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!

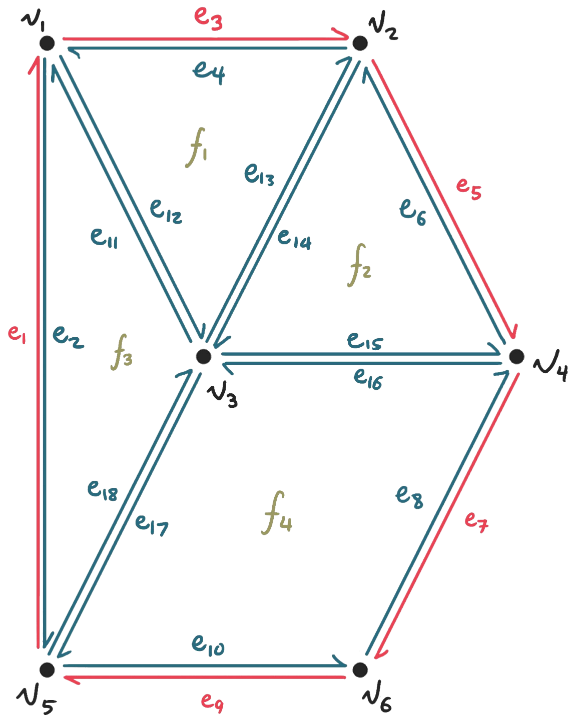
$$v_1 = (1,4) \qquad v_2 = (3,4) \qquad v_3 = (2,2)$$
$$v_4 = (4,2) \qquad v_5 = (1,0) \qquad v_6 = (3,0)$$

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

$$F = \{(v_1, v_3, v_2), (v_2, v_3, v_4), (v_1, v_5, v_3), (v_3, v_5, v_6, v_4)\}$$

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.



$v_1 = (1,4)$ $v_2 = (3,4)$ $v_3 = (2,2)$

$v_4 = (4,2)$ $v_5 = (1,0)$ $v_6 = (3,0)$

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

$$F = \{(v_1, v_3, v_2), (v_2, v_3, v_4), (v_1, v_5, v_3), (v_3, v_5, v_6, v_4)\}$$

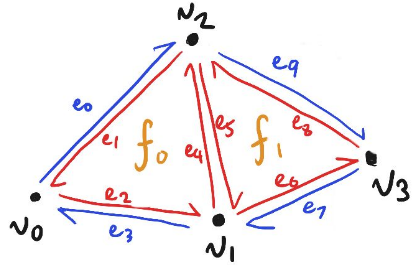
Half-edge vis

- 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

OBJ EDITOR

```
v 0.000000 1.000000 0.000000
v 0.942809 -0.333333 0.000000
v -0.471405 -0.333333 0.400000
v -0.471405 -2.333333 0.300000
f 1 2 3
f 2 4 3
```

VISUAL



MEMORY LAYOUT

VERTEX	COORDINATE	INCIDENT EDGE
v_0	(0, 1, 0)	e_1
v_1	(0.9, -0.3, 0)	e_2
v_2	(-0.5, -0.3, 0.4)	e_4
v_3	(-0.5, -2.3, 0.3)	e_6

FACE	EDGE
f_0	e_2
f_1	e_5

HALF-EDGE	ORIGIN	TWIN	INCIDENT FACE	NEXT	PREV
e_0	v_0	e_1	$\emptyset$	e_9	e_3
e_1	v_2	e_0	f_0	e_2	e_4
e_2	v_0	e_3	f_0	e_4	e_1
⋮					

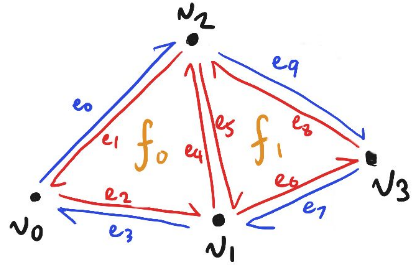
Half-edge vis

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

OBJ EDITOR

```
v 0.000000 1.000000 0.000000
v 0.942809 -0.333333 0.000000
v -0.471405 -0.333333 0.400000
v -0.471405 -2.333333 0.300000
f 1 2 3
f 2 4 3
```

VISUAL



MEMORY LAYOUT

VERTEX	COORDINATE	INCIDENT EDGE
v_0	(0, 1, 0)	e_1
v_1	(0.9, -0.3, 0)	e_2
v_2	(-0.5, -0.3, 0.4)	e_4
v_3	(-0.5, -2.3, 0.3)	e_6

FACE	EDGE
f_0	e_2
f_1	e_5

HALF-EDGE	ORIGIN	TWIN	INCIDENT FACE	NEXT	PREV
e_0	v_0	e_1	Ø	e_9	e_3
e_1	v_2	e_0	f_0	e_2	e_4
e_2	v_0	e_3	f_0	e_4	e_1
⋮					

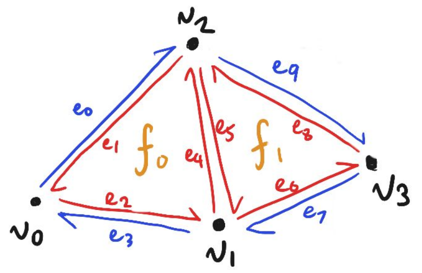
Half-edge vis

- 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)

OBJ EDITOR

```
v 0.000000 1.000000 0.000000
v 0.942809 -0.333333 0.000000
v -0.471405 -0.333333 0.400000
v -0.471405 -2.333333 0.300000
f 1 2 3
f 2 4 3
```

VISUAL



MEMORY LAYOUT

VERTEX	COORDINATE	INCIDENT EDGE
v_0	(0, 1, 0)	e_1
v_1	(0.9, -0.3, 0)	e_2
v_2	(-0.5, -0.3, 0.4)	e_4
v_3	(-0.5, -2.3, 0.3)	e_6

FACE	EDGE
f_0	e_2
f_1	e_5

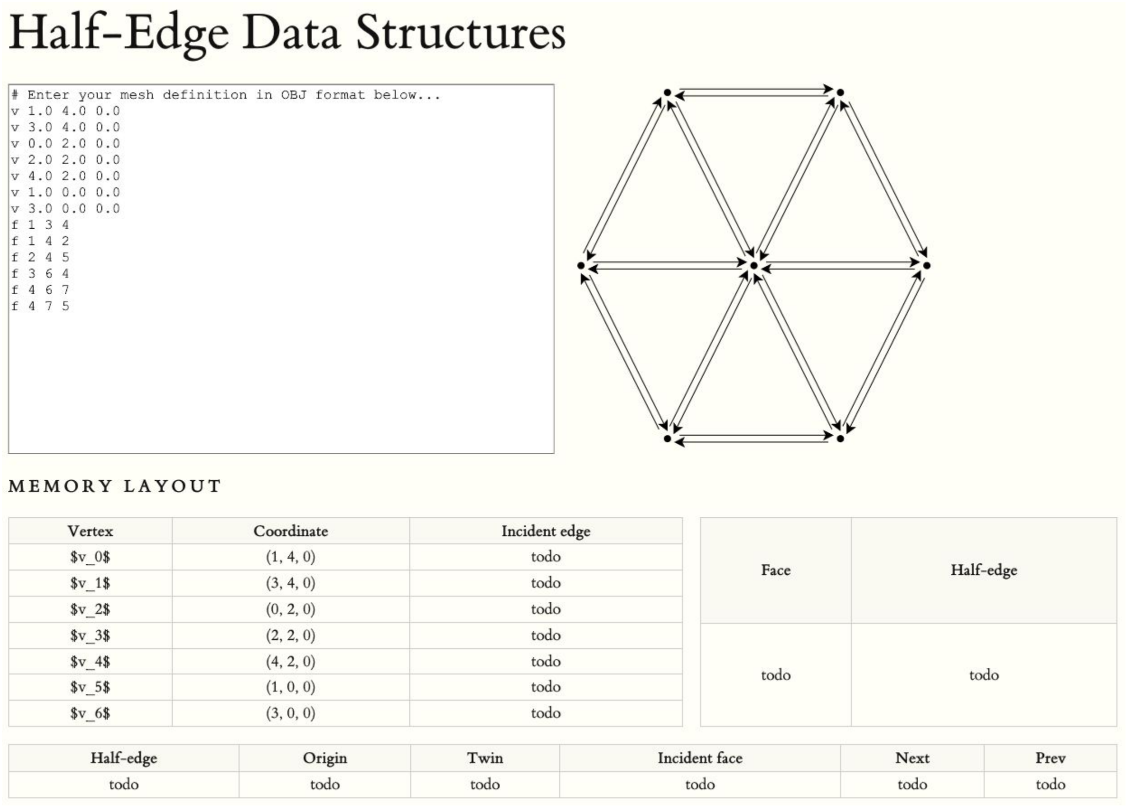
HALF-EDGE	ORIGIN	TWIN	INCIDENT FACE	NEXT	PREV
e_0	v_0	e_1	∅	e_9	e_3
e_1	v_2	e_0	f_0	e_2	e_4
e_2	v_0	e_3	f_0	e_4	e_1
⋮					

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

```
# 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

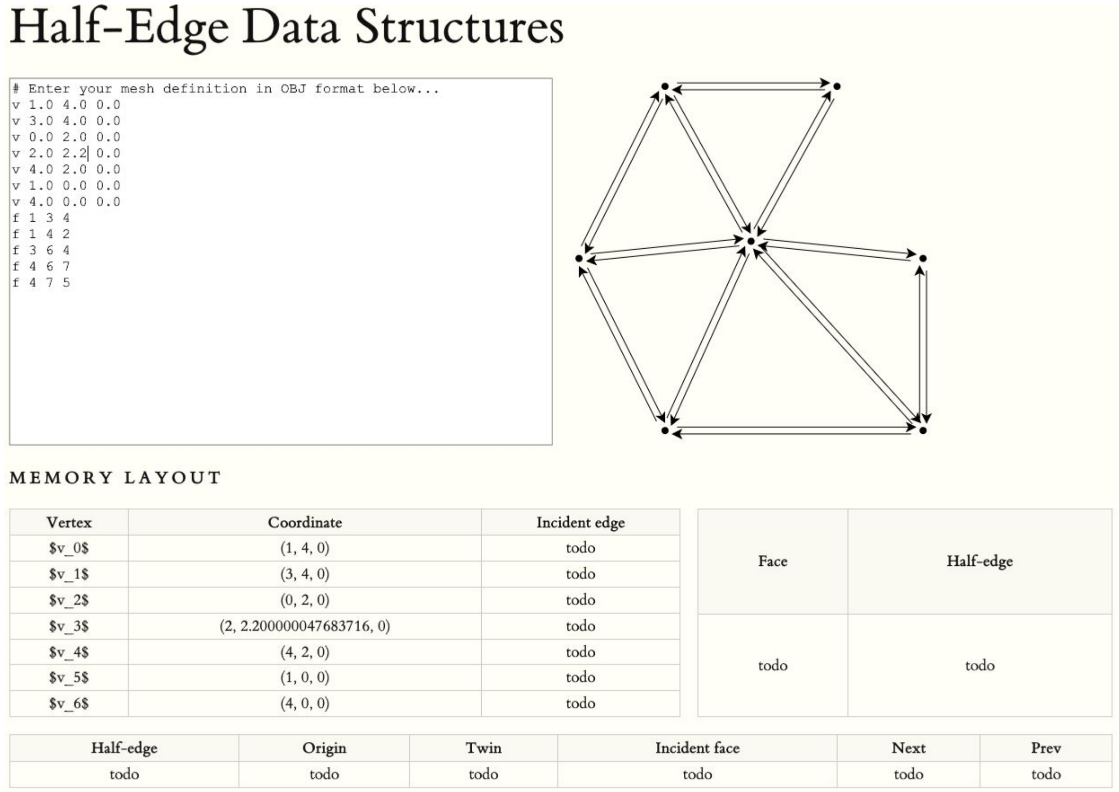
Vertex	Coordinate	Incident edge
v_0	(1, 4, 0)	todo
v_1	(3, 4, 0)	todo
v_2	(0, 2, 0)	todo
v_3	(2, 2, 0)	todo
v_4	(4, 2, 0)	todo
v_5	(1, 0, 0)	todo
v_6	(3, 0, 0)	todo

Face	Half-edge
todo	todo

Half-edge	Origin	Twin	Incident face	Next	Prev
todo	todo	todo	todo	todo	todo

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

```
# 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 2.2 0.0
v 4.0 2.0 0.0
v 1.0 0.0 0.0
v 4.0 0.0 0.0
f 1 3 4
f 1 4 2
f 3 6 4
f 4 6 7
f 4 7 5
```

MEMORY LAYOUT

Vertex	Coordinate	Incident edge
v_0	(1, 4, 0)	todo
v_1	(3, 4, 0)	todo
v_2	(0, 2, 0)	todo
v_3	(2, 2.200000047683716, 0)	todo
v_4	(4, 2, 0)	todo
v_5	(1, 0, 0)	todo
v_6	(4, 0, 0)	todo

Face	Half-edge
todo	todo

Half-edge	Origin	Twin	Incident face	Next	Prev
todo	todo	todo	todo	todo	todo