COS 426 : Precept 4
Half-Edge
Agenda

• Assignment 2 description

• Half-edge data structure
  • Traversal
  • Modification
Assignment 2

- Part 1 - Analysis
  - Implement traversal operation
  - Calculate mesh properties
    - Vertex normal, avg. edge length, etc.
- Part 2 - Filters
  - Filters and Warps similar to assignment 1
  - Topological modifiers
Meshes

- Images had implicit adjacency information
  - Grid around a pixel
  - Easy to express operations

- What about meshes?
  - How to apply smoothing?
Meshes

- Meshes can be quite dense
Meshes

- Meshes can be quite dense
Meshes

- Meshes can be quite dense
Meshes

- How to access adjacency information quickly?
Meshes

- How to access adjacency information quickly?
Meshes

- How to access adjacency information quickly?
Meshes

- How to access adjacency information quickly?
Meshes

- How to access adjacency information quickly?
Meshes

- How to access adjacency information quickly?
Meshes

• How to access adjacency information quickly?
Meshes

• How to access adjacency information quickly?
Meshes

- How to access adjacency information quickly?

One - Ring Neighborhood
# Half-Edge Data Structure

<table>
<thead>
<tr>
<th>Half Edge</th>
<th>Vertex</th>
<th>Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex</td>
<td>Position</td>
<td>Half-Edge</td>
</tr>
<tr>
<td>Opposite Half-Edge</td>
<td>Outgoing Half-Edge</td>
<td>...</td>
</tr>
<tr>
<td>Face</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Next Half-Edge</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Half-Edge Data Structure

- Half-Edge
- Vertex
- Opposite Half-Edge
- Face
- Next Half-Edge
Half-Edge Data Structure

Vertex

Location

Outgoing Half-Edge

…

One of the two outgoing edges will be used
Half-Edge Data Structure

One of the three edges will be used
Traversal

• How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversals

How to get one-ring neighbors?
Traversal

• How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversing

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversal

• How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversal

• How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversal

• How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversals

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?
Traversal

- How to get one-ring neighbors?

```java
original_he = vertex.he;
current = original_he;
do {
    // do something with data
    current = he.opposite.next;
} while (he != original_he)
```

- Assignment will ask you for other kind of adjacency queries

- Vertices around Face, Faces around Vertex etc.
Traversals

- Vertex Normals are defined as weighted average of adjacent faces (weighted by face area).
- How would you compute vertex normals given per face normal and area?
Data Structure Modification

• Split Edge

v1

v2

f1

f2
Data Structure Modification

- Split Edge

\[ v_1 \rightarrow f_1 \rightarrow f_2 \rightarrow v_2 \]
Data Structure Modification

- Split Edge
Data Structure Modification

- Split Edge

\[ v_1 \rightarrow h_{e1} \rightarrow v_2 \]

\[ v_1 \leftarrow f_{2} \rightarrow v_2 \]

\[ f_{1} \rightarrow v_2 \rightarrow h_{e1} \rightarrow v_1 \]
Data Structure Modification

- Split Edge

\[ v1 \xrightarrow{he1} v2 \xleftarrow{he2} \]
\[ f1 \]
\[ f2 \]
Data Structure Modification

- Split Edge

v1 \(\rightarrow\) f1 \(\rightarrow\) he1 \(\rightarrow\) f2 \(\rightarrow\) v2
Data Structure Modification

- Split Edge

v1

he1

f1

he1_next

v2

he2

f2
Data Structure Modification

- Split Edge
Data Structure Modification

• Split Edge

\[ \text{v1} \quad \text{he1} \quad \text{f1} \quad \text{he2} \quad \text{f2} \quad \text{v2} \]

\[ \text{he1} \quad \text{he1\_next} \]
Data Structure Modification

- Split Edge

```plaintext
v1
   \(\text{he1}\)
   \(\text{he2}\)
   \(\text{he1}\_\text{next}\)

v2
   \(\text{he2}\_\text{next}\)
```

- \(f_1\)
- \(f_2\)
Data Structure Modification

- Split Edge

\[ v_1 \quad \text{he1} \quad f_1 \quad \text{he2} \quad v_2 \]

\[ \text{he1}_\text{next} \quad f_2 \quad \text{he2}_\text{next} \]
Data Structure Modification

- Split Edge

\[ v_3 = \text{addVertex}( \text{weightedAvgPos}(v_1, v_2, \text{factor}) ); \]
Data Structure Modification

- Split Edge

\[ v_3 = \text{addVertex}(\text{weightedAvgPos}(v_1, v_2, \text{factor}) \); \]
Data Structure Modification

• Split Edge

\[ v_3 = \text{addVertex}( \text{weightedAvgPos}(v_1, v_2, \text{factor}) ); \]

\[ \text{he1.vertex} = v_3; \]
\[ \text{he2.vertex} = v_3; \]
Data Structure Modification

- Split Edge

\[ v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) ) ; \]

\[ \text{he1.vertex} = v3 ; \]

\[ \text{he2.vertex} = v3 ; \]
Data Structure Modification

- Split Edge

\[ v_3 = \text{addVertex}(\text{weightedAvgPos}(v_1, v_2, \text{factor})) ; \]

\[ \text{he1.vertex} = v_3; \]

\[ \text{he2.vertex} = v_3; \]
Data Structure Modification

- Split Edge

\[ v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) ); \]

\[ \text{he1.vertex} = v3; \]
\[ \text{he2.vertex} = v3; \]
Data Structure Modification

- Split Edge

\[
\text{v3} = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) );
\]

\[
\text{he1.vertex} = \text{v3};
\]

\[
\text{he2.vertex} = \text{v3};
\]
Data Structure Modification

- Split Edge

\[ v3 = \text{addVertex}(\text{weightedAvgPos}(v1, v2, \text{factor}) ); \]

he1.vertex = v3;
he2.vertex = v3;
Data Structure Modification

• Split Edge

\[ v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) ); \]

\[ \text{he1.vertex} = v3; \]

\[ \text{he2.vertex} = v3; \]
Data Structure Modification

- Split Edge

\[
v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) );
\]
\[
\text{he1.vertex} = v3;
\]
\[
\text{he2.vertex} = v3;
\]
\[
\text{he3} = \text{addHalfEdge}( v3, v2, f1 );
\]
\[
\text{he4} = \text{addHalfEdge}( v3, v1, f2 );
\]
Data Structure Modification

- Split Edge

\[ v_3 = \text{addVertex}( \text{weightedAvgPos}(v_1, v_2, \text{factor}) ); \]

\[ \text{he1.vertex} = v_3; \]
\[ \text{he2.vertex} = v_3; \]

\[ \text{he3} = \text{addHalfEdge}( v_3, v_2, f_1 ); \]
\[ \text{he4} = \text{addHalfEdge}( v_3, v_1, f_2 ); \]
Data Structure Modification

- Split Edge

\[
v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) );
\]

\[
\text{he1}.\text{vertex} = v3;
\]

\[
\text{he2}.\text{vertex} = v3;
\]

\[
\text{he3} = \text{addHalfEdge}( v3, v2, f1 );
\]

\[
\text{he4} = \text{addHalfEdge}( v3, v1, f2 );
\]
Data Structure Modification

- Split Edge

\[ v3 = \text{addVertex( weightedAvgPos(v1, v2, factor) )}; \]
\[ \text{he1.vertex} = v3; \]
\[ \text{he2.vertex} = v3; \]
\[ \text{he3} = \text{addHalfEdge( v3, v2, f1 )}; \]
\[ \text{he4} = \text{addHalfEdge( v3, v1, f2 )}; \]
Data Structure Modification

- Split Edge

\[
v3 = \text{addVertex}(\ \text{weightedAvgPos}(v1, v2, \text{factor}) )\; ;
\]
\[
\text{he1.vertex} = v3;
\]
\[
\text{he2.vertex} = v3;
\]
\[
\text{he3} = \text{addHalfEdge}(v3, v2, f1 )\; ;
\]
\[
\text{he4} = \text{addHalfEdge}(v3, v1, f2 )\; ;
\]
Data Structure Modification

- **Split Edge**

\[ v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) ); \]

\[ \text{he1.vertex} = v3; \]
\[ \text{he2.vertex} = v3; \]

\[ \text{he3} = \text{addHalfEdge}( v3, v2, f1 ); \]
\[ \text{he4} = \text{addHalfEdge}( v3, v1, f2 ); \]

\[ \text{he1.next} = \text{he3}; \]
\[ \text{he2.next} = \text{he4}; \]
Data Structure Modification

• Split Edge

\[
v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) );
\]

\[
\text{he1.vertex} = v3;
\]

\[
\text{he2.vertex} = v3;
\]

\[
\text{he3} = \text{addHalfEdge}( v3, v2, f1 );
\]

\[
\text{he4} = \text{addHalfEdge}( v3, v1, f2 );
\]

\[
\text{he1.next} = \text{he3};
\]

\[
\text{he2.next} = \text{he4};
\]
Data Structure Modification

- Split Edge

v3 = addVertex( weightedAvgPos(v1, v2, factor) );
he1.vertex = v3;
he2.vertex = v3;
he3 = addHalfEdge( v3, v2, f1 );
he4 = addHalfEdge( v3, v1, f2 );
he1.next = he3;
he2.next = he4;
Data Structure Modification

- Split Edge

\[ v_3 = \text{addVertex}( \text{weightedAvgPos}(v_1, v_2, \text{factor}) ); \]

\[ \text{he1.vertex} = v_3; \]
\[ \text{he2.vertex} = v_3; \]
\[ \text{he3.next} = \text{he1}_\text{next}; \]
\[ \text{he4.next} = \text{he2}_\text{next}; \]

\[ \text{he3} = \text{addHalfEdge}( v_3, v_2, f_1 ); \]
\[ \text{he4} = \text{addHalfEdge}( v_3, v_1, f_2 ); \]

\[ \text{he1.next} = \text{he3}; \]
\[ \text{he2.next} = \text{he4}; \]
Data Structure Modification

- **Split Edge**

  
  ```
  v3 = addVertex( weightedAvgPos(v1, v2, factor) );
  he1.vertex = v3;
  he2.vertex = v3;
  he3 = addHalfEdge( v3, v2, f1 );
  he4 = addHalfEdge( v3, v1, f2 );
  he1.next = he3;
  he2.next = he4;
  ```

  ```
  he3.next = he1.next;
  he4.next = he2.next;
  ```
Data Structure Modification

- Split Edge

\[ v3 = \text{addVertex( weightedAvgPos(v1, v2, factor) )}; \]

\[ \text{he1.vertex} = v3; \]
\[ \text{he2.vertex} = v3; \]
\[ \text{he3.next} = \text{he1.next}; \]
\[ \text{he4.next} = \text{he2.next}; \]

\[ \text{he3} = \text{addHalfEdge( v3, v2, f1 )}; \]
\[ \text{he4} = \text{addHalfEdge( v3, v1, f2 )}; \]

\[ \text{he1.next} = \text{he3}; \]
\[ \text{he2.next} = \text{he4}; \]
Data Structure Modification

- Split Edge

\[ v_3 = \text{addVertex}( \text{weightedAvgPos}(v_1, v_2, \text{factor}) ); \]

\[ \text{he1.vertex} = v_3; \]
\[ \text{he2.vertex} = v_3; \]
\[ \text{he3} = \text{addHalfEdge}( v_3, v_2, f_1 ); \]
\[ \text{he4} = \text{addHalfEdge}( v_3, v_1, f_2 ); \]
\[ \text{he1.next} = \text{he3}; \]
\[ \text{he2.next} = \text{he4}; \]
\[ \text{he3.next} = \text{he1}\_\text{next}; \]
\[ \text{he4.next} = \text{he2}\_\text{next}; \]
\[ \text{he1.opposite} = \text{he4}; \]
\[ \text{he4.opposite} = \text{he1}; \]
\[ \text{he2.opposite} = \text{he3}; \]
\[ \text{he3.opposite} = \text{he2}; \]
v3 = addVertex( weightedAvgPos(v1, v2, factor) );

he1.vertex = v3;
he2.vertex = v3;

he3 = addHalfEdge( v3, v2, f1 );
he4 = addHalfEdge( v3, v1, f2 );

he1.next = he3;
he2.next = he4;

he1.opposite = he4;
he4.opposite = he1;
he2.opposite = he3;
he3.opposite = he2;

he3.next = he1_next;
he4.next = he2_next;
Data Structure Modification

• Split Edge

\[ v3 = \text{addVertex}( \text{weightedAvgPos}(v1, v2, \text{factor}) ); \]

\[ \text{he1.vertex} = v3; \]
\[ \text{he2.vertex} = v3; \]

\[ \text{he3} = \text{addHalfEdge}( v3, v2, f1 ); \]
\[ \text{he4} = \text{addHalfEdge}( v3, v1, f2 ); \]

\[ \text{he1.next} = \text{he3}; \]
\[ \text{he2.next} = \text{he4}; \]

\[ \text{he3.next} = \text{he1}\_\text{next}; \]
\[ \text{he4.next} = \text{he2}\_\text{next}; \]

\[ \text{he1}\_\text{opposite} = \text{he4}; \]
\[ \text{he4}\_\text{opposite} = \text{he1}; \]
\[ \text{he2}\_\text{opposite} = \text{he3}; \]
\[ \text{he3}\_\text{opposite} = \text{he2}; \]
Data Structure Modification

• Split Face
Data Structure Modification

- Split Face

\[ f_2 = \text{addFace}() \]
Data Structure Modification

• Split Face

```c
f2 = addFace();
```
Data Structure Modification

- Split Face

```java
f2 = addFace();
he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;
```
Data Structure Modification

- Split Face

```java
f2 = addFace();
he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;
```
• **Split Face**

\[
f_2 = \text{addFace}();
\]

\[
\text{he}_5 = \text{addHalfEdge}( v_4, v_2, f_1 );
\]

\[
\text{he}_6 = \text{addHalfEdge}( v_2, v_4, f_2 );
\]

\[
\text{he}_5.\text{opposite} = \text{he}_6;
\]

\[
\text{he}_6.\text{opposite} = \text{he}_5;
\]

\[
\text{he}_5.\text{next} = \text{he}_2;
\]

\[
\text{he}_3.\text{next} = \text{he}_5;
\]

\[
\text{he}_1.\text{next} = \text{he}_6;
\]

\[
\text{he}_6.\text{next} = \text{he}_4;
\]
Data Structure Modification

- Split Face

```c
f2 = addFace();
he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;
he5.next = he2;
he3.next = he5;
he1.next = he6;
he6.next = he4;
```
Data Structure Modification

• Split Face

```java
f2 = addFace();

he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;

he1.next = he6;
he6.next = he4;
he5.next = he2;
he3.next = he5;
he1.next = he6;
he6.next = he4;
```
Data Structure Modification

• Split Face

```plaintext
f2 = addFace();
he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;
he5.next = he2;
he3.next = he5;
he1.next = he6;
he6.next = he4;
```
Data Structure Modification

- Split Face

```java
f2 = addFace();

he5 = addHalfEdge( v4, v2, f1 );  he5.next = he2;
he6 = addHalfEdge( v2, v4, f2 );  he3.next = he5;
he5.opposite = he6;  he1.next = he6;
he6.opposite = he5;  he6.next = he4;
```
Data Structure Modification

• Split Face

```c
f2 = addFace();

he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;
he5.next = he2;
he3.next = he5;
he1.next = he6;
he6.next = he4;
f1.halfedge = he5;
f2.halfedge = he6;
```
Data Structure Modification

• Split Face

```plaintext
f2 = addFace();
he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;
he5.next = he2;
he3.next = he5;
he1.next = he6;
he6.next = he4;
f1.halfedge = he5;
f2.halfedge = he6;
```
Data Structure Modification

- Split Face

```plaintext
f2 = addFace();

he5 = addHalfEdge( v4, v2, f1 );
he6 = addHalfEdge( v2, v4, f2 );
he5.opposite = he6;
he6.opposite = he5;

he5.next = he2;
he3.next = he5;
he1.next = he6;
he6.next = he4;

f1.halfedge = he5;
f2.halfedge = he6;

Remember to re-link he4 and he1 to point to f2
```
Data Structure Modification

• How would you go about subdividing a quad face?
  • You’re given split edge and split face
  • Just use those - guaranteed validity of dataset after use!
• Part of the assignment
  • Think about it during tomorrows class!
Cube Cheat Sheet

Vertex neighbors:

• 0 -> \{ 1, 3, 4 \}
• 1 -> \{ 0, 2, 5 \}
• 2 -> \{ 1, 6, 3 \}
• 3 -> \{ 0, 2, 7 \}
• 4 -> \{ 7, 5, 0 \}
• 5 -> \{ 2, 3 \}
• 6 -> \{ 5, 2, 7 \}