polygon meshes

polyon 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[ni][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

\[
\begin{array}{|c|c|}
\hline
\text{vertex[0]} & (x_0, y_0, z_0) \\
\hline
\text{vertex[1]} & (x_1, y_1, z_1) \\
\hline
\vdots & \vdots \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{strip[0]} & 0,1,2,3,4,5 \\
\hline
\text{strip[1]} & \ldots \\
\hline
\vdots & \vdots \\
\hline
\end{array}
\]

triangles strips

• requires multiple strips for general case

\[
\begin{array}{|c|c|}
\hline
\text{vertex[0]} & (x_0, y_0, z_0) \\
\hline
\text{vertex[1]} & (x_1, y_1, z_1) \\
\hline
\vdots & \vdots \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{strip[0]} & 0,1,2,3,4,5 \\
\hline
\text{strip[1]} & \ldots \\
\hline
\vdots & \vdots \\
\hline
\end{array}
\]
**triangle strips**

- 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

**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

![Diagram of adjacency lists]

[Based on Finkelstein 2004]

partial adjacency lists

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

![Diagram of partial adjacency lists]

[Based on Finkelstein 2004]
**winged edge**

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

![Diagram of winged edge with adjacency stored in edges](image)

(based on Finkelstein 2004)

**winged edge**

- tetrahedron example

![Diagram of winged edge with tetrahedron example](image)

(Written in based on Shirley notation)

---

**computer graphics • polygon meshes** © 2009 fabio pellacini • 17

**computer graphics • polygon meshes** © 2009 fabio pellacini • 18
defining normals

- face normal
  - geometrically correct
  - not for smooth surfaces

- vertex normals
  - geometrically “inconsistent”
  - for smooth surface approx.