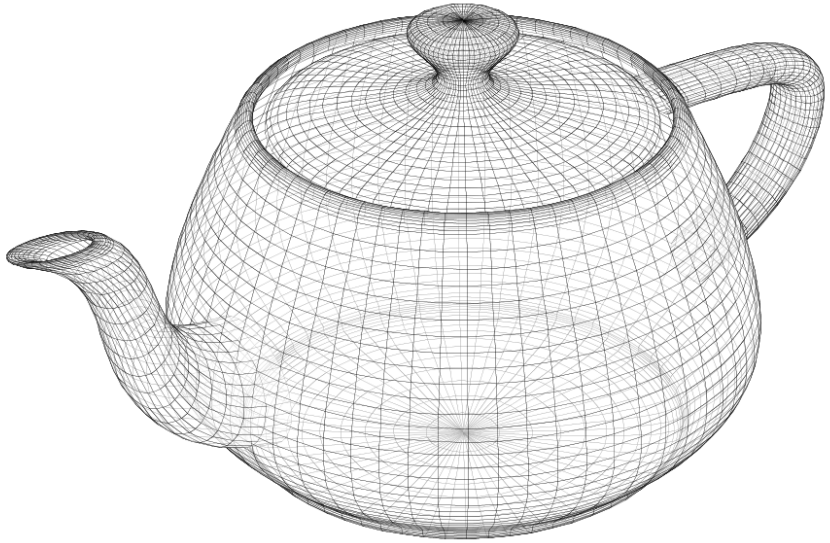
Geometric Modeling: Data Structures for Polygonal Mesh

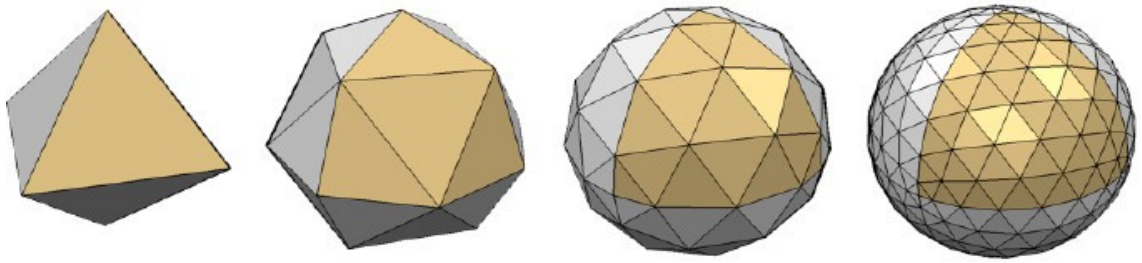


CS 418: Interactive Computer Graphics

Professor Eric Shaffer

ILLINOIS

Mesh Data Structures



Need to store

- Geometry
- Connectivity

Can be used as file formats or internal formats

Considerations

- Space
- Efficient operations

Mesh processing has different requirements than rendering

- Example: Deforming a mesh when simulating physics…

ILLINOIS

Mesh Data Structure: Face Set (STL)

- face:
  - 3 positions

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

$2v \approx f$

36 B/f ≈ 72 B/v

no connectivity!

We will be focusing on triangle meshes…but all of these structures generalize to other polygons…storage requirements will change.

Designed in 1987 for stereolithography (3D printing technology)

No explicit information about which vertices are shared by which triangles
Consider how efficiently we could:

- Deform mesh by moving vertices
- Lookup vertex neighbor information

Mesh Data Structure: Indexed Face Set (OBJ)

- vertex:
  - position
- face:
  - vertex indices

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

Triangles
v_{11} v_{12} v_{13}
...
...
...
...
v_{F1} v_{F2} v_{F3}

24B/v

12 B/v + 12 B/f = 36 B/v

no neighborhood info

2v ≃ F

The OBJ file format is a popular storage format for meshes

Developed by Wavefront Technologies…now part of AutoDesk

Text files with .obj extension

```
# List of geometric vertices
v 0.123 0.234 0.345
v …
v …
# List of triangles
f 1 3 4
f 2 4 5
…
```

Indexed Face Set

One block of data are the vertices

- Each vertex is a set of 3 coordinates
- Often referred to as the geometry of the mesh

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

Triangles
v_{11} v_{12} v_{13}
...
...
...
...
v_{F1} v_{F2} v_{F3}

Another block of data is the set of triangles

- Each triangle is set of 3 integers vertex IDs
- The vertex IDs are indices into the vertex block

What are some advantages of this representation?

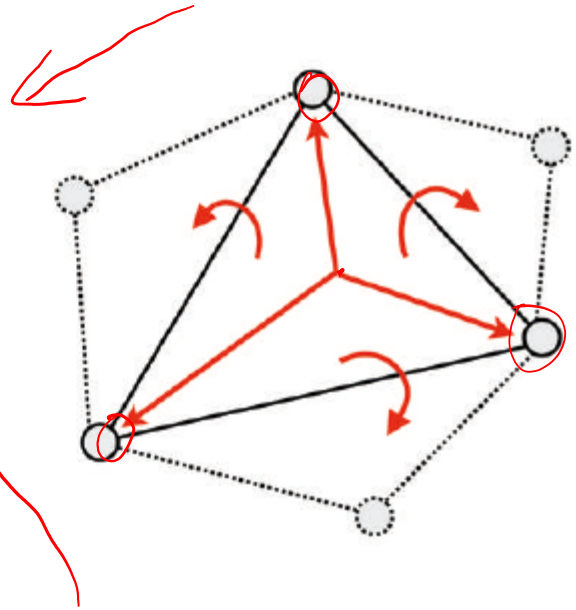
Disadvantages?

ILLINOIS

Face-Based Connectivity

Vertex	
Point	position
FaceRef	face

Face	
VertexRef	vertex[3]
FaceRef	neighbor[3]



Storage: 64

B/v

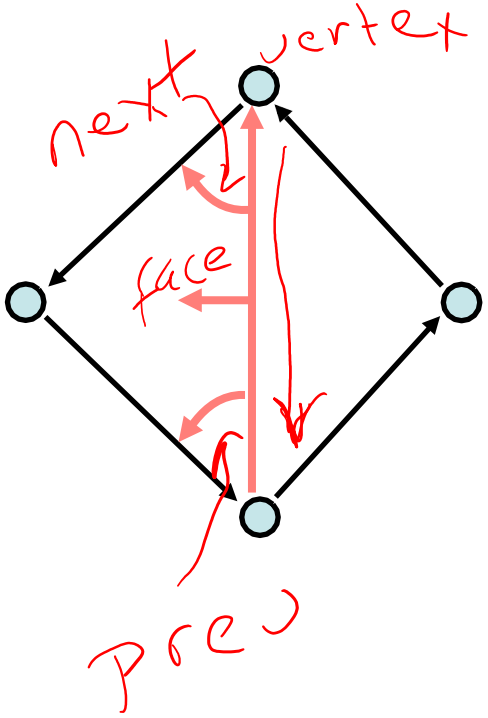
No edges: gathering vertex neighbors is not straight-forward…multiple cases

Halfedge Data Structure

Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
HalfedgeRef	prev
HalfedgeRef	opposite



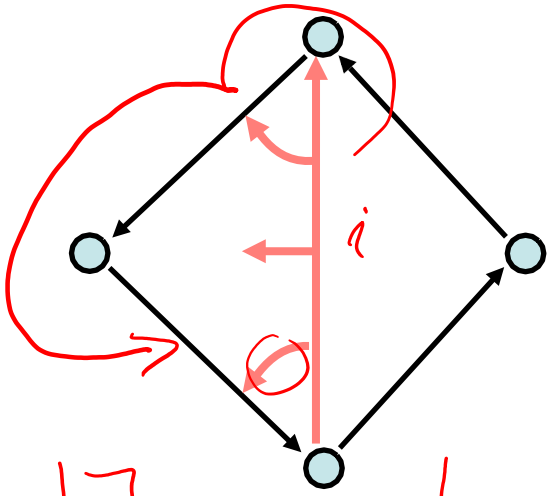
Halfedge Data Structure

H

Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
~~HalfedgeRef~~	~~prev~~
~~HalfedgeRef~~	~~opposite~~



H[i].prev = H[H[i].next].next

H[0] opp
H[1] opp

if i is even → H[i].opp = i+1
if i is odd → H[i].opp = i-1

Halfedge Data Structure

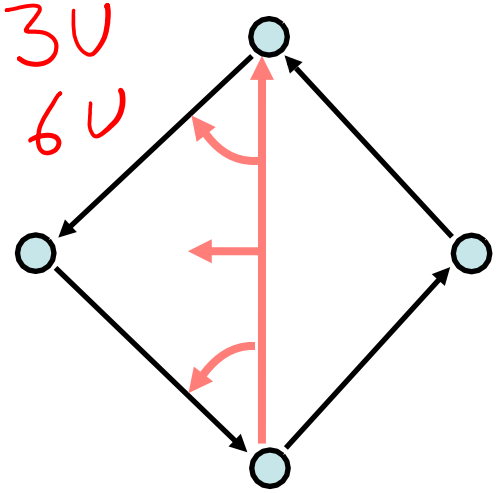
Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
~~HalfedgeRef~~	~~prev~~
~~HalfedgeRef~~	~~opposite~~

$E \approx 3V$

$HE \approx 6V$



vertex + Face + Halfedge

16 B/V + 4 B/F + 12 B/H

16 B/V + 8/V + 72 B/V = 96 B/V

96 to 144 B/v

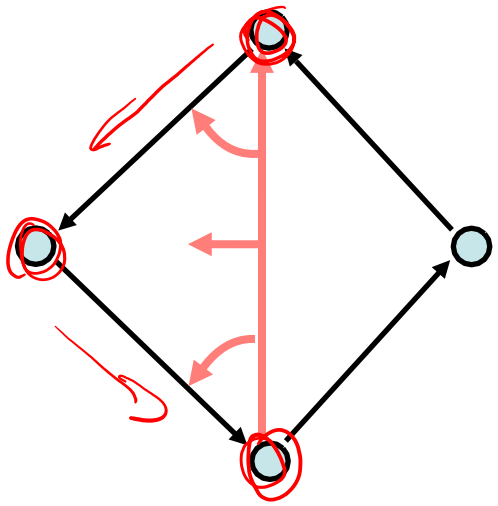
no case distinctions during traversal

Halfedge Data Structure : Gathering Vertices

Vertex	
Point	position
HalfedgeRef	halfedge

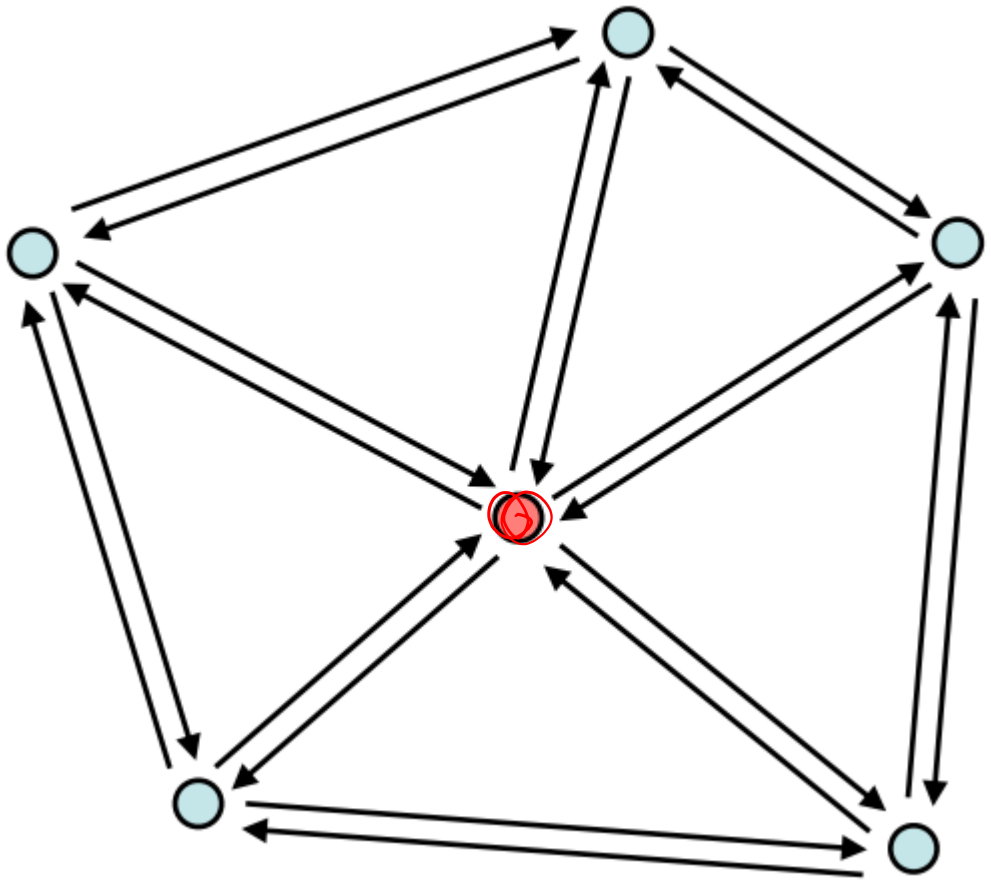
Face	
HalfedgeRef	halfedge

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
HalfedgeRef	prev
HalfedgeRef	opposite



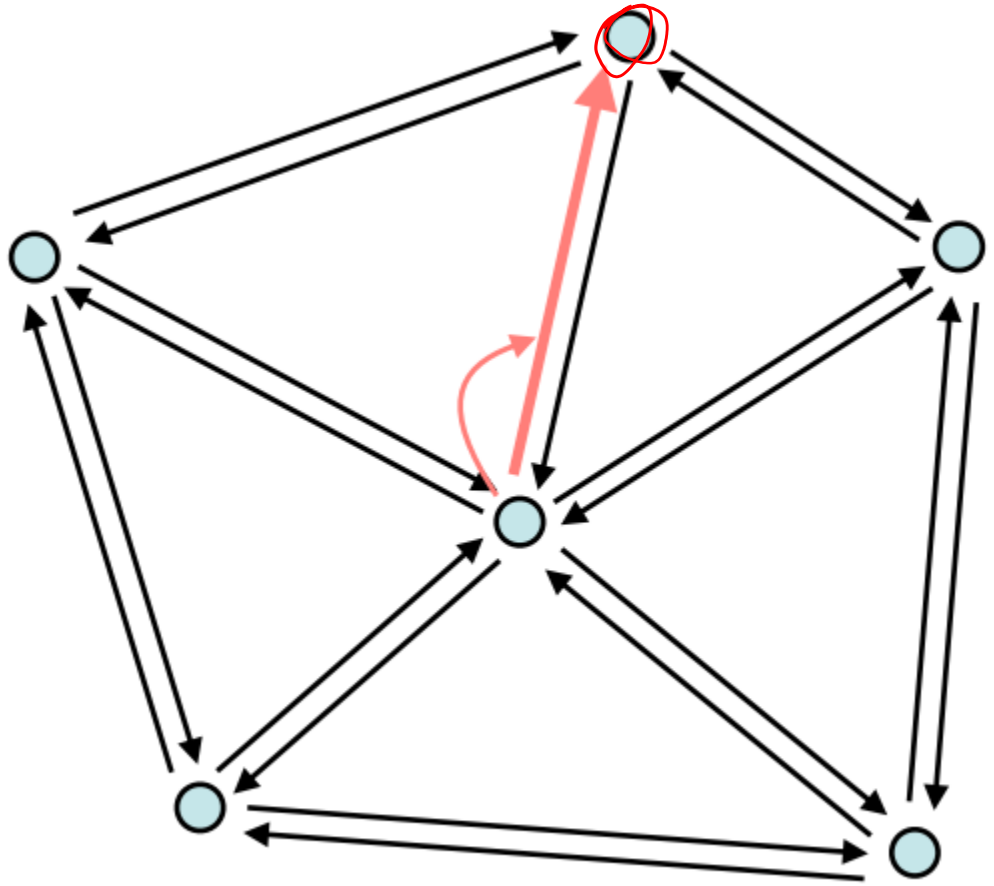
One-Ring Traversal

1. Start at vertex



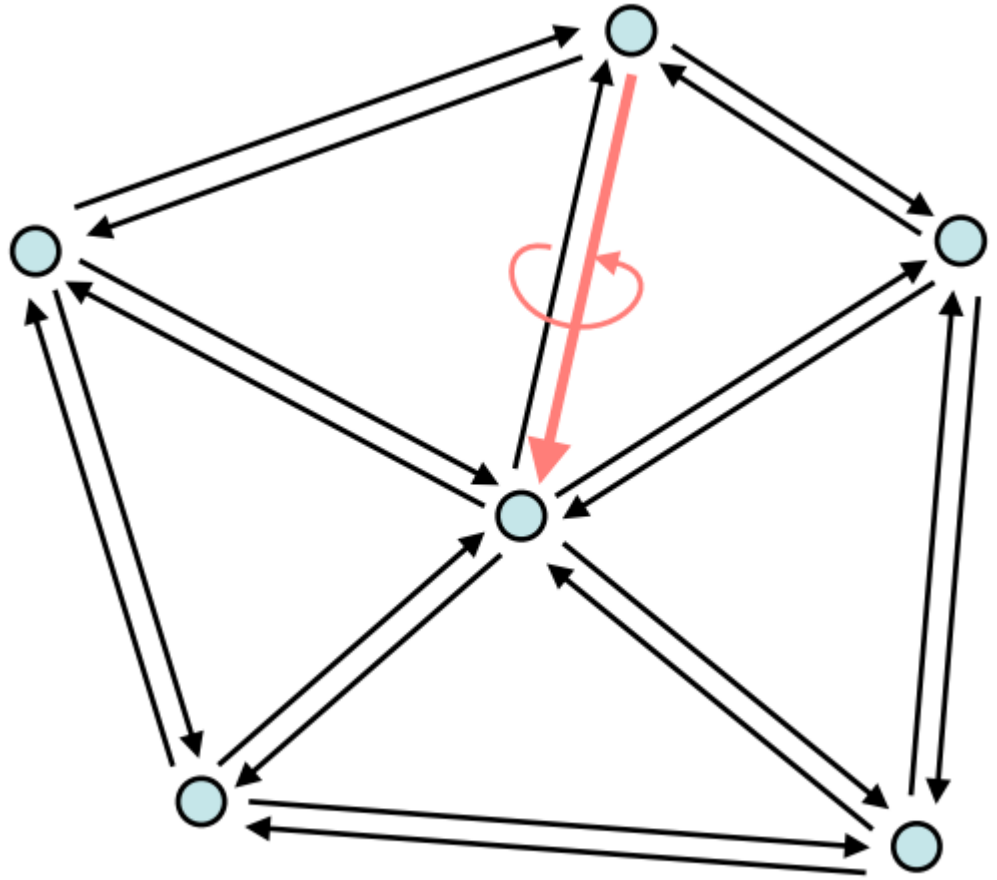
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



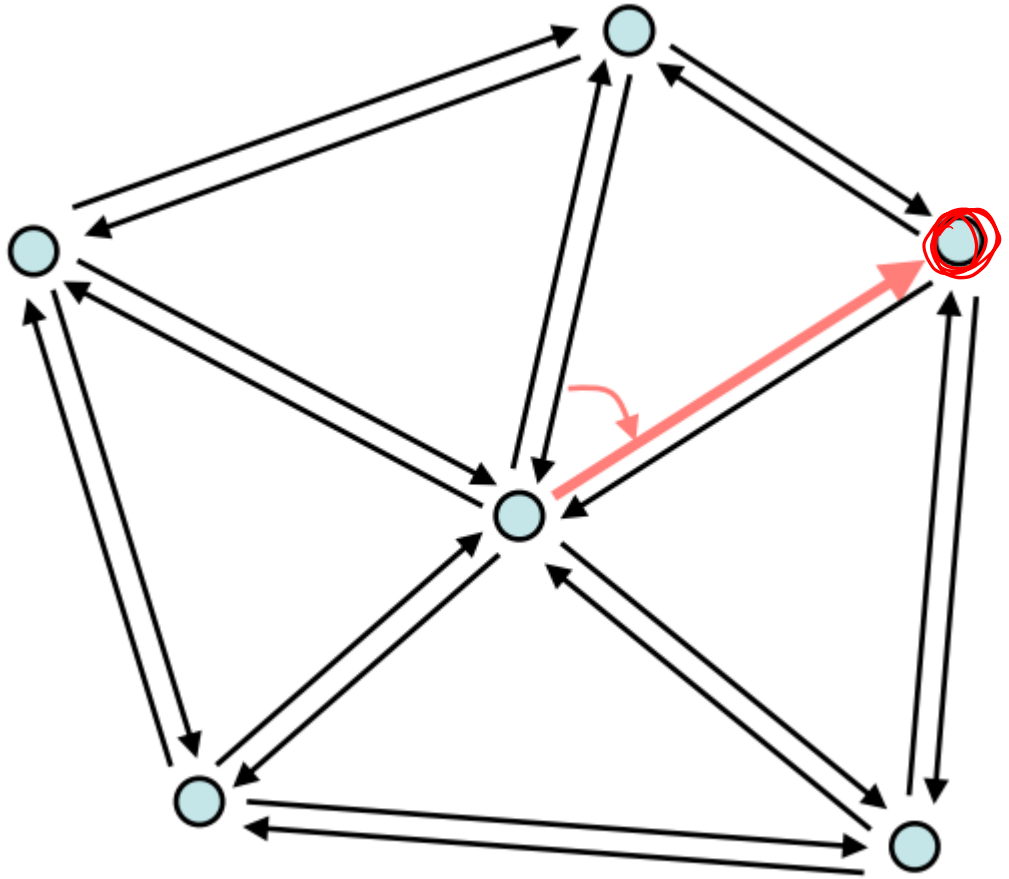
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



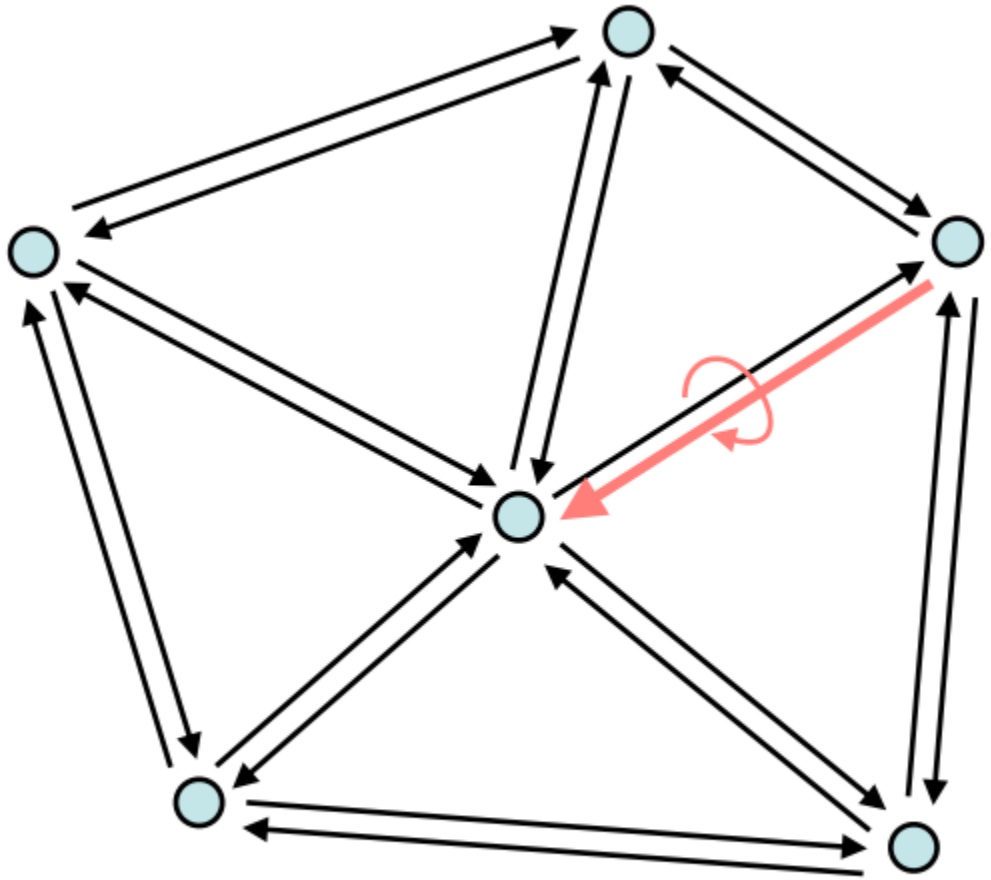
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite

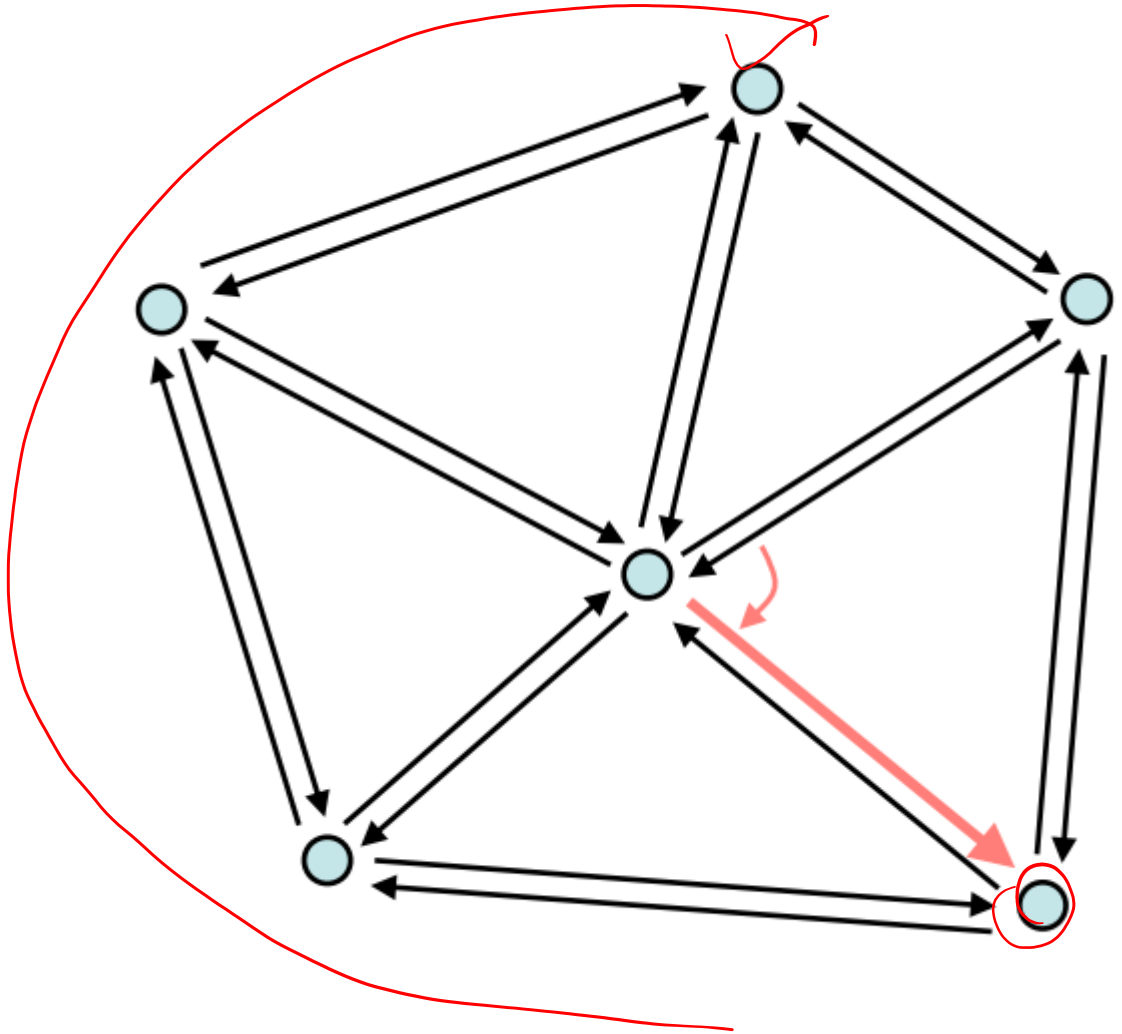


One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite

6.Next

7. ...

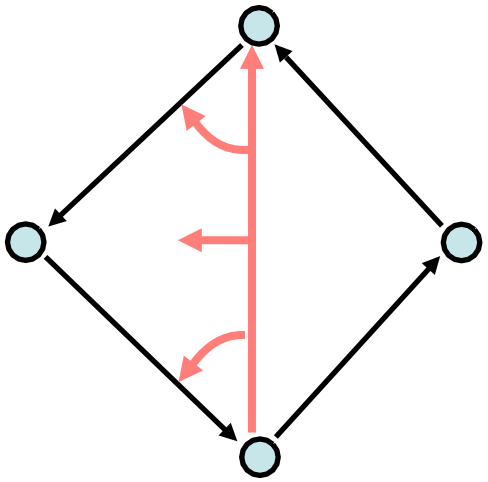


Halfedge Data Structure

Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
HalfedgeRef	prev
HalfedgeRef	opposite



Advantages:

- simple and efficient traversals of vertex neighborhoods
- can be applied to any polygonal mesh

Where to See Them in the Wild….

Computational Geometry Libraries

- CGAL
- OpenMesh

File Formats

- VTK (Visualization Toolkit)
- OBJ

More advanced formats as well…

- gltf for streaming (graphics file format…)
- cgns (computational fluid dynamics)
- …lots of discipline specific formats