CPSC 424 - Tutorial 9

March 24th, 2025

TA: Suzuran Takikawa (stakikaw@cs.ubc.ca)

Half-edge data structures

Exercise 1. Suppose we are given the following 2D mesh specified in terms of its vertices V and faces F.

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

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

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

Half-edge data structures

Exercise 1. Suppose we are given the following 2D mesh specified in terms of its vertices V and faces F.

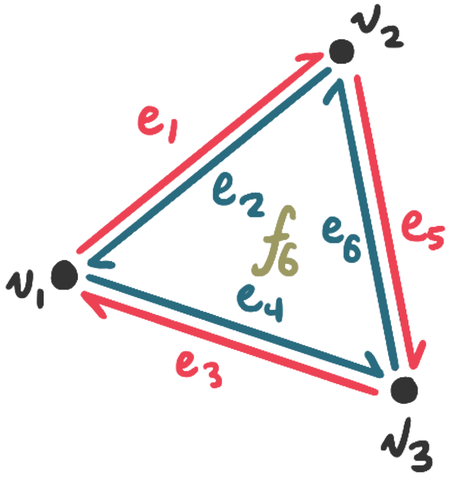
Draw the half-edge diagram corresponding to the mesh. Assume that we create boundary half-edges as appropriate so that every edge always corresponds to two half-edges.

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

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

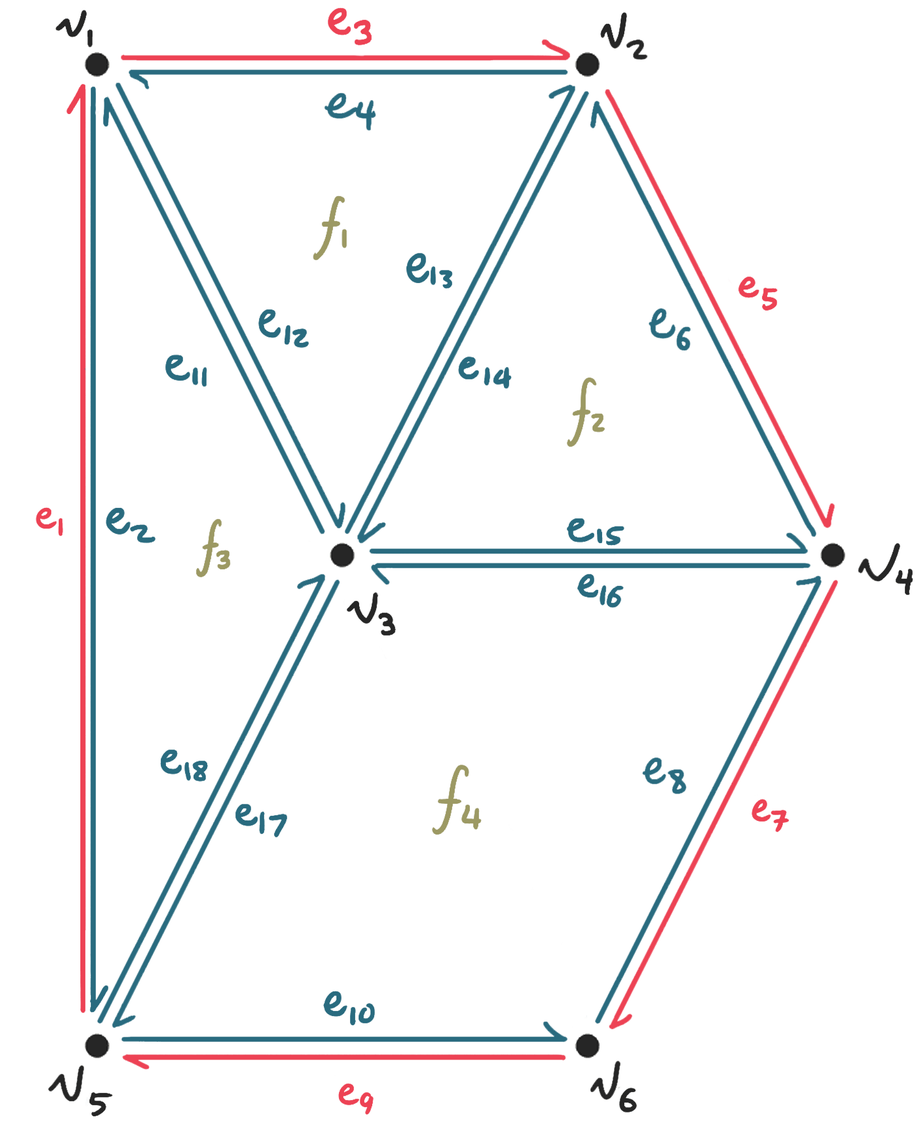
$$F = \{(v_1, v_3, v_2), (v_2, v_3, v_4), (v_1, v_5, v_3), (v_3, v_5, v_6, v_4)\}$$

Note: Face vertices are specified in winding order



Half-edge data structures

Half-edge index	*Origin*	*Twin*	*Next*	*Prev*	*Incident face*
1	v_5	e_2	e_3	e_9	$\varnothing$
2	v_1	e_1	e_{18}	e_{11}	f_3
3	v_1	e_4	e_5	e_1	$\varnothing$
4	v_2	e_3	e_{12}	e_{13}	f_1
5	v_2	e_6	e_7	e_3	$\varnothing$
6	v_4	e_5	e_{14}	e_{15}	f_2
7	v_4	e_8	e_9	e_5	$\varnothing$
8	v_6	e_7	e_{16}	e_{10}	f_4
9	v_6	e_{10}	e_1	e_7	$\varnothing$
10	v_5	e_9	e_8	e_{17}	f_4
11	v_3	e_{12}	e_2	e_{18}	f_3
12	v_1	e_{11}	e_{13}	e_4	f_1
13	v_3	e_{14}	e_4	e_{12}	f_1
14	v_2	e_{13}	e_{15}	e_6	f_2
15	v_3	e_{16}	e_6	e_{14}	f_2
16	v_4	e_{15}	e_{17}	e_8	f_4
17	v_3	e_{18}	e_{10}	e_{16}	f_4
18	v_5	e_{17}	e_{11}	e_2	f_3



Half-edge data structures

Given the following declaration for a half-edge structure:

Exercise 2. Write pseudocode for an algorithm to compute the centroid of a face f

We define the “pseudo centroid” of a face to be:

$$\sum_{v \in f} \frac{v}{|f|}$$

where $|f|$ is the number of vertices in f

```
// HalfEdge structure
struct HalfEdge {
    HalfEdge* next;
    HalfEdge* prev;
    HalfEdge* twin;
    Vertex* origin;
    Face* face;
};

// Vertex structure
struct Vertex {
    Vec3 position;
    HalfEdge* halfedge; // a half-edge that goes out of this vertex
};

// Face structure
struct Face {
    HalfEdge* halfedge; // a half-edge belonging to this face
};
```

Half-edge data structures

```
// Compute the pseudo-centroid of a face
Vec3 ComputePseudoCentroid(Face f) {
    Vec3 pseudo_centroid = Vec3(0, 0, 0);
    int count = 0;
    HalfEdge* start_it = f.halfedge;
    HalfEdge* it = start_it;
    do {
        pseudo_centroid += it->origin->position;
        count += 1;
        it = it->next;
    } while (it != start_it);
    return pseudo_centroid / count;
}
```

Half-edge data structures

Given the following declaration for a half-edge structure:

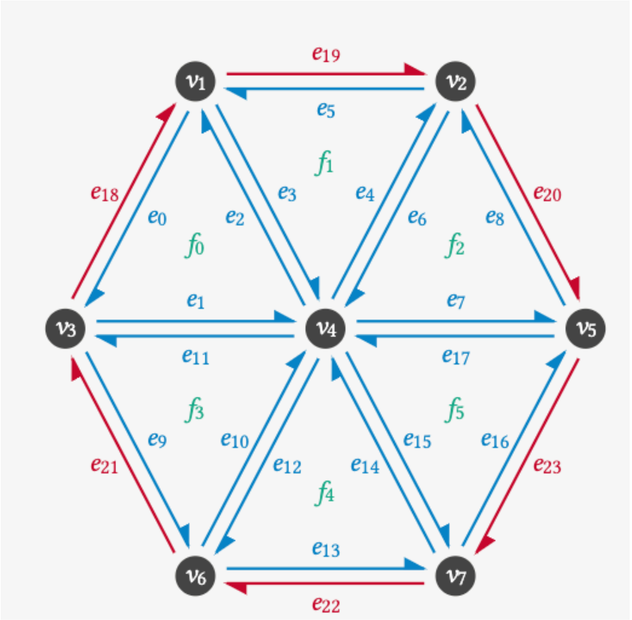
Exercise 3. Write pseudo-code for an algorithm which given a vertex returns the valence of the vertex

Reminder: valence = the number of incident vertices

```
// HalfEdge structure
struct HalfEdge {
    HalfEdge* next;
    HalfEdge* prev;
    HalfEdge* twin;
    Vertex* origin;
    Face* face;
};

// Vertex structure
struct Vertex {
    Vec3 position;
    HalfEdge* halfedge; // a half-edge that goes out of this vertex
};

// Face structure
struct Face {
    HalfEdge* halfedge; // a half-edge belonging to this face
};
```



Half-edge data structures

```
// Count the valence of a vertex
int CountValence(Vertex v) {
    int valence = 0;
    HalfEdge* start_it = v.halfedge;
    HalfEdge* it = start_it;
    do {
        valence += 1;
        it = it->prev->twin;
    } while (it != start_it);
    return valence;
}
```