EECS 487: Interactive Computer Graphics

Lecture 36:
- Polygonal mesh simplification

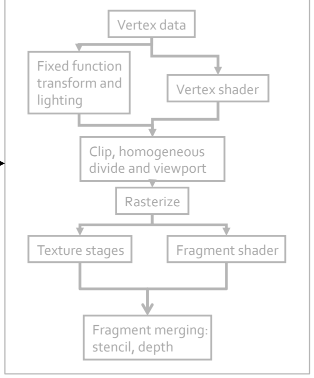
The Modeling-Rendering Paradigm

Modeler:

Modeling complex shapes
- no equation for a chair, face, etc.
- instead, achieve complexity using simple pieces
  - polygons, parametric surfaces, or implicit surfaces
  - with arbitrary precision, in principle

scene graph →

Renderer:



3D Geometry Representations

Represent different kinds of information:
point data, surface data, volumetric data

Points
- 2D: range image
- 3D: point cloud

Surfaces
- Polygonal mesh
- Parametric surfaces
- Subdivision surfaces
- Implicit surfaces

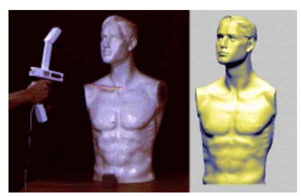
Solids
- Constructive Solid Geometry
- Voxels

Procedural
- Particle system
- Spring-mass system
- Fractals

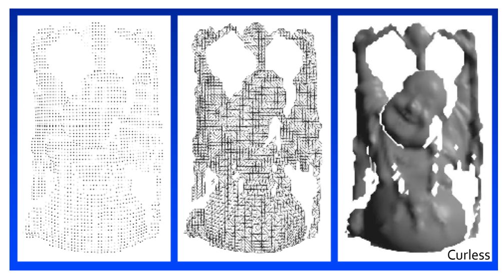
Funkhouser

2D: Range Image

Image with depth information
- acquired from range scanner, incl. Microsoft Kinect and Google Tango
- not a complete 3D description: does not include part of object occluded from viewpoint



Cyberware



Range image   Tessellation   Range surface

Funkhouser, Ramamoorthi

3D: Point Cloud

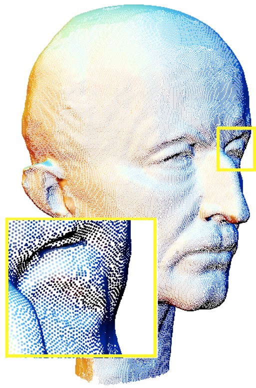
Unstructured set of 3D point samples

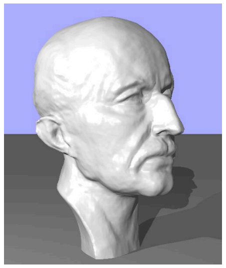
Acquired from range finder

Disadvantage: no structural info

- adjacency/connectivity have to use e.g., k-nearest neighbors to compute

Increasingly hot topic in graphics/vision today



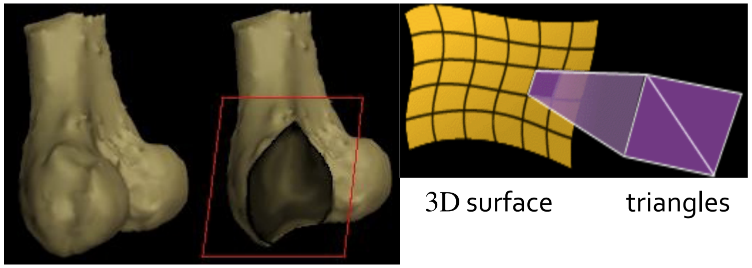


Funkhouser, Ramamoorthi,Ohtake

Surfaces

Boundary representation (B-reps)

- sometimes we only care about the surface, e.g., when rendering opaque objects and performing geometric computations

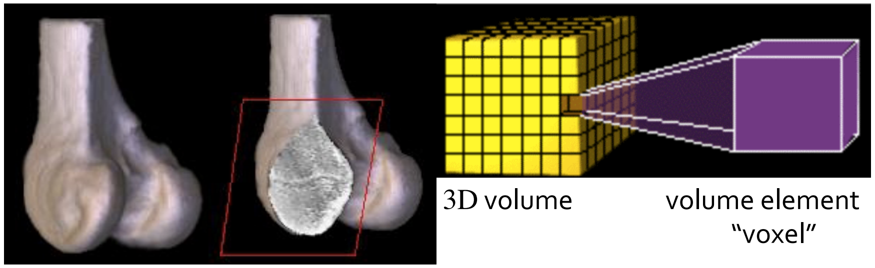


Chenney, Lozanc

Solid Modeling

Some representations are best thought of as defining the space filled

- medical data with information attached to the space
- transparent objects with internal structure
- taking cuts out of an object, "What will I see if I break this object?"



Chenney, Lozanc

Choosing a Representation

Efficiency for different tasks:

- creation/acquisition: by hand, procedurally, by fitting to measurements
- interaction/manipulation: simplification, compression, local control of shape for modeling, animation, etc.
- geometric computation: distance, intersection, normal vectors, smoothness and continuity, ability to evaluate derivatives, curvature, similarity comparisons, indexing, search
- storage and transmission: compactness
- rendering, e.g., with hardware accelerator: convert to polygon

Funkhouser, Chenney

Advantages of Representations

Manipulation:

- splines easiest originally, but now many algorithms for polygon meshes

Acquisition and modeling:

- splines, CSG originally used for modeling
- but increasingly complex meshes, range images, and point cloud acquired from real world

Simplicity: meshes

Efficient hardware rendering: meshes

Ramamoorthi

What do People Use?

Triangle meshes most widely used

Subdivision surfaces used a lot in movies

Spline patches used by modeling programs

Constructive Solid Geometry (CSG) used for modeling machine parts

Volume data used in medical imaging

Range images used in image-based rendering

Point clouds becoming increasingly relevant, especially with computer vision

Ramamoorthi

Comparison of B-Reps

Features	Polygonal Mesh	Implicit Surface	Parametric Surface	Subdivision Surface
Accurate	✗	✔	✔	✔
Compact	✗	✔	✔	✔
Intuitive	✗	✗	✔	✗
Local	✔	✗	✔	✔
Affine	✔	✔	✔	✔
Real objects	✔	✔	✗	✔
Continuity	✗	✔	✔	✔
Parameterization	✗	✗	✔	✗
Rendering	✔	✗	✔	✔
Intersections	✗	✔	✗	✗

Funkhouser

Polygonal Mesh

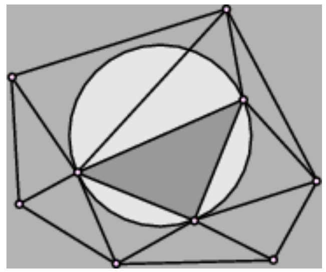
Introduction to five topics:

1. how to draw good looking meshes?
2. mesh simplification
3. level of detail
4. mesh representation
5. error checking and mesh processing

Delaunay Triangulation

How to create a "good looking" triangle mesh from a set of points?

- minimum vertices and triangles
  - fewer, larger triangles
  - no sliver



A triangulation is Delaunay iff for each edge the circumcircle of an adjacent triangle does not contain the opposite vertex

Among all possible triangulations, the Delaunay triangulation maximizes the smallest angle

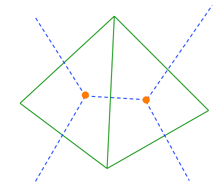
Rhymes with "baloney"

[Hart,Bischoff&Kobbelt]

Dual Meshes and Voronoi Diagram

The dual of a mesh exchanges its faces and vertices

- place new vertices at centroid of faces

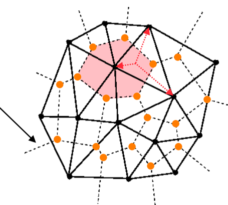


Edges in dual cross original edges

- at right angles
- at midpoints
- (but might not actually "cross" if new vertex is at original edge)

Dual of a Delaunay Triangulation is a Voronoi Diagram

- a Voronoi face denotes a region closer to a given Delaunay (original) vertex than to any other Delaunay vertex



Hart,van Laerhoven

Mesh Simplification

More polygons increases model accuracy, but requires more space and processing/rendering time

- need for accuracy depends on the application
  - game vs. medical imaging
  - approximate solutions vs. final simulations
- screen resolution or viewing distance may not call for a very accurate model

Acquisition systems (e.g., 3D scanner) often produce huge models

- more detailed than necessary
- millions of polygons per object are common
- billions of polygons per object are starting to happen
  - 300 million faces = 3.7 GB after gzip compression

Yu

Mesh Simplification

Reduce polygon count:

- less storage
- faster rendering
- simpler manipulation

Simplification algorithms:

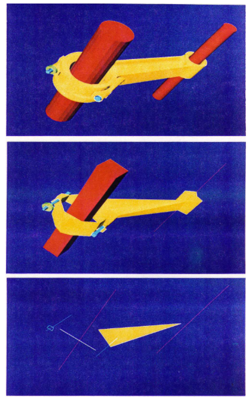
- vertex clustering
- mesh retiling
- mesh decimation
- mesh optimization

Desired properties:

- generality in types of mesh
- efficiency and scalability
- quality of approximation
  - visual
  - geometric: topological modifications
- control of approximation quality
  - continuous LoD
  - smooth transitions between models

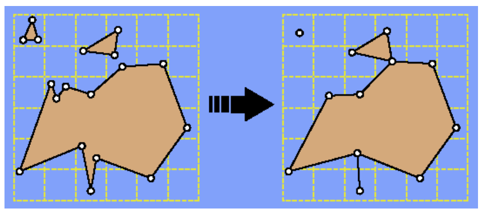
Funkhouser

Vertex Clustering



Method:
- partition space into cells
  - grids, spheres, octrees, ...
- merge all vertices within the same cell
  - triangles with multiple vertices in one cell degenerate into lines or points

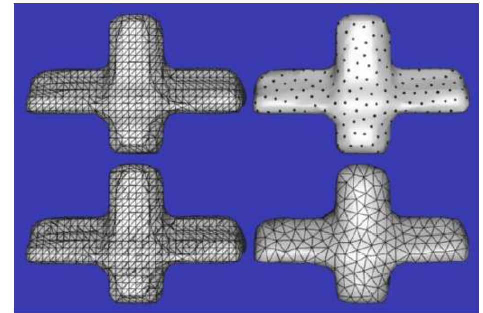
Properties:
- general and robust
- allows topological changes
- not best quality



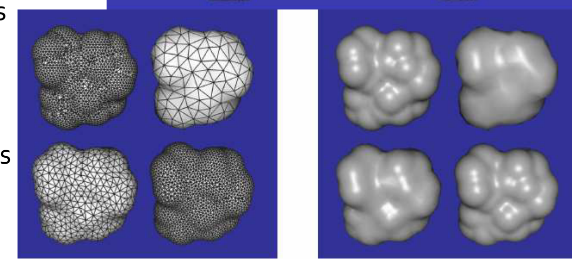
Hart,Funkhouser

Mesh Retiling

Resample mesh with "uniformly spaced" random vertices:
- generate random vertices on surface
- spread them uniformly using diffusion/repulsion
- triangulate vertices



Properties:
- slow
- blurs sharp features



Funkhouser, Turk

Mesh Decimation

Simplification algorithm:
- each operation simplifies the model by a small amount
- apply many operations in an iterative, greedy fashion to gradually reduce complexity of mesh:
  1. measure error introduced by potential decimation operations
  2. place operations in a priority queue sorted by error
  3. perform operations in queue successively
  4. after each operation, re-evaluate error metrics

Types of operations:
- vertex remove
- edge collapse
- vertex cluster (virtual edge collapse)

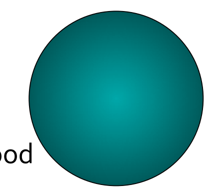
First, some topological properties of meshes . . .

TP3,Funkhouser

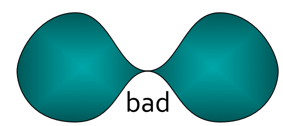
Manifold Surface

Let M be a surface in 3D

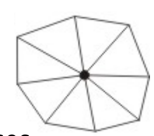
M is a manifold iff a neighborhood of any point $\mathbf{p}$ in M is topologically equivalent to the unit open disk (interior of a unit circle)



Topological equivalence (or homeomorphism) allows deformation that does not rip, tear, or poke holes



On a manifold surface:
- every edge is shared by exactly 2 faces
- around each vertex exists a closed loop of faces



Hart,TP3

Manifolds with Boundary

M' is a manifold with boundary iff a neighborhood of any point $\mathbf{p}$ in M' is homeomorphic to a half disk

On a manifold with a boundary:
- edges on the boundary belong to exactly one face
- around vertices on the boundary the loop of faces is open

a manifold surface with boundary

A 3D surface that is a manifold surface without boundary is a closed surface

TP3

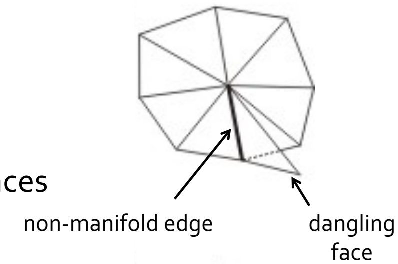
Piecewise Linear Manifolds

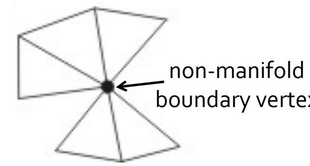
Closed mesh

No dangling faces

Edges only bound two faces

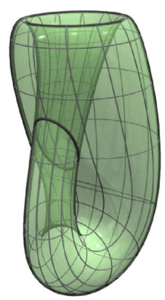
Watertight
- no holes in the surface
- no boundary
- orientable





Hart,TP3

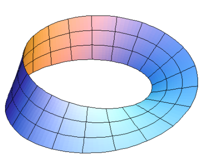
Orientable Surface

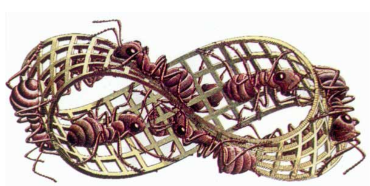


Orientable surface has 2 sides, like a piece of paper

By convention, the normal of a closed orientable surface points "outwards"

The Möbius strip and Klein bottle are non-orientable surfaces:

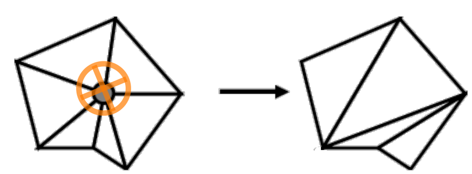


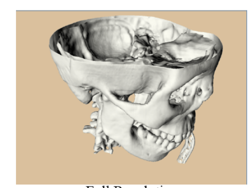


TP3,EscherKarcher,Chu-Carroll,

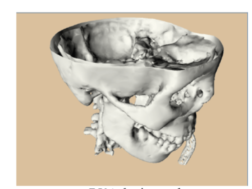
Vertex Remove

Method
- remove vertex and adjacent faces
- fill hole with new triangles (2 less triangles)





Full Resolution
(569K Gouraud shaded triangles)



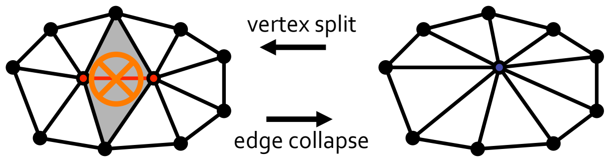
75% decimated
(142K Gouraud shaded triangles)

Properties
- requires manifold surface around vertex
- preserves local topological structure
- filing hole well may not be easy

Hart,Funkhouser,Ramamoothi

Edge Collapse

Method
- merge two edge vertices into one
- delete two degenerate triangles

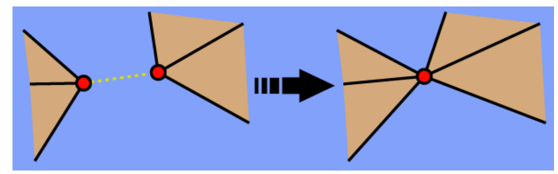


Properties
- requires manifold surface around vertex
- preserves local topological structure
- allows smooth transition

Hart,Funkhouser

Virtual Edge Collapse

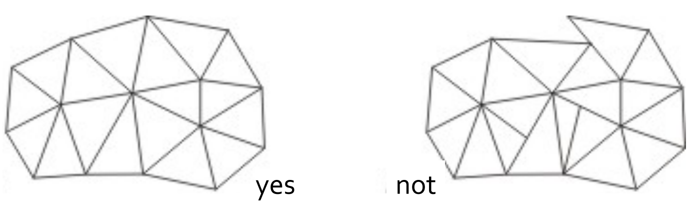
A.k.a. vertex-pair contraction or vertex cluster
- joins previously unconnected areas
- allows topological simplification
- usually limited to small distance to avoid $O(N^2)$ virtual edges
- not best quality



Hart,Manocha

Simplicial Complex Surface

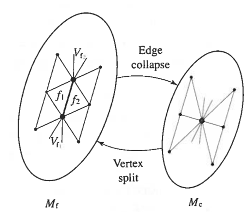
A surface is simplicial complex iff
- polygons meet only along their edges
- edges intersect only at their endpoints

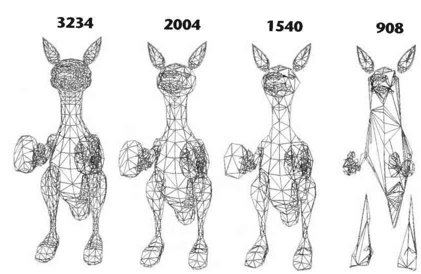


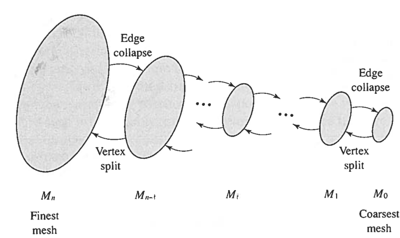
TP3

Iterative Edge Collapse

Properties
- well-defined for any simplicial complex surfaces
- induces hierarchy on the surface: allows smooth transition (geomorph)
- currently most popular technique (standard feature in Direct3D)







Hart,Funkhouser,Watt00

Error Metrics for Simplification

Used to rank edges during simplification
- reflects amount of geometric error introduced
- main differentiating feature among algorithms

Must address two interrelated problems
- what is the best contraction to perform?
- what is the best position $\mathbf{v}'$ for remaining vertex?
  - can just choose one of the endpoints
  - but can often do better by optimizing position of $\mathbf{v}'$

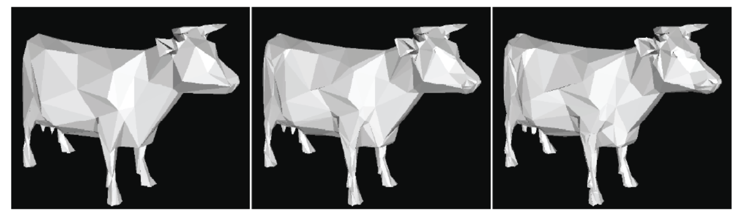
See TP3 §6.5 for details ☺

Hart

Level of Detail (LoD) Pyramid

Single resolution not enough
- application context dictates required detail
- context varies over time (and space)

Level of detail:
- replace each object in the scene graph with a hierarchy of objects at differing resolutions
- choose the model appropriate for current context: collision detection vs. rendering, frame rate vs. quality



RTR,

Discrete LoD

Given a model, build a set of approximations
- can be produced by any simplification system
- at run time, simply select which to render
  - fairly efficient
    - storage required < 2x original
    - cost of changing level of detail while rendering not significant
  - image pyramids (mip-maps) a good example

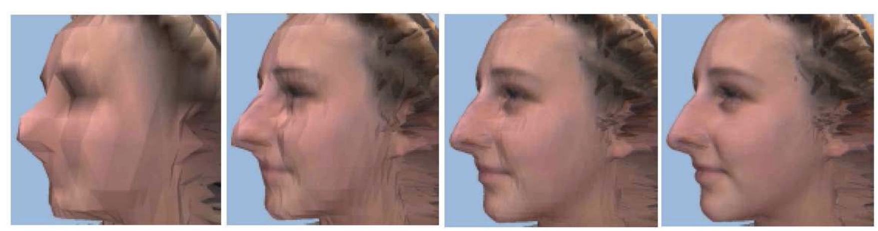
Inter-frame switching causes “popping”
- can smooth transition with image blending
- or geometry blending (requires continuous LoD)

Supported by several scenegraphs:
- RenderMan, Open Inventor, IRIS Performer, …

Hart

Continuous LoD

Need for multi-resolution meshes:
- to reduce “popping” when switching models, geometric morph between two LoDs
- sometimes cannot choose a single LOD, may need different amounts of details on the same surface (see next slide)
- progressive transmission (detail increases over time)

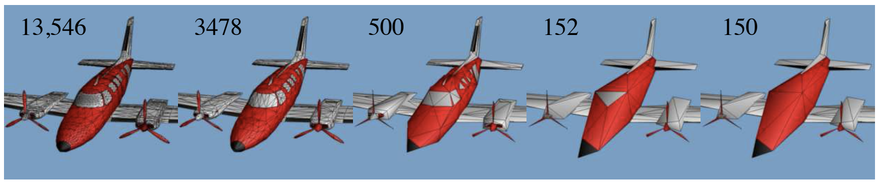


Funkhouser,Certain et al.

Progressive Meshing

Iteratively decimate a mesh using edge collapse
Store the inverse vertex split for each collapse
The most simplified mesh (base mesh) and vertex split records form the progressive mesh:

$$\hat{M} = M^n \xrightarrow{ecol_{n-1}} \dots \to M^{175} \to \dots \xrightarrow{ecol_i} M^1 \xrightarrow{ecol_0} M^0$$



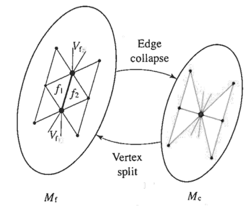
$$M^n \xleftarrow[vspl_{n-1}]{} \dots \leftarrow M^i \xleftarrow[vspl_{i-1}]{} \dots \leftarrow M^{175} \leftarrow \dots \leftarrow M^1 \xleftarrow[vpsl_0]{} M^0 = \hat{M}$$

Hoppe96,Manocha,Funkhouse

Progressive Meshing

Rather than a few discrete LODs we have a full range
- vertex split does not require much storage
- the meshes are flexible and easily reversible
  - requires original positions to be kept with each edge collapse

Support for selective refinement
- requires more info on adjacent vertices and faces of each collapsed edge



Can geomorph smoothly between LODs
- minimizes "popping"

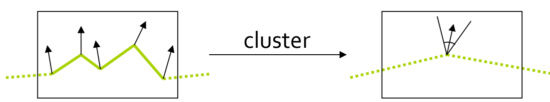
Manocha,TP3

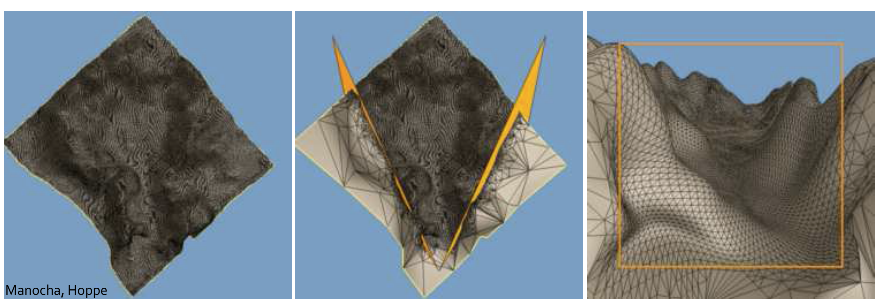
View-Dependent Simplification

Commonly used for terrain generation

Terrain close to viewer is shown with a greater LoD

Preserve:
- silhouette
- specular highlights





Manocha, Hoppe

Polygonal Mesh

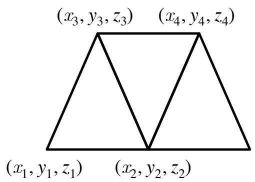
Introduction to five topics:

1. how to draw good looking meshes?
2. mesh simplification
3. level of detail
4. mesh representation
5. error checking and mesh processing

Polygonal Mesh Representation

How would you represent a polygonal mesh?

Three alternatives:
1. explicit mesh (face list)
2. vertex list
3. edge list



Important properties:
- efficient traversal of topology (for drawing, e.g.)
- efficient use of memory (compactness)
- efficient updates (UI, vertex removal, computing per-vertex normals)

Polygonal Mesh Representation

Extension: more topology information: in addition to a list of vertices, each face also
- points to its adjacent faces
- and for each adjacent face, the index of the shared edge

Especially convenient for subdivision and multiresolution hierarchies

[Zorin, Bischoff&Kobbelt]

Polygonal Mesh Representation

Adjacency operations important in mesh simplification and many other applications:
- given face, find its vertices
- given vertex, find faces touching it
- given face, find neighboring faces
- given vertex, find neighboring vertices
- given edge, find vertices and faces it touches

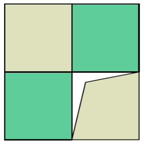
Polygonal Mesh Representation

Explicit mesh or "polygonal soup" model: a list of polygonal faces

Example: $P = \{P_1, P_2, P_3, \ldots, P_n\}$

$$P_1 = ((x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)),$$
$$P_2 = ((x_3, y_3, z_3), (x_2, y_2, z_2), (x_4, y_4, z_4))$$

Problems:
- shared vertices are duplicated
- no topology information (e.g., which vertices and edges are shared), must search the whole list to modify a vertex
- round off errors: cracks and failure to match vertices



Polygonal Mesh Representation

Vertex list or "indexed face set":

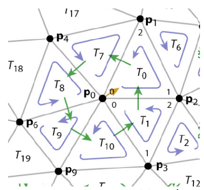
Example: $V = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4)\}$,

$P_1 = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3), P_2 = (\mathbf{v}_3, \mathbf{v}_2, \mathbf{v}_4)$

Used in 3dsmax, X3D OBJ file format

Characteristics:
- shared vertices are stored only once
- but still no topology information: finding shared edges still requires a search

Extension: triangle neighbor list:
- vertex points to a single neighboring triangle
- triangle points to its three neighboring triangles
- can enumerate triangles around a vertex



Marschner

Polygonal Mesh Representation

Edge list:

Example: $V = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4)\}$,

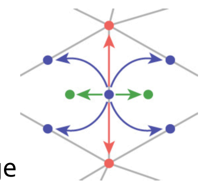
$E_1 = (\mathbf{v}_1, \mathbf{v}_2, P_1, \emptyset)$,

$E_2 = (\mathbf{v}_2, \mathbf{v}_3, P_1, P_2)$,

$E_3 = (\mathbf{v}_3, \mathbf{v}_1, P_1, \emptyset)$,

$P_1 = (E_1, E_2, E_3), P_2 = (E_2, E_4, E_5)$

Extension "winged edge":
- each edge points to:
  - two endpoint vertices
  - two faces that share edge
  - four edges emanating from its endpoints
- faces, vertices contain pointer to one edge



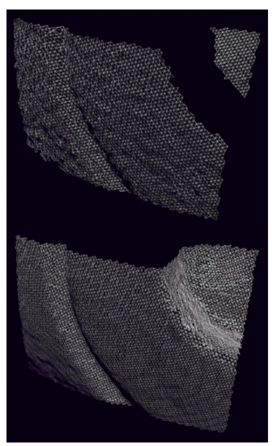
Marschner

Polygon Mesh Error Checking

Polygon mesh can arrive in error due to human errors or scanner limitations

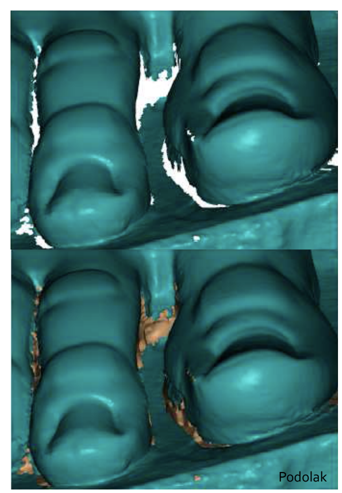
Things to check:
- every vertex is an end point to at least two edges
- every edge is part of at least one polygon
- every polygon is closed
- every polygon has at least one shared edge

Other checks:
- vertex normal
- normal to the plane
- plane's implicit function
- convex or not
- holes? surface water tight?

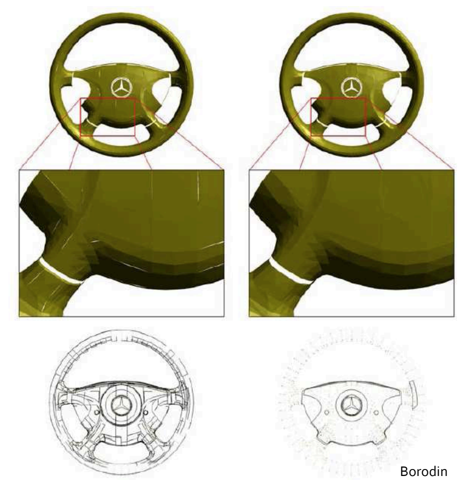


Polygon Mesh Processing

Topological fixups: fix holes, cracks, self-intersection

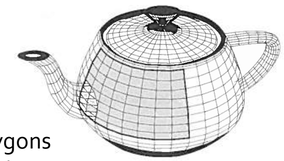


Podolak



Borodin
Funkhouser

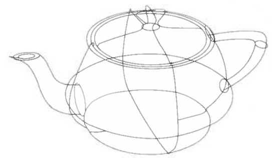
The Problems with Polygons



Not a very compact representation

- needs a lot of flat elements to represent smooth or highly detailed surfaces
- accuracy: exactness of representation can only be approximated by increasing the number of polygons
  - if image is enlarged, planar surfaces again become obvious

Intersection test? Inside/outside test?



Hard to edit

- creating polygonal objects is straightforward … though laborious and tedious
- how do you edit a polygonal-mesh surface?
  - don't want to move individual vertices …
- difficult to deform object: a region of low curvature, represented with low polygon count, cannot be deformed into a high curvature region
- it is more a machine representation than a convenient user representation