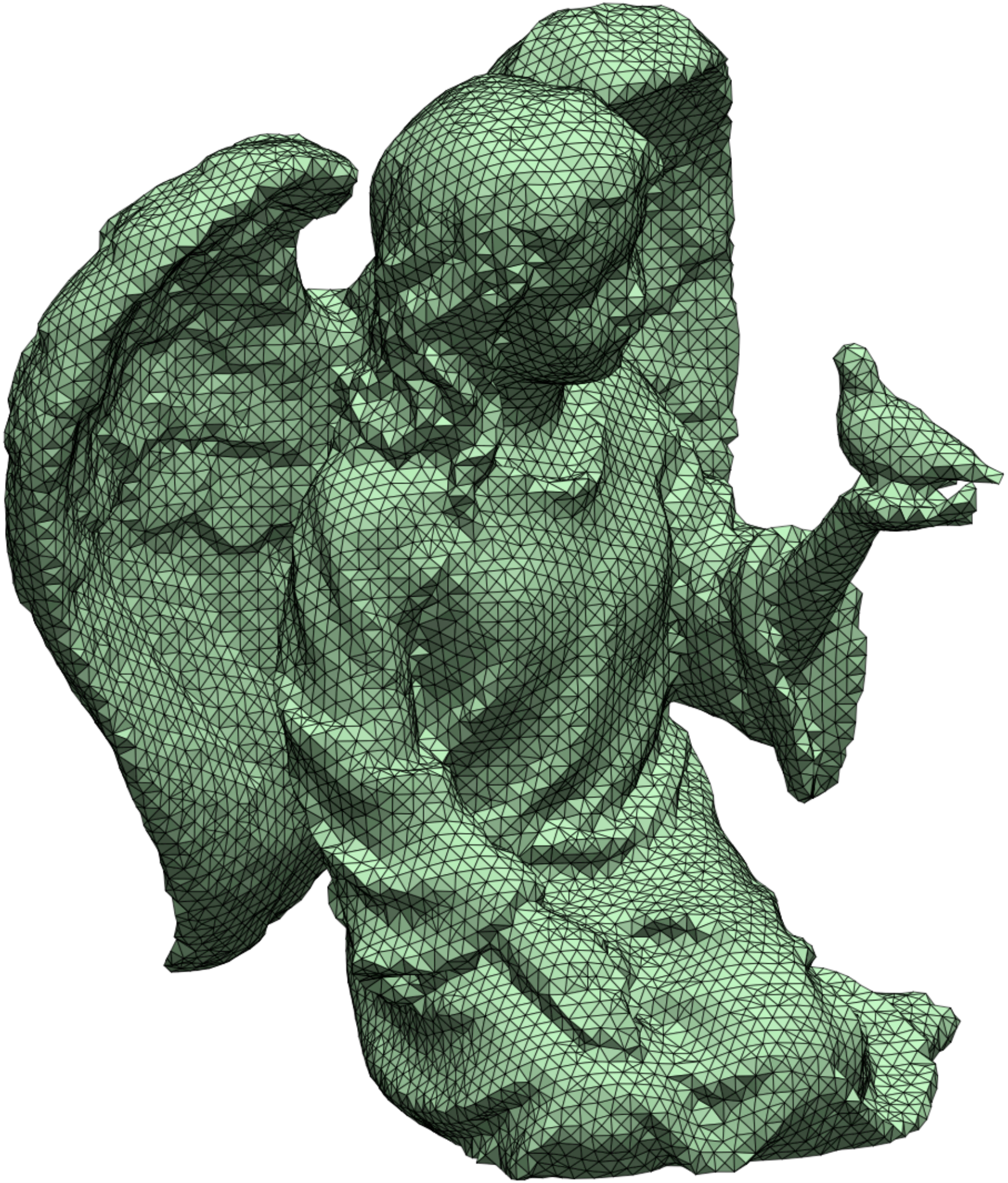
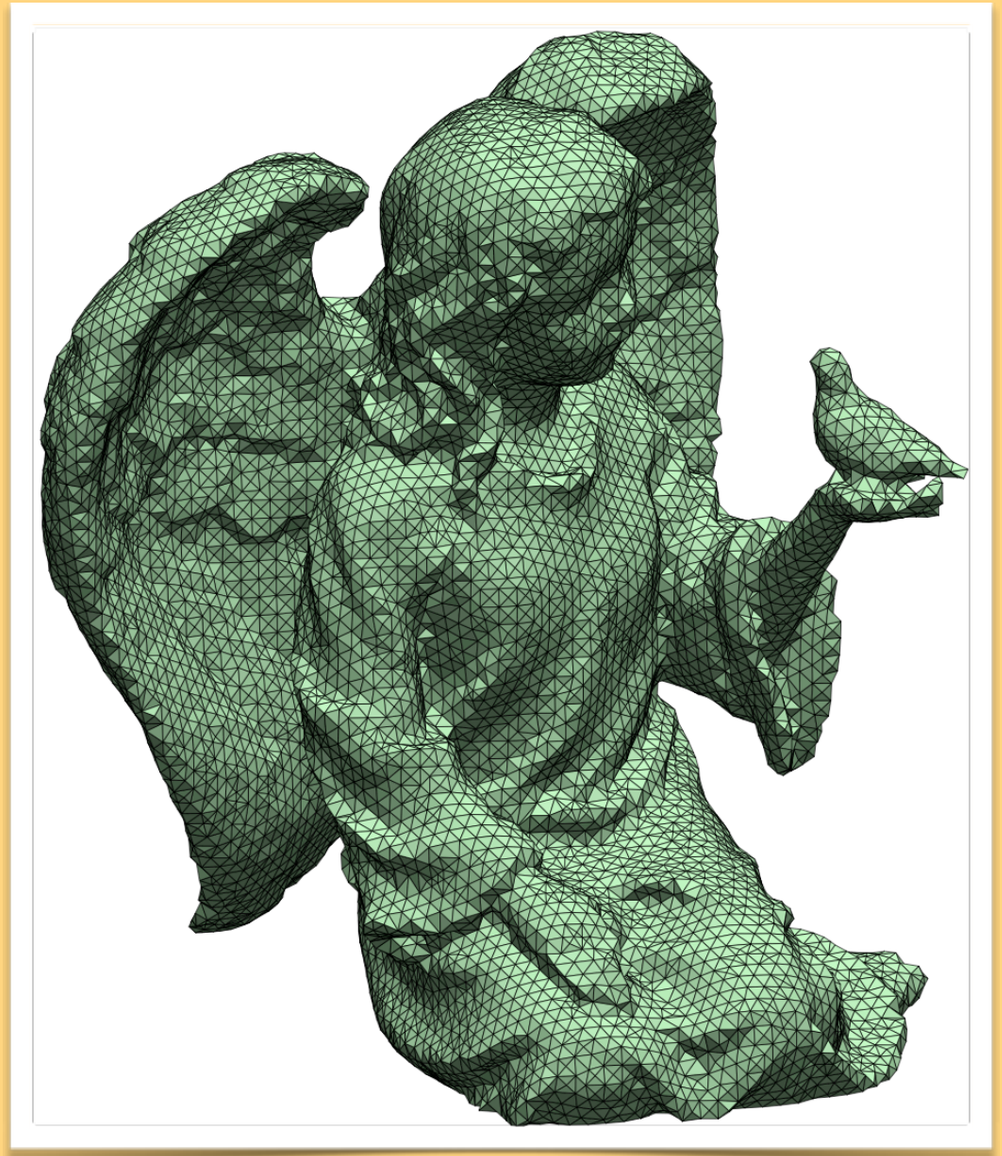


Laplacian Mesh Editing

Adrien Treuille
Carnegie Mellon University

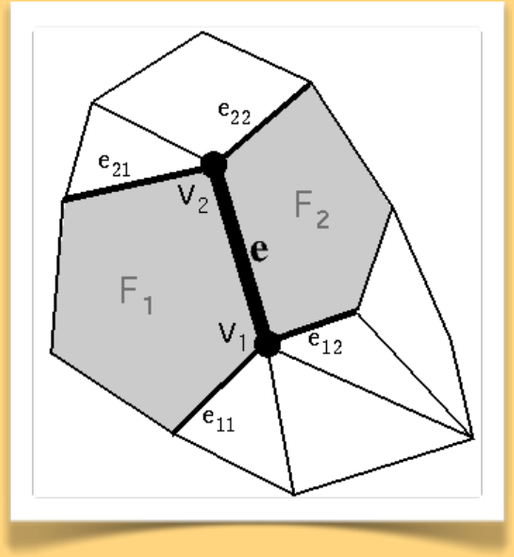
Meshes

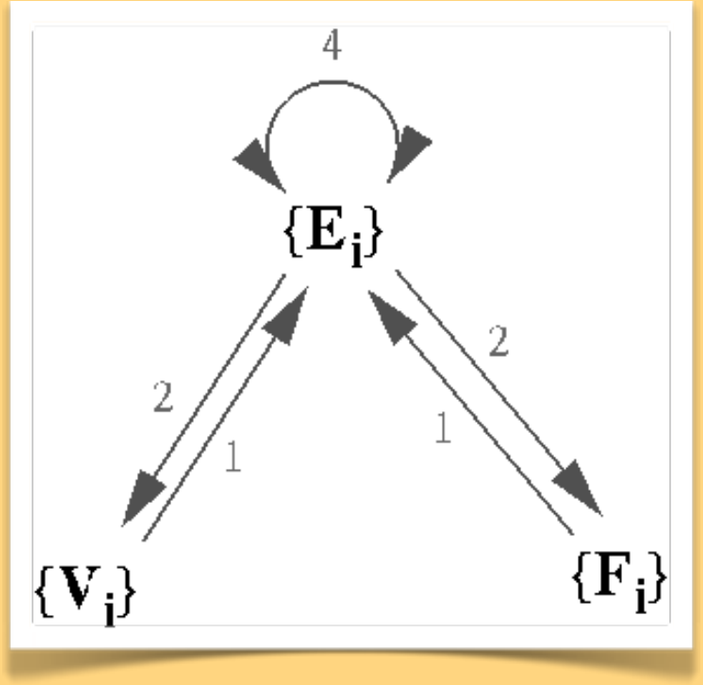


What can we do with them?

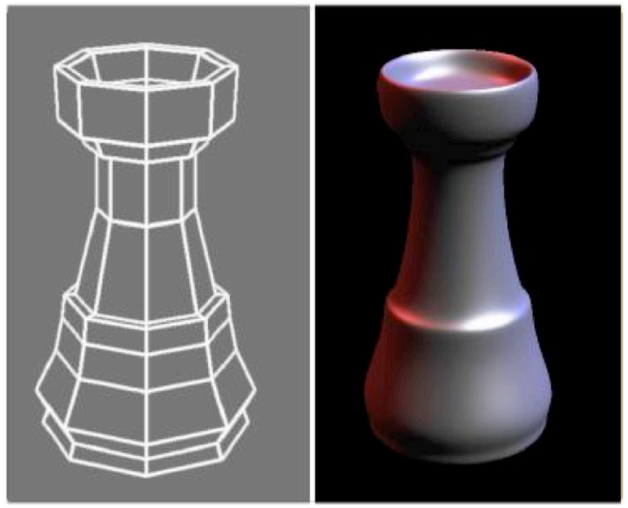
Basic Operations

- Render Them.
- Compute adjacent elements, etc.

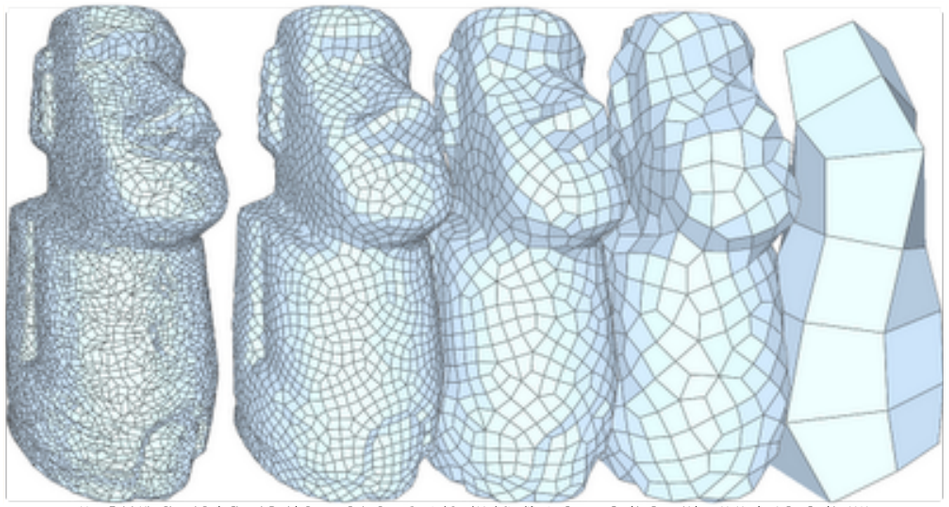




Sophisticated Operations

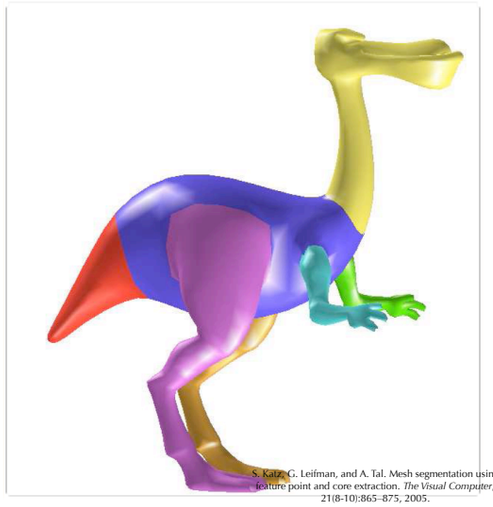


Subdivision



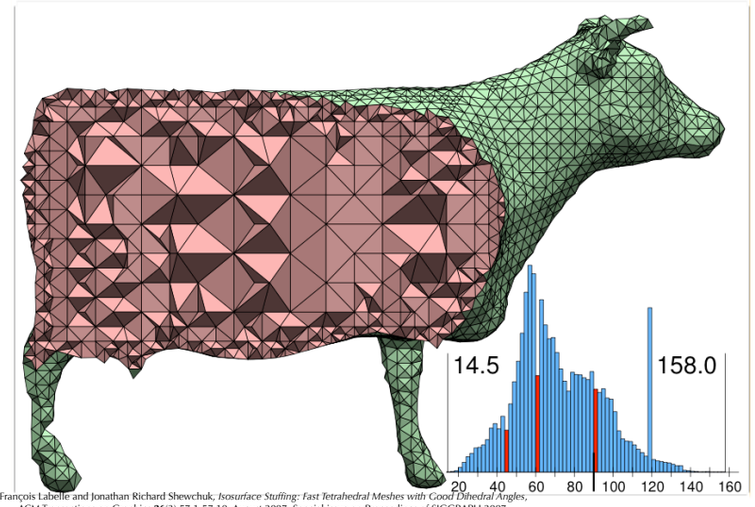
Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, Enrico Puppo *Practical Quad Mesh Simplification* Computer Graphics Forum, Volume 29, Number 2, EuroGraphics 2010

Simplifcation



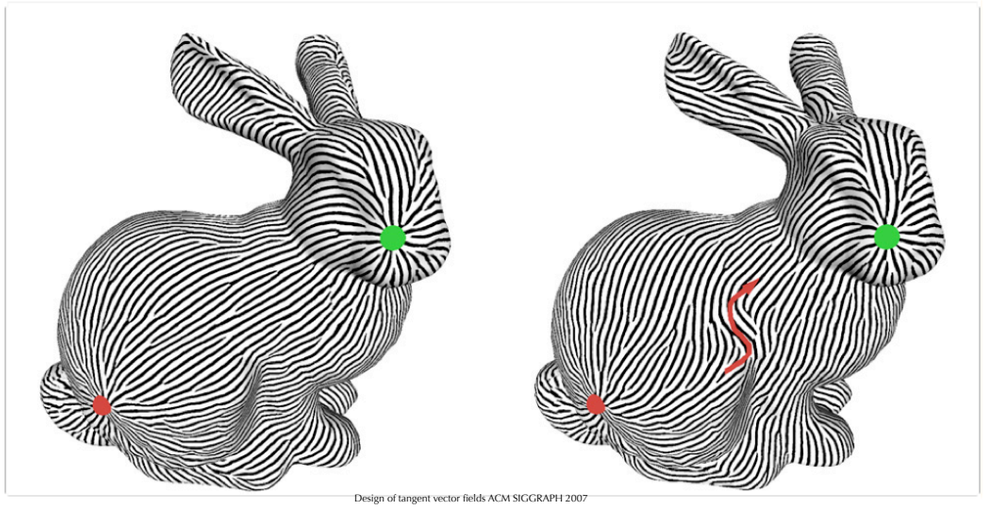
S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):865–875, 2005.

Segmentation



François Labelle and Jonathan Richard Shewchuk, *Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles*, ACM Transactions on Graphics **26**(3):57.1-57.10, August 2007. Special issue on Proceedings of SIGGRAPH 2007.

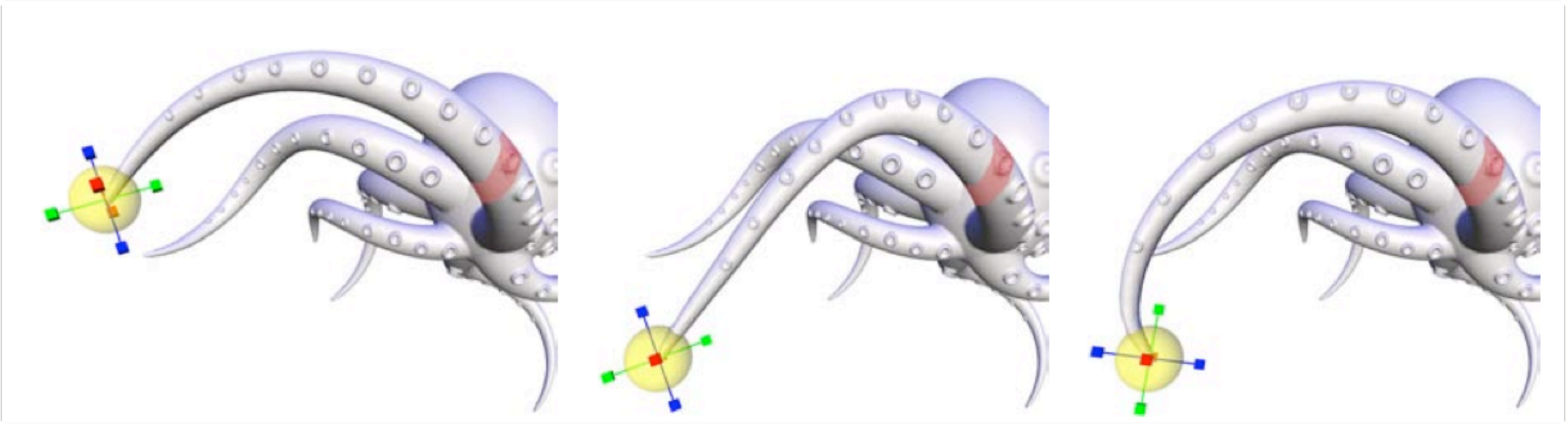
Tetrahedralization



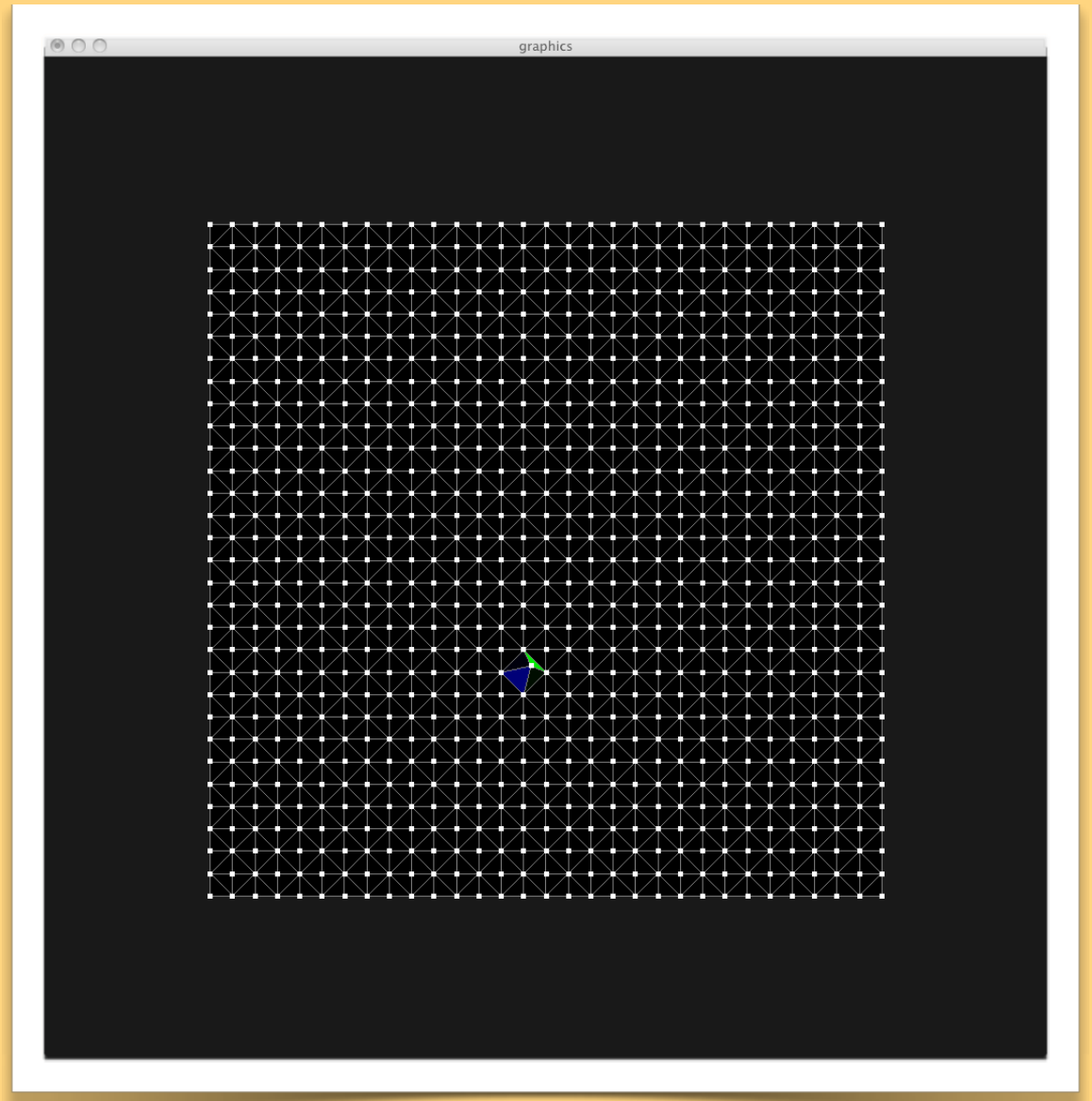
Design of tangent vector fields ACM SIGGRAPH 2007

...and Much, Much More

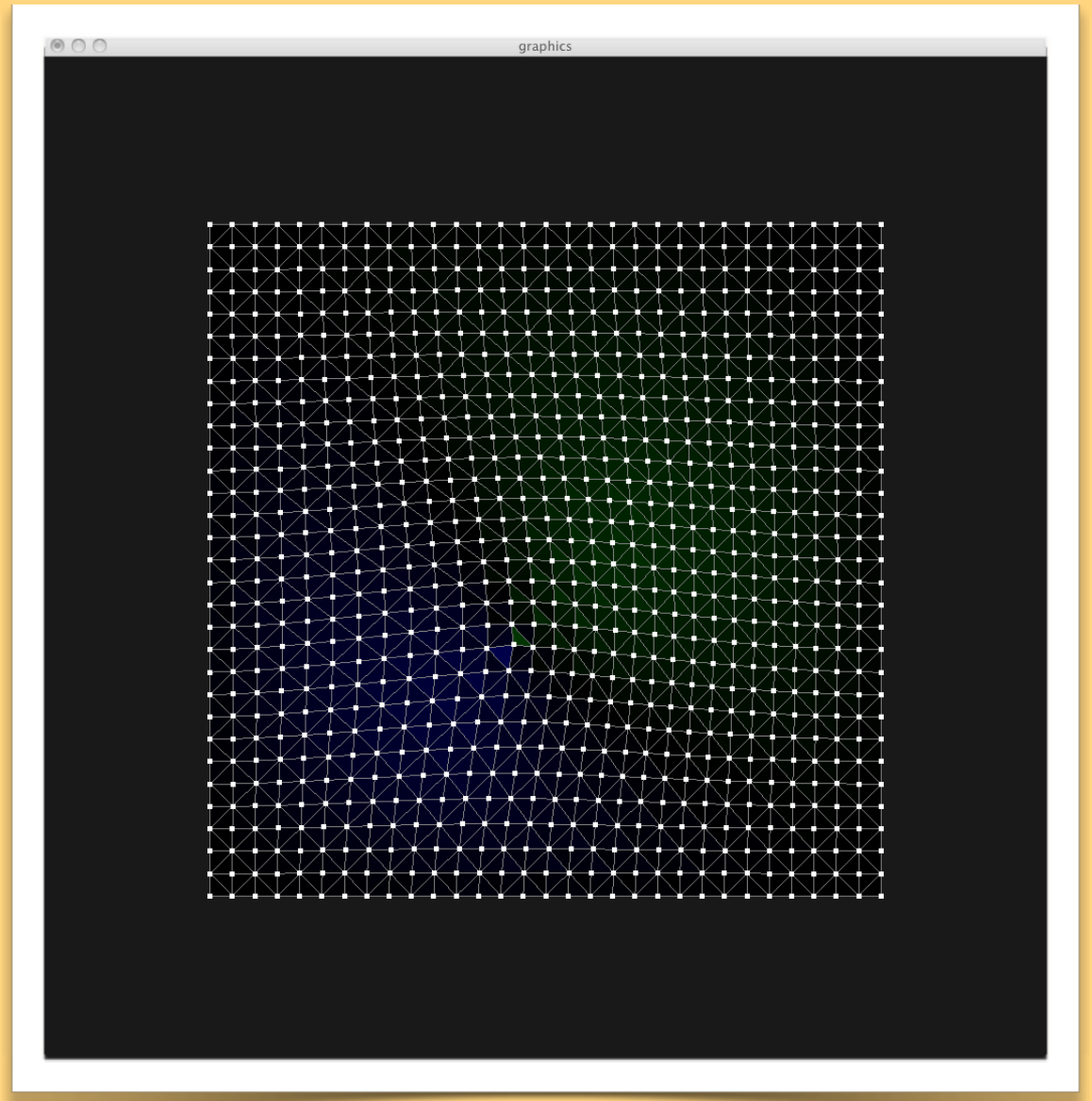
Manipulation



Demo

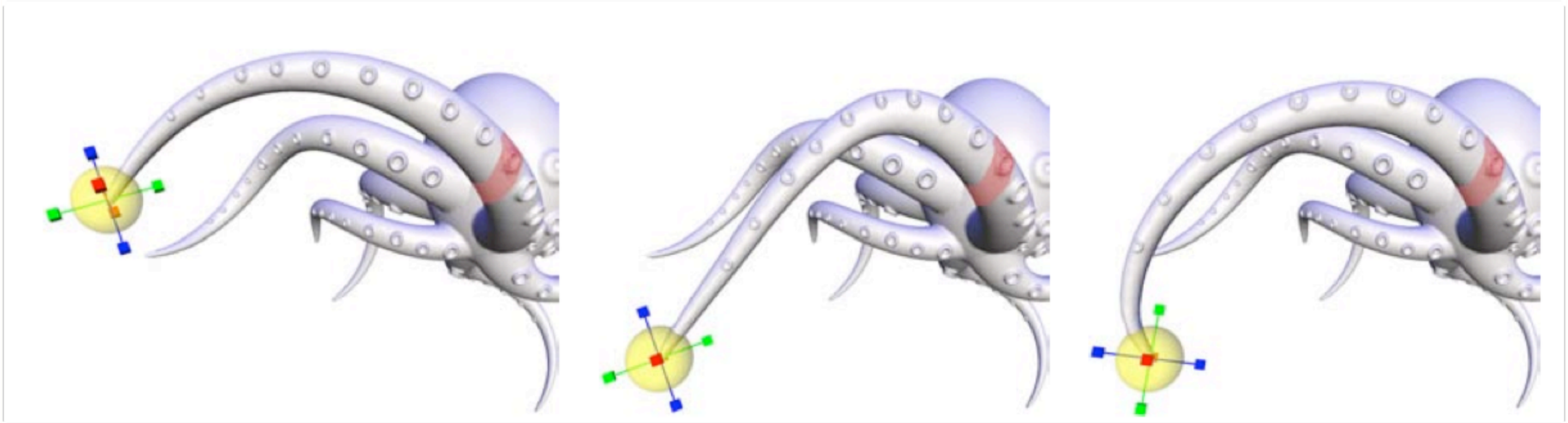


Bad



Good

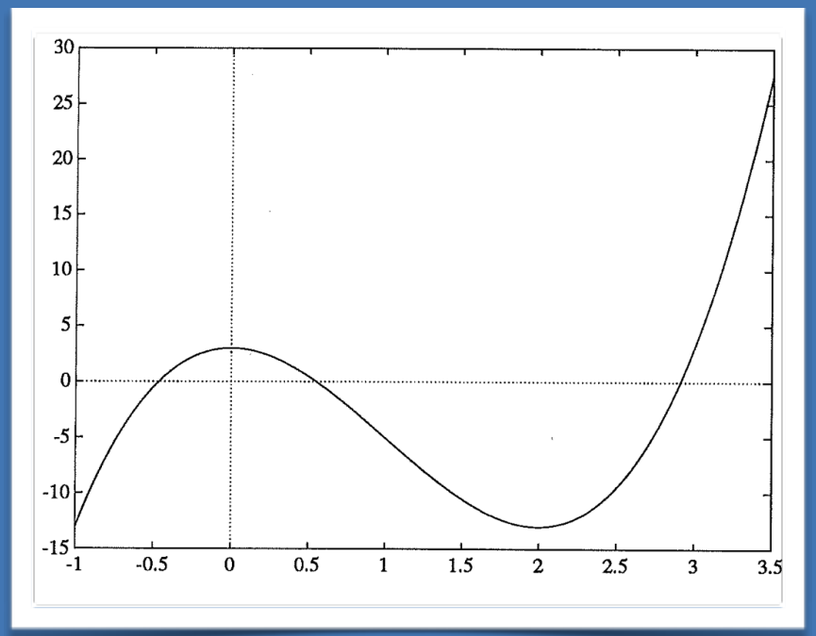
Manipulation



How can we do this?

New Concepts

- Use of objective functions.
  - ...to express desires.
- Using **BIG** matrices.



$$\begin{pmatrix} 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 8 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 4 & 0 \\ 0 & 1 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 7 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \end{pmatrix}$$

Objective Function

- **Big Idea**: Express some desired outcome as a mathematical function

$$\min_{\mathbf{x}} f(\mathbf{x})$$

- The vector **x** are the variables we can play with.
- The lower the function the better.
  - We assume $f \geq 0$, however $f = 0$ may not be achievable.

Objective Function

- **Big Idea**: Express some desired outcome as a mathematical function

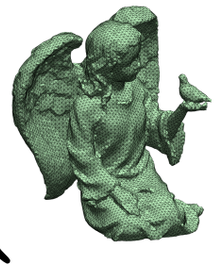
$$\min_{\mathbf{x}} f(\mathbf{x})$$

How do we find the minimum of *f*?

Set the *derivative* to 0!

$$\frac{df}{d\mathbf{x}} = 0$$

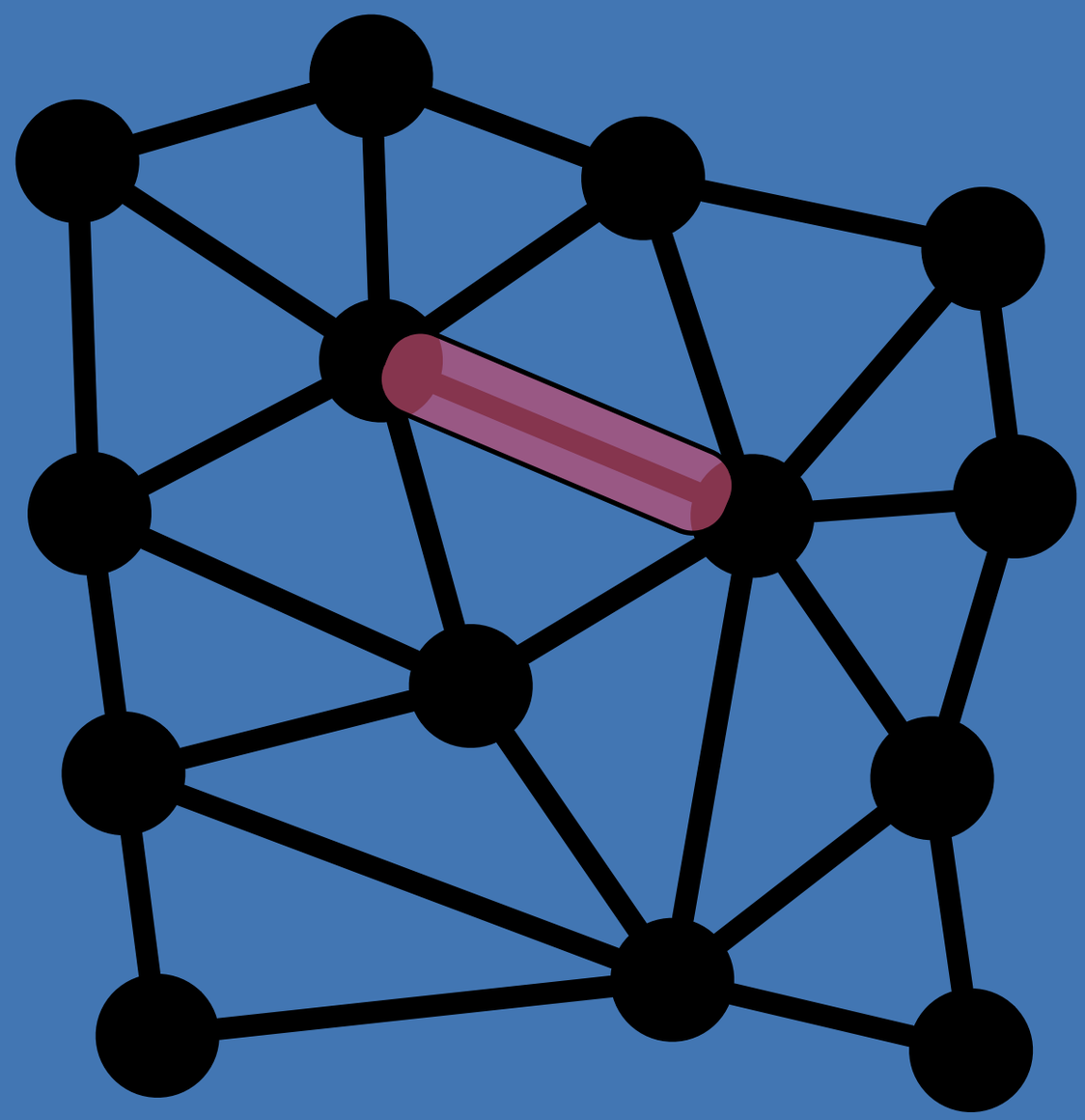
What are the Variables?

$$\min_{\mathbf{x}} f\Big($$  $$\Big)$$

The variables are the positions of each vertex:

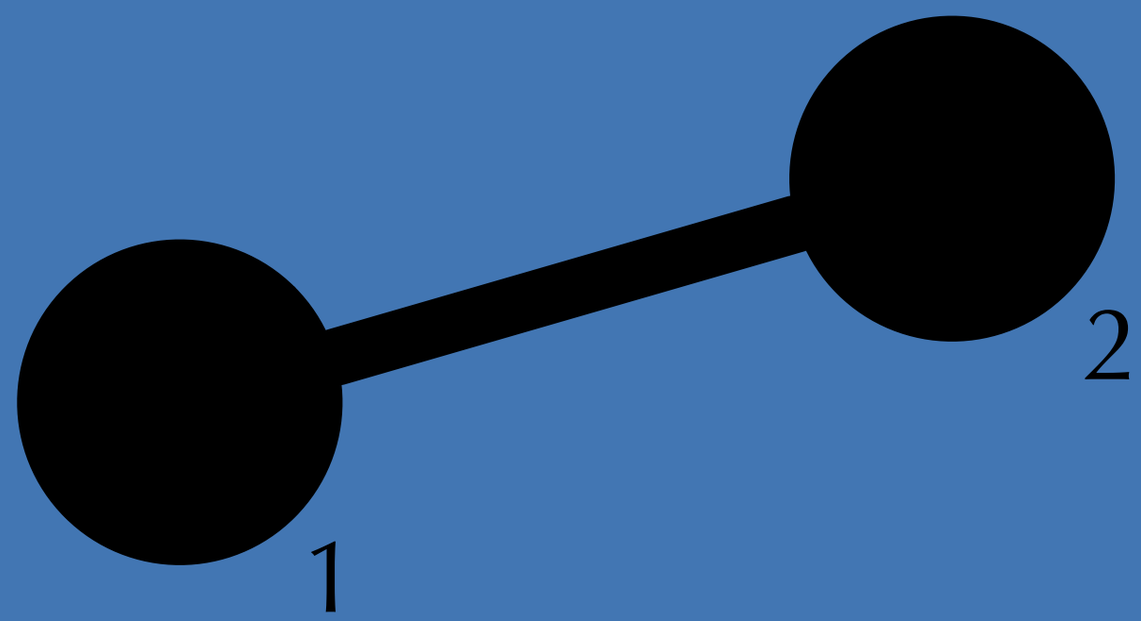
$$\mathbf{x} = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n]^T$$

What is a good objective function?



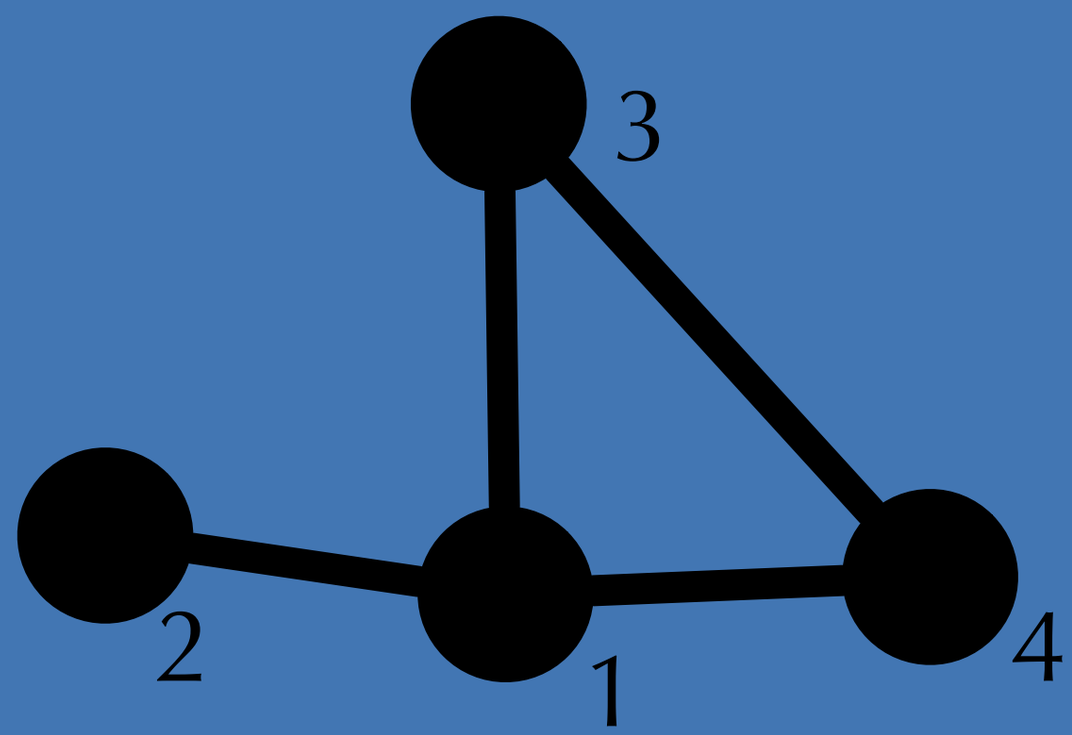
Let's look at one edge.

One Edge



Black board.

Multiple Edges



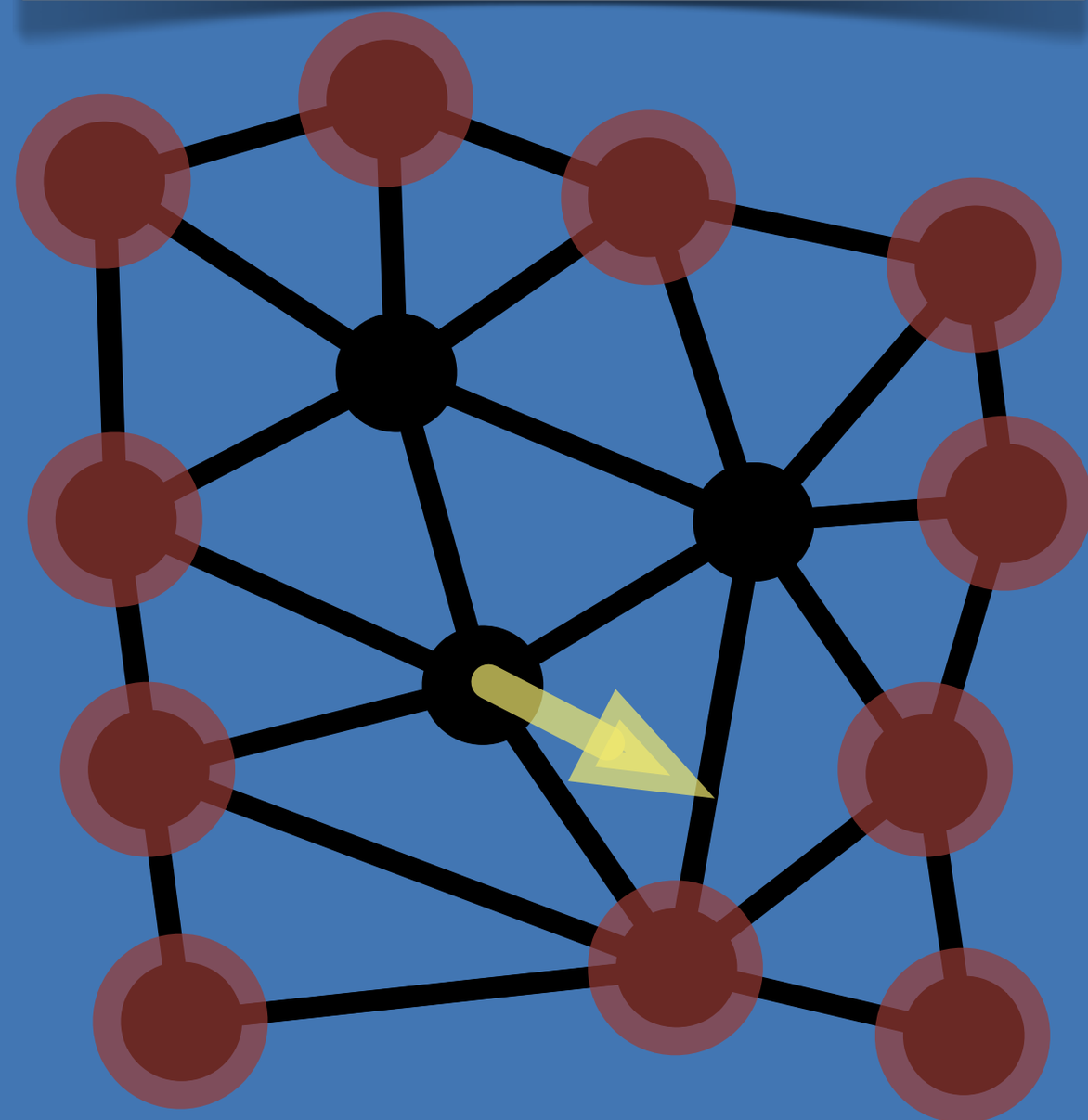
Black board.

Another Interpretation

$$L\mathbf{x} = L\mathbf{x}'$$

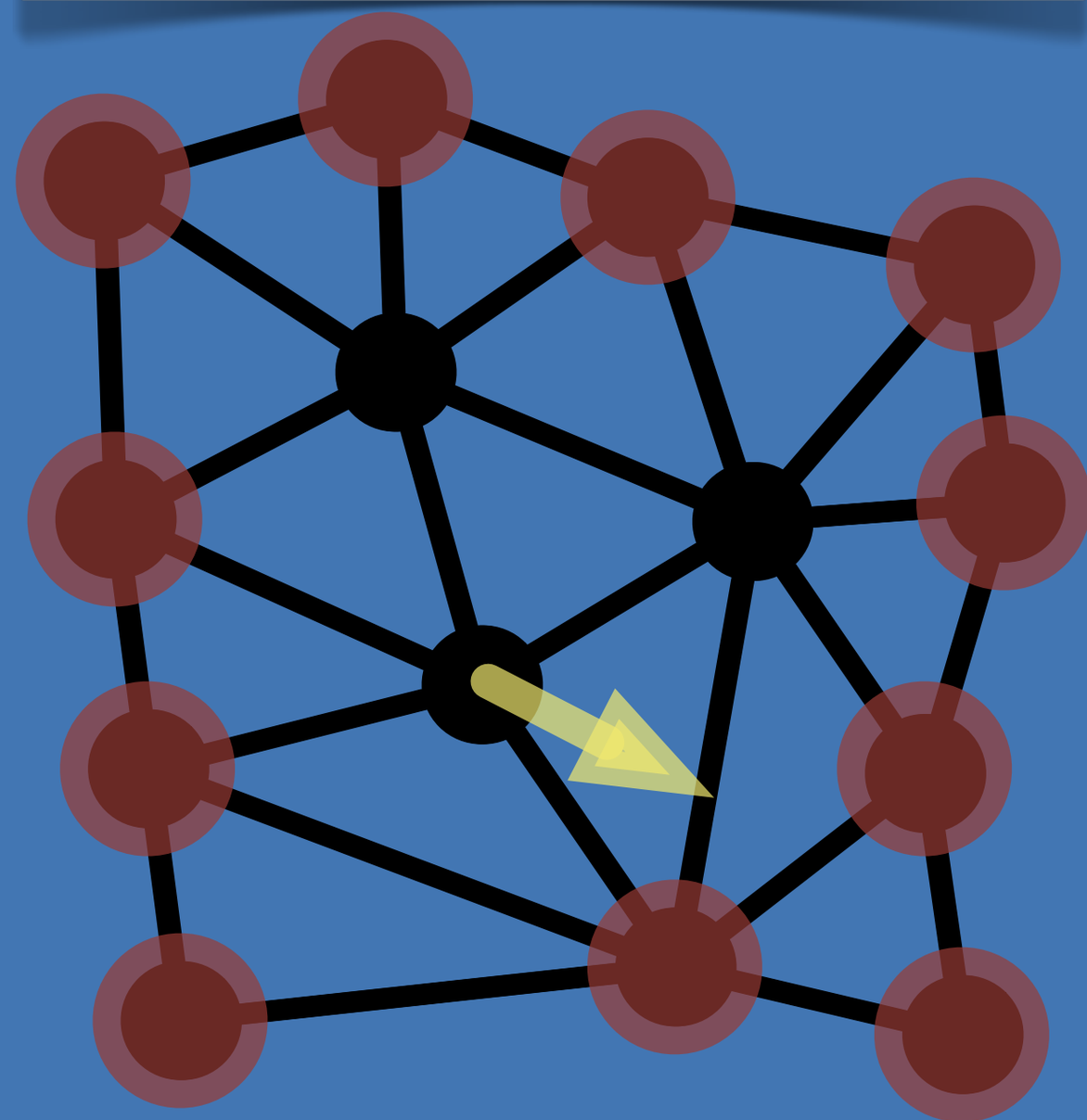
- The **Lagrangian** converts from **absolute** to ***relative*** mesh coordinates.
- This equation says that we want the **relative coordinates** of the mesh to **remain unchanged** after the deformation.

Constraints



- How do we deal with them?

Constraints



Blackboard

Constraints

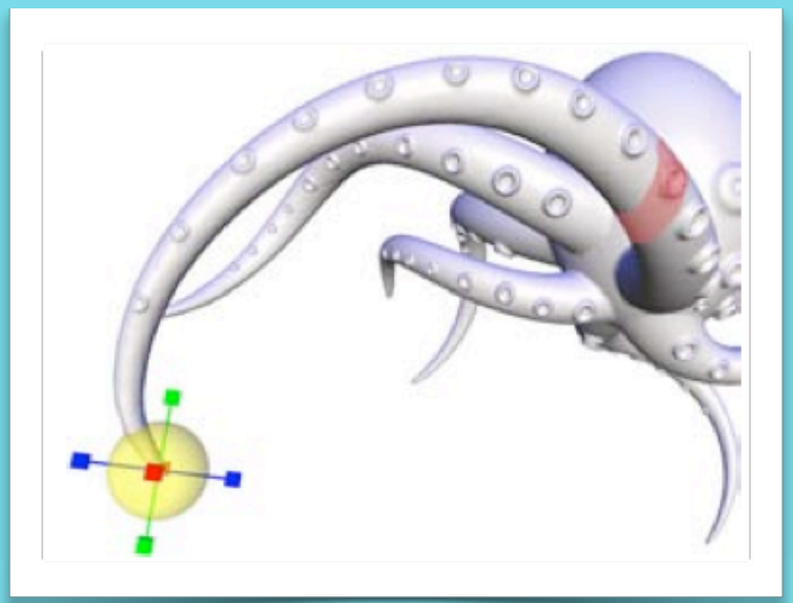
$$L\mathbf{x} = L\mathbf{x}'$$

Unconstrained Equation

$$L_f^T L_f \mathbf{x}_f = L_f^T L\mathbf{x}' - L_f^T L_c \mathbf{x}_c$$

Constrained Equation

Implementation



1. Create Laplacian
2. Create Full Equation
3. Solve

1: Create Laplacian

$$\begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$

$$L\mathbf{x} = L\mathbf{x}'$$

```
# Step 1: Create Laplacian

full_laplacian = sparse_matrix((num_vertices, num_vertices))
for ii in vertices:
    adjacent_vertices = vertices_adjacent_to(ii)
    for jj in adjacent_vertices:
        full_laplacian[ii,jj] = -1.0
    full_laplacian[ii,ii] = len(adjacent_vertices)
```

2: Create Full Equation

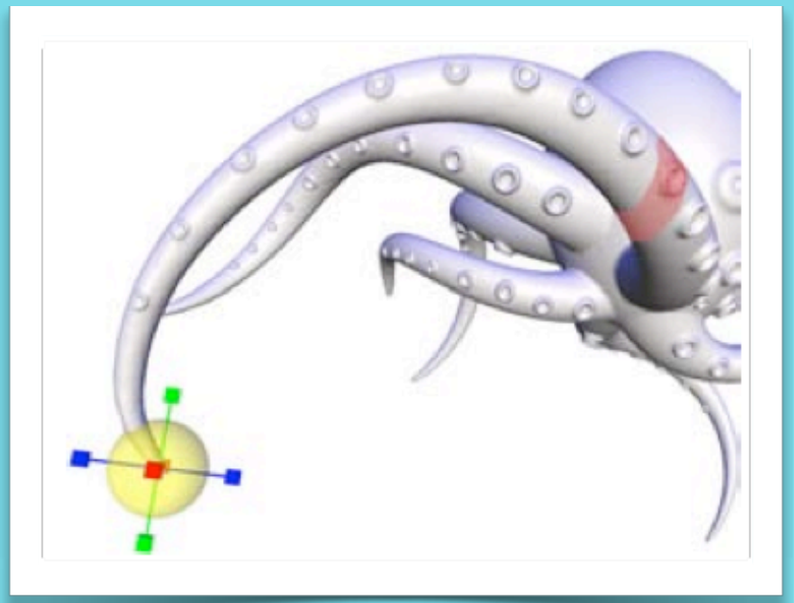
$$L_f^T L_f \mathbf{x}_f = L_f^T L \mathbf{x}' - L_f^T L_c \mathbf{x}_c$$

```
# Step 2: Create Full Equation

free_laplacian = full_laplacian[:,free_vertex_indices]
fixd_laplacian = full_laplacian[:,fixd_vertex_indices]

lhs = free_laplacian.transpose() * free_laplacian
rhs = free_laplacian.transpose() * (full_laplacian * orig_vertices) -
      free_laplacian.transpose() * (fixd_laplacian * fixd_vertices)
```

3. Solve



```
# Step 3: Solve

free_vertices = conjugate_gradient(lhs, rhs)
```

Full Implementation

```
# Step 1: Create Laplacian

full_laplacian = sparse_matrix((num_vertices, num_vertices))
for ii in vertices:
    adjacent_vertices = vertices_adjacent_to(ii)
    for jj in adjacent_vertices:
        full_laplacian[ii,jj] = -1.0
    full_laplacian[ii,ii] = len(adjacent_vertices)

# Step 2: Create Full Equation

free_laplacian = full_laplacian[:,free_vertex_indices]
fixd_laplacian = full_laplacian[:,fixd_vertex_indices]

lhs = free_laplacian.transpose() * free_laplacian
rhs = free_laplacian.transpose() * (full_laplacian * orig_vertices) -
      free_laplacian.transpose() * (fixd_laplacian * fixd_vertices)

# Step 3: Solve

free_vertices = conjugate_gradient(lhs, rhs)
```

Video

Laplacian Mesh Editing

A short editing session
with the *Octopus*