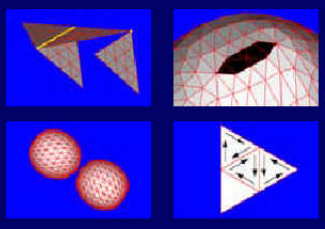
Mesh Data Structures

Polygonal Meshes

- Components
  - Connectivity (Vertices, Edges, Faces)
  - Geometry (Vertex Coordinates)
  - Properties (Normals, Colors, Texture Coordinates)
- Connectivity (combinatorial algorithms)
  - Boundary / Regular / Singular Edges and Vertices
  - Connected Components
  - Manifold / Non-manifold
  - Orientable / Oriented
  - Topology



Operations

Draw (® Compute normals)

Rotate

Cut (with plane, cut patch, etc)

Unfold patch

Find shortest paths between vertices

Operations

Visit all vertices / faces / edges

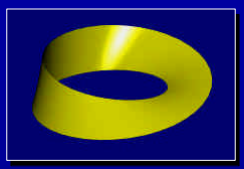
Find all vertices of a face (ordered)

Find all vertices / faces adjacent to a vertex

Face opposite face, across edge

Create / destroy vertex / face / edge

Objects

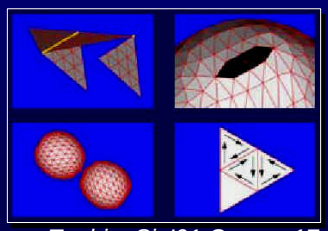


Triangles / polygons

Holes

Manifold

Orientable

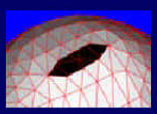


Taubin, Sig'01 Course 17

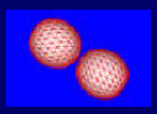
www.kleinbottle.com

Connectivity / Classification of Elements

- Edges
  - Boundary (1 incident face)
  - Regular (2 incident faces)
  - Singular (3 or more incident faces)



- Vertices
  - Regular / Singular
- Connected components
  - Connected Components of Dual Graph



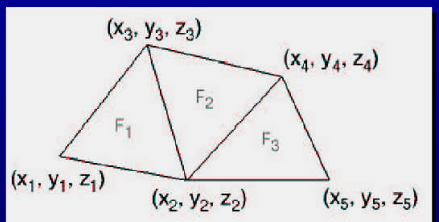
Manifold Meshes

- No singular edges
  - Boundary
    - Edge with 1 incident face
  - Regular
    - Edge with 2 incident faces
- No singular vertices
  - Boundary
    - dual graph of set of incident faces form a path
  - Regular
    - dual graph of set of incident faces form a cycle
- Data Structure to represent and operate on ?

Polygon Soup

Draw: O(n) – ☺ Vertex neighbor – ??

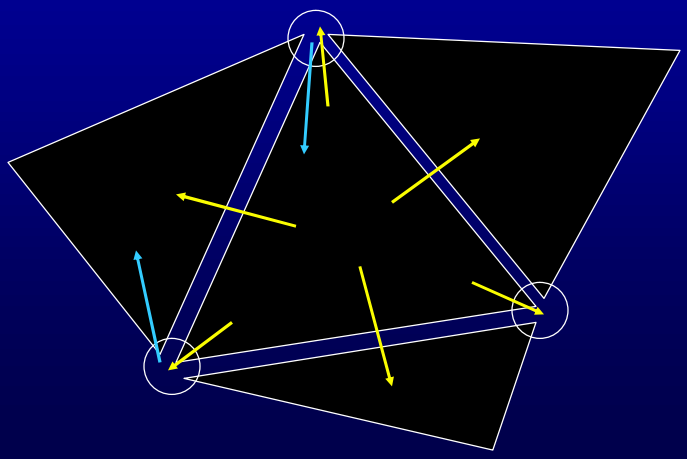
Larson



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)

Adjacency Lists

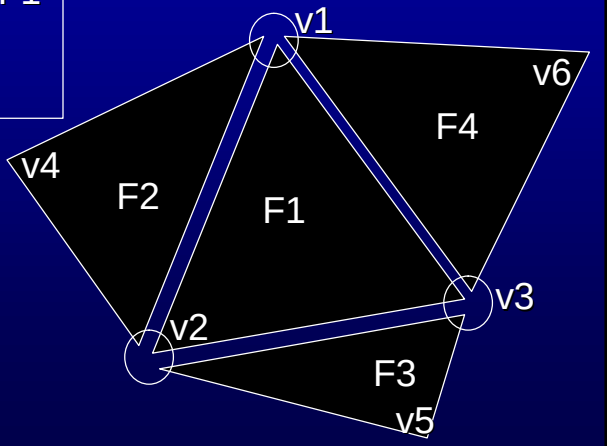
Topology stored in triangles



Adjacency Lists

F1	v1 v2 v3	F3 F4 F2
F2	v2 v1 v4	F1
F3	v2 v5 v3	F1
F4	v3 v6 v1	F1

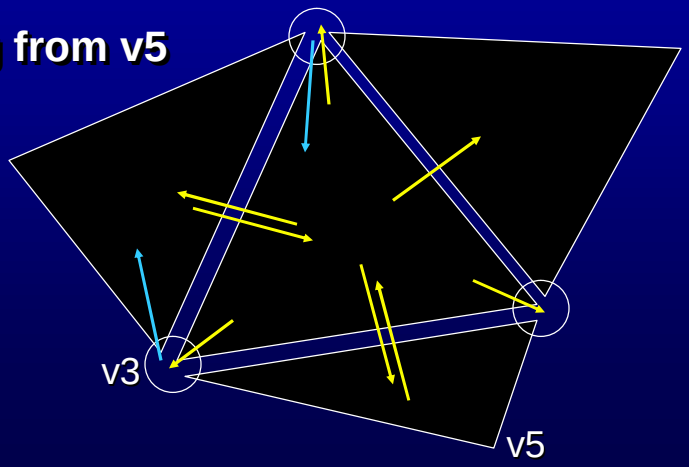
v1	F4
v2	F1
v3	F4
v4	F2
…	



Adjacency Lists

Walk around vertex

v3 starting from v5



Adjacency Lists

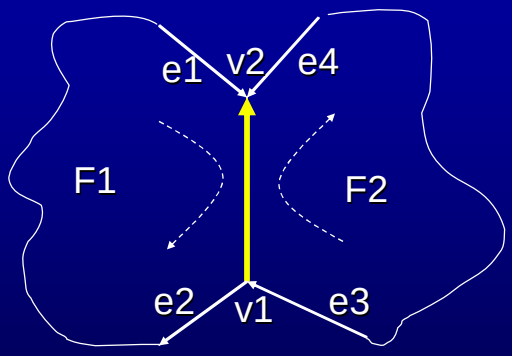
Draw ☺

Manage ☺

Traverse 😐

Polygons: variable storage / face ☹

Winged Edge [Baumgart ’75]



Org	Dest	Left	Right	Lprev	Lnext	Rprev	Rnext
v1	v2	F1	F2	e1	e2	e3	e4

Winged Edge

Edges store topology (fixed size)

Faces, vertices point to 1 adjacent edge

Can represent non-orientable surfaces

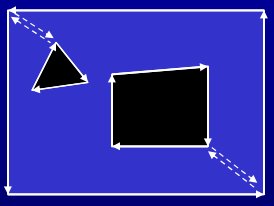
Traversing: test orientation

Holes

Faces point to >1 edge (contour)

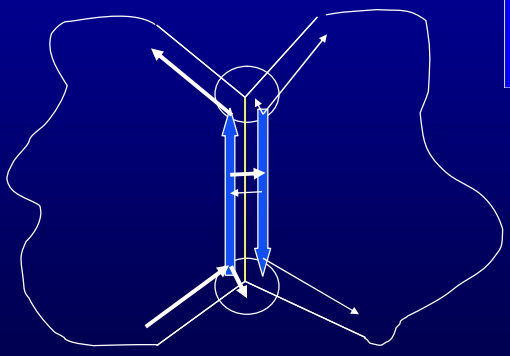
OR

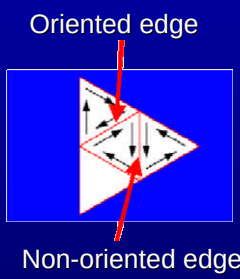
Auxiliary edges



Half-Edge [Eastman ’82]

Orientable manifold

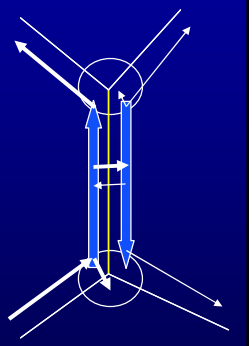




(prevEdge)
nextEdge
symEdge
origin vertex

Half-Edge

Faces, vertices point to 1 edge



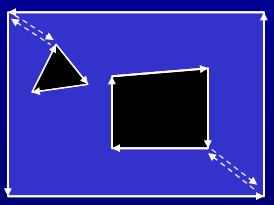
Dest: Org(Sym(he))

Opp Face (he): Sym(he)

CCW(he,v=Org(he)):
Sym(Prev(he))

Holes

Faces point to >1 edge (contour)



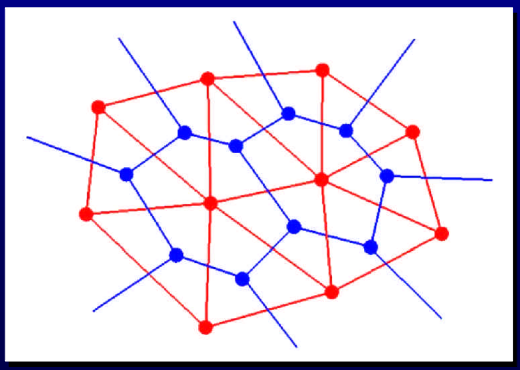
OR

Auxiliary edges

Quad-Edge

Dual: Faces « Vertices; Edges « Edges

Application: Voronoi / Delaunay



Quad Edge [Giubas & Stolfi ’85]

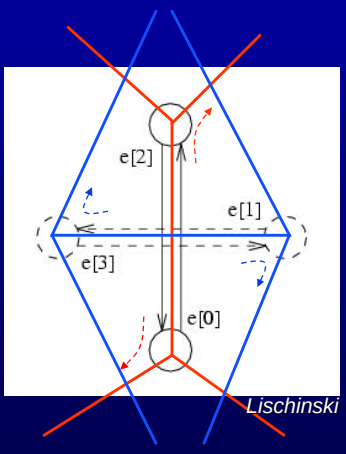
QuadEdge: Edge[4]

Edge:

- Index (0..3)
- Next
- Origin

Sym?

Dest?



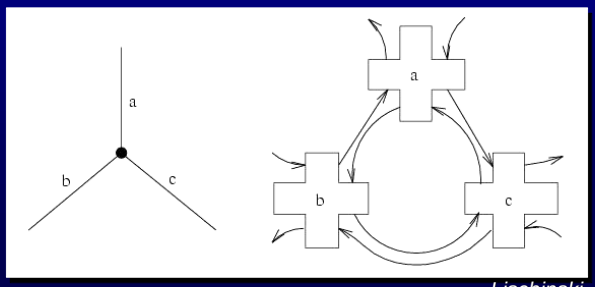
Lischinski

Quad-Edge

Vertices/Faces symmetrical

Rings of Quad Edges

- duplicate coords or use pointers



Lischinski

Build Meshes

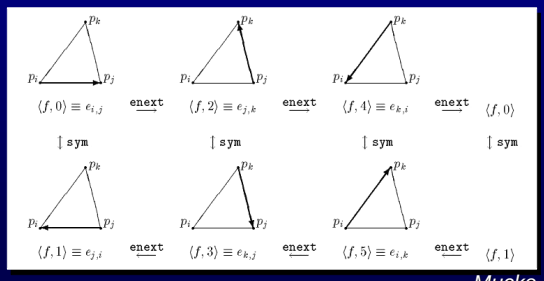
Make Edge

Splice

Triangle-Edge [Muecke ’93]

[Dobkin-Lazlo] Edge-Facet
extend Quad-Edge to 3-manifolds

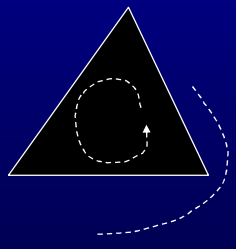
Triangle-Edge: simplification for
triangulations

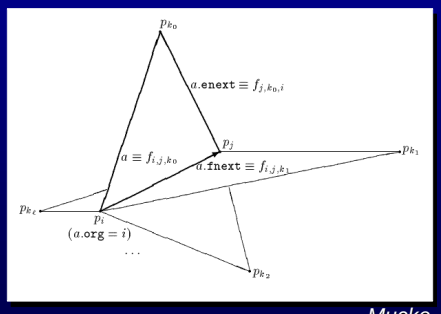


Mucke

Triangle-Edge

Edge rings, Face rings





Mucke

Triangle Edge

Triangle-Edge <f,e> = f*6+e_orient

Store neighbor triangles, 3 vertices

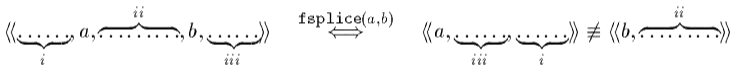
Edge rings obtained by index manipulation

```
type Triangle = record
  origin: array [0..2] of Integer;
  next_f: array [0..5] of Integer;
end;

const
  vo: array [0..5] of Integer ← {0,1,1,2,2,0};
  ve: array [0..5] of Integer ← {2,5,4,1,0,3};

var tr: dynamic array [0..*] of Triangle;
```

(0..2 for 2-manifold)

Mucke

Triangle-Edge




```
procedure fsplice(a, b) is
  swap pointers a, b;
  swap pointers a.fnext.sym, b.fnext.sym;
end;

refinement swap pointers a, b is
  assert a = ⟨f_a, v_a⟩ and b = ⟨f_b, v_b⟩;
  tr[f_a].next_f[v_a] ↔ tr[f_b].next_f[v_b];
end;
```

Mucke

Triangle-Edge

Fsplice, Fmerge: link faces

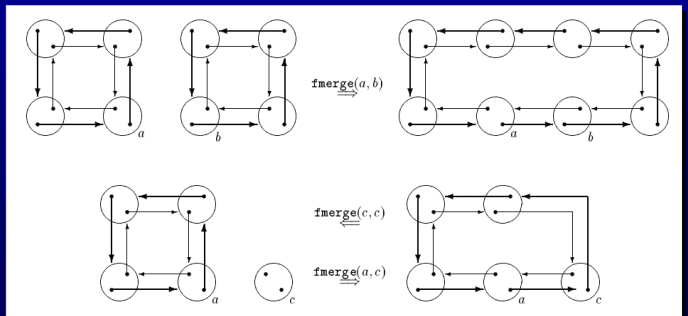


Figure 3.3: The effects of `fmerge`(a, b) = `fsplice`$(a, b$`.sym.fnext.sym`$)$.

Mucke