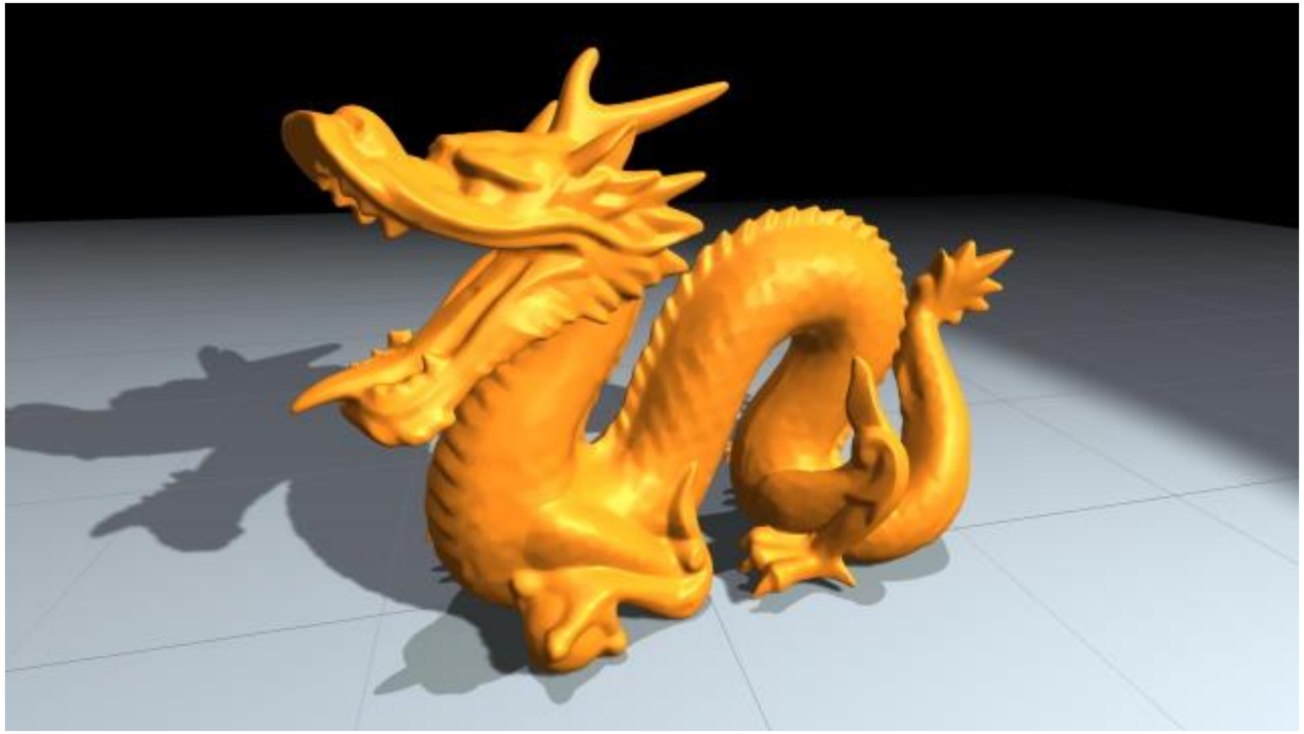
100 x Simulation Speedup with Mesh Embedding

Matthias Müller, Ten Minute Physics

www.matthiasmueller.info/tenMinutePhysics



Soft Body Simulation

- Given 60,000 triangle surface mesh

- Tetrahedralize the volume
- 300,000 tetrahedra

Key Observation

300,000 tetrahedra

3000 tetrahedra

- The essential motions can be captured with a lower resolution simulation mesh

Two Solutions

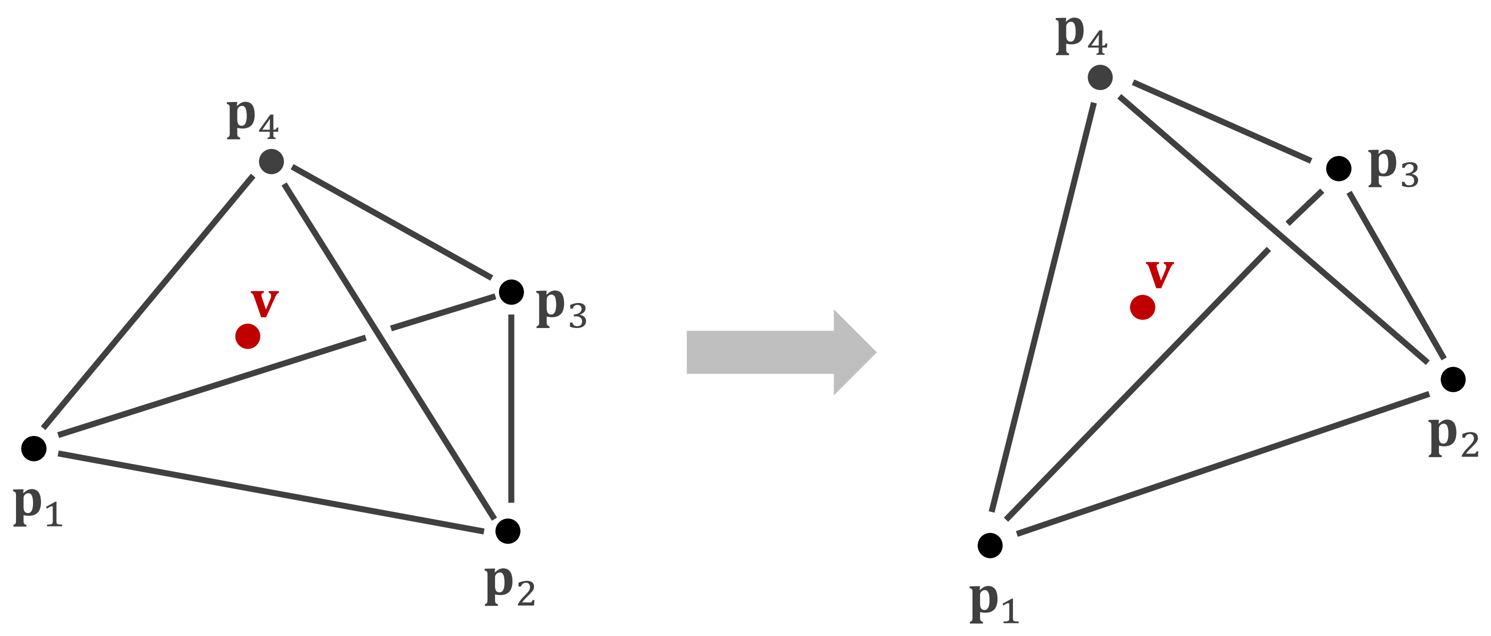
Model reduction

- Use high-resolution tetrahedral mesh
- Decompose system matrix into eigenmodes (deformation patterns)
- Select k first modes only
- Mathematically involved (non-linearities, collision handling)
- Non-trivial to implement

Surface embedding

- Create feature aware decimated surface (e.g. with Blender or hand tuned)
- Tetrahedralize simplified surface (later tutorial)
- Embed visual mesh (this tutorial)
- Very simple to implement

Tetrahedral Skinning



- Express $\mathbf{v}$ as weighted sum of $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{p}_4$

$$\mathbf{v} = b_1\mathbf{p}_1 + b_2\mathbf{p}_2 + b_3\mathbf{p}_3 + b_4\mathbf{p}_4$$

- Scalars b_1, b_2, b_3 and b_4 are the barycentric coordinates of $\mathbf{v}$
- Unique for four points (not contained in a plane)

Computing the Barycentric Coordinates

(using concepts introduced in tutorial 7)

$$\mathbf{v} = b_1\mathbf{p}_1 + b_2\mathbf{p}_2 + b_3\mathbf{p}_3 + b_4\mathbf{p}_4$$

- We can translate all points by the same amount without changing the result

  $$\mathbf{v} - \mathbf{p}_4 = b_1(\mathbf{p}_1 - \mathbf{p}_4) + b_2(\mathbf{p}_2 - \mathbf{p}_4) + b_3(\mathbf{p}_3 - \mathbf{p}_4)$$

- Three unkowns left, put them in a vector $\mathbf{b} = [b_1, b_2, b_3]^{\mathrm{T}}$
- Create the matrix $\mathbf{P} = [\mathbf{p}_1 - \mathbf{p}_4, \mathbf{p}_2 - \mathbf{p}_4, \mathbf{p}_3 - \mathbf{p}_4]$
- Now we can write: $\mathbf{v} - \mathbf{p}_4 = \mathbf{P}\,\mathbf{b}$
- Solving for **b**: $\mathbf{b} = \mathbf{P}^{-1}(\mathbf{v} - \mathbf{p}_4)$
- Deriving b_4 using the translated equation:

$$\begin{aligned}\mathbf{v} &= b_1\mathbf{p}_1 + b_2\mathbf{p}_2 + b_3\mathbf{p}_3 - b_1\mathbf{p}_4 - b_2\mathbf{p}_4 - b_3\mathbf{p}_4 + 1\mathbf{p}_4 \\ &= b_1\mathbf{p}_1 + b_2\mathbf{p}_2 + b_3\mathbf{p}_3 + (1 - b_1 - b_2 - b_3)\,\mathbf{p}_4\end{aligned}$$

Properties of Barycentric Coordinates

- They sum to 1

$$b_4 = 1 - b_1 - b_2 - b_3$$

$$b_1 + b_2 + b_3 + b_4 = 1$$

- For all point inside the tetrahedron and only them we have

$$b_1 \geq 0, b_2 \geq 0, b_3 \geq 0, b_4 \geq 0$$

- For a point outside thet tetrahedron the interpolation still works with potential artifacts (see upcoming tutorial for a solution)

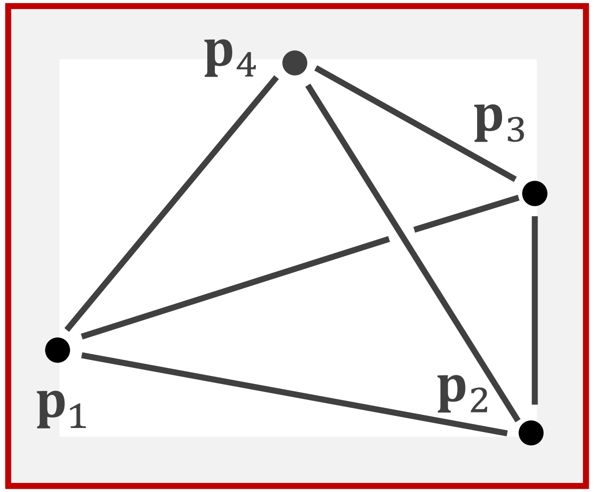
- Define a barycentric distance as

$$d = \max(-b_1, -b_2, -b_3, -b_4)$$

- Attach $\mathbf{v}$ to the tetrahedron with the smallest distance!

Attachment Computation

- Each vertex stores $d_{\min} = \infty$
- For each tetrahedron query vertex hash with inflated bounding box



- Skip vertices with $d_{\min} \leq 0$ (surrounding tetrahedrahon found)
- Compute barycentric coords and current d
- If $d < d_{\min}$ overwrite attachment and update $d_{\min}$
- Fast enough to do on the fly at startup

Let’s implement it…