Syllabus

Curves in 2D and 3D
Properties of Curves and Surfaces
Surfaces
Polygonal meshes
- Definitions & Data Structures
- Acquisition
- Analysis & Smoothing
- ...
- Subdivision (Loop, $\sqrt{3}$, …)
Storing mesh data

Uses of mesh data:
- Rendering
- Geometry queries
  - What are the vertices of face #3?
  - Are vertices i and j adjacent?
  - Which faces are adjacent to face #7?
- Geometry operations
  - Remove/add a vertex/face

Storage of generic meshes – hard to implement efficiently
Assume: orientable, manifold & triangular

Storing mesh data (cont.)

How “good” is a data structure?
- Time to construct - preprocessing
- Time to answer a query
- Time to perform an operation (update the data structure)
- Space complexity
- Redundancy
List of faces

List of vertices (coordinates)

List of faces - triplets of pointers to face vertices \((c_1, c_2, c_3)\)

**List of faces - example**

<table>
<thead>
<tr>
<th>vertex</th>
<th>coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_1)</td>
<td>((x_1, y_1, z_1))</td>
</tr>
<tr>
<td>(v_2)</td>
<td>((x_2, y_2, z_2))</td>
</tr>
<tr>
<td>(v_3)</td>
<td>((x_3, y_3, z_3))</td>
</tr>
<tr>
<td>(v_4)</td>
<td>((x_4, y_4, z_4))</td>
</tr>
<tr>
<td>(v_5)</td>
<td>((x_5, y_5, z_5))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>face</th>
<th>vertices (ccw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1)</td>
<td>((v_1, v_2, v_3))</td>
</tr>
<tr>
<td>(f_2)</td>
<td>((v_2, v_5, v_3))</td>
</tr>
<tr>
<td>(f_3)</td>
<td>((v_3, v_2, v_6))</td>
</tr>
<tr>
<td>(f_4)</td>
<td>((v_4, v_1, v_6))</td>
</tr>
</tbody>
</table>
List of faces

**Queries:**
- What are the vertices of face #3?
  - Answered in $O(1)$ - checking third triplet
- Are vertices $i$ and $j$ adjacent?
  - A pass over all faces is necessary – NOT GOOD

List of faces – pros and cons

**Pros:**
- Convenient and efficient (memory wise)
- Can represent non-manifold meshes

**Cons:**
- Too simple - not enough information on relations between vertices & faces
Half-Edge Data Structure

Record for each face, edge and vertex:

- Geometric information
- Topological information
- Attribute information

also called DCEL (Doubly-Connected Edge List) or Winged-Edge Structure

Half-Edge Data Structure (cont.)

- Vertex record:
  - Coordinates
  - Pointer to one half-edge that has v as its origin
- Face record:
  - Pointer to one half-edge on its boundary

Half-edge record:

- Pointer to its origin, origin(e)
- Pointer to its twin half-edge, twin(e)
- Pointer to the face it bounds, IncidentFace(e) (face lies to left of e when traversed from origin to destination)
- Next and previous edge on boundary of IncidentFace(e)
Half-Edge Data Structure (cont.)

**Operations supported:**
- Walk around boundary of given face
- Visit all edges incident to vertex $v$

**Queries:**
- Most queries are $O(1)$

### Half-Edge Data Structure - example

<table>
<thead>
<tr>
<th>Vertex $v_i$</th>
<th>Coordinate $(x_i, y_i, z_i)$</th>
<th>Incident Edge $e_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$(x_1, y_1, z_1)$</td>
<td>$e_{2,1}$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$(x_2, y_2, z_2)$</td>
<td>$e_{5,1}$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$(x_3, y_3, z_3)$</td>
<td>$e_{1,1}$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$(x_4, y_4, z_4)$</td>
<td>$e_{3,1}$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>$(x_5, y_5, z_5)$</td>
<td>$e_{9,1}$</td>
</tr>
<tr>
<td>$v_6$</td>
<td>$(x_6, y_6, z_6)$</td>
<td>$e_{7,2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Face $f_i$</th>
<th>Edge $e_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$e_{1,1}$</td>
</tr>
<tr>
<td>$f_2$</td>
<td>$e_{5,1}$</td>
</tr>
<tr>
<td>$f_3$</td>
<td>$e_{4,2}$</td>
</tr>
<tr>
<td>$f_4$</td>
<td>$e_{8,1}$</td>
</tr>
<tr>
<td>$f_5$</td>
<td>$e_{9,1}$</td>
</tr>
</tbody>
</table>

© Wolfgang Heidrich & Alla Sheffer
Half-Edge – example (cont.)

Half-edge | origin | twin | IncidentFace | next | prev
---|---|---|---|---|---
$e_{1,1}$ | $v_2$ | $e_{1,2}$ | $f_1$ | $e_{1,1}$ | $e_{2,1}$
$e_{1,2}$ | $v_3$ | $e_{1,1}$ | $f_2$ | $e_{5,1}$ | $e_{4,1}$
$e_{4,1}$ | $v_4$ | $e_{4,2}$ | $f_2$ | $e_{3,2}$ | $e_{4,1}$
$e_{4,2}$ | $v_3$ | $e_{4,1}$ | $f_3$ | $e_{7,1}$ | $e_{6,1}$

Half-Edge – pros and cons

**Pros:**
- All queries in $O(1)$ time
- All operations are $O(1)$ (usually)

**Cons:**
- Represents only manifold meshes
Processing Pipeline

Acquisition (& repair)
- Input Data
  - Laser scan
  - CAD
  - Tomography
- Removal of topological and geometrical errors

Analysis & Smoothing
- Analysis of surface quality
  - Surface smoothing for noise removal

Simplification
- Simplification for complexity reduction

Parameterization

Remeshing
- Remeshing for improving mesh quality

Modeling
- Surface form and multi-resolution modeling

[Botch, 2006]

© Wolfgang Heidrich & Alla Sheffer