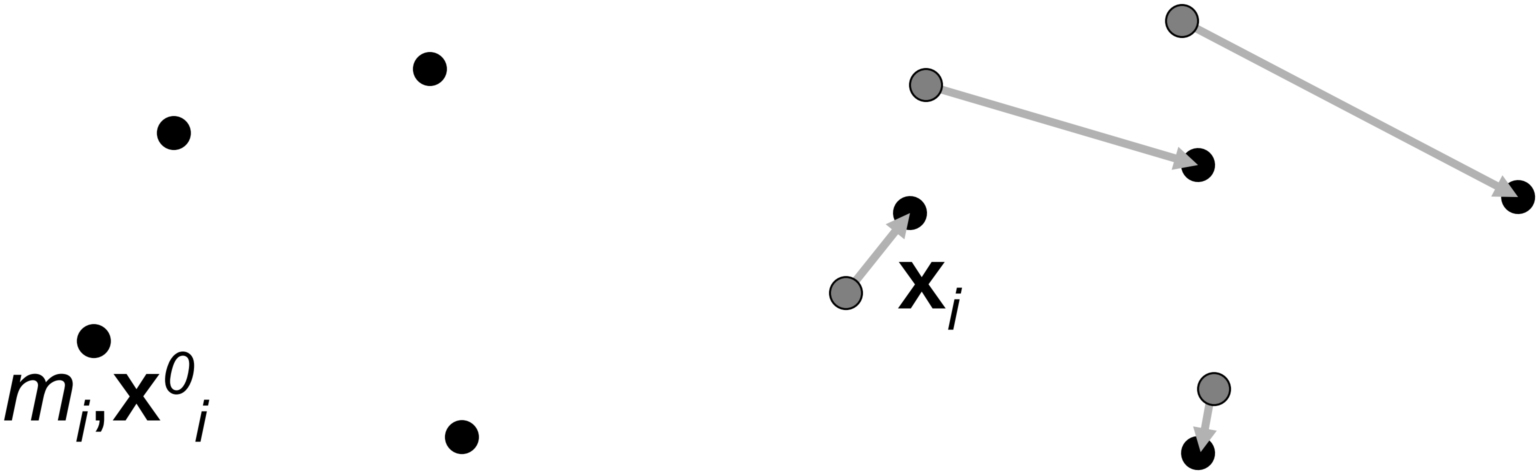
# Basic Algorithm

- Given $n$ points with masses $m_i$ and original positions $\mathbf{x}^0_i$



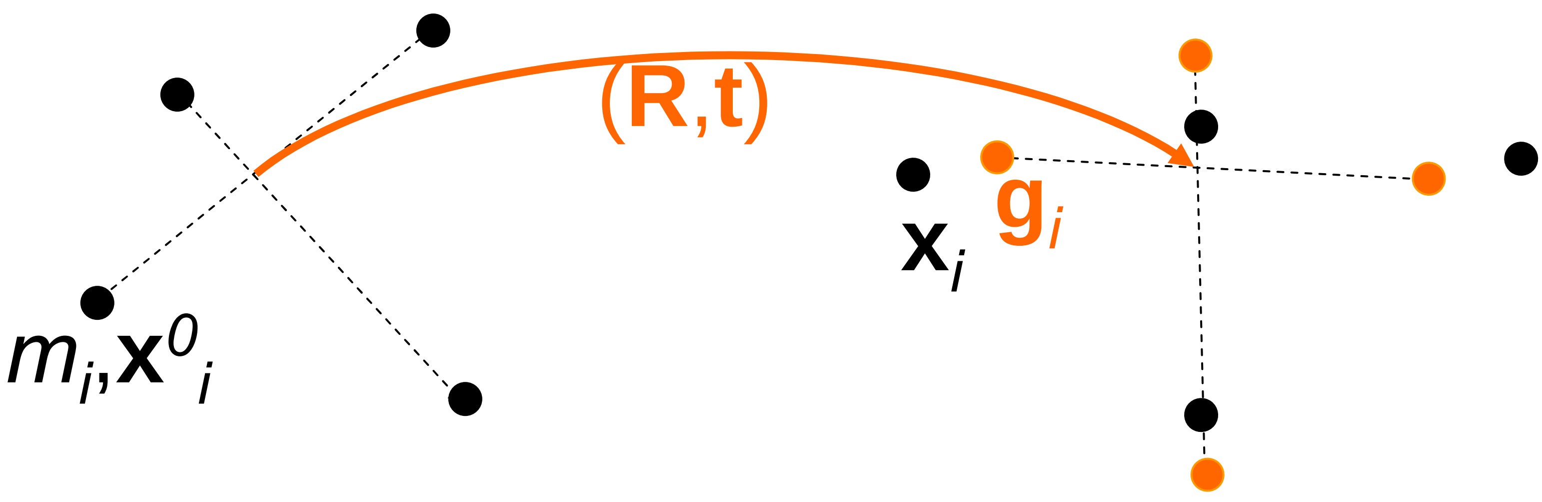
# Basic Algorithm

- Animate the points as unconnected particle system considering only external forces



# Basic Algorithm

- At every time step, match the original configuration $\mathbf{x}^0_i$ to the actual configuration $\mathbf{x}_i$ yielding goals $\mathbf{g}_i$



# Basic Algorithm

- Pull the actual points towards the goal positions $\mathbf{g}_i$

# Algorithm

Has two main components:

1. Finding the optimal transformation (shape matching to find the goal points)

2. Moving the particles towards those points (modeling acceleration and velocity)

Trade Off:

Subside to external forces vs. Retain the original configuration

# Shape Matching

- Given $\mathbf{x}_i^{0}$, $\mathbf{x}_i$ and weights $w_i$
- Find rigid body transform $\mathbf{t}_0$, $\mathbf{t}$, $\mathbf{R}$ minimizing

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

- Select $w_i = m_i$
- Optimal $\mathbf{t}_0$ is center of mass of the $\mathbf{x}_i^{0}$
- Optimal $\mathbf{t}$ is center of mass of the $\mathbf{x}_i$

[Kanatani 1994], [Umeyama 1991] and [Lorusso et al.1995]

# Optimal Rotation

- Define $\mathbf{p}_i = \mathbf{x}_i - \mathbf{x}_{cm}$ and $\mathbf{q}_i = \mathbf{x}_i^0 - \mathbf{x}_{cm}^0$
- Minimize $\sum_i m_i \left( \mathbf{A}\mathbf{q}_i - \mathbf{p}_i \right)^2$
- With optimal linear transform

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

- Optimal **R** from polar decomposition of $\mathbf{A}_{pq}$

# Goal positions

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

$$\mathbf{R} = \mathbf{A}_{pq} \mathbf{S}^{-1} \text{ where } \mathbf{S} = \sqrt{\mathbf{A}_{pq}^T \mathbf{A}_{pq}}$$

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

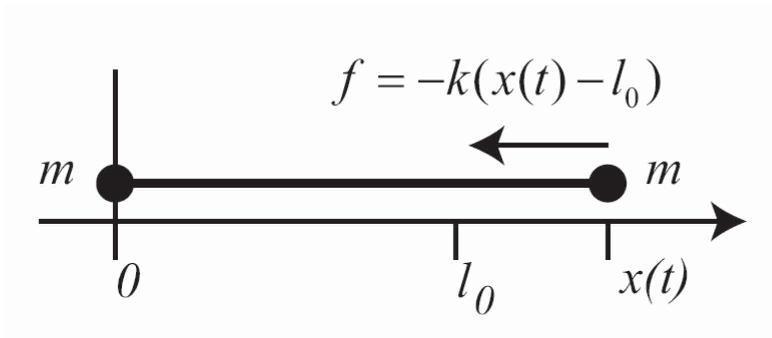
# Stable Explicit Integration

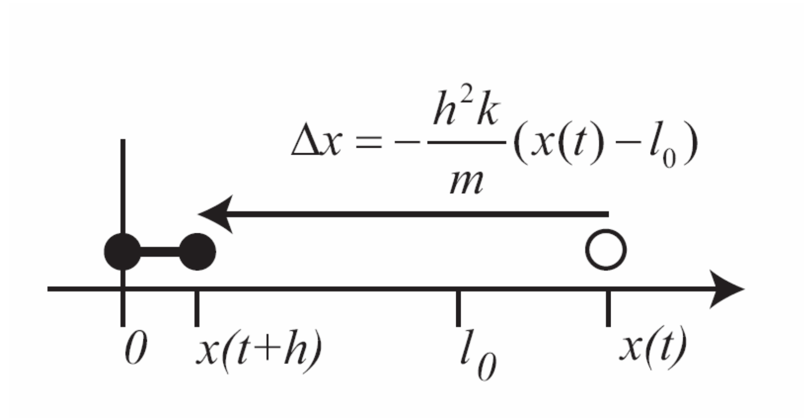
$$\mathbf{v}_i(t+h) = \mathbf{v}_i(t) + \alpha \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + \frac{h}{m_i}\mathbf{f}_{ext}(t)$$

$$\mathbf{x}_i(t+h) = \mathbf{x}_i(t) + h\mathbf{v}_i(t+h)$$

- Unconditionally stable
- Controls stiffness (rigid for $\alpha$ =1)
- Pulled out of a hat

# Compare to Euler Integration




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

$$
A = \begin{bmatrix} 1 & -\frac{kh}{m} \\ h & 1 - \frac{h^2 k}{m} \end{bmatrix}
$$

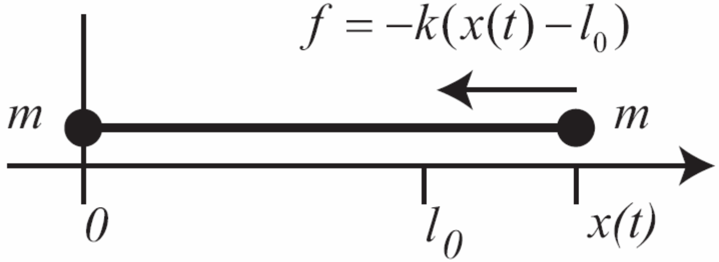
$$e_0 = 1 - \frac{1}{2m}\left(h^2 k - \sqrt{-4mh^2k + h^4k^2}\right)$$

$$e_1 = 1 - \frac{1}{2m}\left(h^2 k + \sqrt{-4mh^2k + h^4k^2}\right)$$

♣$e_0$♣ $<$1 for $h^2k \rightarrow \infty$

♣$e_1$♣ $<$1 iff $h < 2\sqrt{\frac{m}{k}}$

# Compare to Euler Integration




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

$$
\begin{bmatrix} v(t+h) \\ x(t+h) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix} \begin{bmatrix} v(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} \alpha l_0/h \\ \alpha l_0 \end{bmatrix}
$$

$$
e_{0,1} = (1-\alpha/2) \pm i\sqrt{4\alpha - \alpha^2}/2
$$

♣$e_{0,1}$♣ ⁄ 1

# Extensions
# (shape matching cont’d)

1. Linear Deformations
2. Quadratic Deformations
3. Cluster-based Deformations
4. Plasticity

# Linear Deformations

Recall that **R** is the polar decomposition of $\mathbf{A}_{pq}$ ($\mathbf{A}_{pq}$ = RS, where S = $\sqrt{\mathbf{A}_{pq}^T \mathbf{A}_{pq}}$)

Now, instead of using only **R** to compute the goal points, also use the matrix **A**

$$\mathbf{g}_i = (\beta \mathbf{A} + (1 - \beta)\mathbf{R})(\mathbf{x}_i^0 - \mathbf{x}_{cm}^0) + \mathbf{x}_{cm}$$

Objects are allowed to shear and stretch

# Quadratic Deformations

Further modify the matrix **A** by setting

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{Q} & \mathbf{M} \end{bmatrix}$$

and minimizing $\sum_i m_i(\tilde{\mathbf{A}}\tilde{\mathbf{q}}_i - \mathbf{p}_i)^2$ where

$$\tilde{\mathbf{q}} = [q_x, q_y, q_z, q_x^2, q_y^2, q_z^2, q_x q_y, q_x q_z, q_y q_z]$$

$$\mathbf{g}_i = (\beta\tilde{\mathbf{A}} + (1-\beta)\tilde{\mathbf{R}})(\mathbf{x}_i^0 - \mathbf{x}_{cm}^0) + \mathbf{x}_{cm}$$

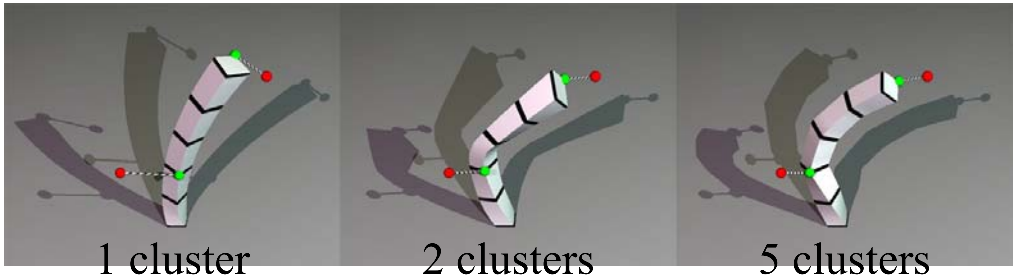
Objects are allowed to twist and bend

# Cluster Based Deformations

The set of particles is divided into overlapping clusters (cubes in space)

Run shape matching for each cluster

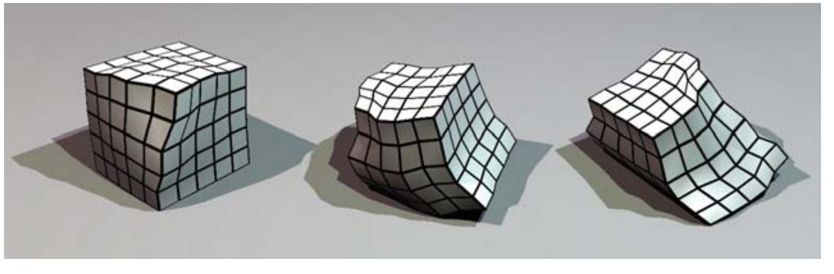
Average goal positions for each point



# Plasticity

Recall $\mathbf{A} = \mathbf{RS}$ where $\mathbf{R}$ is the rotational part of the transformation

In each cluster, if the unrotated deformation $\|\mathbf{S} - \mathbf{I}\|_2$ exceeds a threshold $c_{\text{yield}}$, incorporate it into the original shape



# Conclusions

- Pros
  - Straight forward to implement
  - Stable and scalable
  - Time and memory efficient
- Cons
  - Geometric - **not** physically derived
  - Tuned via non-physical (geometric) parameters