CS 532: 3D Computer Vision
12th Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

Lecture Outline

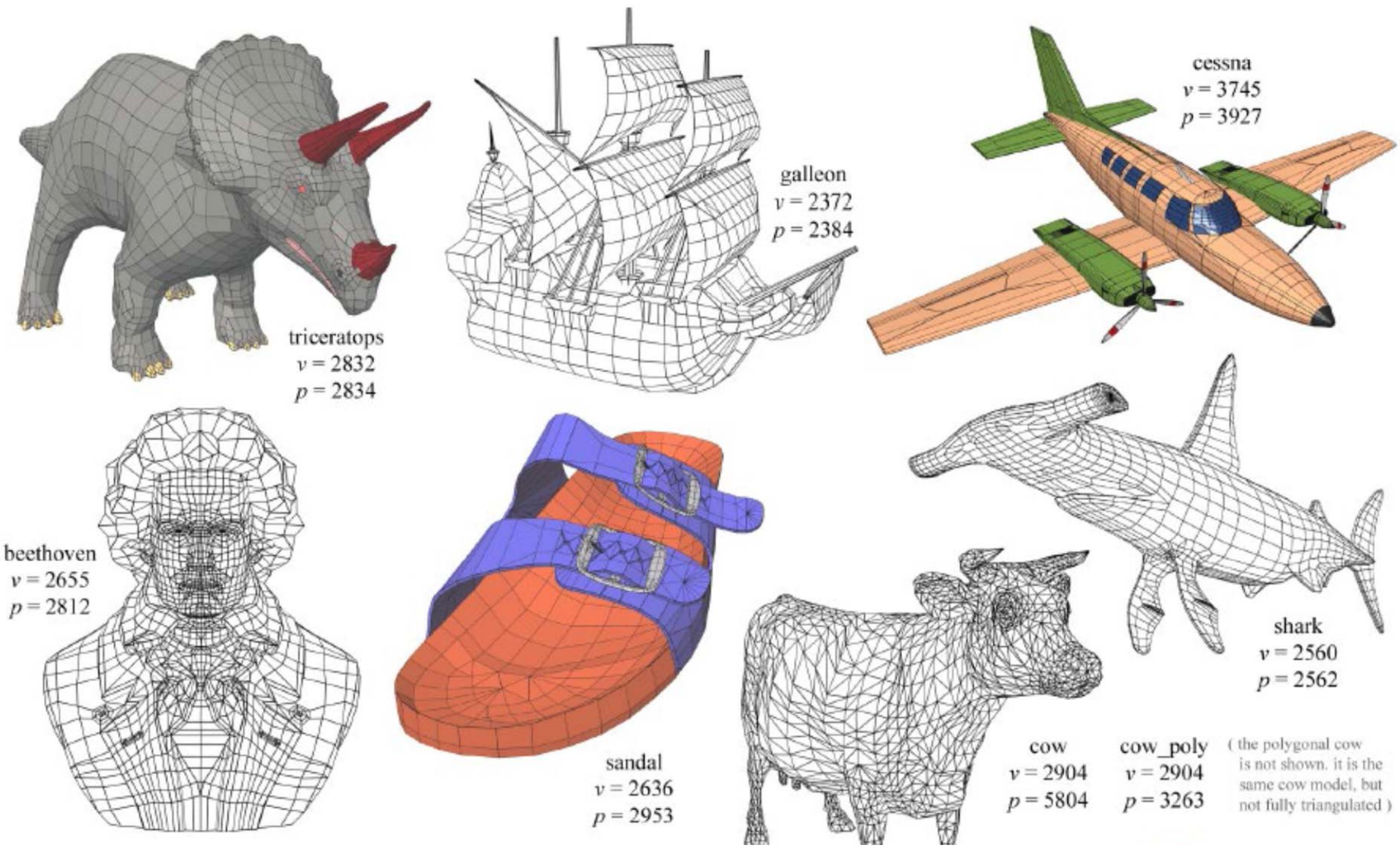
- Meshes

- Slides by:
  - S. Rusinkiewicz, T. Liu and V. Kim (Princeton University)

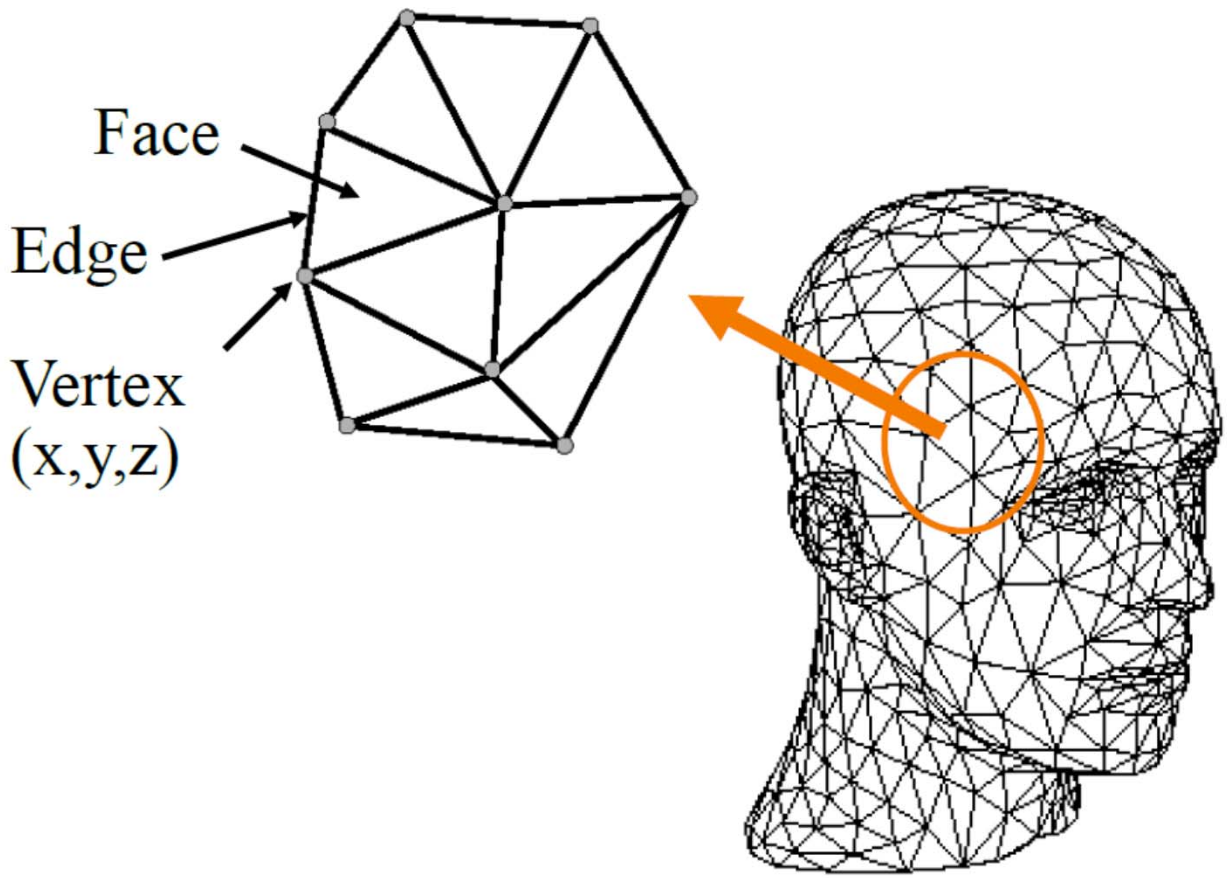
- David M. Mount, CMSC 754: Computational Geometry lecture notes, Department of Computer Science, University of Maryland, Spring 2012
  - Lecture 22

3D Polygonal Mesh

- Set of polygons representing a 2D surface embedded in 3D

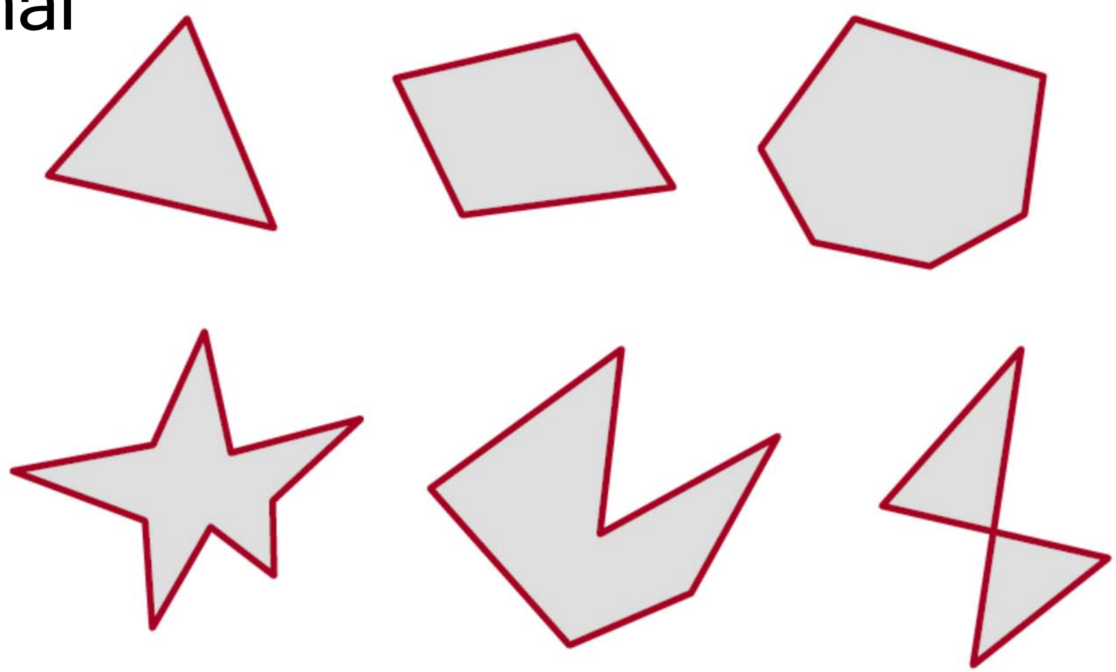


3D Polygonal Mesh



3D Polygon

- Region "inside" a sequence of coplanar points
- Points in counter-clockwise order
  - Define normal

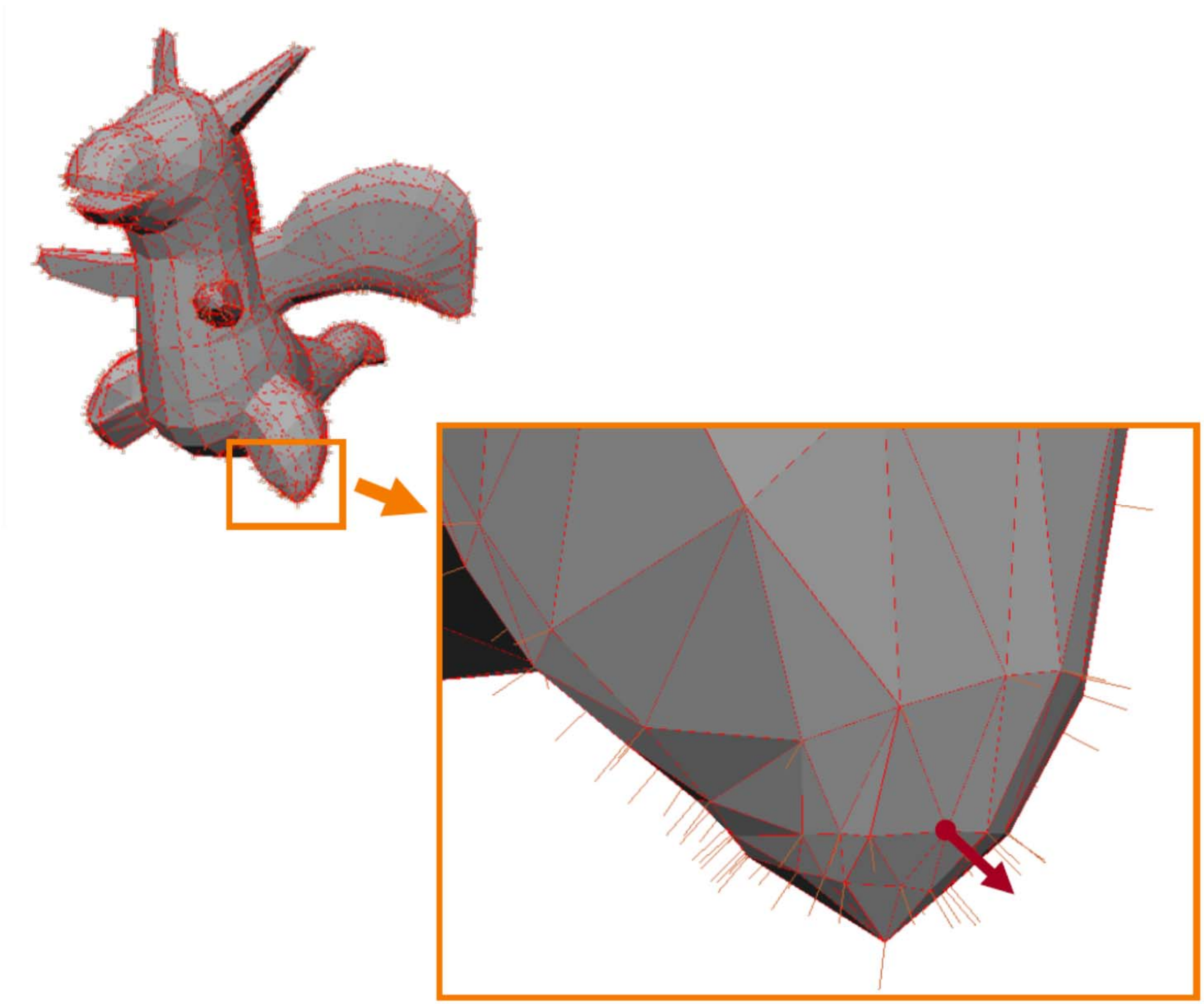


3D Polygonal Meshes

Why are they of interest?

- Simple, common representation
- Rendering with hardware support
- Output of many acquisition tools
- Input to many simulation/analysis tools

Surface Normals



Curvature

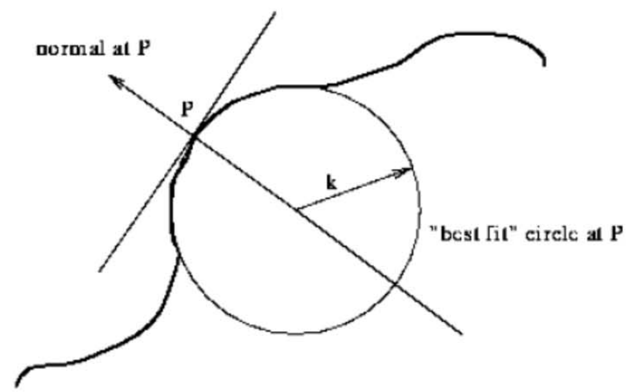
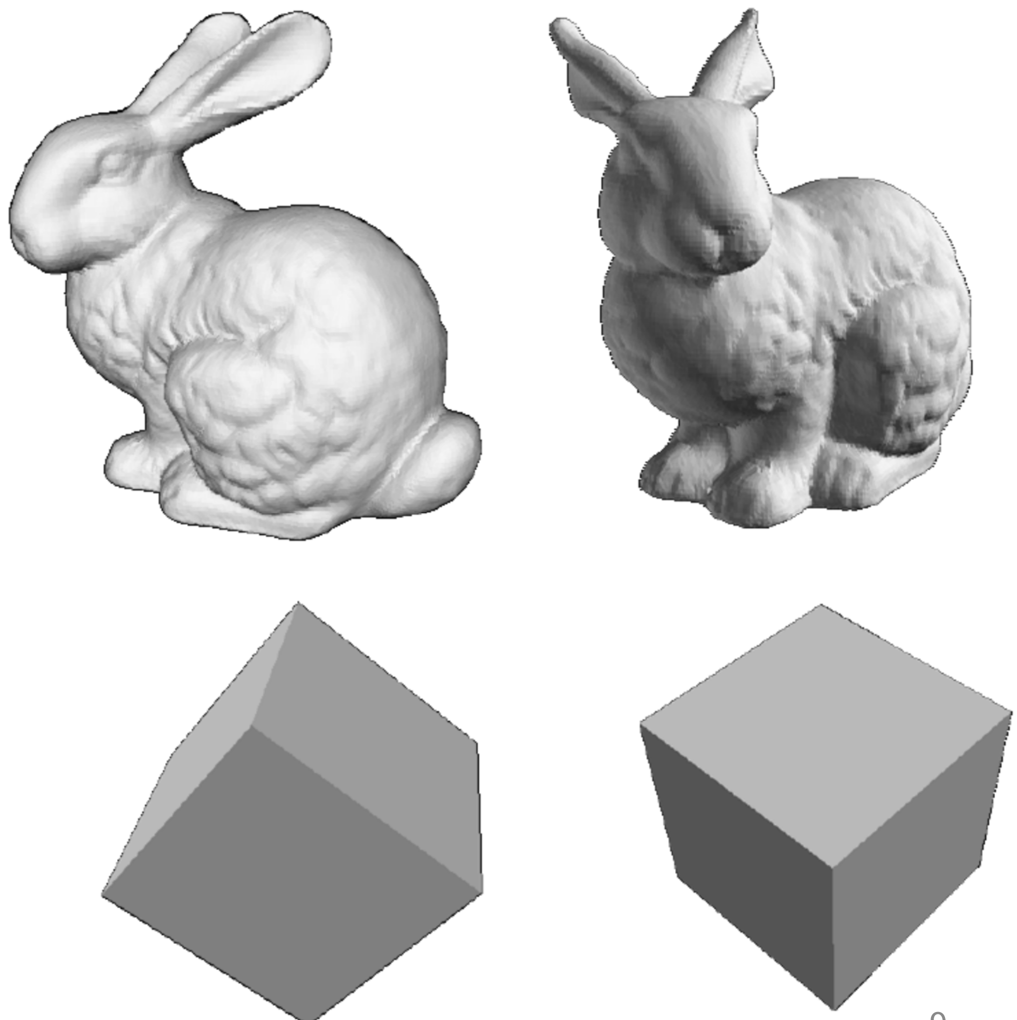


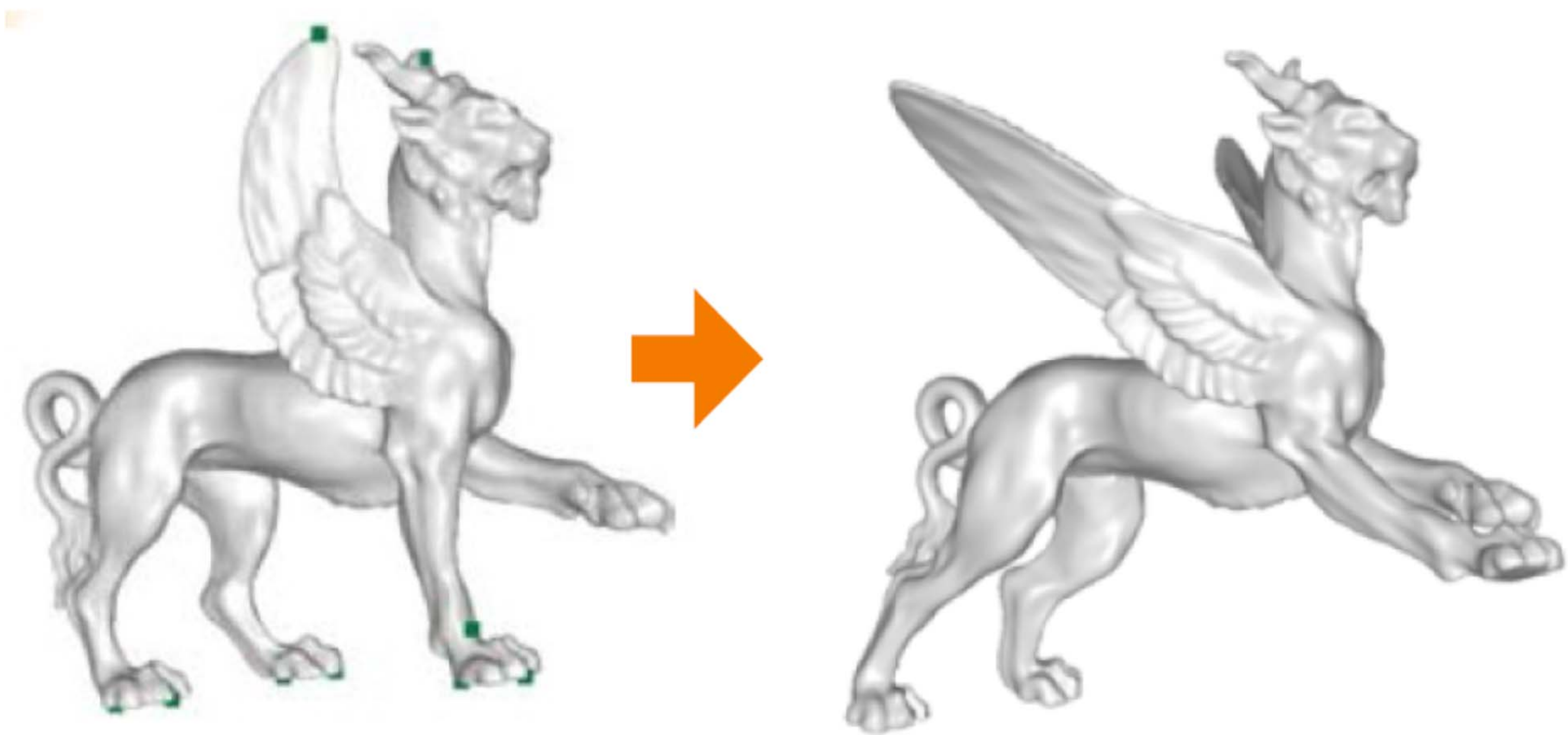
Figure 32: curvature of curve at P is $1/k$

Rigid Transformations

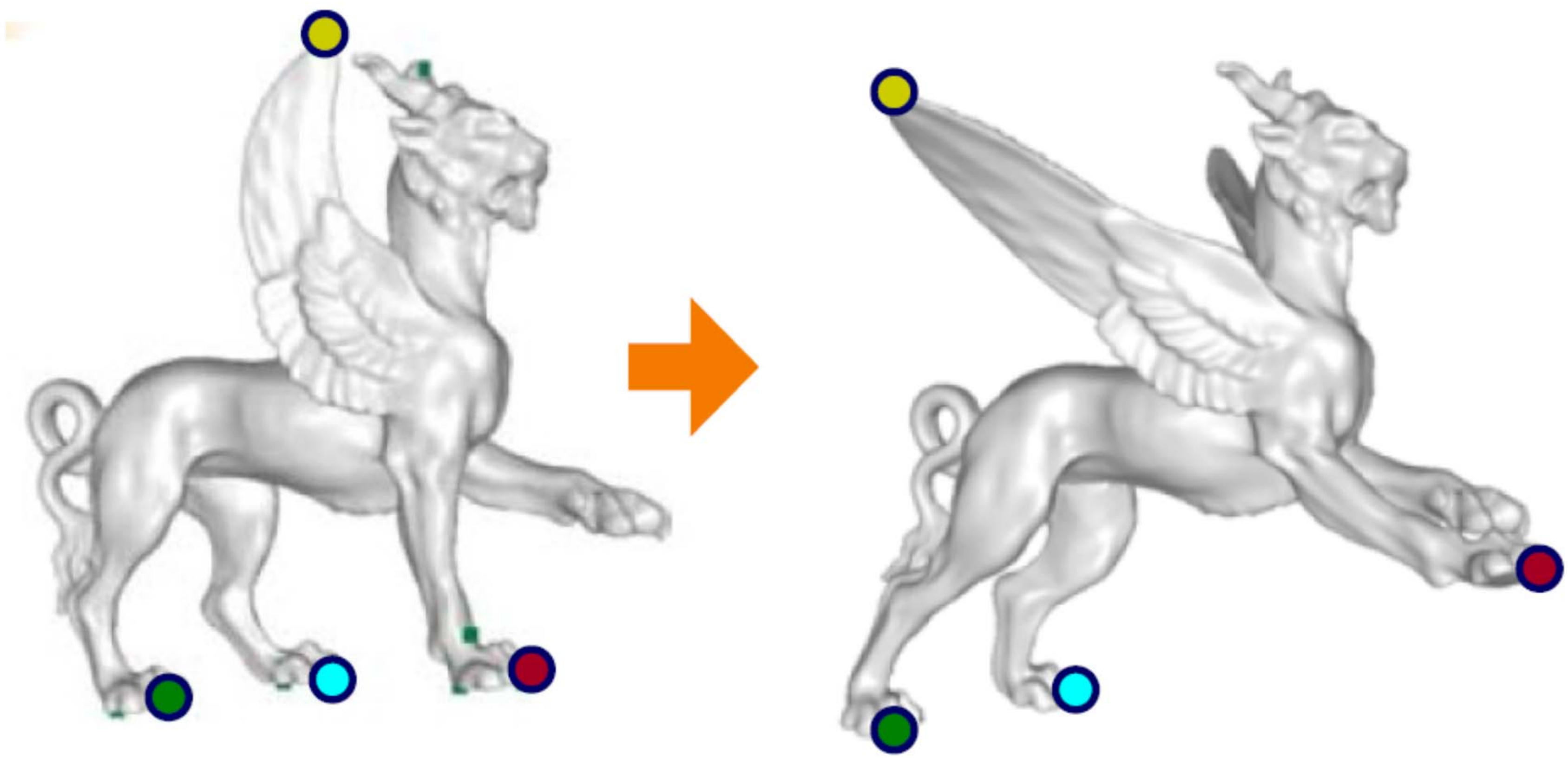
- Compare with implicit representations
  - level sets



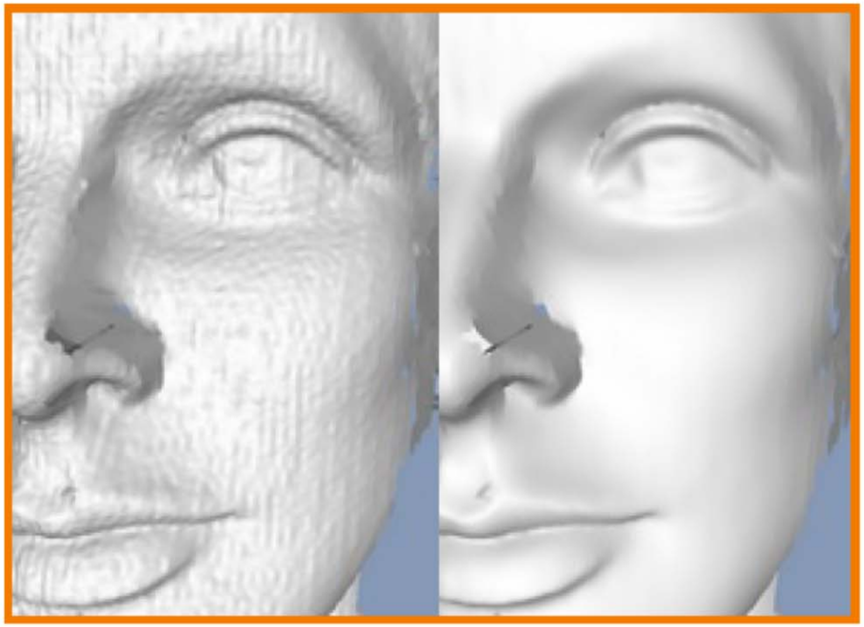
Deformations



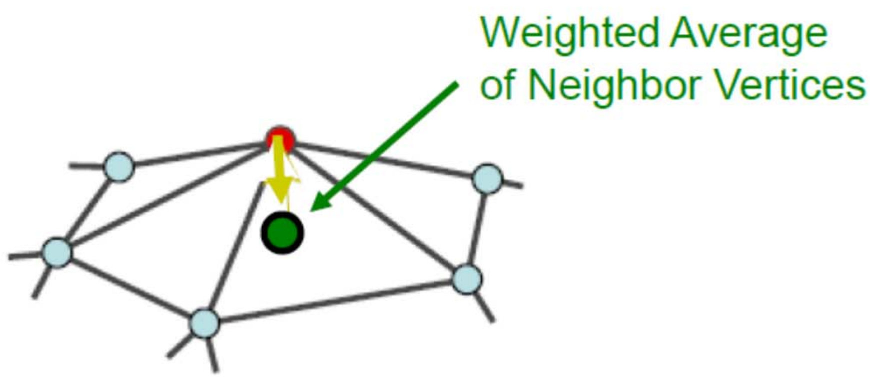
Deformations



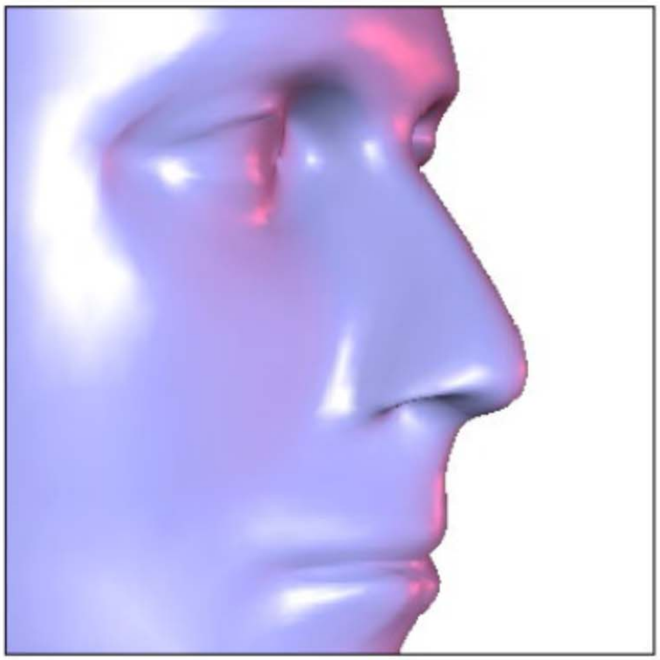
Smoothing

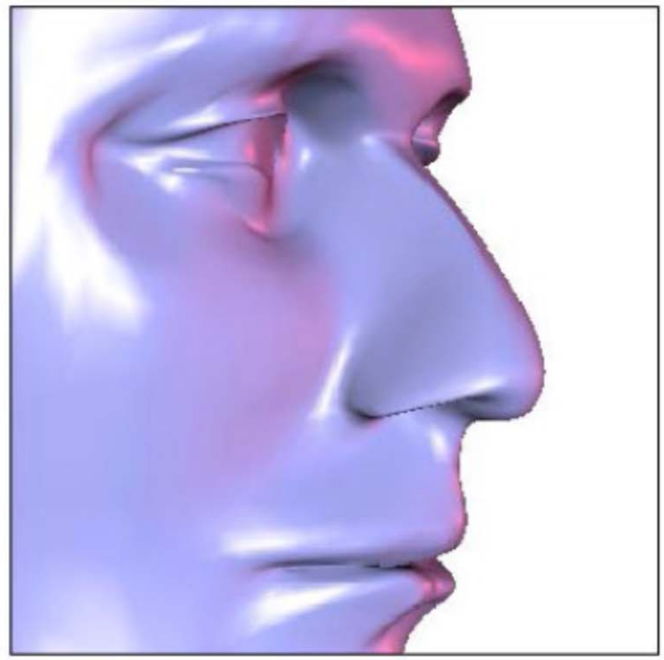


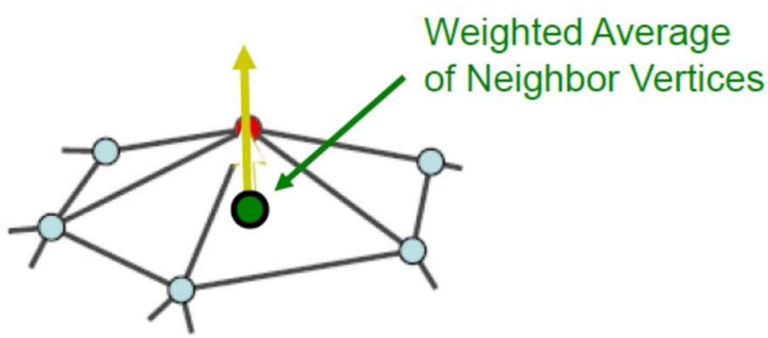
Thouis "Ray" Jones



Sharpen





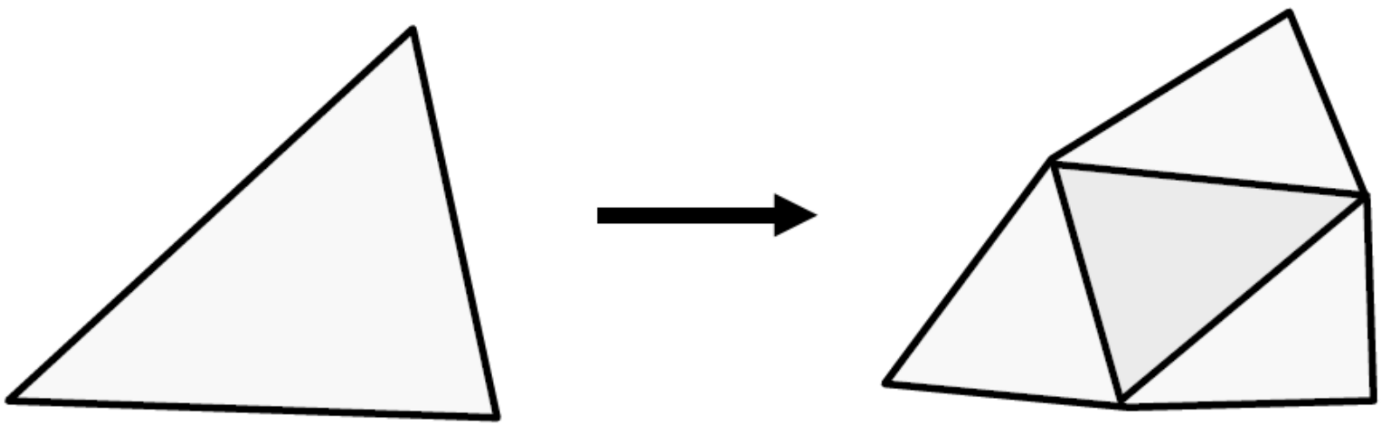


Low-level Operations

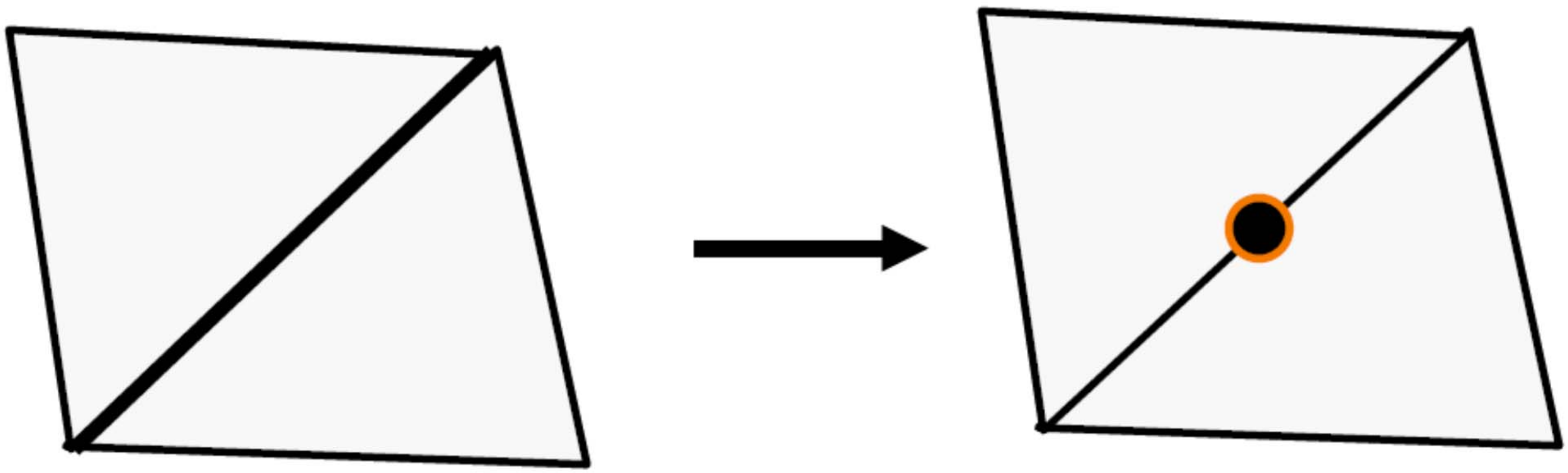
- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

Subdivide Face

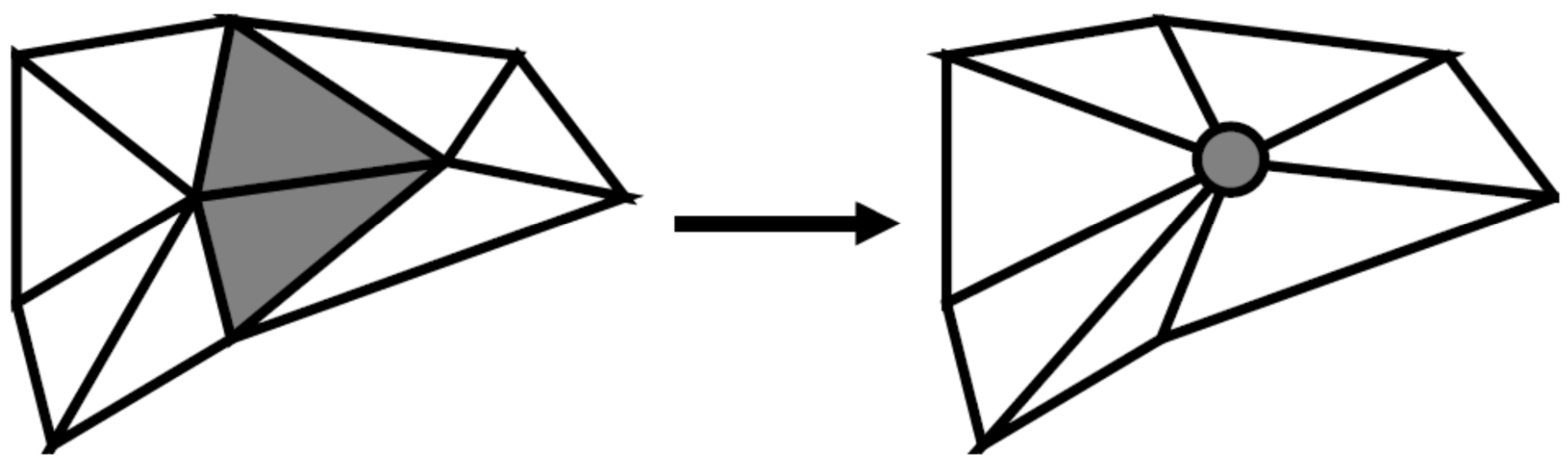
- How should we split current triangle?



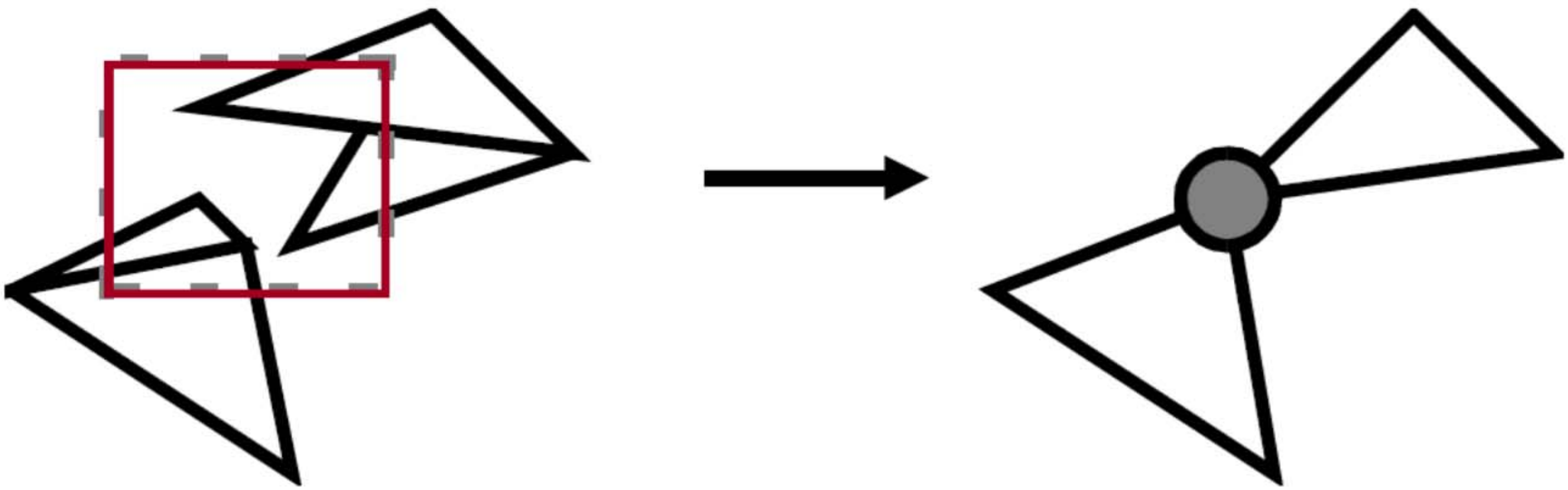
Subdivide Edge



Collapse Edge



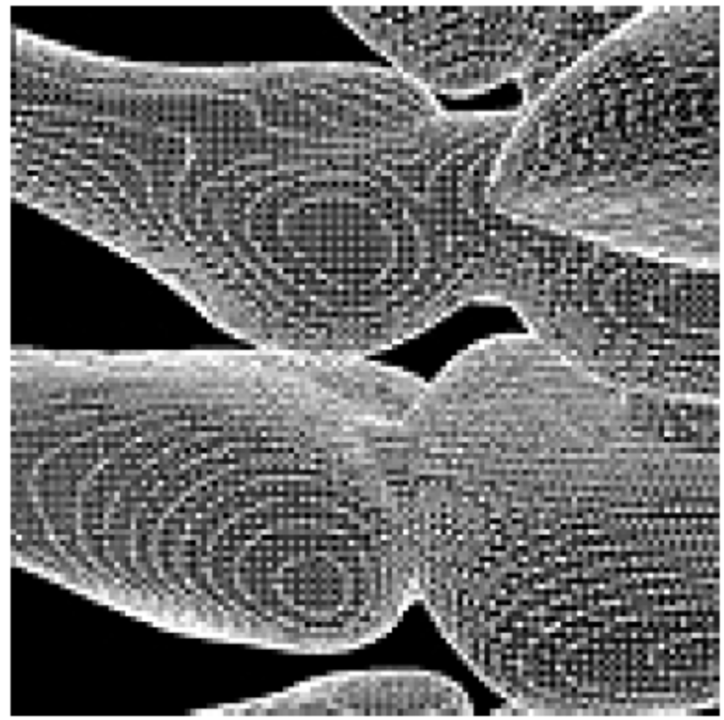
Merge Vertices

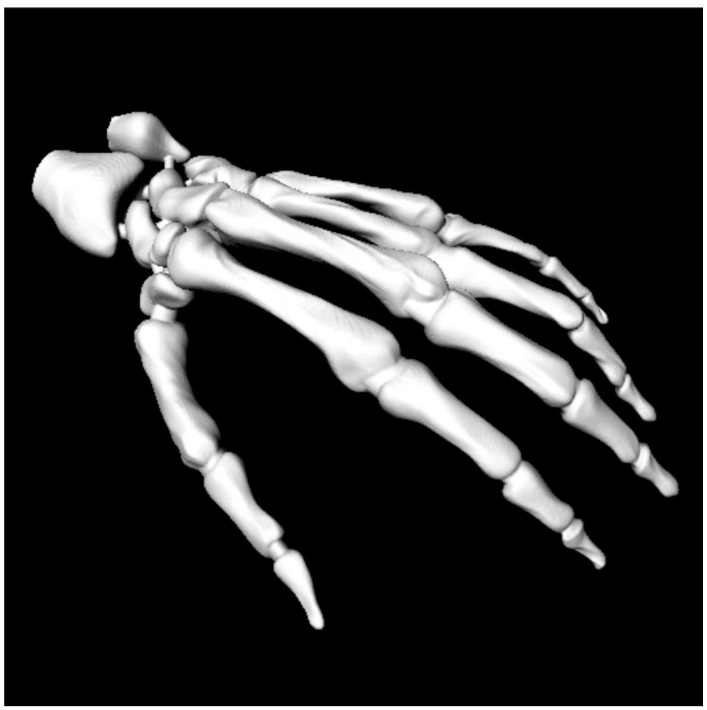


Polygonal Mesh Representation

Important properties of mesh representation

- Efficient traversal of topology
- Efficient use of memory
- Efficient updates



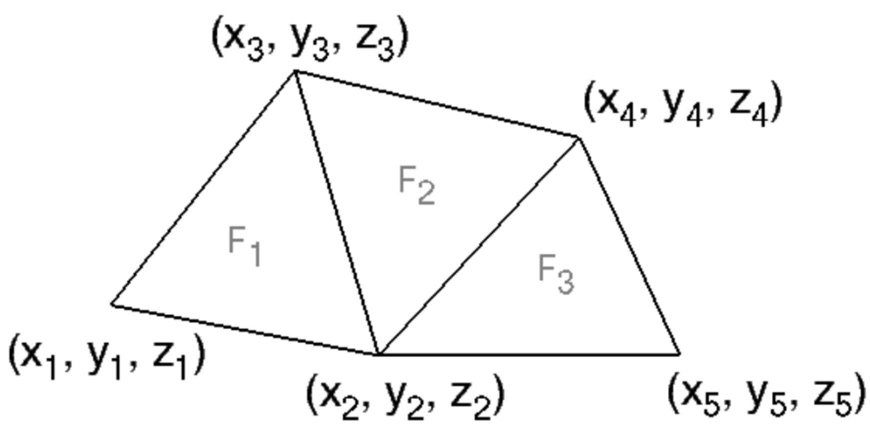


Possible Data Structures

- List of independent faces
- Vertex and face tables
- Adjacency lists
- Winged edge
- Half edge
- etc.

Independent Faces

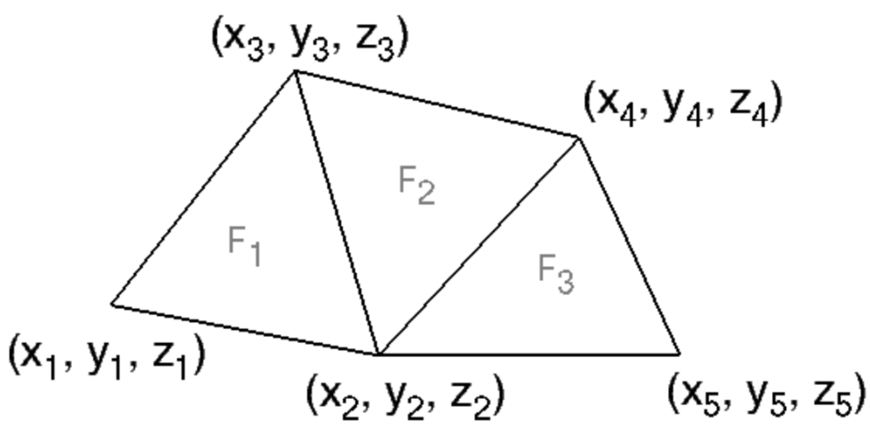
- A.k.a triangle soup
- Each face lists vertex coordinates
  - Redundant vertices
  - No adjacency information



FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

- Each face lists vertex references
  - Shared vertices
  - Still no adjacency information

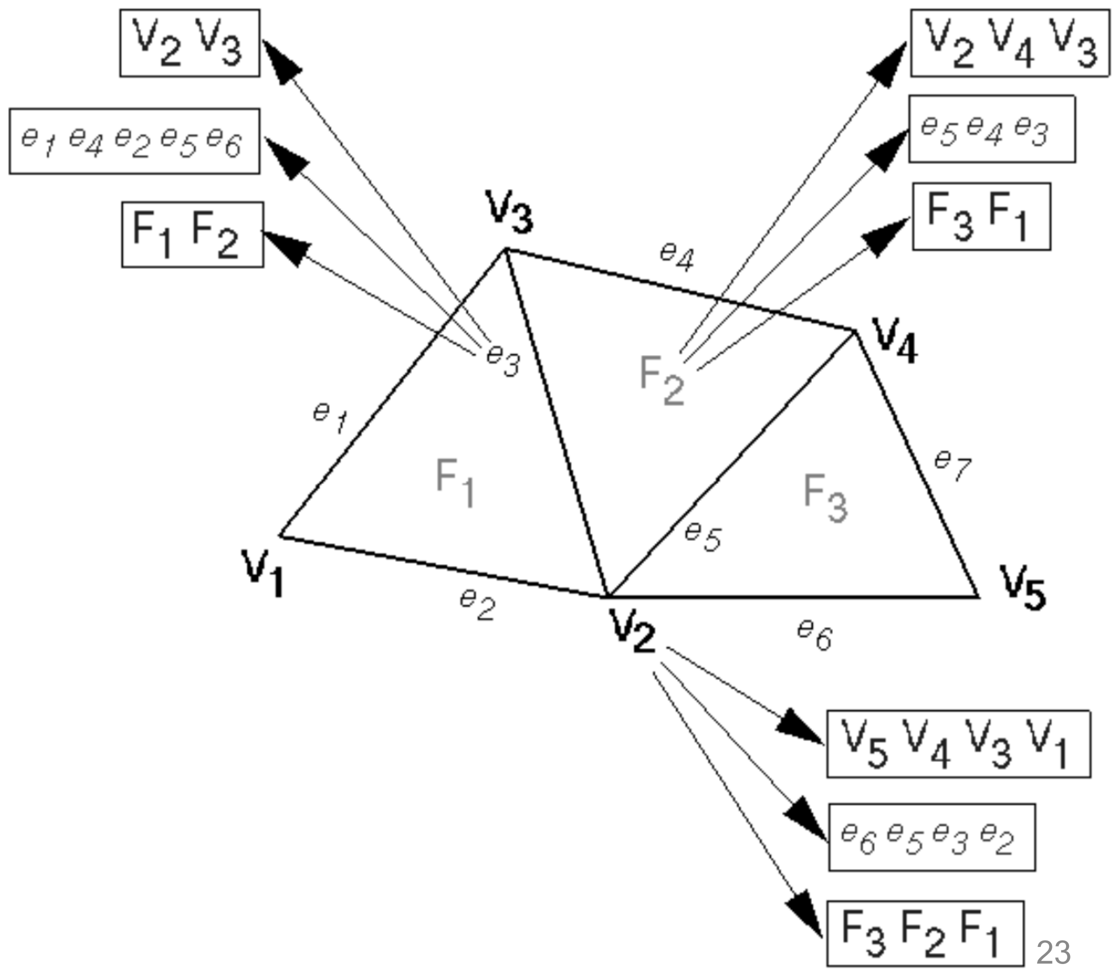


VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

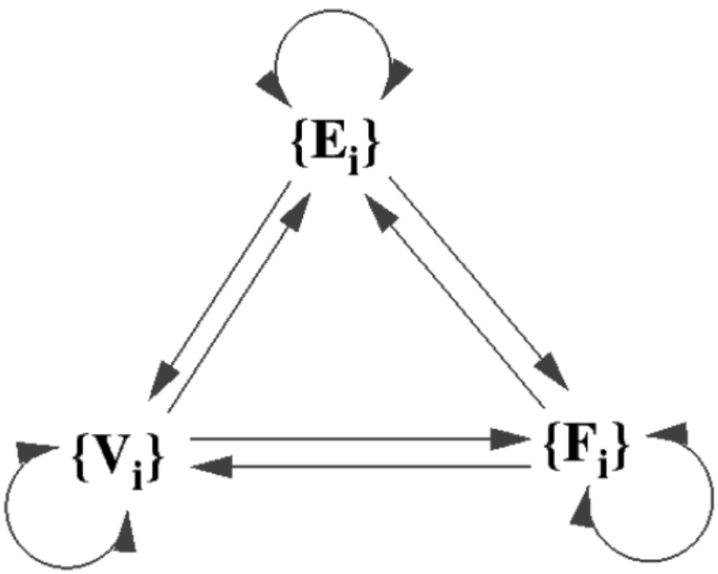
Adjacency Lists

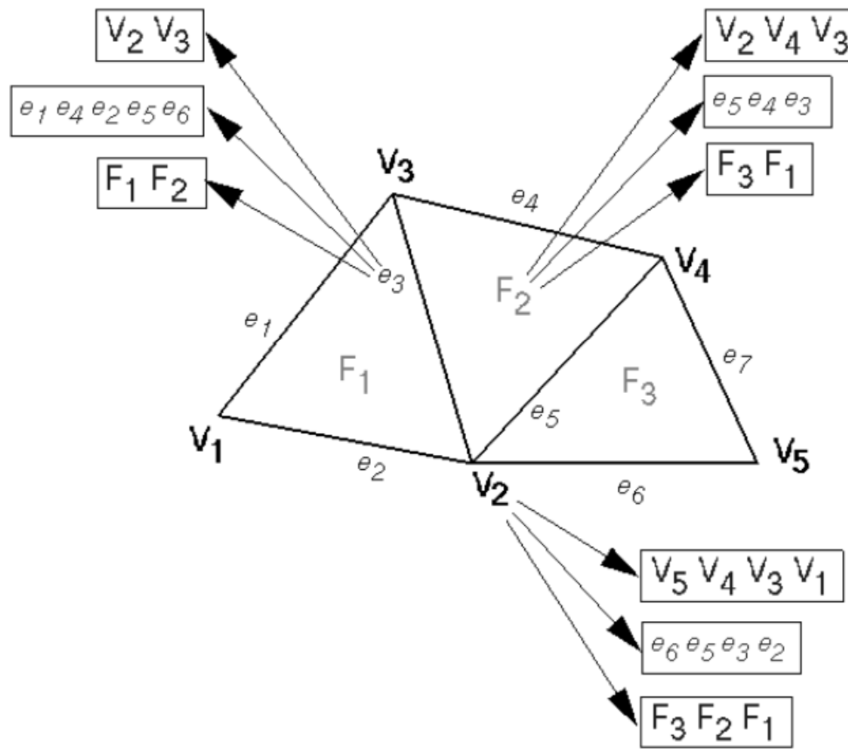
- Store all vertex, edge and face adjacencies
  - Efficient adjacency traversal
  - Extra storage requirements



Partial Adjacency Lists

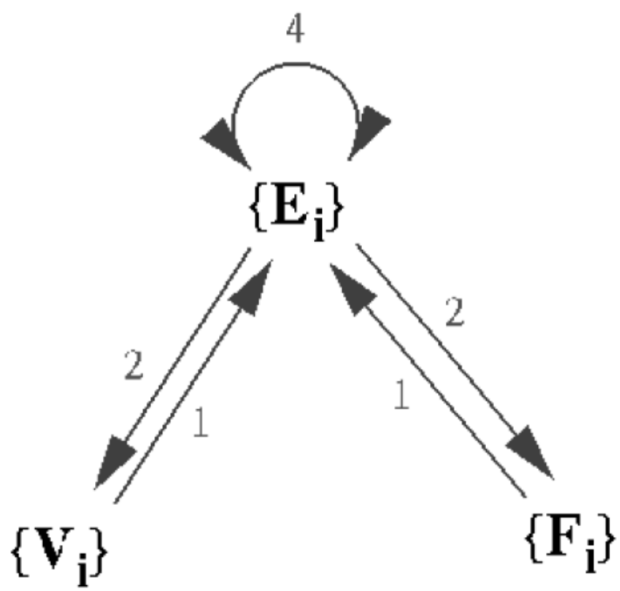
- Can we store only some adjacency relationships and derive others?

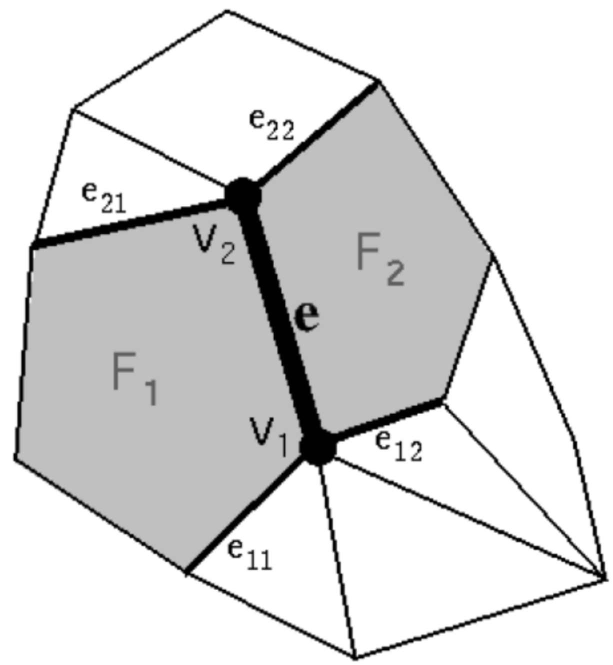




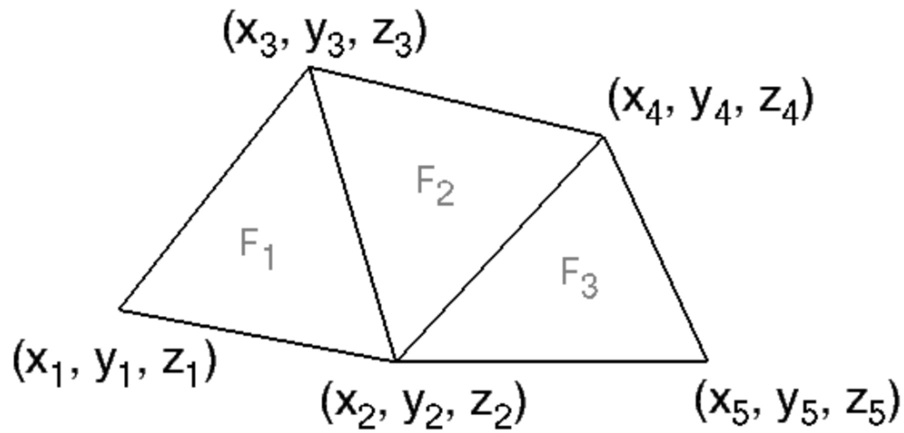
Winged Edge

- Adjacency encoded in edges
  - All adjacencies in O(1) time
- Little extra storage (fixed records)
- Arbitrary polygons





Winged Edge



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

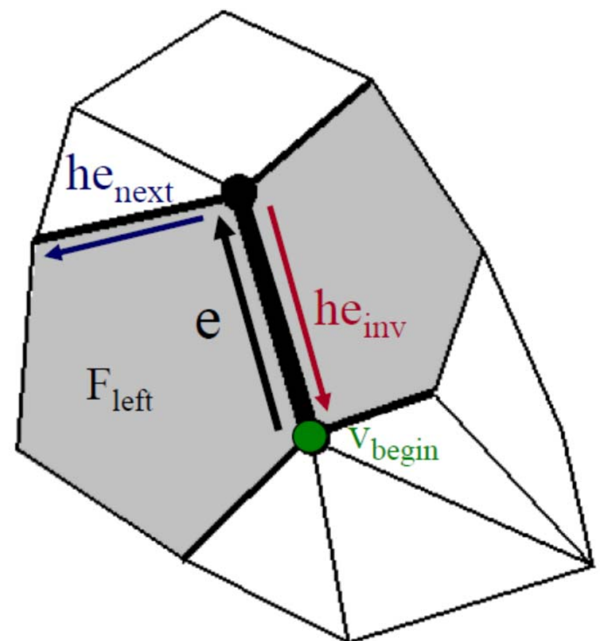
					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	$V3$	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

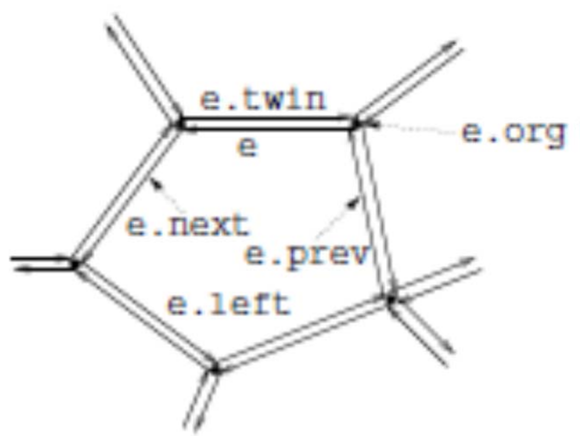
Half Edge

- Adjacency encoded in edges
  - All adjacencies in O(1) time
  - Little extra storage (fixed records)
  - Arbitrary polygons
- Similar to winged-edge, except adjacency encoded in half-edges



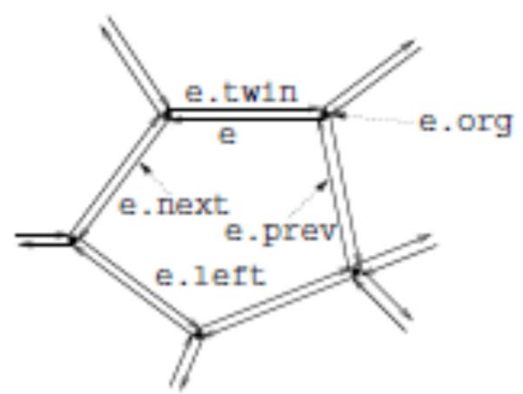
Half Edge

- Each undirected edge represented by two *directed* half edges
  - Unambiguously defines left and right
- Assume that there are no holes in faces



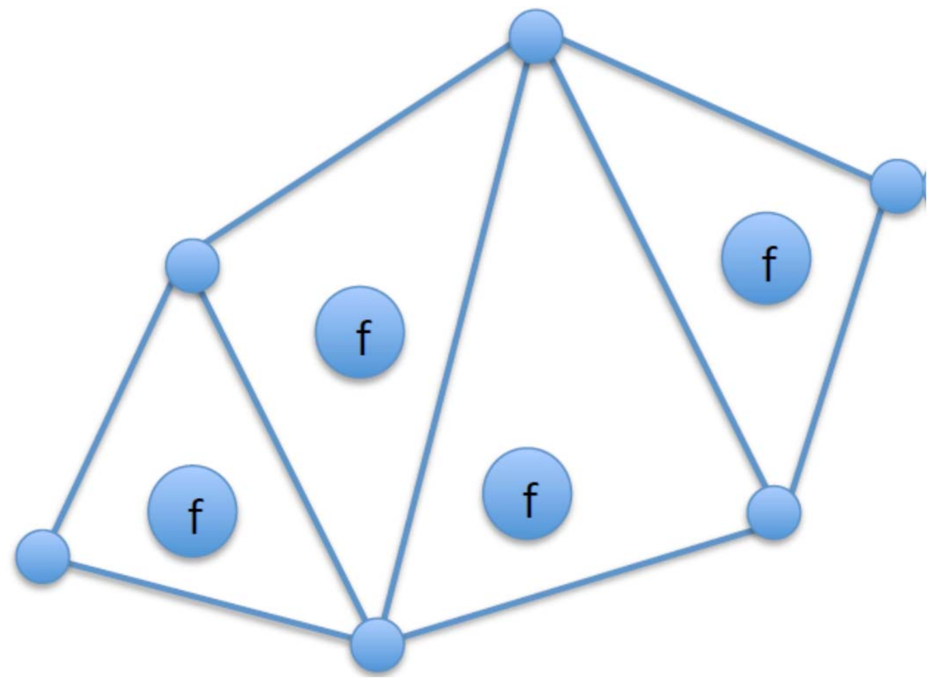
Half Edge

- Each vertex stores:
  - its coordinates
  - a pointer v.inc_edge to any directed edge that has vertex as its origin
- Each directed edge is associated with:
  - a pointer to the oppositely directed edge, called its twin
  - an origin and destination vertex
  - two faces, one to its left and one to its right.
- We only store:
  - a pointer to the origin vertex e.org (e.dest can be accessed as e.twin.org)
  - a pointer to the face to the left of the edge e.left (we can access the face to the right from the twin edge)
  - pointers to the next and previous directed edges in counterclockwise order about the incident face, e.next and e.prev, respectively
- Each face f stores a pointer to a single edge for which this face is the incident face, f.inc_edge



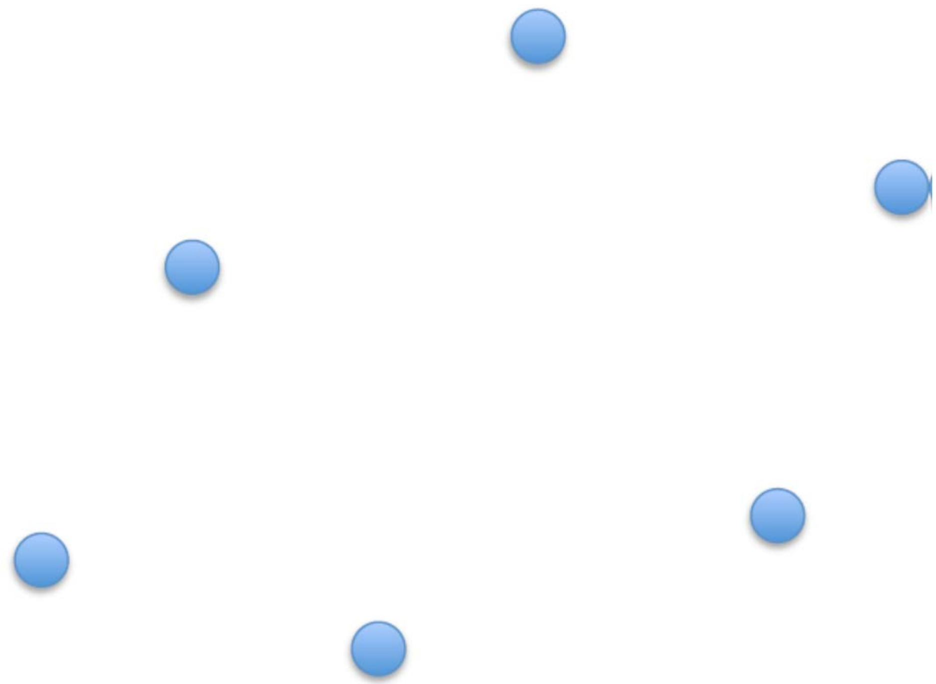
How to Load a Shape

- From file with vertices and triangles



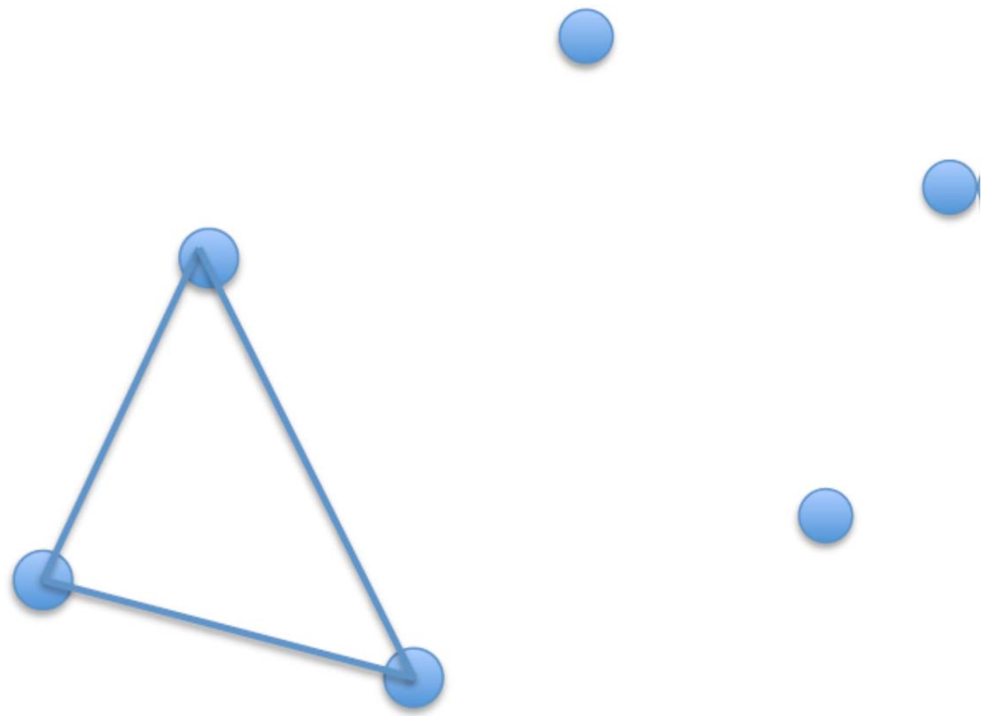
How to Load a Shape

- Add vertex coordinates to list



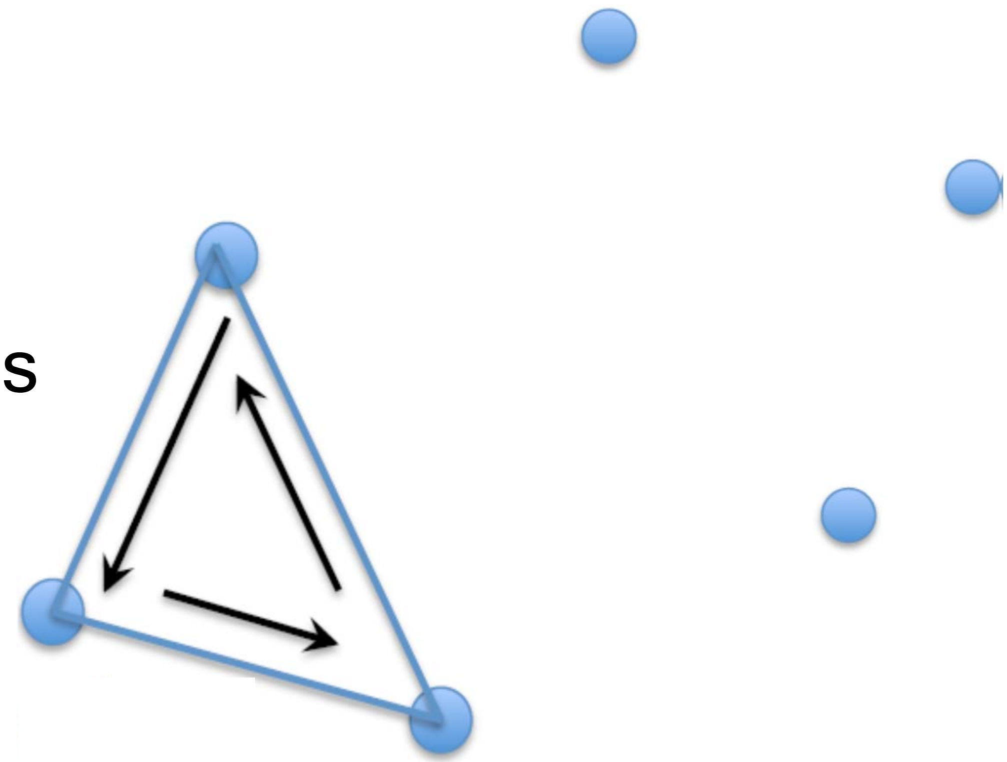
How to Load a Shape

- Add vertex coordinates to list
- Add half-edges with faces



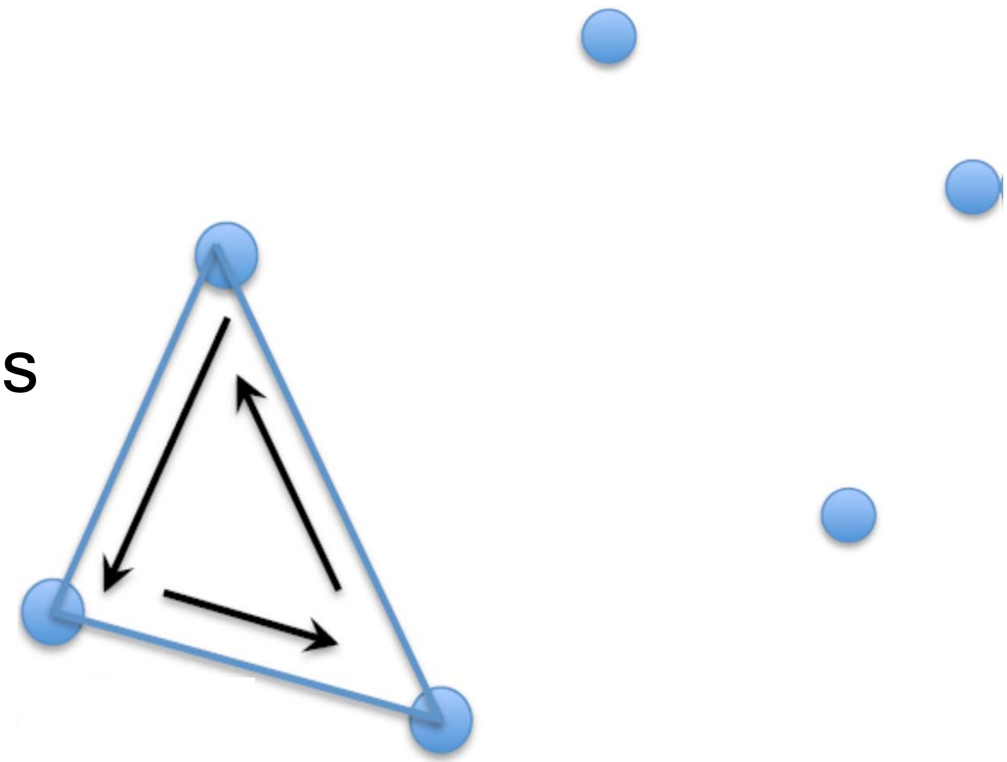
How to Load a Shape

- Add vertex coordinates to list
- Add half-edges with faces
  - Inner half-edges are sufficient



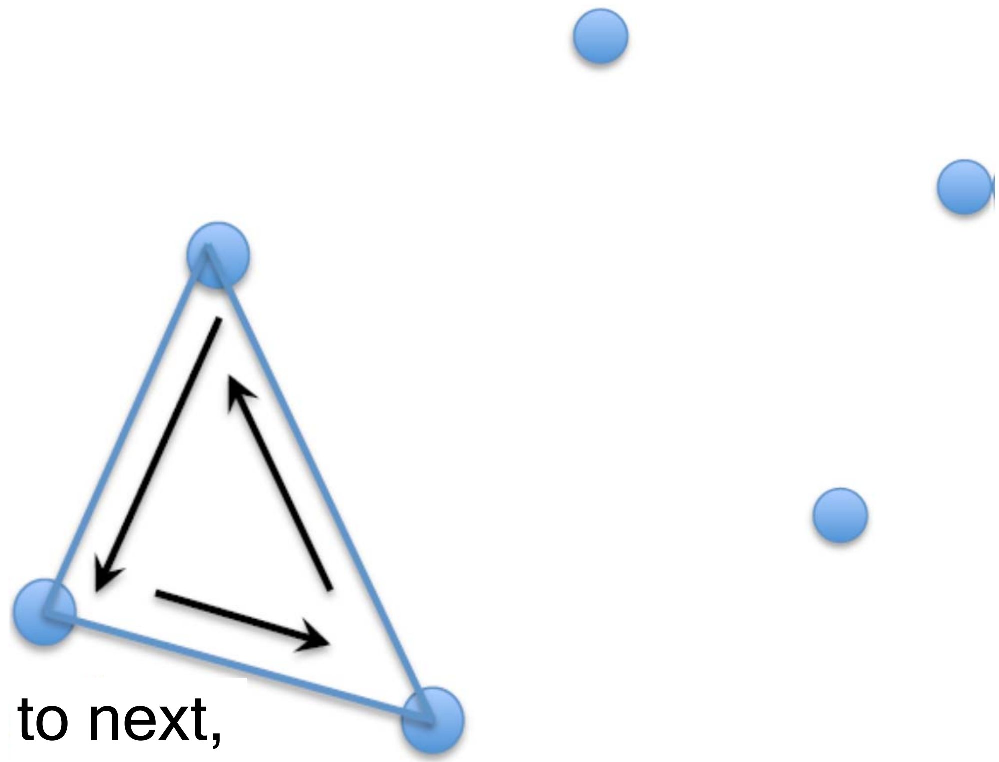
How to Load a Shape

- Add vertex coordinates to list
- Add half-edges with faces
  - Inner half-edges are sufficient
  - Update vertex pointers to half-edges



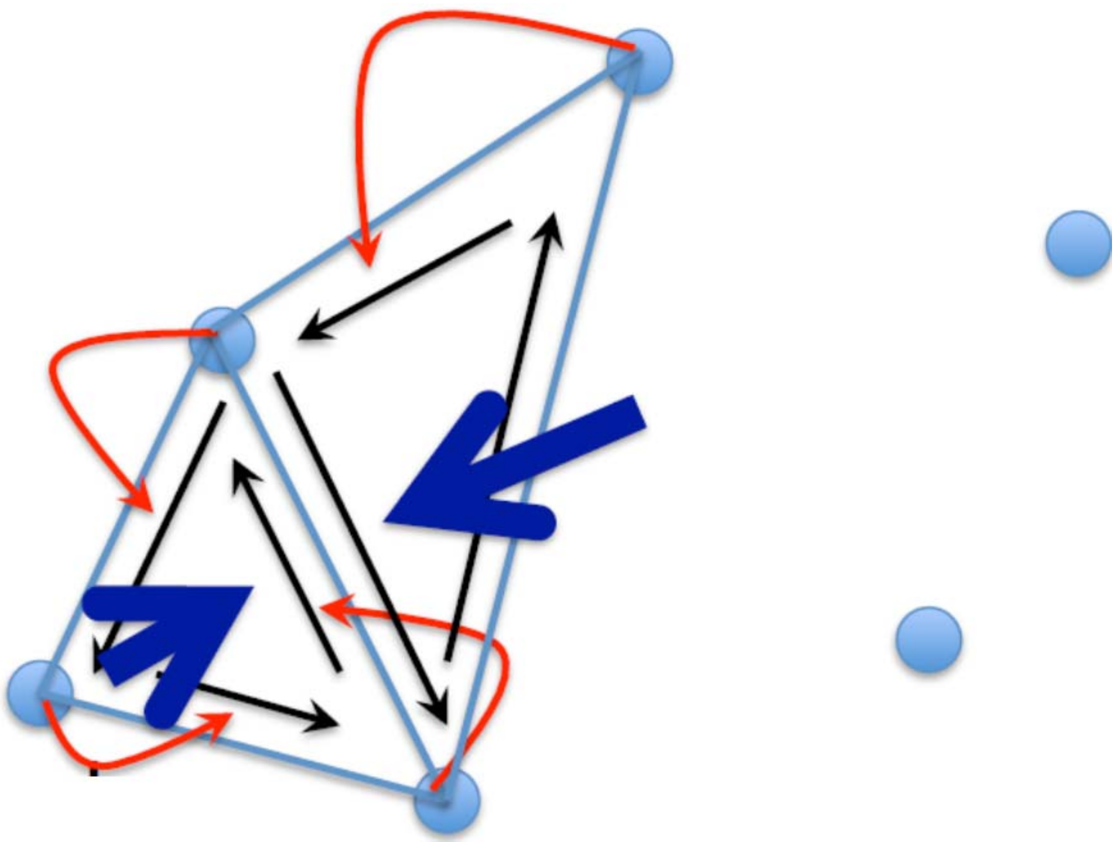
How to Load a Shape

- Add vertex coordinates to list
- Add half-edges with faces
  - Inner half-edges are sufficient
  - Update vertex pointers to half-edges
  - Half-edges: pointer to next, pointer to face
  - Faces: pointer to one of the inner half-edges



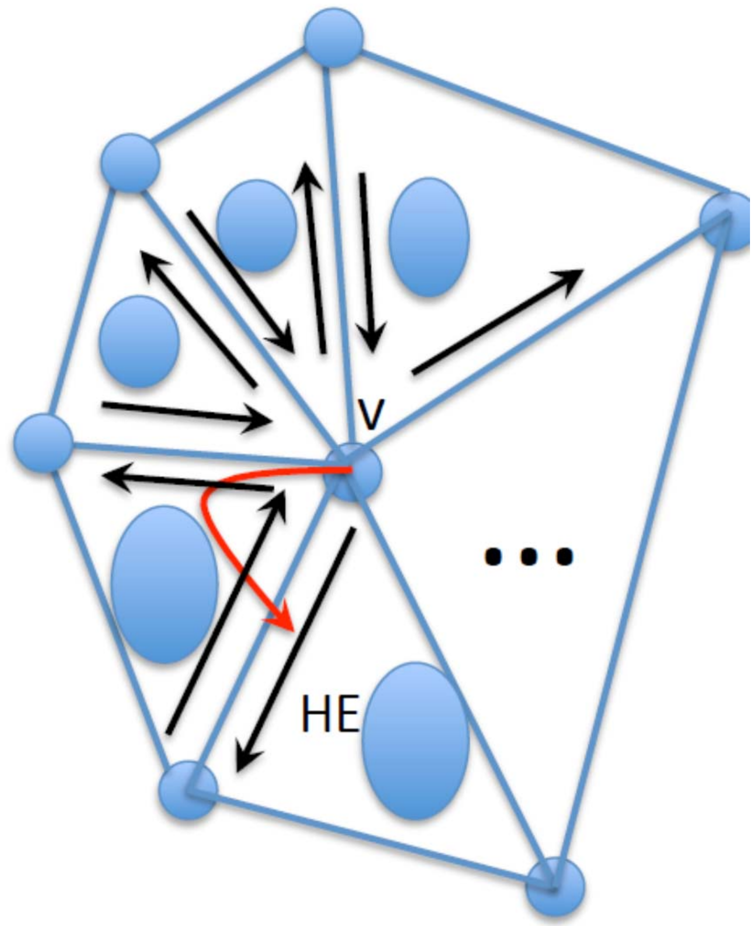
How to Load a Shape

- Continue adding incrementally



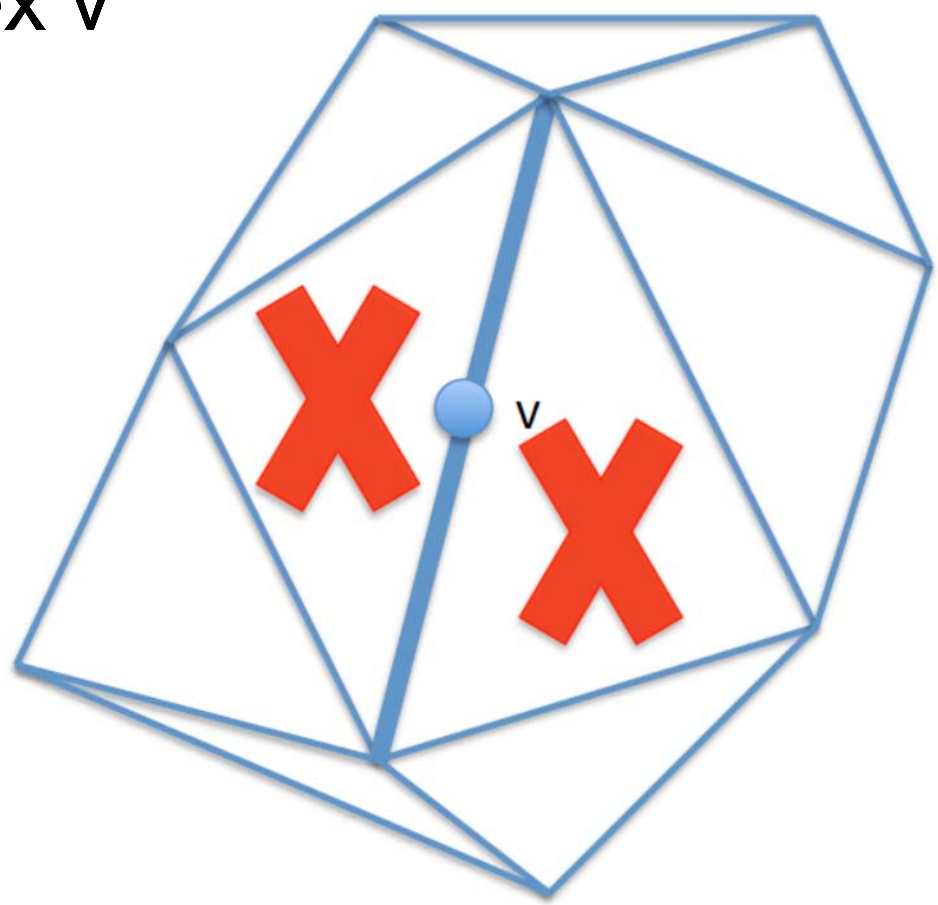
Finding Adjacent Faces

- Check all outgoing half edges
  - V points to a half edge HE
  - ADD_FACE(HE)
  - Iterate:
    - X=HE.twin
    - Y=X.next
    - ADD_FACE(Y)
    - HE:=Y



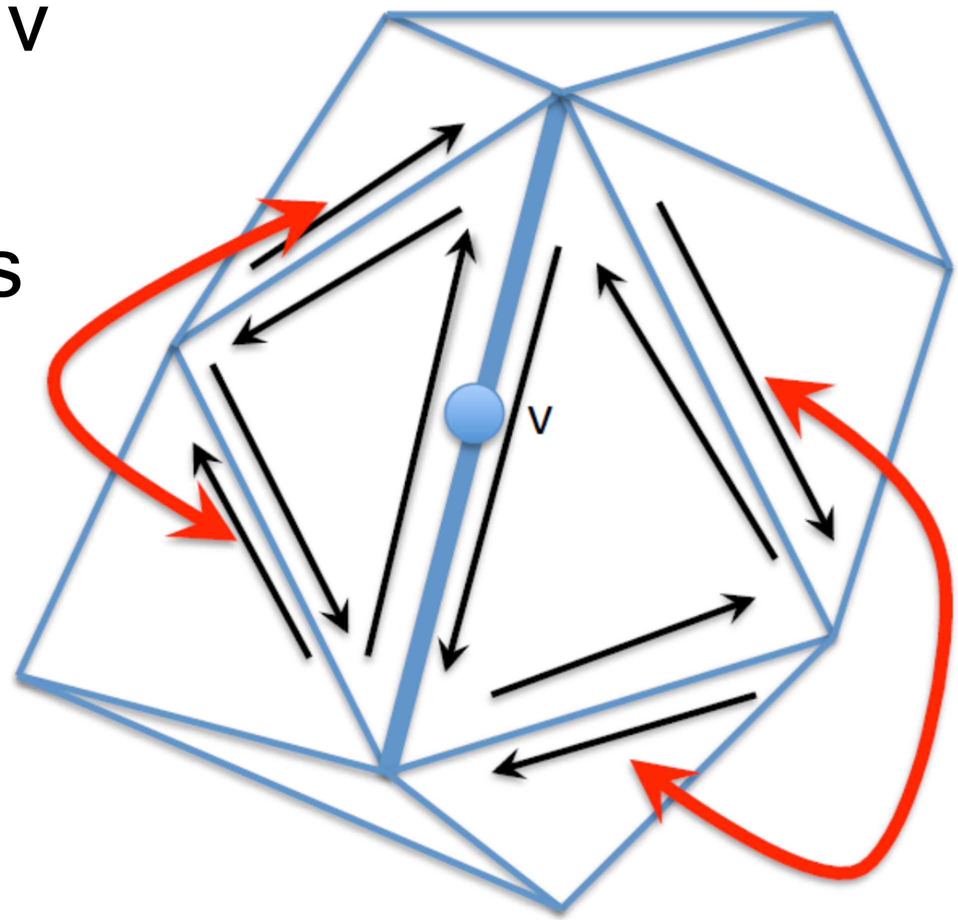
Collapsing an Edge

- Create a new vertex v
- Remove faces



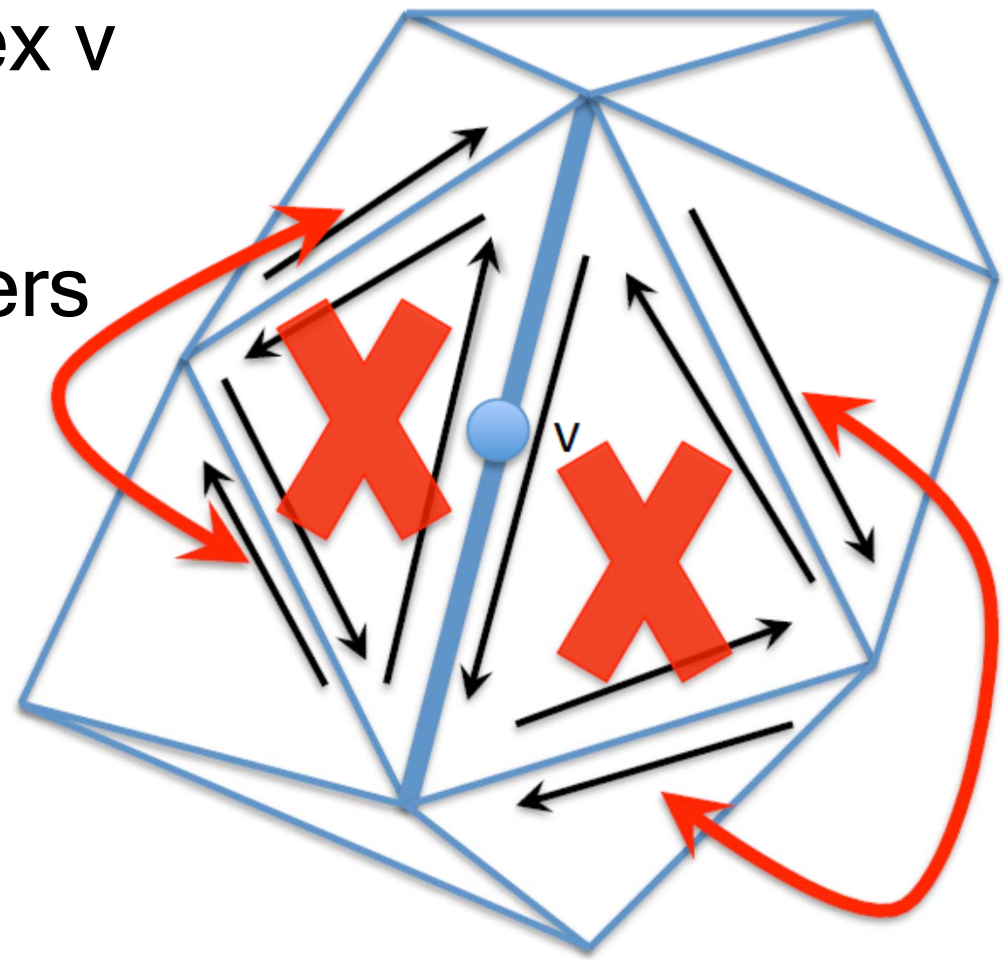
Collapsing an Edge

- Create a new vertex v
- Remove faces
- Change twin pointers



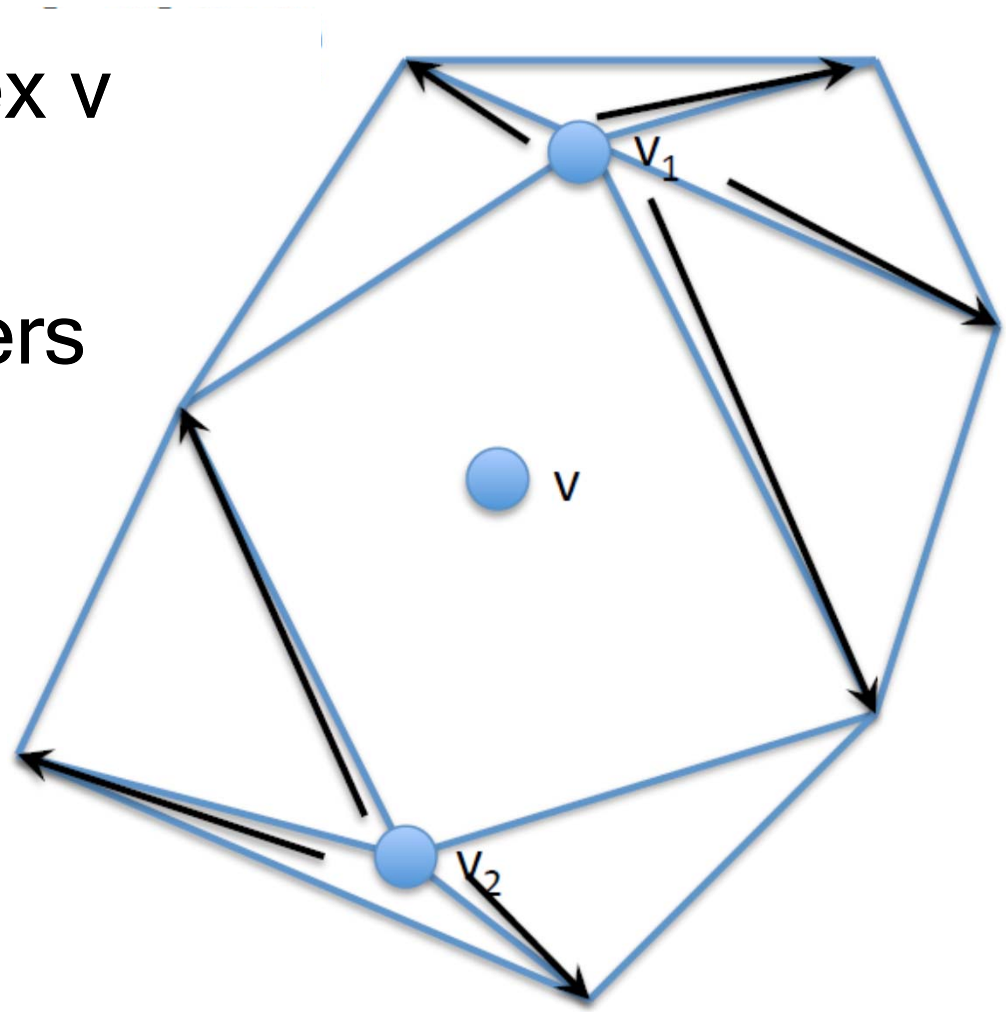
Collapsing an Edge

- Create a new vertex v
- Remove faces
- Change twin pointers
- Remove edges



Collapsing an Edge

- Create a new vertex v
- Remove faces
- Change twin pointers
- Remove edges
- Change pointers from half-edges to v_1 and v_2



Collapsing an Edge

- Create a new vertex v
- Remove faces
- Change twin pointers
- Remove edges
- Change pointers from half-edges to v_1 and v_2
- Remove v_1 and v_2
- Pick an outgoing edge for v