# Manifold and Mesh

USTC, 2024 Spring

Qing Fang, fq1208@mail.ustc.edu.cn

https://qingfang1208.github.io/

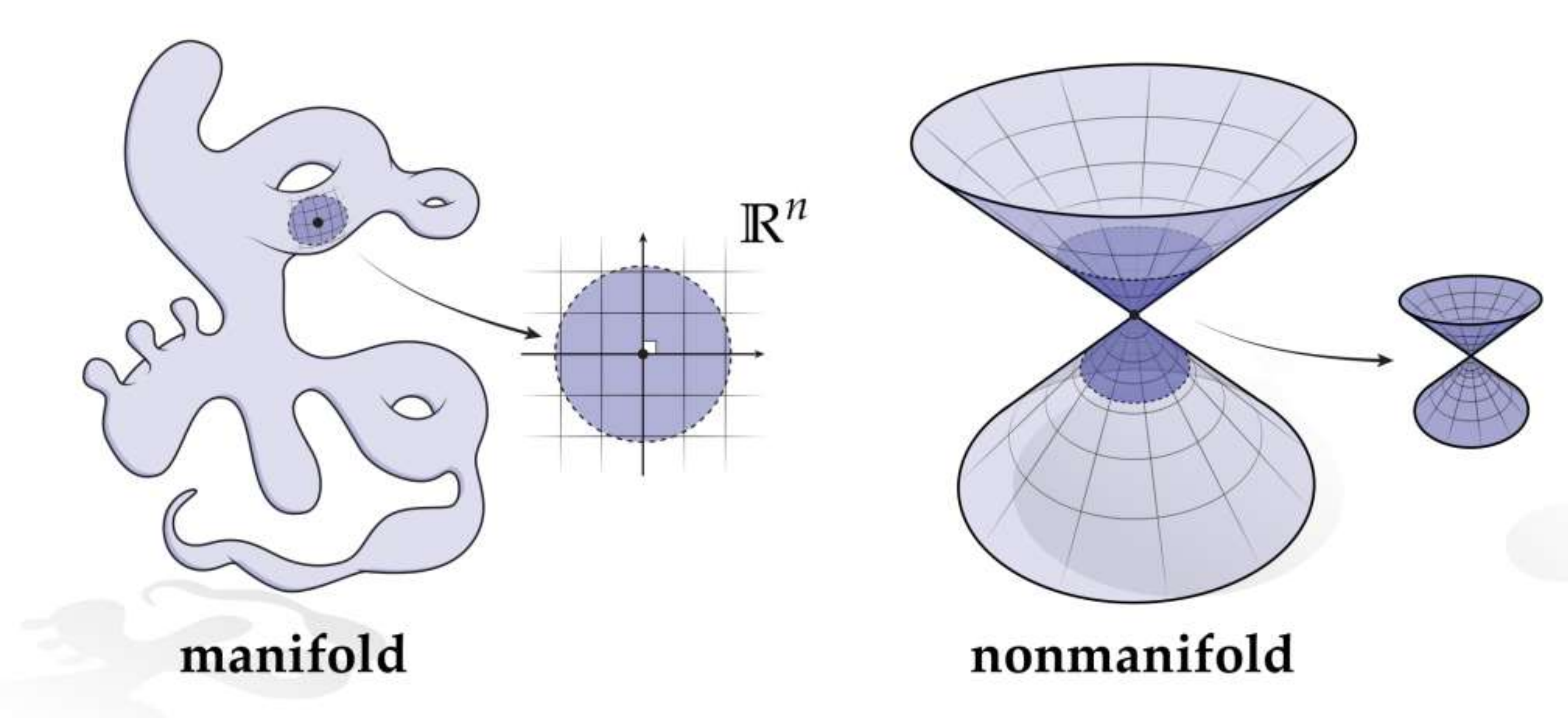
# Manifolds

# $n$-manifolds

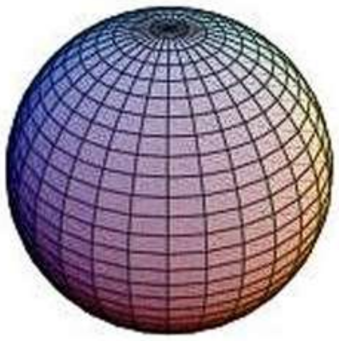
- A topological space $\mathcal{M}$ with the property that each point has a neighborhood that is homeomorphic to an open subset of $\mathbb{R}^n$.

- Homeomorphic: exists continuous and bijective function.

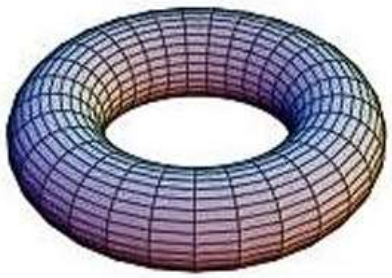
# Key idea



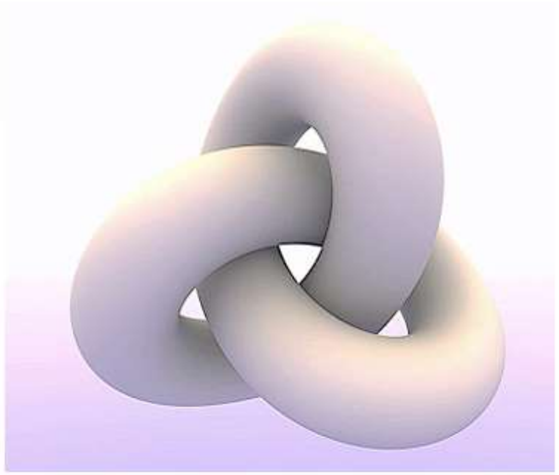
# Examples



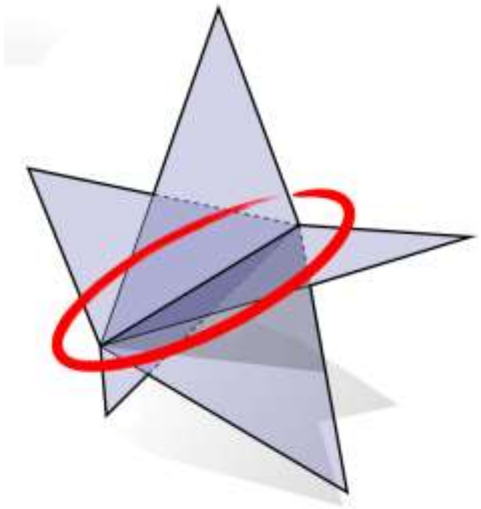
Case 1



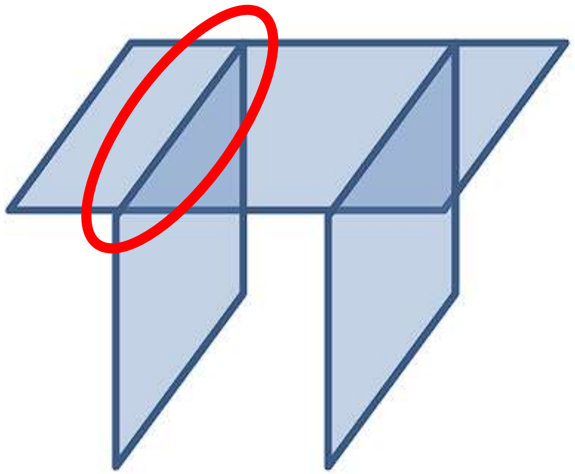
Case 2



Case 3

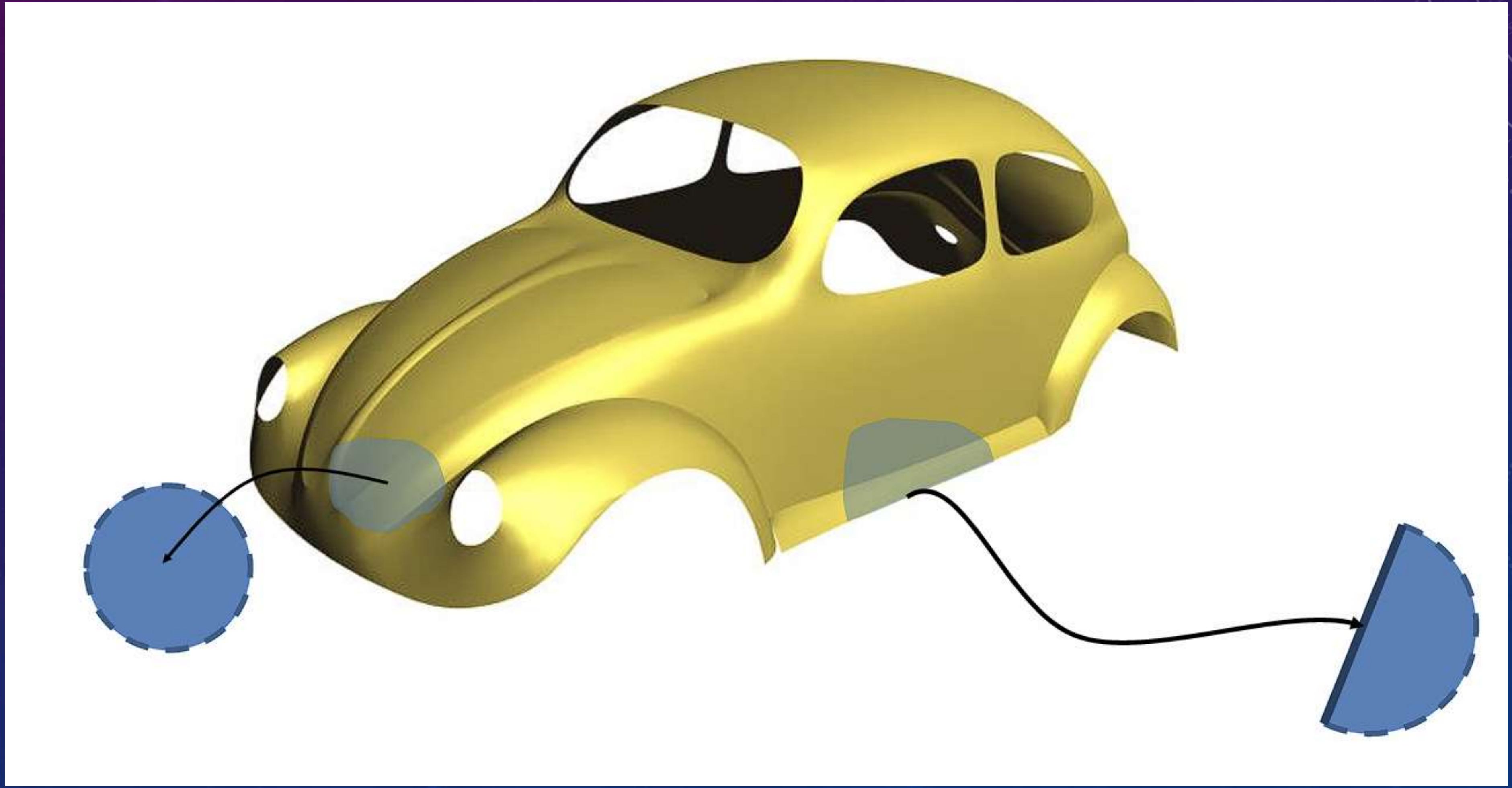


Case 4



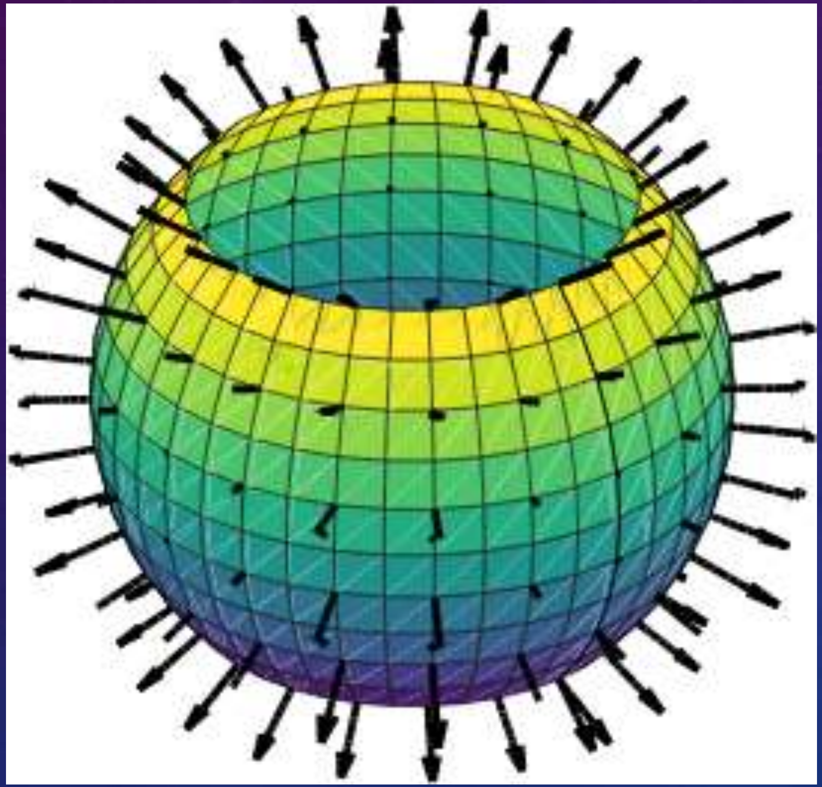
Case 5

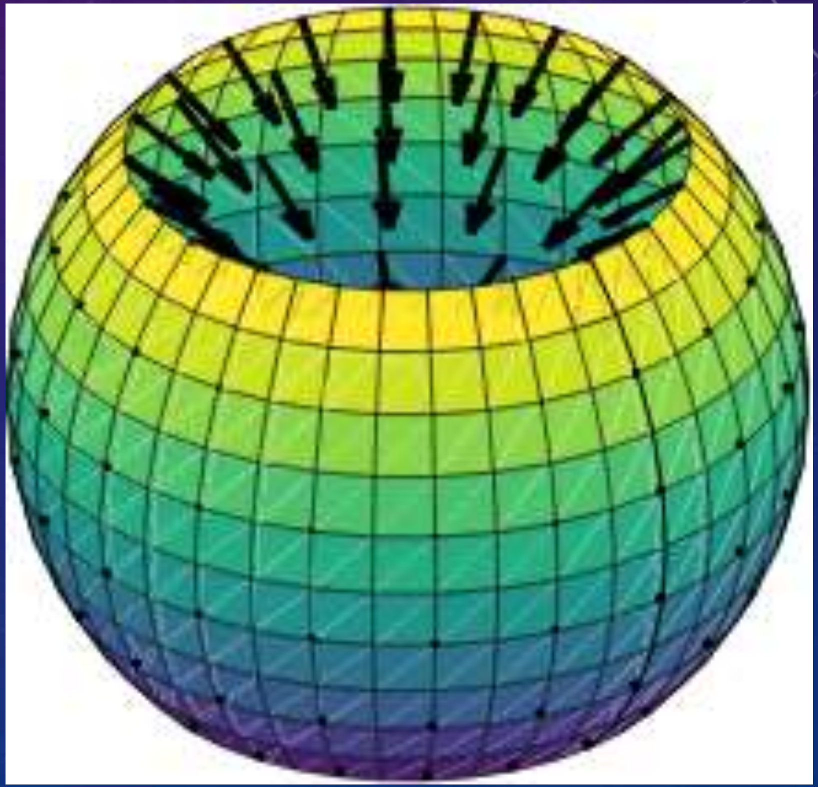
# Manifold with boundary



# Orientable manifolds

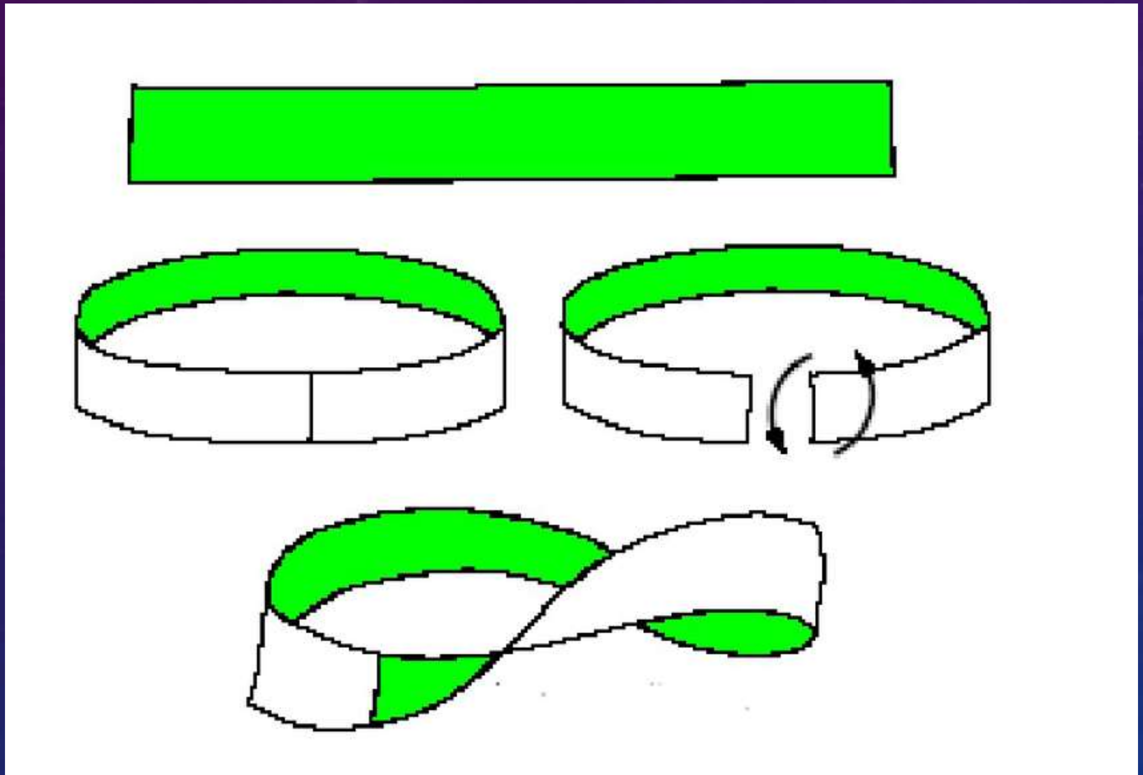
# Orientable manifolds

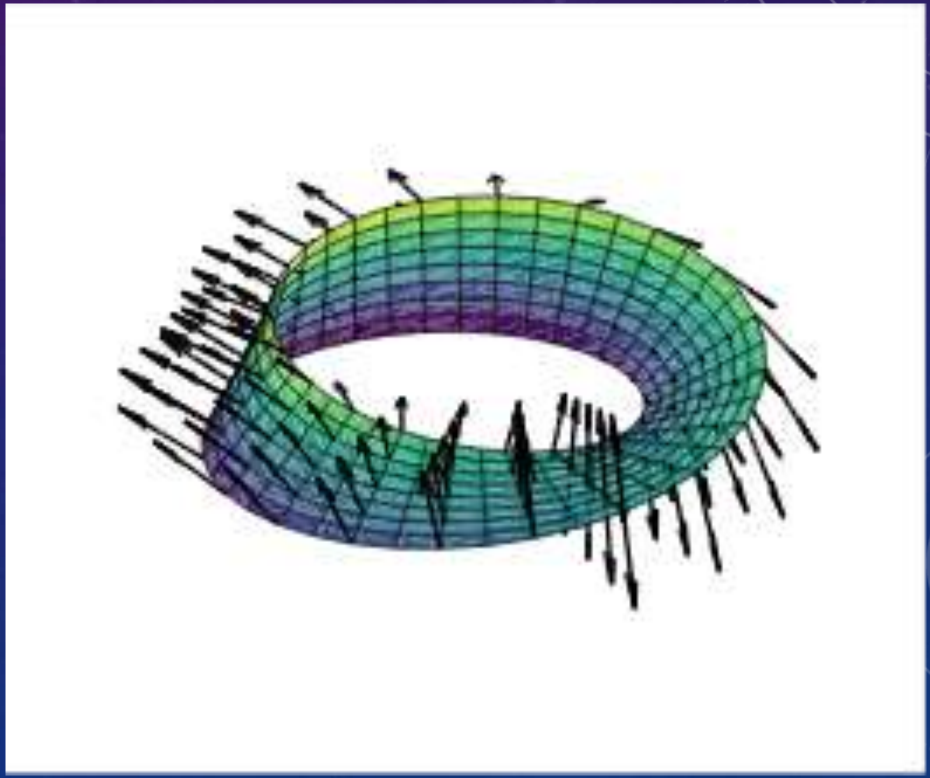




Vector field of normals

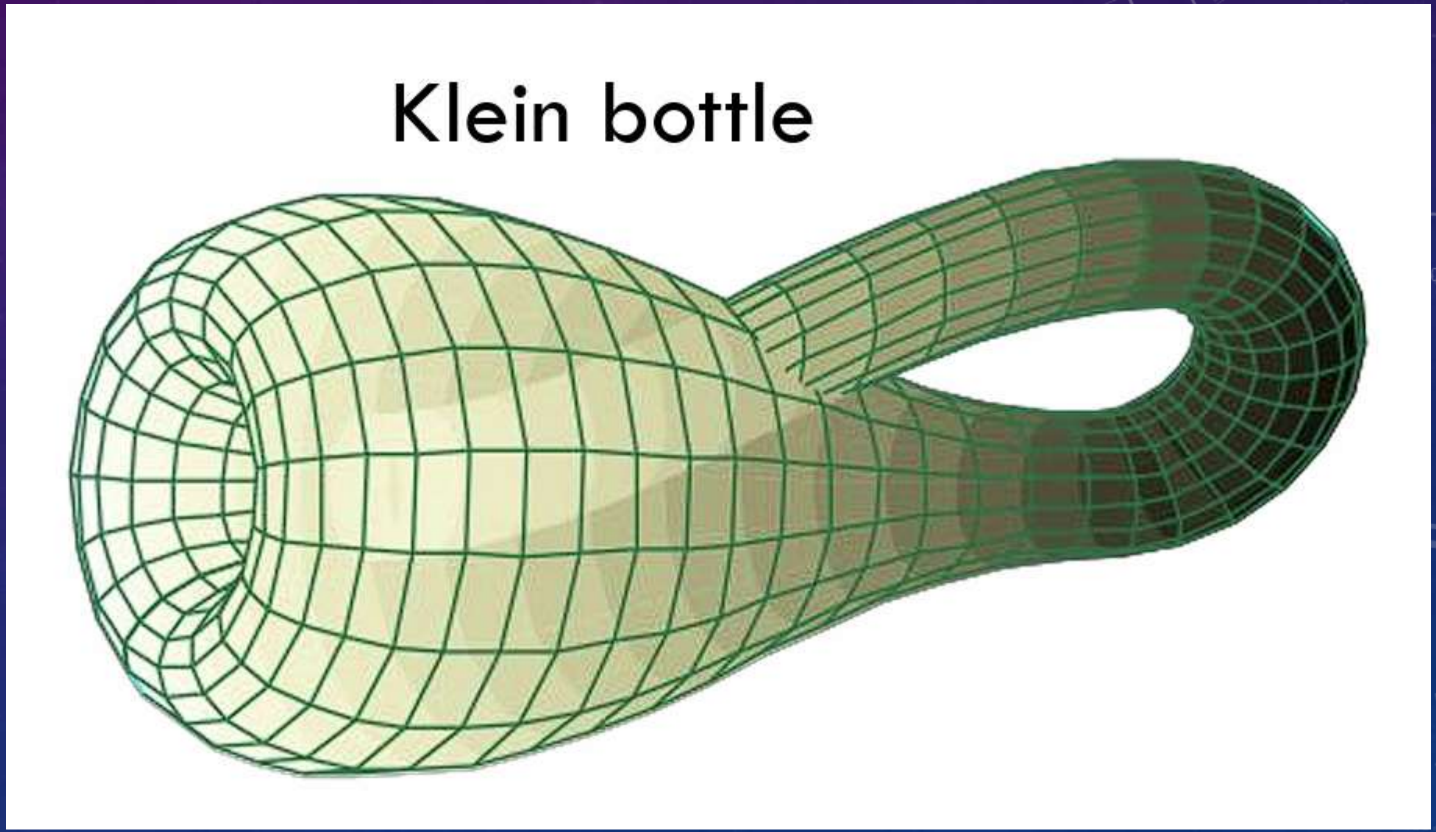
# Non-orientable manifolds





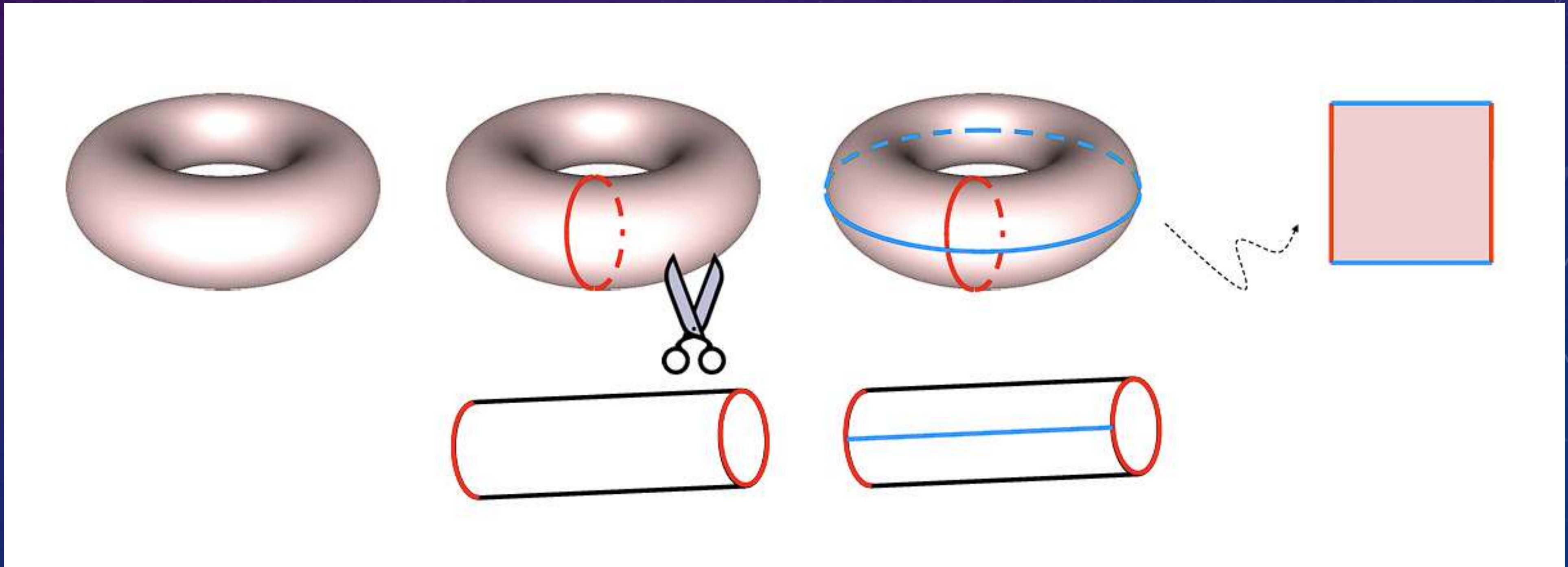
Möbius strip

# Non-orientable manifolds



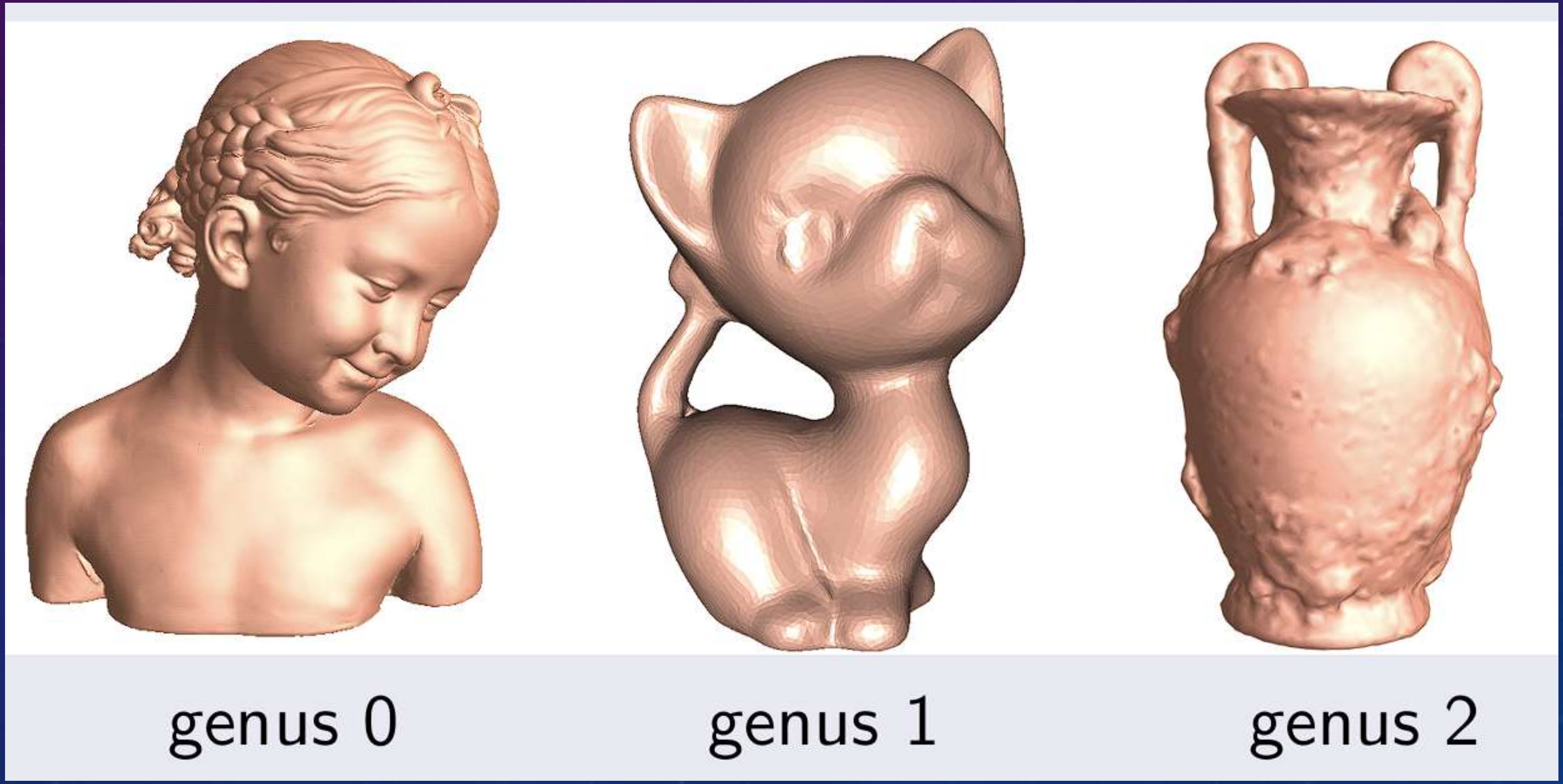
# Genus of orientable manifolds

Genus: ½ × the maximal number of closed simple curves that do not disconnect the manifold.



# Genus of orientable manifolds

Informally, the number of “donut holes”.



# Genus of orientable manifolds

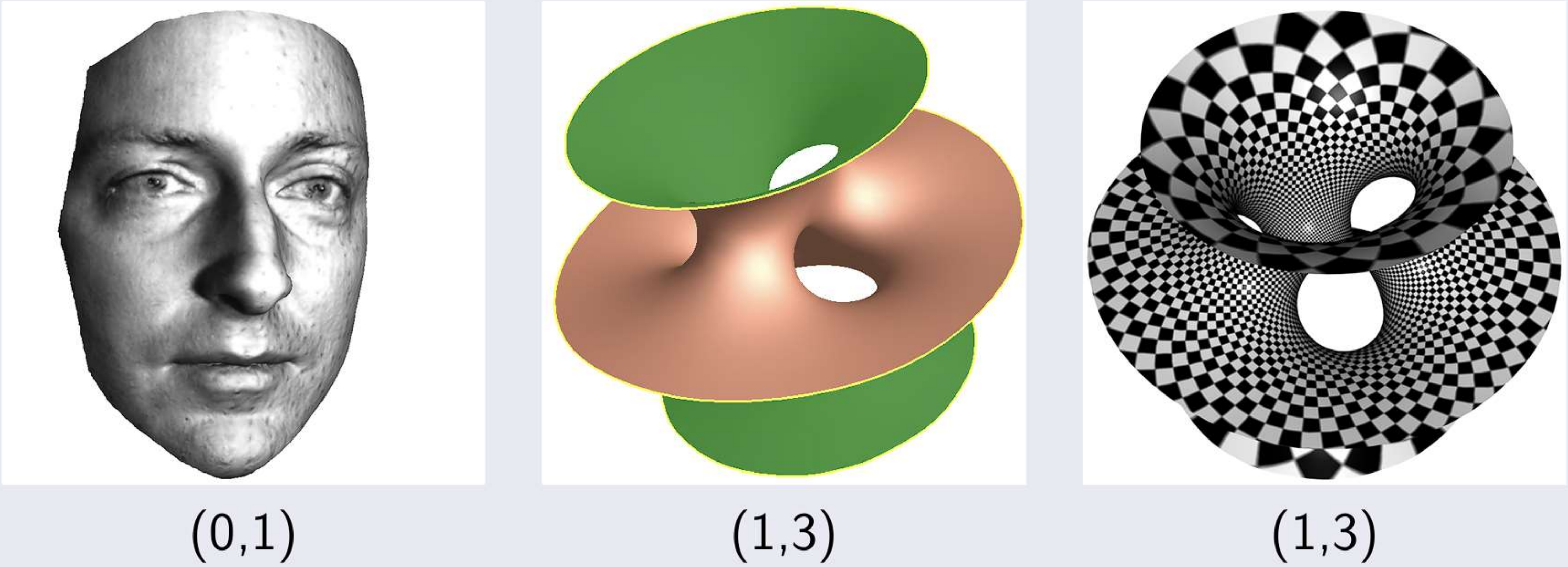
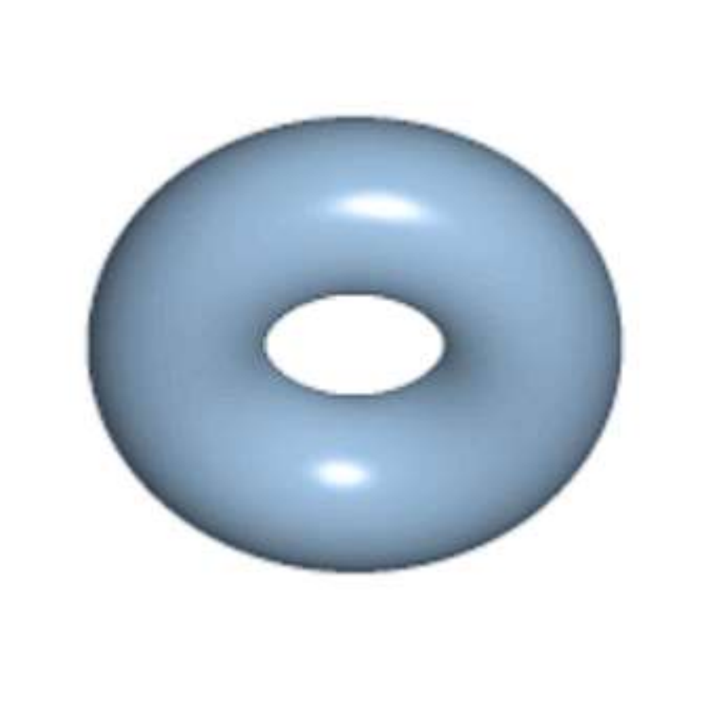



Figure: Topological classification for surfaces with boundaries $(g, b)$.

# Genus of orientable manifolds

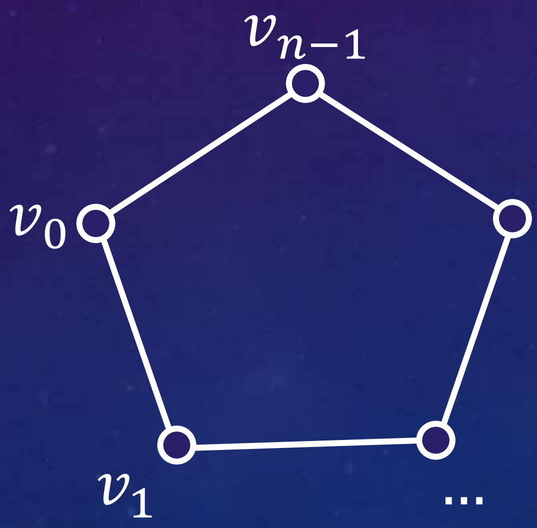


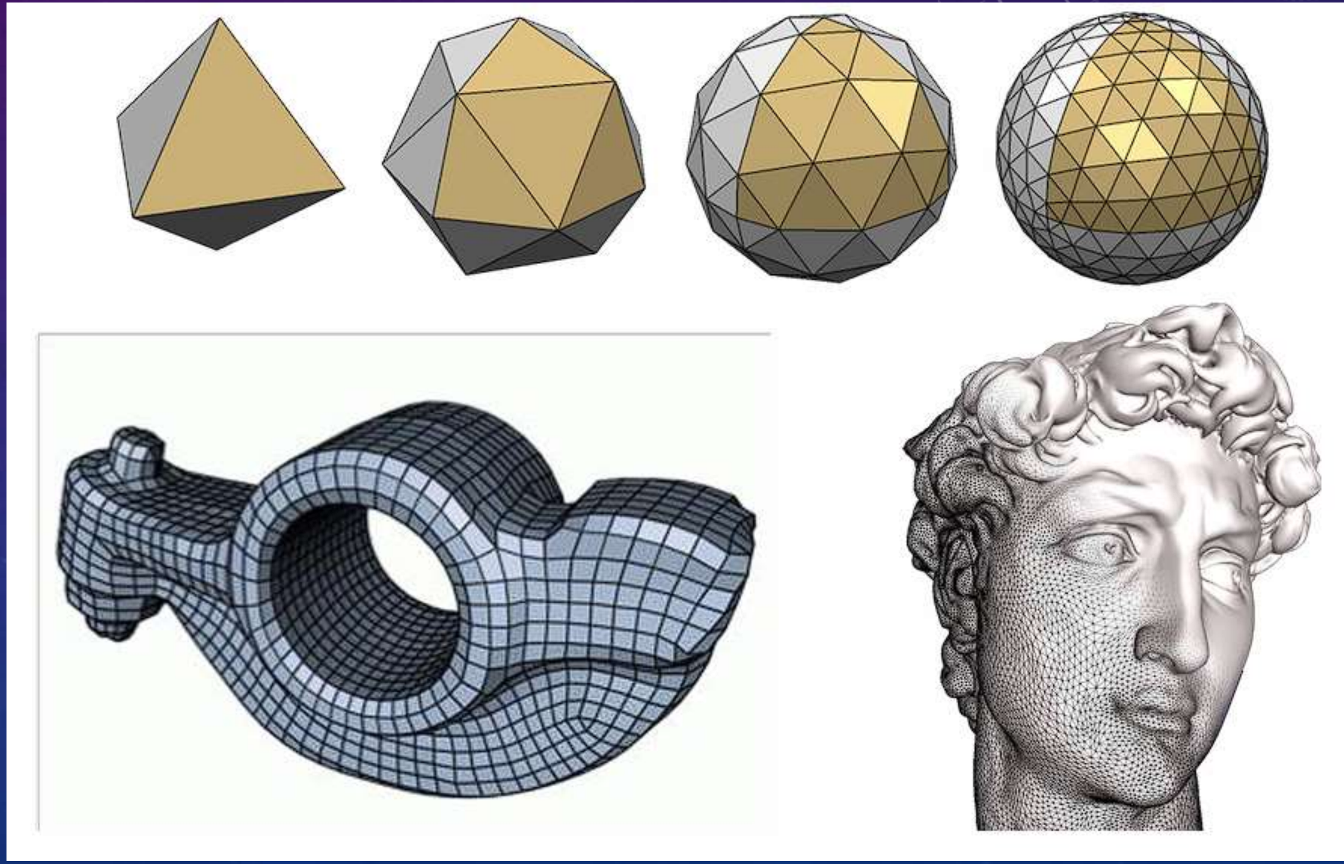
Genus 1

# Mesh

# Polygonol mesh

- Vertices: $v_0, v_1 \ldots, v_{n-1}$
- Edges: $\{(v_0, v_1), \ldots, (v_{n-1}, v_0)\}$
- Face: Planar



# Manifold polygonal mesh

A finite set $M = (V, E, F)$ is a polygonal mesh:

- The intersection of two polygons in $M$ is either empty, a vertex, or an edge.

# Manifold polygonal mesh

A finite set $M = (V, E, F)$ is a polygonal mesh:

- The intersection of two polygons in $M$ is either empty, a vertex, or an edge.
- Every edge $E_i$ belongs to at least one polygon.

# Manifold polygonal mesh

A finite set $M = (V, E, F)$ is a polygonal mesh:

- The intersection of two polygons in $M$ is either empty, a vertex, or an edge.
- Every edge $E_i$ belongs to at least one polygon.
- Each face $F_i$ defines a face of the polygonal mesh.

# Manifold polygonal mesh

A finite set $M = (V, E, F)$ is a polygonal mesh:

- The intersection of two polygons in $M$ is either empty, a vertex, or an edge.
- Every edge $E_i$ belongs to at least one polygon.
- Each face $F_i$ defines a face of the polygonal mesh.

# Manifold polygonal mesh

- Vertex degree or valence: #incident edges

# Manifold polygonal mesh

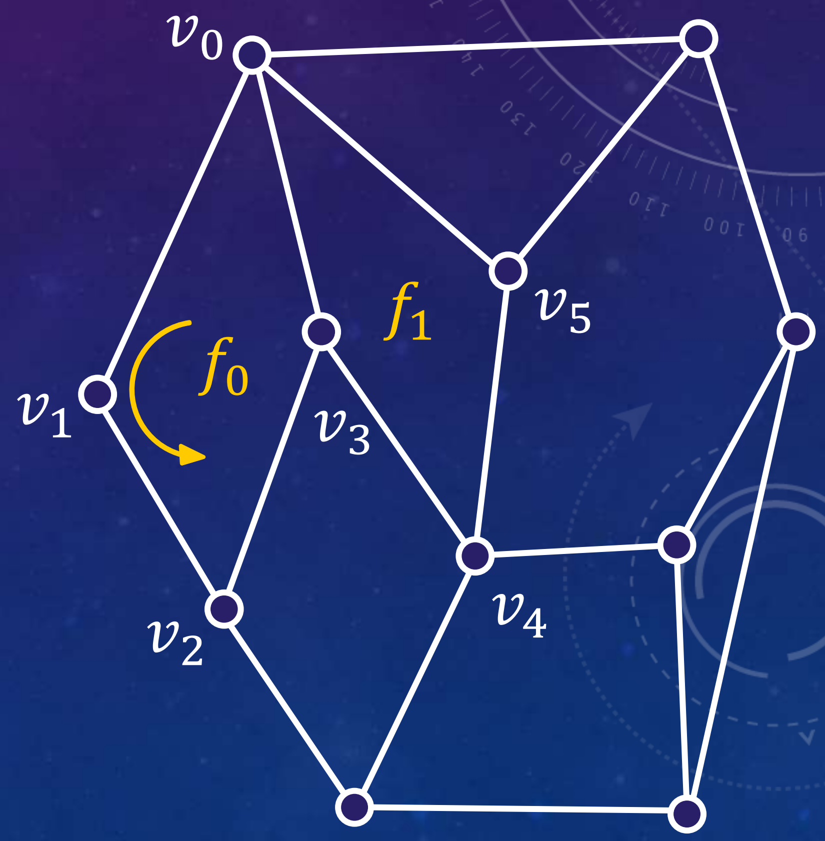
- Vertex degree or valence: #incident edges

# Manifold polygonal mesh

- Vertex degree or valence: #incident edges
- Boundary: the set of all edges that belong to only one polygon.
    - Closed loops
    - Empty

# Orientability (anticlockwise)

- Face $f_0 = \{v_0, v_1, v_2, v_3\}$
- Face $f_1 = \{v_0, v_3, v_4, v_5\}$
- Edge $f_0 \supset (v_3, v_0) \leftrightarrow (v_0, v_3) \subset f_1$



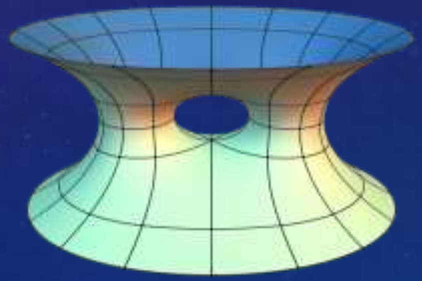
# Euler-Poincaré Formula

- For orientable manifold meshes:

$$n_V - n_E + n_F = 2(c - g) + b = \chi(M)$$

  - $c$: # connected components
  - $g$: genus
  - $b$: # boundary loops

$\chi = 2(1 - 0) + 0$



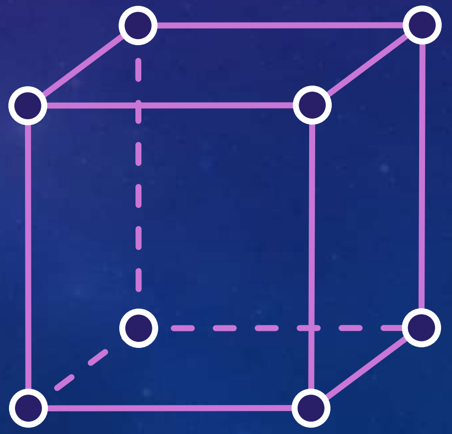
$\chi = 2(1 - 1) + 2$

# Euler-Poincaré Formula

- For orientable manifold meshes:

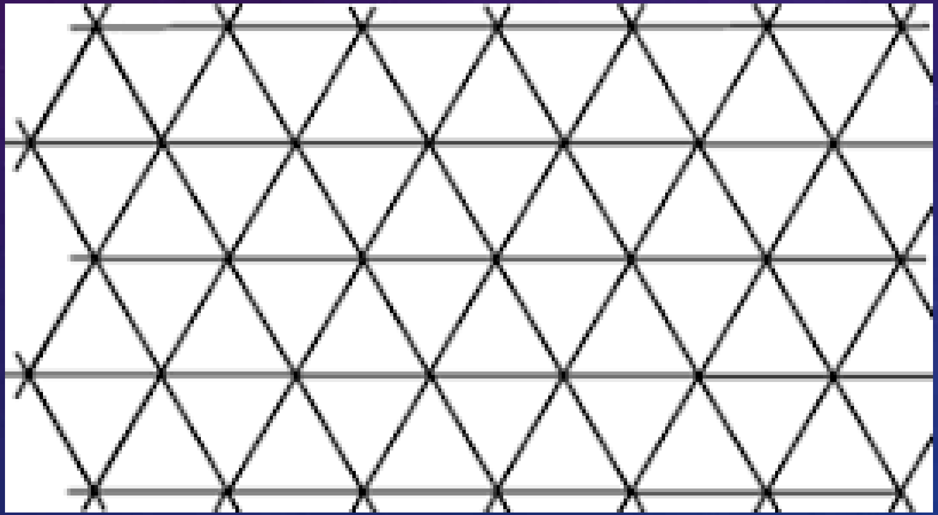
$$n_V - n_E + n_F \approx 0$$

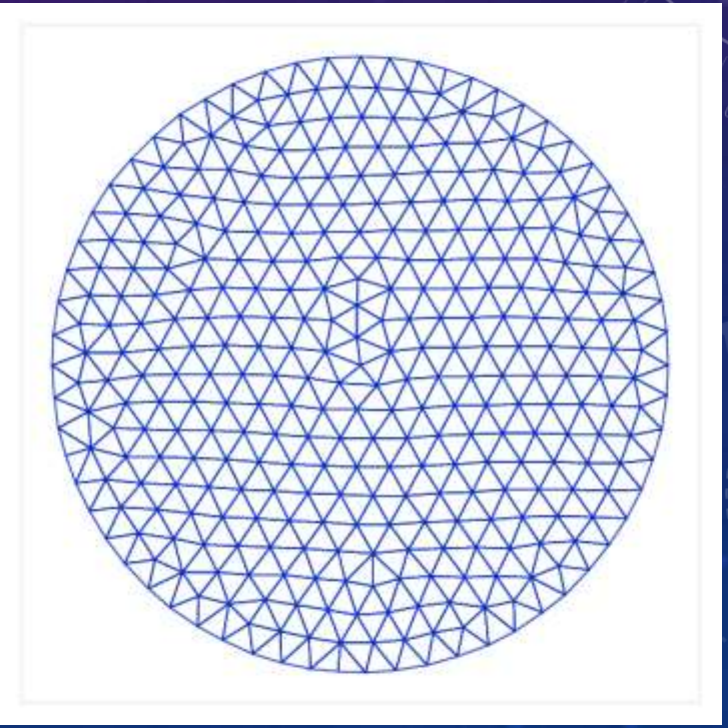
- Specially for triangular meshes:
  - $n_F \approx 2 \times n_V$
  - $n_E \approx 3 \times n_V$
  - Average vertex valence is 6



# Regularity

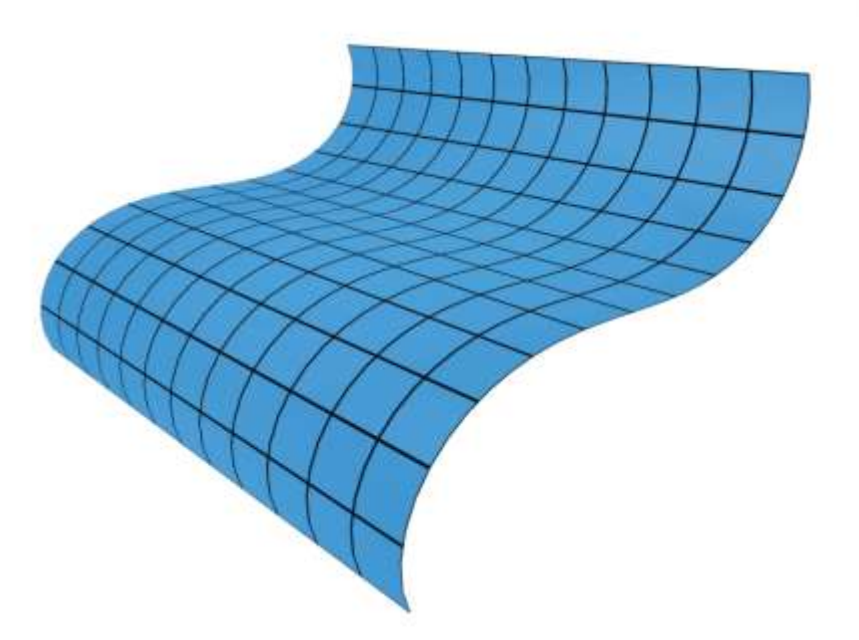
- Regular VS quasi regular

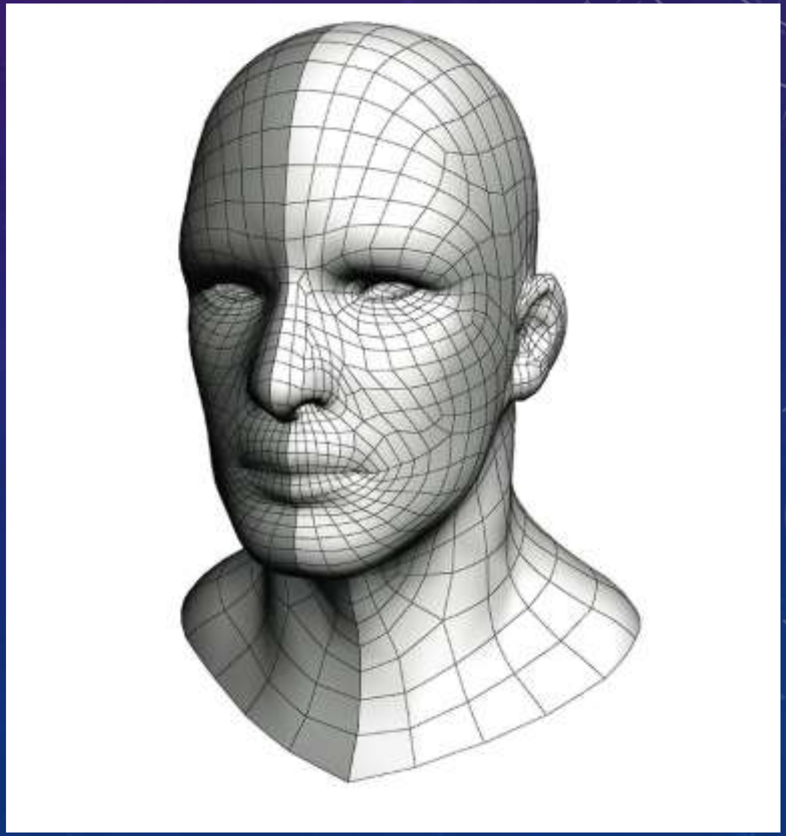




# Regularity

- Regular VS quasi regular

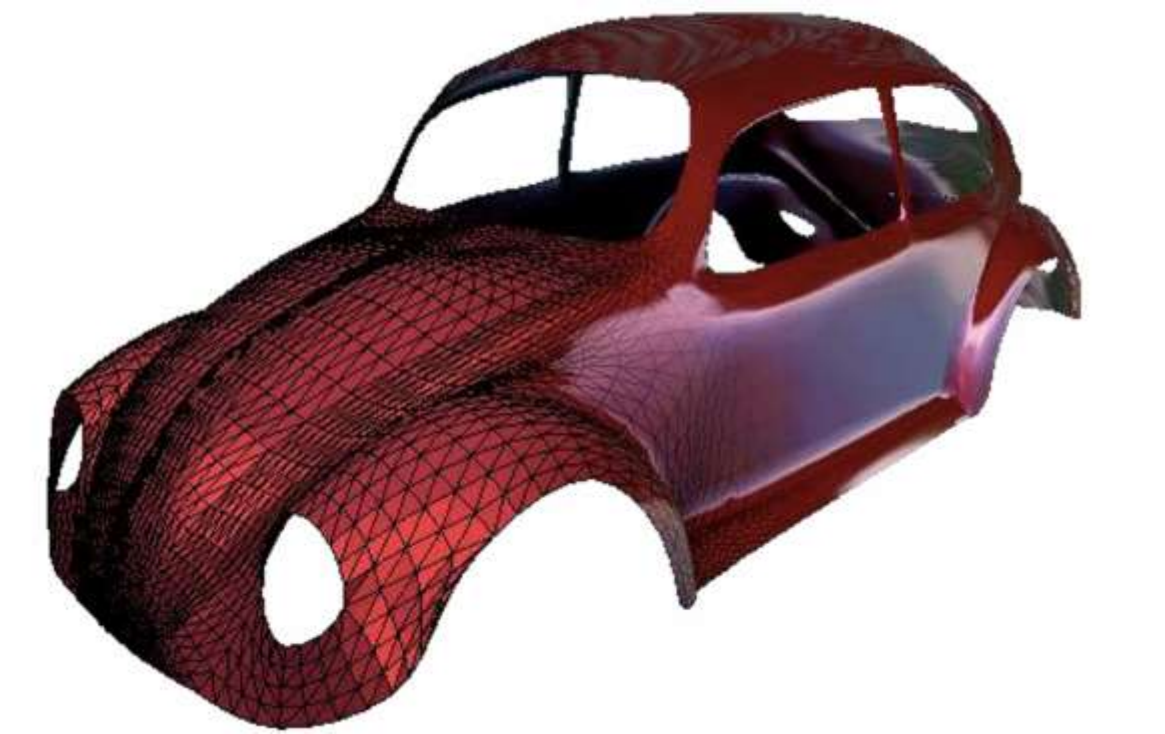




# Data structures for mesh

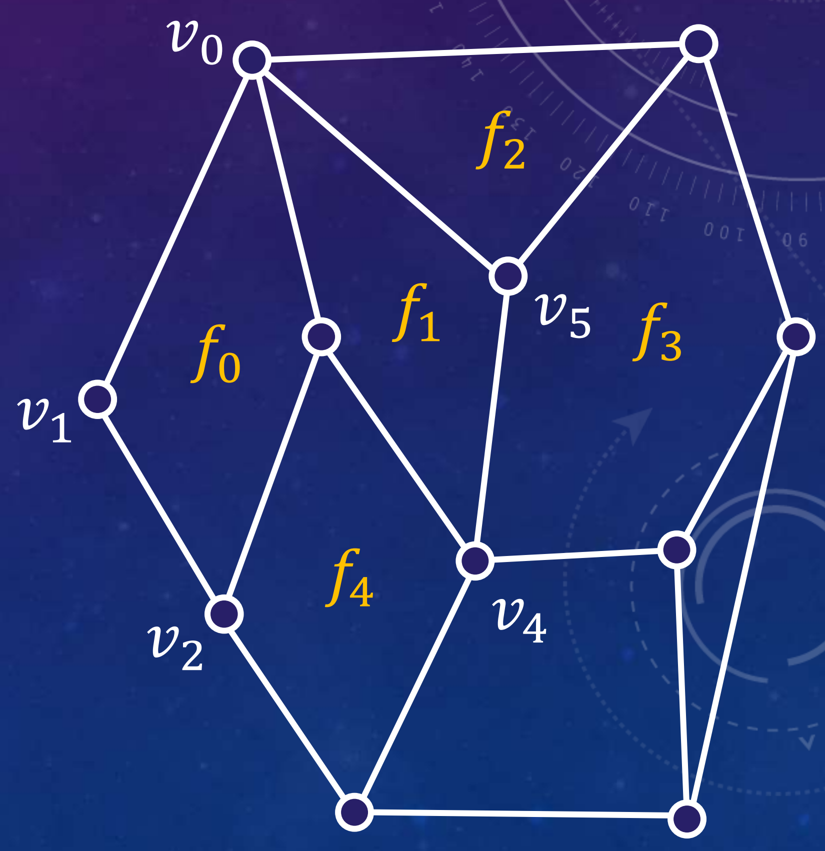
# What should be stored?

- Geometry: 3D coordinates
- Attributes
  - Normal, color, texture coordinates
  - Per vertex, face, edge Vertices
- Connectivity: adjacency relationships



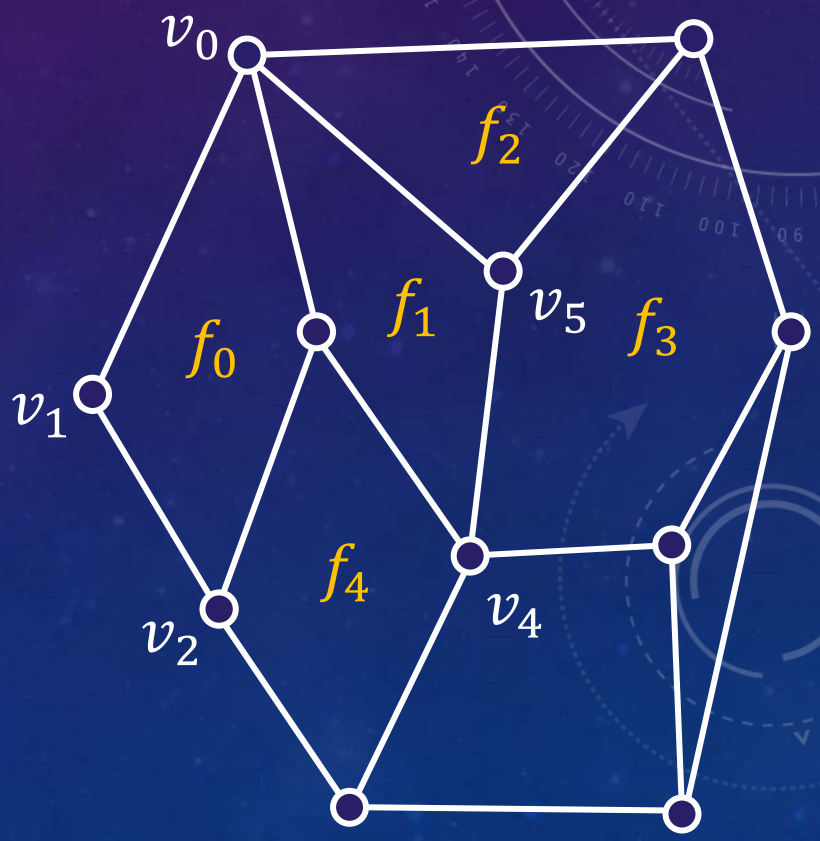
# What should be supported?

- Geometry queries
  - What are the vertices of face $f_0$?
  - Is vertex $v_0$ adjacent to vertex $v_1$?
  - Which faces are adjacent to face $f_1$?
- Modifications
  - Remove/add a vertex/face
  - Vertex split, edge collapse



# Neighborhood Relations

- All possible neighborhood relationships:
  - Vertex – Vertex VV
  - Vertex – Edge VE & EV
  - Vertex – Face VF & FV
  - Edge – Edge EE
  - Edge – Face EF & FE
  - Face – Face FF



# File format(obj)

# List of geometric vertices, with (x,y,z) coordinates

v 0.123 0.234 0.345

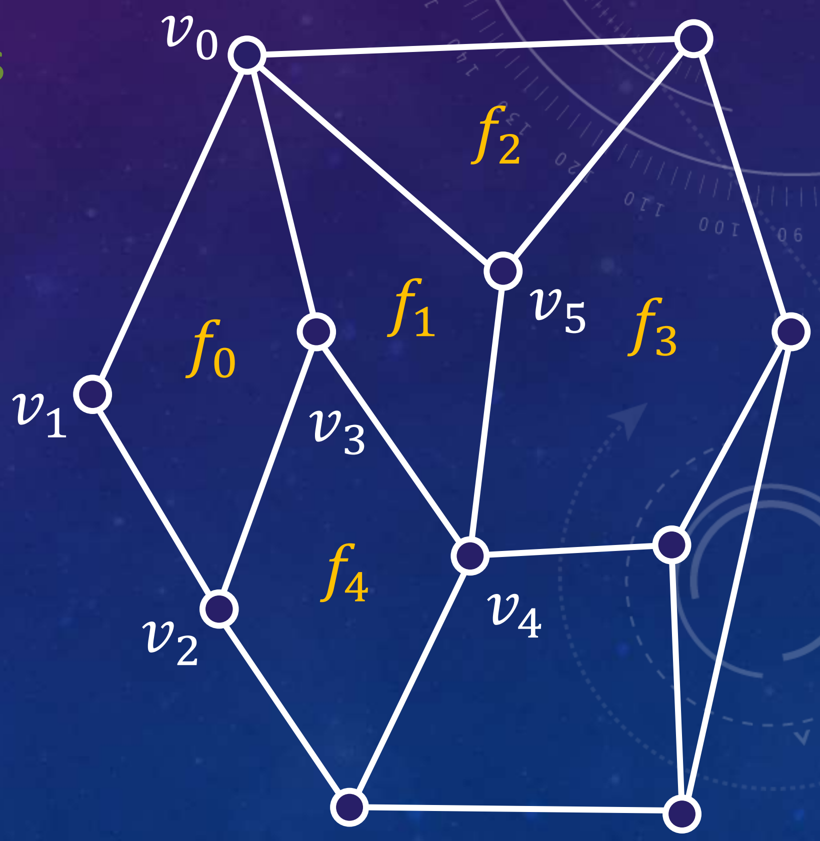
…

# Polygon face element (see below)

f 1 2 3 4

f 1 4 5 6

…



# File format(off)

OFF # Line 1

vertex_count face_count edge_count # Line 2

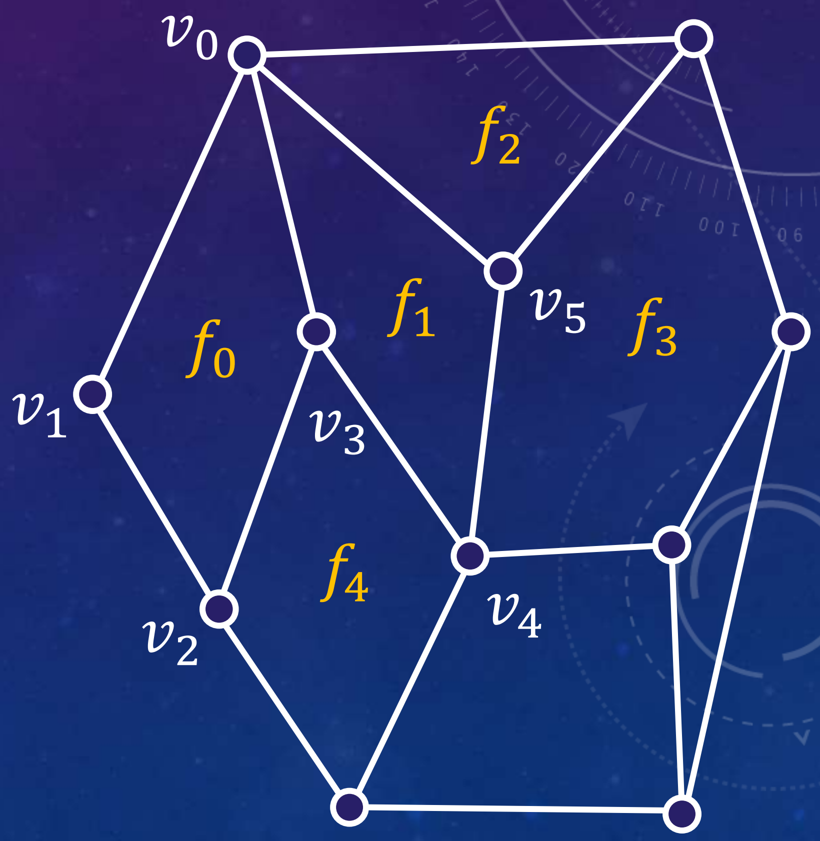
x y z # One line for each vertex

…

4 0 1 2 3 # One line for each polygon face

4 0 3 4 5 # $n\ v_0\ v_1\ \ldots v_{n-1}$, vertex id from 0

…



# Data structure – indexed face set

- Storage
  - Vertex: position
  - Face: vertex indices
  - 12 bytes per vertex (single precision)
  - $n \times 4$ bytes per face ($n$-polygon)
- Only vertices info of faces (FV)

| **Vertices** | | | |
|---|---|---|---|
| v0 | x0 | y0 | z0 |
| v1 | x1 | y1 | z1 |
| v2 | x2 | y2 | z2 |
| v3 | x3 | y3 | z3 |
| v4 | x4 | y4 | z4 |
| v5 | x5 | y5 | z5 |
| v6 | x6 | y6 | z6 |
| … | … | … | … |

| **Polygons** | | | | | |
|---|---|---|---|---|---|
| f0 | v0 | v1 | v2 | v3 | |
| f1 | v0 | v3 | v4 | v5 | |
| f2 | v0 | v5 | v6 | | |
| … | … | … | … | … | |

# Data structure – signed incidence matrix

- Vertices $V = \{0,1,2,3,4,5,6,7\}$
- Edges $E = \{0,1,2,3,4,5,6,7,8,9,10,11\}$

$$M_1 = \begin{array}{c} \\ 0\\1\\2\\3\\4\\5\\6\\7 \end{array} \begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ -1 & 0 & & & & & & & & & 0 & 0 \\ 1 & 0 & & & . & & & & & & 0 & 0 \\ 0 & 0 & & & . & . & & & & & 0 & 0 \\ 0 & 0 & & & . & . & . & & & & 0 & 0 \\ 0 & -1 & & & & . & . & . & & & 0 & -1 \\ 0 & 1 & & & & & . & . & & & -1 & 0 \\ 0 & 0 & & & & & & . & & & 1 & 0 \\ 0 & 0 & & & & & & & & & 0 & 1 \end{array}$$

# Data structure – signed incidence matrix

- Faces $F = \{0,1,2,3,4,5\}$

  (front, back, top, bottom, left, right)

- Edges $E = \{0,1,2,3,4,5,6,7,8,9,10,11\}$

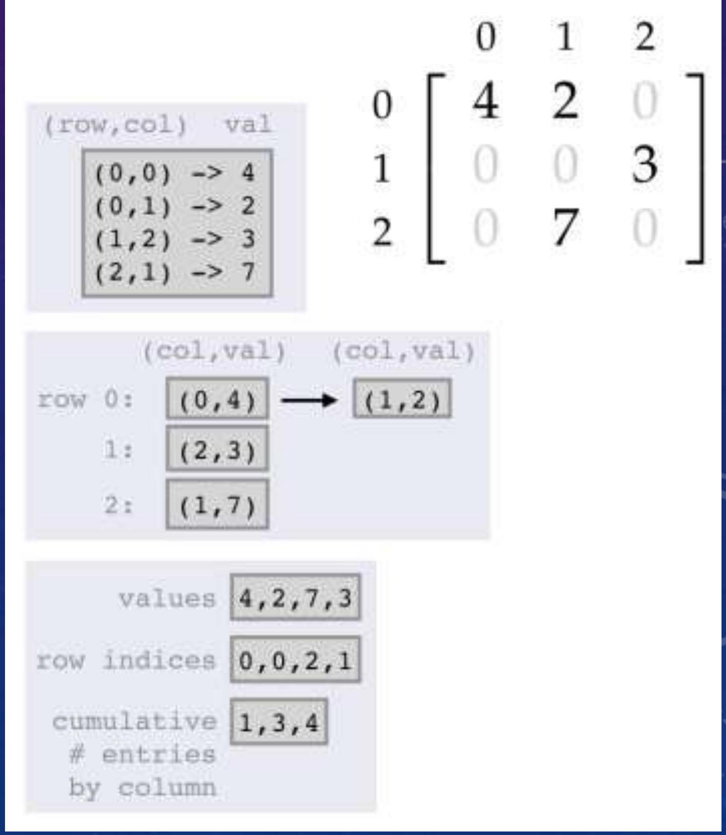
$$
M_2 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\end{array}
$$

$$
\begin{pmatrix}
1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
 & & & & \cdot & \cdot & & & & & & \\
 & & & & & \cdot & \cdot & & & & & \\
 & & & & & & \cdot & \cdot & & & & \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0
\end{pmatrix}
$$

# Data structure – signed incidence matrix

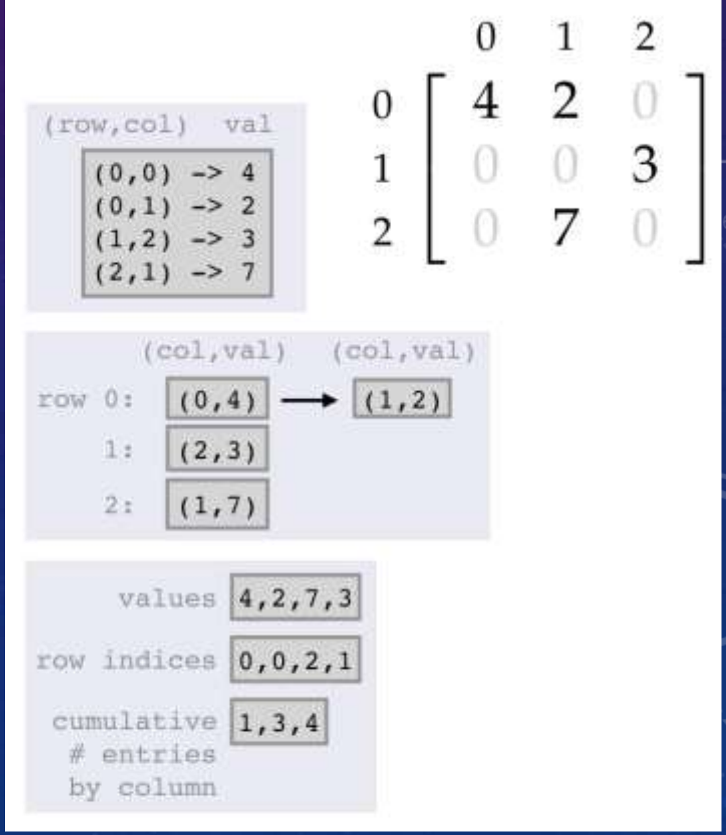
- Edge – Vertex $M_1 : n_E \times n_V$, Face – Edge $M_2 : n_F \times n_E$
- Face – Vertex $M_{21} = \mathrm{abs}(M_2) \times \mathrm{abs}(M_1) : n_F \times n_V$

$$M_{21} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 2 & 2 \\ & & . & . & & & & \\ & & & . & . & & & \\ & & & & . & . & & \\ 0 & 2 & 2 & 0 & 0 & 2 & 2 & 0 \end{pmatrix} \end{array}$$



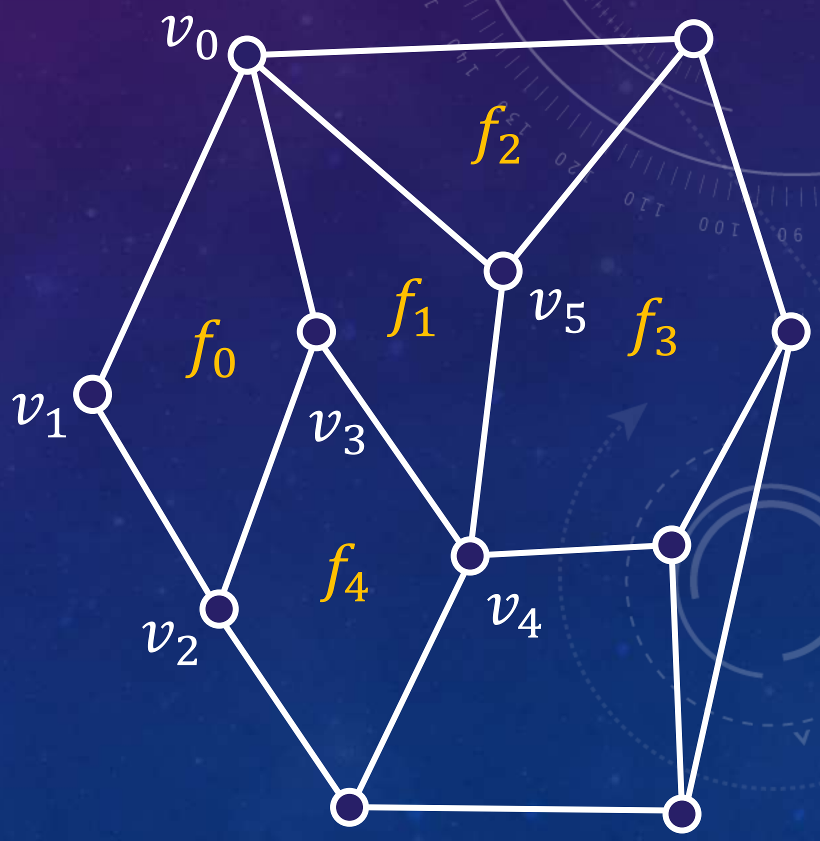
# Data structure – signed incidence matrix

- Edge – Vertex $M_1 : n_E \times n_V$, Face – Edge $M_2 : n_F \times n_E$
- Face – Face $M_{2T} = \mathrm{abs}(M_2) \times \mathrm{abs}(M_2^T) : n_F \times n_F$
- Vertex – Vertex $M_{T1} = \mathrm{abs}(M_1^T) \times \mathrm{abs}(M_1) : n_V \times n_V$
- …



# Data structure – signed incidence matrix

- Neighbor query: $M_1, M_2, M_1^T, M_2^T$
- Order info (clockwise & anticlockwise) lost!
  - 1-ring faces
  - 1-ring vertices



# Data structure – halfedge structure

- Halfedge
  - Origin vertex index
  - Incident face index
  - Next, prev, opposite halfedge indices
- Vertex : outgoing halfedge index
- Face : adjacent halfedge index

# Data structure – halfedge structure

- 1 ring vertices traversal (clockwise)
  - Start at vertex (outgoing halfedge)

# Data structure – halfedge structure

- 1 ring vertices traversal (clockwise)
  - Start at vertex (outgoing halfedge)
  - Opposite halfedge (origin vertex)

# Data structure – halfedge structure

- 1 ring vertices traversal (clockwise)
  - Start at vertex (outgoing halfedge)
  - Opposite halfedge (origin vertex)
  - Next halfedge

# Data structure – halfedge structure

- 1 ring vertices traversal (clockwise)
  - Start at vertex (outgoing halfedge)
  - Opposite halfedge (origin vertex)
  - Next halfedge
  - Opposite halfedge (origin vertex)

# Data structure – halfedge structure

- 1 ring vertices traversal (clockwise)
  - Start at vertex (outgoing halfedge)
  - Opposite halfedge (origin vertex)
  - Next halfedge
  - Opposite halfedge (origin vertex)
  - …