Polygon Meshes

Polygon Meshes Representation

- which representation is good?
  - often triangles/quads only - will work on triangles

- compact
- efficient for rendering
  - fast enumeration of all faces
- efficient for geometry algorithms
  - finding adjacency (what is close to what)

Vertices, Edges, Faces

- fundamental entities
  - $n_v$ vertices
  - $n_e$ edges
  - $n_f$ faces
  - simple closed surface: $n_v - n_e + n_f = 2$

- fundamental properties:
  - topology: how faces are connected
  - geometry: where faces are in space
  - separate issues
    - algorithms mostly care about topology

Topology vs. Geometry

- same geometry
different topology

- same topology
different geometry
### Triangles

- array of vertex data
  - vertex[nv][3]
  - vertex stores position and optional data (normal, uvs)
  - ~72 bytes per triangle with vertex position only

- redundant
- adjacency is not well defined
  - floating point errors in comparing vertices

---

### Indexed Triangles

- array of vertex data
  - vertex[nv]
  - 12 bytes per vertex with position only

- array of vertex indices (3 per triangle)
  - int[nf][3], often flattened in a single array
  - 24 bytes per triangle

- total storage: ~36 bytes (50% memory)

- topology/geometry stored separately/explicitly
  - adjacency queries are well defined
**Triangles Strips**

- since triangle share edges, let every triangle reuse the last one’s vertices

**Array of Vertex Data**

- vertex[nv]
  - 12 bytes per vertex with position only

**Array of Lists of Vertex Indices**

- int[nf][varyingLength]

- for long lists saves about 1/3 index memory

**Triangles Strips**

- requires multiple strips for general case

---

**Triangle Strips**

- array of vertex data
  - vertex[nv]
  - 12 bytes per vertex with position only

- array of lists of vertex indices
  - int[nf][varyingLength]

**Triangle Fans**

- idea similar to triangle strips
- different arrangement
**Quad Meshes**

- similar options as for storing triangles
  - flat quads
  - indexed quad meshes
  - quad strips, no fans

**Adjacency Queries**

- example queries
  - given a face, find adjacent faces
  - given an edge, find faces that share it
  - given a vertex, find faces that share it

- previous data structures
  - inefficient adjacency queries, O(n)

**Adjacency Lists**

- store all vertex, edge, face adjacency
  - efficient adjacency queries, O(1)
  - extra storage

**Partial Adjacency Lists**

- store some vertex, edge, face adjacency
  - goal: efficient adjacency queries
  - goal: less storage
Winged Edge

- adjacency stored in edges
  - all adjacency in $O(1)$
  - little extra storage

\begin{align*}
\{E_k\} & \quad \{V_i\} \\
1 & \quad 2 & \quad 1 & \quad 2
\end{align*}

[based on Finkelstein 2004]

\begin{tabular}{|c|c|c|c|c|c|}
\hline
edge & vertex 1 & vertex 2 & face left & face right & pred left & pred right & succ left & succ right \\
\hline
A & 1 & 2 & 0 & a & b & c & d & e \\
B & 2 & 1 & 0 & c & d & a & b & e \\
C & 0 & 1 & 1 & b & a & c & d & e \\
D & 3 & 1 & 0 & e & c & b & a & d \\
\hline
\end{tabular}

[Shirley]

Winged Edge

- tetrahedron example

\begin{align*}
\text{vertex} & \quad \text{edge} & \quad \text{face} & \quad \text{edge} \\
A & a & 0 & a \\
B & b & 1 & c \\
C & c & 2 & d \\
D & d & 3 & a \\
\end{align*}

[Shirley]