CS3451: Computer Graphics

Mesh

Bo Zhu

School of Interactive Computing

Georgia Institute of Technology

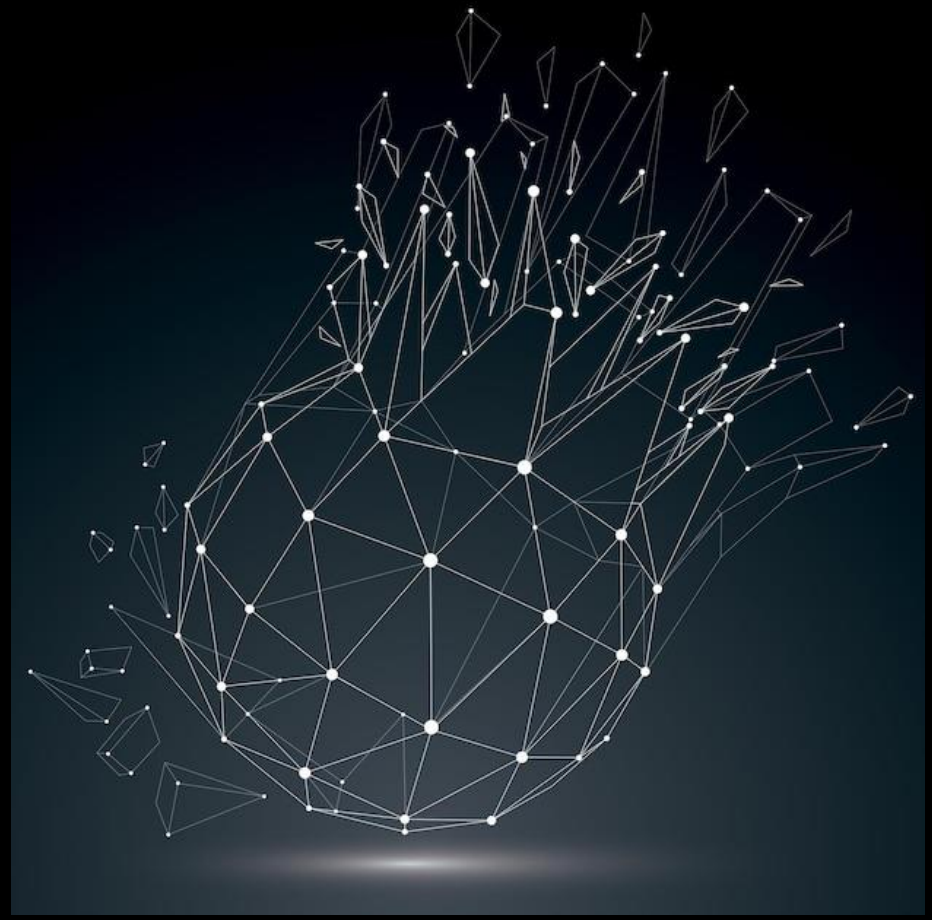
Slides courtesy of Prof. Wojciech Jarosz, Olga Sorkine-Hornung, Steve Marschner, Hao Li, Keenan Crane

Motivational Video: MeshGPT

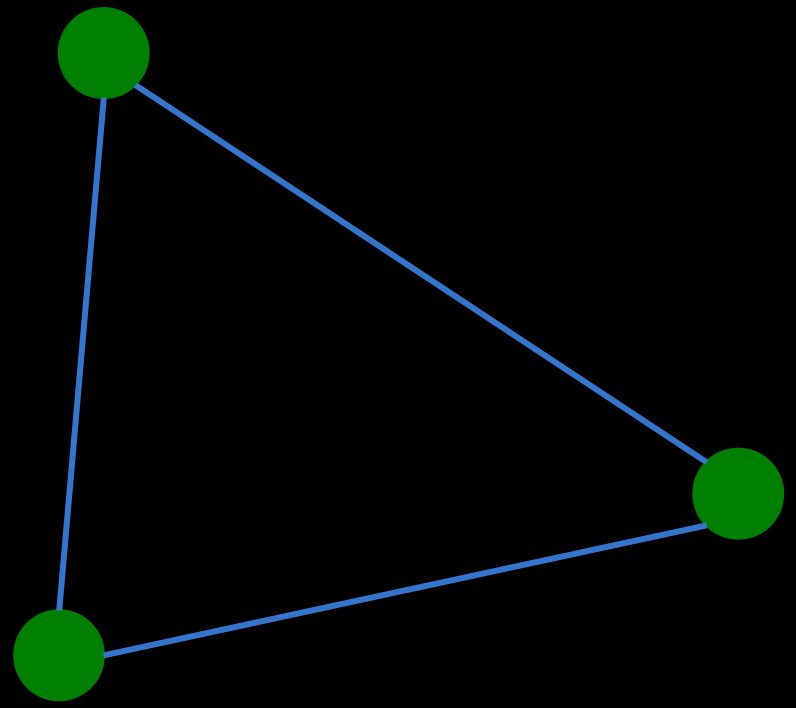
https://www.youtube.com/watch?v=UV90O1_69_o

Topics

- Triangle
- Mesh
- Normal
- Orientation
- Topology
- Euler-Poincare Formula
- Data structures

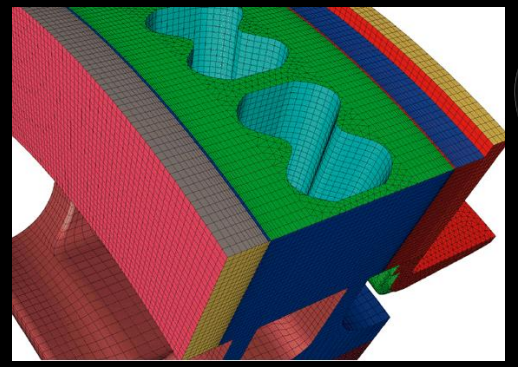
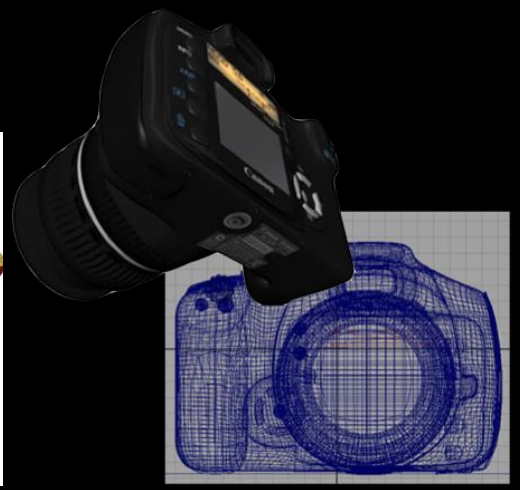


TRIANGLE

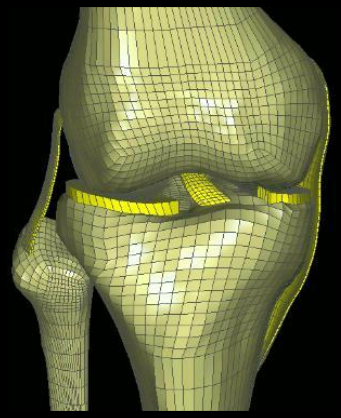


Digital Shapes

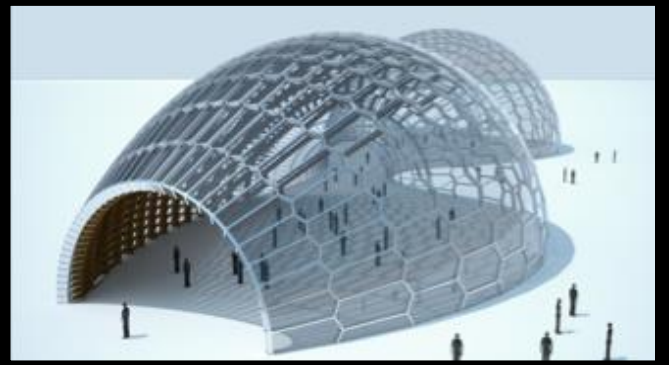
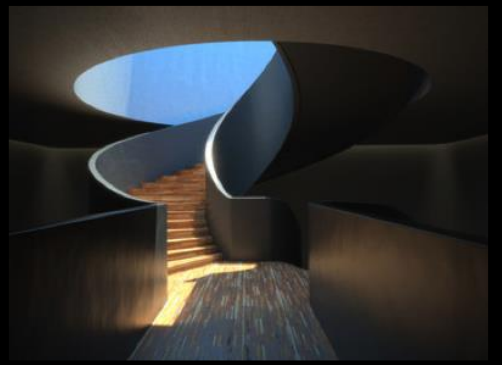

Games/Movies

Engineering/Product design

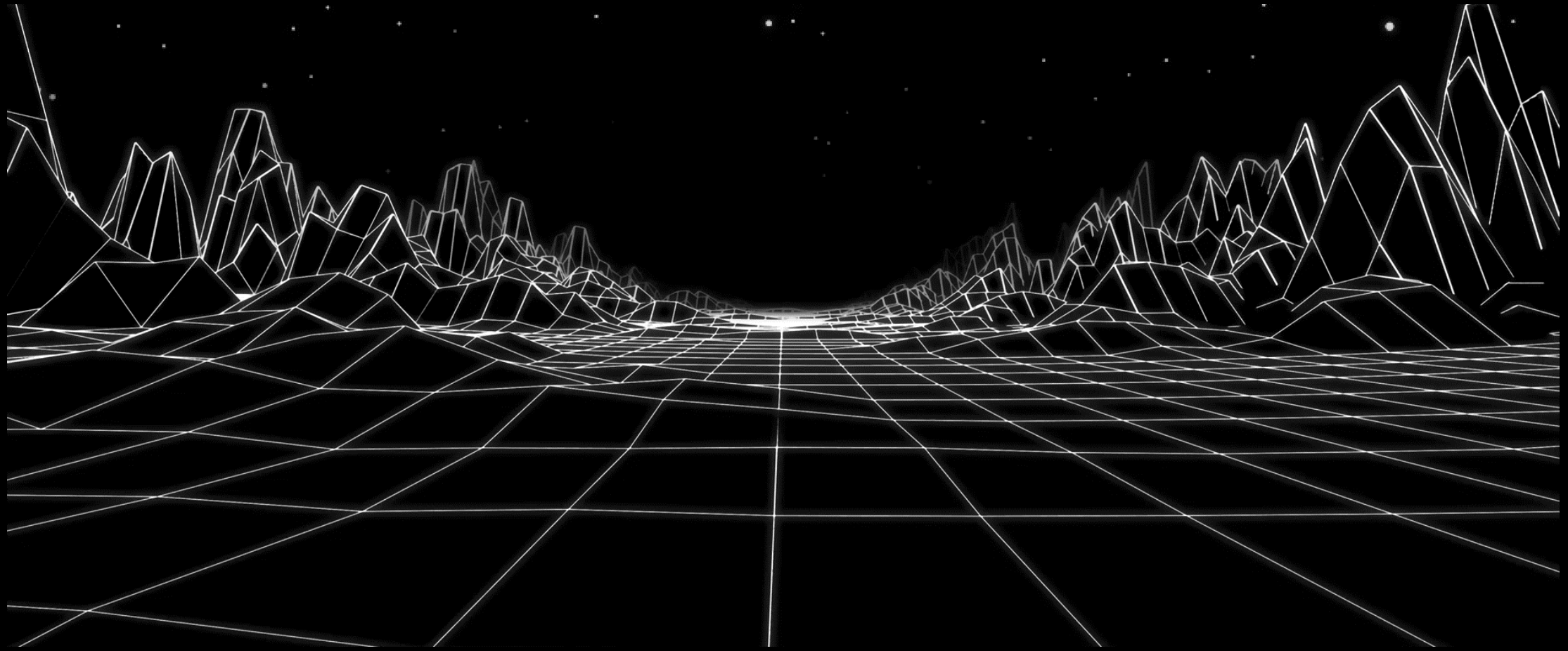



Medicine/Biology

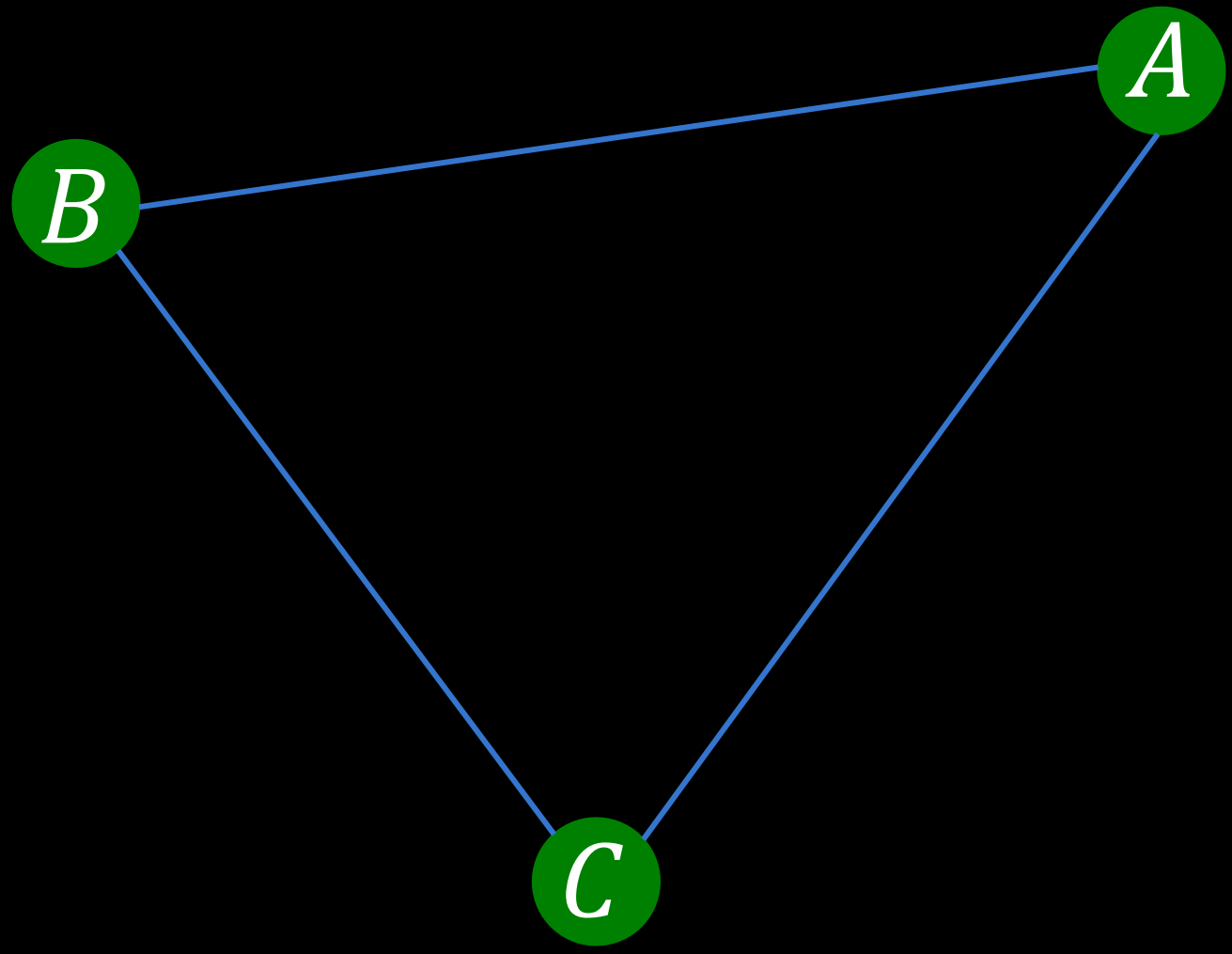
Architecture

Mesh Representation for Geometry Shapes



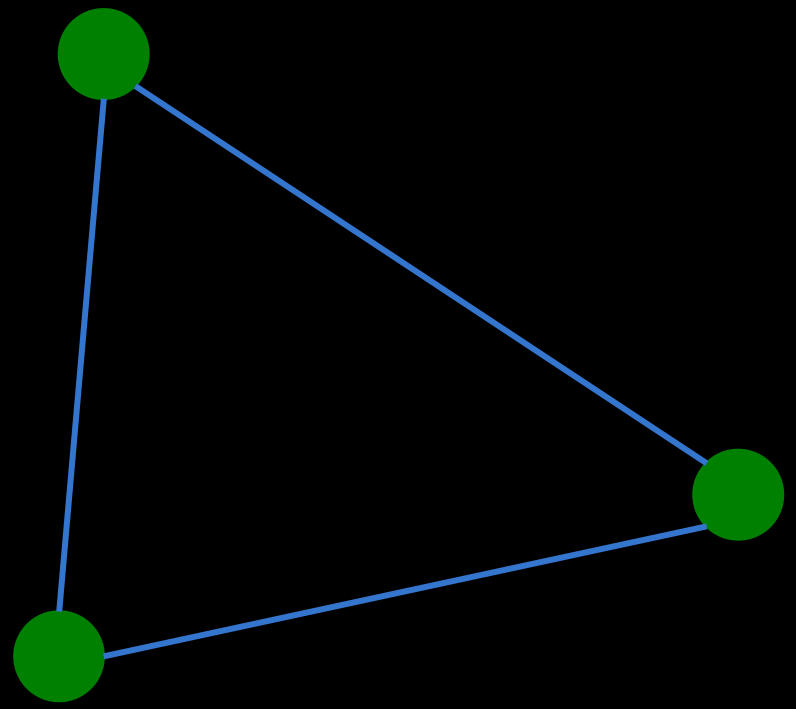
Triangle

- Three Vertices
  - A, B, C
- Three Edges
  - AB, BC, CA
- One Face
  - Triangle ABC

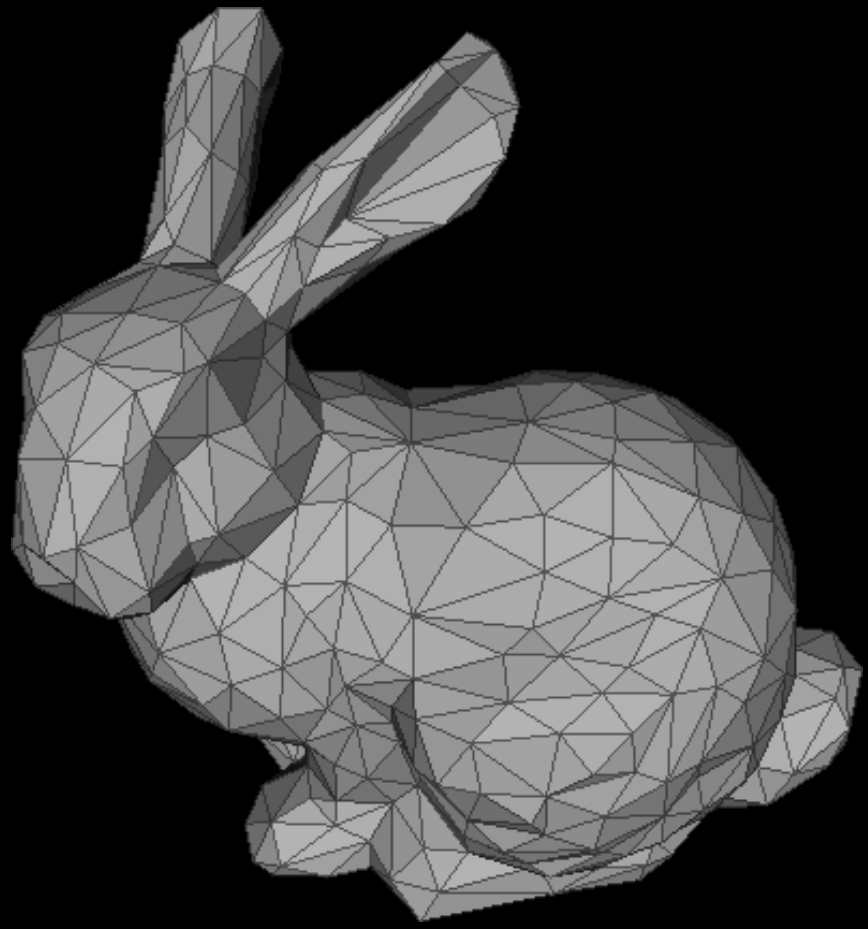


Triangle

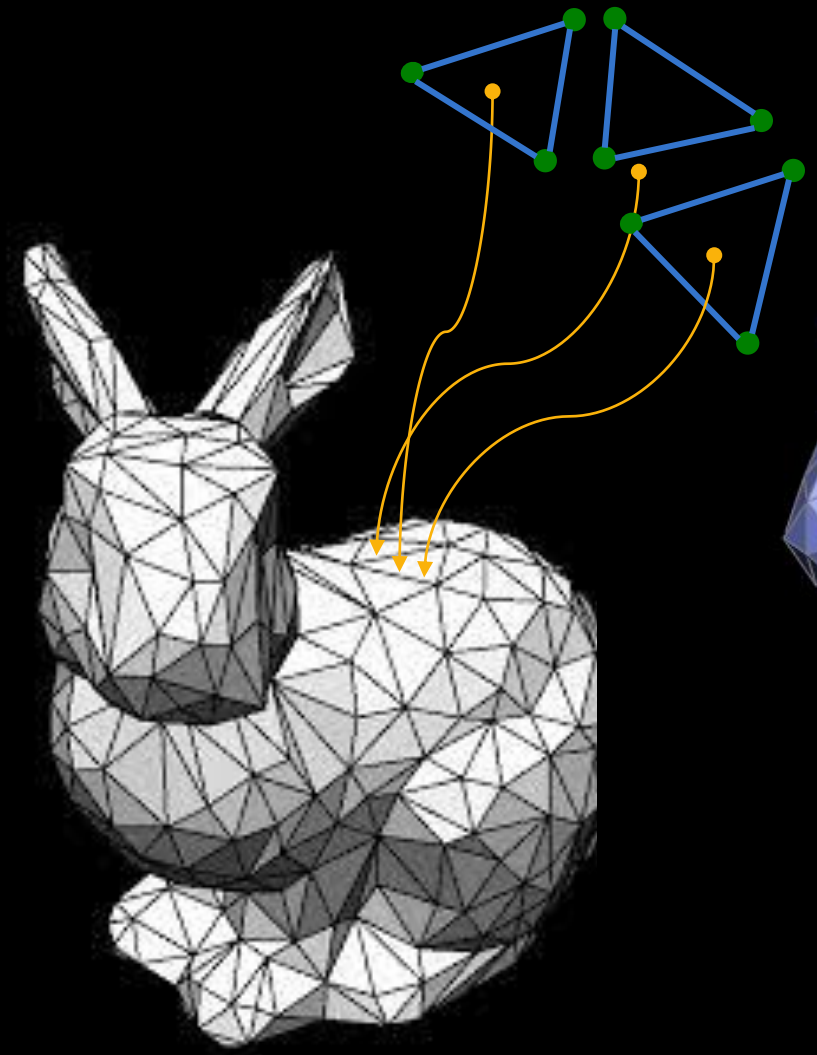
- We can store properties on triangle vertices or faces
  - For example
    - Vertex positions
    - Vertex colors
    - Texture coordinates
    - Face normals
    - Transparency
    - Etc.



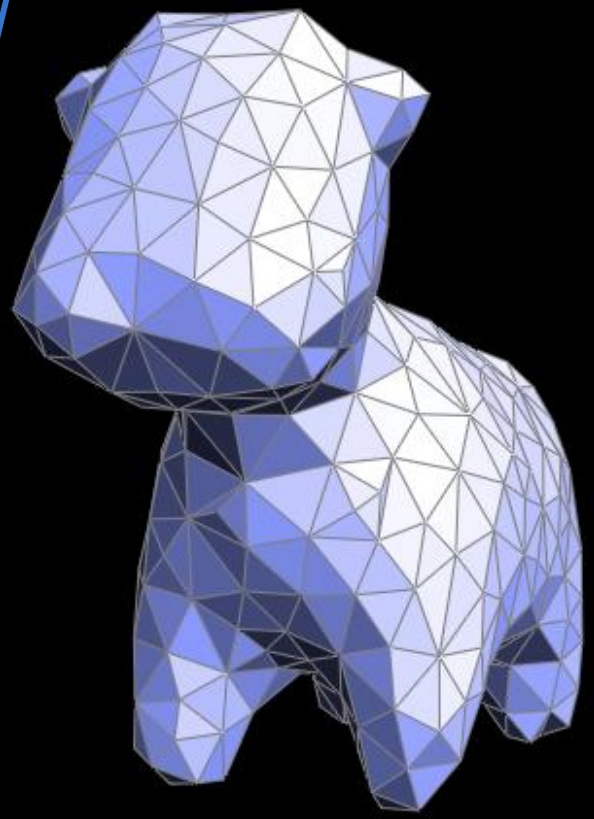
Mesh

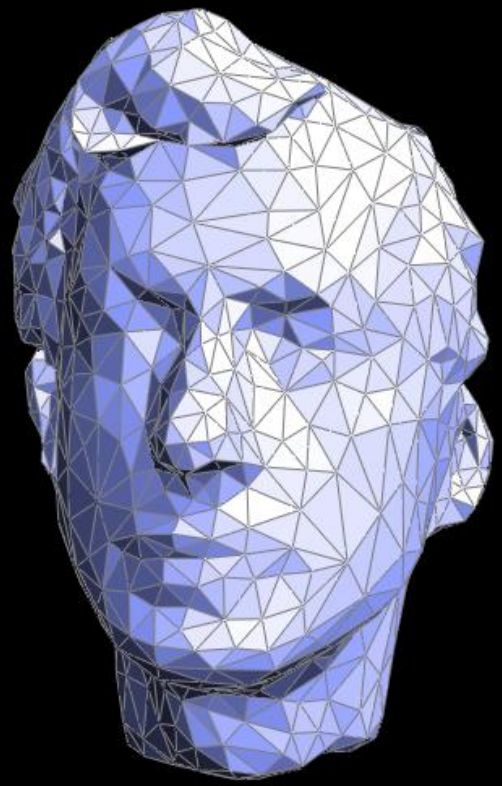


Triangle Mesh

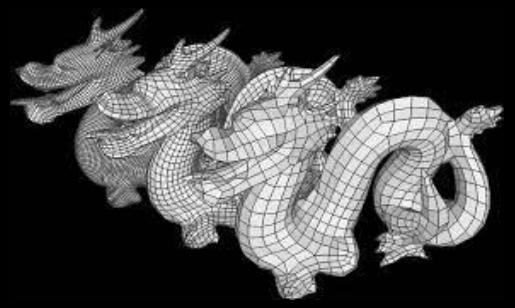


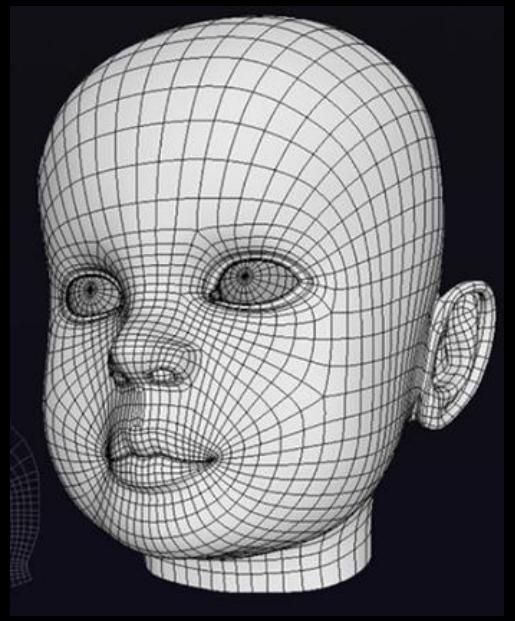
We combine a group of triangles together to get a triangle mesh

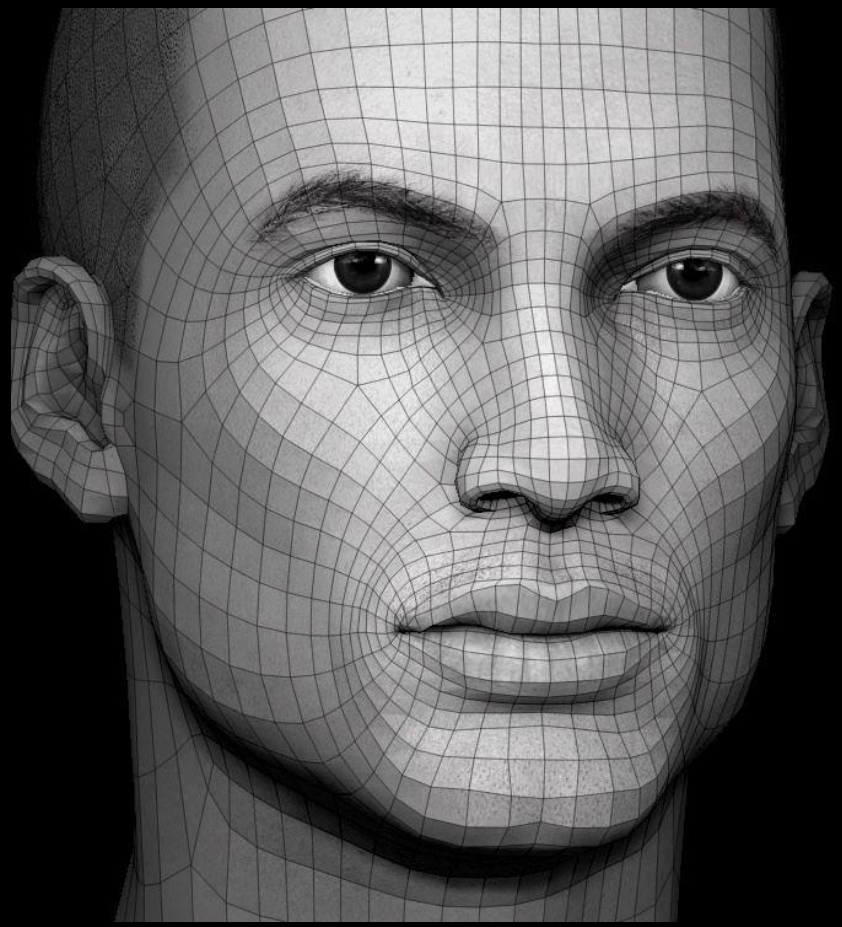


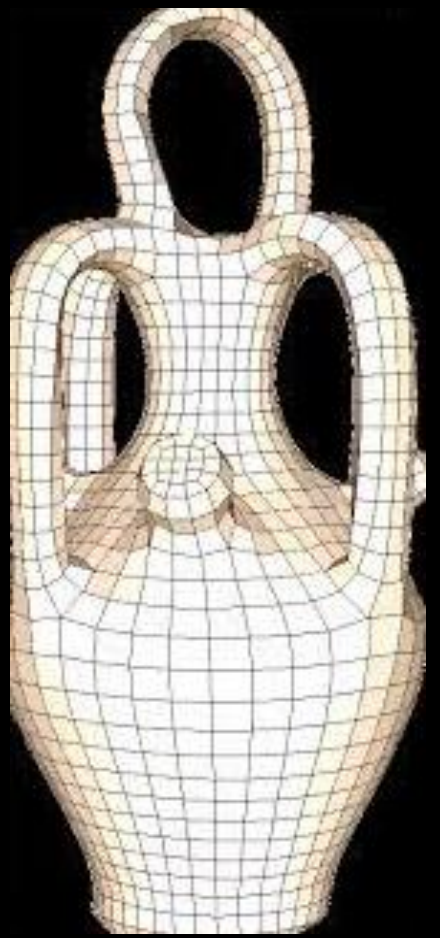


Other Mesh Types





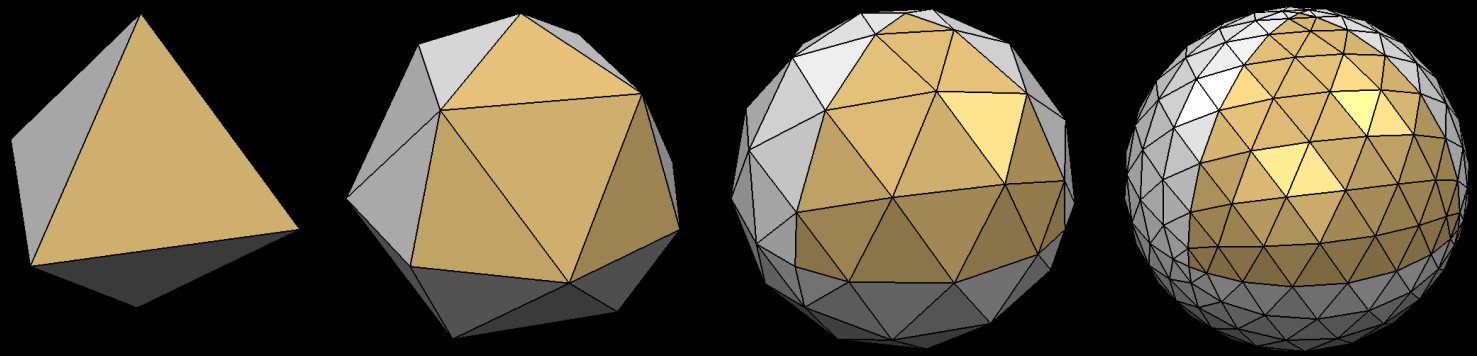




Quad mesh

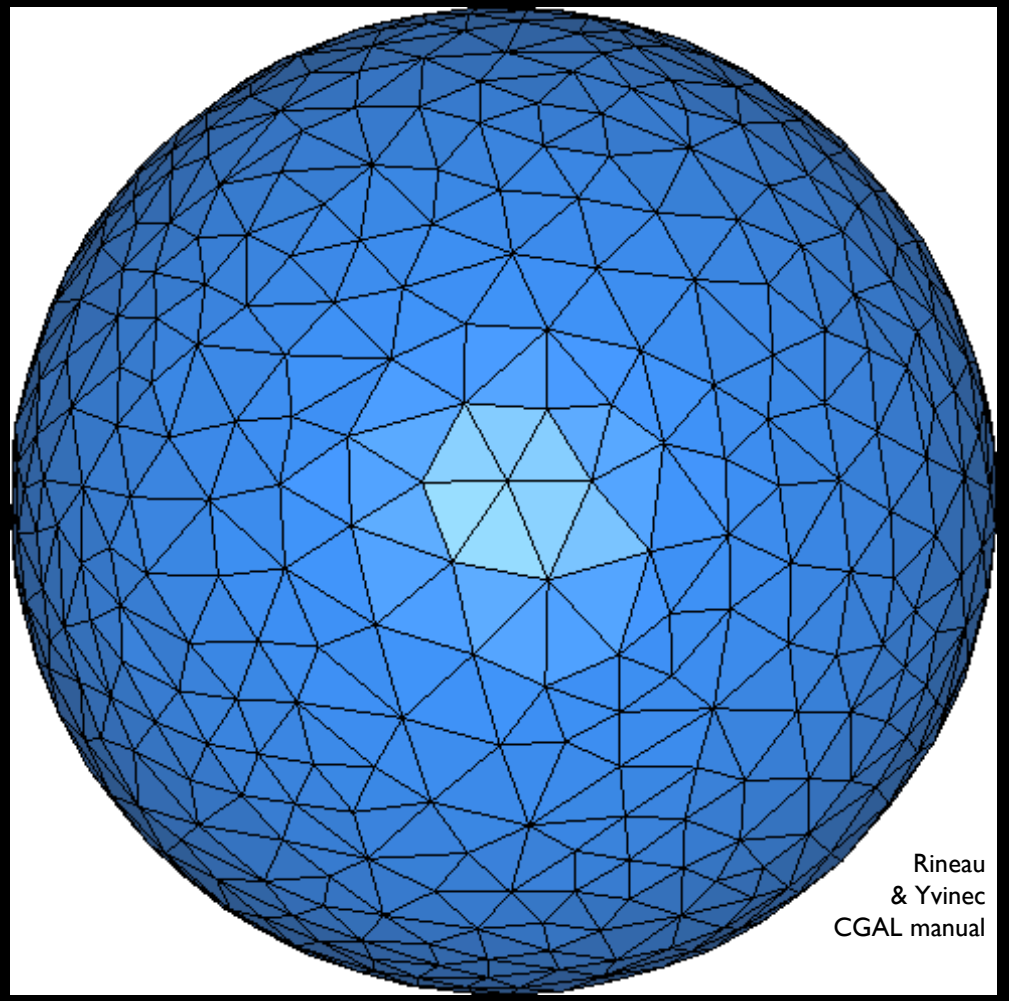
Mesh Resolution

- Boundary representations of objects
- Piecewise linear



Mesh Resolution

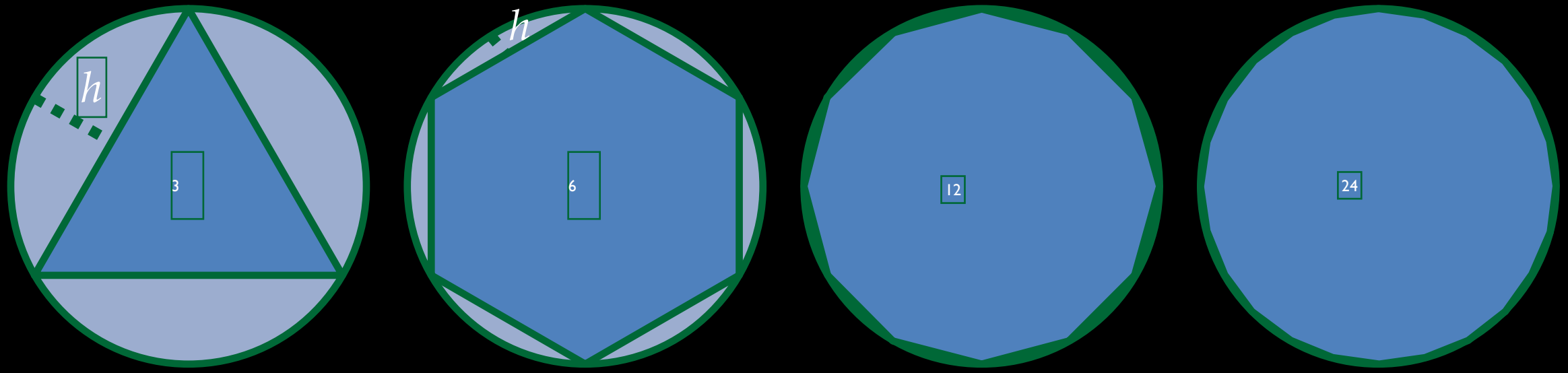
Spheres



Discrete Sphere

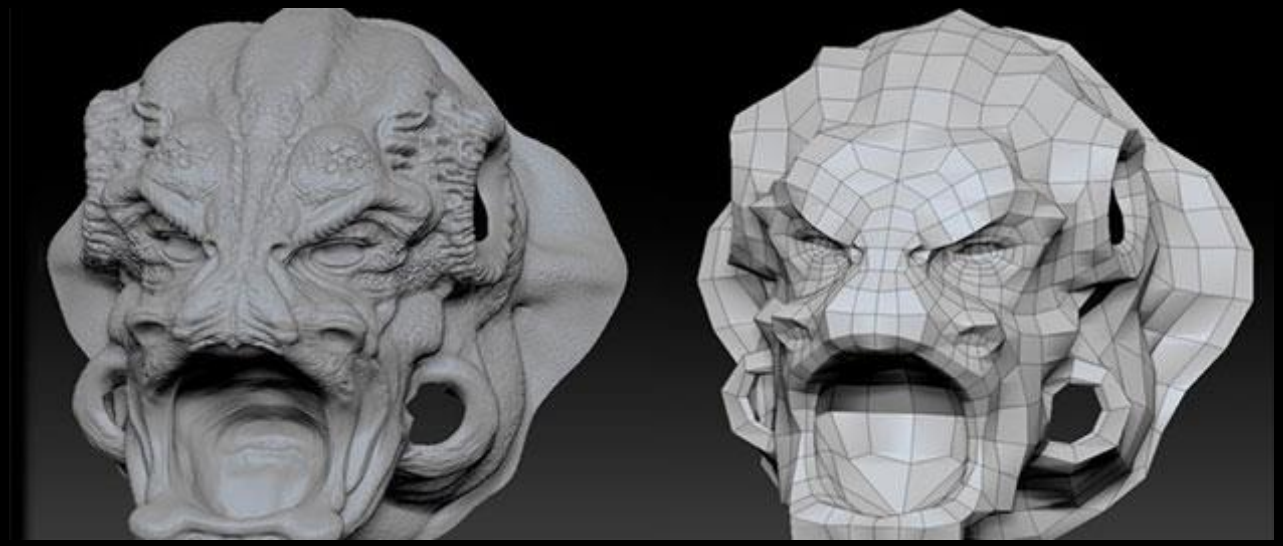
Meshes as Approx. of Smooth Surfaces

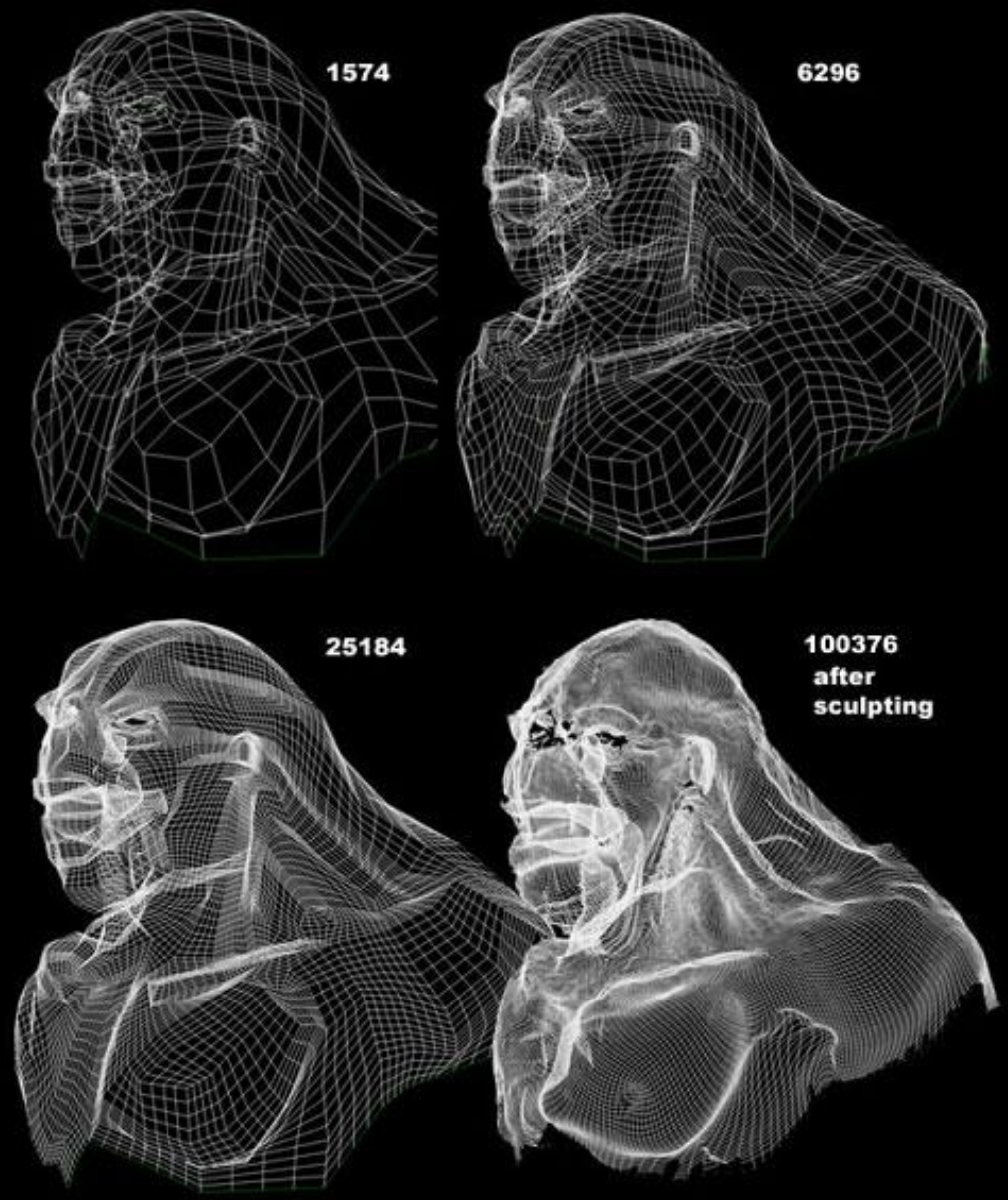
- Piecewise linear approximation
  - Error is $O(h^2)$



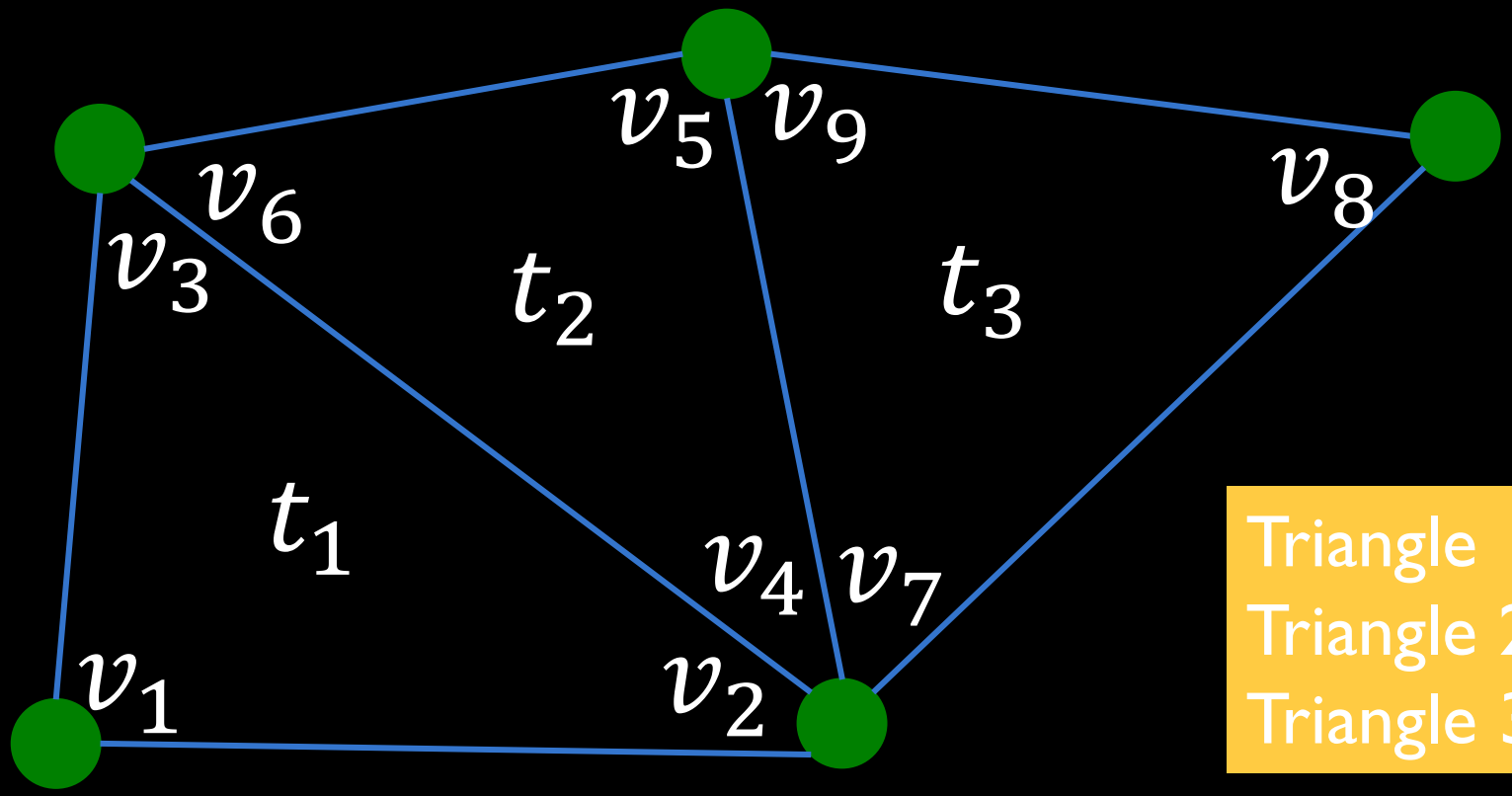
25% 6.5% 1.7% 0.4%

Low-Res v.s. High-Res



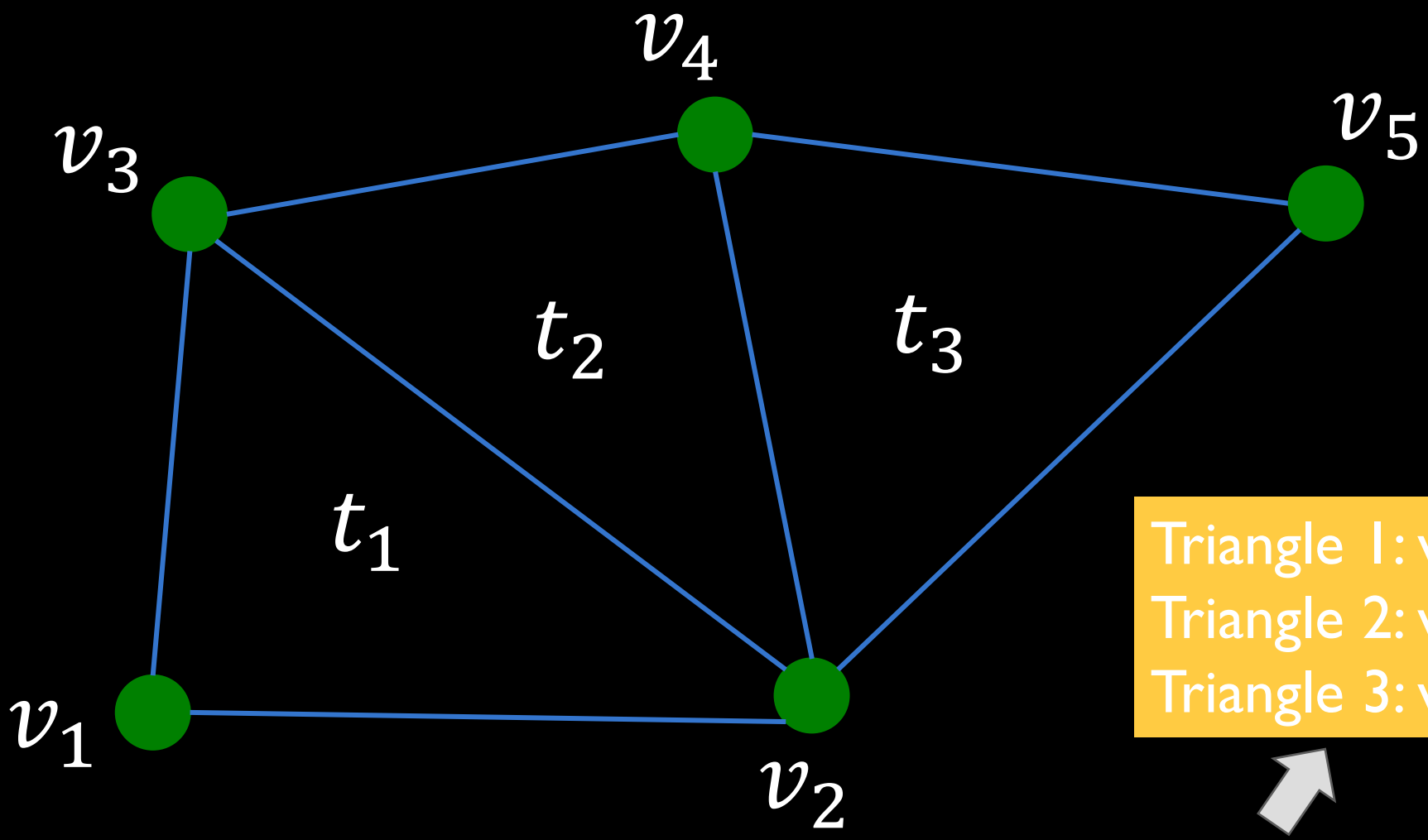


A Mesh with Three Disconnected Triangles



Triangle 1: v1, v2, v3
Triangle 2: v4, v5, v6
Triangle 3: v7, v8, v9

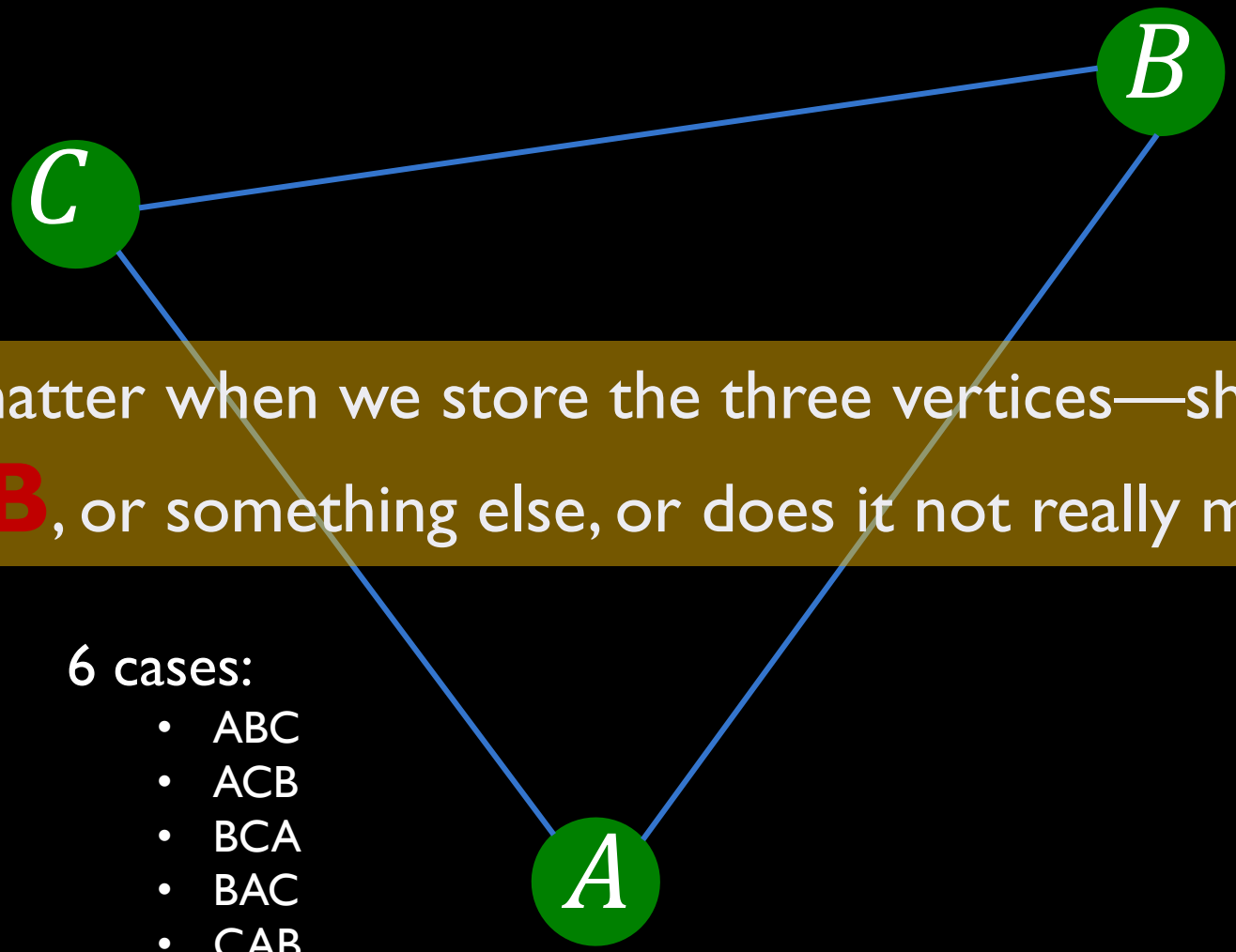
A Mesh with Three Connected Triangles



Triangle 1: v1, v2, v3
Triangle 2: v2, v3, v4
Triangle 3: v2, v4, v5

Look at the orders of these vertices

Triangle Vertex Order

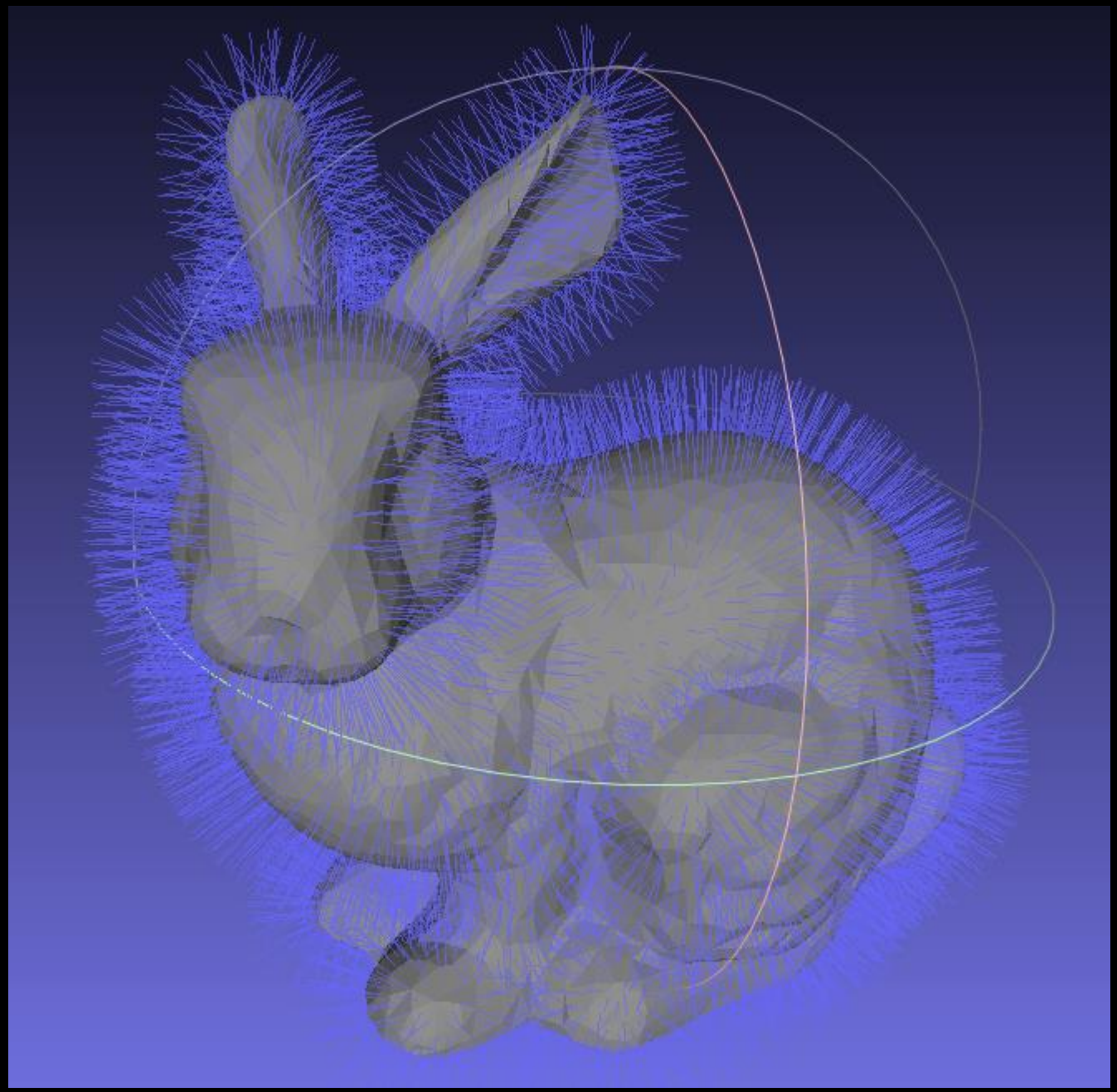


Does the order matter when we store the three vertices—should it be **ABC**, **ACB**, or something else, or does it not really matter?

6 cases:

- ABC
- ACB
- BCA
- BAC
- CAB
- CBA

Normal

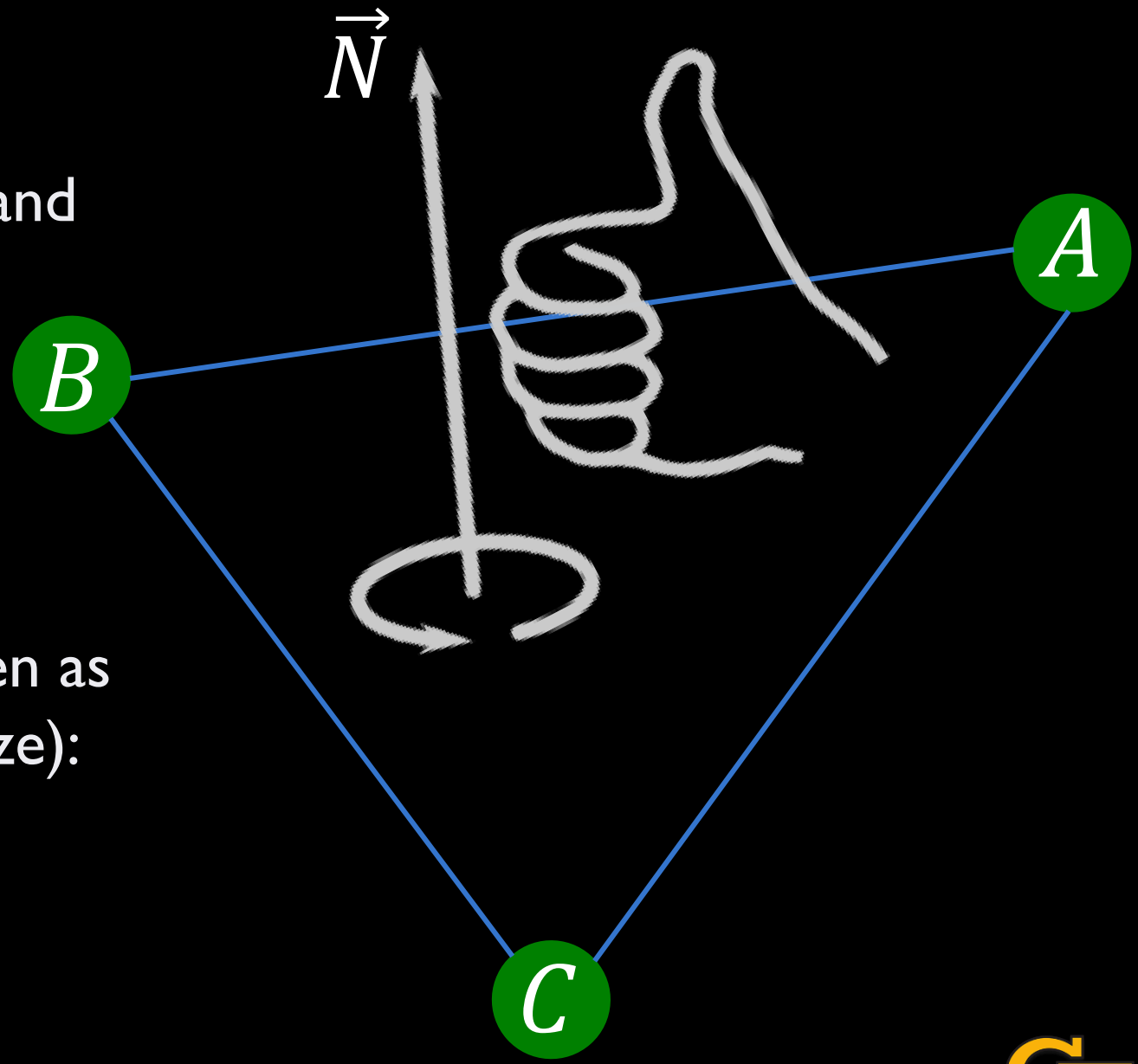


Triangle Normal

For triangle ABC, apply the right-hand rule to determine its normal:

- Fingers curl from vertex A to B,
- then sweep towards C.
- The thumb points to the normal vector.
- Mathematically, this can be written as cross product (and then normalize):

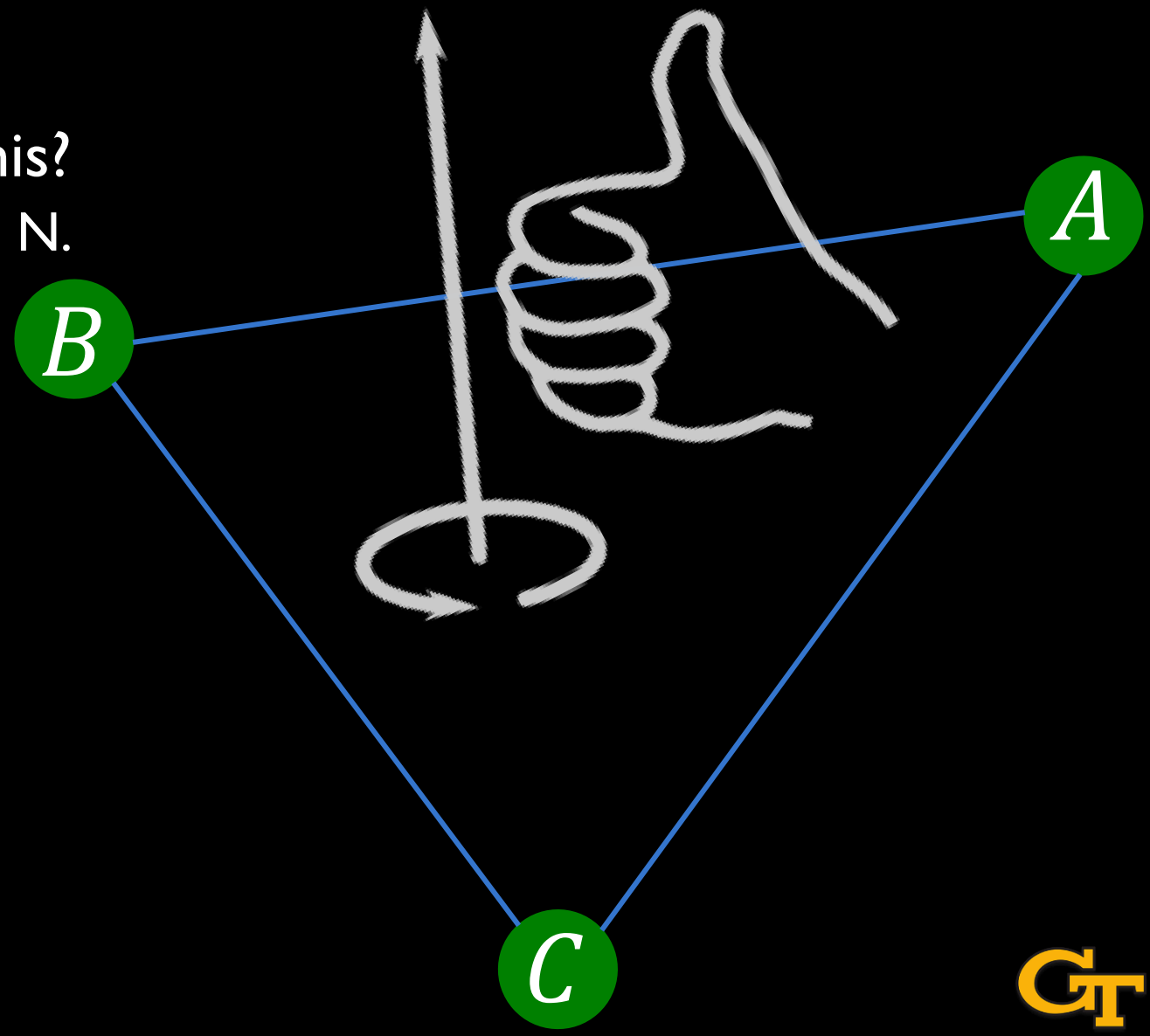
$$\vec{N} = \frac{\overrightarrow{AB} \times \overrightarrow{AC}}{|\overrightarrow{AB} \times \overrightarrow{AC}|}$$



Practice: Pseudocode for Normal Calculation

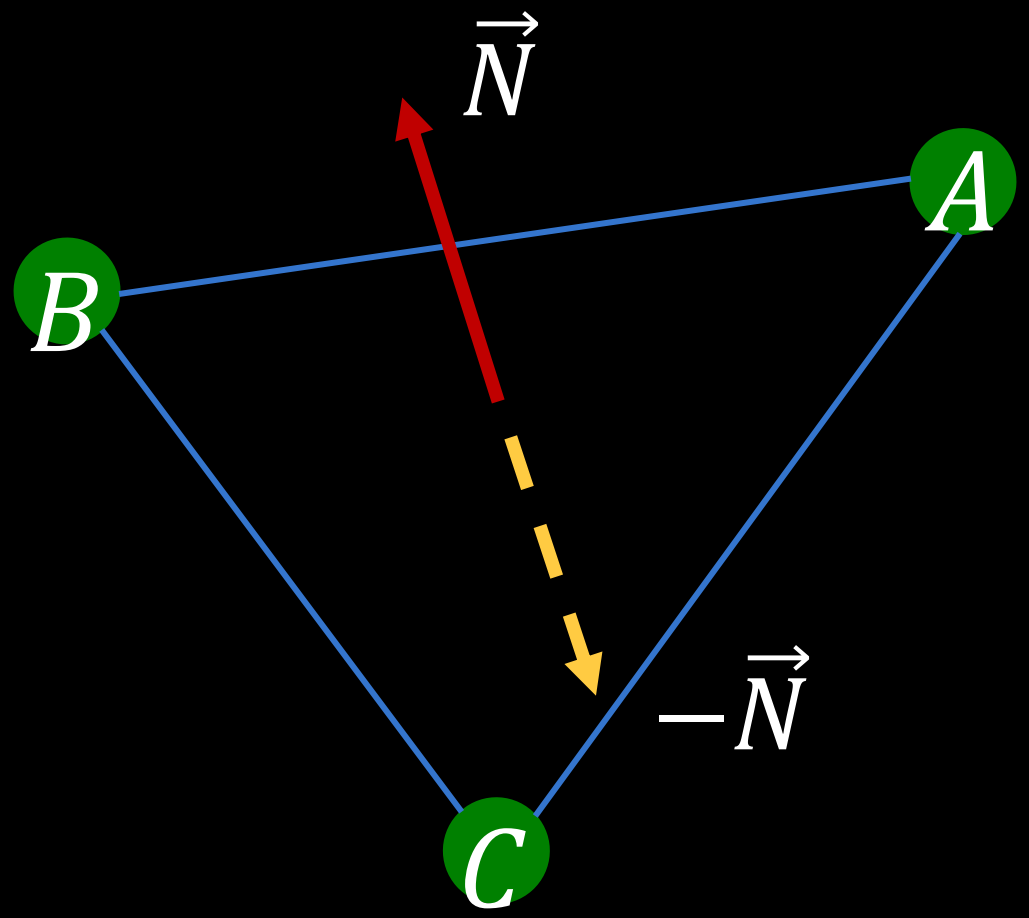
- Can you write pseudocode for this?
  - Given positions of A, B, C, calculate N.

$$\vec{N} = \frac{\overrightarrow{AB} \times \overrightarrow{AC}}{|\overrightarrow{AB} \times \overrightarrow{AC}|}$$

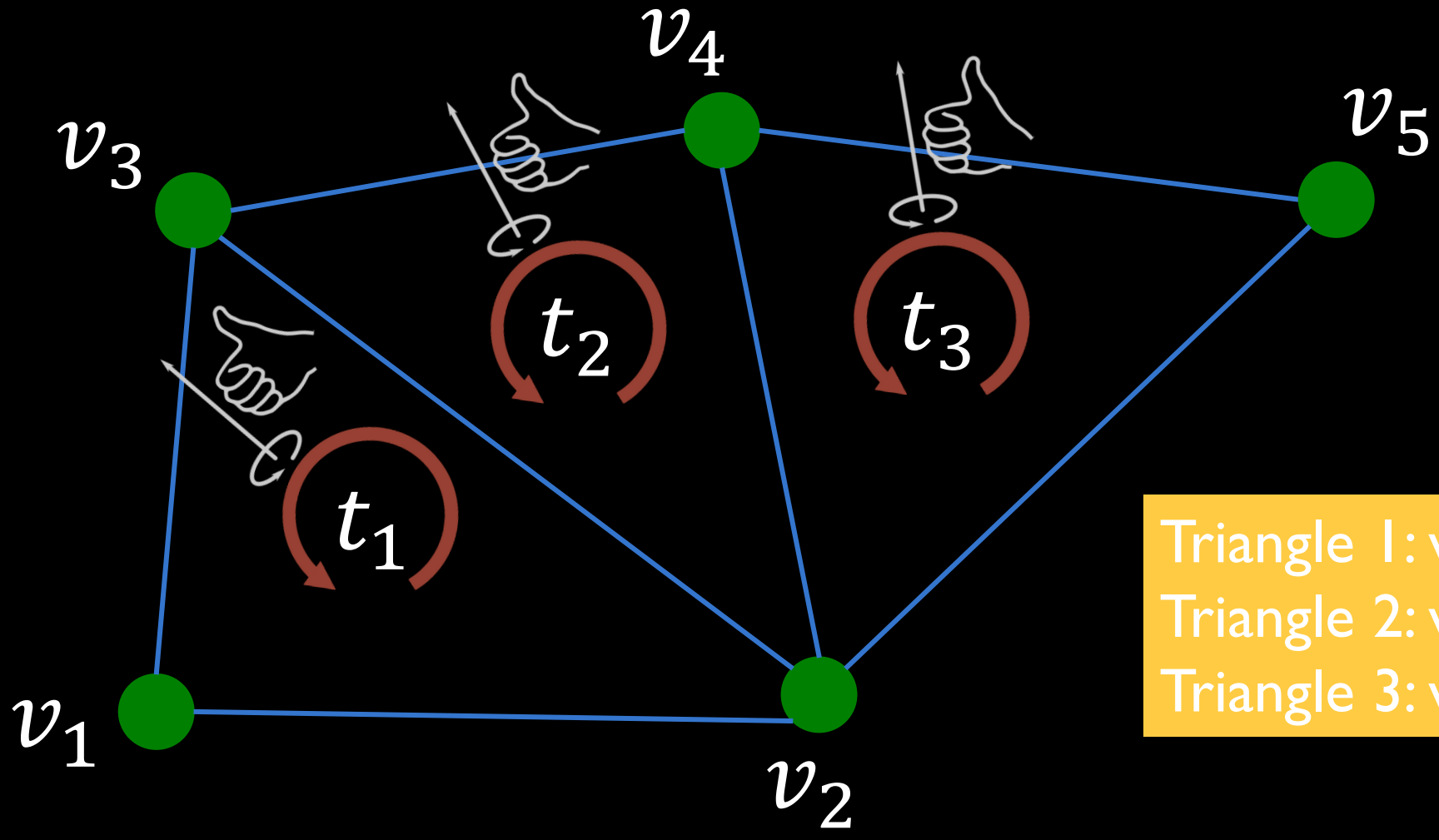


What is the Normal of Triangle ACB?

- ABC, BCA, CAB -> $\vec{N}$
- ACB, CBA, BAC -> $-\vec{N}$

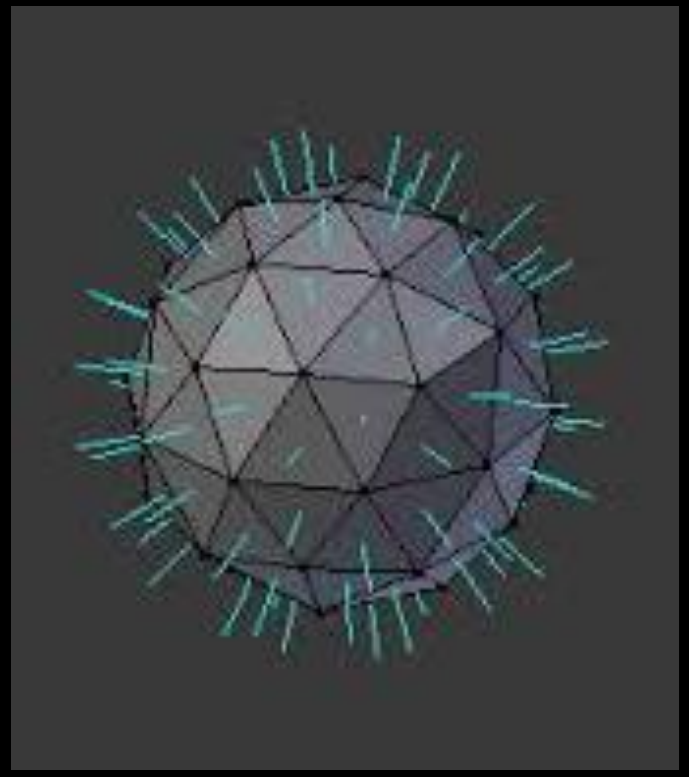


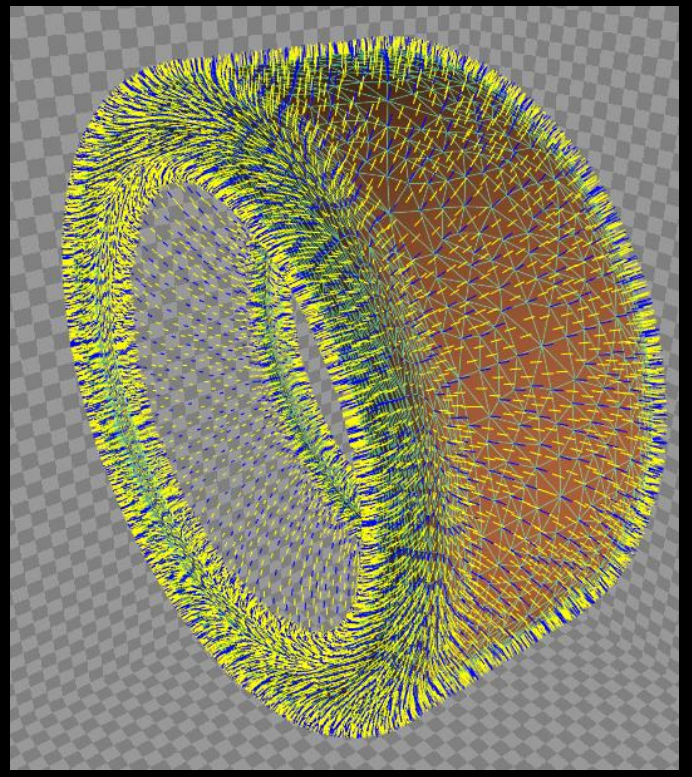
Mesh normals must point outward!

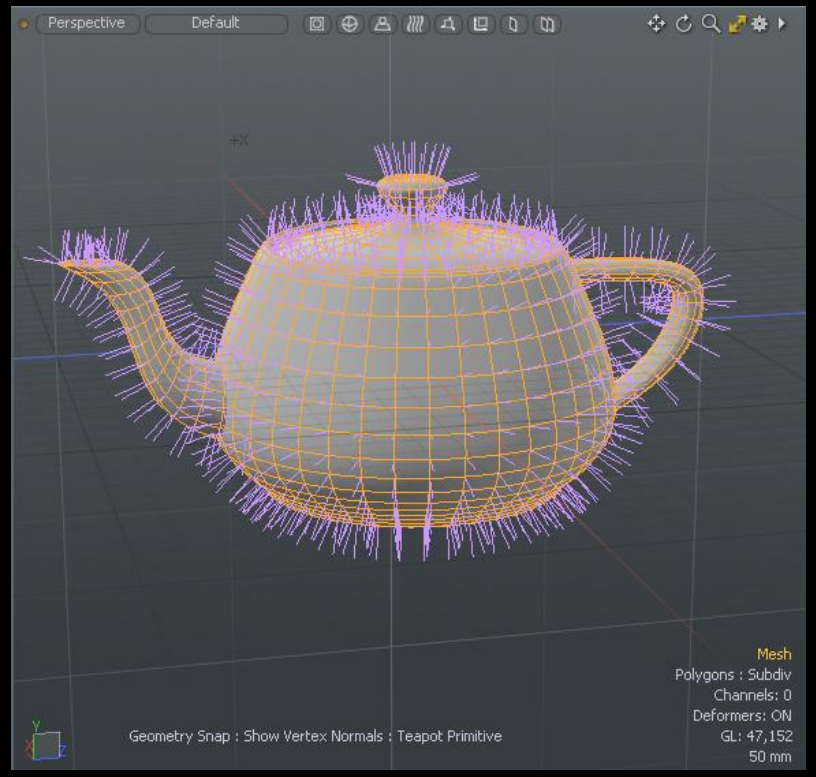


Triangle 1: v1, v2, v3
Triangle 2: v2, v4, v3
Triangle 3: v2, v5, v4

Example: Mesh Normal Visualization

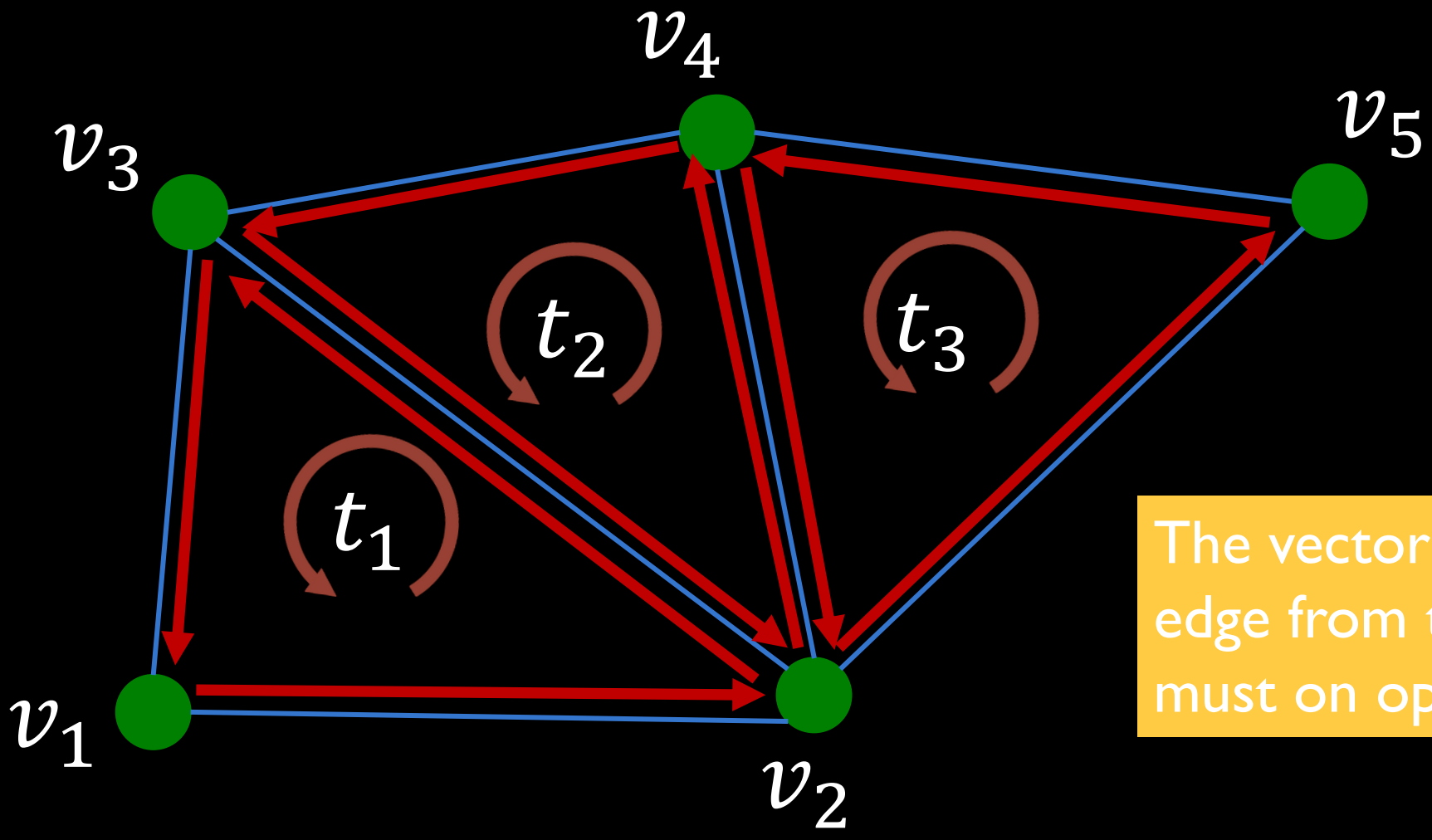






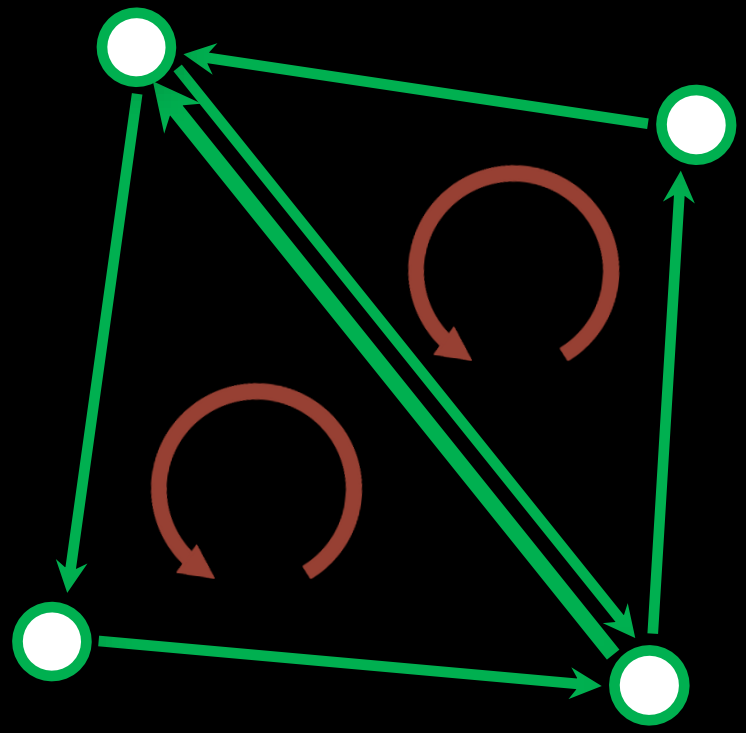
Notice the normal vectors are all pointing outward!

Visualize the Orientation with Vectors



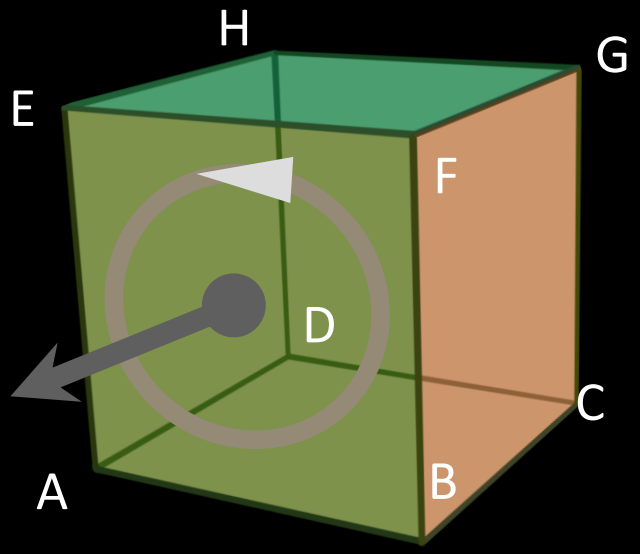
The vectors on each edge from two sides must on opposite!

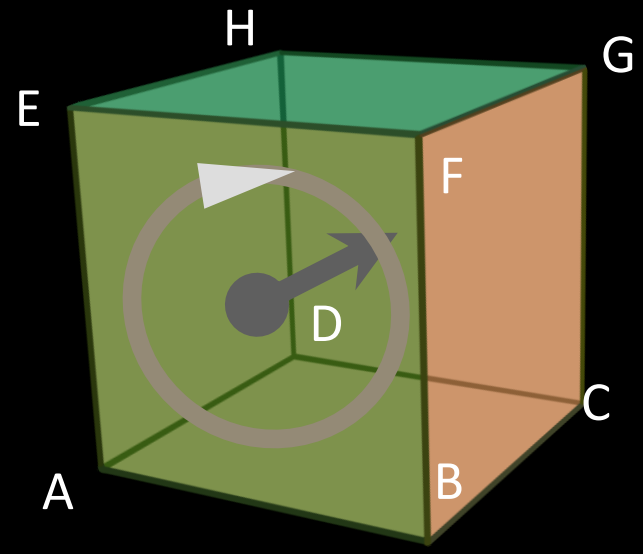
Orientation



Convention: Counter-clockwise Order

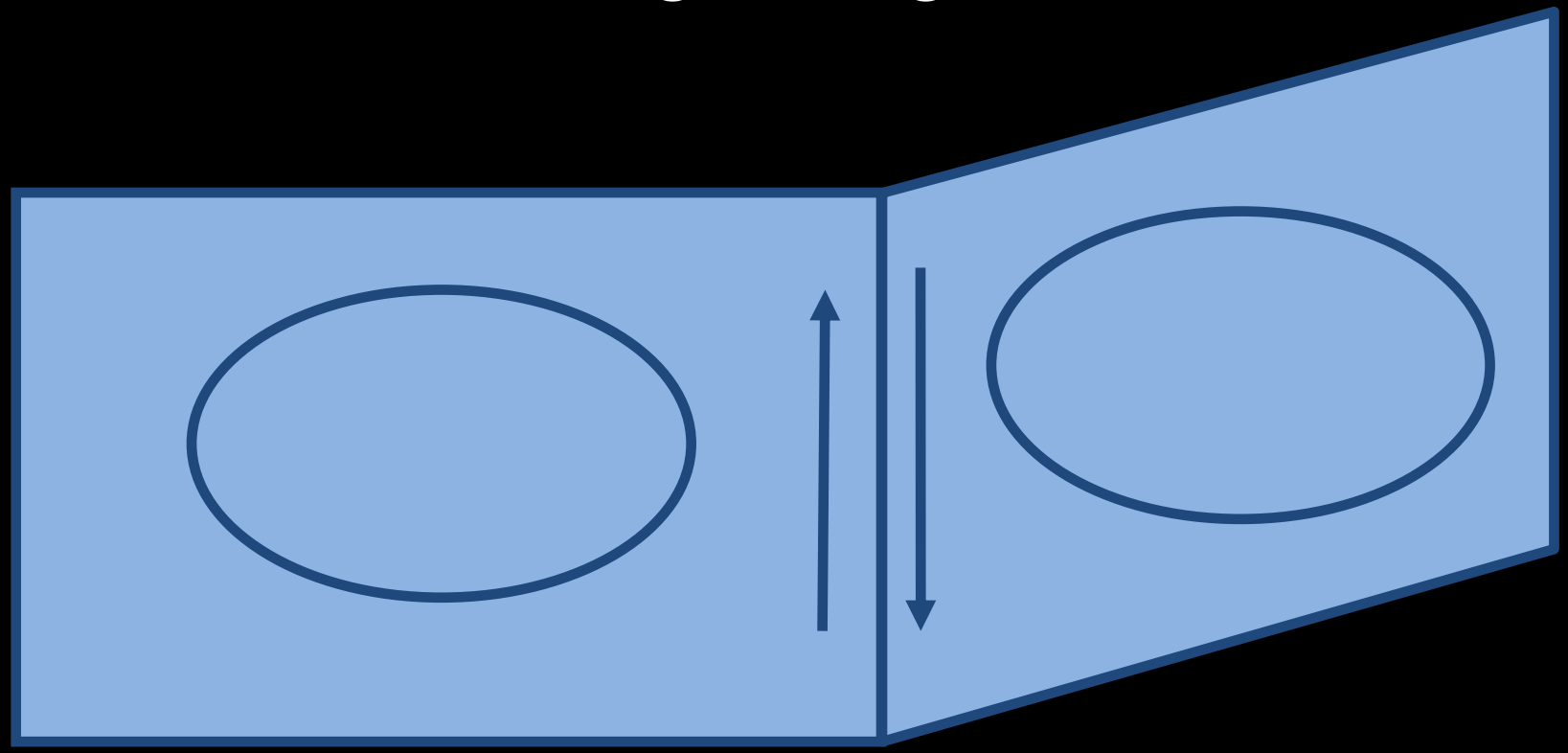
- Which side is the "front/outside" and which "back/inside"?
- Clockwise vs. counterclockwise order of face vertices defines sign/direction of the surface normal
- Convention: you are on the outside when you see the vertices in **counter-clockwise** order





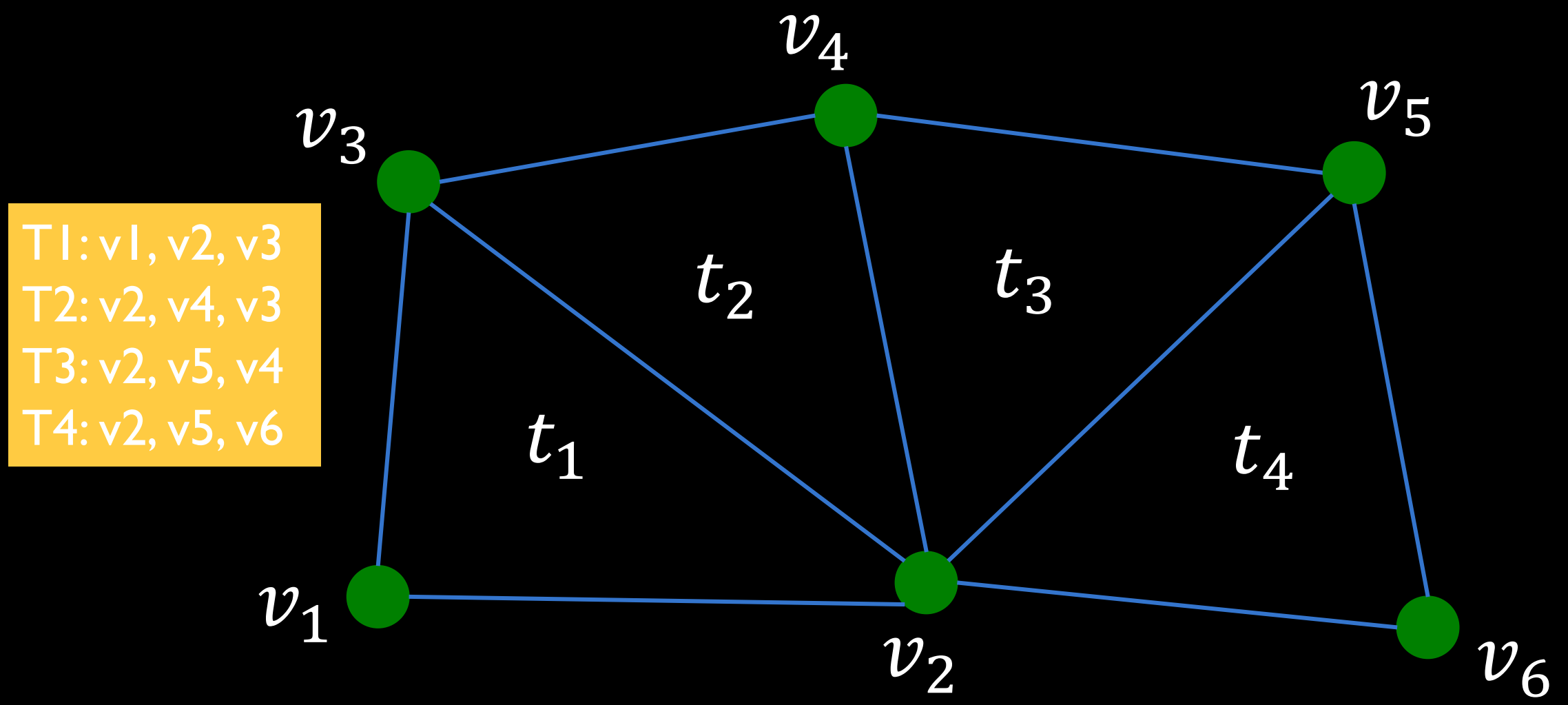
Orientation in Differential Geometry

- Consistent orientation of neighboring faces:



Reading material: Discrete Differential Geometry https://www.cs.cmu.edu/~kmcrane/Projects/DDG/paper.pdf

Practice: Which triangle has an inconsistent orientation?

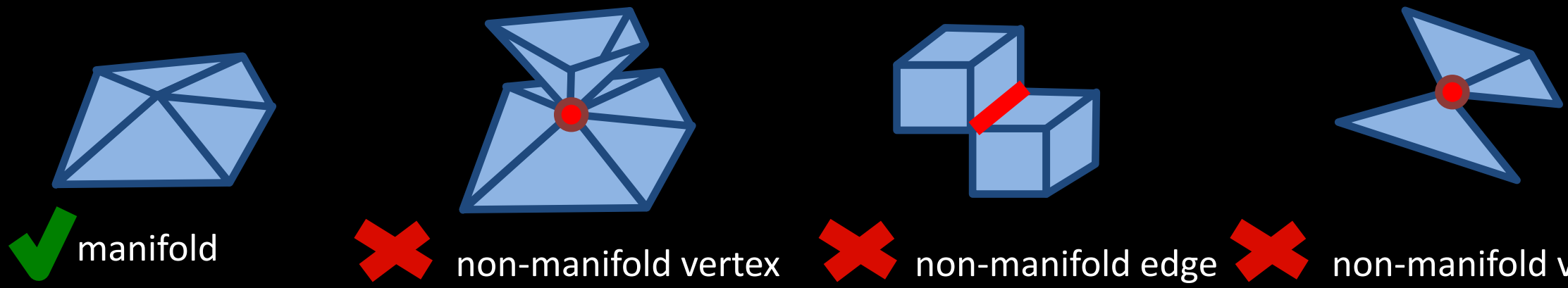


T1: v1, v2, v3
T2: v2, v4, v3
T3: v2, v5, v4
T4: v2, v5, v6

Topology

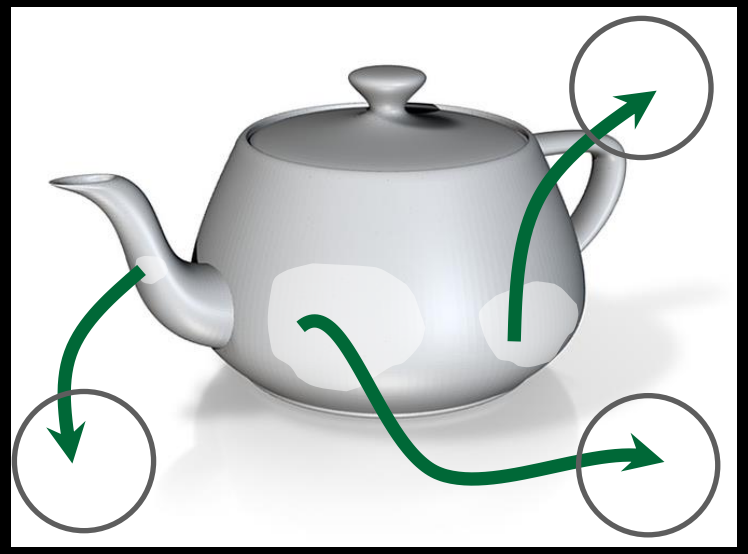
Mesh Manifolds

- 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)

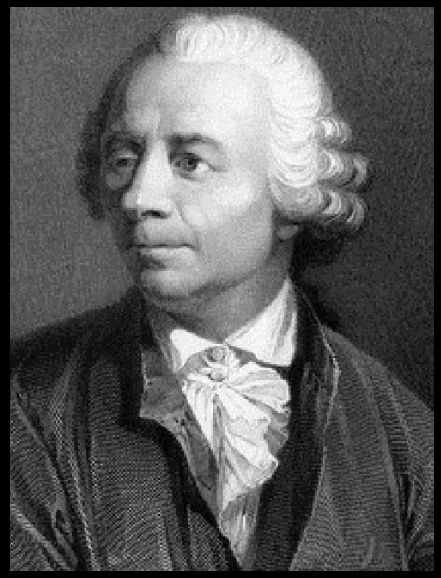


Manifold Boundary

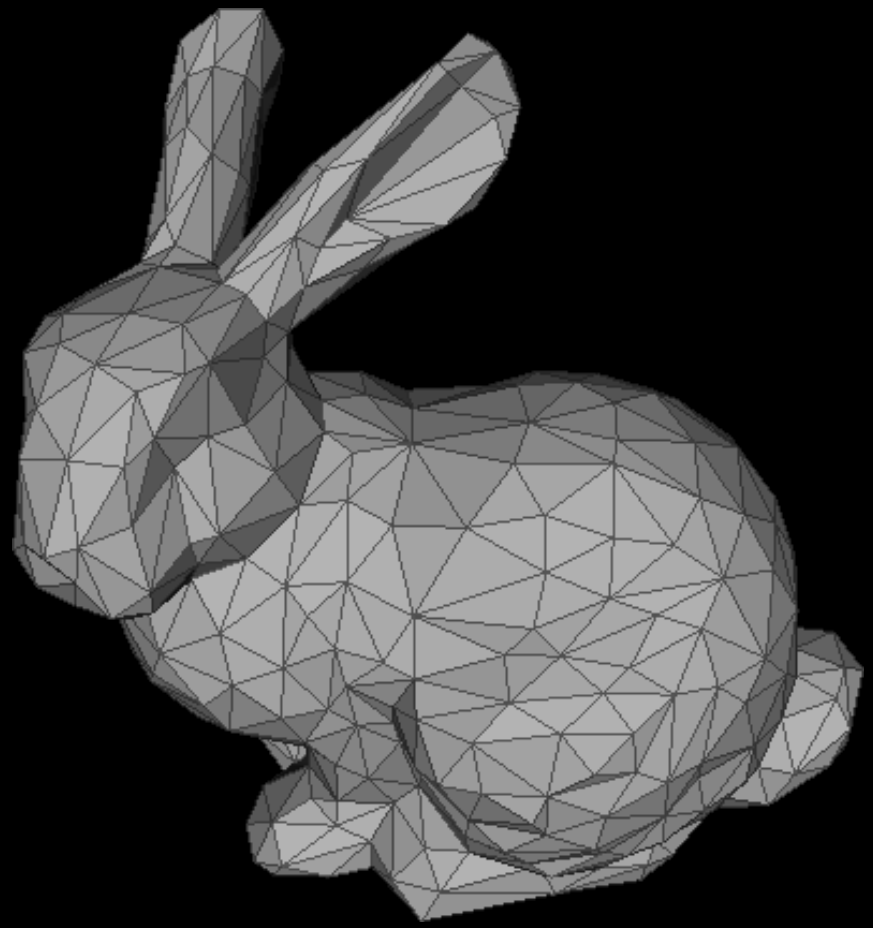
- Closed manifold: A surface is a closed **2-manifold** if it is everywhere locally homeomorphic to a disk
- Open manifold: The vicinity of each boundary point is homeomorphic to a half-disk



Euler-Poincare Formula



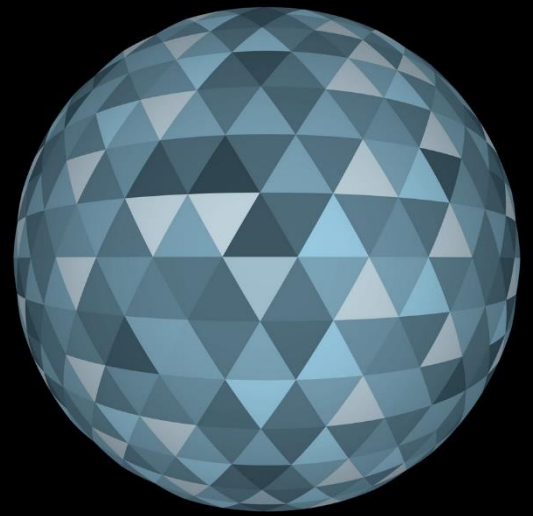
Question: Which number is the greatest, #vertices, #edges, or #faces?



Question: Ratios among #faces, #edges, and #vertices

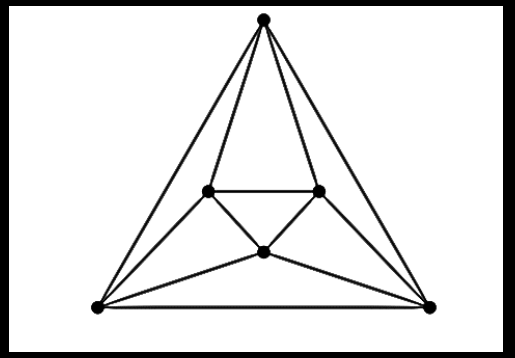
- Given an arbitrary triangle mesh, can we approximate the ratios among the number of triangles, edges, and vertices?

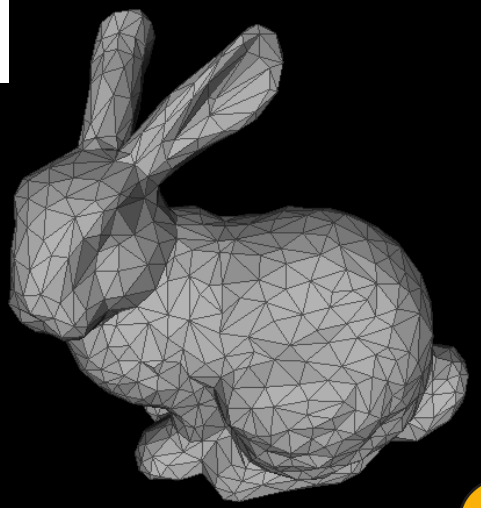
$$e:f:v \sim ?:?:?$$



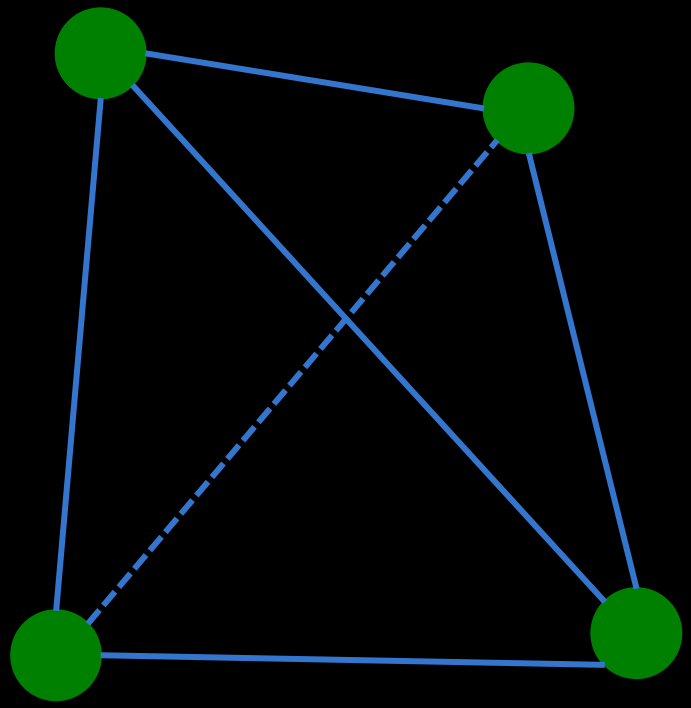
Euler-Poincaré Formula

- The sum of $v - e + f$ is **constant** for a given surface topology, no matter which (manifold) mesh we choose.
  - v = number of vertices
  - e = number of edges
  - f = number of faces
- v-e+f=1 for a planar graph
- v-e+f=2 for a closed manifold
  - Most triangle meshes are closed manifolds

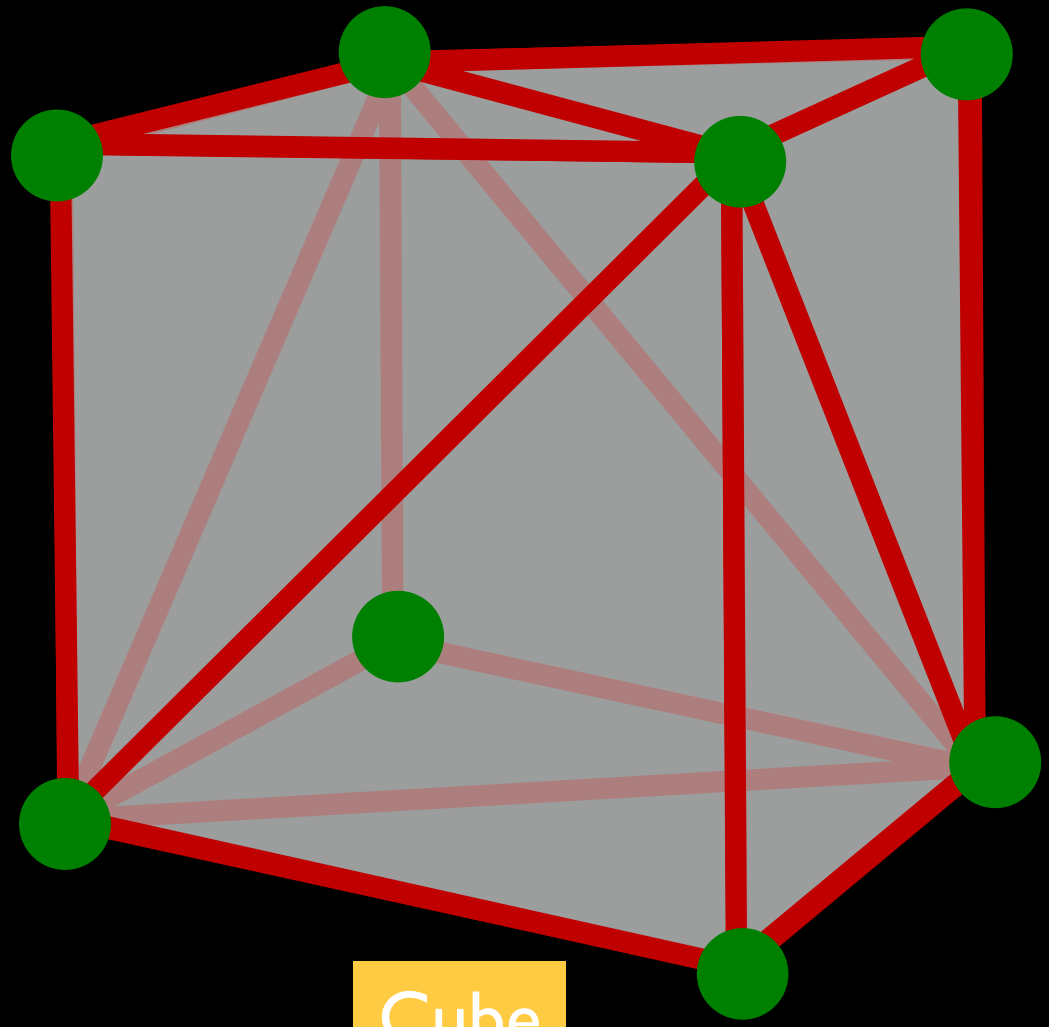




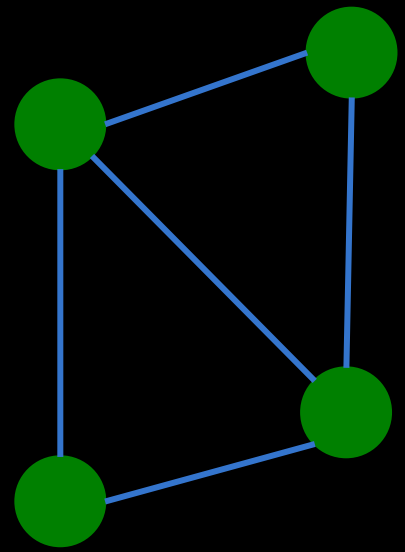
Let's check these shapes



Tetrahedron



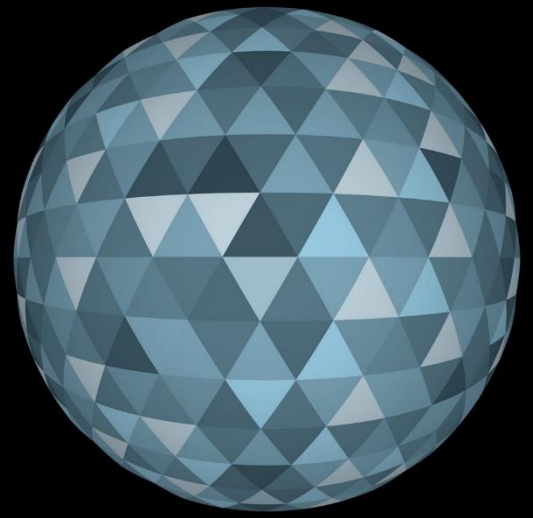
Cube



Quad

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: $e \sim 3v$
- e:f:v=3:2:1



Data Structure

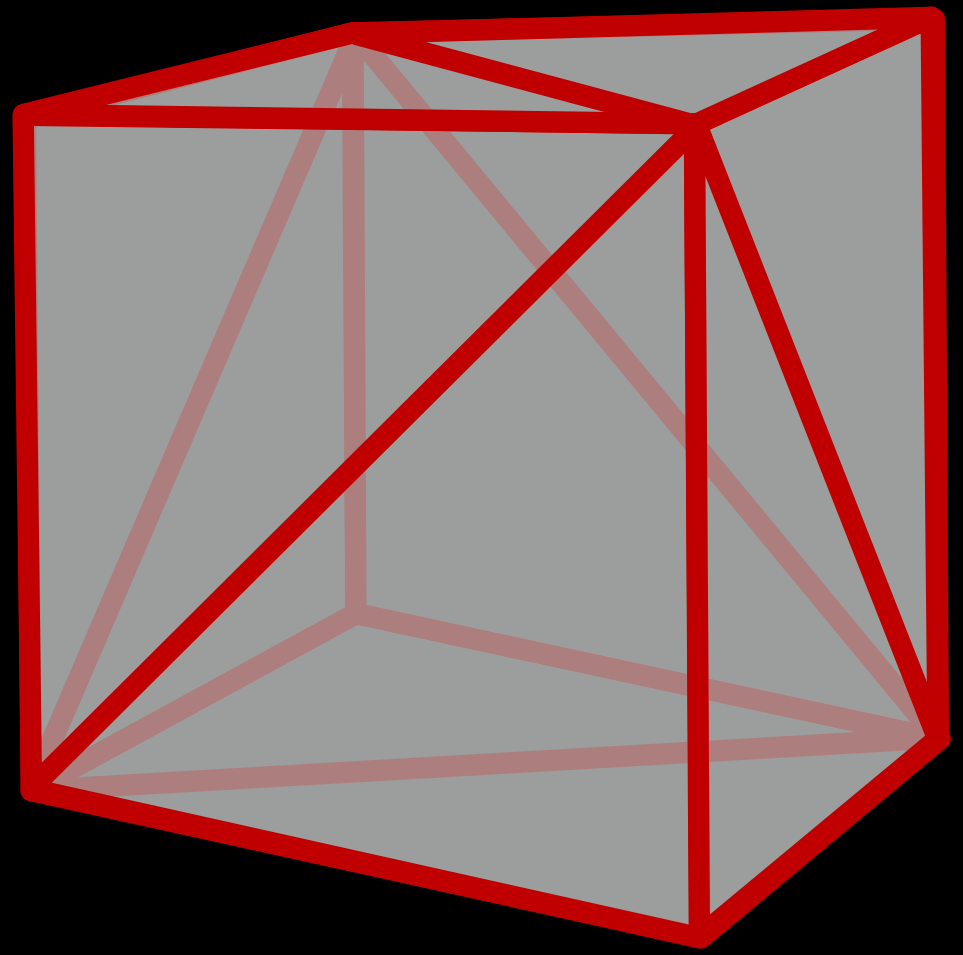
Triangles

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

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
...	...	...	...

How do we describe a triangle mesh?



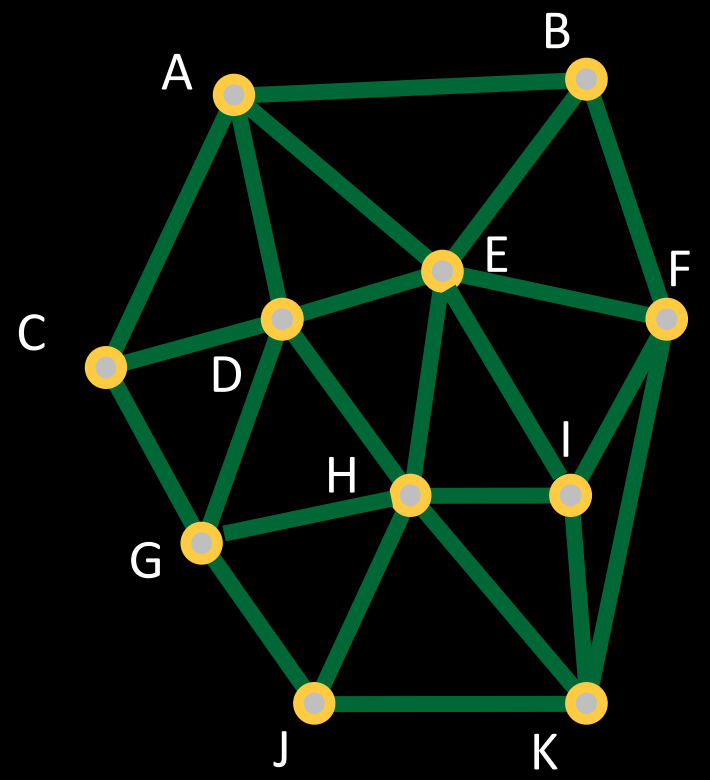
12 triangles, 8 vertices

Data Structures: What should be stored?

- **Geometry**
  - 3D coordinates
- **Topology**
  - Adjacency relationships
- **Attributes**
  - Normal, color, texture coordinates
  - Per vertex, face, edge

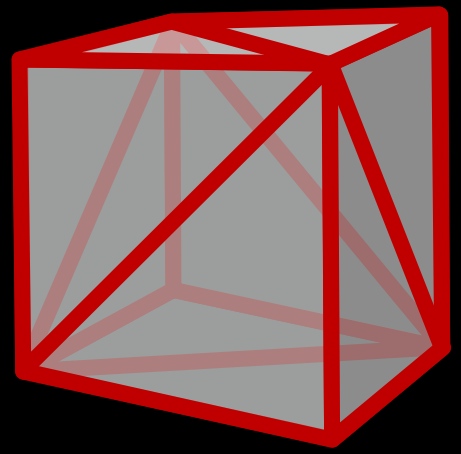
Data Structures: What should be supported?

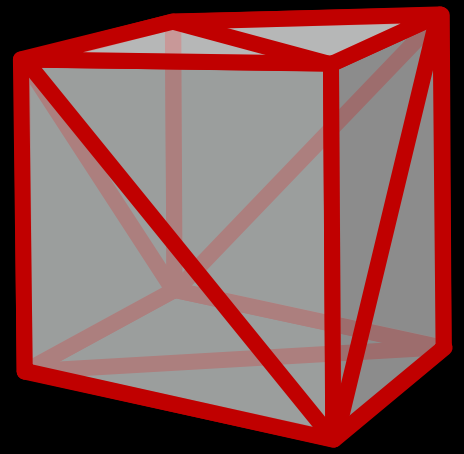
- **Rendering**
- **Geometry queries**
  - What are the vertices of face #2?
  - Is vertex A adjacent to vertex H?
  - Which faces are adjacent to face #1?
- **Modifications**
  - Remove/add a vertex/face
  - Vertex split, edge collapse



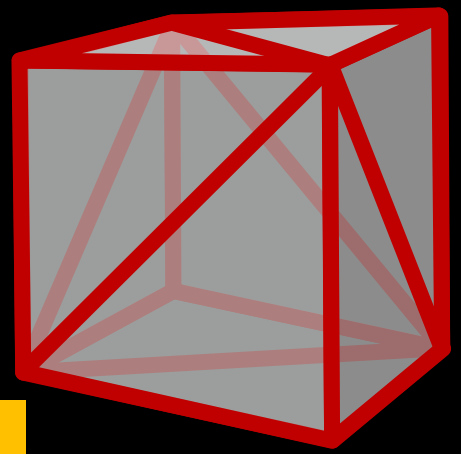
Topology vs. Geometry

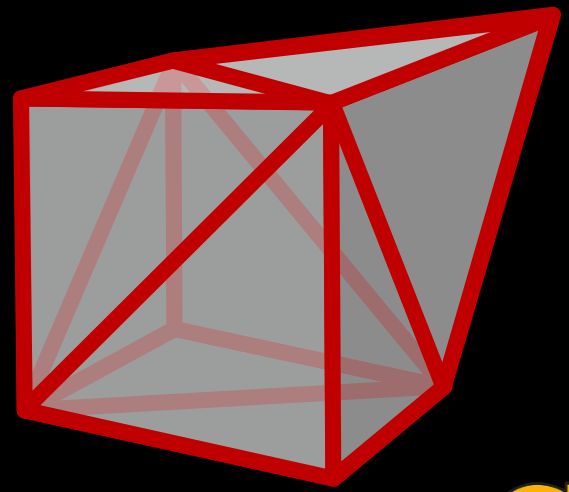
- **Topology**: how the triangles are connected (ignoring positions)
  - Same geometry
  - Different mesh topology





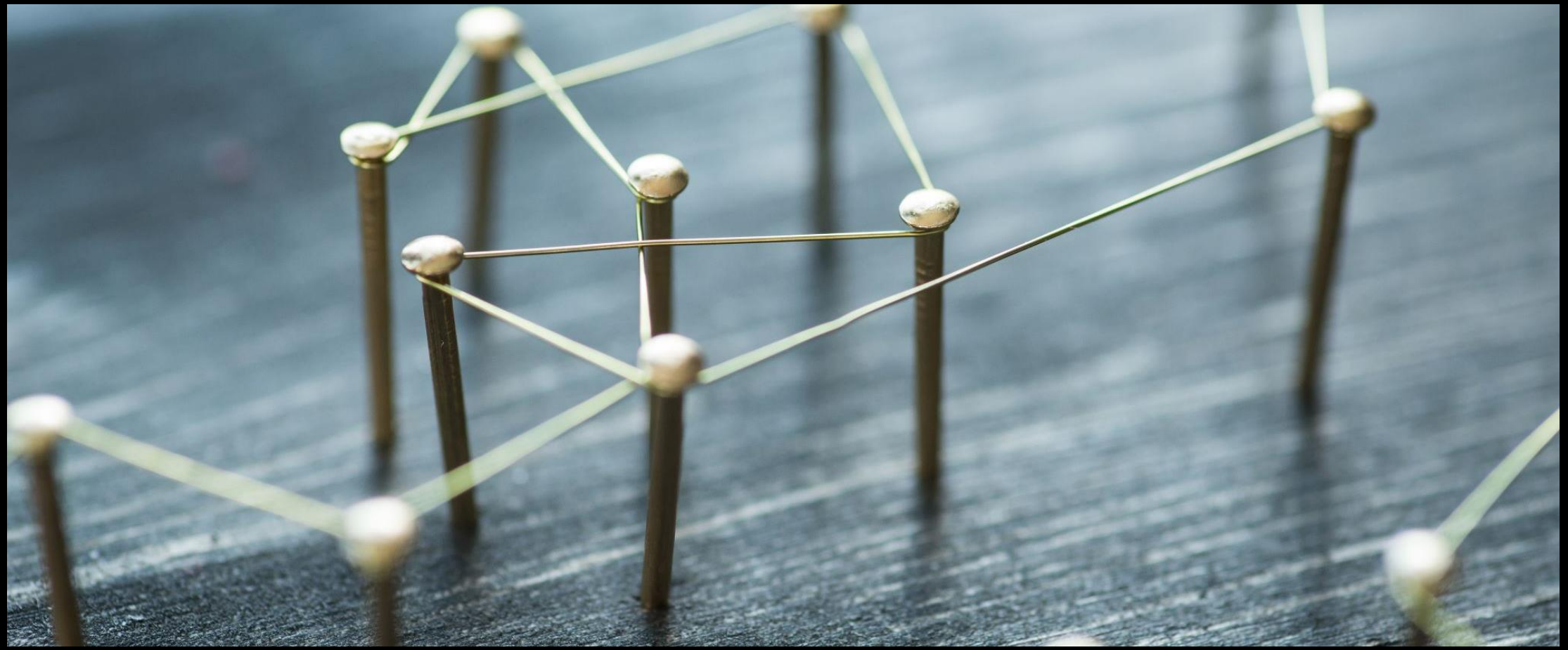
- **Geometry**: where the triangles are in 3D space
  - Same mesh topology
  - Different geometry





Topology specifies the connectivities among vertices and triangles
Geometry specifies the shape of the object

Can we brainstorm some way to store a triangle mesh?



Separate Triangle List or Face Set (STL)

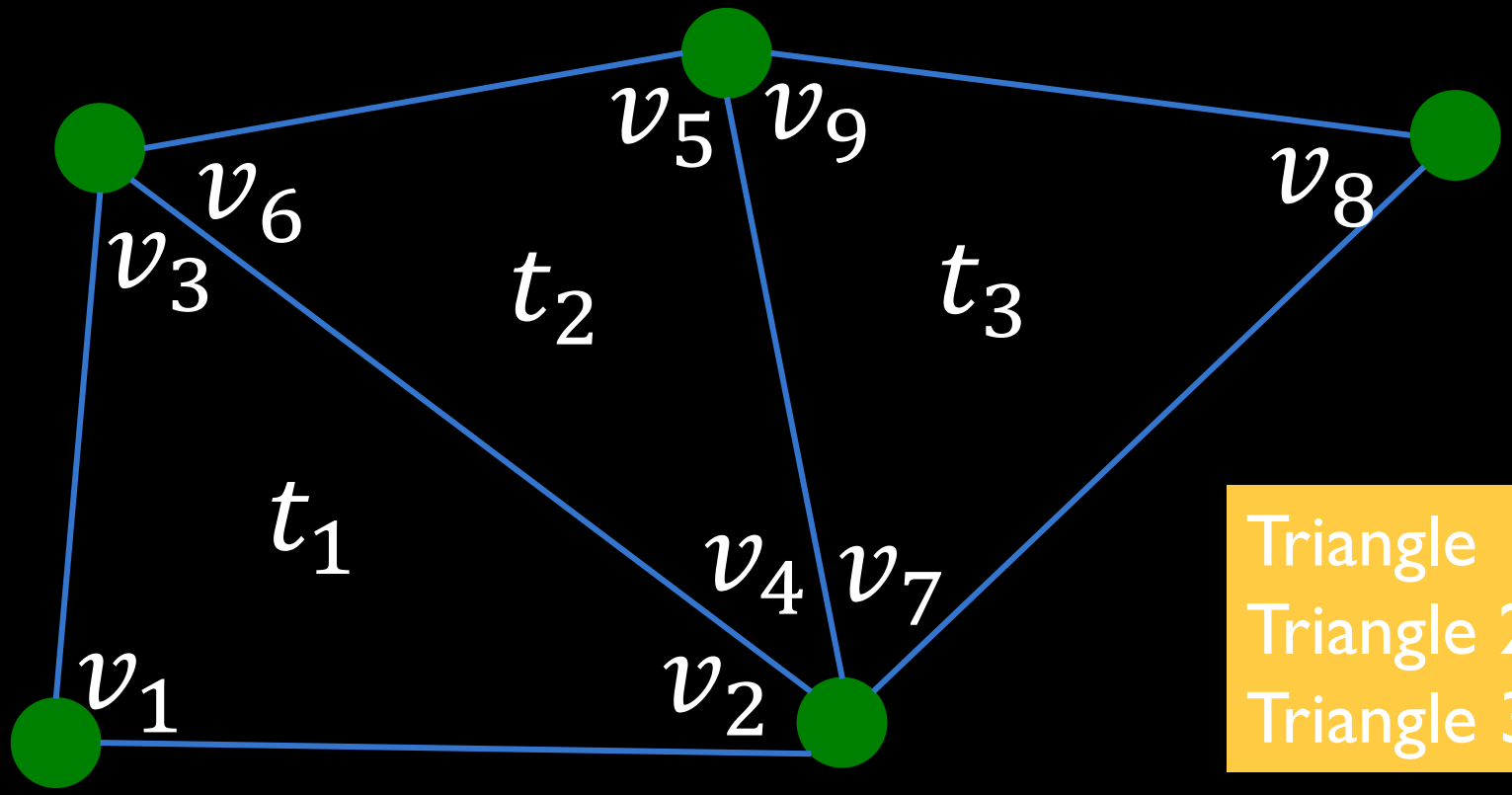
- Face:
  - 3 vertex positions
- Storage:
  - 4 Bytes/coordinate
  - 36 Bytes/face
- **Store a single array of vertex positions**
- **Wastes space**
- **No connectivity information**

Triangles

0	x0	y0	z0
1	x1	y1	z1
2	x2	y2	z2
3	x3	y3	z3
4	x4	y4	z4
5	x5	y5	z5
6	x6	y6	z6
…	…	…	…

x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	x_4	y_4	z_4	…

Example: STL Format



Triangle 1: v1, v2, v3
Triangle 2: v4, v5, v6
Triangle 3: v7, v8, v9

x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	x_4	y_4	z_4

• • •

27 floats in total

Indexed Face Set (OBJ, OFF, WRL)

- Vertex: position
- Face: vertex indices
- Storage:
  - 3 floats per vertex
  - 3 ints per face
- **Separates topology from geometr**
- **No explicit neighborhood info**

Triangles

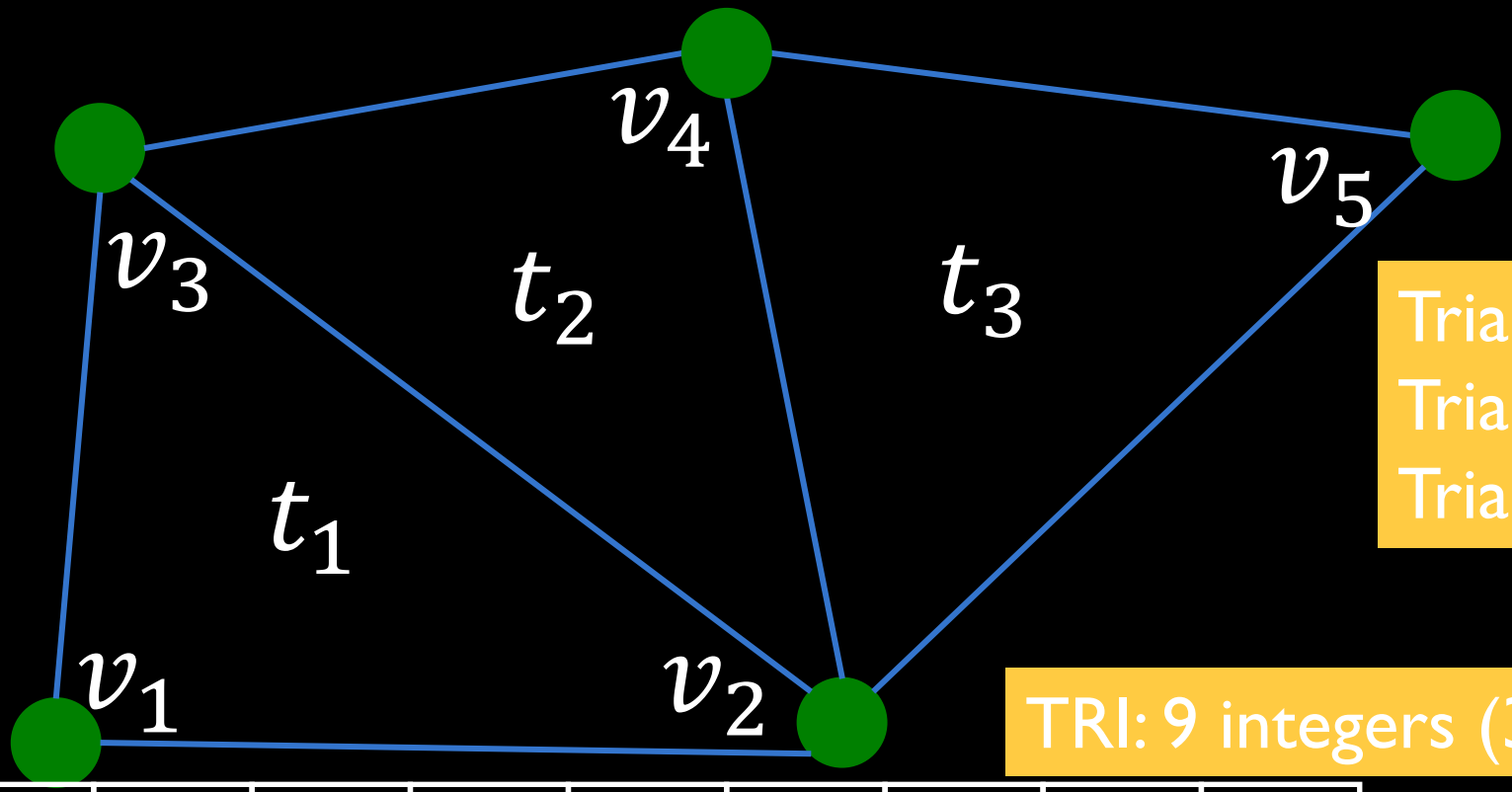
t0	v0	v1	v2
t1	v0	v1	v3
t2	v2	v4	v3
t3	v5	v2	v6
…	…	…	…

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
…	…	…	…

E.g., How do we find the neighboring faces of a given edge?

Example: OBJ Format



Triangle 1: v1, v2, v3
Triangle 2: v2, v4, v3
Triangle 3: v2, v5, v4

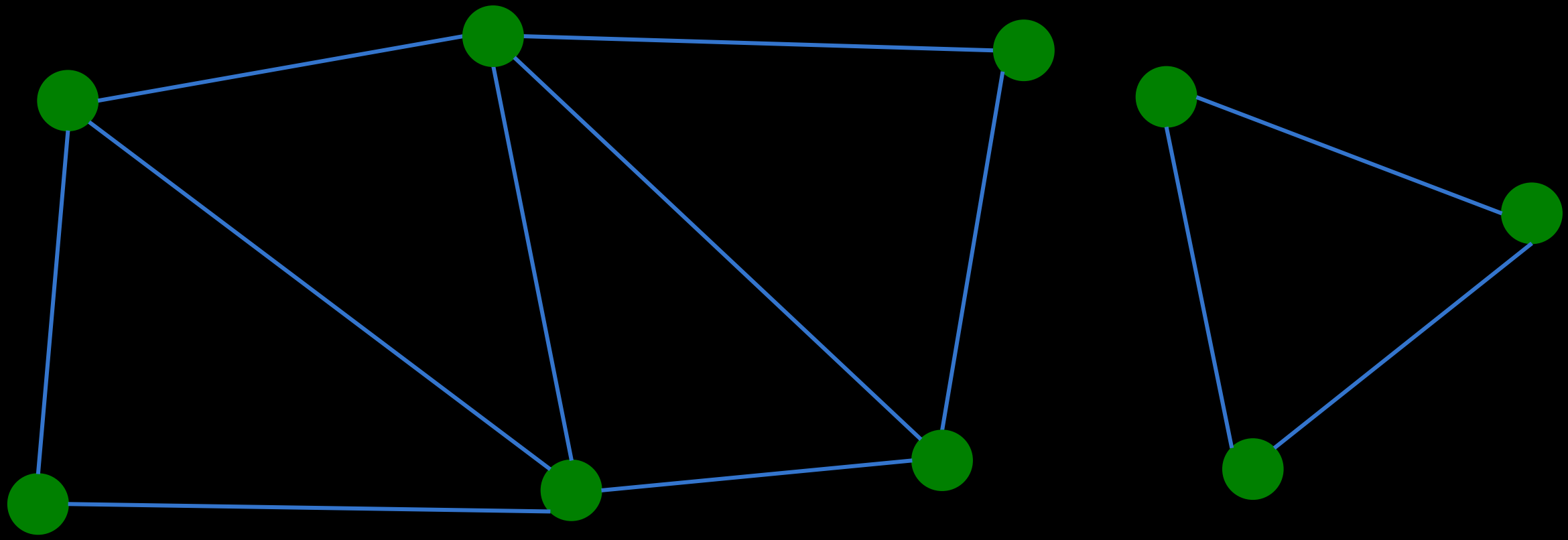
TRI: 9 integers (3 Vector3i)

1	2	3	2	4	3	2	5	4

VTX: 15 floats (5 Vector3f)

x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	x_4	y_4	z_4	x_5	y_5	z_5

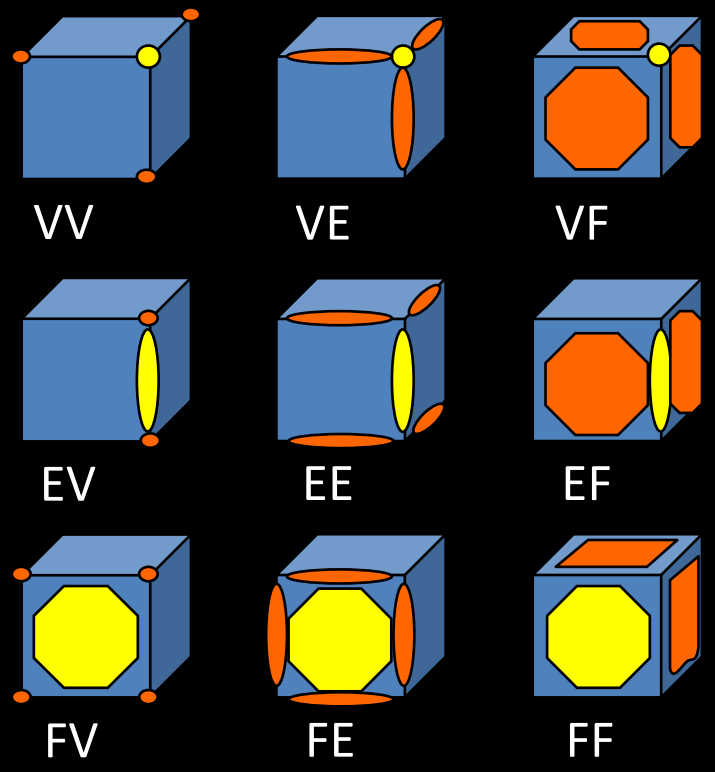
Practice: Can you write the TRI and VTX arrays for the following mesh?



Topology Queries

- All possible neighborhood relationships:

Vertex	–	Vertex	VV
Vertex	–	Edge	VE
Vertex	–	Face	VF
Edge	–	Vertex	EV
Edge	–	Edge	EE
Edge	–	Face	EF
Face	–	Vertex	FV
Face	–	Edge	FE
Face	–	Face	FF



We'd like *O*(1) time for topology queries!

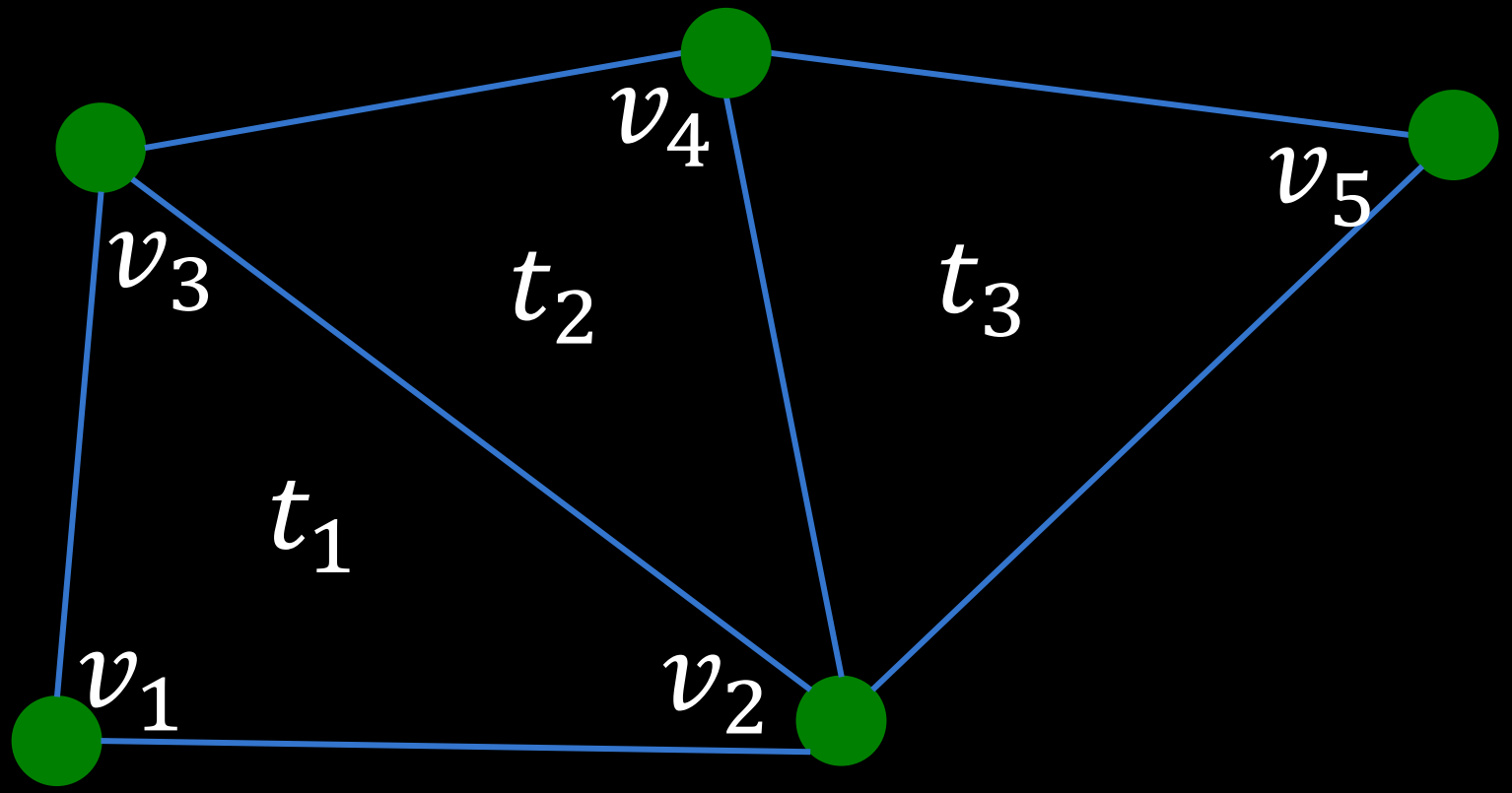
Solution: Auxiliary Data Structures

- E.g., store an unordered map to maintain the vertex-vertex relationship, and store another unordered map to maintain the vertex-edge relationship

C++ Code:

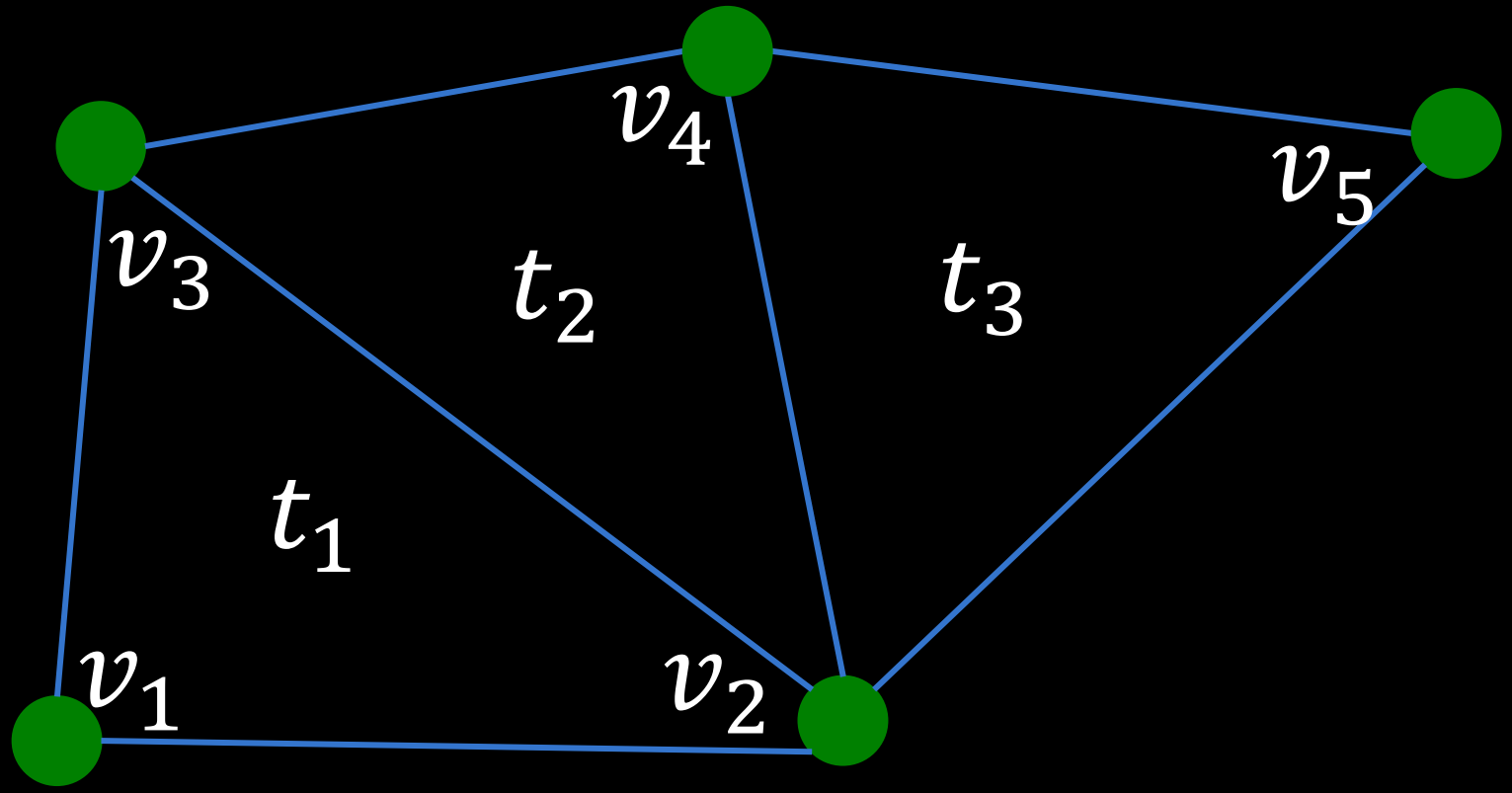
```
std::unordered_map<Vector2i,std::vector<int> > edge_tri_map;
std::unordered_map<int,std::vector<int> > vtx_vtx_map;
```

Example: Vtx-Vtx Map



1	2, 3
2	1, 3, 4, 5
3	1, 2, 4
4	2, 3, 5
5	2, 4

Example: Edge-Tri Map



1,2	1
2,3	1, 2
1,3	1
2,4	2, 3
3,4	2
2,5	3
4,5	3