

DOUBLY-LINKED HALF-EDGE DATA
STRUCTURE, IMPLICIT SURFACES,
ISO-SURFACES ON TETRAHEDRAL
MESHES

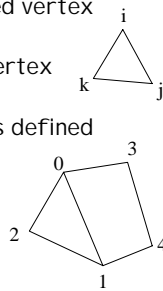
EE-148 3D Photography
Caltech Spring 2001
Gabriel Taubin

3D Representations

- Surfaces
 - Polygonal meshes
 - No connectivity (3 3D vertices per triangle)
 - IndexedFaceSet (VRML file format)
 - Half-Edge data structure (manifold meshes)
 - Boundaries of solid objects
- Volumes (solid objects)
 - Implicit surfaces
 - inside-outside boundary
 - How to convert to polygonal mesh

IndexedFaceSet

- Array of vertex coordinates
- Each 3D vertex has an associated vertex index in $\{0, \dots, V-1\}$
- A triangle is defined by three vertex indices (i, j, k)
- A polygonal face without holes is defined by more indices
- coordIndex [0,1,2,-1,0,3,4,1,-1]
- VRML'97 file format



Polygonal Mesh Components

- Connectivity
 - coordIndex (faces)
- Geometry
 - coord (vertex coordinates)
- Properties
 - color/colorIndex/colorPerVertex
 - normal/normalIndex/normalPerVertex
 - texCoord/texCoordIndex

Connectivity

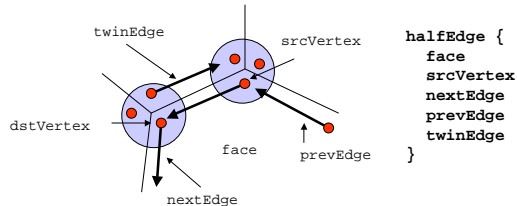
- Edges
 - Boundary (1 incident face)
 - Regular (2 incident faces)
 - Singular (3 or more incident faces)
- Vertices
 - Regular / Singular
- Connected components
 - Connected Components of Dual Graph

Manifold Meshes

- No singular edges
 - Boundary
 - 1 incident face
 - Regular
 - 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 ?

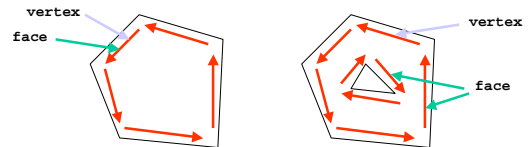
Doubly-linked data structure

- Planar subdivisions
- Orientation
- Vertices / Faces / Half-Edges



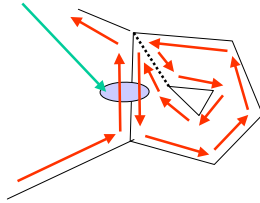
Doubly-linked data structure

- One half-edge per corner of mesh
- Simple Face
 - closed loop of half-edges
- Multiply connected face
 - 1 external loop + one or more internal loops



Orientation

- Consistent if edge is added or removed
- Counterclockwise for outer loop
- Clockwise for inner loops
- Twin edges have opposite orientations



Doubly-linked data structure

- Operations
 - Traversal / triangle strips / compression
 - Surgery / Euler operations
 - Simplification / subdivision
- How to construct from IndexedFaceSet ?
 - Simply connected faces
 - Geometric intersections ignored
- Conversion to manifold
 - Removal of singular edges and vertices

How to construct from IndexedFaceSet ?

- 0) First use original vertex indices
 - $0, \dots, V-1$
- 1) Construct all the half-edges
 - Traverse `coordIndex` array
 - For each face
 - One half-edge per corner
 - Set face, srcVertex, nextEdge and prevEdge
 - Set twinEdge to NULL

How to construct from IndexedFaceSet ?

- 2) Determine regular edges by counting incident faces per edge
 - Use symmetric sparse matrix data structure `m` with operations `m.get(i,j)` and `m.set(i,j,value)`
 - Initialize to 0
 - For each half-edge connecting vertices (i,j) , increment `m.set(i,j,m.get(i,j)+1)`
 - Can be done during previous `coordIndex` traversal

How to construct from IndexedFaceSet ?

- 3) Link half-edges corresponding to regular edges
 - Have to set `twEdge` for half-edges corresponding to `m.get(i,j)==2`
 - Use another symmetric sparse matrix data structure `tw` (or reuse `m`) initialized to `NULL`
 - Traverse half-edges again and save in the `(i,j)` position a pointer to the first corresponding half-edge (`tw.get(i,j)==null`)
 - When the second half-edge corresponding to the `(i,j)` position is visited (`tw.get(i,j)!=null`), and if the orientations are opposite, set the `twEdge` fields of the current half-edge and the one stored in the `(i,j)` position of `tw`.
 - Resulting mesh has no singular edges (have been converted to boundary edges)

How to construct from IndexedFaceSet ?

- 4) Remove singular vertices
 - Use a partition data structure `p` with operations `p.find(i)`, and `p.join(i,j)` to maintain a partition of the corners of the coordinate array
 - Initialize to one singleton per corner
 - `{(0)...(C-1)}`
 - For each half-edge `e` with `e.twEdge!=null`, join the corresponding corners
 - `P.join(e.corner,e.twEdge.nextEdge.corner)`
 - Assign consecutive indices to the sets in the partition `0,...,P-1`
 - Make new vertex coordinates array of length `P`
 - Set `e.srcVertex` to the partition number of `e.corner`

Implicit surfaces

- Set of zeros of a function
 - $\{ (x,y,z) : f(x,y,z) = 0 \}$
- Good for boolean operations (CSG)
- Difficult to render (ray-tracing)
- Iso-surface
 - Function defined by piecewise function
 - Volumetric mesh
 - 1 function value per vertex
- Iso-surface algorithm
 - Conversion to triangle or polygon mesh representation

Piecewise Linear Functions

- Triangle : Barycentric coordinates
 - Triangle / Tetrahedron / Simplex
- Every point in 3D can be written as a unique affine combination of 4 non-coplanar points (affine basis)
- Every linear function in 3D can be specified by its values at the 4 vertices of an affine basis
- A piecewise-linear function is specified in 3D by its values at the vertices of a tetrahedral mesh (volumetric).

Affine bases / Linear function

$$p = \lambda_0 p_0 + \lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3$$

$$\begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

$$f(p) = \lambda_0 f(p_0) + \lambda_1 f(p_1) + \lambda_2 f(p_2) + \lambda_3 f(p_3)$$

Implicit Linear Surfaces / Curves



0000 : []	0001 : [bd, cd, ed]	0010 : [ac, cd, bc]	0011 : [ad, bd, bc, ac]
0100 : [ab, bc, bd]	0101 : [ad, ab, bc, cd]	0110 : [ab, ac, cd, bd]	0111 : [ab, ac, ad]
1000 : [ab, ad, ac]	1001 : [ab, bd, cd, ac]	1010 : [ab, ad, cd, bc]	1011 : [ab, bd, bc]
1100 : [ad, ac, bc, bd]	1101 : [ad, ac, bc]	1110 : [bd, cd, cd]	1111 : []

Iso-surfaces on tetrahedral meshes

- Piecewise linear function defined on vertices of tetrahedral mesh $f(i)$
- For each edge (i,j) such that $f(i)f(j)<0$
 - create a surface vertex $v(i,j)$
- For each tetrahedron (i,j,k,l)
 - Skip if all vertices are positive or negative
 - Else if 3 positive or 3 negative create a triangle
 - Else (if 2 positive and 2 negative) create two triangles
- Output triangle mesh is IndexedFaceSet
- Is it a manifold mesh ? Why ?