
An Introduction to Geometric Modeling using Polygonal Meshes

Joaquim Madeira

April 2024

Overview

- Motivation
- Polygonal meshes
- Exact vs. Approximate representation
- Geometrical and topological information
- Valid vs. non-valid models
- The Euler formula
- Computational representation

COMPUTER GRAPHICS & GEOMETRIC MODELING

CG is not alone...

- Core areas:

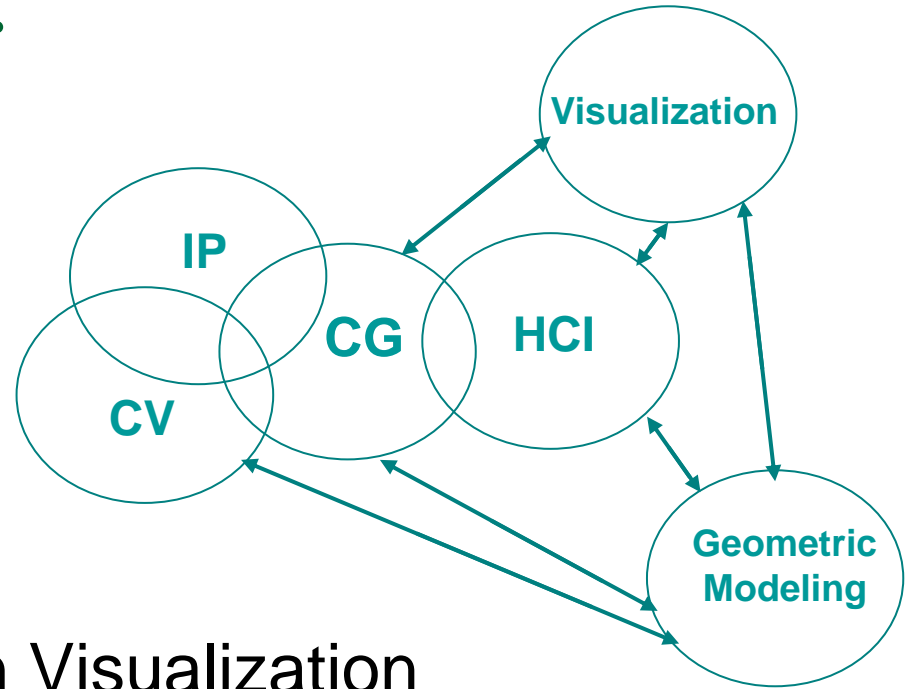
- CG, IP, CV and HCI

- Satellite areas:

- Geometric Modeling
- Data and Information Visualization

- What is common?

- CG, IP : image file formats, color models, ...
- CG, CV : 3D model representations, ...
- IP, CV : noise removal, filters, ...



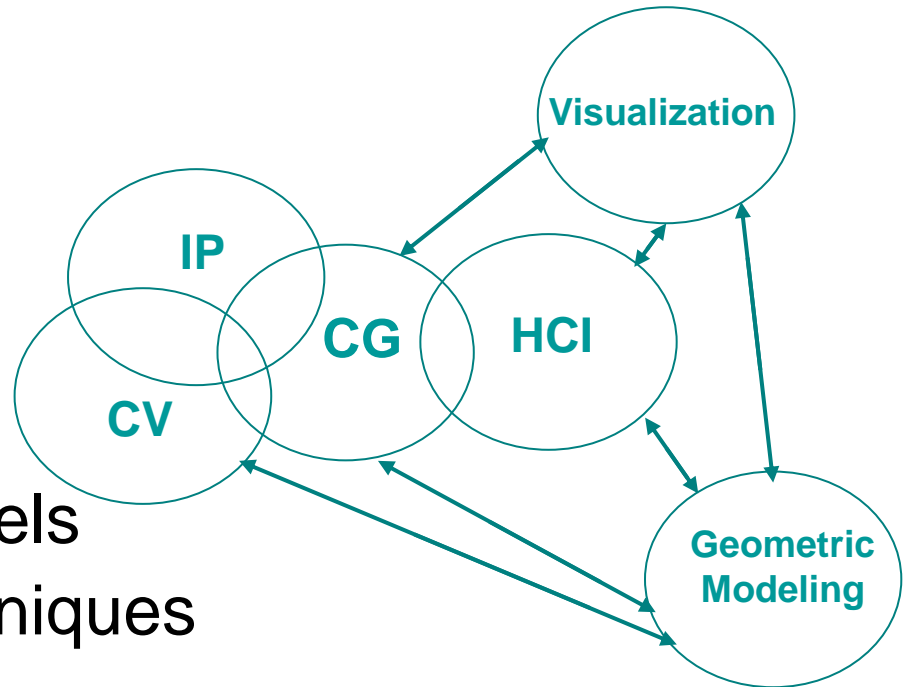
CG is not alone...

■ Geometric Modeling

- ❑ CV : 3D scanning
- ❑ CG : 2D and 3D models
- ❑ HCI : interaction techniques

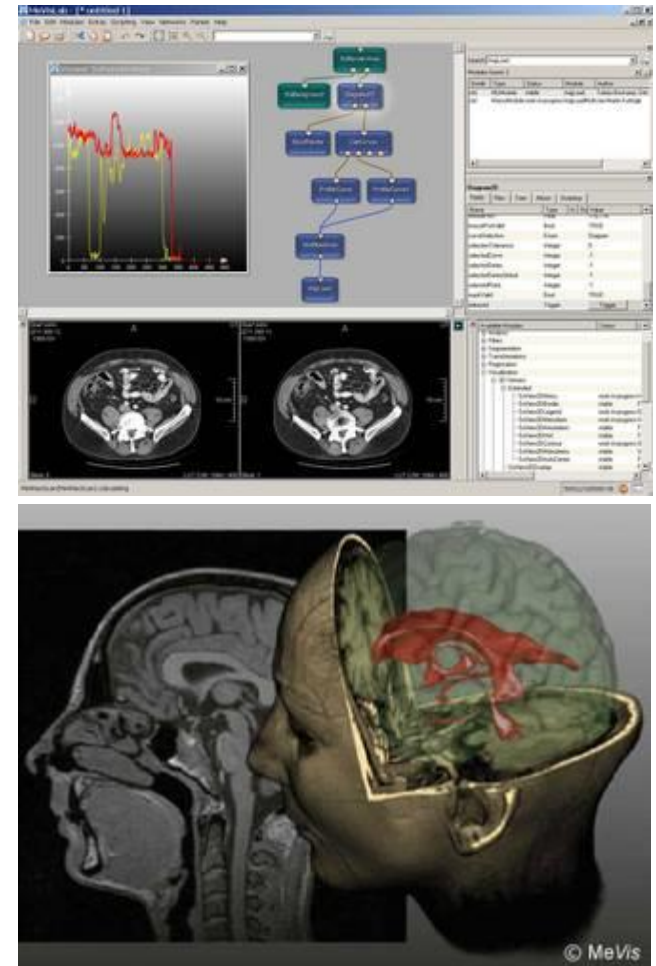
■ Visualization

- ❑ HCI : interaction techniques
- ❑ GeoM : 2D and 3D models
- ❑ CG : rendering



Example – Medical Imaging

- Processing pipeline
 - Noise removal
 - Segmentation
 - Generating 2D / 3D models
 - Data visualization
 - User interaction
 - ...



[www.mevislab.de]

CG Main Tasks

■ Modeling

- ❑ Construct individual models / objects
- ❑ Assemble them into a 2D or 3D scene



■ Animation

- ❑ Static vs. dynamic scenes
- ❑ Movement and / or deformation

■ Rendering

- ❑ Generate final images
- ❑ Where is the observer?
- ❑ How is he / she looking at the scene?

Geometric Modeling

- A **geometric model** describes the **shape** of an (real or virtual) object
- How ?
 - ❑ Different mathematical representations ?
 - ❑ Data structures ?
 - ❑ Possible operations ?
 - ❑ Compactness ? Robustness ? Efficiency ?
 - ❑ Interpolation vs Approximation ?
 - ❑ ...

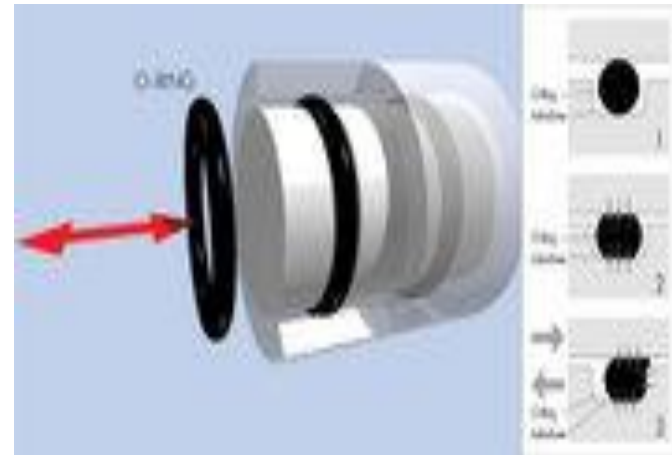
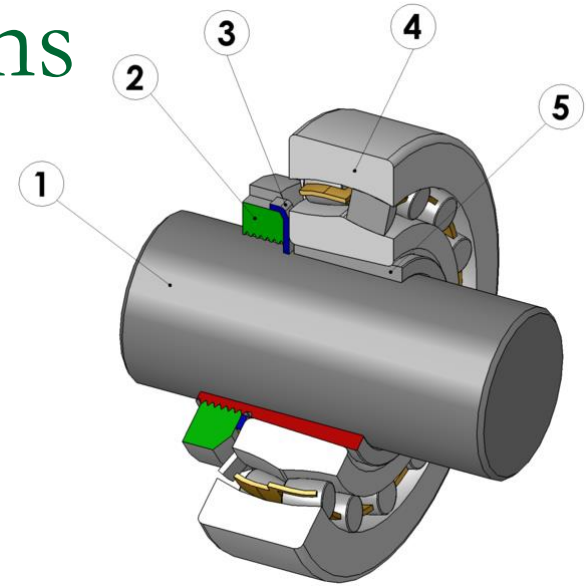
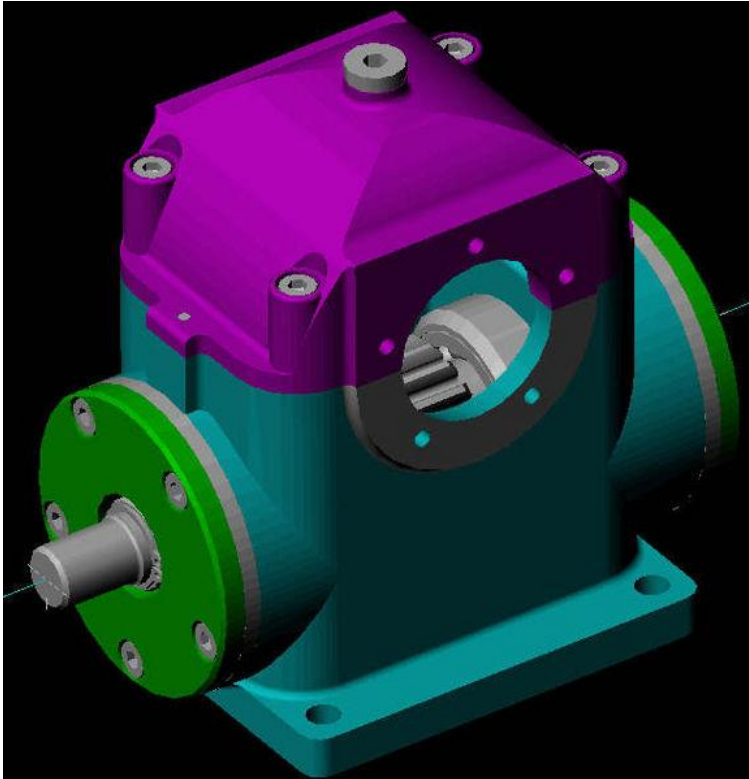
Geometric Modeling

■ What for ?

- ❑ Distinguish between **inside**, **outside** and **border** of a model
- ❑ Compute **properties**
 - Centroid
 - Area / Volume
 - ...
- ❑ Detect interferences / **collisions**
- ❑ Compute **light reflections** and / or transparencies
- ❑ ...

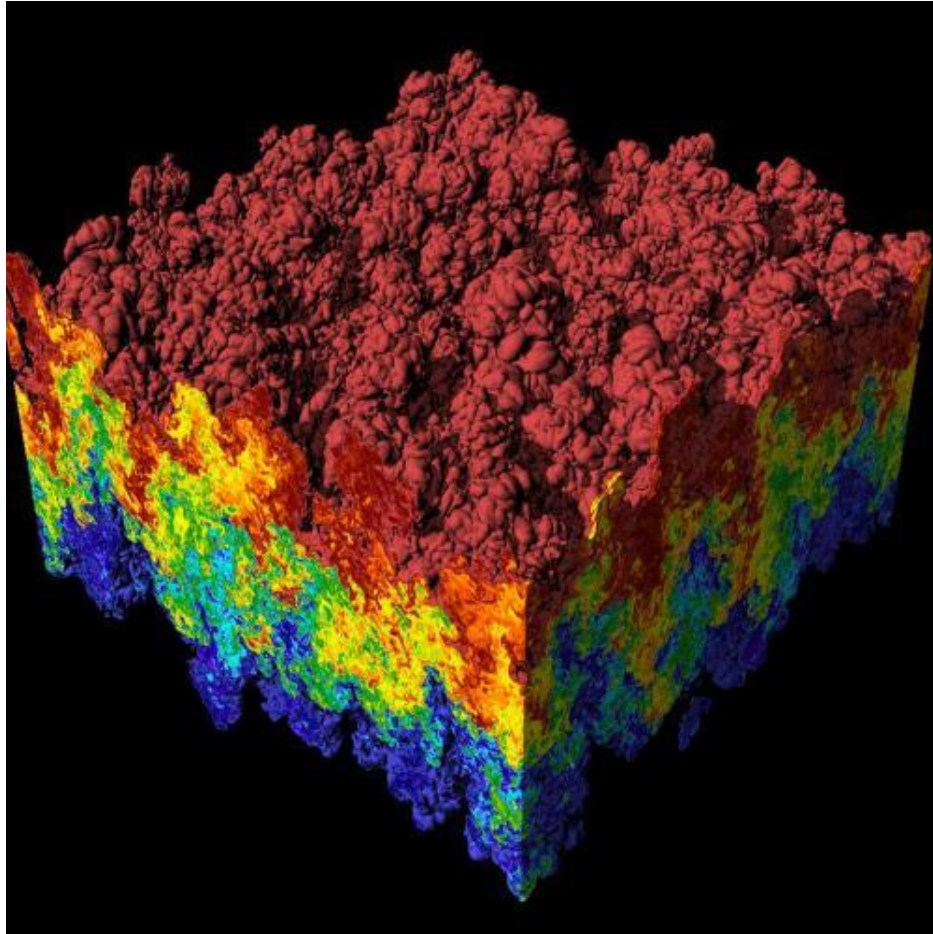
3D models – Applications

■ CAD / CAM



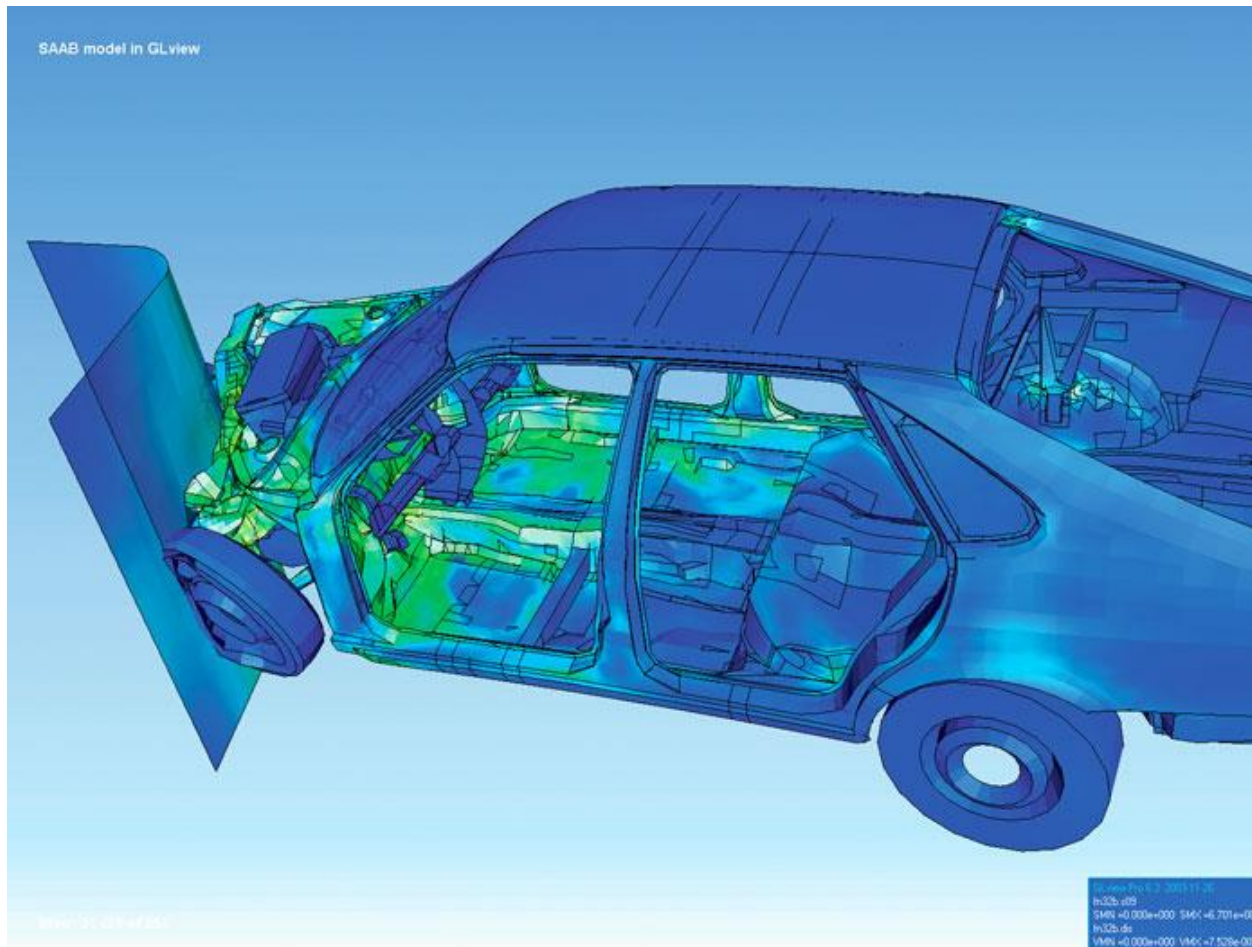
[Wikipedia]

Data Visualization



[Wikipedia]

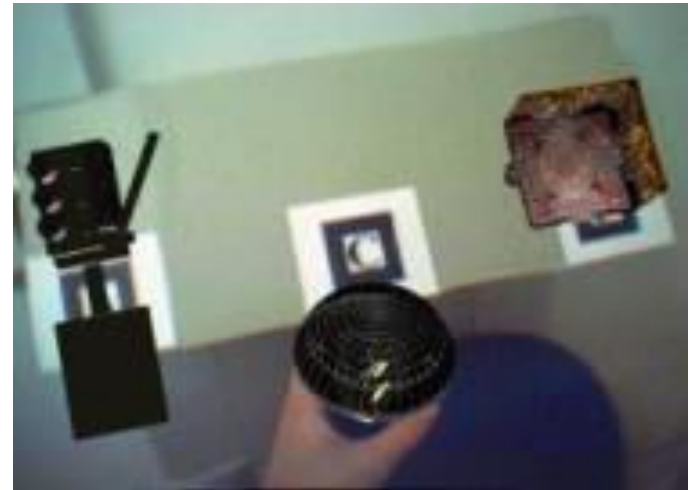
CAD – Simulation and Visualization



[Wikipedia]

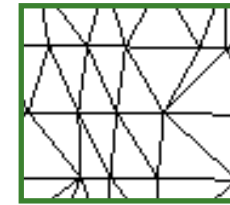
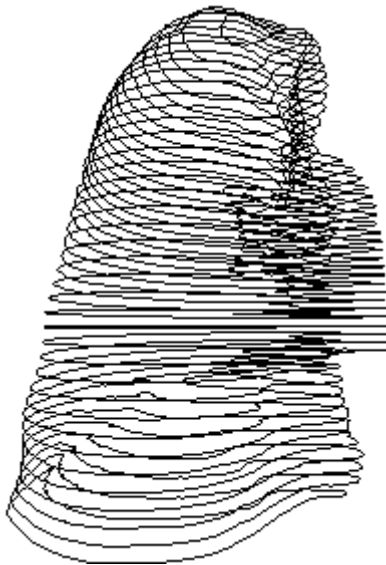
3D models – Applications

- Virtual / Augmented reality



3D models – Applications

■ Medical Data Processing



Mesh detail



Mesh

3D models – Applications

- Other application areas
 - Computer games
 - Geographical information systems (GIS)
 - Engineering analysis
 - 3D printing / Rapid prototyping
 - Medical solid modeling
 - ...

3D models – Shape

- Define **from scratch** using VRML / X3D, OpenGL, VTK, ...
 - ❑ Tedious; requires skill
- Obtain from **CAD files** or **model databases**
 - ❑ Convert to compatible formats
 - ❑ Use of existing models in manufacturing applications
- Create using a **3D digitizer** or a **3D scanner**
 - ❑ 3D digitizer : stylus
 - ❑ 3D scanner : tracker, cameras and laser

3D modeling tools

- Spatial Corp.'s **ACIS**; 3D modeling engine
 - <http://www.spatial.com>
- Siemens's **Parasolid**; 3D modeling engine
 - <http://www.plm.automation.siemens.com>
- Dassault Systemes's **CATIA**; CAD / CAM / CAE
 - <http://www.3ds.com>
- PTC's **Pro/ENGINEER**; 3D feature modeling
 - <http://www.ptc.com>
- **SolidWorks**; 3D feature modeling
 - <http://www.solidworks.com>


3D modeling tools

- Autodesk's **3ds max** and **Maya**
 - <http://www.autodesk.com>
- **Blender**: Free open-source 3D content-creation suite
 - <http://www.blender.org>
- **Rhino**: Uninhibited free-form 3D modeling
 - <http://www.rhino3d.com>
- **Trimble SketchUp**: Intuitive 3D modeler
 - <http://www.sketchup.com>
- **POV-Ray**: Persistence of Vision Ray-Tracer
 - <http://www.povray.org>

POLYGONAL MESHES

Geometric Modeling

■ Main areas

- ❑ Curve and surface modeling 
 - Computer-Aided Geometric Design (CAGD)
- ❑ Solid Modeling
- ❑ Volume Modeling

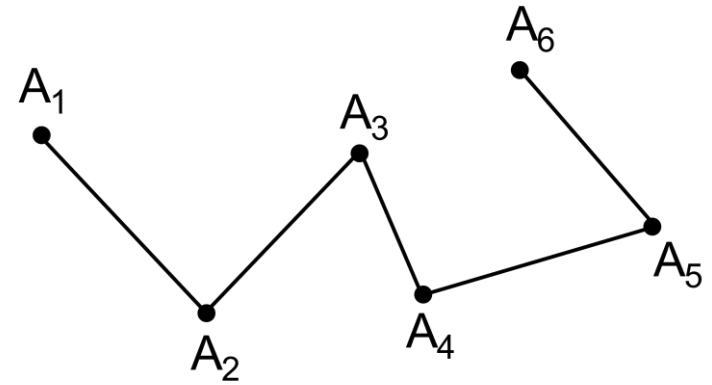
■ Simplest models

- ❑ Curves : Polygonal lines
- ❑ Surfaces : Polygonal meshes 

Polylines

■ Questions :

- ❑ Open or closed polylines ?
- ❑ Exact representation ? When ?
- ❑ Approximate representation ?



[Wikipedia]

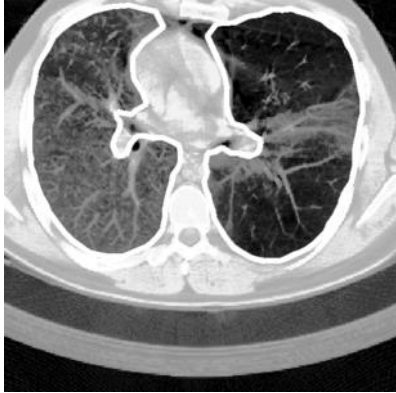
■ A “good” approximation usually needs a larger number of points

- ❑ Level of detail (LOD)

■ Typical application

- ❑ Representing contours in processed images, after locating dominant points

Polylines



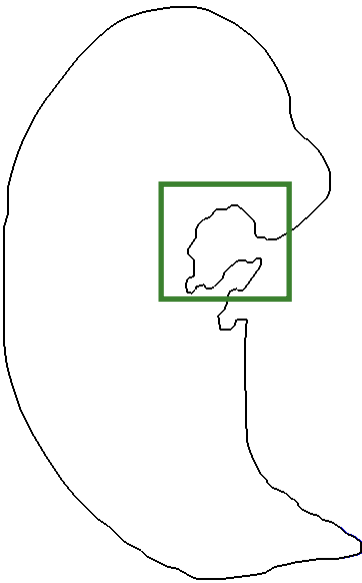
Contours segmented from CT images

Contour description is point by point

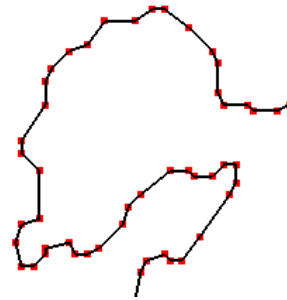


Too many points !

Solution: Dominant Point Detection



Contour
detail



Dominant points

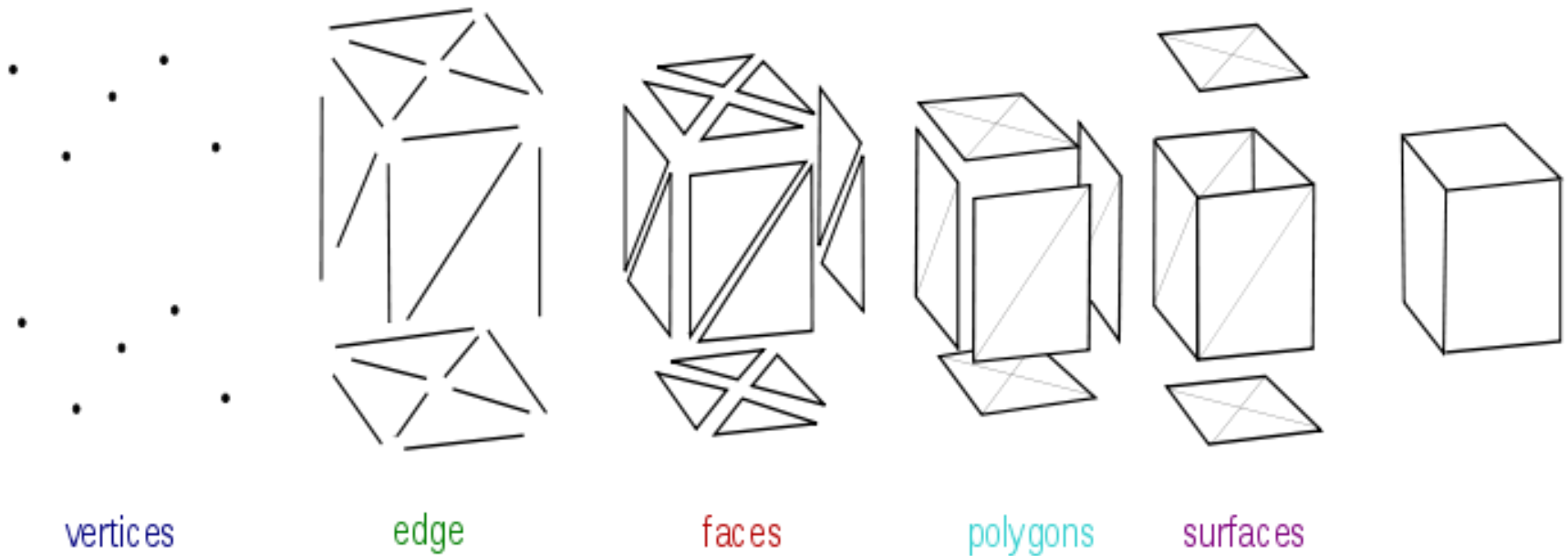


Polyline

Polygonal Meshes

- Surface is defined as a collection of neighboring faces (e.g., triangles)
 - **Geometry** + **Topology** (i.e., connectivity)
 - Vertices, edges, faces
- Euler formula for closed surfaces
 - $V + F - E = 2$
- Exact vs approximate representations
 - Polyhedral models
 - Curved surfaces
 - Terrain models
 - Complex surfaces / models

Polygonal Meshes



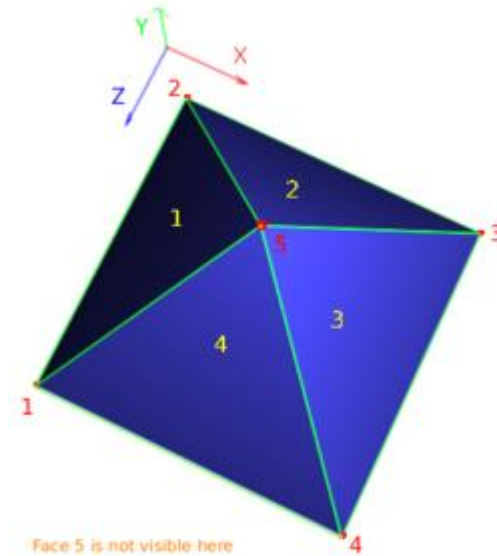
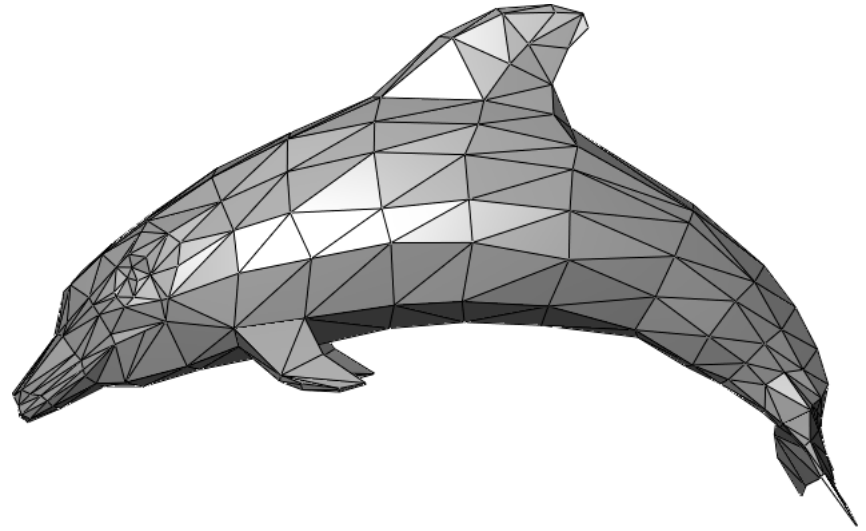
[Wikipedia]

Polygonal Meshes

- Collection of neighboring **vertices**, **edges** and **polygons**
 - Usually **triangles** !!
- Vertex
 - Shared by, at least, 2 edges
- Edge
 - Connects 2 vertices
 - Shared by 2 polygons, if the surface is closed
- Polygon
 - Sequence of, at least, 3 vertices

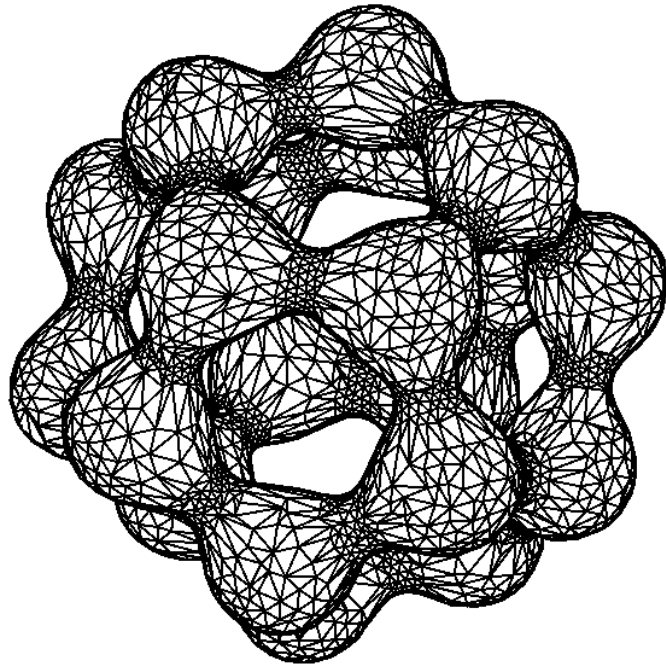
Polygonal Meshes

- Homogeneous ?
- Adaptive ?
- Easy to render
- Usually triangles !
- Pyramid
 - How many entities?
 - Check Euler formula!

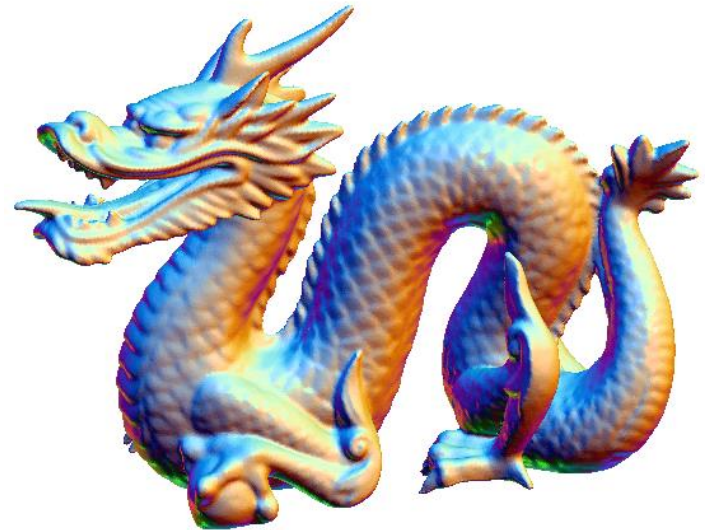


[Wikipedia]

Polygonal Meshes



Complex topology



Complex geometry

[Seidel and Belyaev, 2006]

Polygonal Meshes

- What / How to store?
 - ❑ Memory or file?
 - ❑ List of vertices – Topology ?
 - ❑ List of triangles – Neighbors?
 - ❑ Lists of vertices, edges and triangles – Efficiency?
 - ❑ Winged-edge or half-edge data structure
- Common operations
 - ❑ Smoothing
 - ❑ Decimation
- Toolboxes / Libraries
 - ❑ CGAL
 - ❑ OpenMesh

Polygonal Meshes

- The **surface** (i.e., the model) is defined as a set of adjacent faces (e.g., **triangles**)
- Which **geometric information** should be stored ?
 - Vertex coordinates
- Which **topological information** (i.e., connectivity) should be stored ?
 - How are edges and faces arranged ?
 - How to identify **neighboring / incident / adjacent** entities ?
 - Efficiency !
- Which **additional properties** should be stored ?
 - Normal vector to each face / vertex
 - Texture coordinates
- How to check the **validity** of a model ?
 - 2-manifolds
 - Euler Formulae

Some basic operations

- Find the vertices defining an edge
- Find the edges incident in a vertex
- Find all polygons sharing
 - A vertex
 - An edge
- Identify mesh errors. I.e., the lack of
 - A vertex / an edge / a face
- Rendering a mesh

Polygonal Meshes

- Supported by most applications
- Various file formats
- Triangle meshes are the most common !!
 - Planar faces
 - Algorithm simplicity
 - Numerical robustness
 - Efficient rendering

Polygonal Meshes

- Exact vs. approximate rep. – When ?
 - Polyhedral models
 - Curved surfaces
 - Terrain models
 - More complex models / surfaces
- A “good” approximation might require a large number of faces
 - Levels-of-Detail (LODs)

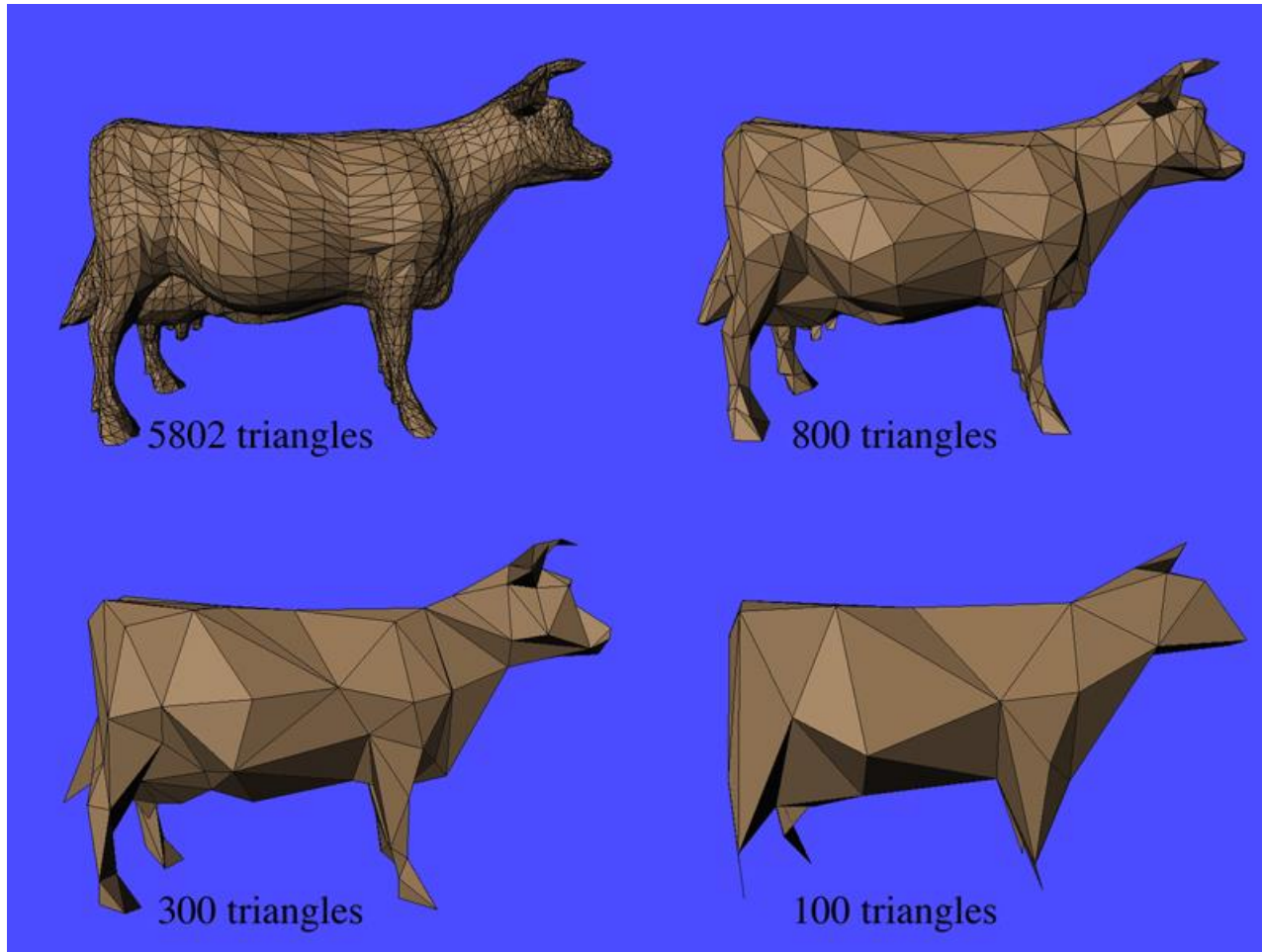
Polyhedral models

- The same polyhedral model might be represented by **different polygonal meshes** !!
 - Useful for shading / rendering
- Degrees of freedom
 - **Number** of mesh vertices
 - **Distribution** of mesh vertices
 - **Arrangement** of edges / polygons
- Example
 - Represent a cube using different polygonal meshes

Curved surfaces

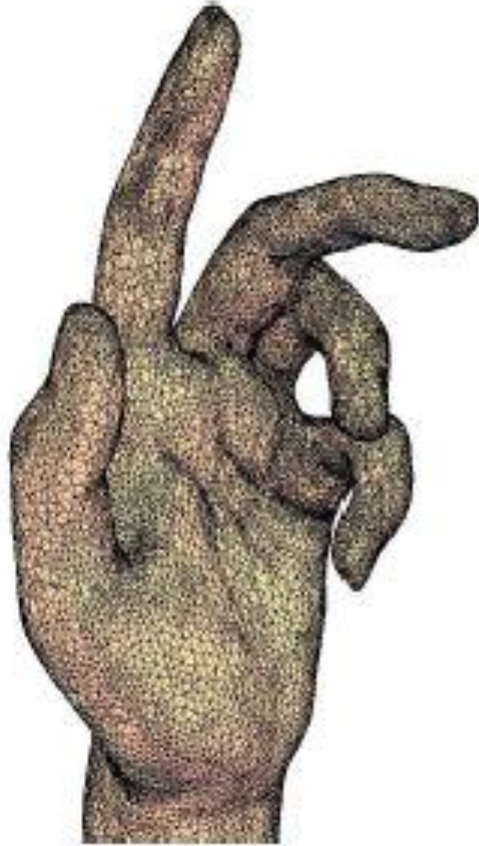
- Representing the shape of a curved surface is an **approximation** process
- There is no “unique” model!!
- Degrees of freedom
 - **Number** of mesh vertices
 - **Distribution** of mesh vertices
 - **Arrangement** of edges / polygons

How many triangles should be used?

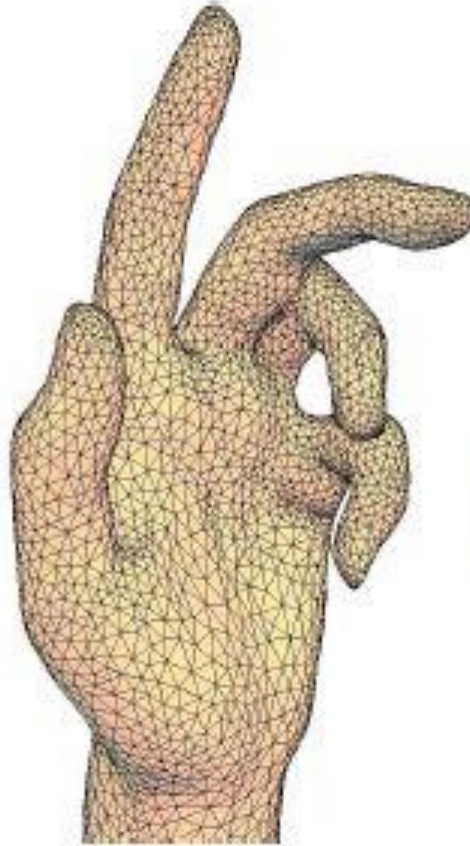


[CMU, 2000]

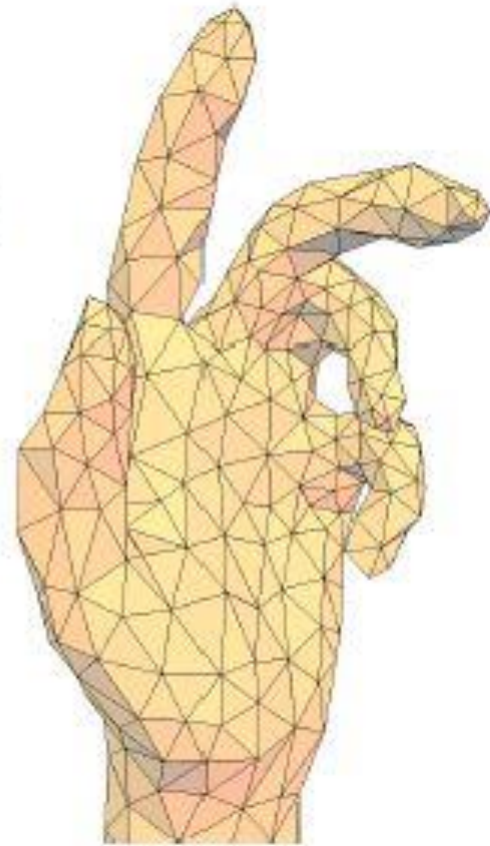
How many vertices should be used?



(a) 25,000 vertices.



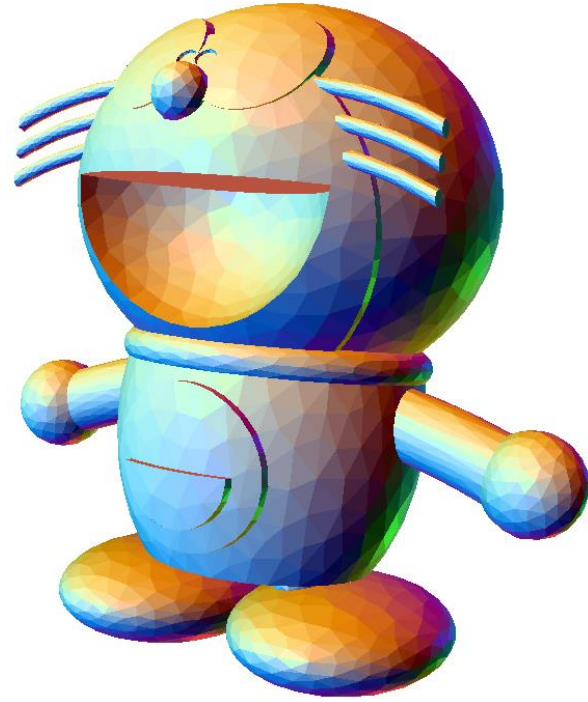
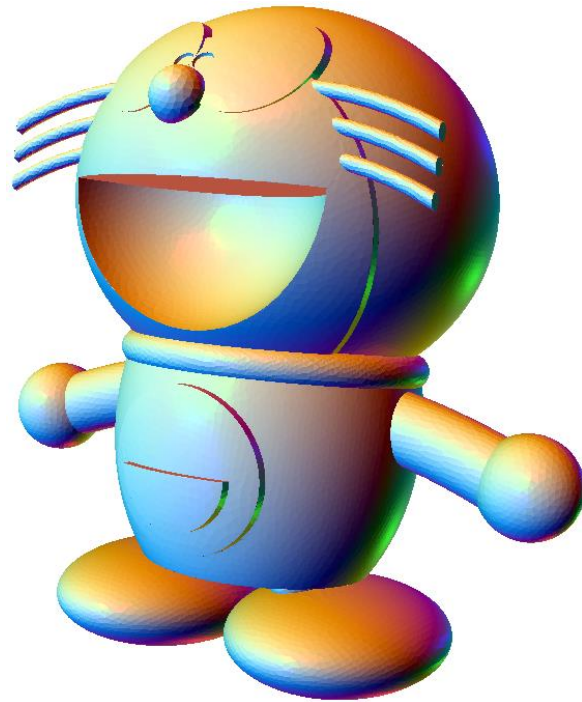
(b) 5,000 vertices.



(c) 500 vertices.

[Dyer et al.]

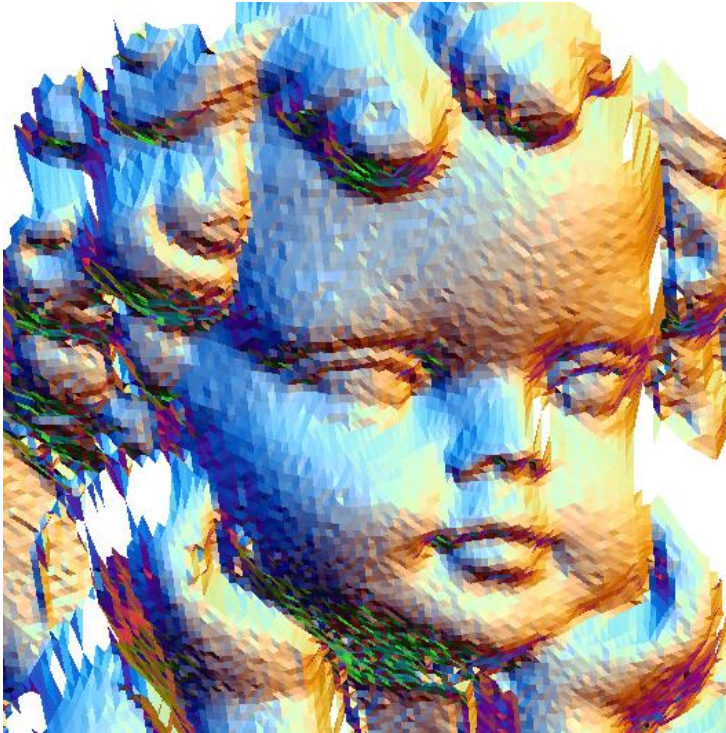
Mesh decimation



**90%
reduction**

[Seidel and Belyaev, 2006]

Mesh smoothing



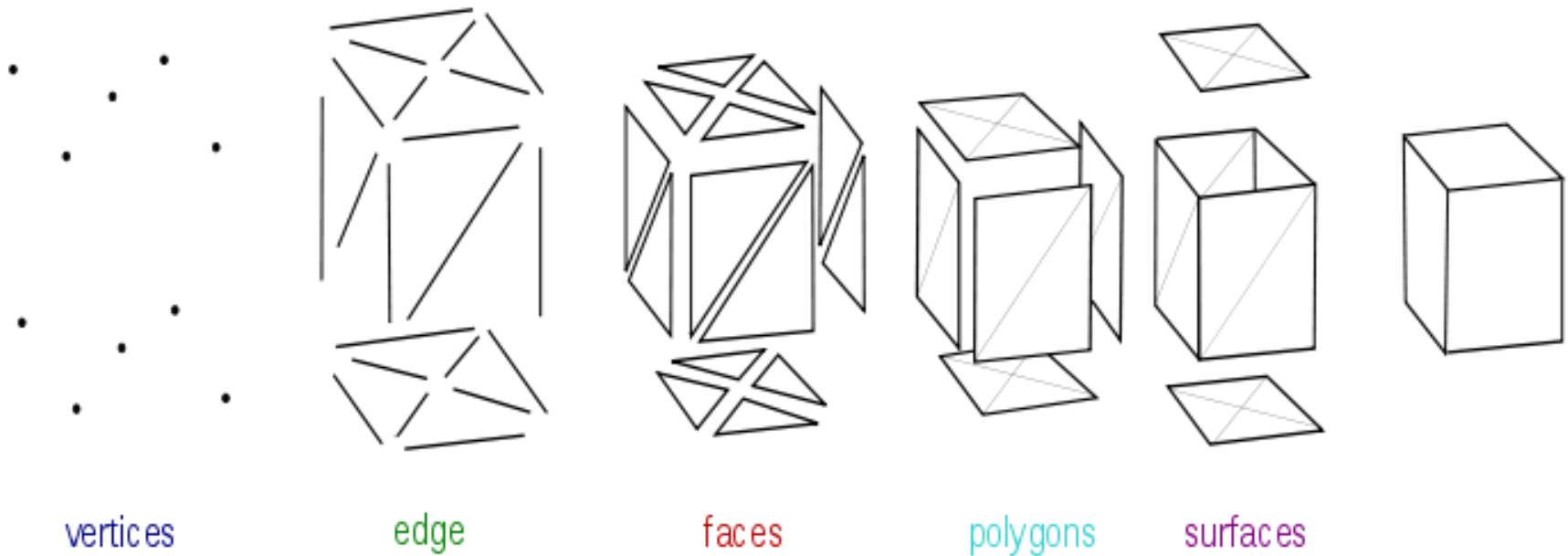
[Seidel and Belyaev, 2006]

GEOMETRY + TOPOLOGY

Representing model surfaces

- Geometrical information
 - Vertex coordinates
- Topological or connectivity information
 - **Abstract definition** of vertices, edges and faces
 - **Incidence** and **adjacency** information
- Properties
 - Normal vectors (“Normal Maps”)
 - Texture coordinates

Topological information



[Wikipedia]

Topological information

■ Vertex

- ❑ Regular ?
- ❑ Singular ?

■ Edge

- ❑ 2 vertices
- ❑ Border edge : just 1 incident face
- ❑ Regular edge : 2 incident faces
- ❑ Singular edge : 3 or more incident faces

■ Loop

- ❑ Ordered edge sequence

Topological information

- Face

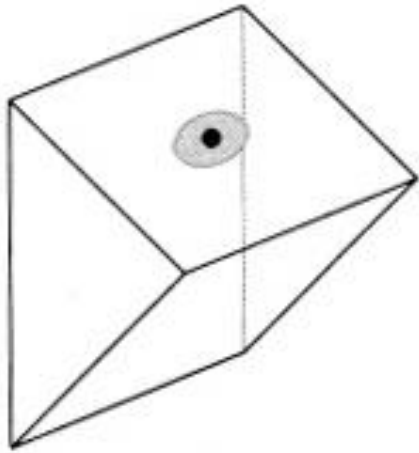
- Limited by a set of disjoint edge sequences
- Outer border
- Possible inner borders (“holes”)

- Shell

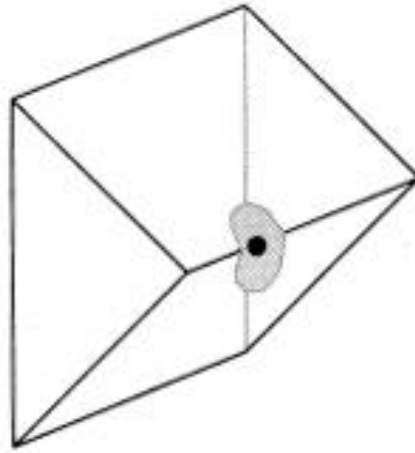
- Set of connected faces

- Examples ?

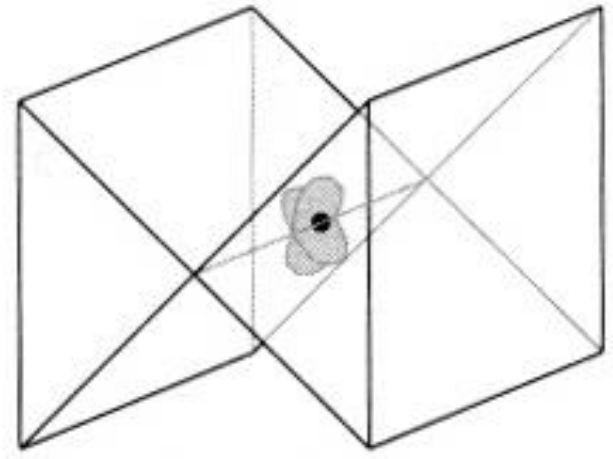
Valid vs. non-valid models



(a)



(b)



(c)

[Foley et al.]

Valid vs. non-valid models

■ 2-Manifold Model

- ❑ Any point has a “disk” neighborhood
- ❑ No singular vertices !!
- ❑ No singular edges!!

■ Non-Manifold Model

- ❑ **Dangling** Edges / Faces
- ❑ **Touching** Faces
- ❑ ...
- ❑ **Non-valid** / non-manufacturable models !!

■ Examples ?

Euler Formula

- Allows checking the consistency of the topological information !!
- $V + F - E = 2$
- When to apply?
 - ❑ Model has a closed, orientable surface !
 - ❑ Each face is limited by a single edge loop !
 - ❑ No through-holes !
 - ❑ Nor cavities !
- Examples
 - ❑ Tetrahedron
 - ❑ Different mesh representations of a cube

Euler-Poincaré Formula

- Generalization !!
- $V + F - E - (L - F) - 2(S - G) = 0$
- L – Number of loops
- S – Number of shells
- G – Genus : number of “handles”
- When to apply ?
 - Through-holes
 - Cavities
- Example ?

Consistency checking

- Check if
 - All polygons have closed borders
 - All edges are used at least once
 - Every vertex belongs at least to
 - 2 edges
 - 1 polygon
 - ...

COMPUTATIONAL REPRESENTATION

Computational representation

- Memory or file ?
- Vertices list
 - Topological information ??
- Polygons list / Detached triangles
 - How to identify neighbors ??
- Vertices, edges and polygons lists
 - Efficiency ?
- **Winged-edge** or **half-edge** data structures

Detached / Isolated Polygons List

- Each polygon is represented by the ordered list of its vertices coordinates
 - CCW
- Inefficient !!
 - Memory space : multiple vertex representation
 - Lack of information about shared vertices / edges
 - Cumbersome detection !!
 - Rendering : edges are drawn twice !!
- Example ?

Vertices List

- Vertices list / array
 - Store just once the coordinates of each vertex !
 - Easy to edit / modify one vertex
- Each polygon is described by its vertices sequence
 - Pointer / index
 - Usage : storing in a file
- **Inefficient !!**
 - Hard to detect which polygons share a given edge !!
 - Rendering : edges are drawn twice !!
- Example ?

Indexed Face Set

- VRML or MCGL or ...
- Array 3D vertex coordinates
 - One index for each vertex
- Convex n-sided polygons defined by n indices
- Example
 - [0,1,2,-1,2,1,3,4,-1]

OBJ File Format

■ Vertices list

v 10 15 20

v 23 34 56

...

■ Faces list

f 1 2 3

f 2 3 4

...

■ Additional information

- Normal vectors
- Texture coordinates
- ...

Topological information

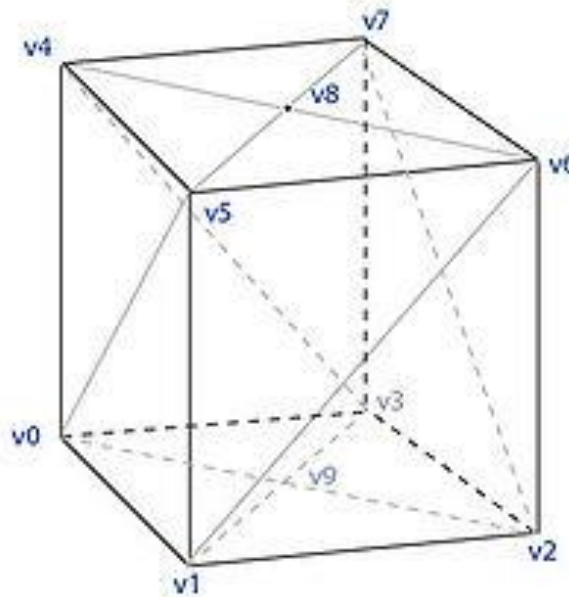
- How to store incidence and adjacency information ?
- How to answer **basic queries** fast ?
 - Which are the **end vertices** of a given edge ?
 - Which are the **adjacent polygons** of a given edge ?
 - Which are the **incident edges** in a given edge ?
 - Which are the **incident edges** in a given vertex?
 - Which are the **neighboring vertices** of a given vertex ?
 - ...
- Efficiency
 - Time ?
 - Space ?

Adjacent Vertices List

Vertex-Vertex Meshes (VV)

Vertex List

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3



[Wikipedia]

Vertices List + Faces List

Face-Vertex Meshes

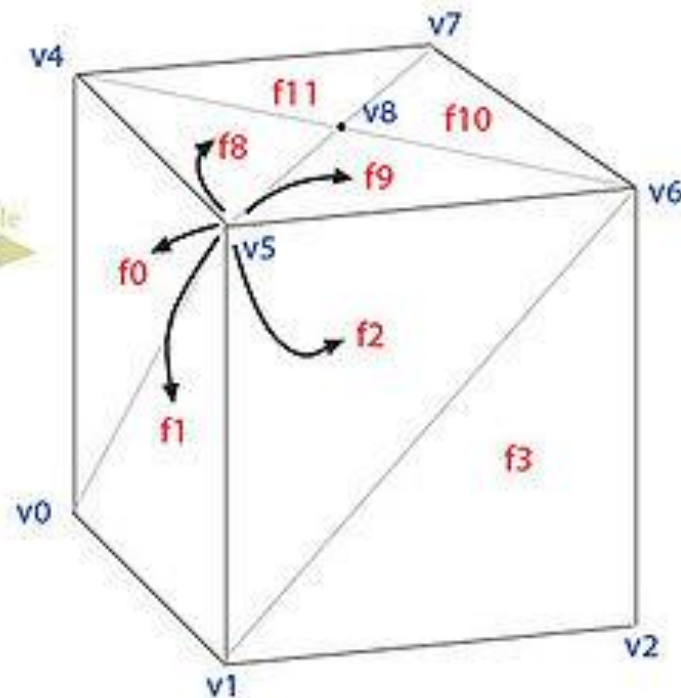
Face List

f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

Vertex List

v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f15 f14 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5,.5,0	f8 f9 f10 f11
v9	.5,.5,1	f12 f13 f14 f15

example →



[Wikipedia]