Interactive Explainers for Geometry Processing Algorithms

Jerry Yin, Jeffrey Goh
Introduction

- We are creating a set of interactive course notes ("interactive explainers") for the undergraduate geometric modelling course.
- We are planning on creating articles on two topics: half-edge data structures (this week’s demo), and mesh subdivision.
Meshes

- Meshes are graphs with vertices and edges, plus a set of faces.
- Each face is a cycle of vertices.
- Representing faces as a set of cycles is compact (good for storage) but bad for mesh algorithms.
  - Asking questions like “are \( v_3 \) and \( v_5 \) connected?” requires searching through all the faces!

\[
\begin{align*}
v_1 &= (1, 4) & v_2 &= (3, 4) & v_3 &= (2, 2) \\
v_4 &= (4, 2) & v_5 &= (1, 0) & v_6 &= (3, 0)
\end{align*}
\]

\[
V = \{v_1, v_2, v_3, v_4, v_5, v_6\}
\]

\[
F = \{(v_1, v_3, v_2), (v_2, v_3, v_4), (v_1, v_5, v_3), (v_3, v_5, v_6, v_4)\}
\]
Half-edge data structures

- Represent each edge as a pair of *half-edges*, each going in opposite directions.
- Each face is represented by a counter-clockwise cycle of half-edges.
- Boundary is represented by a clockwise cycle of half-edges.
- Each half-edge stores next and previous half-edges, its twin, its origin vertex, and its corresponding face.
  - Can answer most common queries in ~constant time.
**Half-edge vis**

- **OBJ Editor view** allows user to edit a mesh defined in the popular OBJ format.
  - Specify positions and connectivity
- **Visual view** shows a half-edge diagram.
  - Colour encodes boundary / interior half-edge
Half-edge vis

- Memory layout view shows all the records stored in the data structure.
  - Colours are the same as in the half-edge diagram.

<table>
<thead>
<tr>
<th>VERTEX</th>
<th>COORDINATE</th>
<th>INCIDENT EDGE</th>
<th>FACE</th>
<th>EDGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0</td>
<td>(0, 1, 0)</td>
<td>e1</td>
<td>f0</td>
<td>e2</td>
</tr>
<tr>
<td>v1</td>
<td>(0.9, -0.3, 0)</td>
<td>e2</td>
<td>f1</td>
<td>e5</td>
</tr>
<tr>
<td>v2</td>
<td>(-0.5, -0.3, 0.4)</td>
<td>e4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td>(-0.5, -2.3, 0.3)</td>
<td>e6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HALF-EDGE</th>
<th>ORIGIN</th>
<th>TWIN</th>
<th>INCIDENT</th>
<th>FACE</th>
<th>NEXT</th>
<th>PREV</th>
</tr>
</thead>
<tbody>
<tr>
<td>e0</td>
<td>v0</td>
<td>e1</td>
<td>e0</td>
<td>f0</td>
<td>e2</td>
<td>e3</td>
</tr>
<tr>
<td>e1</td>
<td>v2</td>
<td>e0</td>
<td>e1</td>
<td>f0</td>
<td>e2</td>
<td>e4</td>
</tr>
<tr>
<td>e2</td>
<td>v0</td>
<td>e3</td>
<td>e3</td>
<td>f0</td>
<td>e4</td>
<td>e1</td>
</tr>
</tbody>
</table>
Half-edge vis

- Interactivity:
  - Can edit OBJ contents
  - Can drag vertices to change position
  - Linked highlighting
  - Idea (might not be feasible): can edit memory layout (and corrupt / uncorrupt data structure)
Implementation

- 2D Visualization:
  - Multiple single pages generated using Idyll.
  - Create using D3 and implement it with Idyll.

- Idyll:
  - a markup language and toolkit for writing interactive articles.
  - can be integrated with React / D3 to create custom components.
Current progress (demo)

- Can edit vertex positions, diagram and tables update automatically
- Editing connectivity can be a bit buggy, but removing faces works

**Half-Edge Data Structures**

```plaintext
$\text{Enter your mesh definition in OBJ format below...}$

v 1.0 4.0 0.0
v 3.0 4.0 0.0
v 0.0 2.0 0.0
v 2.0 2.0 0.0
v 4.0 2.0 0.0
v 1.0 0.0 0.0
v 3.0 0.0 0.0
f 1 3 4
f 1 4 2
f 2 4 5
f 3 6 4
f 4 6 7
f 4 7 5
```

**Memory Layout**

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Coordinate</th>
<th>Incident edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_0$</td>
<td>(1, 4, 0)</td>
<td>todo</td>
</tr>
<tr>
<td>v_1$</td>
<td>(3, 4, 0)</td>
<td>todo</td>
</tr>
<tr>
<td>v_2$</td>
<td>(0, 2, 0)</td>
<td>todo</td>
</tr>
<tr>
<td>v_3$</td>
<td>(2, 2, 0)</td>
<td>todo</td>
</tr>
<tr>
<td>v_4$</td>
<td>(4, 2, 0)</td>
<td>todo</td>
</tr>
<tr>
<td>v_5$</td>
<td>(1, 0, 0)</td>
<td>todo</td>
</tr>
<tr>
<td>v_6$</td>
<td>(3, 0, 0)</td>
<td>todo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Face</th>
<th>Half-edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>todo</td>
<td>todo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Half-edge</th>
<th>Origin</th>
<th>Twin</th>
<th>Incident face</th>
<th>Next</th>
<th>Prev</th>
</tr>
</thead>
<tbody>
<tr>
<td>todo</td>
<td>todo</td>
<td>todo</td>
<td>todo</td>
<td>todo</td>
<td>todo</td>
</tr>
</tbody>
</table>
Current progress (demo)

- Can edit vertex positions, diagram and tables update automatically
- Editing connectivity can be a bit buggy, but removing faces works