Polygonal models

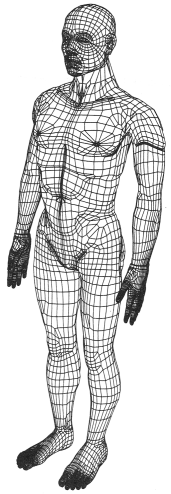
COM3404

Richard Everson

School of Engineering, Computer Science and Mathematics
University of Exeter

R.M.Everson@exeter.ac.uk
http://www.secamlocal.ex.ac.uk/studyres/COM304

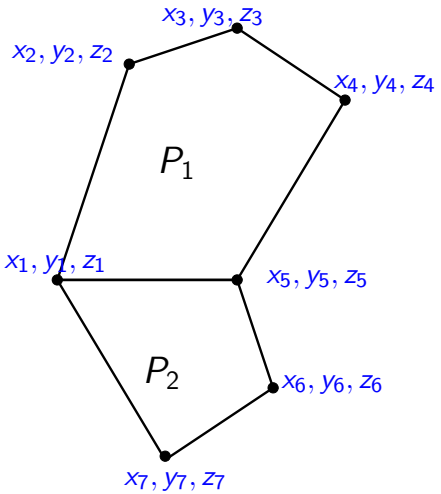
Outline



References

- Principles of Computer Animation. O'Rourke. Chapter 1
- Fundamentals of 3D Computer Graphics. Watt. Chapters 1 & 2
- Computer Graphics: Principles and Practice. Foley, van Dam, Feiner, Hughes

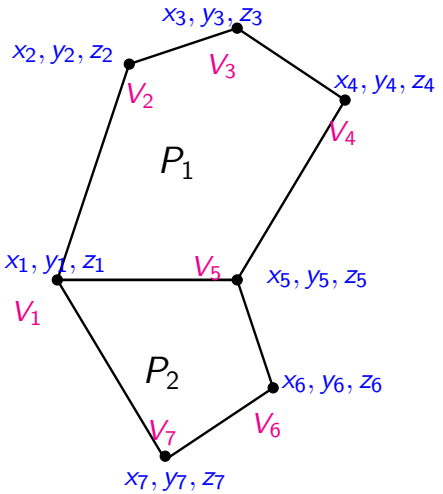
Polygon representation: Vertices explicit



$P_1 = [(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots]$
$P_2 = [(x_1, y_1, z_1), (x_5, y_5, z_5), \ldots]$

- Conceptually simple.
- Requires a convention on order of vertices
- No knowledge of shared vertices or edges
- Shared edges drawn twice
- Vertex duplication, so difficult to move vertices

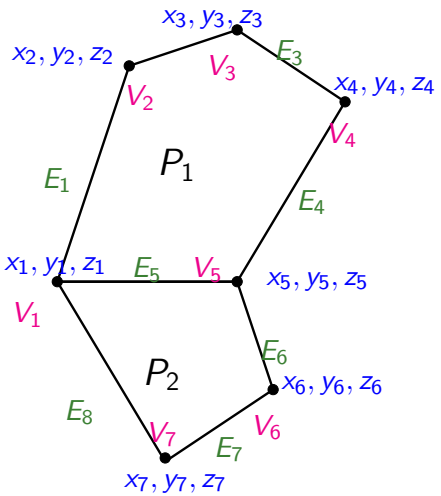
Polygon representation: Pointer to vertex list



$P_1 = [V_1, V_2, V_3, V_4, V_5]$
$P_2 = [V_1, V_5, V_6, V_7]$
$V_1 = (x_1, y_1, z_1), \quad V_2 = (x_2, y_2, z_2)$

- Each vertex stored once
- Vertex location easily updated
- Difficult to find common edges
- Edges drawn twice
- Requires convention on order of vertices

Polygon representation: Edge based

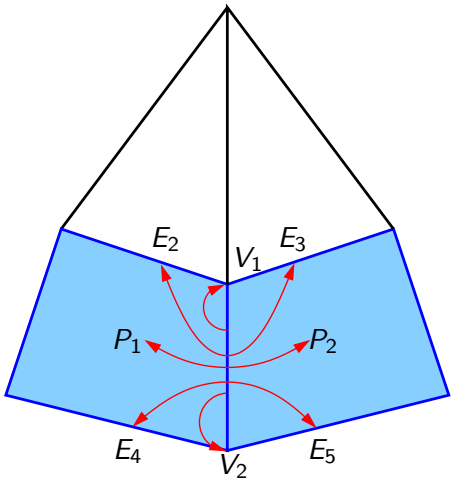


$$E_1 = (V_1, V_2, P_1, P_0)$$
$$E_5 = (V_1, V_5, P_1, P_2)$$

- Simple data structure
- Each edge has two vertices
- Each edge shared by at most two polygons
- Dummy polygon P_0 used for edges adjacent to only one polygon
- Good for scan line based rendering
- Vertex location easily updated if pointers to vertices are kept

Winged edge data structure

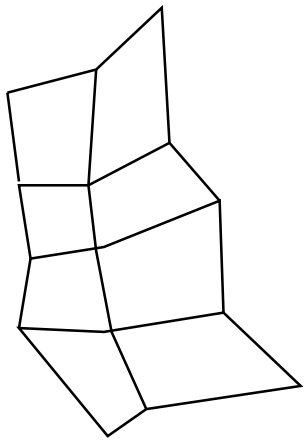
- Adjacency relationships: which vertices, edges, polygon are adjacent to a vertex, edge or polygon?
- Winged edge data structure permits vertices and polygons adjacent to an edge to be located in constant time.



Back-pointers from V_1, V_2, P_1 and P_2 to one of their edges.

Polygon representation: Rectangular mesh

Vertices arranged in a regular $N \times M$ array

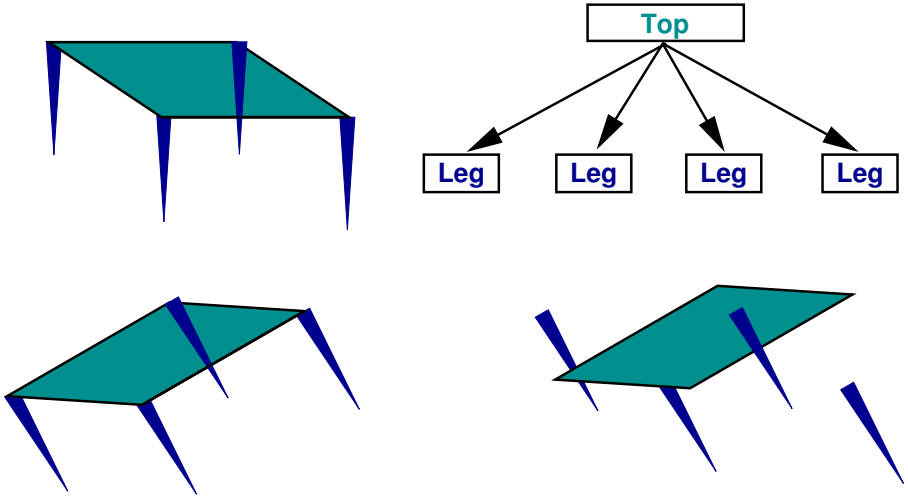


- Edges connect vertices to form mesh
- Edge shared by at most two vertices
- Vertices indexed by mesh coordinates

- Frequently generated by procedural methods (e.g., fractal subdivision)
- Some geometries may not conform to Cartesian structure (e.g., insertion of a new vertex)

- Some graphics hardware (e.g., Silicon Graphics) supports *triangle strips* and *triangle fans*

Object representation



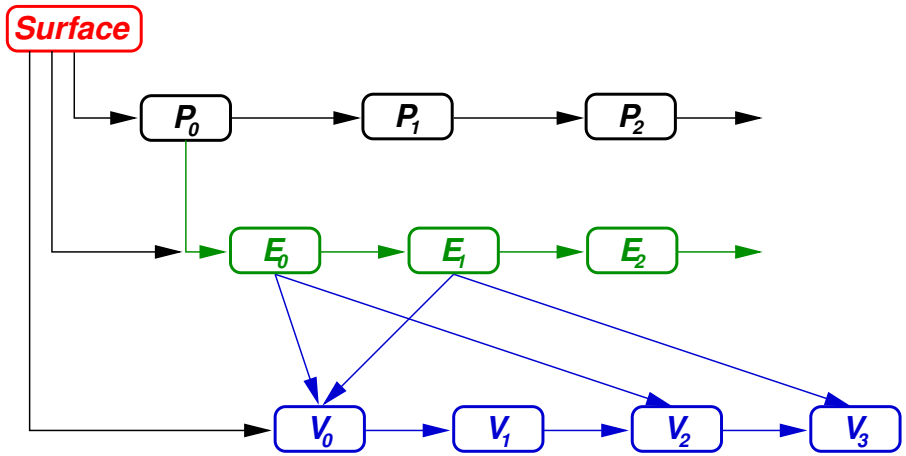
Organise objects in a hierarchy

- Propagate transformations to the object through to component parts

Object representation

Hierarchical structure

Object ⟶ Surface ⟶ Polygon ⟶ Edge ⟶ Vertex



Attributes

Object

- List of vertices
- List of surfaces
- Transformation

Surface

- List of polygons
- Common details of appearance

Polygon

- Normal
- Area
- Convex?
- Type (triangular, etc)
- Coefficients defining plane

Edge

- Length
- Between polygons or surfaces?
- Polygons

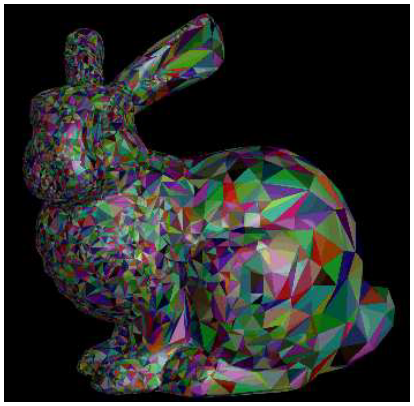
Vertex

- Location
- Refs to polygons containing vectex
- Vertex normal
- Coordinates in texture map

Polygon simplification

Minimise polygon count while maintaining perceptual fidelity

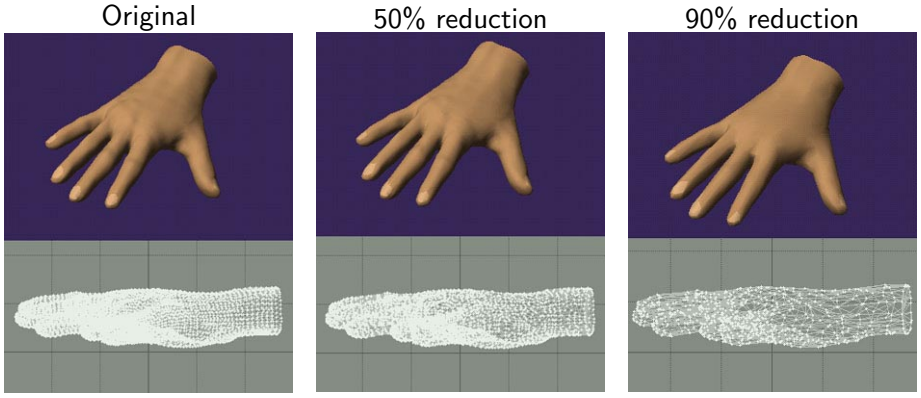
- Merge adjacent polygons when dihedral angle is small
- Delete vertices close to common plane of shared polygons
- Merge polygons when curvature is small
- Mesh simplification: merge polygons dependent on required level of detail.
- Hidden edge removal
- Progressive rendering
- Skillful design



Simplification dependent upon curvature.

Cohen et al, "Simplification envelopes" SIGRAPH 1996

Polygon optimisation



13382 f; 7339 v | 7886 f; 3985 v | 1576 f; 805 v

`http://www.mootools.com/plugins/us/polygoncruncher/examples2.htm`