Mesh Data Structures
Data Structures

• What should be stored?
  – Geometry: 3D coordinates
  – Attributes
    • e.g. normal, color, texture coordinate
    • Per vertex, per face, per edge
  – Connectivity
    • Adjacency relationships
Data Structures

• What should it support?
  – Rendering
  – Geometry queries
    • What are the vertices of face #2?
    • Is vertex A adjacent to vertex H?
    • Which faces are adjacent to face #1?
  – Modifications
    • Remove/add a vertex/face
    • Vertex split, edge collapse
Data Structures

• How good is a data structure?
  – Time to construct (preprocessing)
  – Time to answer a query
  – Time to perform an operation
  – Space complexity
  – Redundancy
Mesh Data Structures

• Shared Vertex
• Face Set
• Adjacency Matrix
• Half Edge
• Face Based Connectivity
• Edge Based Connectivity
• Corner Table
Shared Vertex

- Connectivity
- No neighborhood
- Oriented triangles

### TRIANGLES

<table>
<thead>
<tr>
<th>Vertex Index</th>
<th>Vertex Index</th>
<th>Vertex Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

### VERTICES

<table>
<thead>
<tr>
<th>Vertex Coord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[40 5 20]</td>
</tr>
<tr>
<td>[10 20 30]</td>
</tr>
<tr>
<td>[10 4 3]</td>
</tr>
</tbody>
</table>

- •
- •
- •
Shared Vertex

**TRIANGLES**

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
</table>

**VERTICES**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>[20 10 0]</td>
</tr>
<tr>
<td>v2</td>
<td>[19 20 0]</td>
</tr>
<tr>
<td>v3</td>
<td>[14 15 0]</td>
</tr>
</tbody>
</table>
• What are the vertices of face $f_1$?
  – $O(1)$ – first triplet from face list
• What are the one-ring neighbors of $v_3$?
  – Requires a full pass over all faces $O(nf)$
Shared Vertex

- Are vertices $v_1$ and $v_5$ adjacent?
  - Requires a full pass over all faces $O(nf)$
## Face Set

<table>
<thead>
<tr>
<th>TRIANGLES</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[10 20 30]</td>
<td>[40 5 20]</td>
<td>[10 4 3]</td>
</tr>
</tbody>
</table>

- Simple
- STL File
- No connectivity
- Redundancy
- Adjacency Matrix “A”
- If there is an edge between $v_i$ & $v_j$ then $A_{ij} = 1$
Adjacency Matrix

• Symmetric for undirected simple graphs
• $(A^n)_{ij} = \# \text{ paths of length } n \text{ from } v_i \text{ to } v_j$
• Pros:
  – Can represent non-manifold meshes
• Cons:
  – No connection between a vertex and its adjacent faces
Adjacency Matrix

• How can it be constructed using shared vertex?
• What about faces?
• Vertex-face?
• How can we test it?
Half Edge Data Structure

- Vertex stores
  - Position
  - 1 outgoing halfedge
Half Edge Data Structure

- **Halfedge** stores
  - 1 origin vertex index
  - 1 incident face index (counter-clockwise orientation)
  - next, prev, twin halfedge indices
Half Edge Data Structure

- Face stores
  - 1 adjacent halfedge index
Half Edge Data Structure

- Neighborhood Traversal
Half Edge Data Structure

• How can it be constructed using shared vertex?
  – How can we construct “edges”? 
Face Based Connectivity

- **Vertex:**
  - position
  - 1 adjacent face index

- **Face:**
  - 3 vertex indices
  - 3 neighboring face indices

- No (explicit) edge information
Edge Based Connectivity

• Vertex
  – position
  – 1 adjacent edge index

• Edge
  – 2 vertex indices
  – 2 neighboring face indices
  – 4 edges

• Face
  – 1 edge index

• No edge orientation information
Corner Table

• Corner is a vertex with one of its incident triangles
Corner Table

- Corner is a vertex with one of its incident triangles

Corner – c
Corner Table

- Corner is a vertex with one of its incident triangles
  Corner – c
  Triangle – c.t
Corner Table

- Corner is a vertex with one of its incident triangles
  - Corner — c
  - Triangle — c.t
  - Vertex — c.v
Corner Table

- Corner is a vertex with one of its incident triangles
  - Corner – c
  - Triangle – c.t
  - Vertex – c.v
  - Next corner in c.t (ccw) – c.n
Corner Table

- Corner is a vertex with one of its incident triangles
  - Corner – c
  - Triangle – c.t
  - Vertex – c.v
  - Next corner in c.t (ccw) – c.n
  - Previous corner – c.p (== c.n.n)
Corner is a vertex with one of its incident triangles

- Corner – c
- Triangle – c.t
- Vertex – c.v
- Next corner in c.t (ccw) – c.n
- Previous corner – c.p (== c.n.n)
- Corner opposite c – c.o

Edge E opposite c not incident on c.v
Triangle T adjacent to c.t across E
c.o.v vertex of T that is not incident on E
Corner Table

• Corner is a vertex with one of its incident triangles
  Corner – c
  Triangle – c.t
  Vertex – c.v
  Next corner in c.t (ccw) – c.n
  Previous corner – c.p (== c.n.n)
  Corner opposite c – c.o
    Edge E opposite c not incident on c.v
    Triangle T adjacent to c.t across E
    c.o.v vertex of T that is not incident on E
  Right corner – c.r – corner opposite c.n (== c.n.o)
Corner Table

• Corner is a vertex with one of its incident triangles
  Corner – c
  Triangle – c.t
  Vertex – c.v
  Next corner in c.t (ccw) – c.n
  Previous corner – c.p (== c.n.n)
  Corner opposite c – c.o
    Edge E opposite c not incident on c.v
    Triangle T adjacent to c.t across E
    c.o.v vertex of T that is not incident on E
  Right corner – c.r – corner opposite c.n (== c.n.o)
  Left corner – c.l (== c.p.o == c.n.n.o)
Corner Table

• Corner is a vertex with one of its incident triangles
  Corner – c
  Triangle – c.t
  Vertex – c.v
  Next corner in c.t (ccw) – c.n
  Previous corner – c.p (== c.n.n)
  Corner opposite c – c.o
    Edge E opposite c not incident on c.v
    Triangle T adjacent to c.t across E
    c.o.v vertex of T that is not incident on E
  Right corner – c.r – corner opposite c.n (== c.n.o)
  Left corner – c.l (== c.p.o == c.n.n.o)
Corner Table

- Corner is a vertex with one of its incident triangles

<table>
<thead>
<tr>
<th>corner</th>
<th>c.v</th>
<th>c.t</th>
<th>c.n</th>
<th>c.p</th>
<th>c.o</th>
<th>c.r</th>
<th>c.l</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>v₁</td>
<td>f₁</td>
<td>c₂</td>
<td>c₃</td>
<td>c₆</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Corner Table

- Corner is a vertex with one of its incident triangles
Corner Table

• Store:
  – Corner table
  – For each vertex – a list of all its corners

• Corner number \( j \times 3 - 2, j \times 3 - 1 \) and \( j \times 3 \) match face number \( j \)
• What are the vertices of face #3?
  – Check c.v of corners 9, 8, 7
Corner Table

• Are vertices 2 and 6 adjacent?
  – Scan all corners of vertex 2, check if c.p.v or c.n.v are 6
Corner Table

- Which faces are adjacent to vertex 3?
  - Check c.t of all corners of vertex 3
Corner Table

• One ring neighbors of vertex $v_4$?
  – Get the corners $c_6 \ c_8 \ c_{10}$ of this vertex
  – Go to $c_i.n.v$ and $c_i.p.v$ for $i = 6, 8, 10$.
  – Remove duplicates
Corner Table

• Pros:
  – All queries in O(1) time
  – Most operations are O(1)
  – Convenient for rendering (triangle fans)

• Cons:
  – Only triangular, manifold meshes
  – Redundancy
In Practice?

• Also depends on the framework
  – In MATLAB, most existing code uses the “shared vertex” structure
  – Sparse adjacency matrices can be helpful in addition, in case there are many adjacency queries (simple to construct and store)