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

Segmentation


...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.
Objective Function

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

\[
\min_x f(x)
\]

• 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

\[ \min_x f(x) \]

How do we find the minimum of \( f \)?

Set the *derivative* to 0!

\[ \frac{df}{dx} = 0 \]
What are the Variables?

\[
\min_x f(x)
\]

The variables are the positions of each vertex:

\[
x = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n]^T
\]
What is a good objective function?

Let's look at one edge.
One Edge

Black board.
Multiple Edges

Black board.
Another Interpretation

\[ L_x = L_{x'} \]

- 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

\[ Lx = Lx' \]

Unconstrained Equation

\[ L_f^T L_f x_f = L_f^T Lx' - L_f^T L_c x_c \]

Constrained Equation
Implementation

1. Create Laplacian
2. Create Full Equation
3. Solve
# Step 1: Create Laplacian

```python
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)
```

\[
L x = L x'
\]
2: Create Full Equation

\[ L^T_f L_f x_f = L^T_f L x'_f - L^T_f L_c x_c \]

# Step 2: Create Full Equation

```python
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

```python
free_vertices = conjugate_gradient(lhs, rhs)
```
# 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