

DOUBLY-LINKED HALF-EDGE DATA STRUCTURE

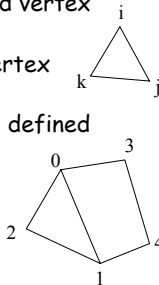
EN-193s08 3D Photography
Brown Fall 2003
Gabriel Taubin

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

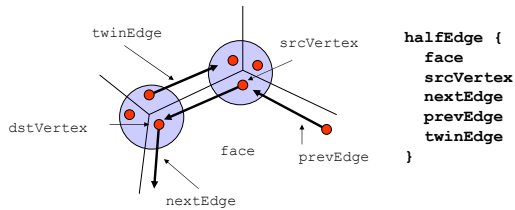
- Edges
 - Boundary (1 incident face)
 - Regular (2 incident faces)
 - Singular (3 or more incident faces)
- Vertices
 - Regular / Singular
- Connected components
 - Connected Components of Dual Graph

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 ?

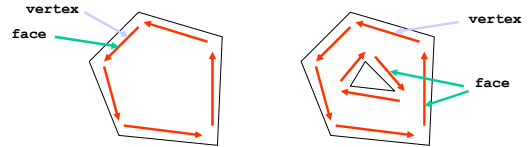
Doubly-linked data structure

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



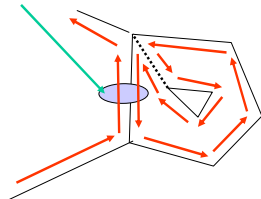
Doubly-linked data structure

- 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,...,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 twinEdge 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 twinEdge 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 coordIndex array
 - Initialize to one singleton per corner
 - `{{0},...{C-1}}`
 - For each half-edge `e` with `e.twinEdge!=null`, join the corresponding corners
 - `P.join(e.corner,e.twinEdge.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`