

02 - Geometric Meshes

Acknowledgements: Daniele Panozzo

80412 - 2024/25 - Geometric Modeling - Enrico Puppo

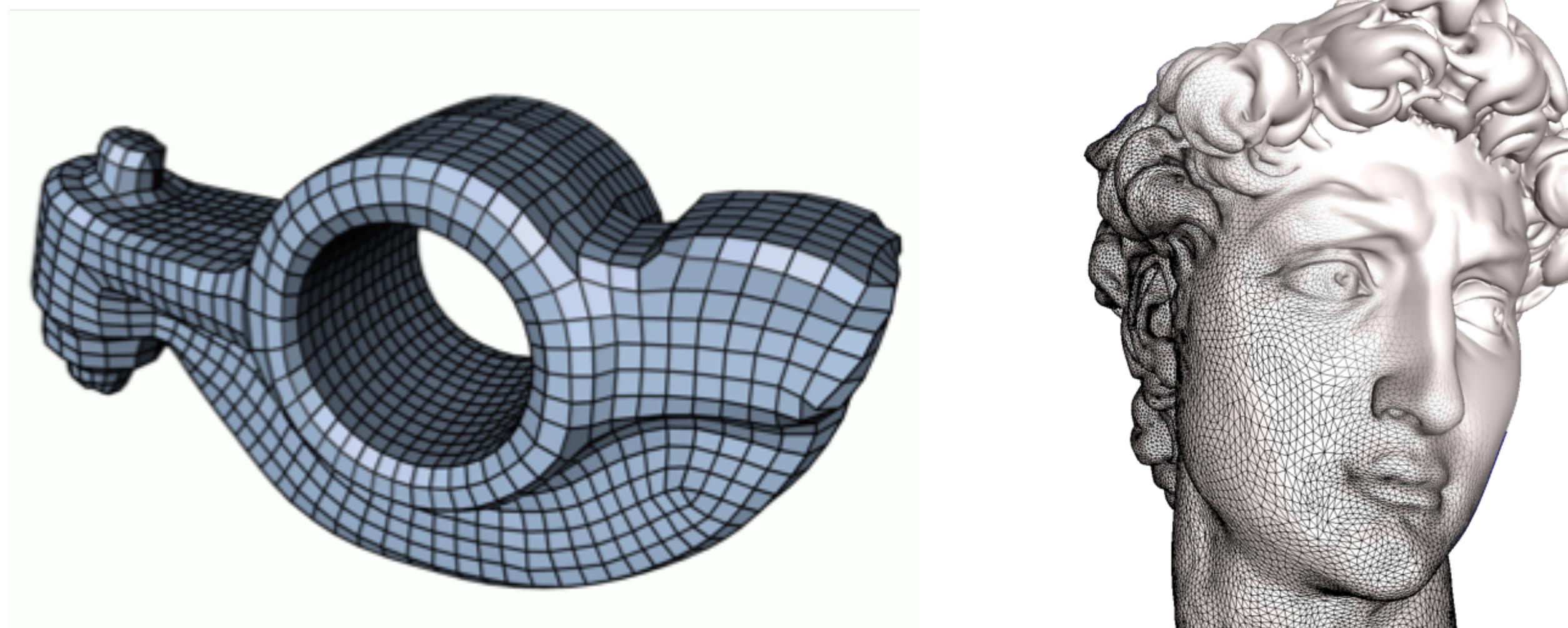
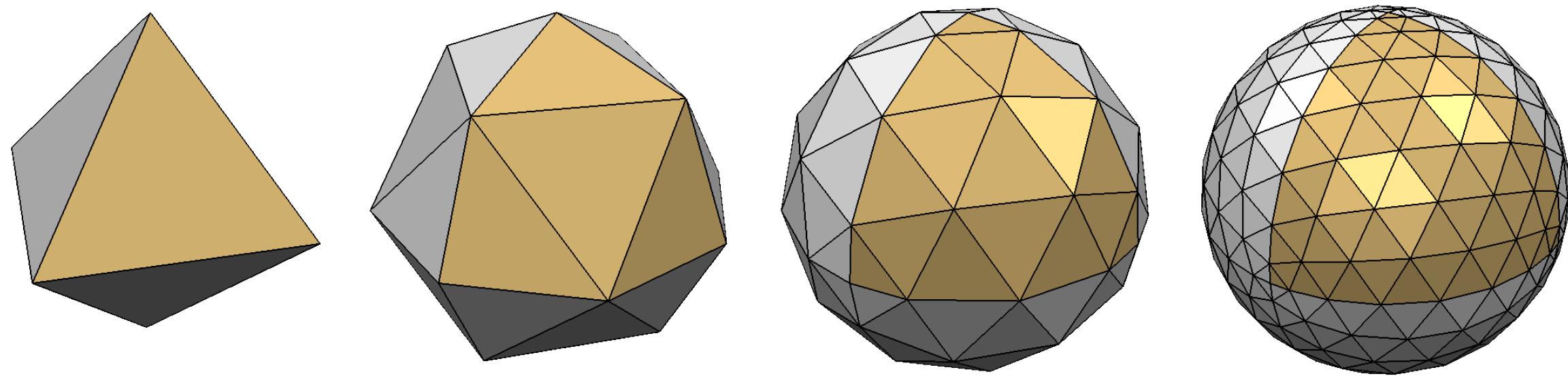
In this lecture

- All you need to know about meshes:
 - what they are
 - how to manipulate
 - how to encode

Polygonal meshes

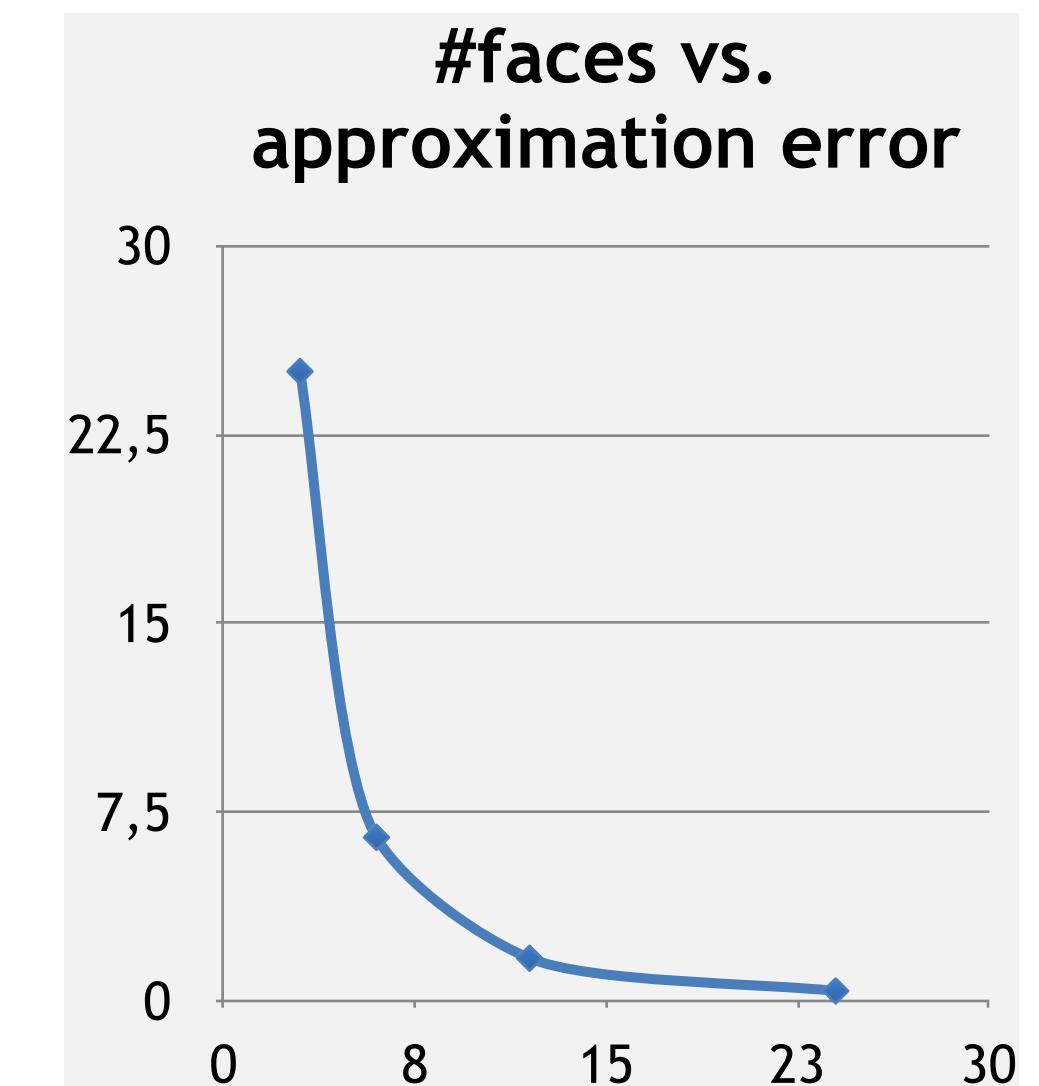
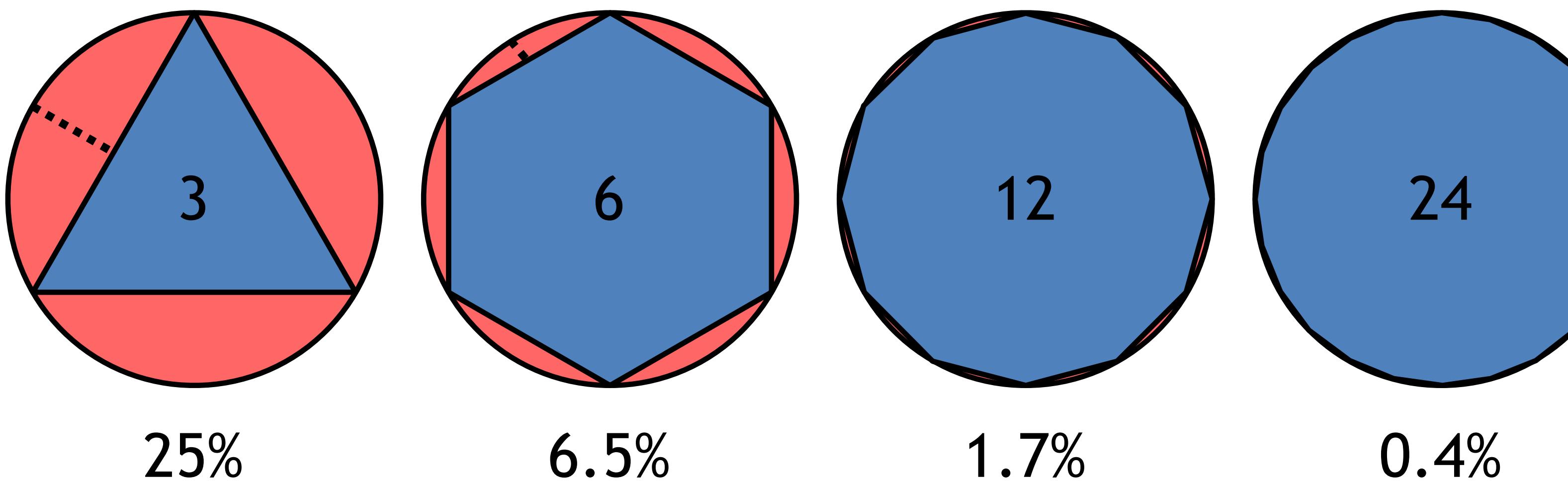
Polygonal Meshes

- Boundary representations of objects



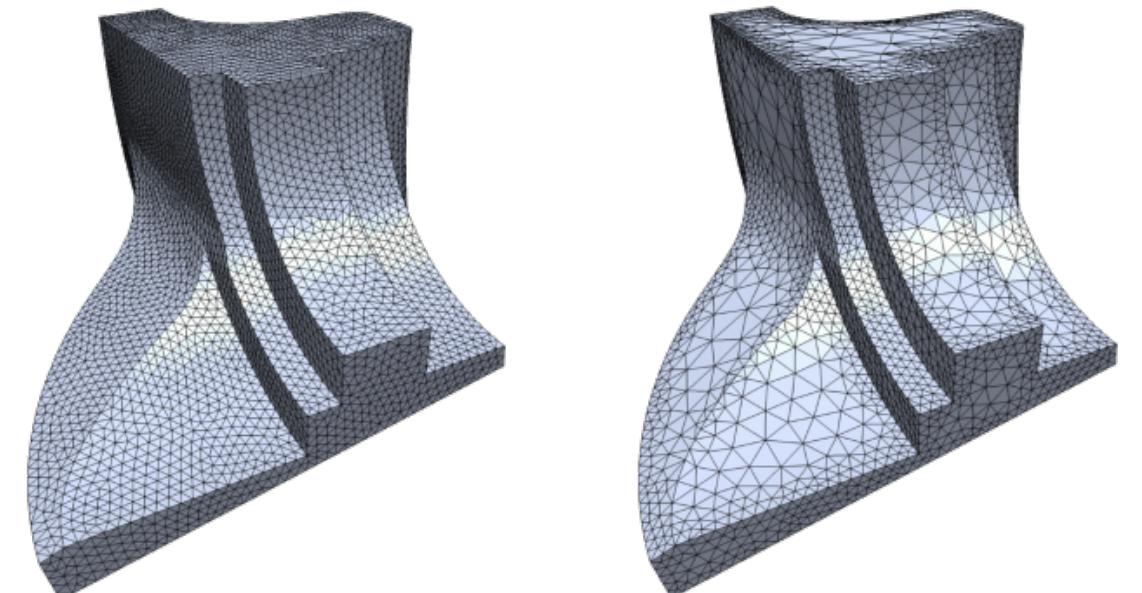
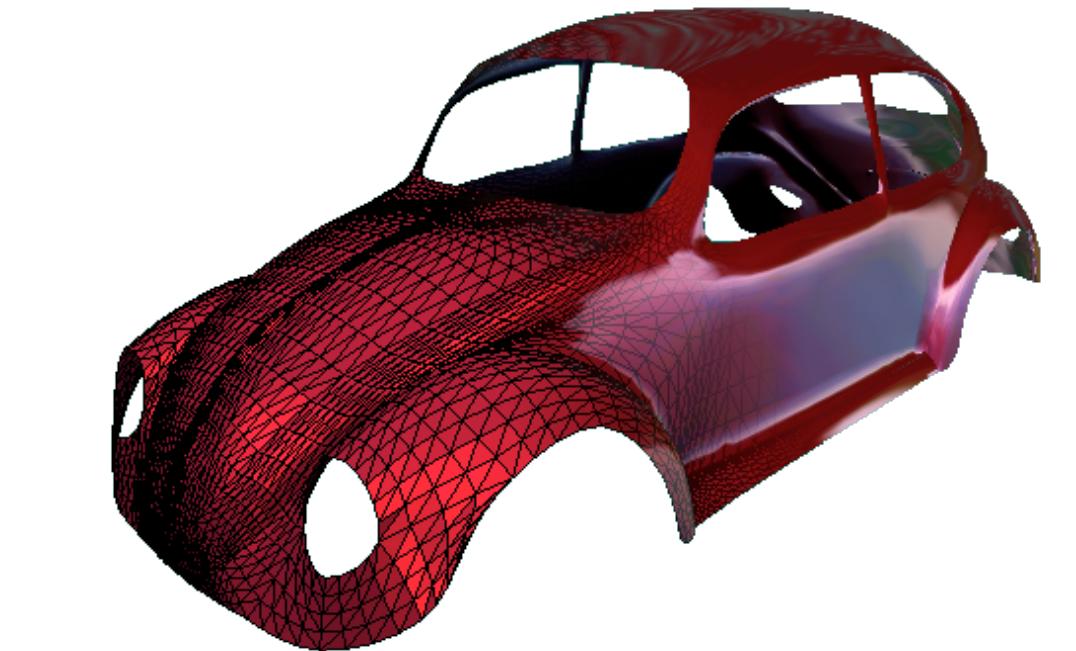
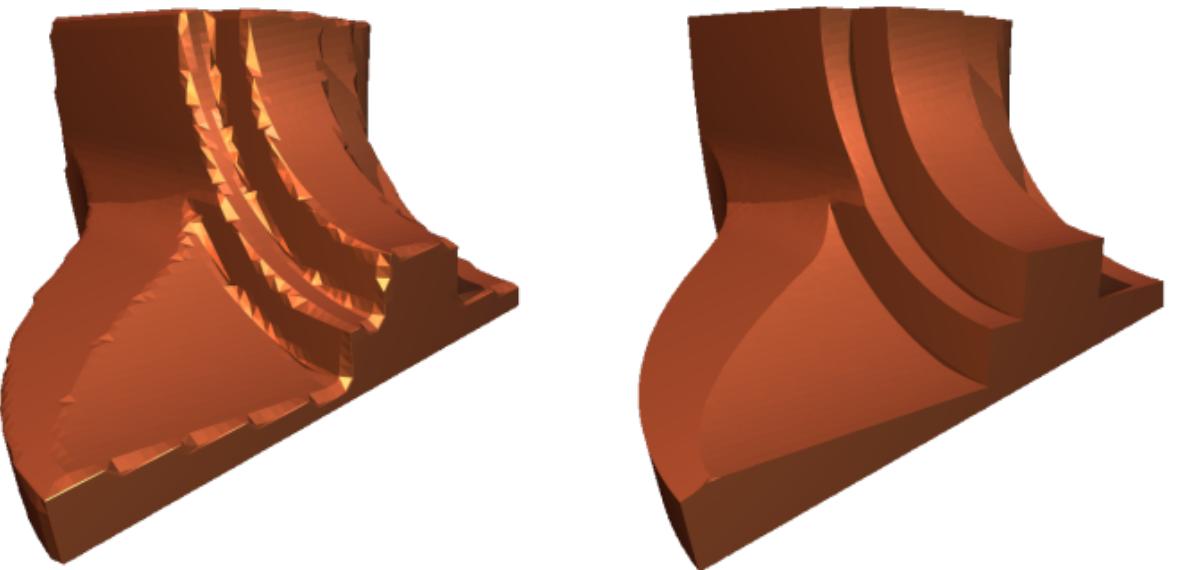
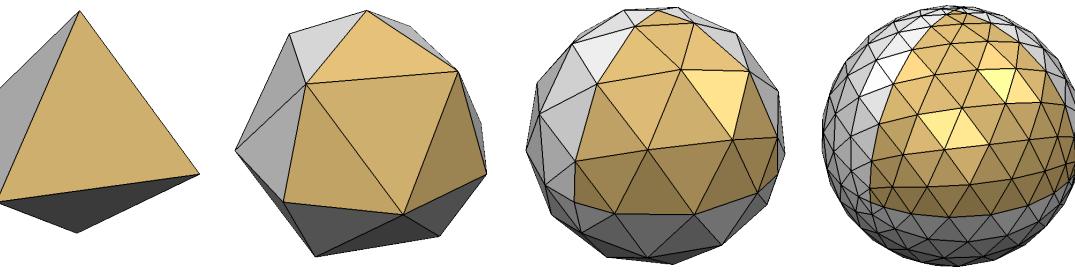
Meshes as Approximations of Smooth Surfaces

- Piecewise linear approximation
 - Error is $O(h^2)$ with h length of edges



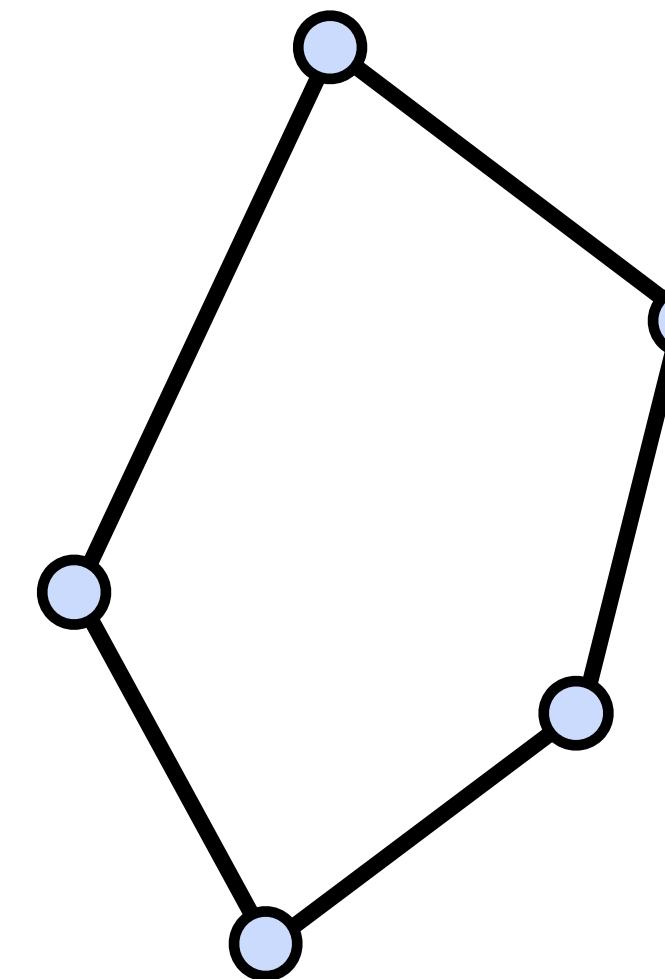
Polygonal Meshes

- Polygonal meshes are a good representation
 - approximation $O(h^2)$
 - arbitrary topology
 - piecewise smooth surfaces
 - adaptive refinement
 - efficient rendering

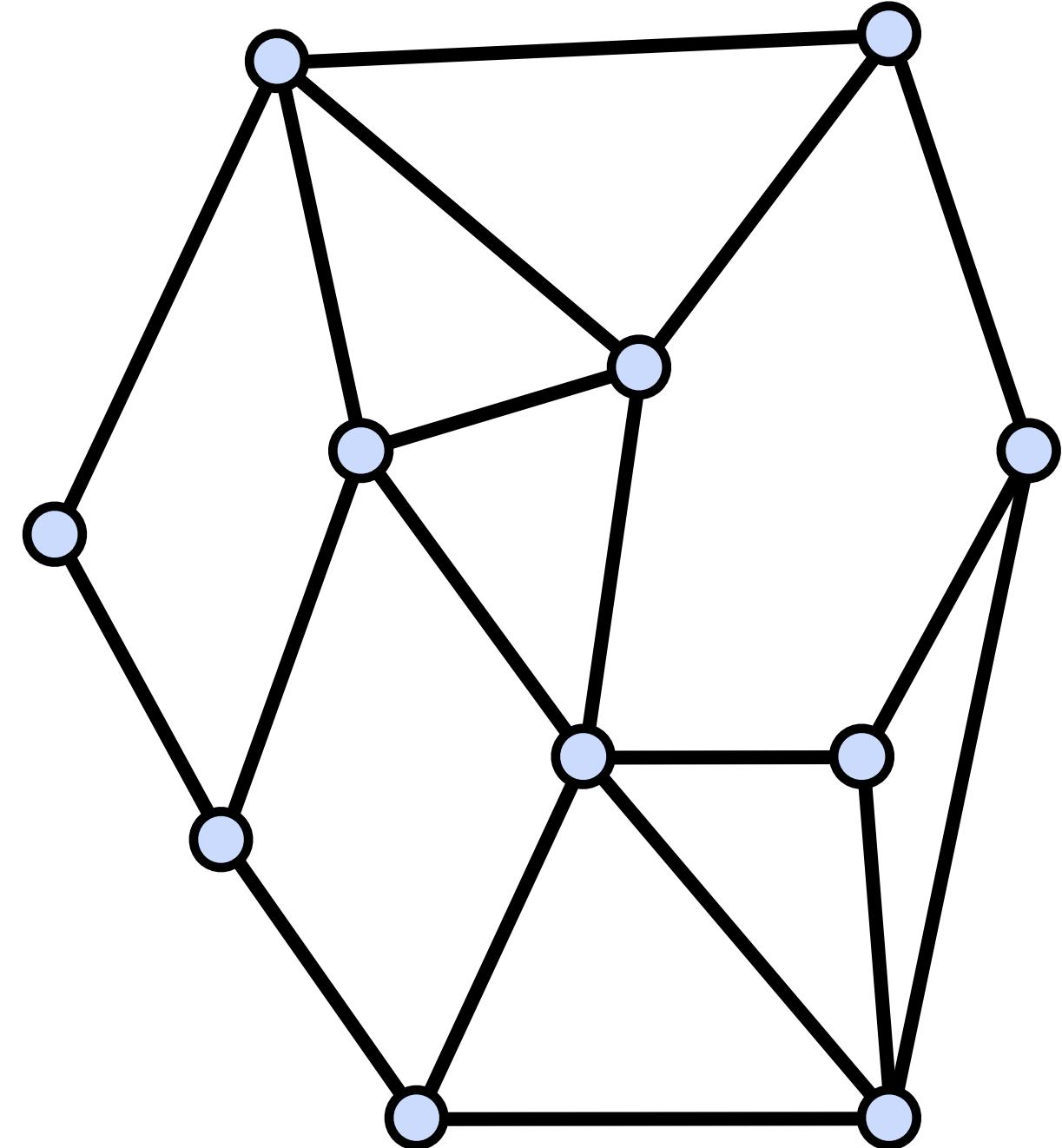


Polygon

- Vertices: v_0, v_1, \dots, v_{n-1}
- Edges: $\{(v_0, v_1), \dots, (v_{n-2}, v_{n-1}), (v_{n-1}, v_0)\}$
- Closed
- Planar: all vertices on a plane
- Simple: not self-intersecting



Polygonal Mesh

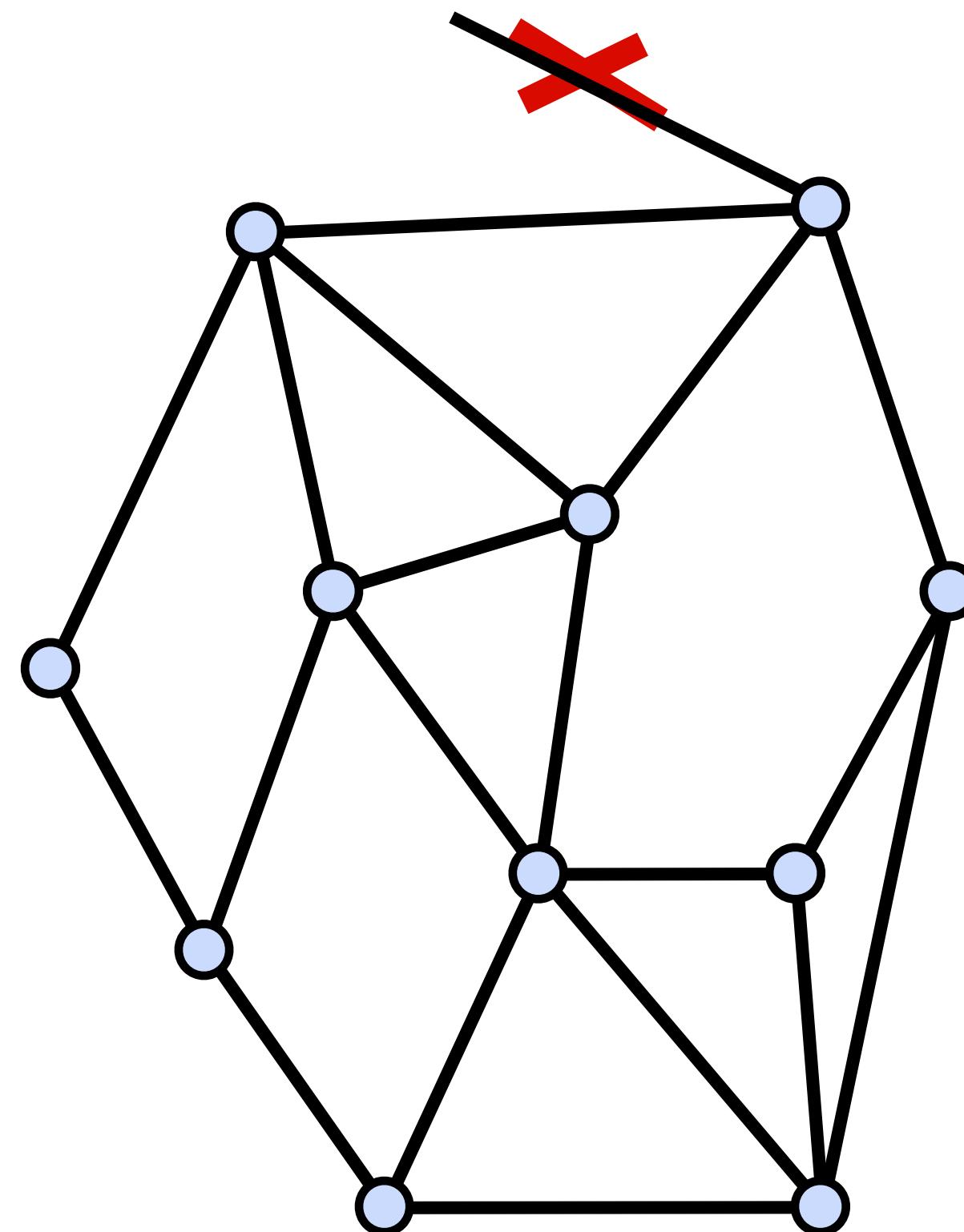


- A finite set M of closed, simple polygons Q_i
- The intersection of two polygons in M is either empty, or a vertex, or an edge

$$M = \langle V, E, F \rangle$$

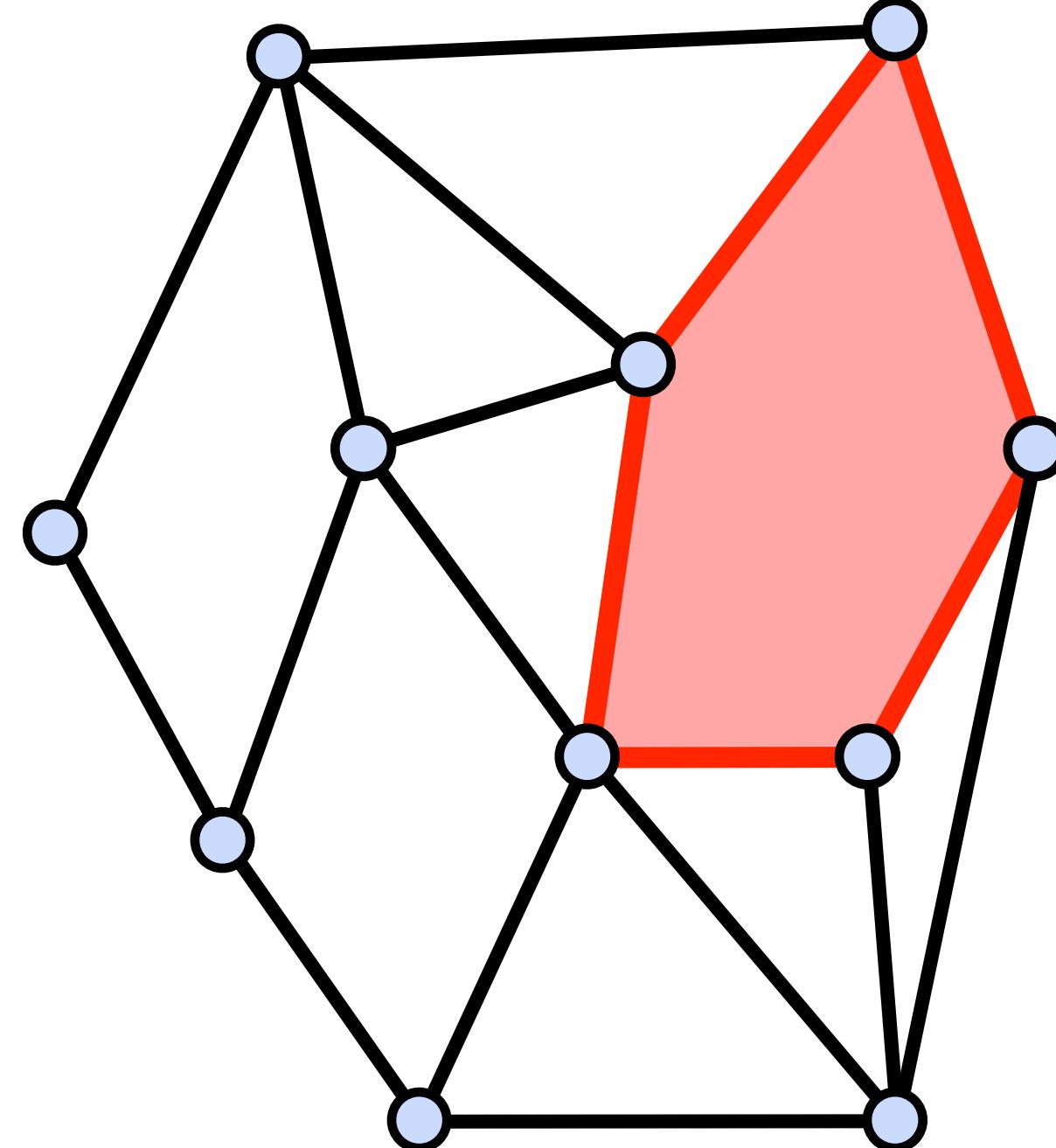
vertices edges faces

Polygonal Mesh



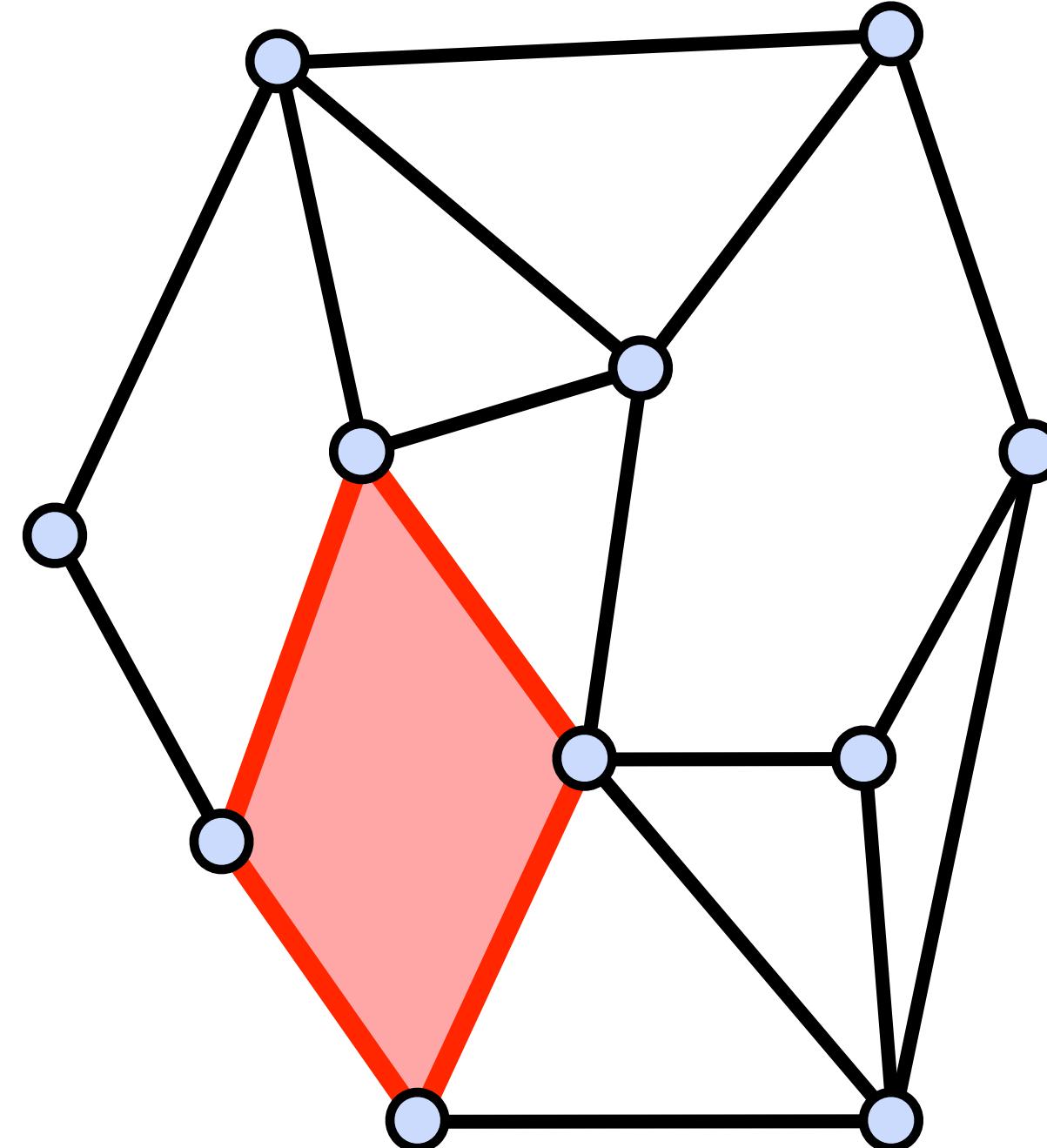
- A finite set M of closed, simple polygons Q_i
- The intersection of two polygons in M is either empty, or a vertex, or an edge
- Every edge belongs to at least one polygon

Polygonal Mesh



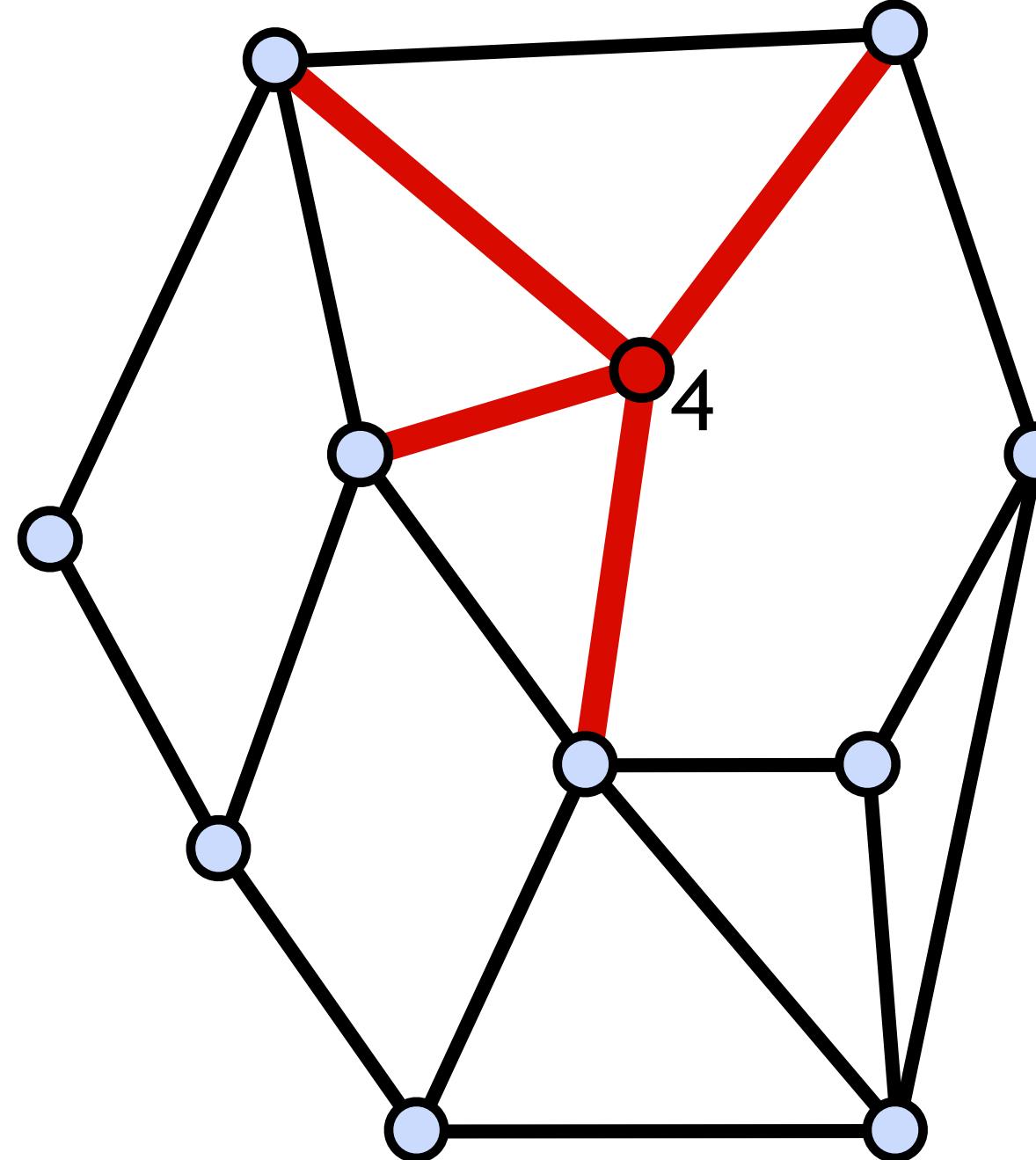
- A finite set M of closed, simple polygons Q_i
- The intersection of two polygons in M is either empty, or a vertex, or an edge
- Every edge belongs to at least one polygon
- Each Q_i defines a **face** of the polygonal mesh

Polygonal Mesh



- A finite set M of closed, simple polygons Q_i
- The intersection of two polygons in M is either empty, or a vertex, or an edge
- Every edge belongs to at least one polygon
- Each Q_i defines a **face** of the polygonal mesh

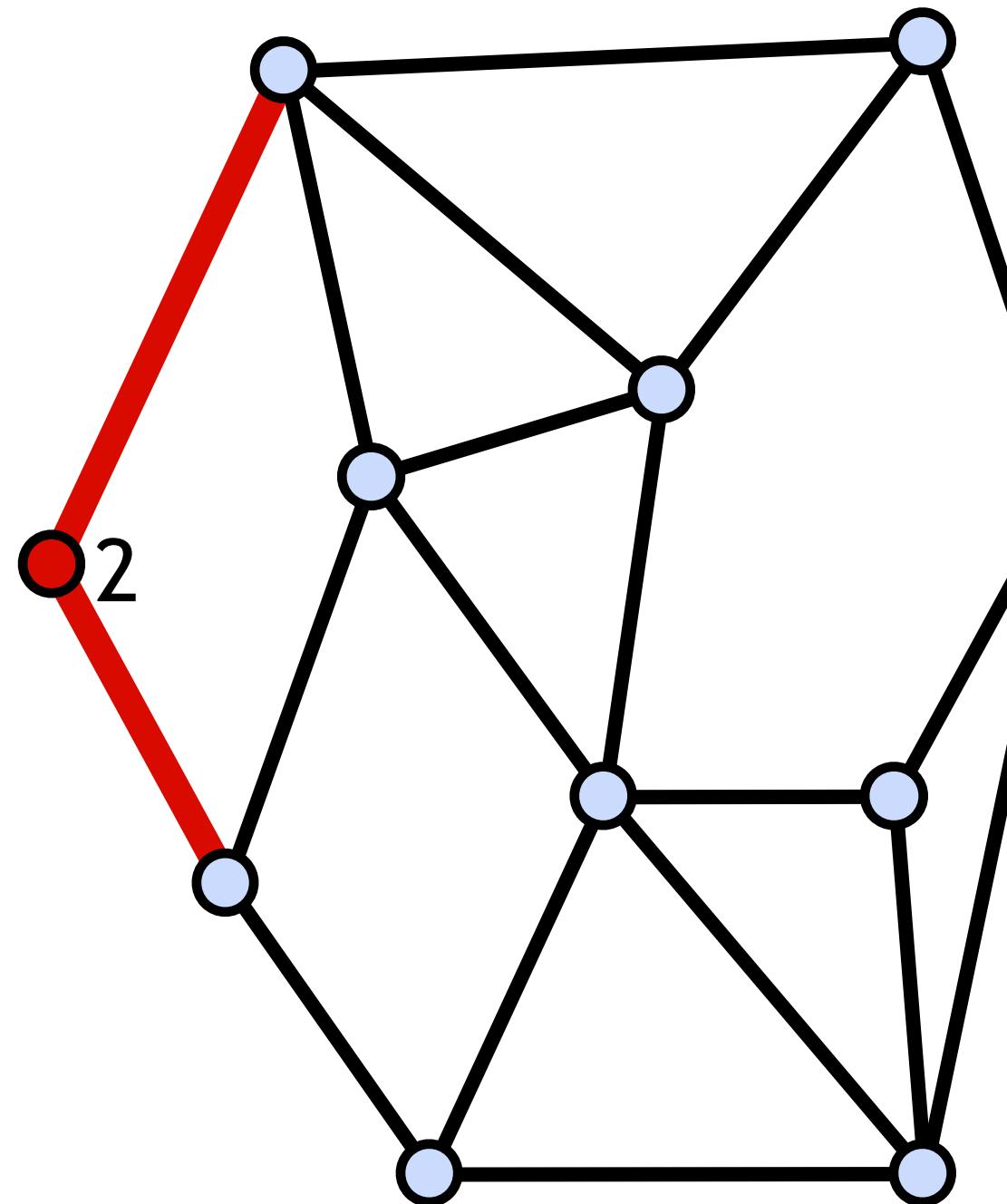
Polygonal Mesh



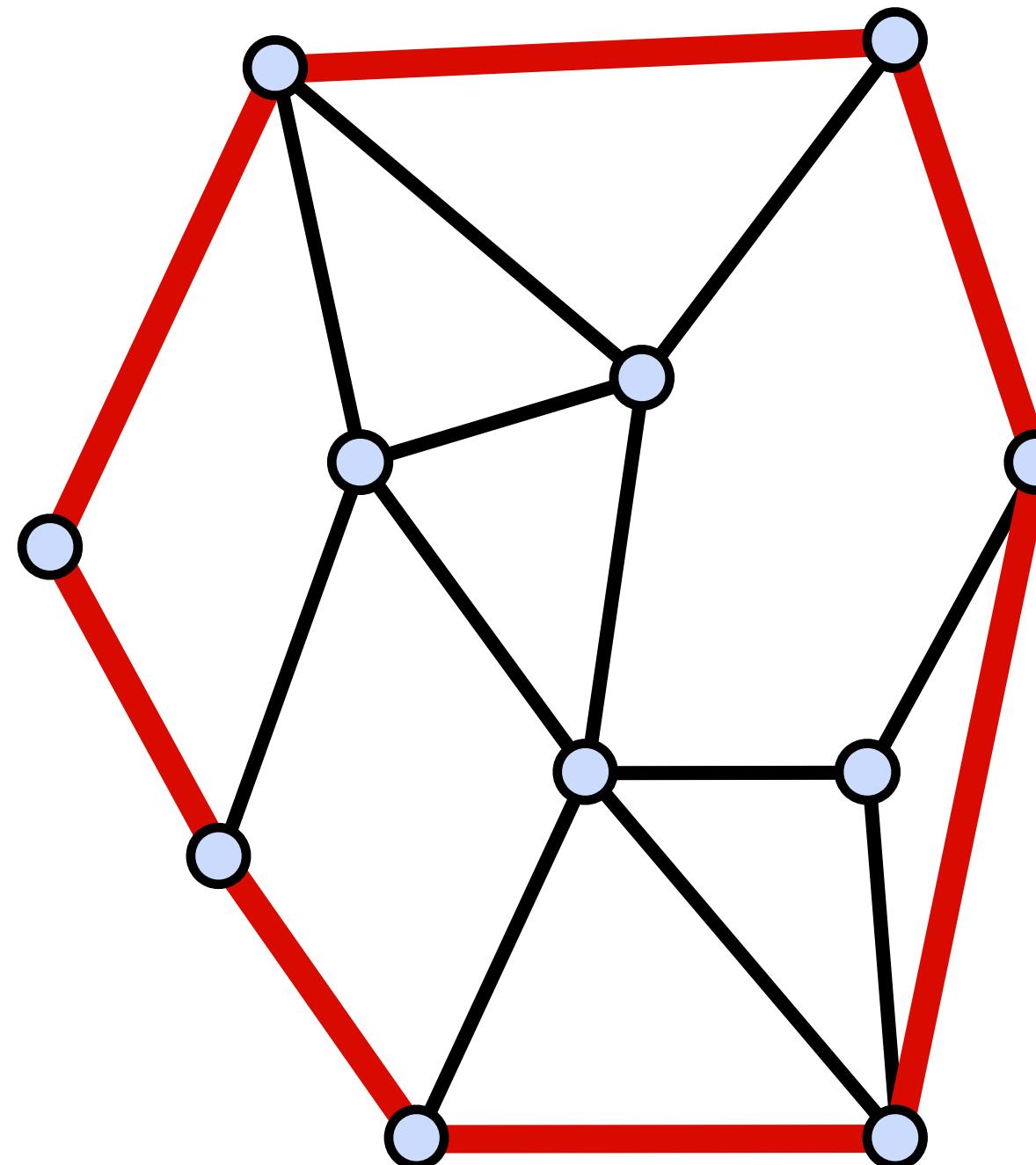
- Vertex **degree** or **valence** = number of incident edges

Polygonal Mesh

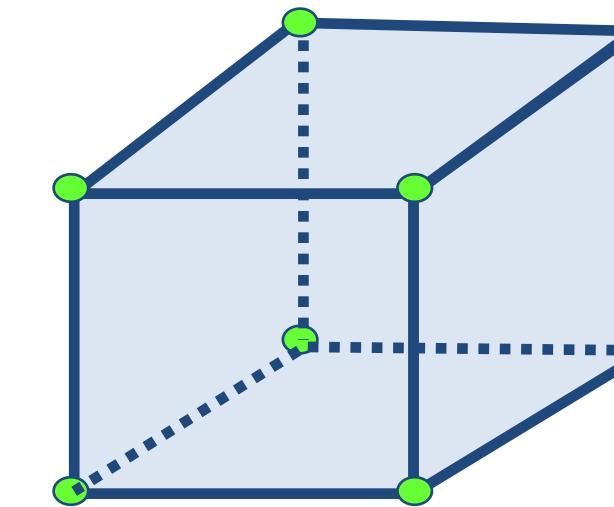
- Vertex **degree** or **valence** = number of incident edges



Polygonal Mesh



- **Boundary:** the set of all edges that belong to only one polygon
 - Either empty or forms closed loops
 - If empty, then the polygonal mesh is said to be **closed or watertight**



Triangle Meshes

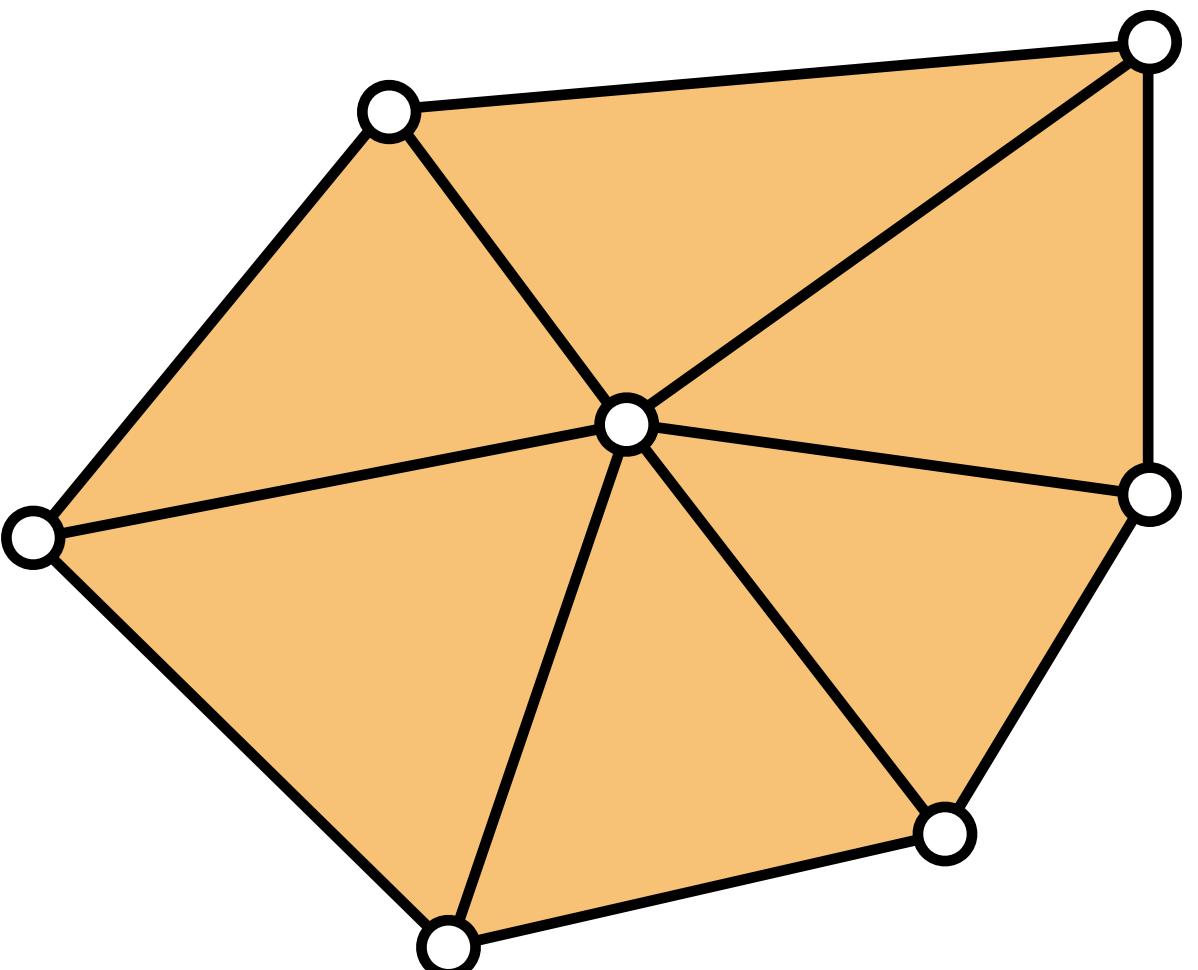
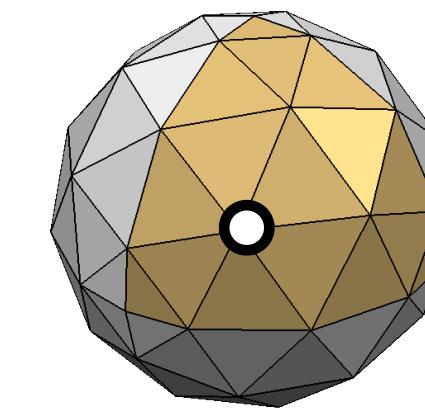
- Connectivity: vertices, edges, triangles
- Geometry: vertex positions

$$V = \{v_1, \dots, v_n\}$$

$$E = \{e_1, \dots, e_k\}, \quad e_i \in V \times V$$

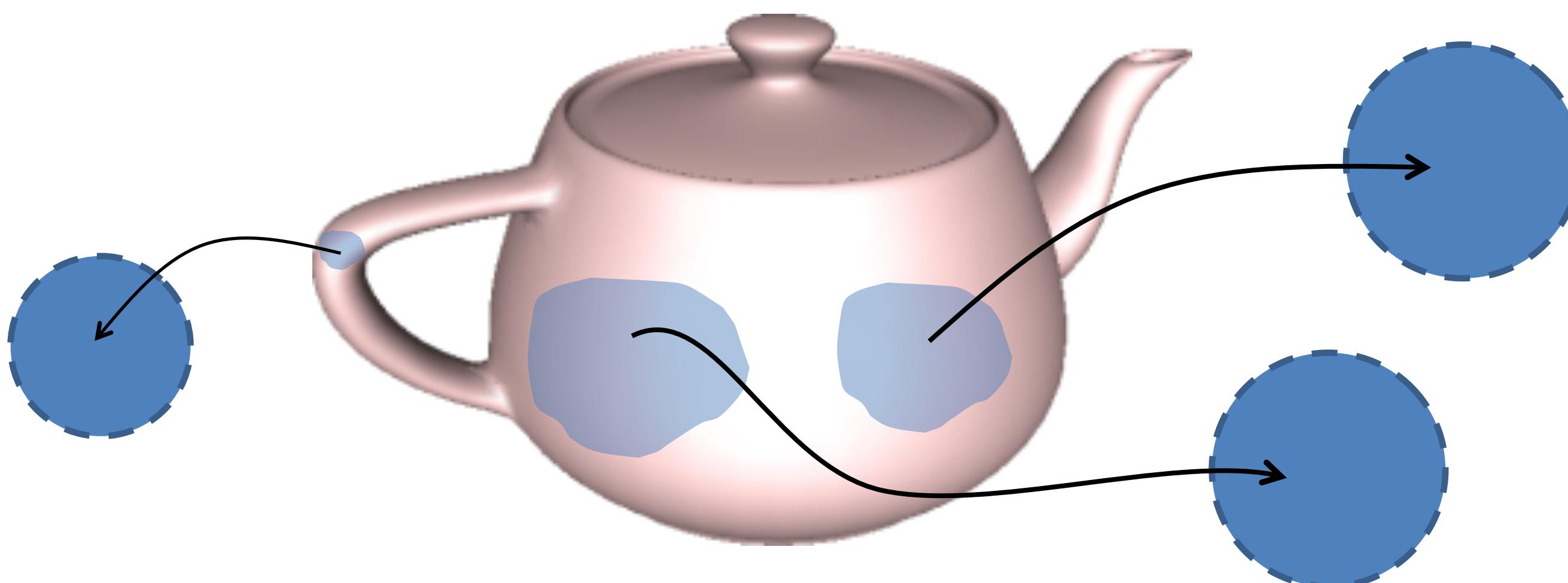
$$F = \{f_1, \dots, f_m\}, \quad f_i \in V \times V \times V$$

$$P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}, \quad \mathbf{p}_i \in \mathbb{R}^3$$



Manifolds

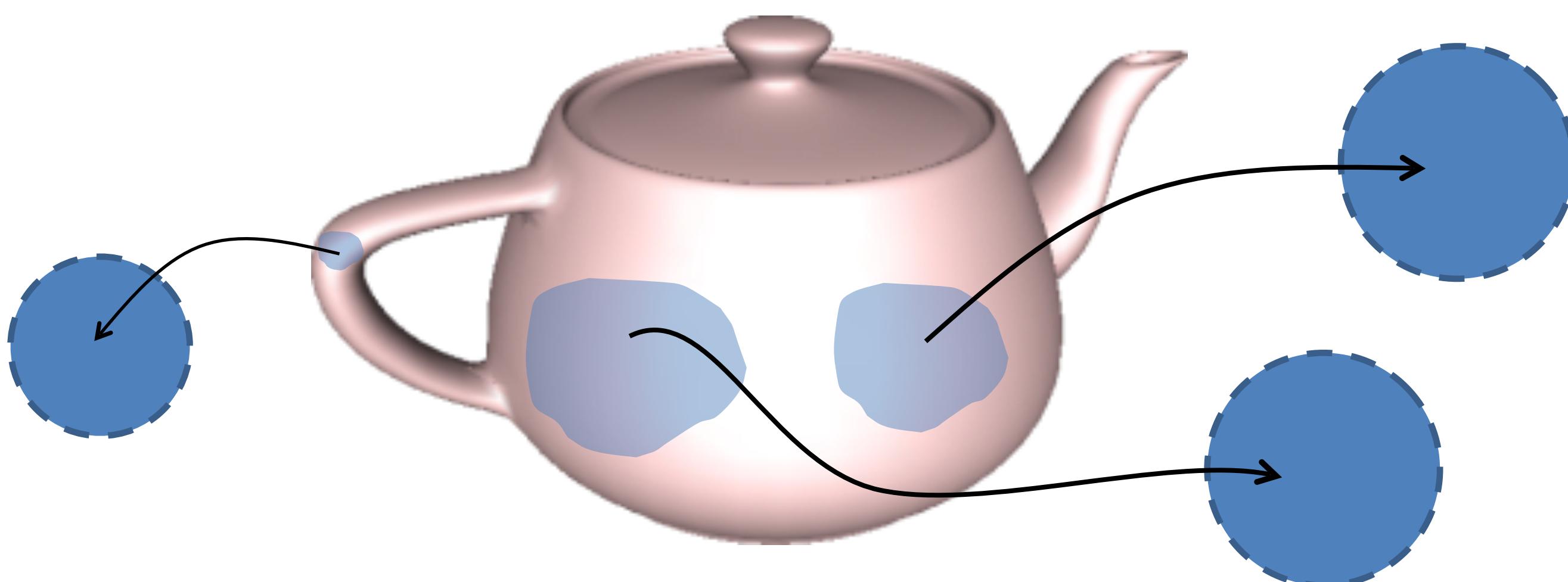
- A surface is a closed **2-manifold** if it is everywhere locally homeomorphic to a disk



Manifolds

- For every point x in M , there is an **open ball** $B_x(r)$ of radius $r > 0$ centered at x such that $M \cap B_x$ is homeomorphic to an open disk

$$B_x(r) = \{\mathbf{y} \in \mathbb{R}^3 \text{ s.t. } \|\mathbf{y} - \mathbf{x}\| < r\}$$



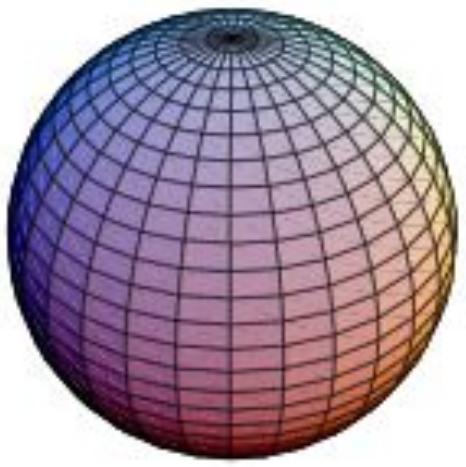
Manifolds

- Manifold with boundary: a vicinity of each boundary point is homeomorphic to a half-disk

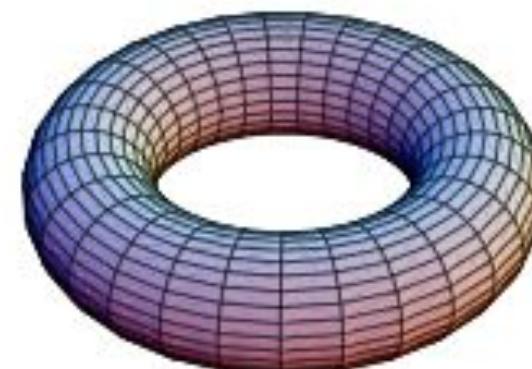


Examples

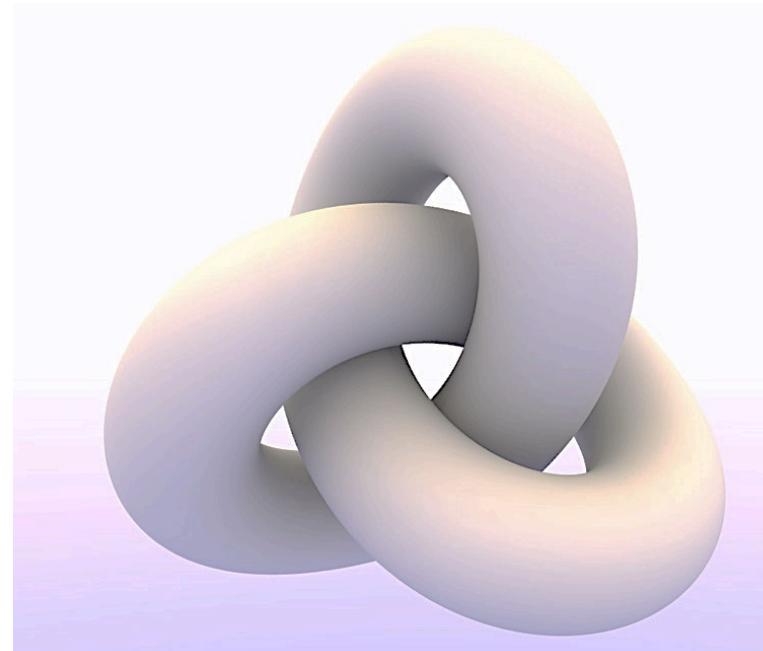
- For each case, decide if it is a 2-manifold (possibly with boundary) or not. If not, explain why not.



Case 1



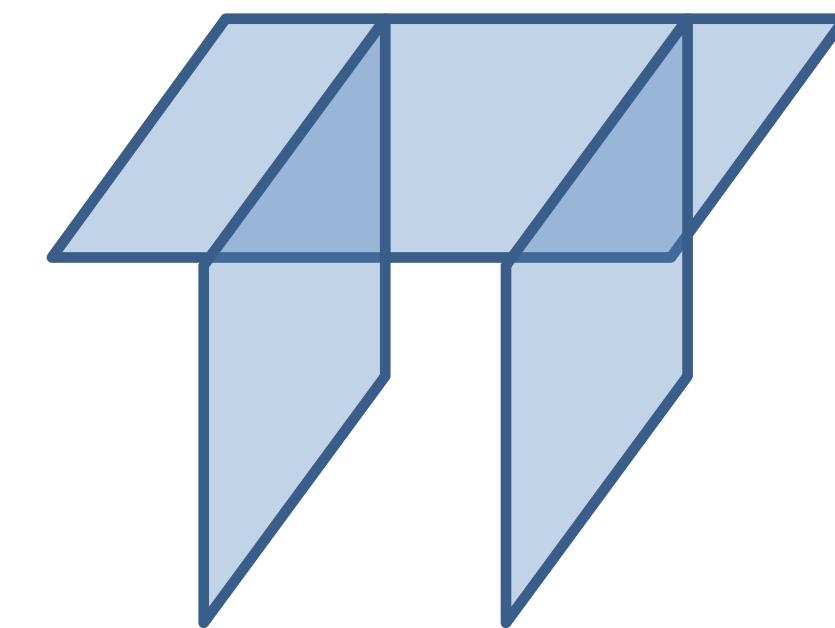
Case 2



Case 3



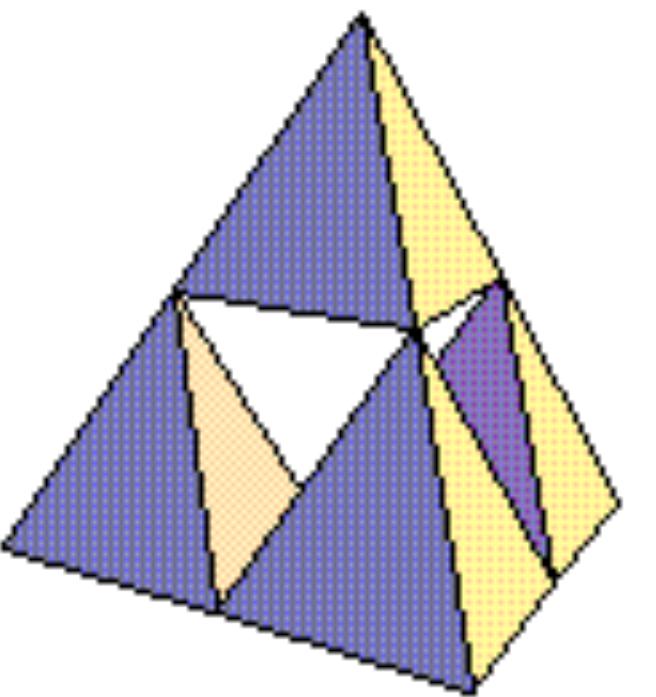
Case 4



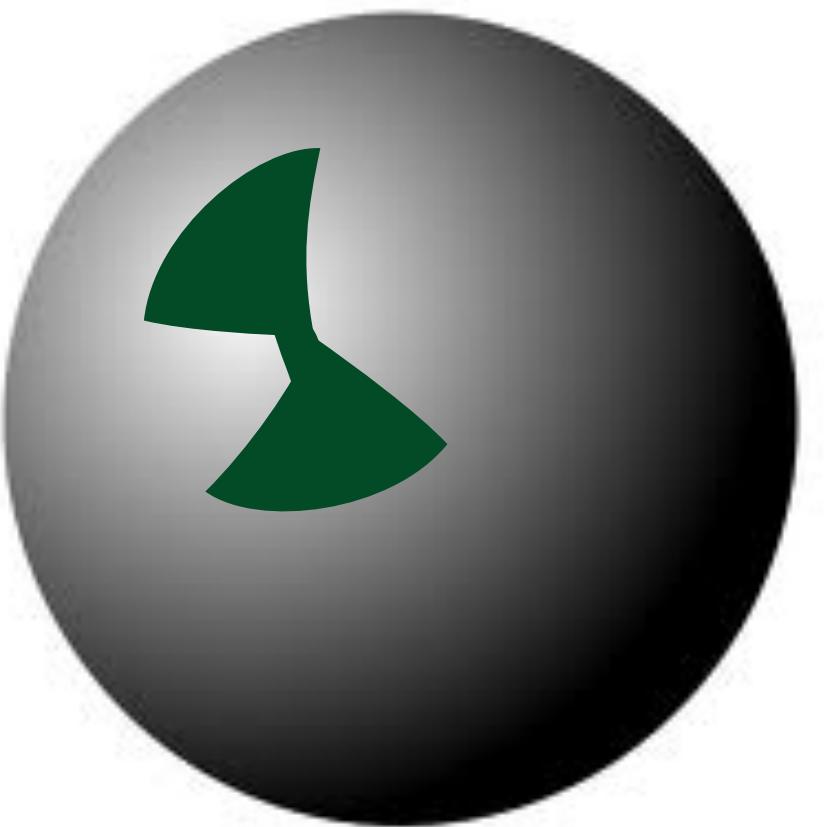
Case 5

Examples

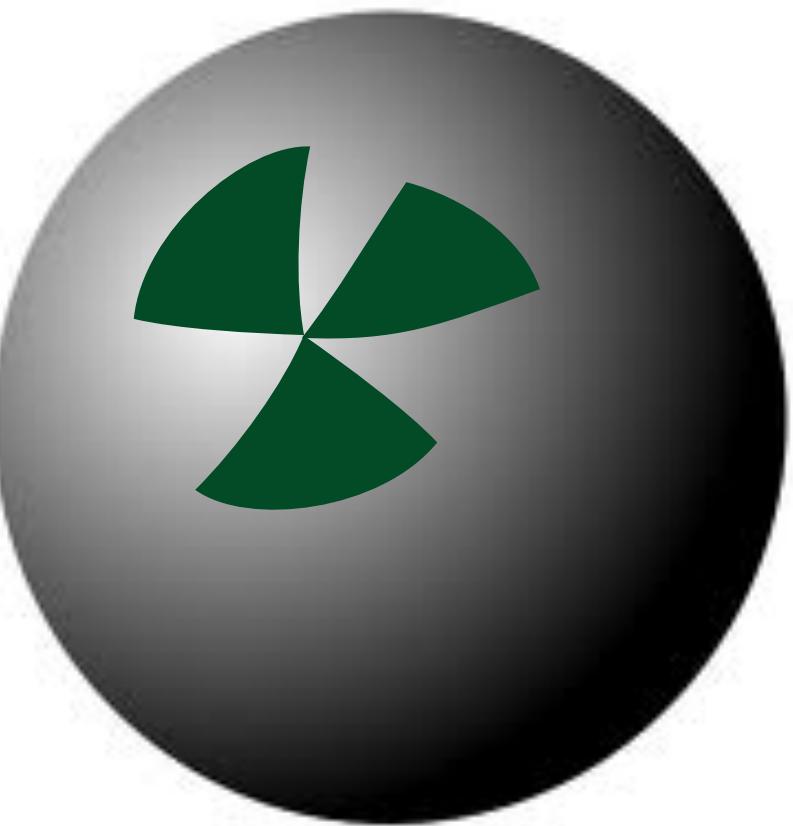
- Bonus cases



Case 6



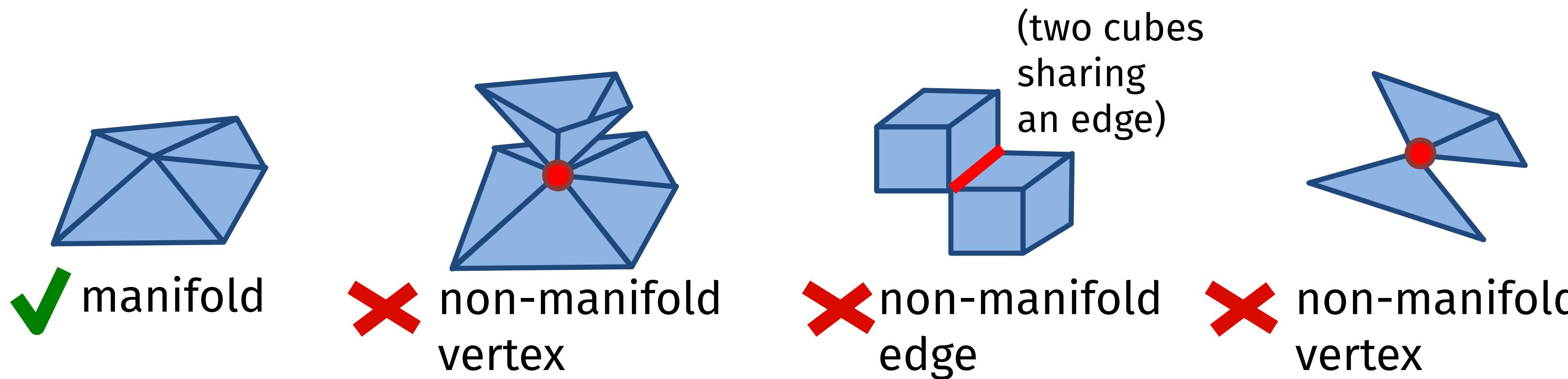
Case 7



Case 8

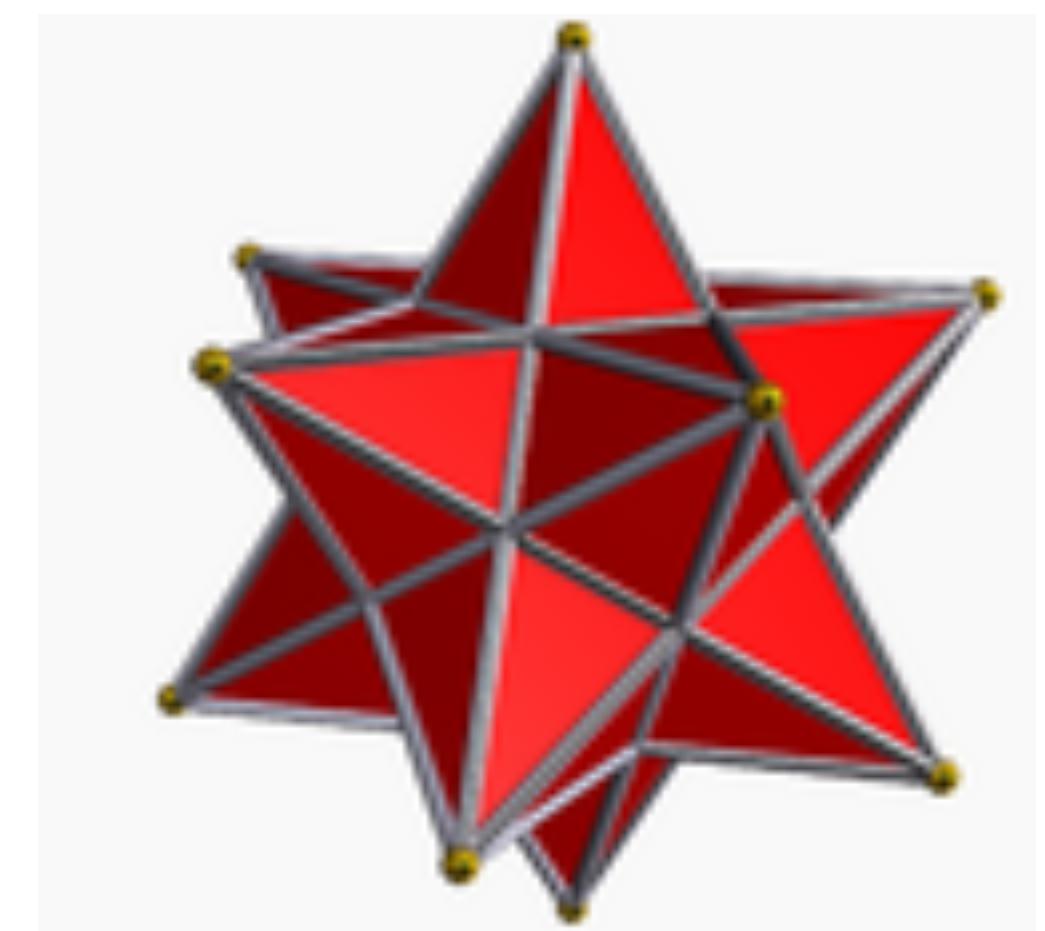
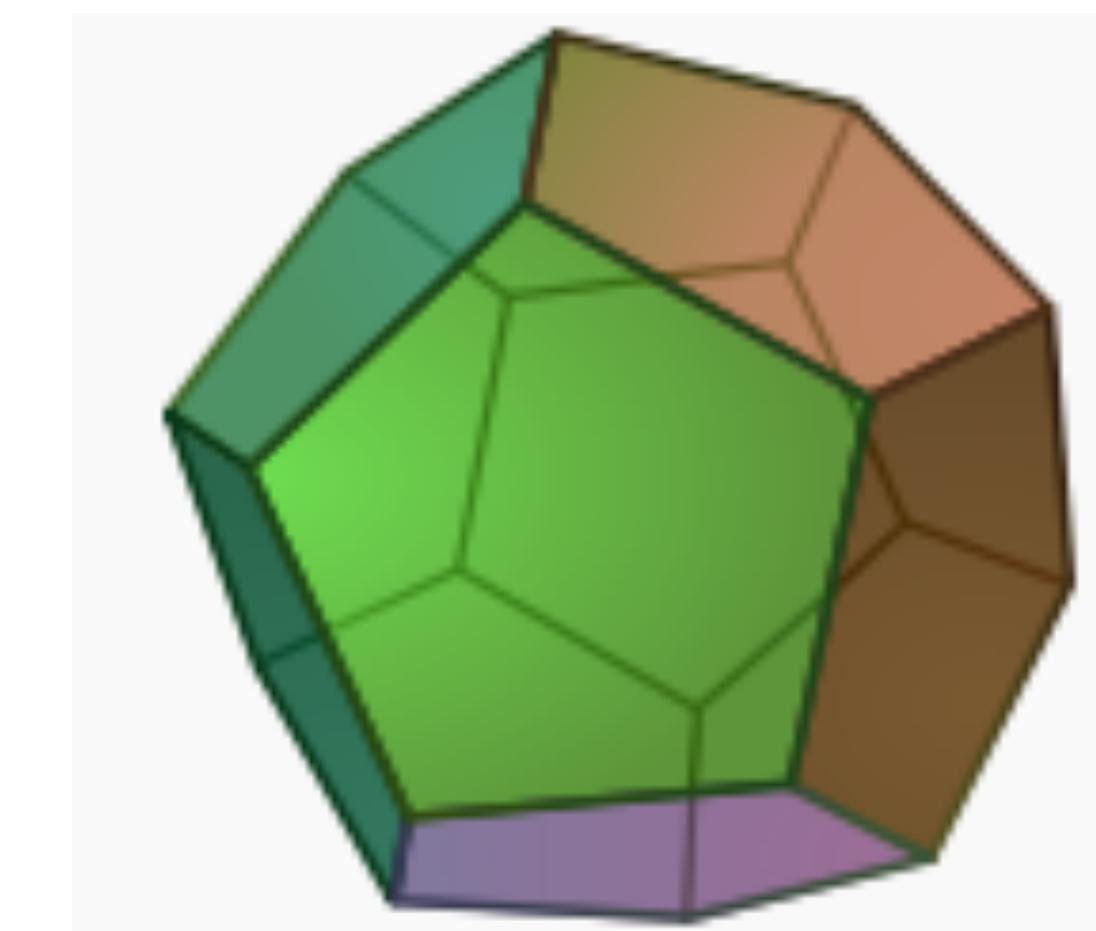
Manifolds

- In a manifold mesh, there are at most 2 faces sharing an edge
 - Boundary edges: have one incident face
 - Inner edges have two incident faces
- A manifold vertex has 1 connected ring of faces around it, or 1 connected half-ring (boundary)



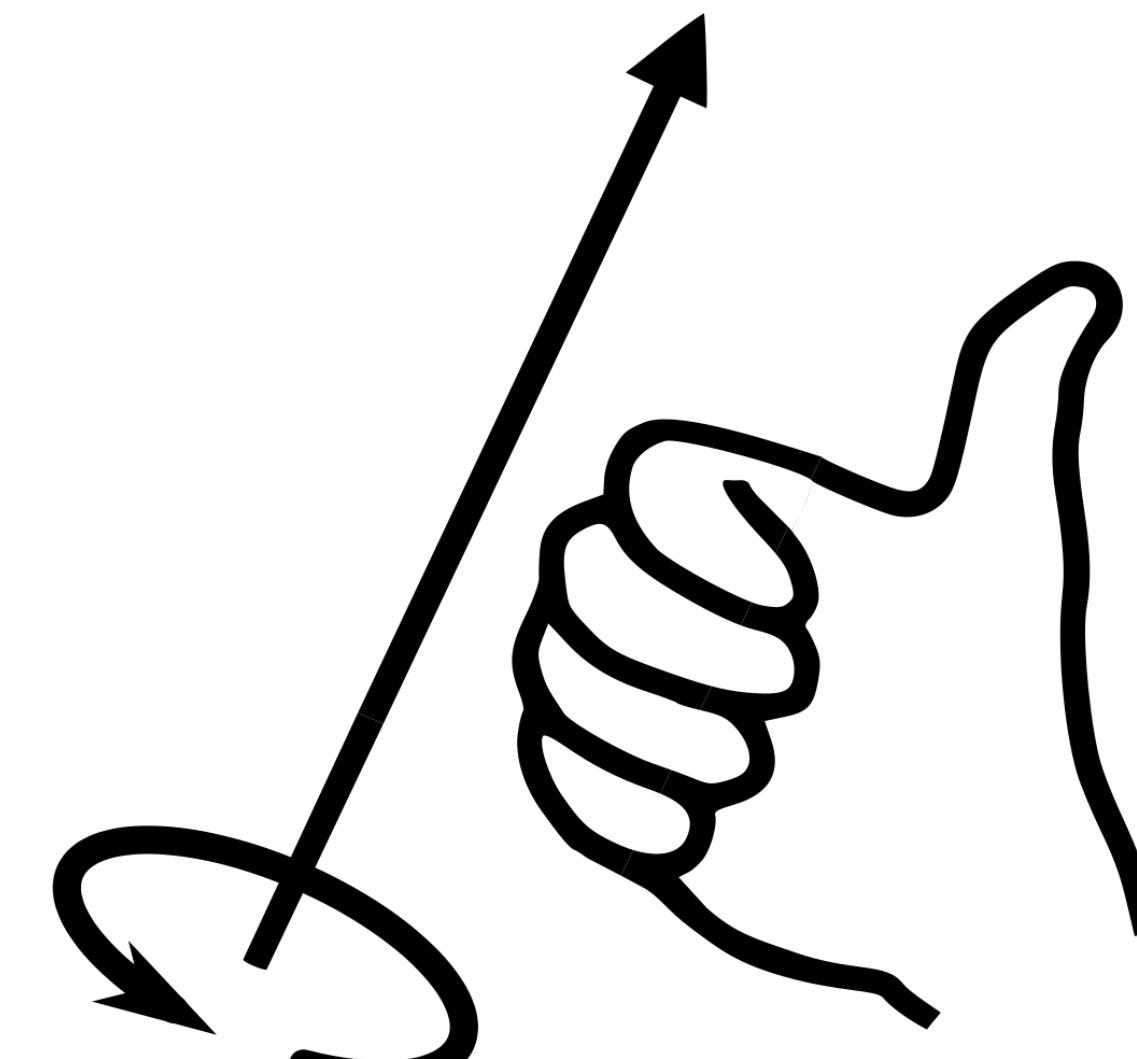
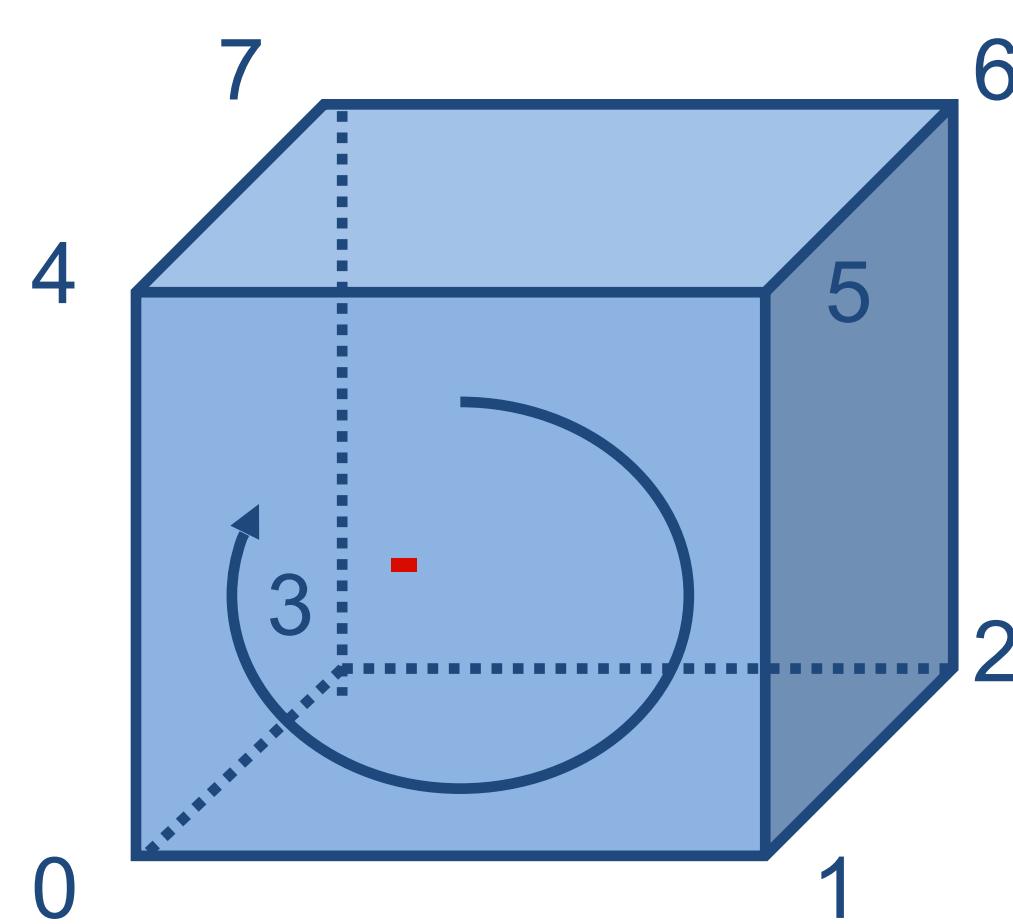
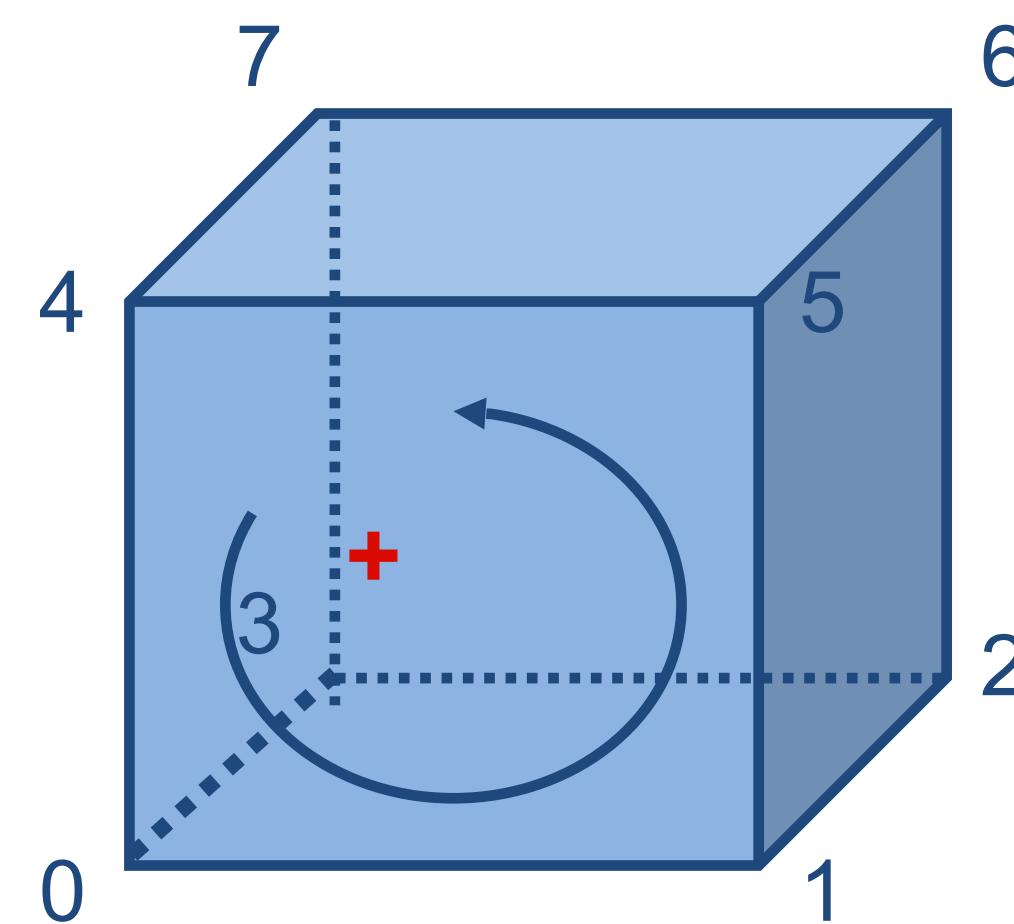
Jordan theorem and polyhedra

- If closed and not intersecting, a manifold divides the space into **interior** and **exterior**
- The manifold is the **boundary** of the object that occupies the interior
- A closed manifold polygonal mesh is called a **polyhedron**



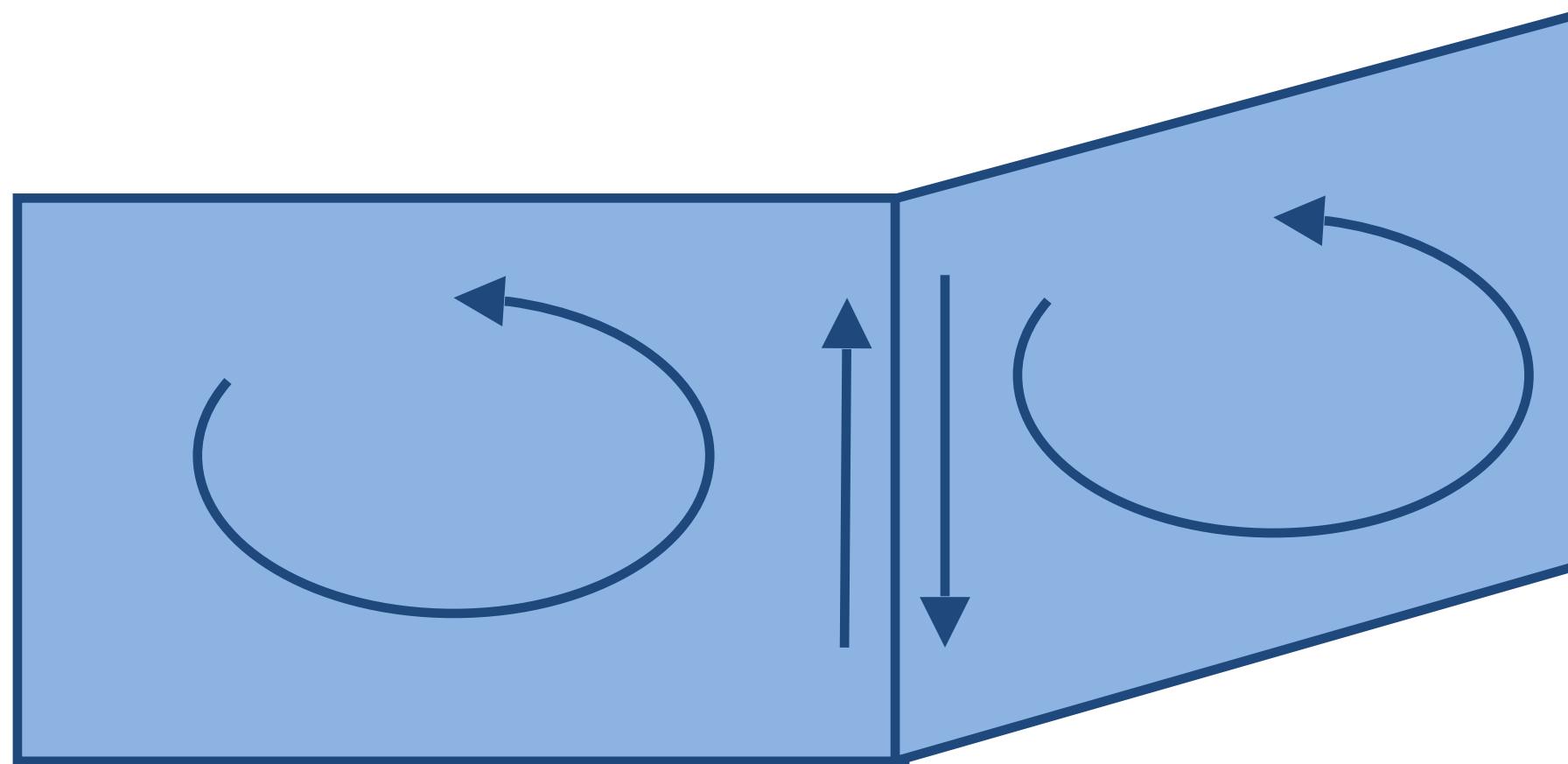
Orientation

- Every face of a polygonal mesh is orientable
 - Clockwise vs. counterclockwise order of face vertices
 - Defines sign/direction of the surface normal



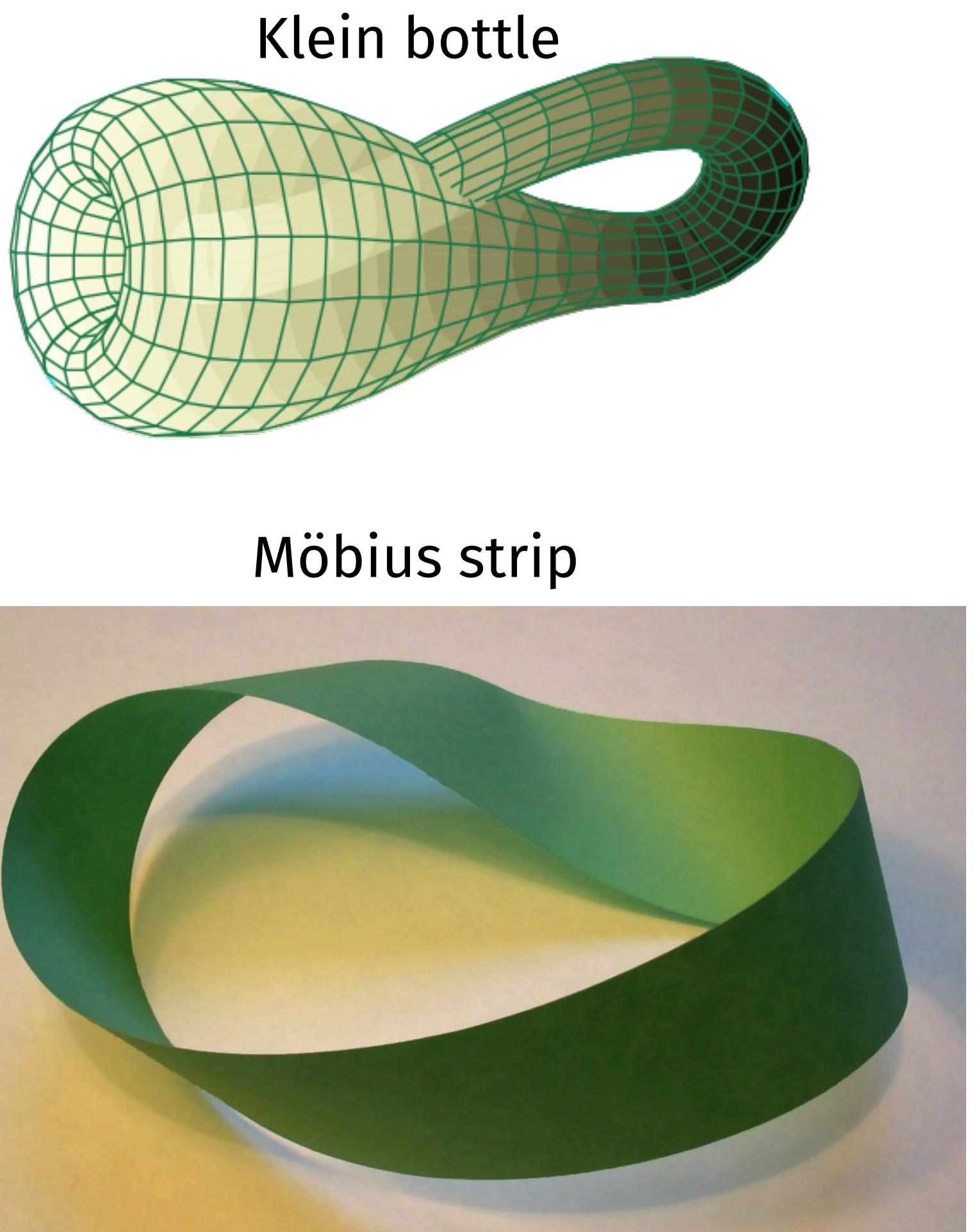
Orientation

- Consistent orientation of neighboring faces:



Orientability

- A polygonal mesh is orientable, if the incident faces to every edge can be consistently oriented
 - If the faces are consistently oriented for every edge, the mesh is oriented
- Notes
 - Every non-orientable closed mesh embedded in \mathbb{R}^3 intersects itself
 - The surface of a polyhedron is always orientable



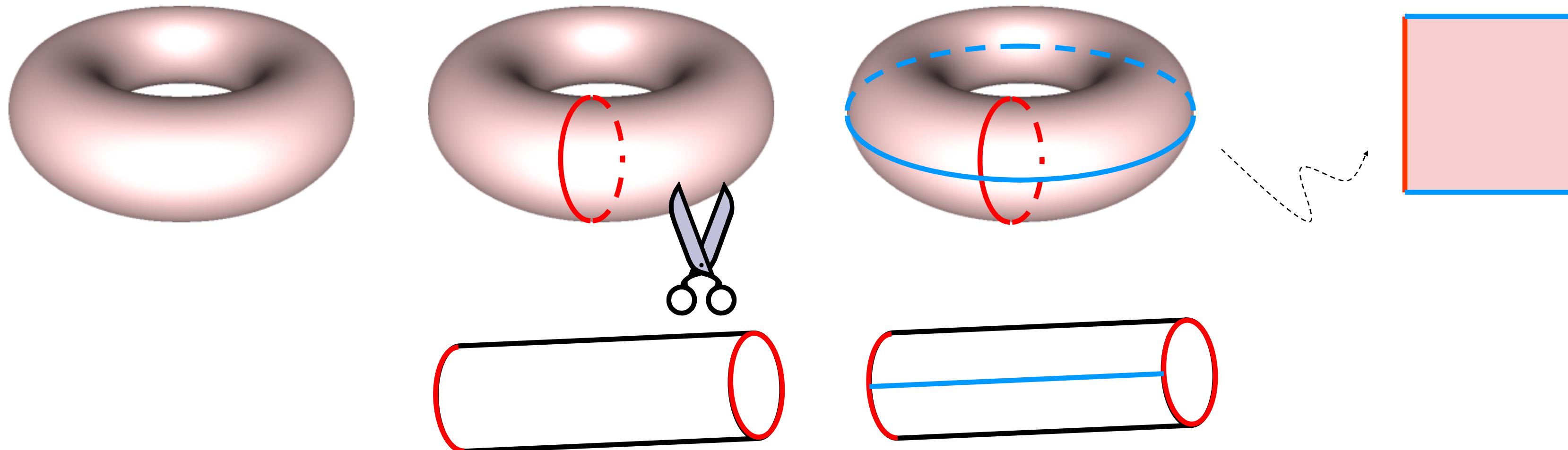
By David Benbennick - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=50359>

By Tttrung - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=960446>

80412 - 2024/25 - Geometric Modeling - Enrico Puppo

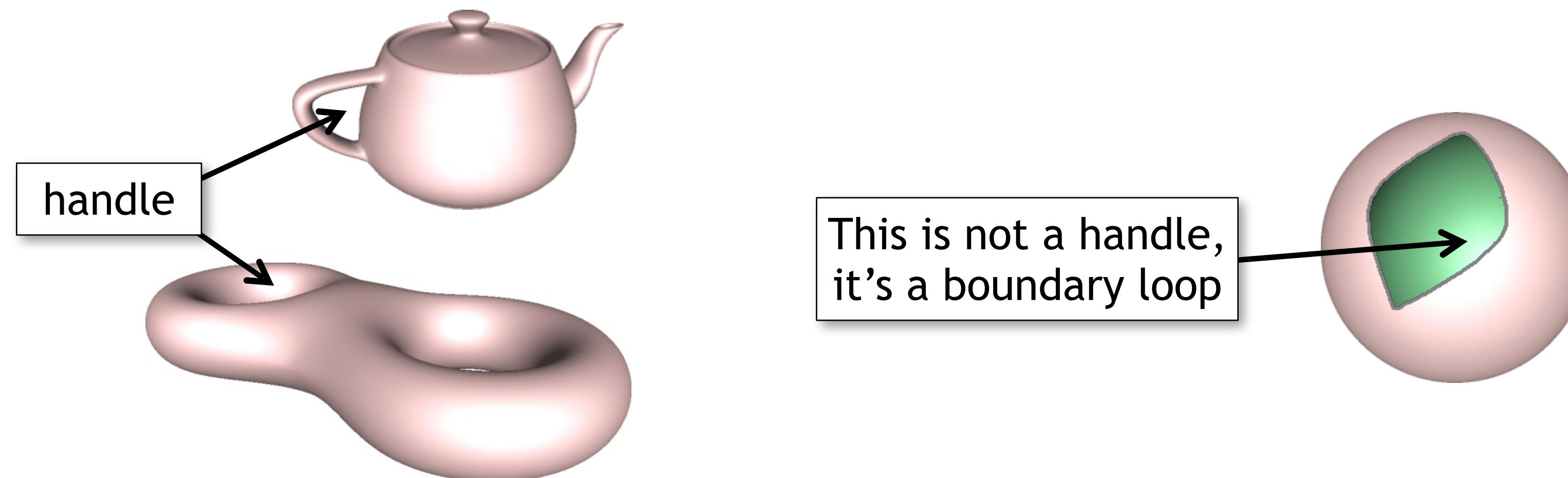
Global Topology of Meshes

- **Genus:** $\frac{1}{2} \times$ the maximal number of independent closed paths that do not disconnect the graph.
 - Informally, the number of handles (“donut holes”).



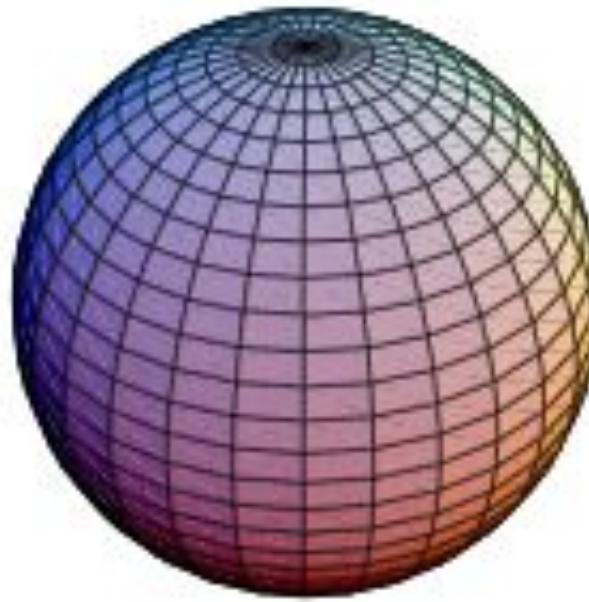
Global Topology of Meshes

- **Genus:** $\frac{1}{2} \times$ the maximal number of closed independent paths that do not disconnect the graph.
 - Informally, the number of handles (“donut holes”).

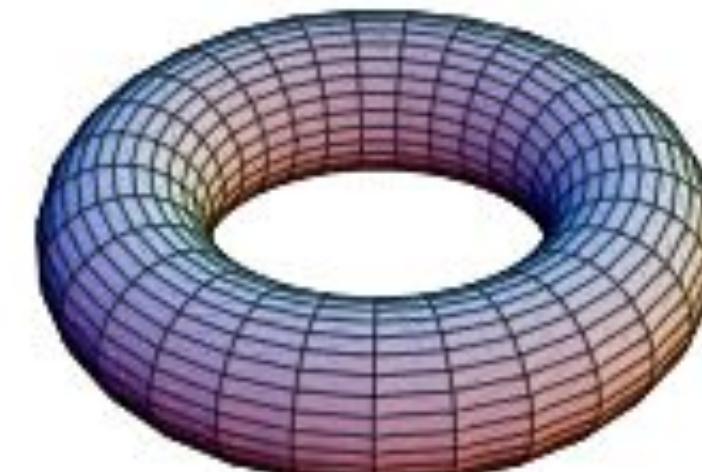


Global Topology of Meshes

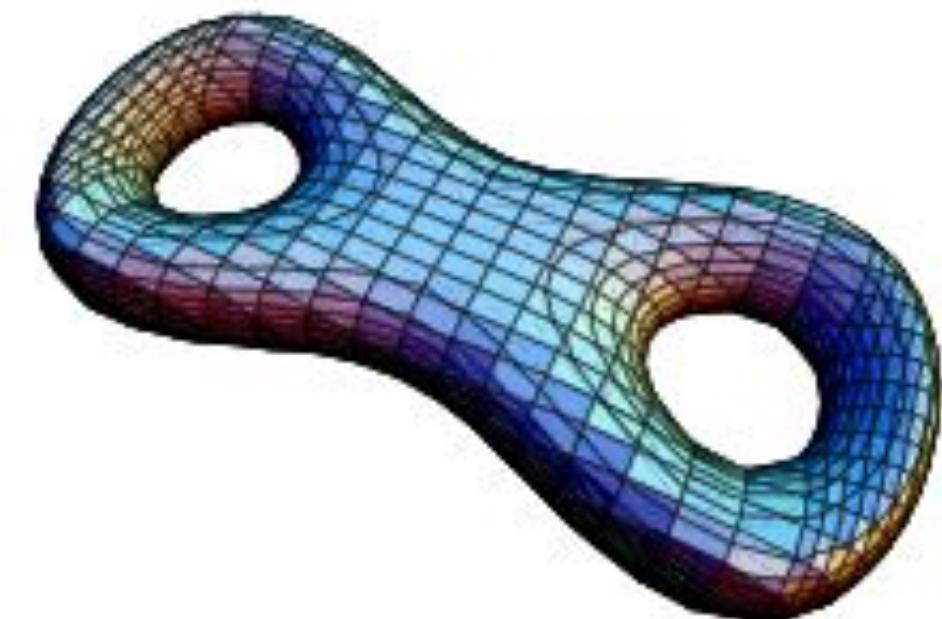
- **Genus:** $\frac{1}{2} \times$ the maximal number of closed independent paths that do not disconnect the graph.
 - Informally, the number of handles (“donut holes”).



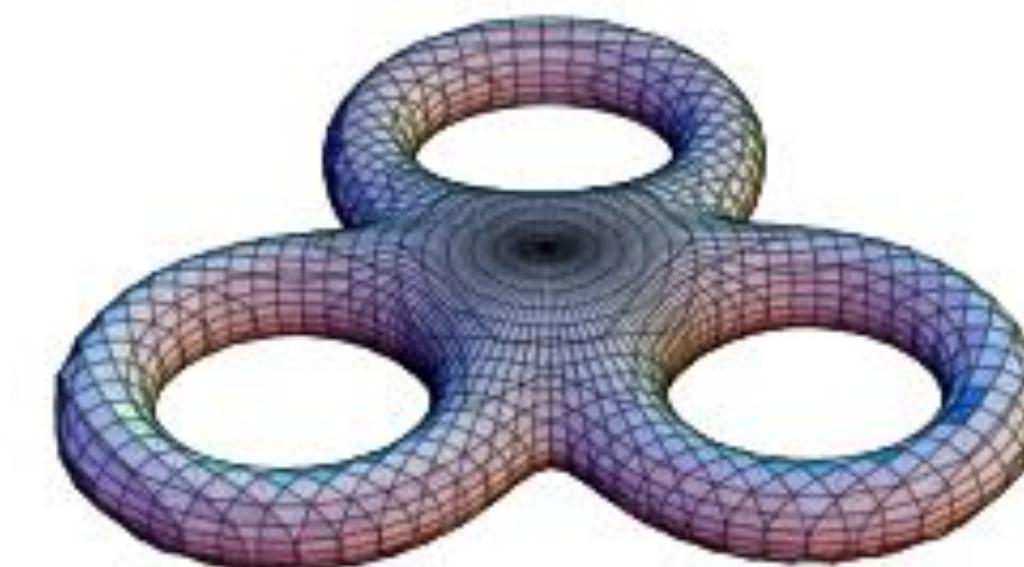
Genus 0



Genus 1



Genus 2



Genus 3

Euler-Poincaré Formula

- Theorem (Euler): The sum

$$\chi(M) = v - e + f$$

is **constant** for a given surface topology, no matter which (manifold) mesh we choose to cover M .

- v = number of vertices
- e = number of edges
- f = number of faces

Euler-Poincaré Formula

- For orientable meshes:

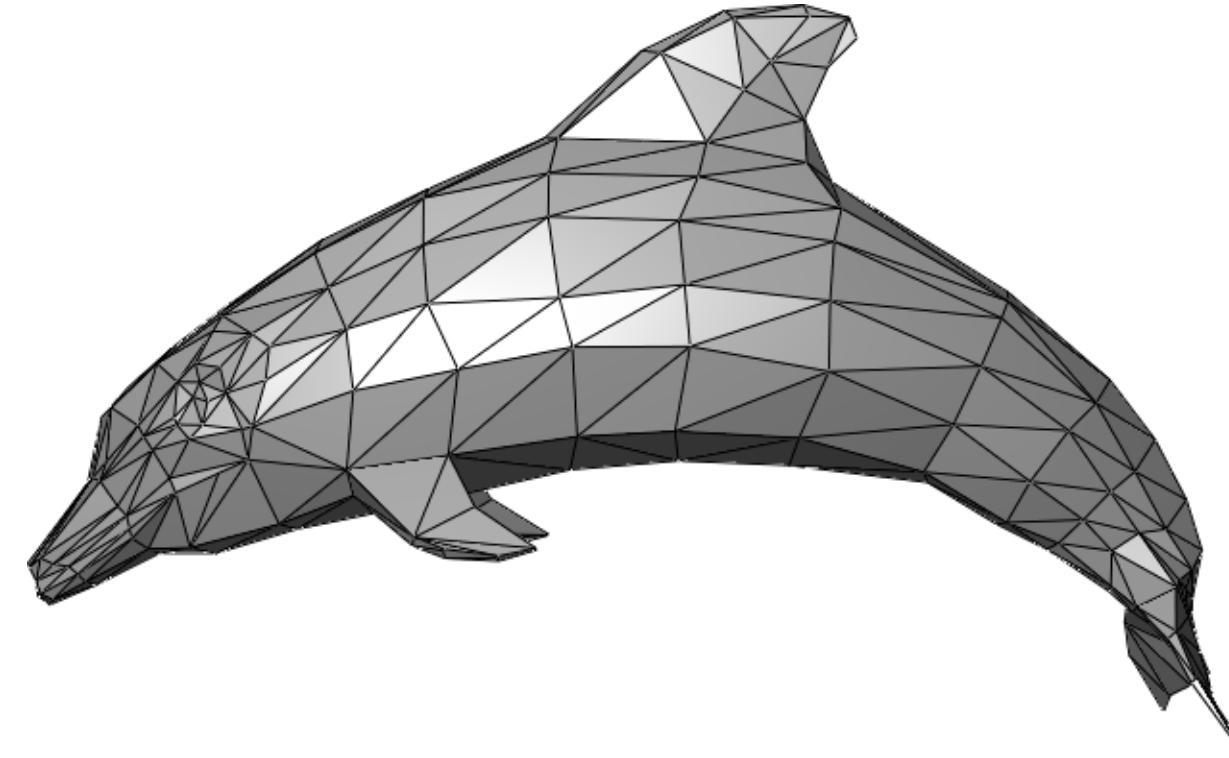
$$v - e + f = 2(c - g) - b = \chi(M)$$

- c = number of connected components
- g = genus
- b = number of boundary loops

$$\chi(\text{Sphere}) = 2 \quad \chi(\text{Torus}) = 0$$

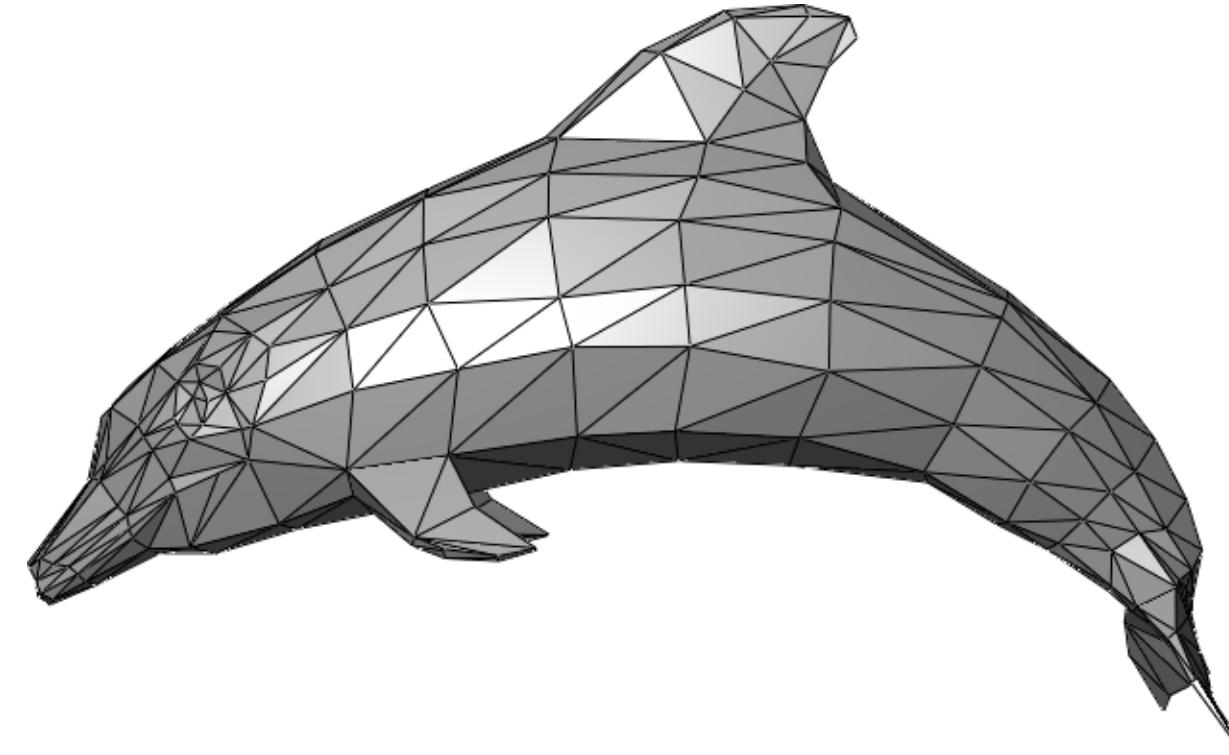
Implication for Mesh Storage

- Let's count the edges and faces in a **closed triangle mesh**:
 - Ratio of edges to faces: $e = 3/2 f$
 - each edge belongs to exactly 2 triangles
 - each triangle has exactly 3 edges



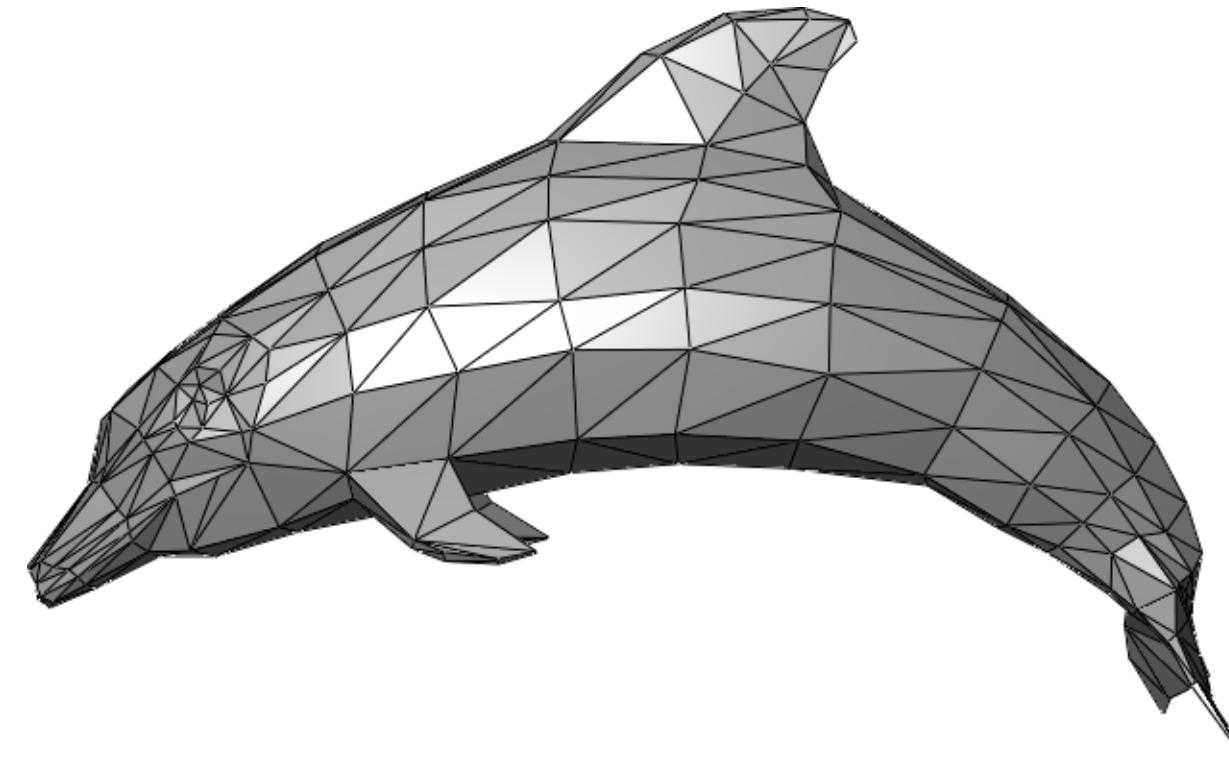
Implication for Mesh Storage

- Let's count the edges and faces in a **closed triangle mesh**:
 - Ratio of edges to faces: $e = 3/2 f$
 - each edge belongs to exactly 2 triangles
 - each triangle has exactly 3 edges
 - Ratio of vertices to faces: $f \sim 2v$
 - $2 = v - e + f = v - 3/2 f + f$
 - $2 + f/2 = v$



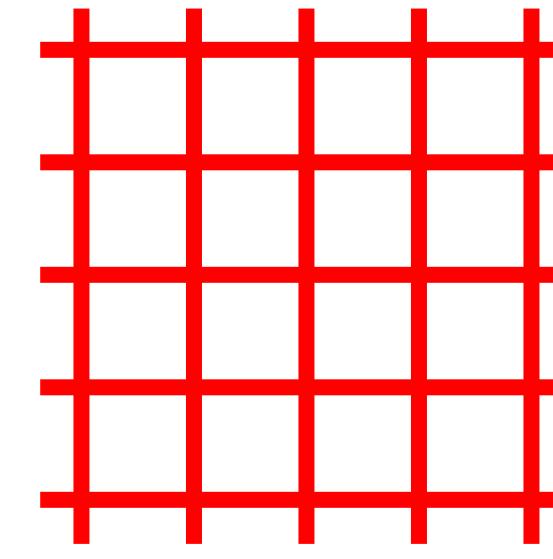
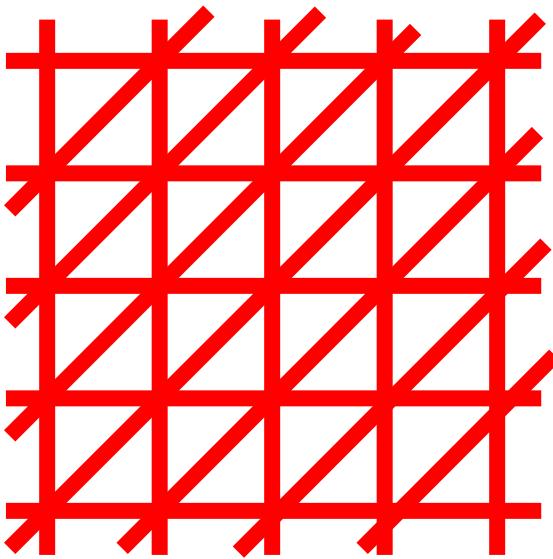
Implication for Mesh Storage

- Let's count the edges and faces in a closed **triangle** mesh:
 - Ratio of edges to faces: $e = 3/2 f$
 - each edge belongs to exactly 2 triangles
 - each triangle has exactly 3 edges
 - Ratio of vertices to faces: $f \sim 2v$
 - $2 = v - e + f = v - 3/2 f + f$
 - $2 + f / 2 = v$
 - Ratio of edges to vertices: $e \sim 3v$
 - **Average degree of a vertex:** 6



Regularity

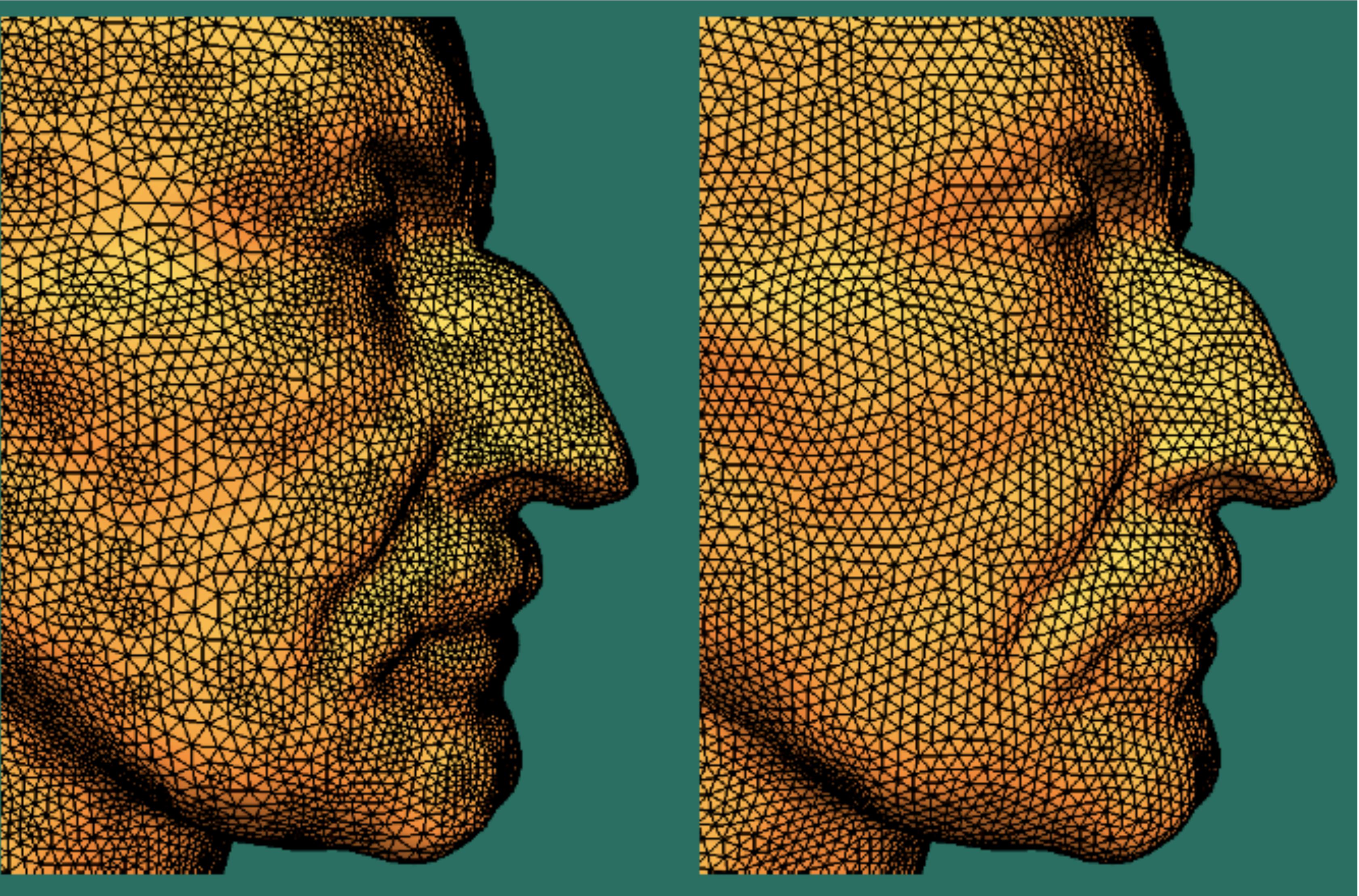
- Triangle mesh: average valence = 6
- Quad mesh: average valence = 4



- **Regular mesh:** all faces have the same number of edges and all vertex degrees are equal
- **Quasi-regular mesh:**
 - many vertices have regular degree 6 (4). Sometimes also refers to mostly equilateral faces.

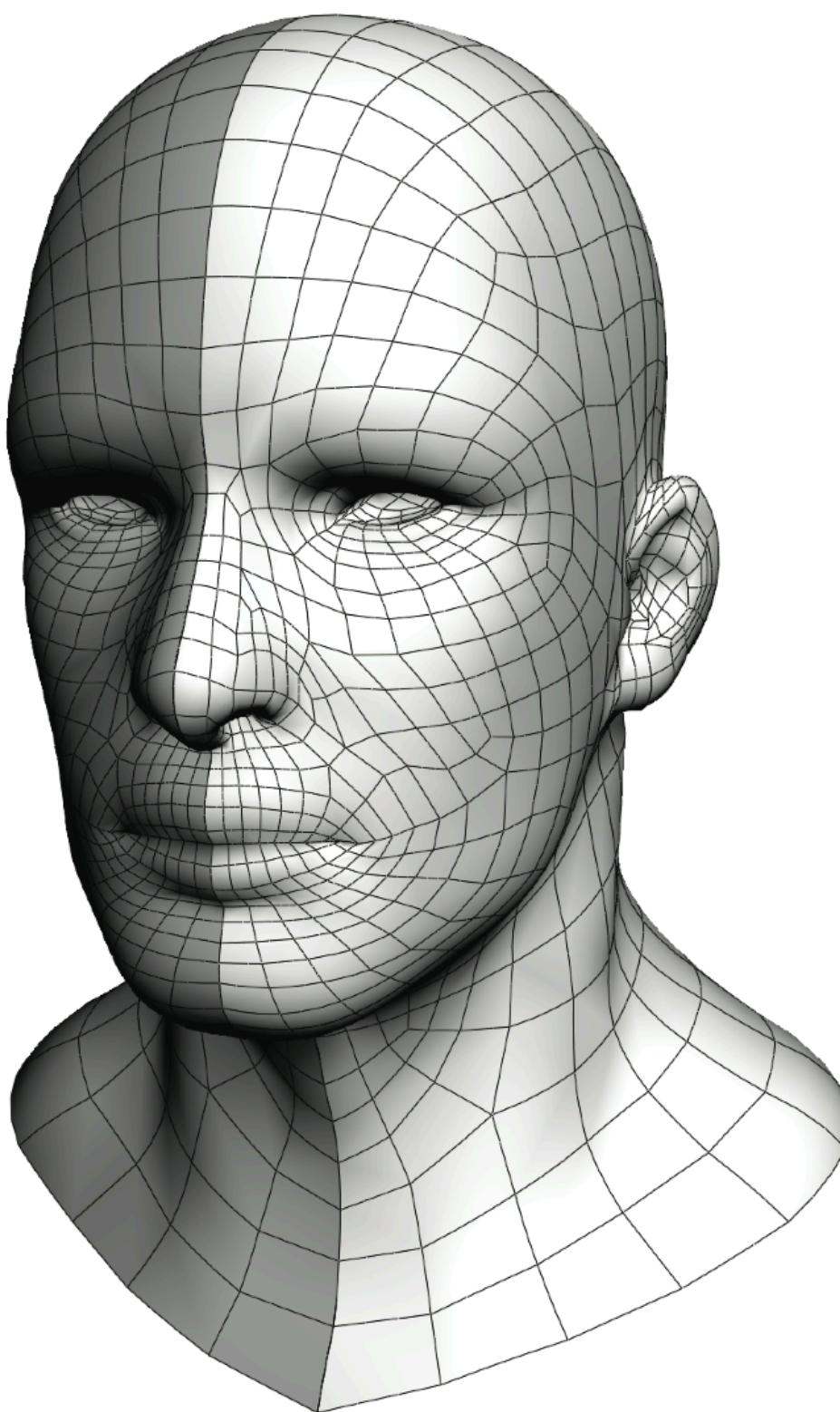
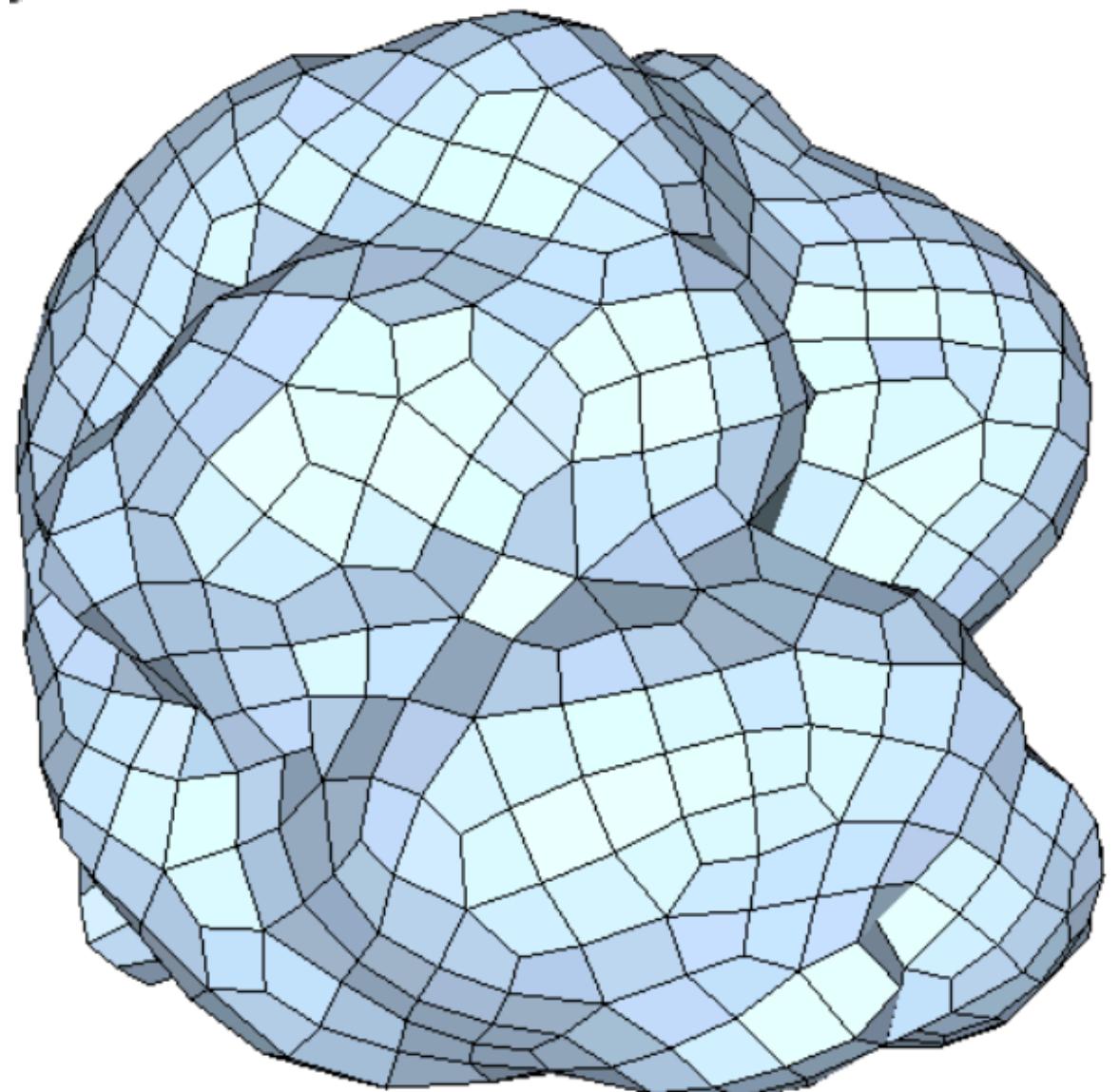
Regularity

- Quasi-regular



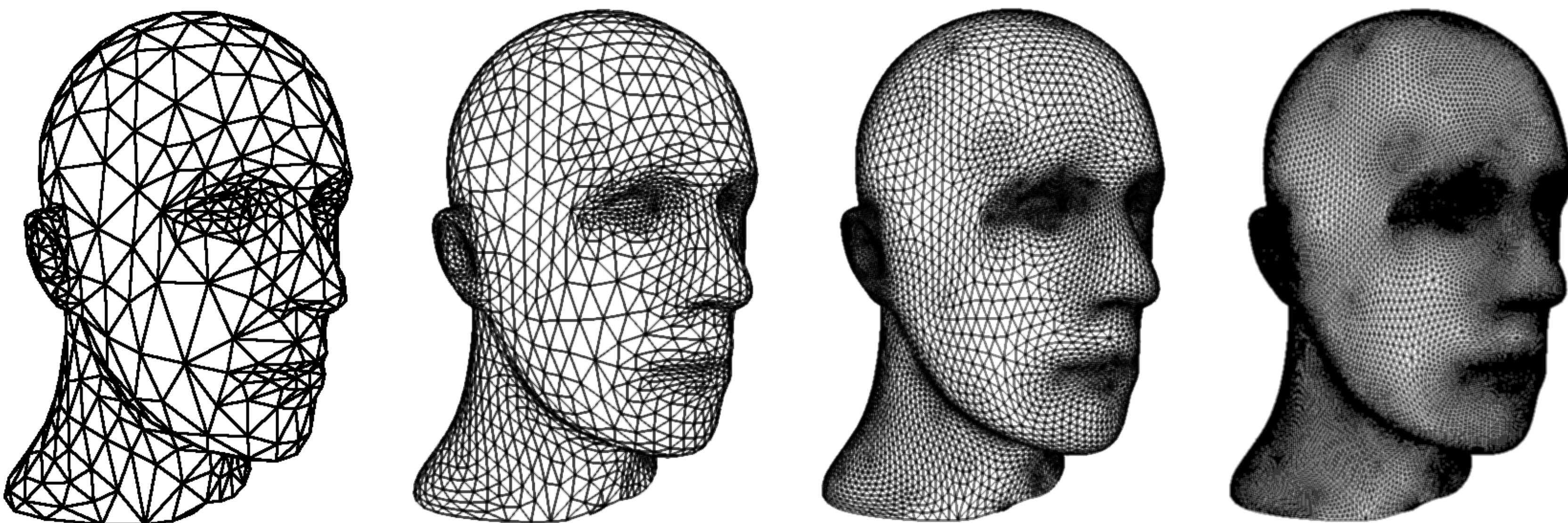
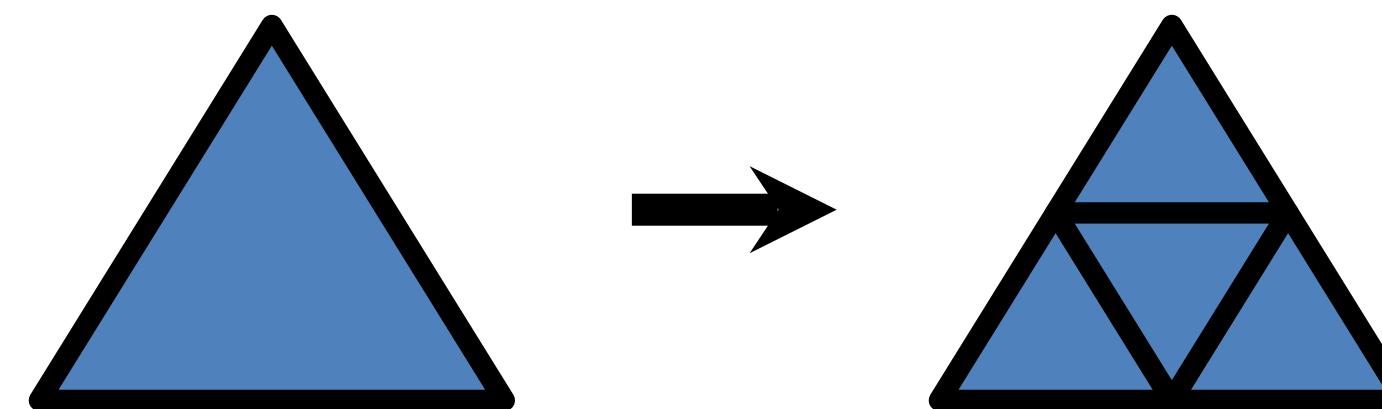
Regularity

- Quasi-regular



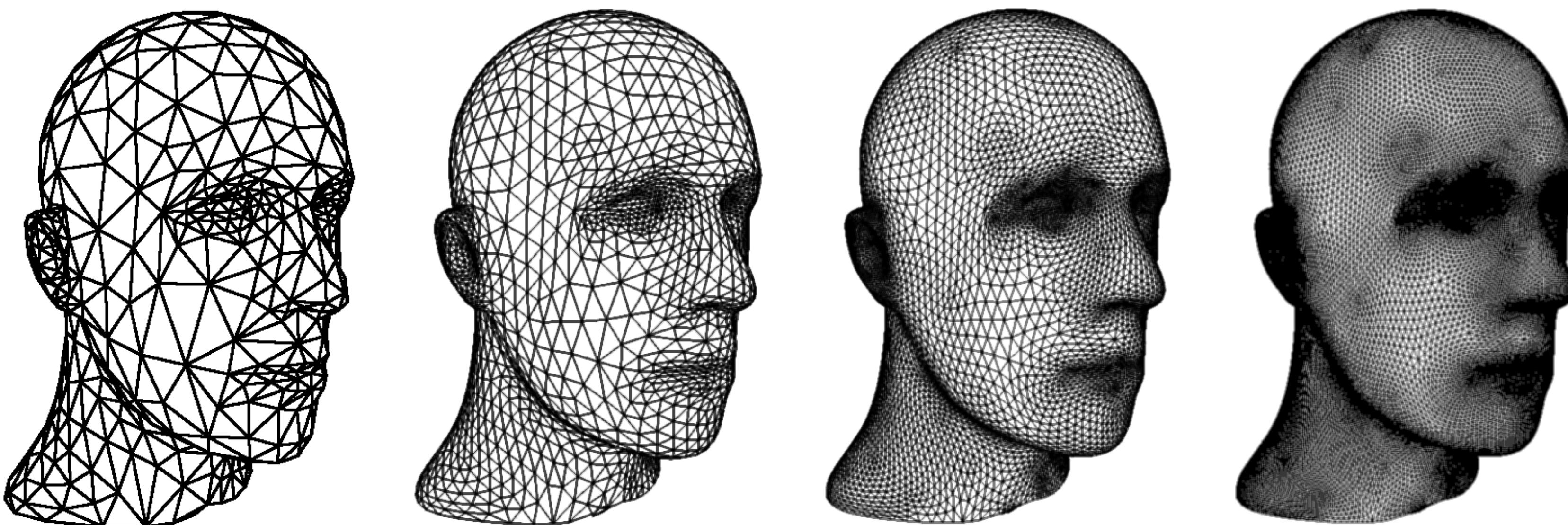
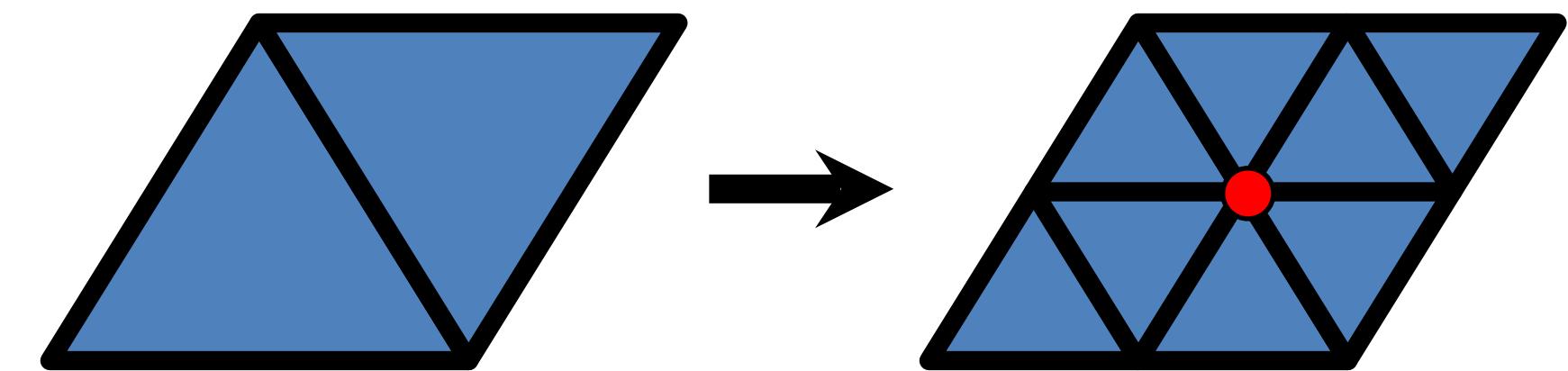
Regularity

- **Semi-regular mesh:**
connectivity is a result
of $N > 0$ subdivision steps

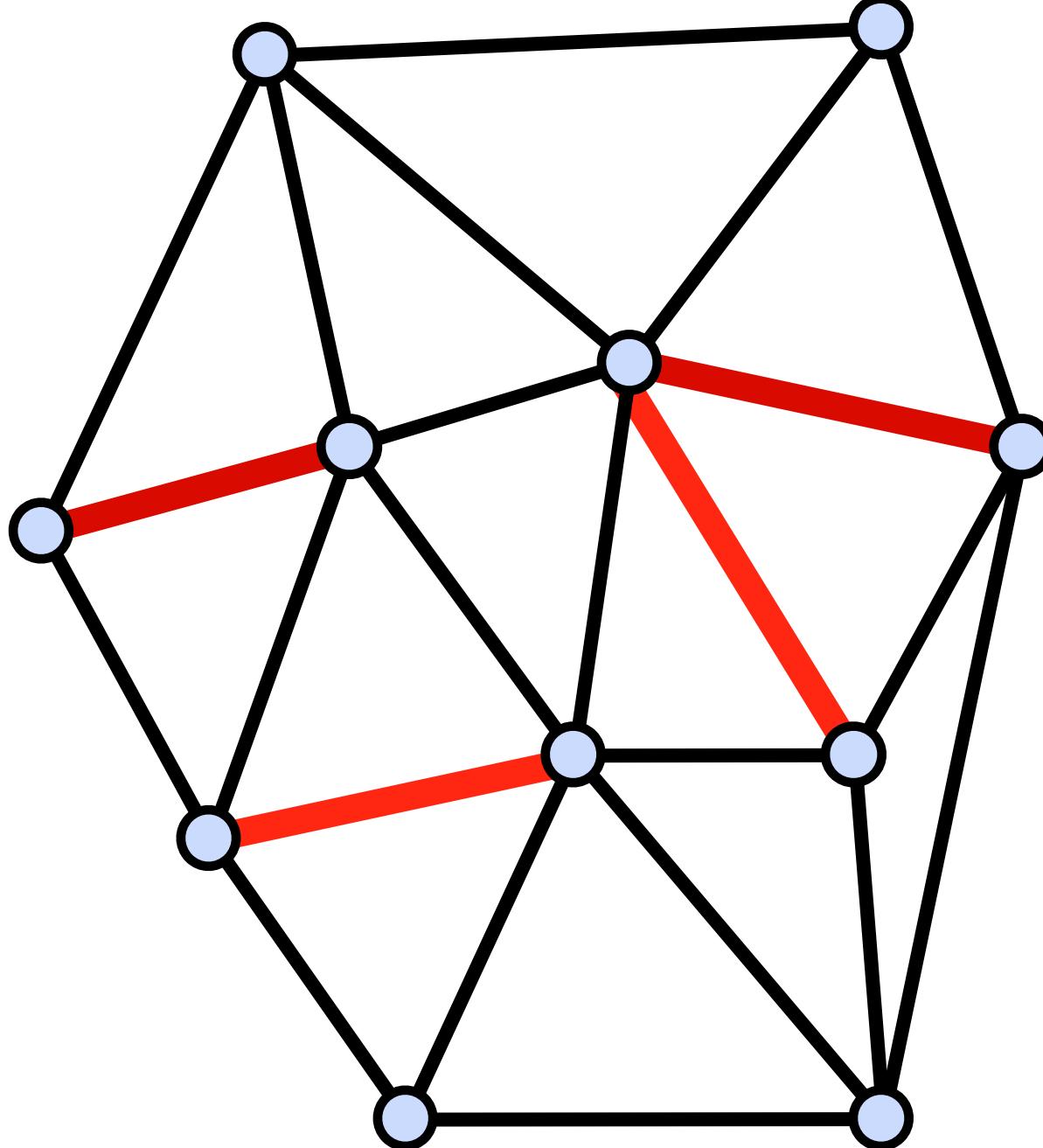


Regularity

- **Semi-regular mesh:**
connectivity is a result
of $N > 0$ subdivision steps



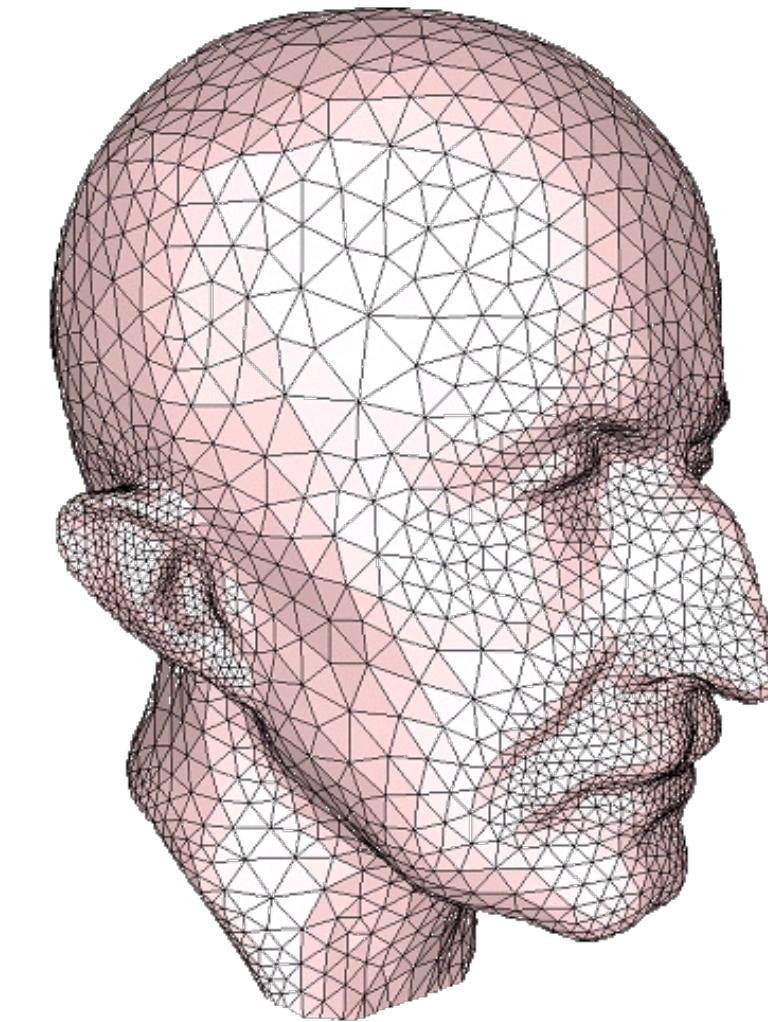
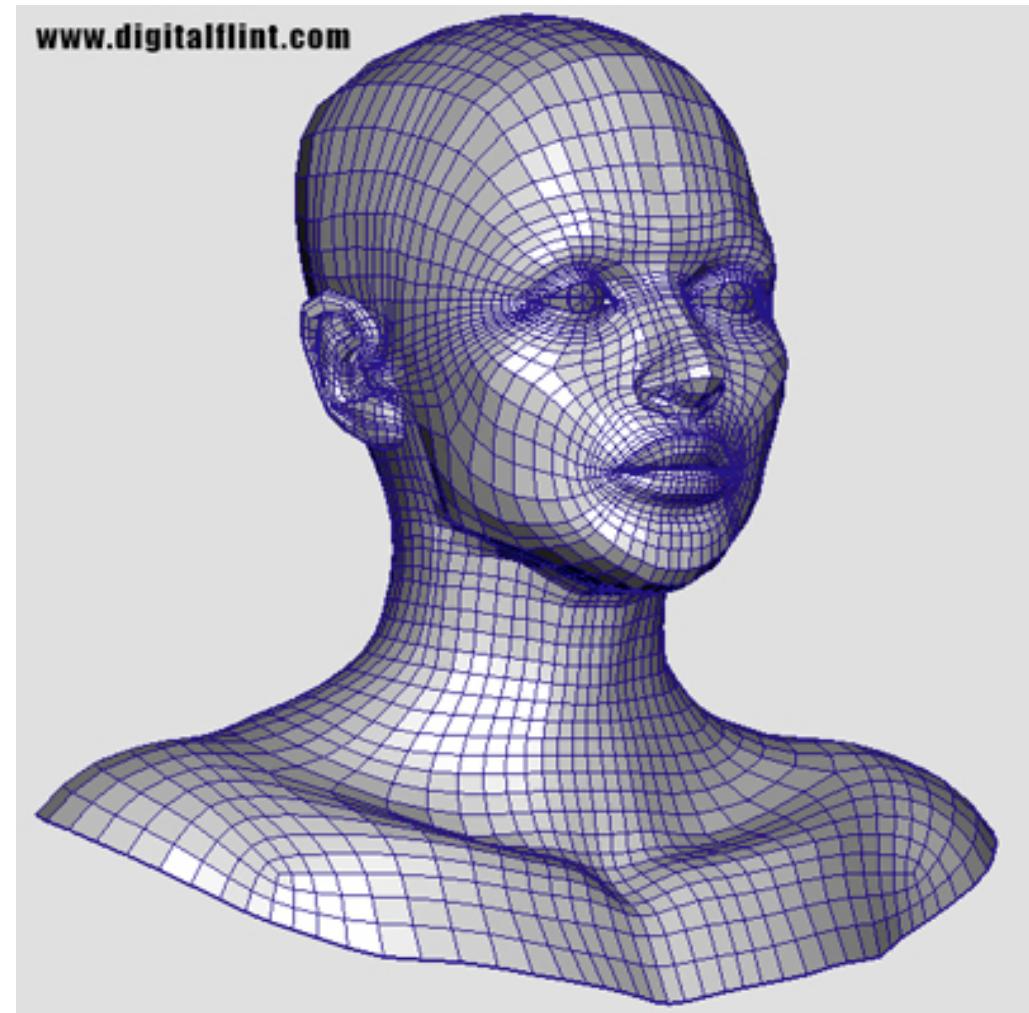
Triangulation



- Polygonal mesh where every face is a triangle
- Simplifies data structures
- Simplifies rendering
- Simplifies algorithms
- Each face planar and convex
- Any polygon can be triangulated

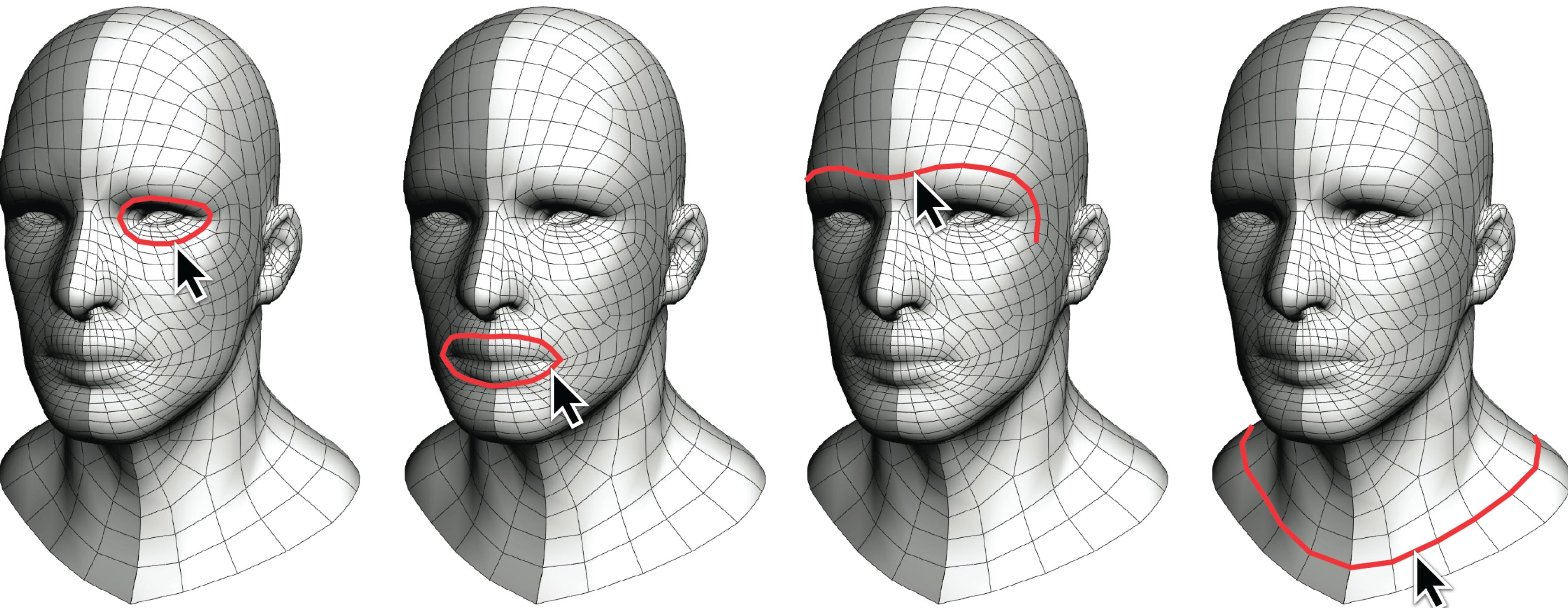
Polygonal vs. Triangle Meshes

- Triangles are flat and convex
 - Easy rasterization, normals
 - Uniformity (same # of vertices at all faces)
 - 3-way symmetry is less natural
-
- General polygons are flexible
 - Quads have natural symmetry
 - Can be non-planar, non-convex
 - Difficult for graphics hardware
 - Varying number of vertices
 - Fixed for pure quad meshes



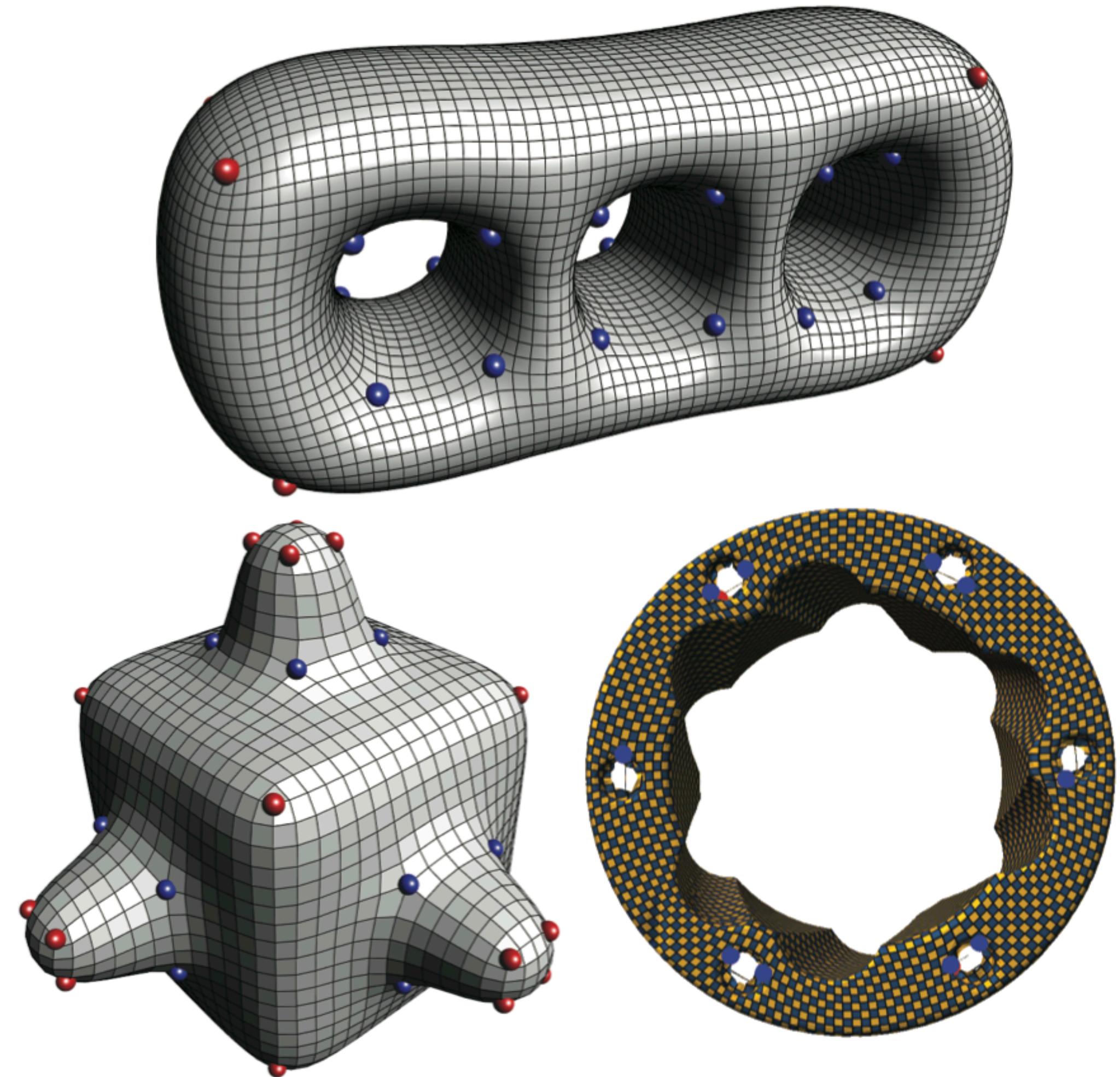
Polygonal vs. Triangle Meshes

- Edge loops are ideal for editing

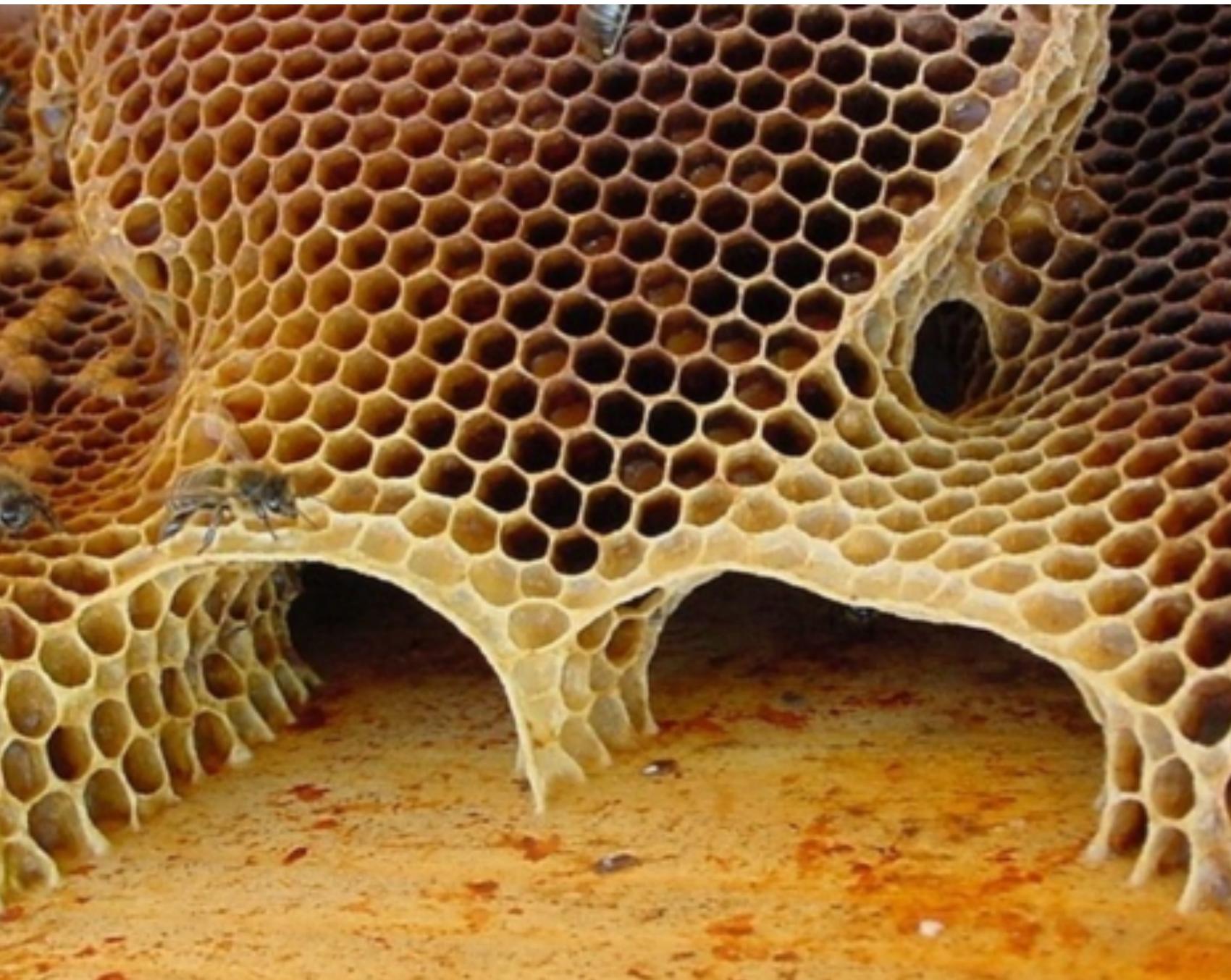


Polygonal vs. Triangle Meshes

- Quality of triangle meshes
 - Uniform Area
 - Angles close to 60
- Quality of quadrilateral meshes
 - Number of irregular vertices
 - Angles close to 90
 - Good edge flow
 - e.g., along curvature directions



Polygonal vs. Triangle Meshes

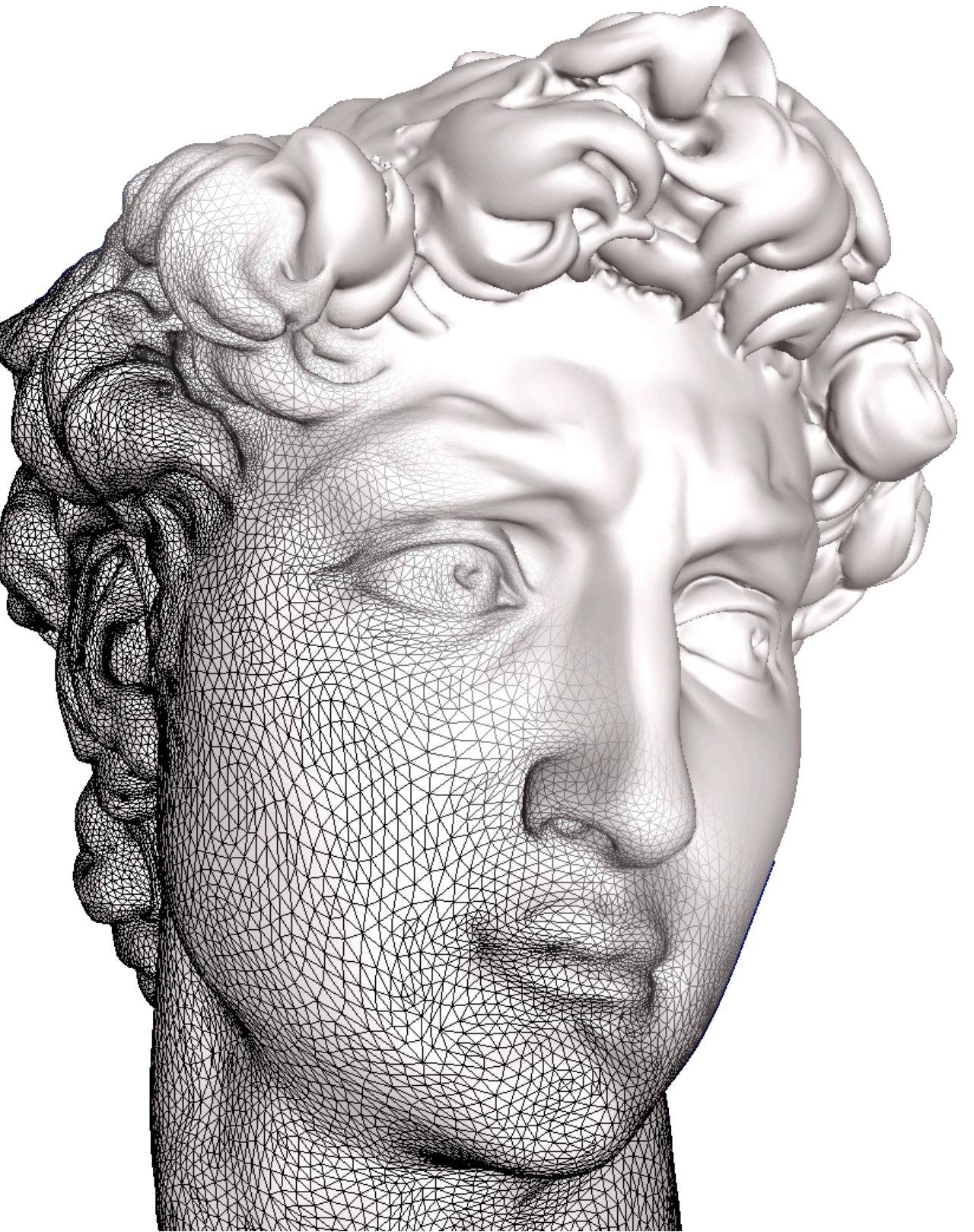


E. Van Egeraat

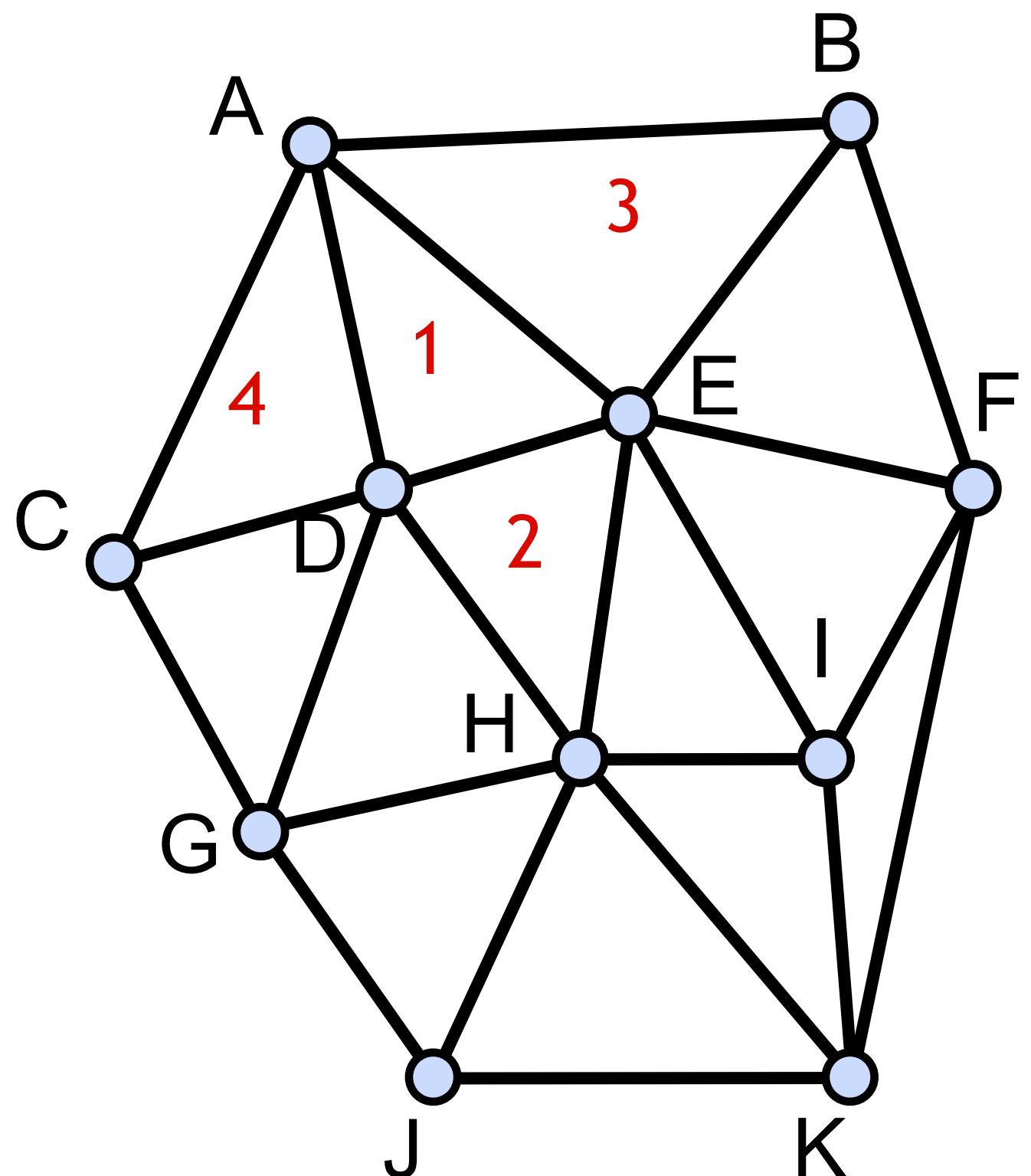
Data structures

Data Structures

- What should be stored?
 - Geometry: 3D coordinates
 - Connectivity
 - Adjacency relationships
 - Attributes
 - Normal, color, texture coordinates
 - Per vertex, face, edge

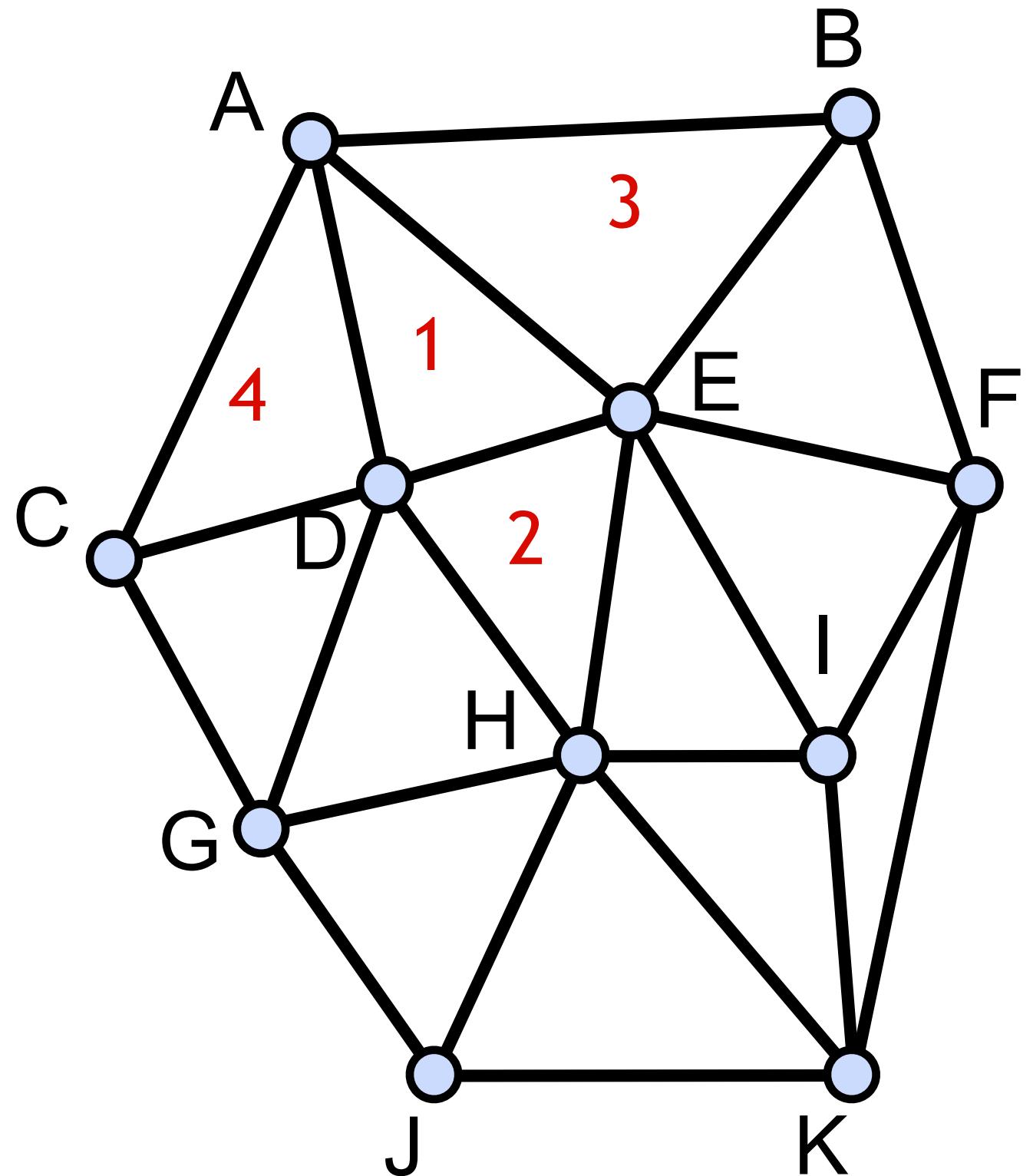


Data Structures



- What should be supported?
 - Rendering (enumeration)
 - Geometry queries
 - What are the vertices of face #2?
 - Is vertex A adjacent to vertex H?
 - Which faces are adjacent to face #1?
 - Modifications
 - Remove/add a vertex/face
 - Vertex split, edge collapse, ...

Data Structures



- How good is a data structure?
 - Time to construct
 - Time to answer a query
 - Time to perform an operation
 - Space complexity
 - Redundancy
- Criteria for design
 - Expected number of vertices
 - Available memory
 - Required operations
 - Distribution of operations

Triangle List (a.k.a. Polygon Soup)

- STL format (used in CAD)
- Storage
 - Face: 3 positions
 - 4 bytes per coordinate
 - 36 bytes per face
 - Euler: $f = 2v$
 - 72^*v bytes for a mesh with v vertices
- No connectivity information
- Redundant:
 - each vertex appears several times

Triangles			
0	x0	y0	z0
1	x1	x1	z1
2	x2	y2	z2
3	x3	y3	z3
4	x4	y4	z4
5	x5	y5	z5
6	x6	y6	z6
...

Indexed Face Set

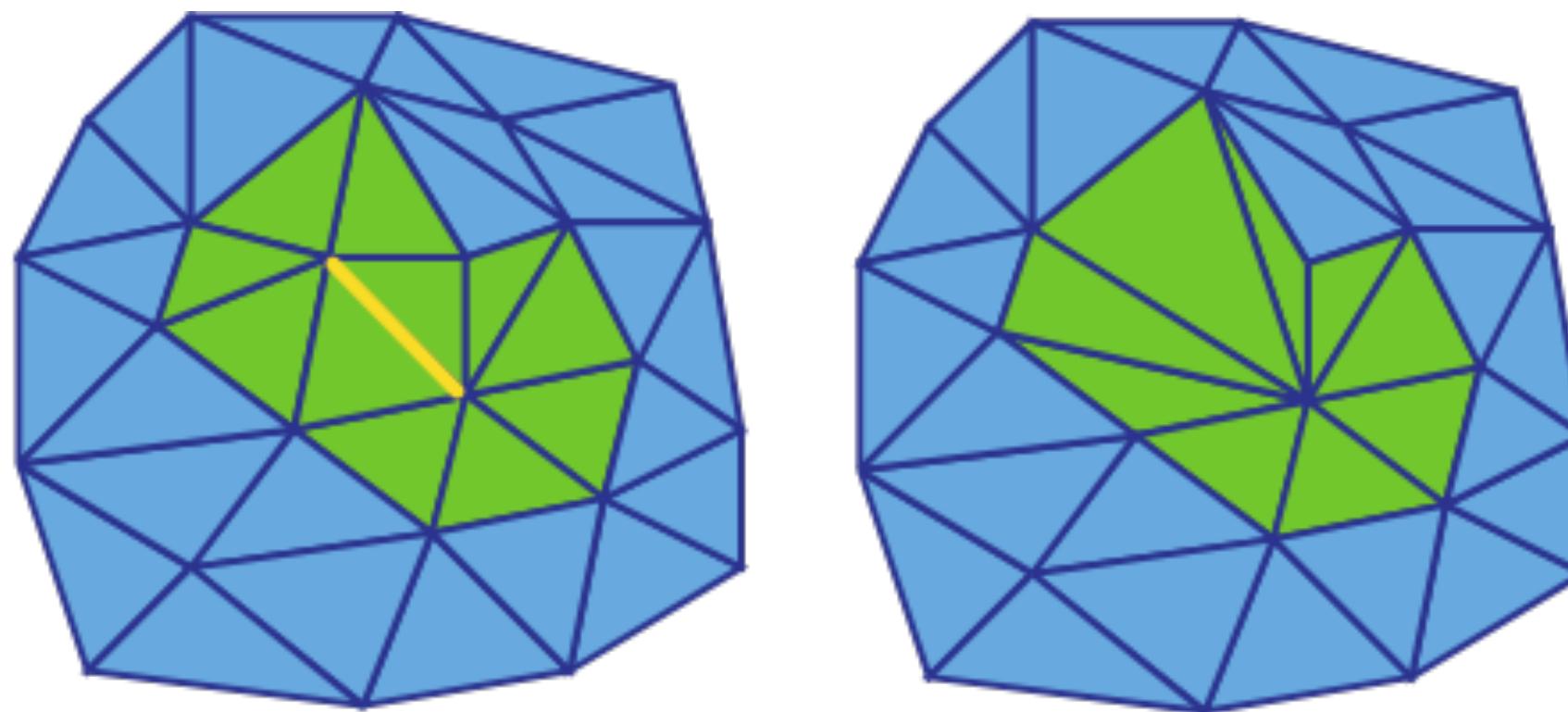
- Used in file formats
OBJ, OFF, WRL
- Storage
 - Vertex: position
 - Face: vertex indices
 - 12 bytes per vertex
 - 12 bytes per face
 - $36 \cdot v$ bytes for the mesh
- No *explicit* neighborhood info

Vertices			
v0	x0	y0	z0
v1	x1	x1	z1
v2	x2	y2	z2
v3	x3	y3	z3
v4	x4	y4	z4
v5	x5	y5	z5
v6	x6	y6	z6
...

Triangles			
t0	v0	v1	v2
t1	v0	v1	v3
t2	v2	v4	v3
t3	v5	v2	v6
...

Indexed Face Set: Problems

- Information about neighbors is not explicit
 - Finding neighboring vertices/edges/faces costs $O(\#V)$ time!
 - Local mesh modifications cost $O(\#V)$



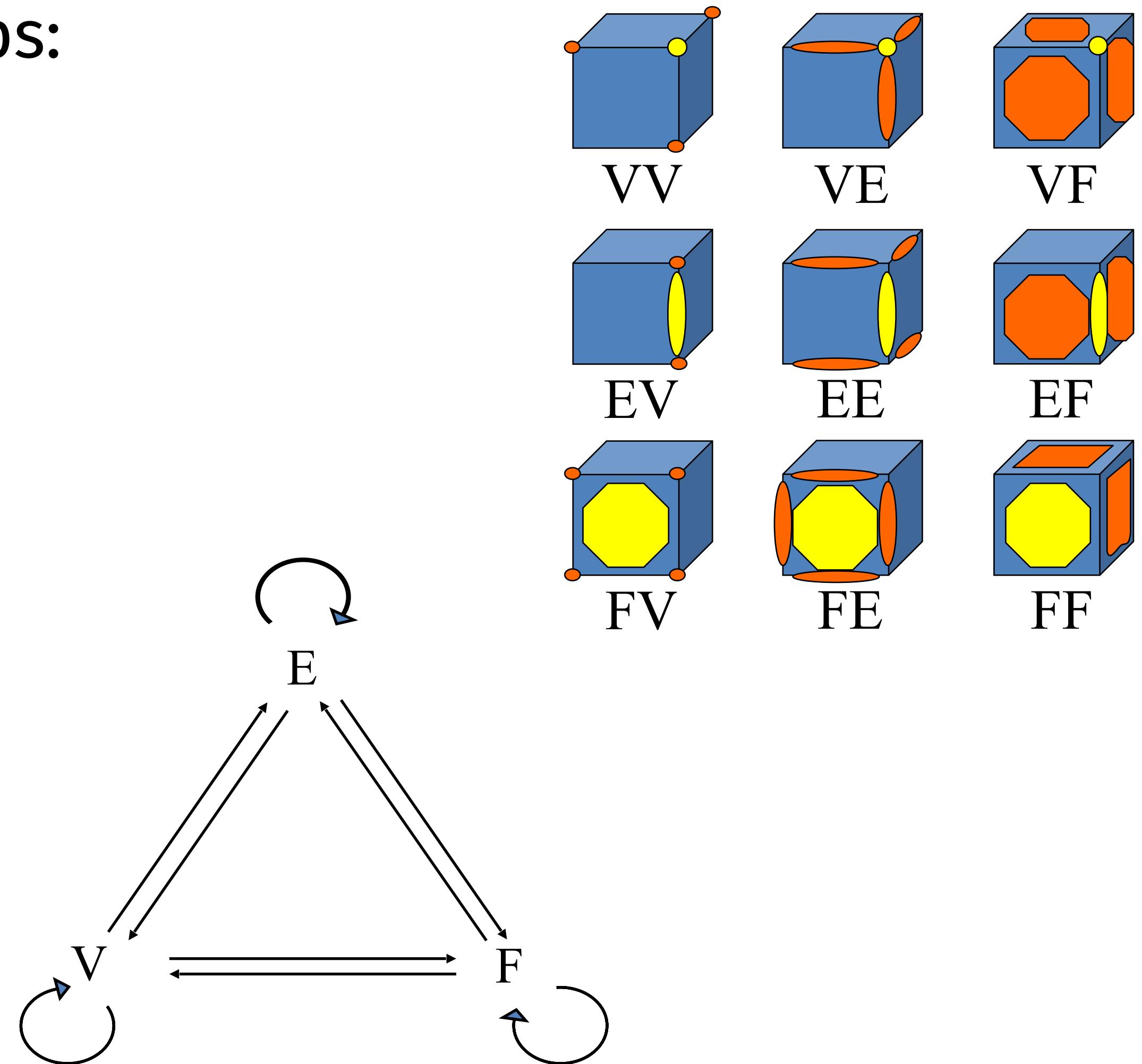
- Breadth-first search costs $O(k * \#V)$ where $k = \#$ found vertices

Neighborhood Relations

- All possible neighborhood relationships:

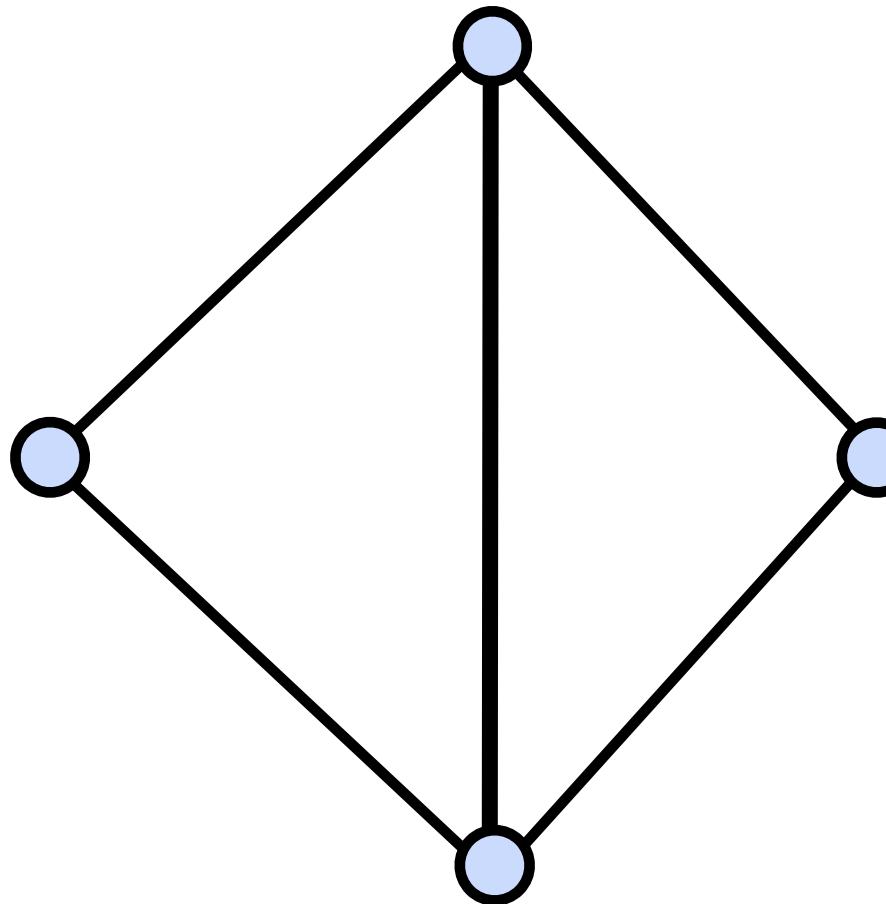
1. Vertex – Vertex VV
2. Vertex – Edge VE
3. Vertex – Face VF
4. Edge – Vertex EV
5. Edge – Edge EE
6. Edge – Face EF
7. Face – Vertex FV
8. Face – Edge FE
9. Face – Face FF

We'd like $O(k)$ time for queries and local updates of these relationships
(k = number of found entities)



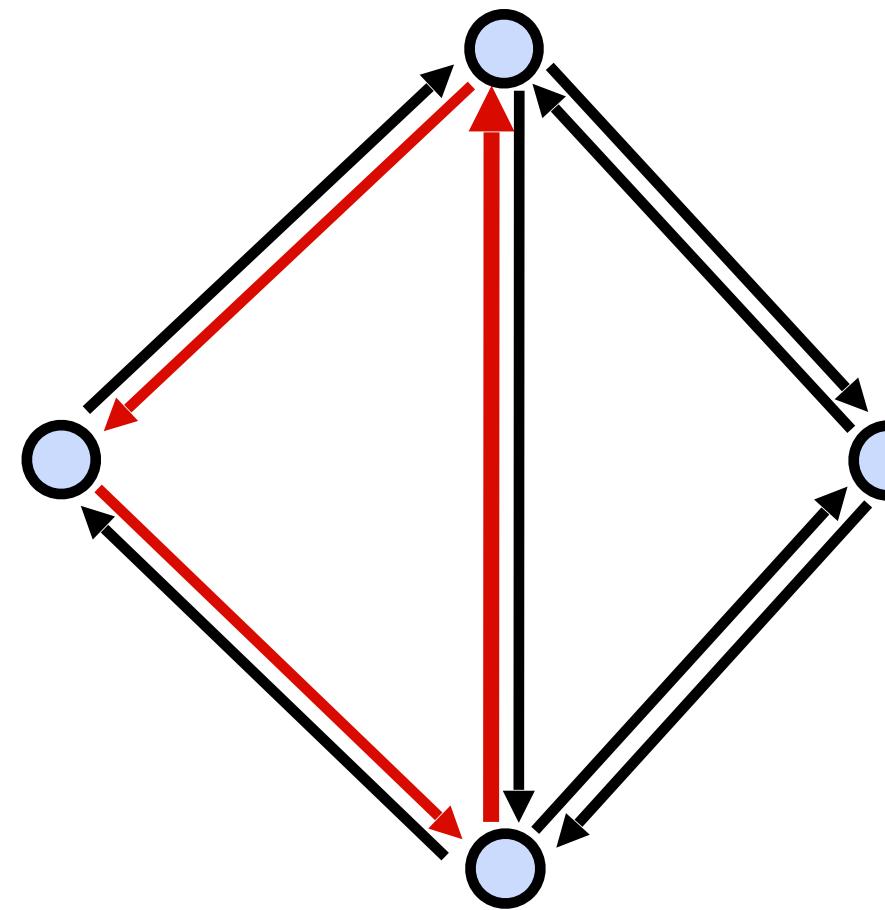
Half-edge data structure

- Introduce orientation into data structure
 - Oriented edges



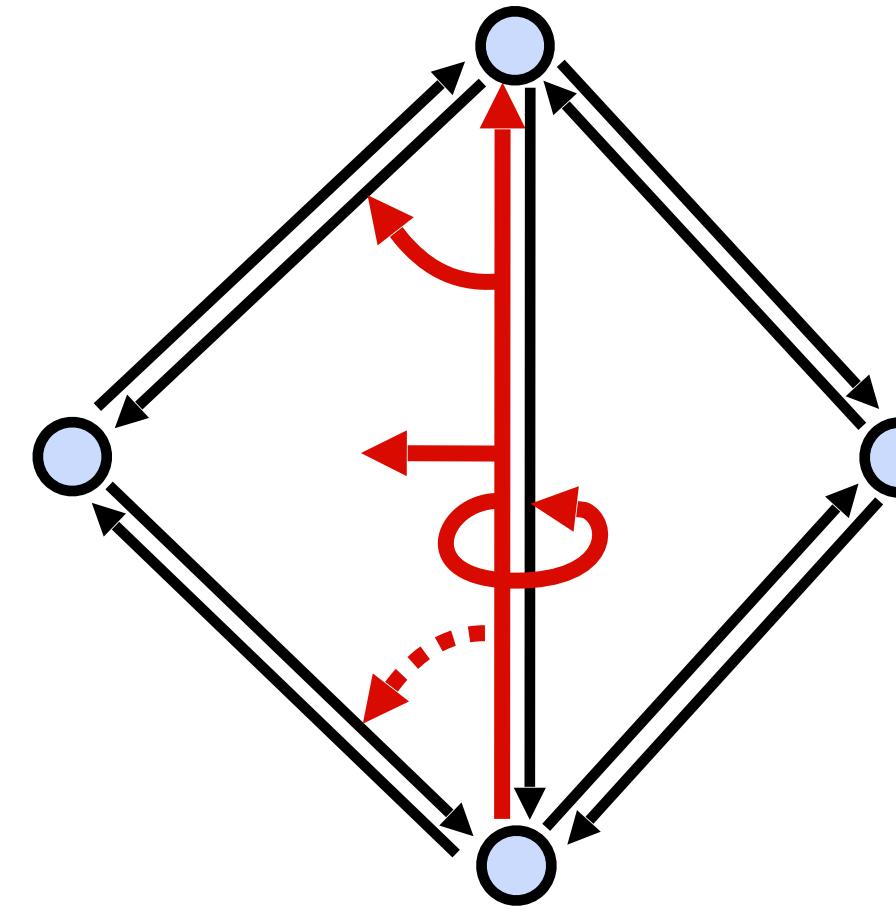
Half-edge data structure

- Introduce orientation into data structure
 - Oriented edges



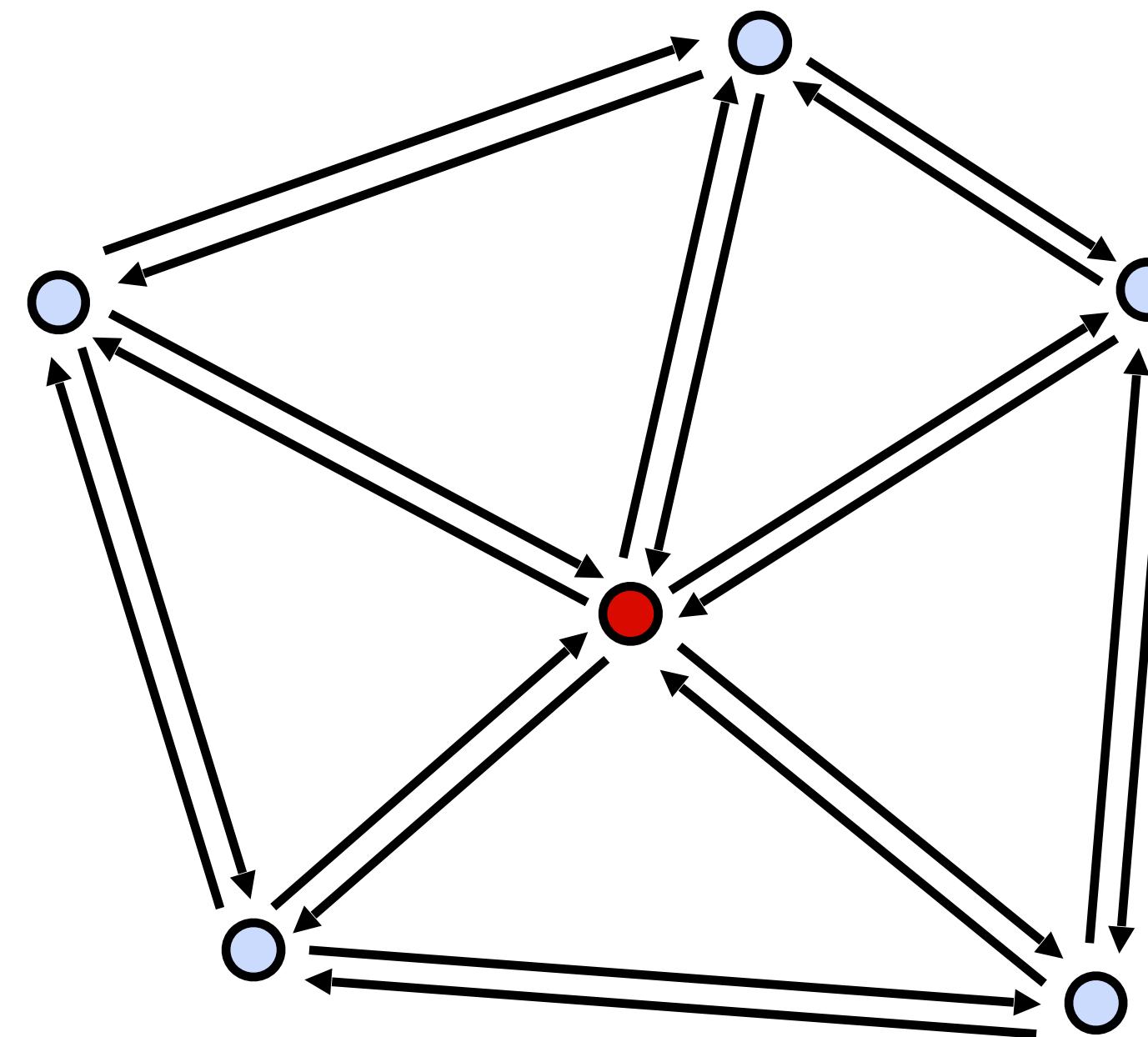
Half-edge data structure

- Introduce orientation into data structure
 - Oriented edges
- Vertex
 - Position
 - 1 outgoing halfedge index
- Halfedge
 - 1 origin vertex index
 - 1 incident face index
 - 3 next, prev, twin halfedge indices
- Face
 - 1 incident halfedge index
- Easy traversal, full connectivity



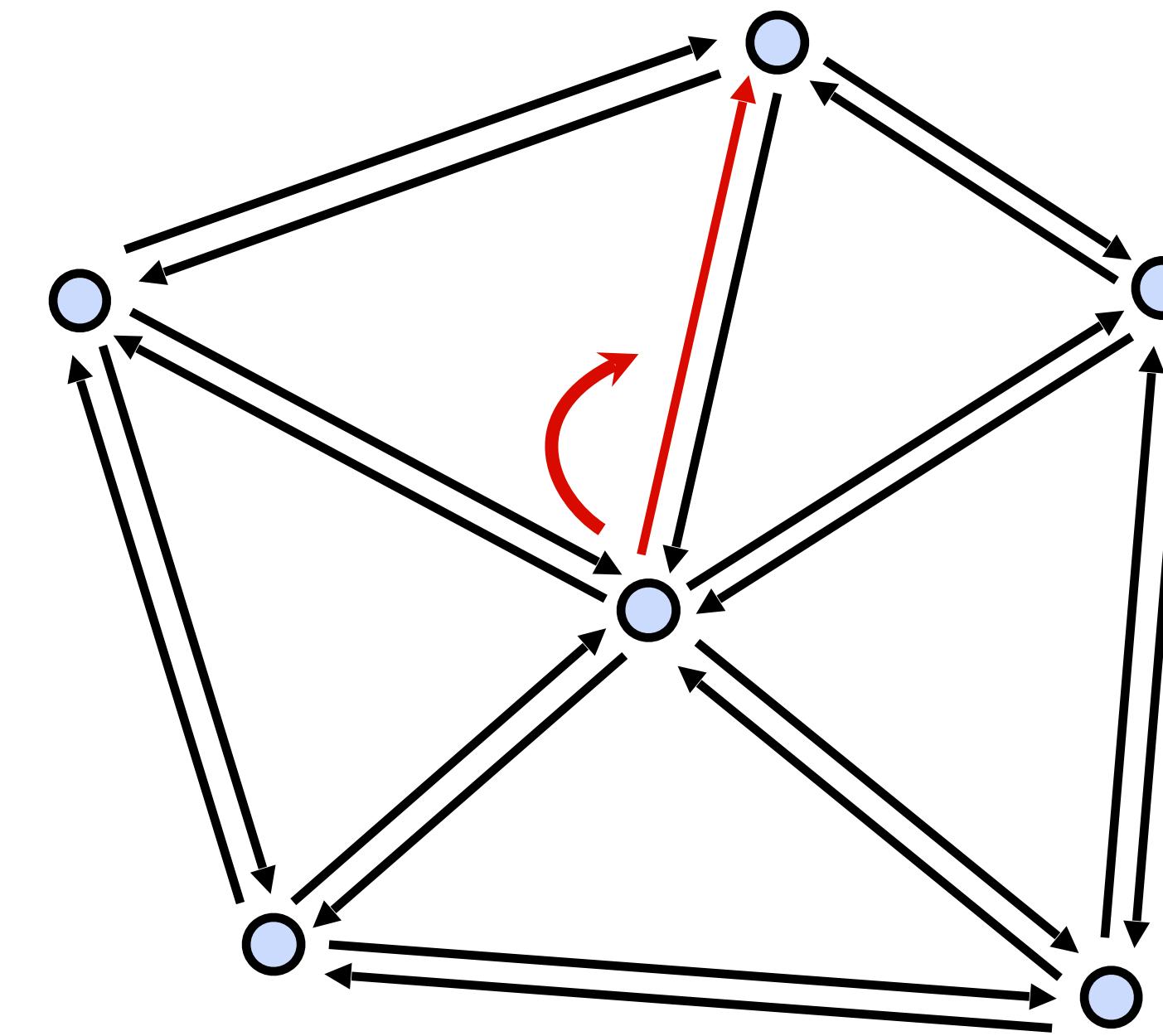
Half-edge data structure

- One-ring traversal
 - Start at vertex



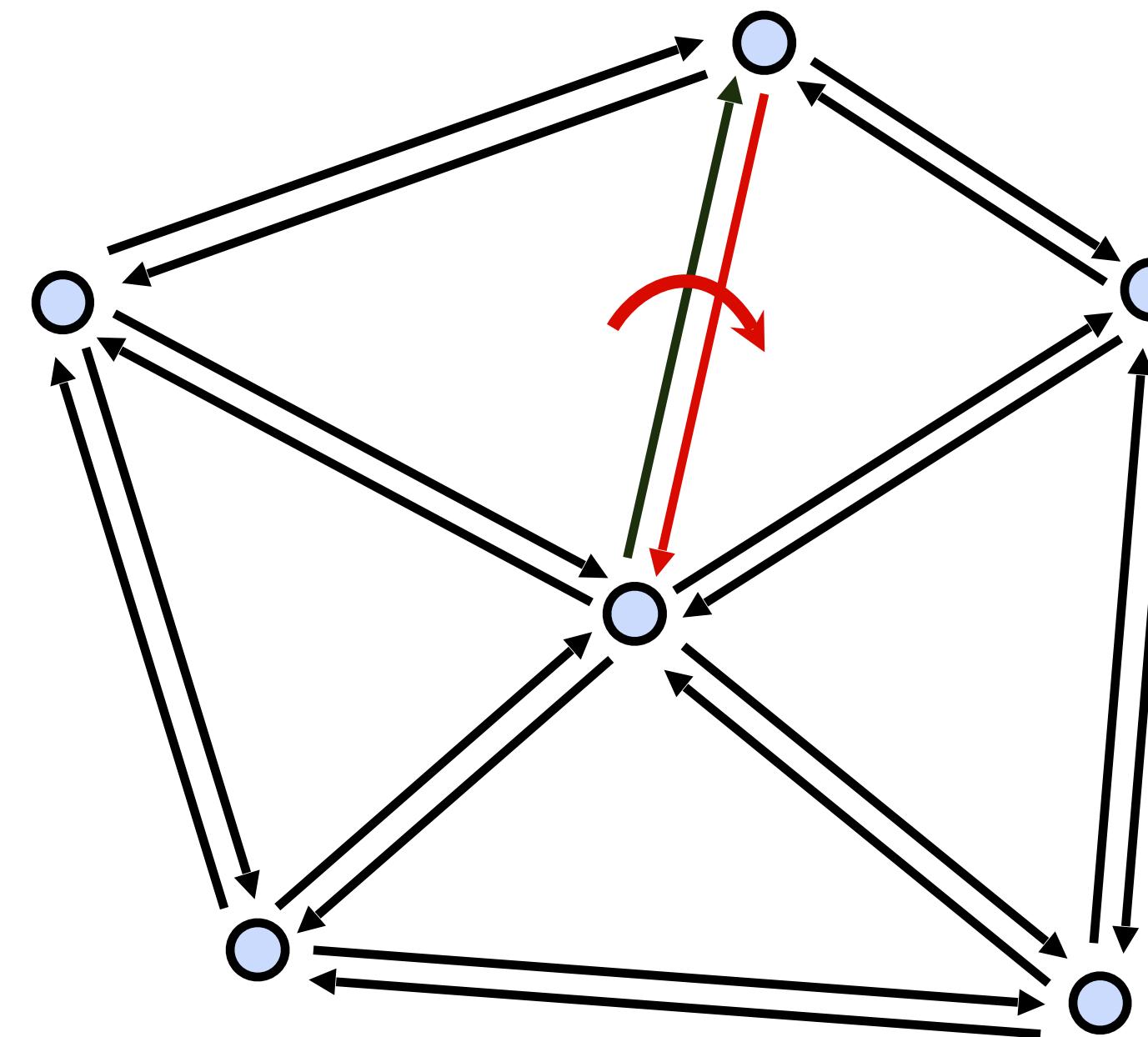
Half-edge data structure

- One-ring traversal
 - Start at vertex
 - Outgoing halfedge



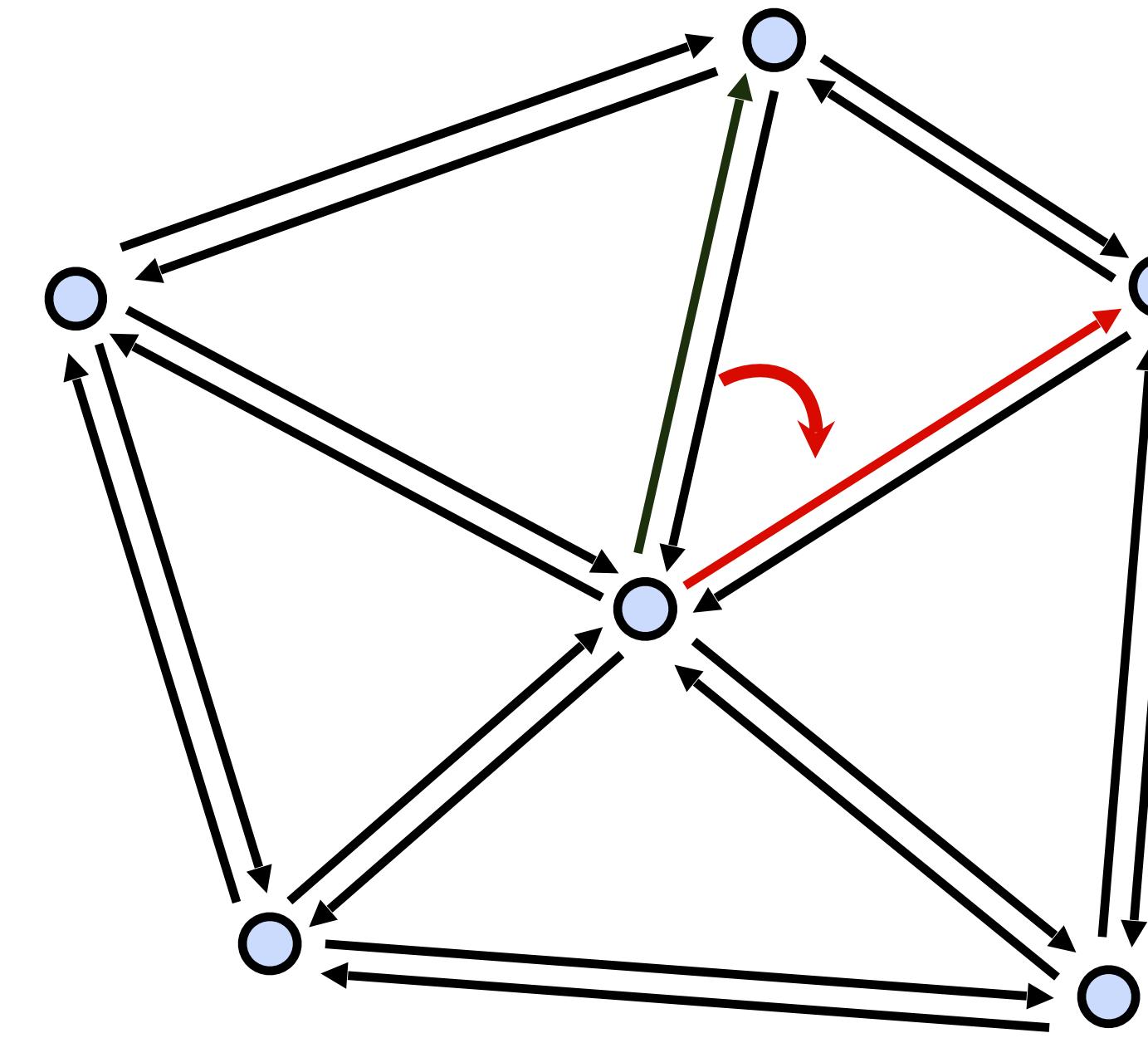
Half-edge data structure

- One-ring traversal
 - Start at vertex
 - Outgoing halfedge
 - Twin halfedge



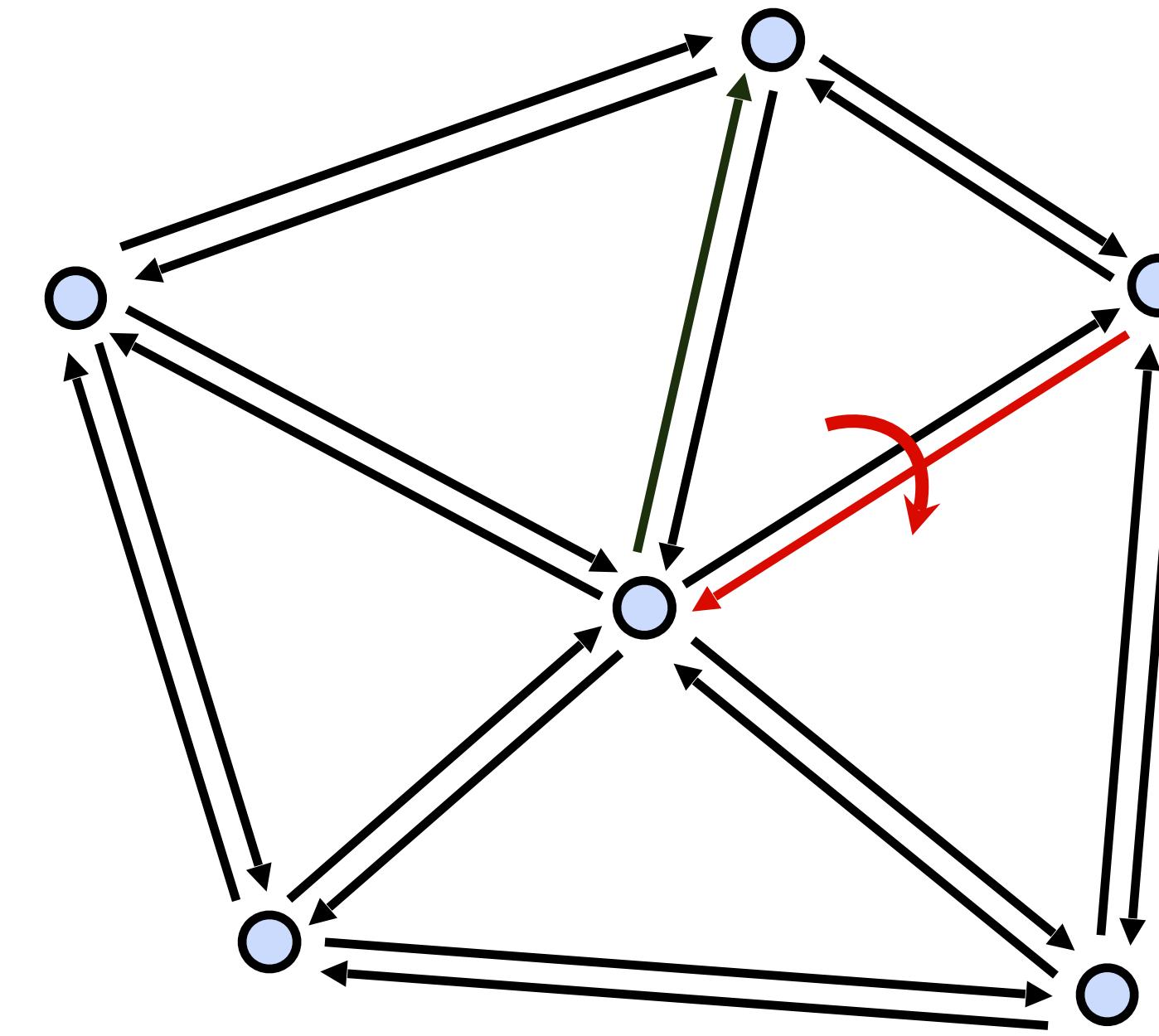
Half-edge data structure

- One-ring traversal
 - Start at vertex
 - Outgoing halfedge
 - Twin halfedge
 - Next halfedge



Half-edge data structure

- One-ring traversal
 - Start at vertex
 - Outgoing halfedge
 - Twin halfedge
 - Next halfedge
 - Twin ...



Half-edge data structure

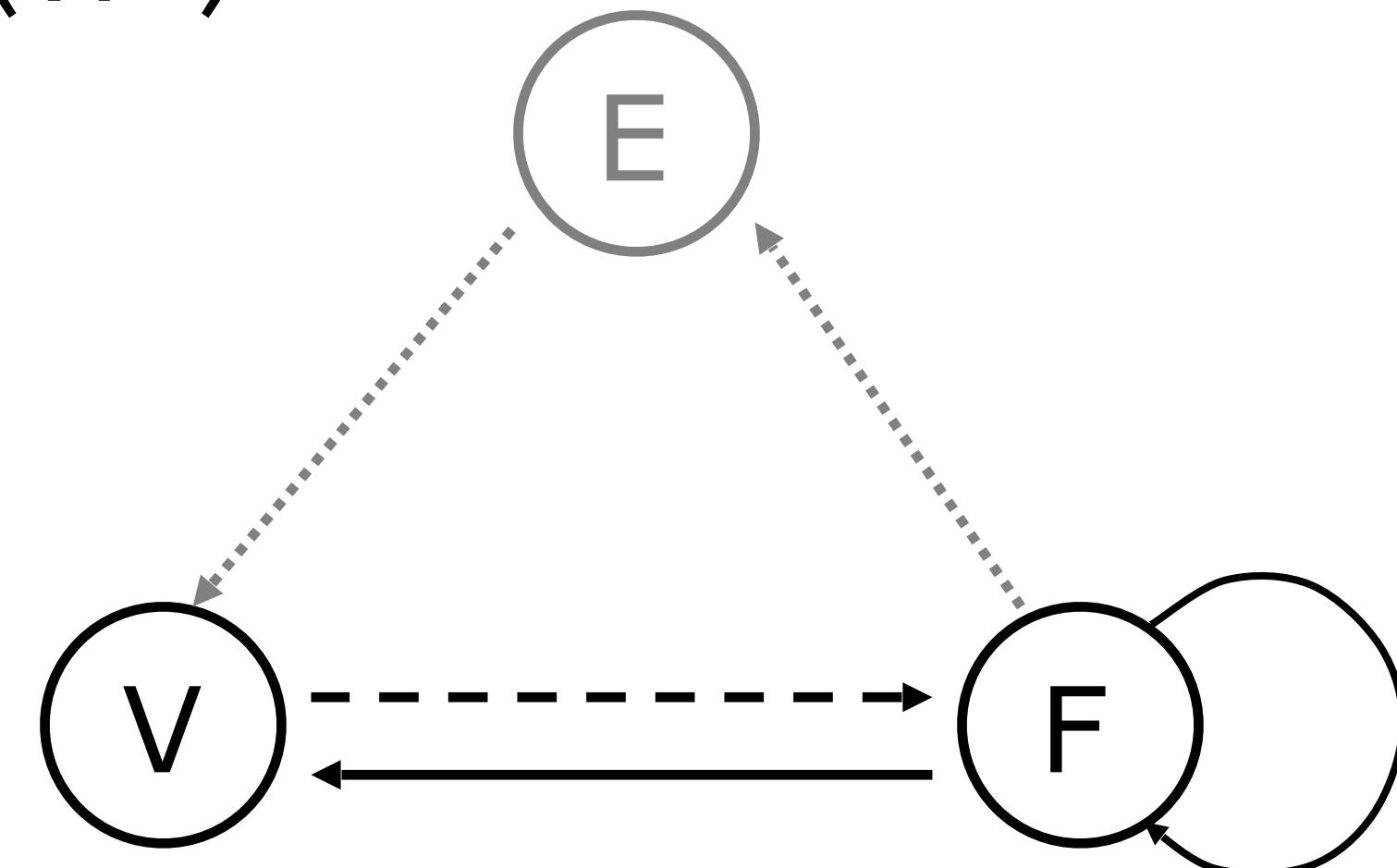
- Pros: *(assuming bounded vertex valence)*
 - $O(k)$ time for neighborhood relationship queries
 - $O(k)$ time and space for local modifications (edge collapse, vertex insertion...)
- Cons:
 - Heavy – requires storing and managing extra pointers
 - $(60*\nu - 84*\nu$ bytes for the mesh)
 - Not as trivial as Indexed Face Set for rendering with OpenGL / Vertex Buffer Objects

Half-edge Libraries

- CGAL
 - www.cgal.org
 - Computational geometry
- OpenMesh
 - www.openmesh.org
 - Mesh processing
- We will not implement a half-edge data structure in the class.
Instead we will work with Indexed Face Set and augment it to have fast queries.

Indexed Face Set with *Adjacencies*

- Same basis of the indexed structure:
 - array of vertices
 - array of faces, each pointing to its three vertices (FV)
- Plus:
 - Each face points to its adjacent faces (FF)
 - Each vertex points to one of its incident faces (VF*)



Indexed Face Set with *Adjacencies*

Evaluation of topological relations:

- FV, FF encoded – optimal
- FE implicit
- VF = VF* + FF + FV optimal (as exercise, similar to symmetric, easier)
- VV, VE: similar to VF, optimal by considering that all faces are triangles
- Relations based on edges (edges exist just implicitly as pairs of vertices):
 - EV implicit
 - EF/EE: optimal by encoding an edge ad a pair (f,i) where i is the index of e about f (encoding is not unique, as there are two incident faces)

Other Libraries

- VCG
 - <http://vcg.isti.cnr.it/vcglib/>
 - Based on Indexed Face Set with Adjacencies (triangles only)
 - Mesh processing
- igl
 - <https://github.com/libigl/libigl> <https://libigl.github.io/>
 - Mesh processing
 - Based on Indexed Face Set, augmented on request
 - This will be our choice in class

igl tutorial

Thank you