Mesh Representations & Geometry Processing

Computer Graphics and Imaging
UC Berkeley CS184

Announcements

Congratulations on finishing 1/4 of the class!

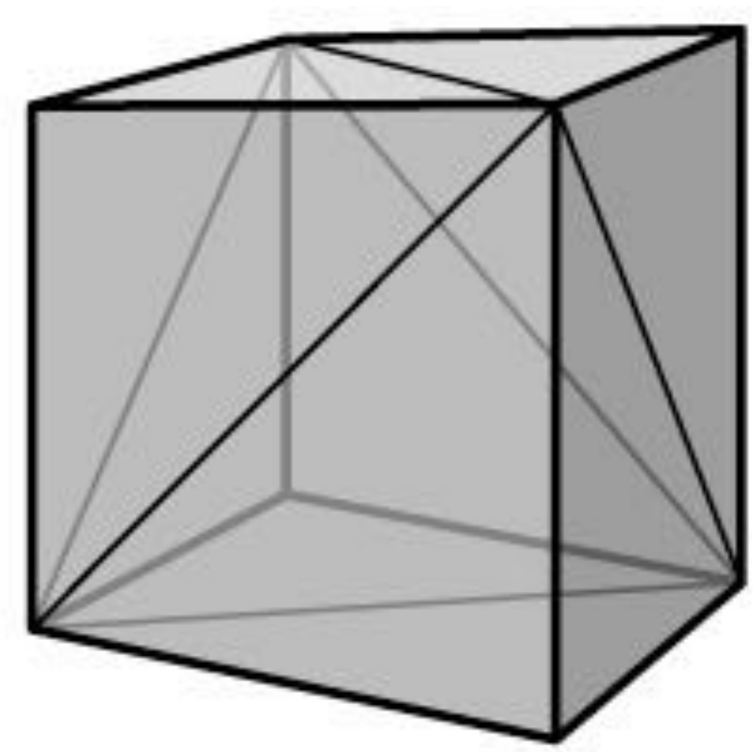
Week 1-2 Survey was released — please fill it out! Check Piazza for details.

Assignment 2 released and due Friday!

Today: Meshes & Geometry Processing review, demo via Assignment 2!

Tomorrow/This week: Raytracing!!!

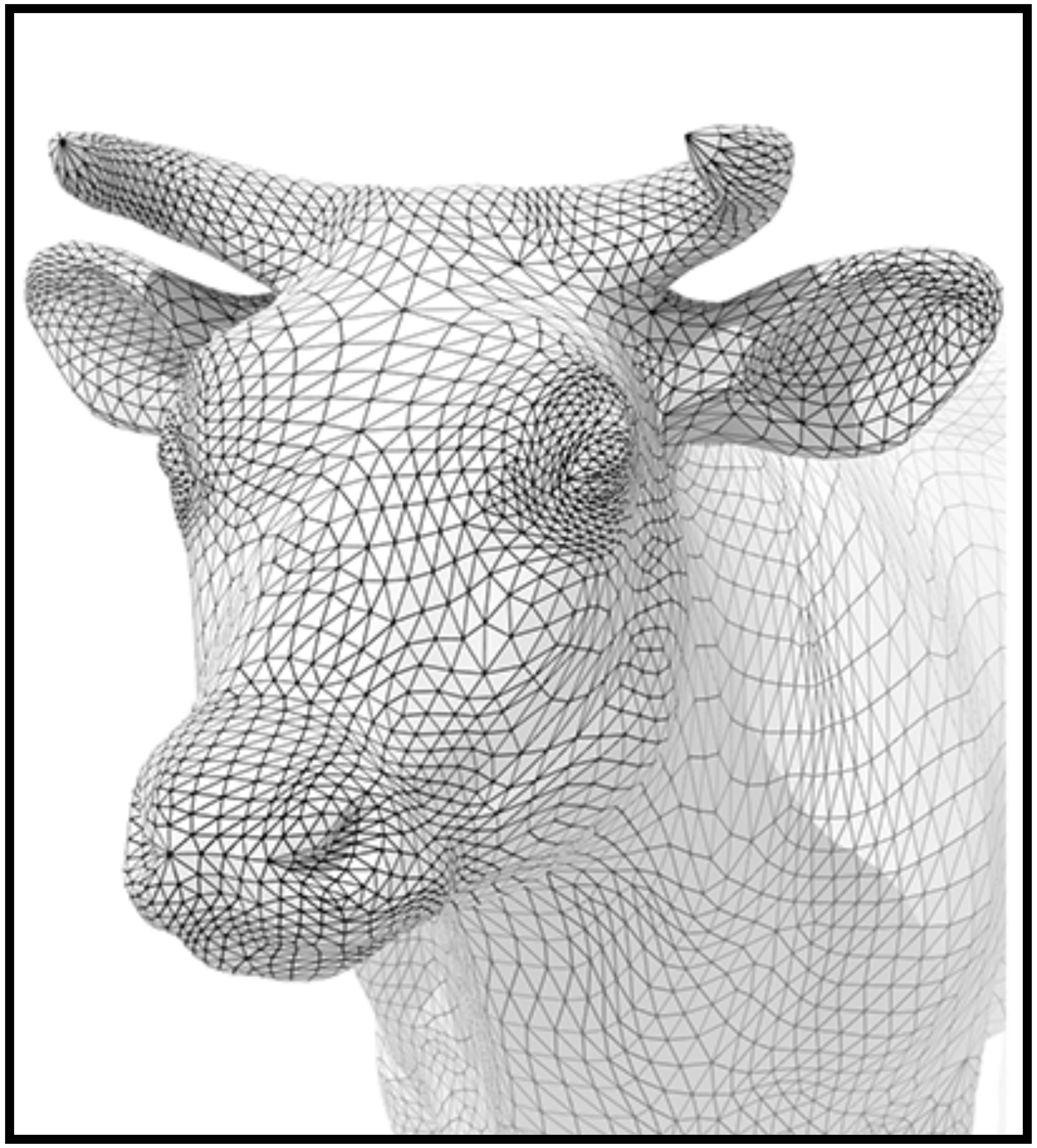
A Small Triangle Mesh

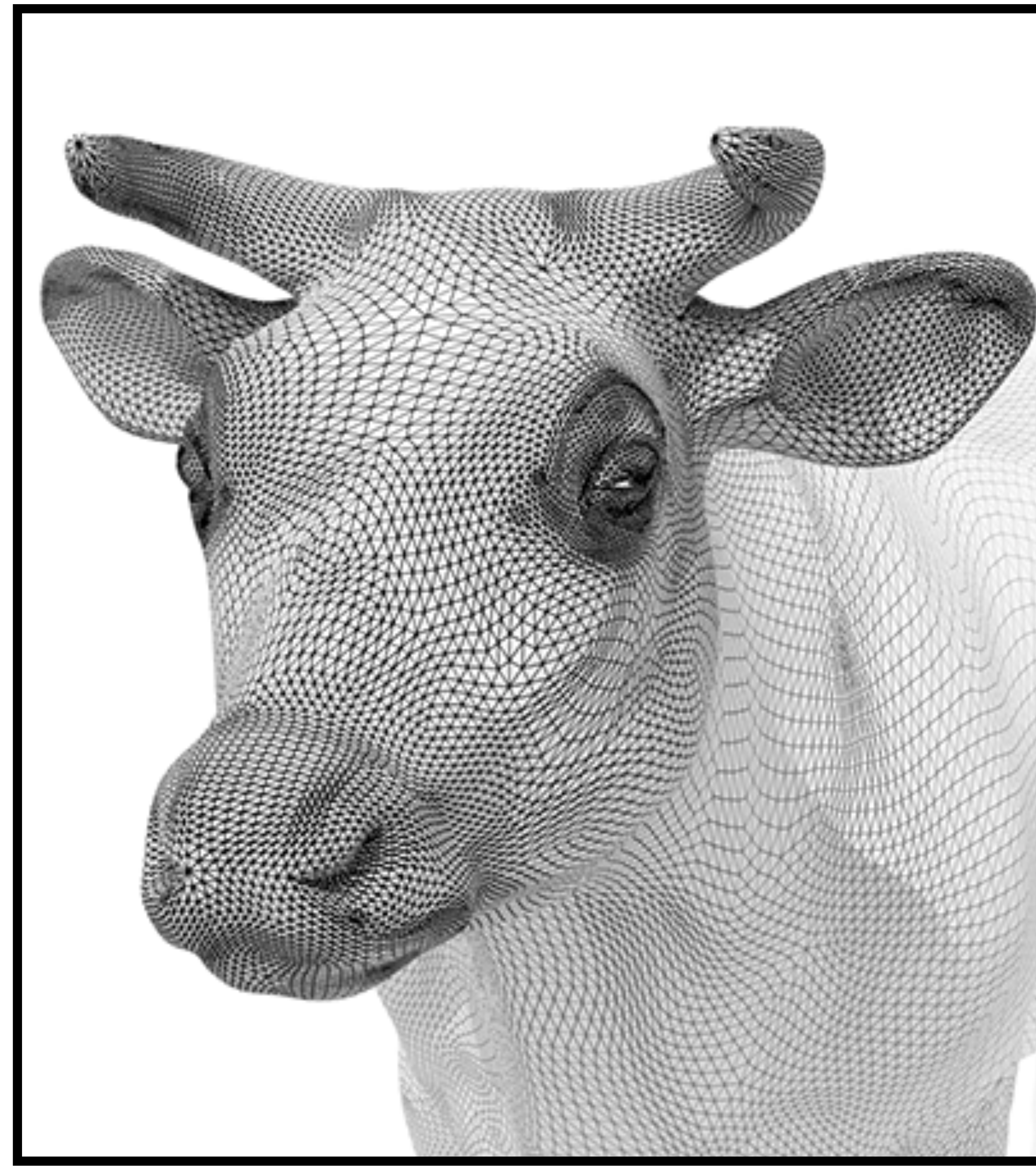


8 vertices, 12 triangles

Geometry Processing Tasks: 3 Examples

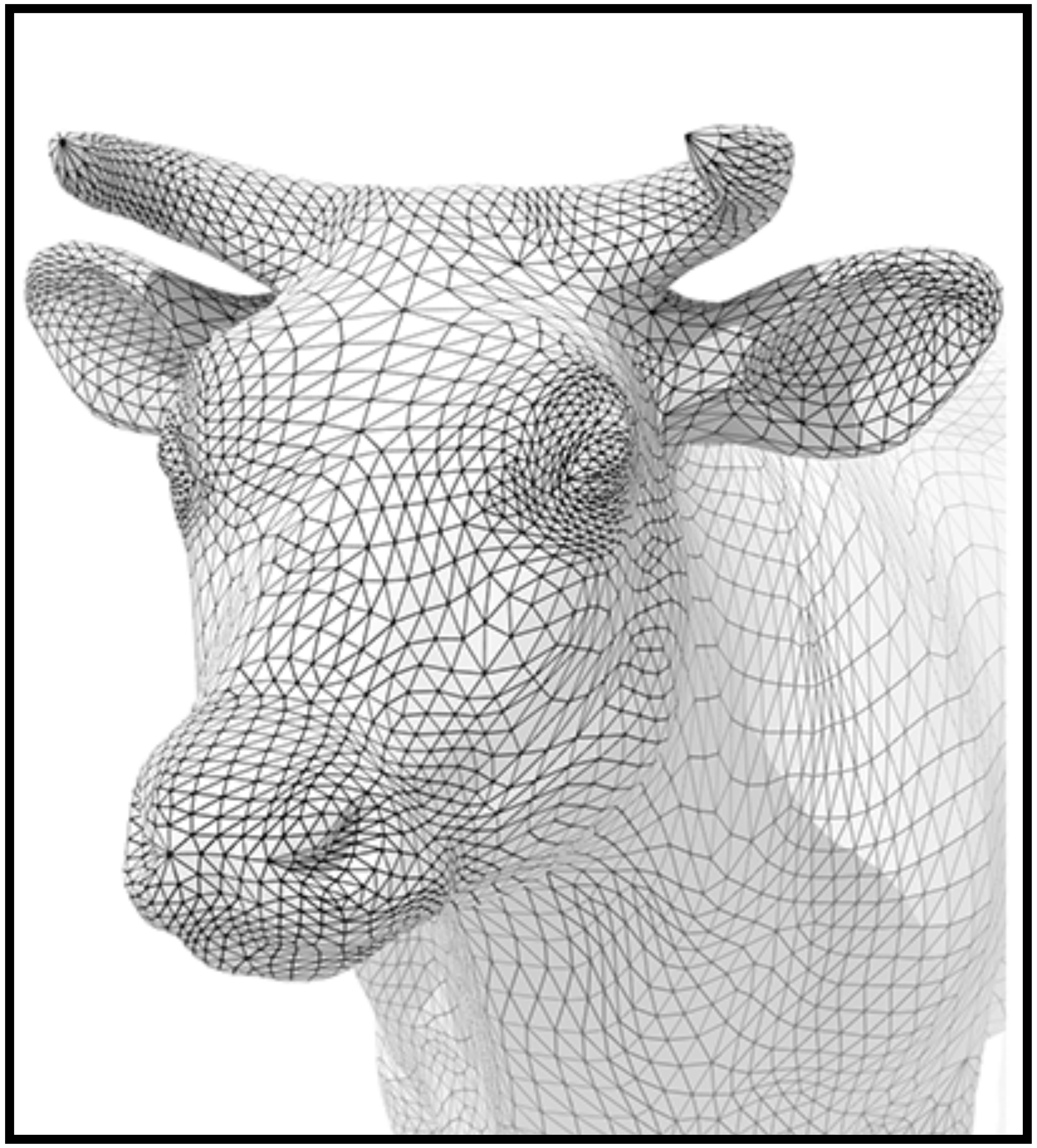
Mesh Upsampling – Subdivision

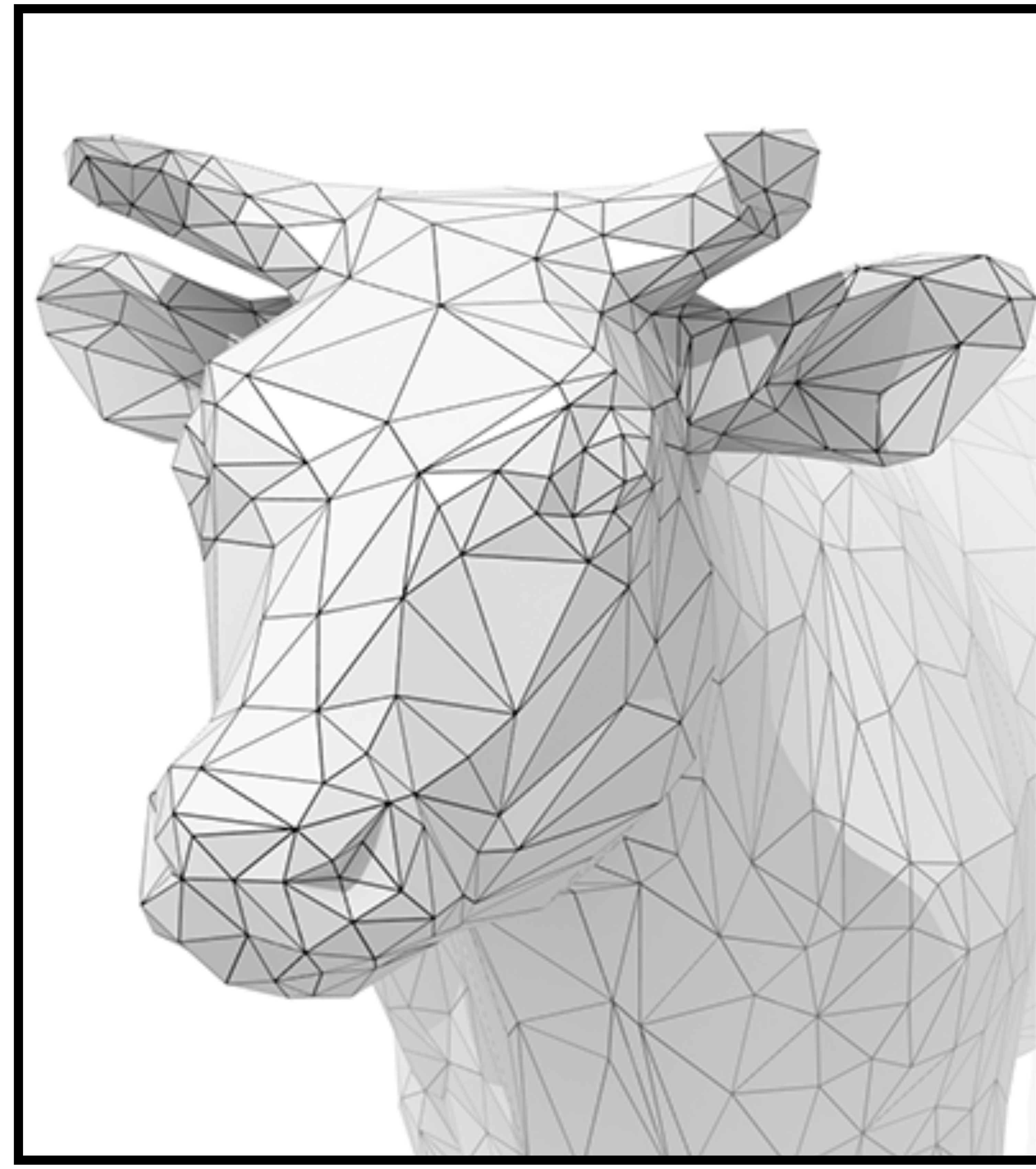




Increase resolution via interpolation

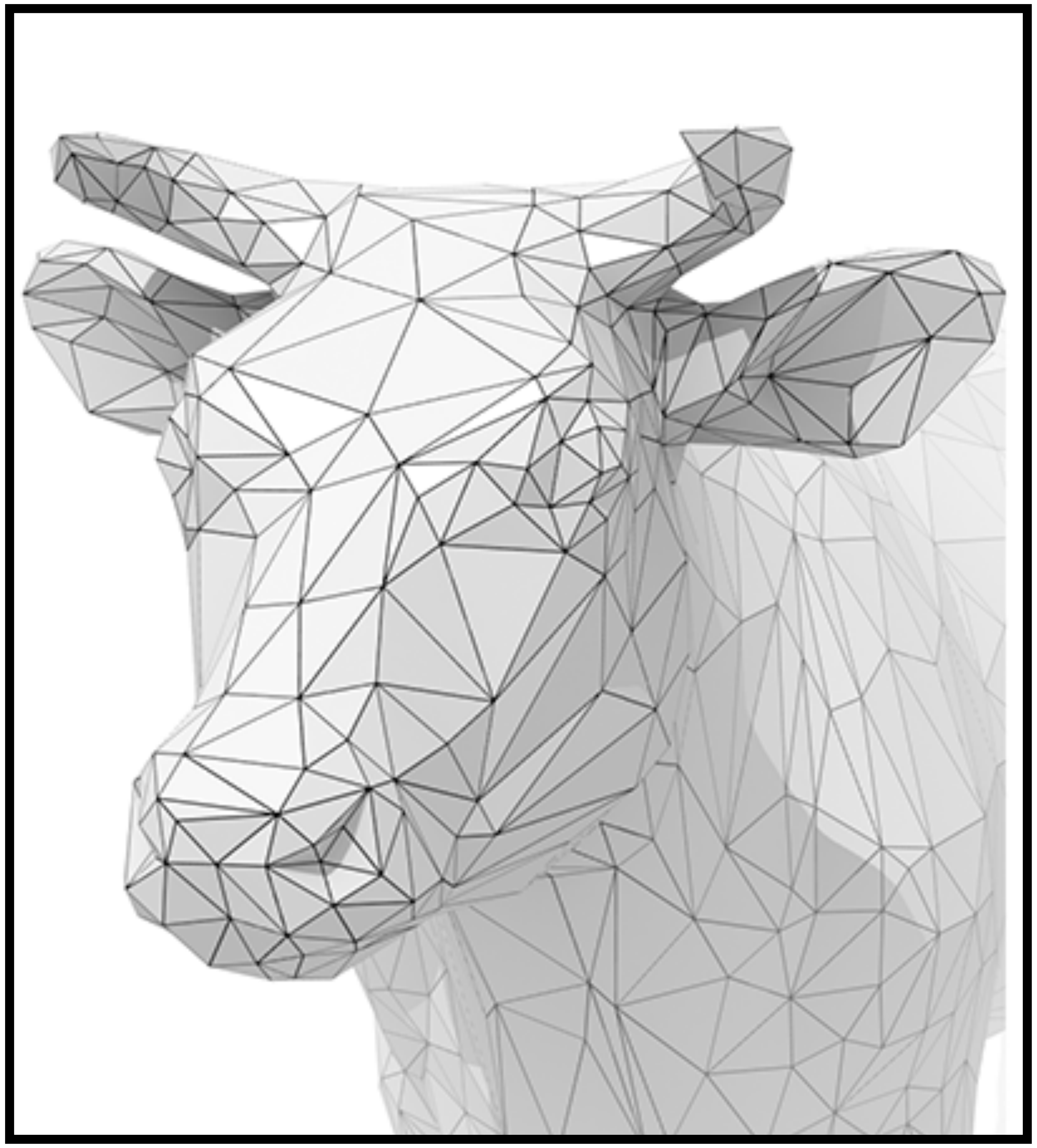
Mesh Downsampling – Simplification

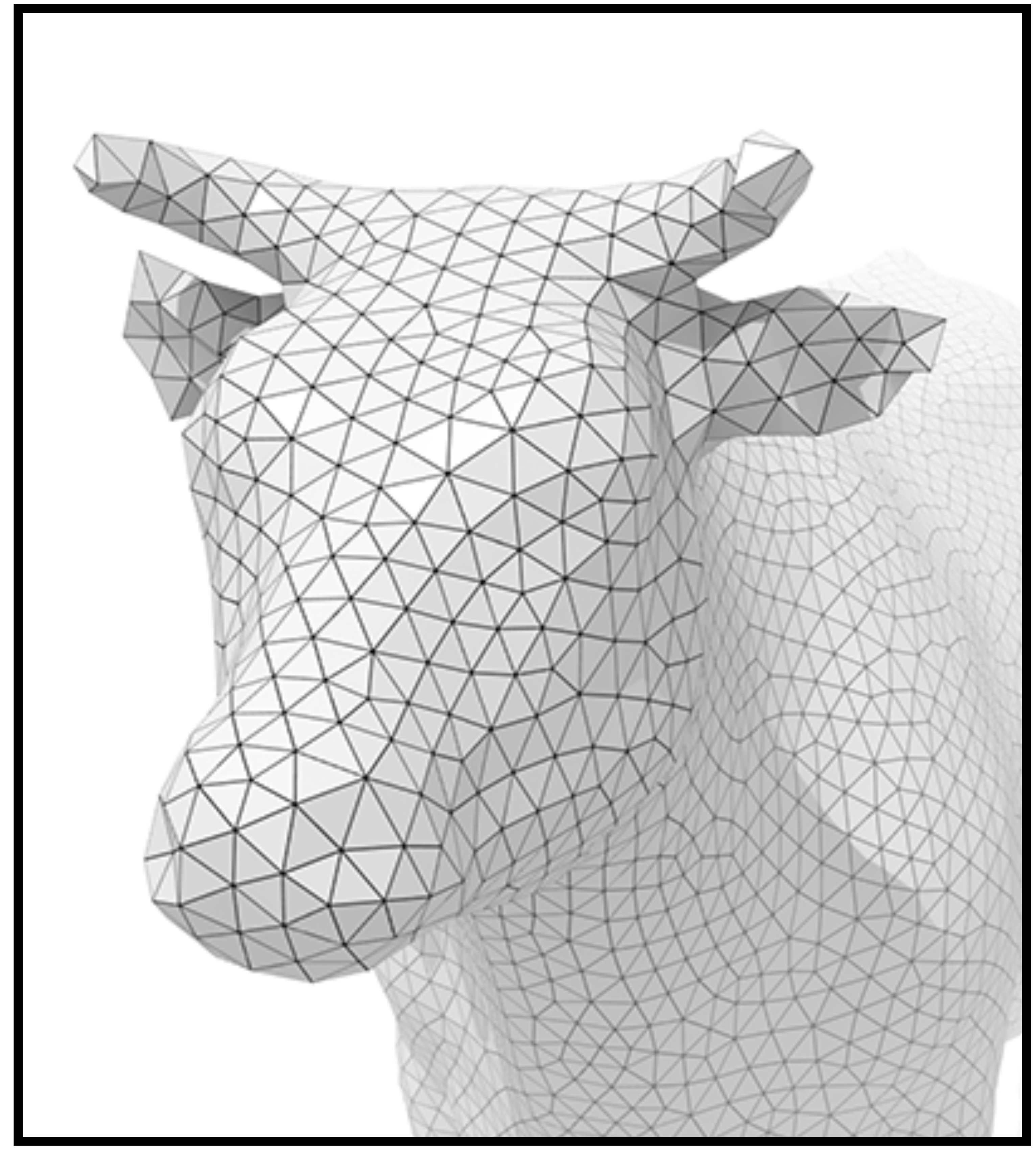




Decrease resolution; try to preserve shape/appearance

Mesh Regularization

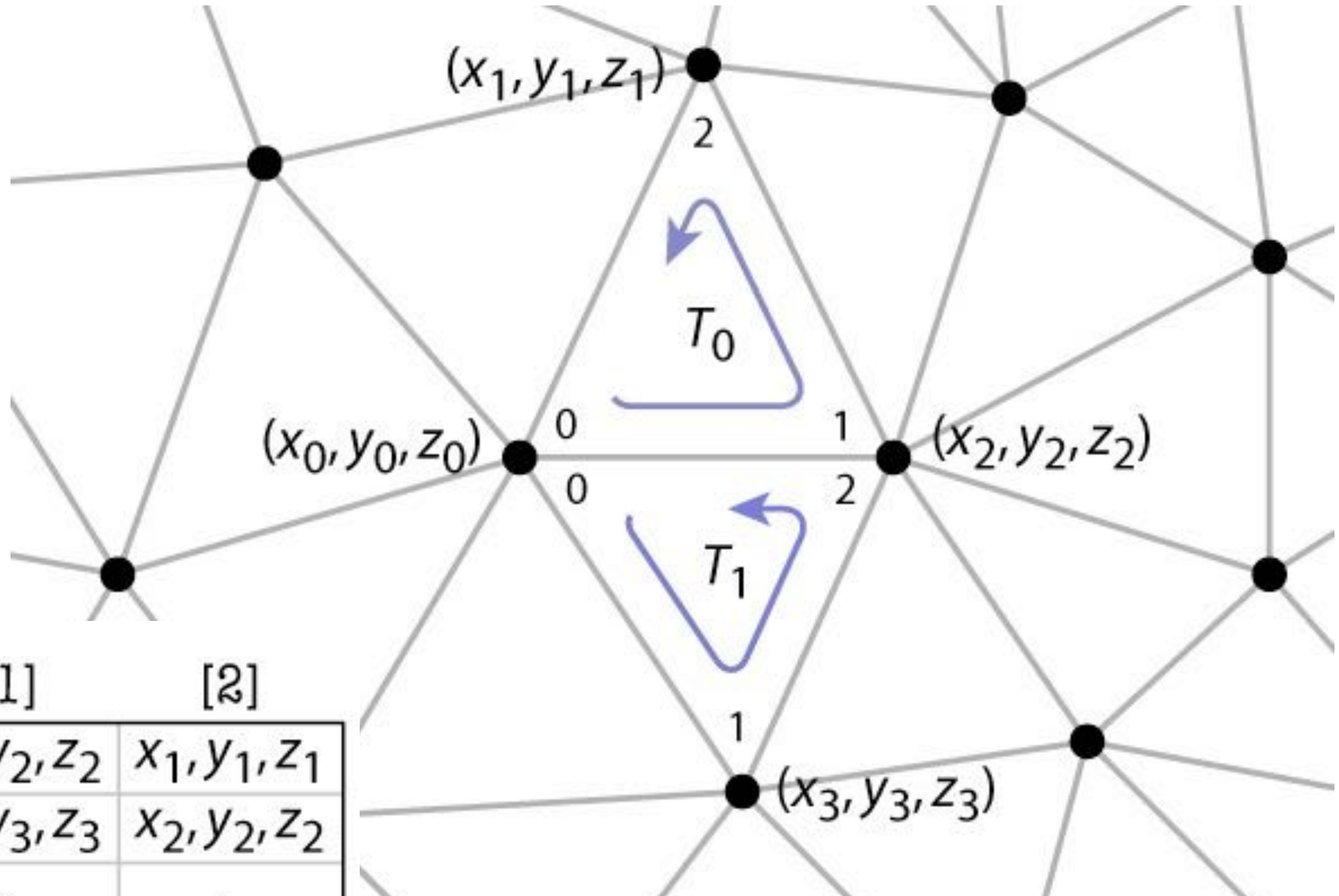




Modify sample distribution to improve quality

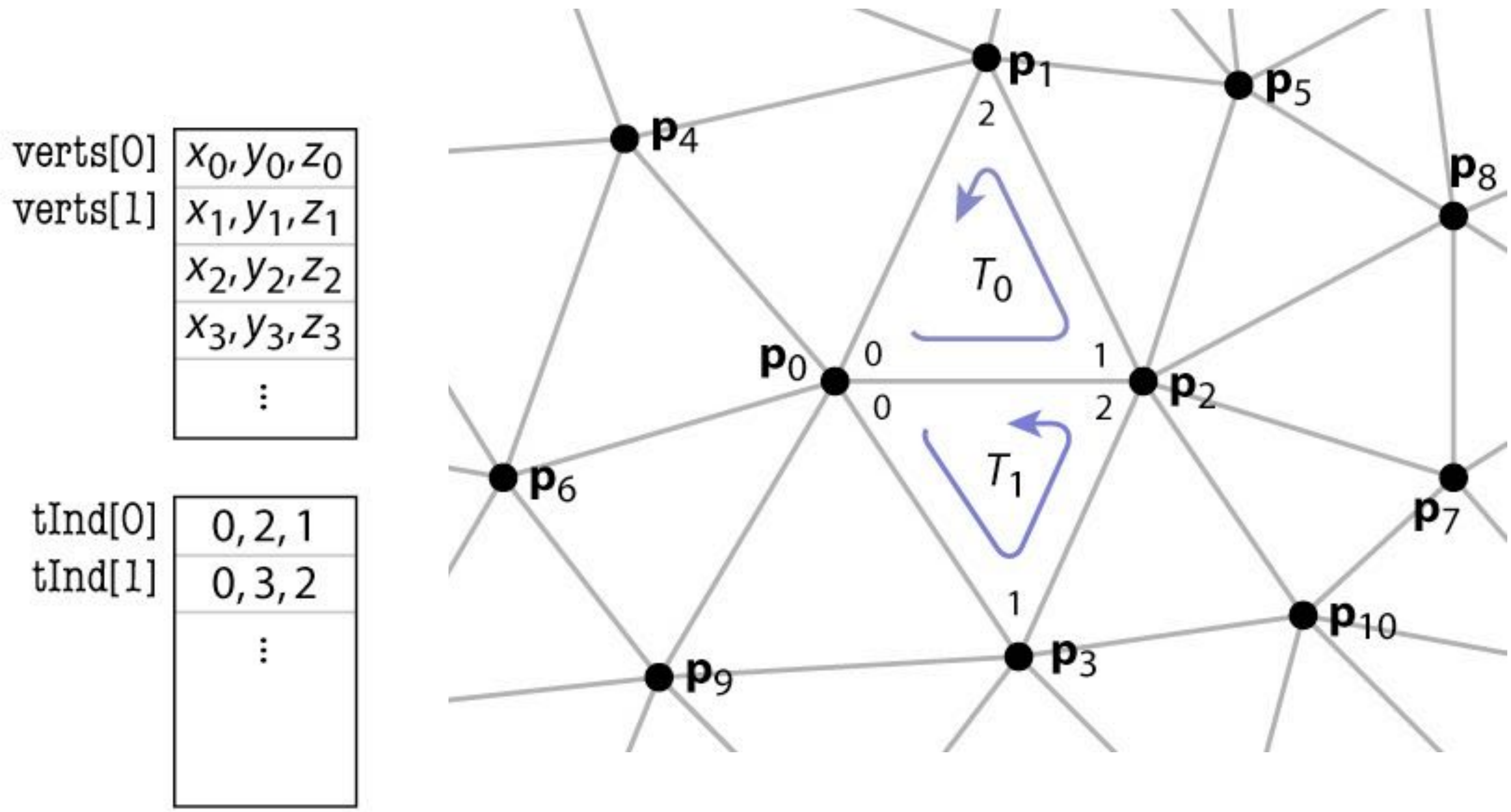
Mesh Representations

List of Triangles



	[0]	[1]	[2]
tris[0]	x_0, y_0, z_0	x_2, y_2, z_2	x_1, y_1, z_1
tris[1]	x_0, y_0, z_0	x_3, y_3, z_3	x_2, y_2, z_2
	⋮	⋮	⋮

Lists of Points / Indexed Triangle

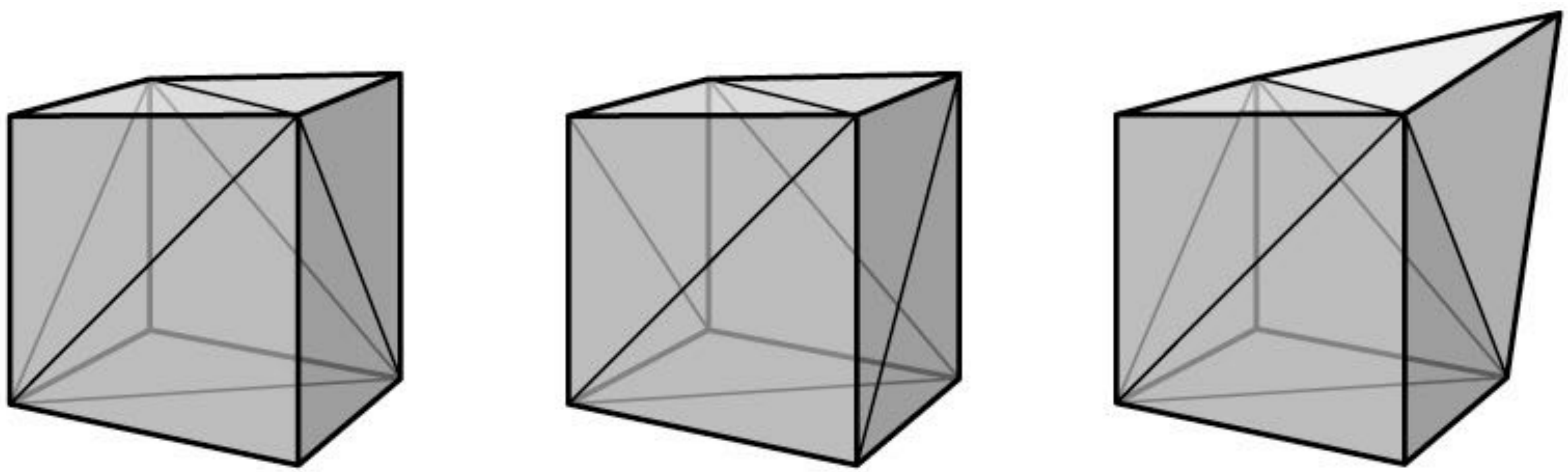


How much data storage?

Topology vs Geometry

Which one has different topology from the first?

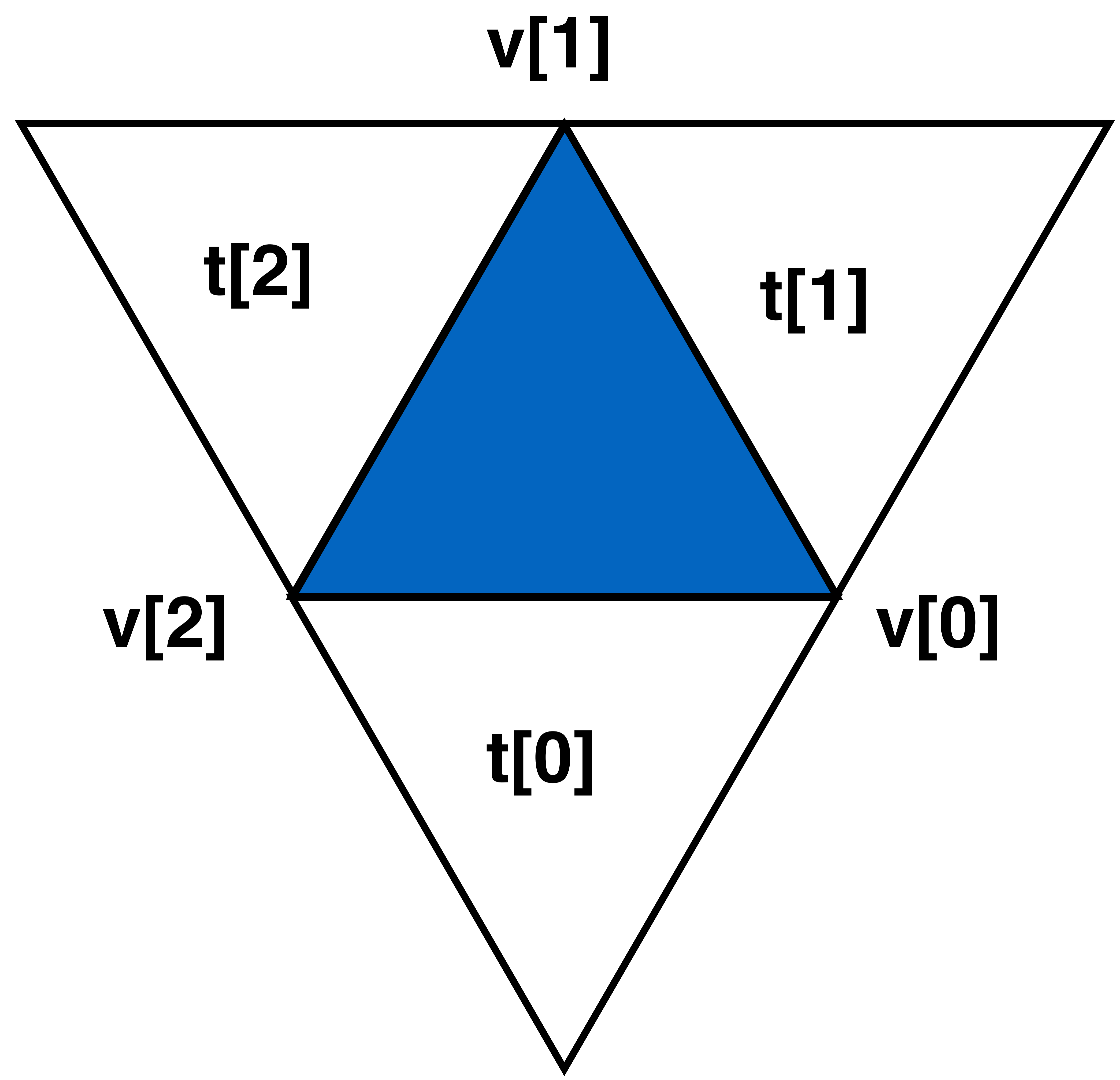
Different geometry?



Triangle-Neighbor Data Structure

```
struct Tri {
    Vert     * v[3];
    Tri   * t[3];
}

struct Vert {
    Point    pt;
    Tri   *t;
}
```



Comparison

Triangles?

+ Simple

– Redundant information (In what way?)

Points + Triangles?

+ Sharing vertices reduces memory usage

+ Ensure integrity of the mesh (how so?)

Topological Data Structures?

+ Access to neighbors (how?)

– More complex

Topological Validity: Manifold

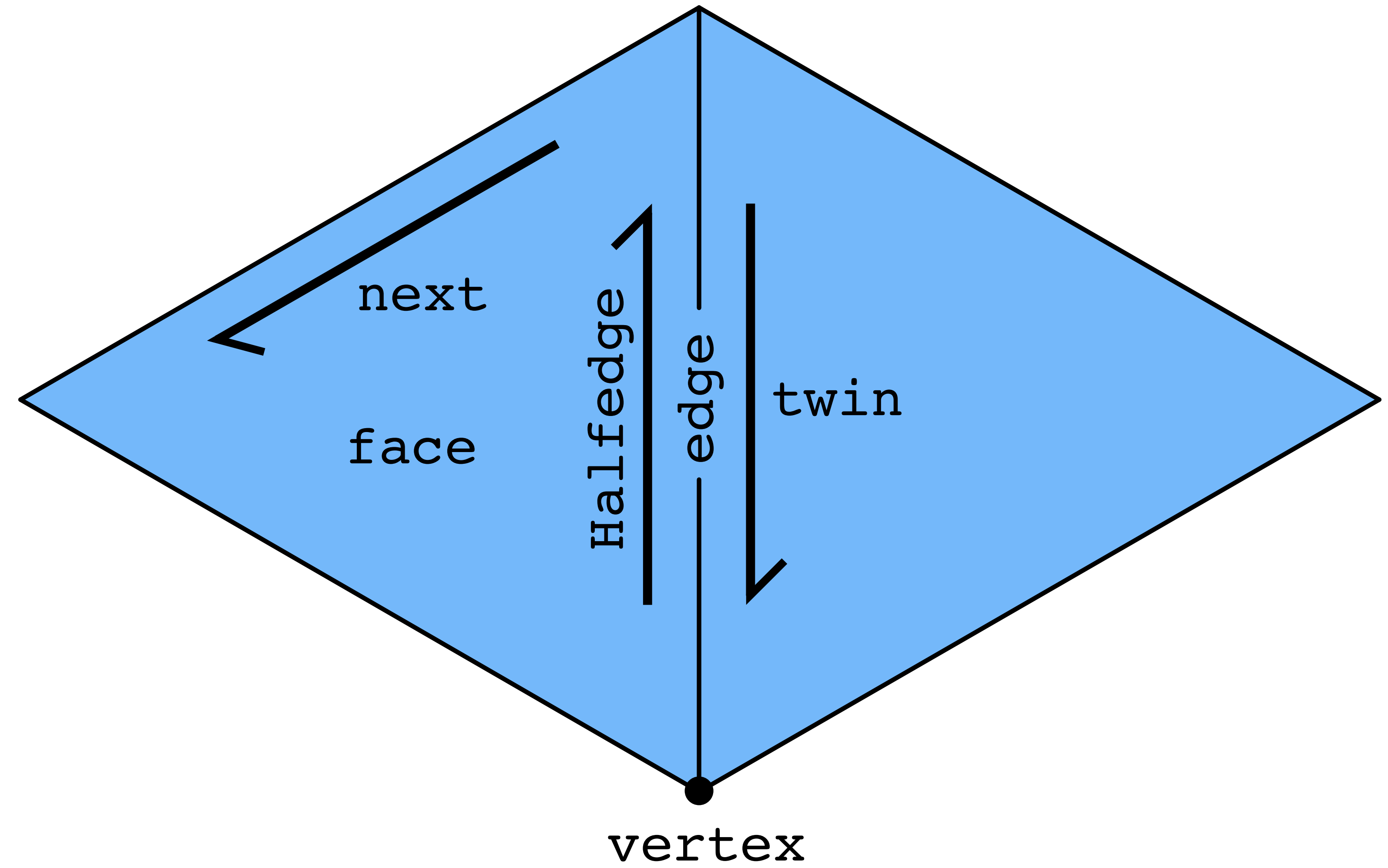
Definition: a 2D manifold is a surface that when cut with a small sphere always yields a disk.

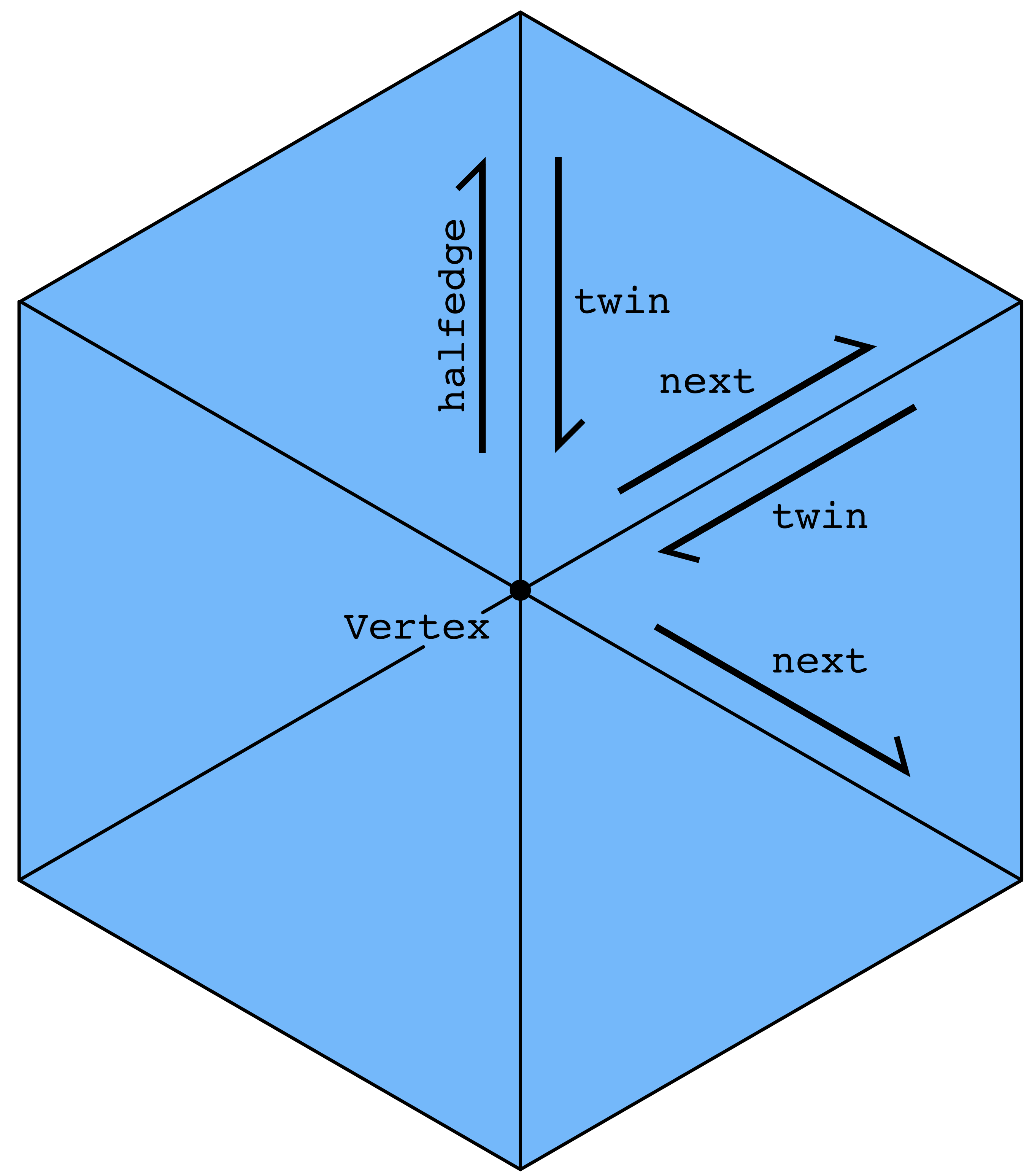
If a mesh is manifold* we can rely on these useful properties:

- An edge connects exactly two faces
- An edge connects exactly two vertices
- A face consists of a ring of edges and vertices
- A vertex consists of a ring of edges and faces
- Euler's polyhedron formula holds: #f – #e + #v = 2
  (for a surface topologically equivalent to a sphere)
  (Check for a cube: 6 – 12 + 8 = 2)

* (without boundary)

- An edge connects exactly two faces
- An edge connects exactly two vertices
- A face consists of a ring of edges and vertices
- A vertex consists of a ring of edges and faces

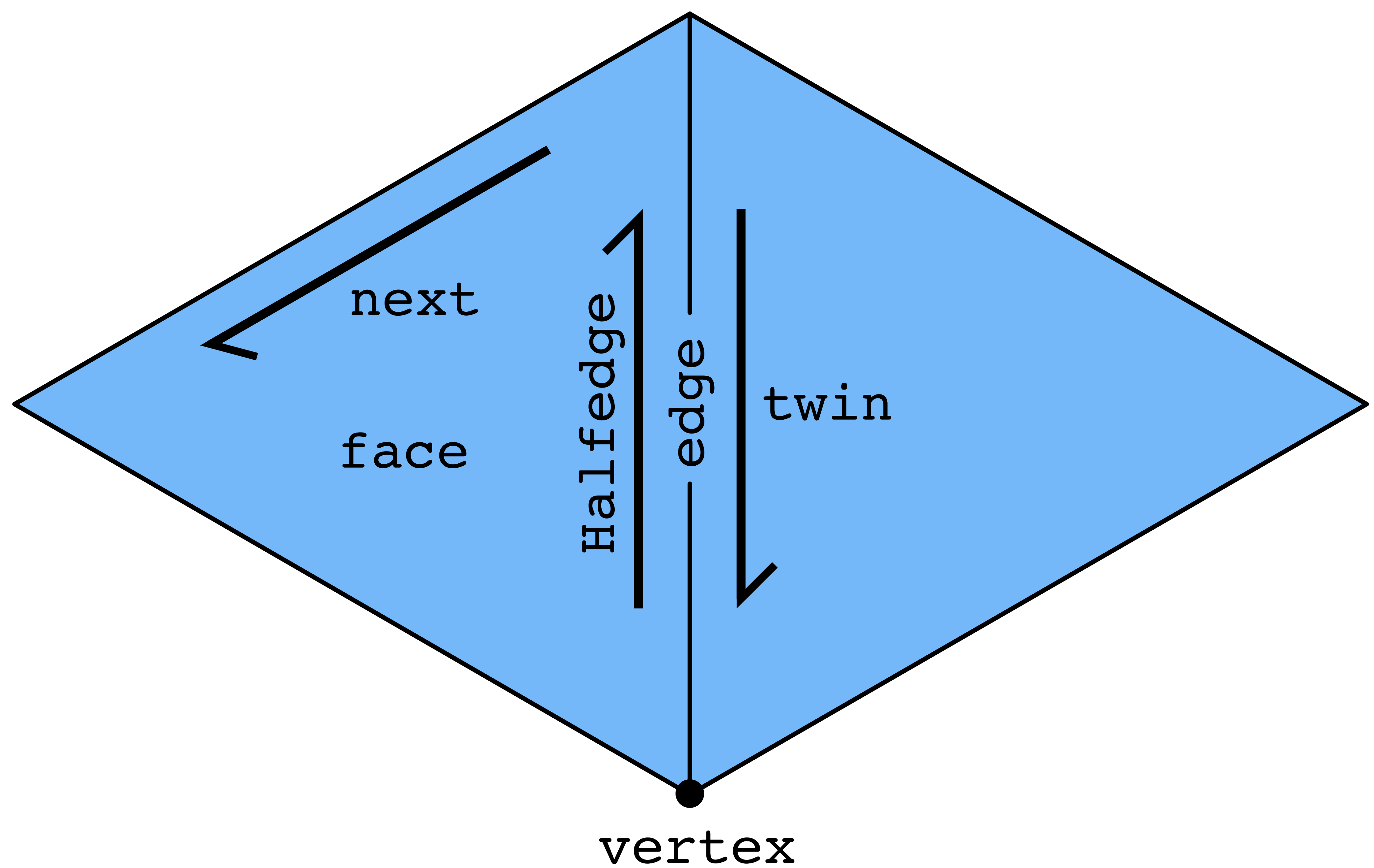




Half-Edge Data Structure

```
struct Halfedge {
   Halfedge *twin,
   Halfedge *next;
   Vertex *vertex;
   Edge *edge;
   Face *face;
}
struct Vertex {
   Point pt;
   Halfedge *halfedge;
}
struct Edge {
   Halfedge *halfedge;
}
struct Face {
   Halfedge *halfedge;
}
```

Key idea: two half-edges act as "glue" between mesh elements



Each vertex, edge and face points to one of its half edges

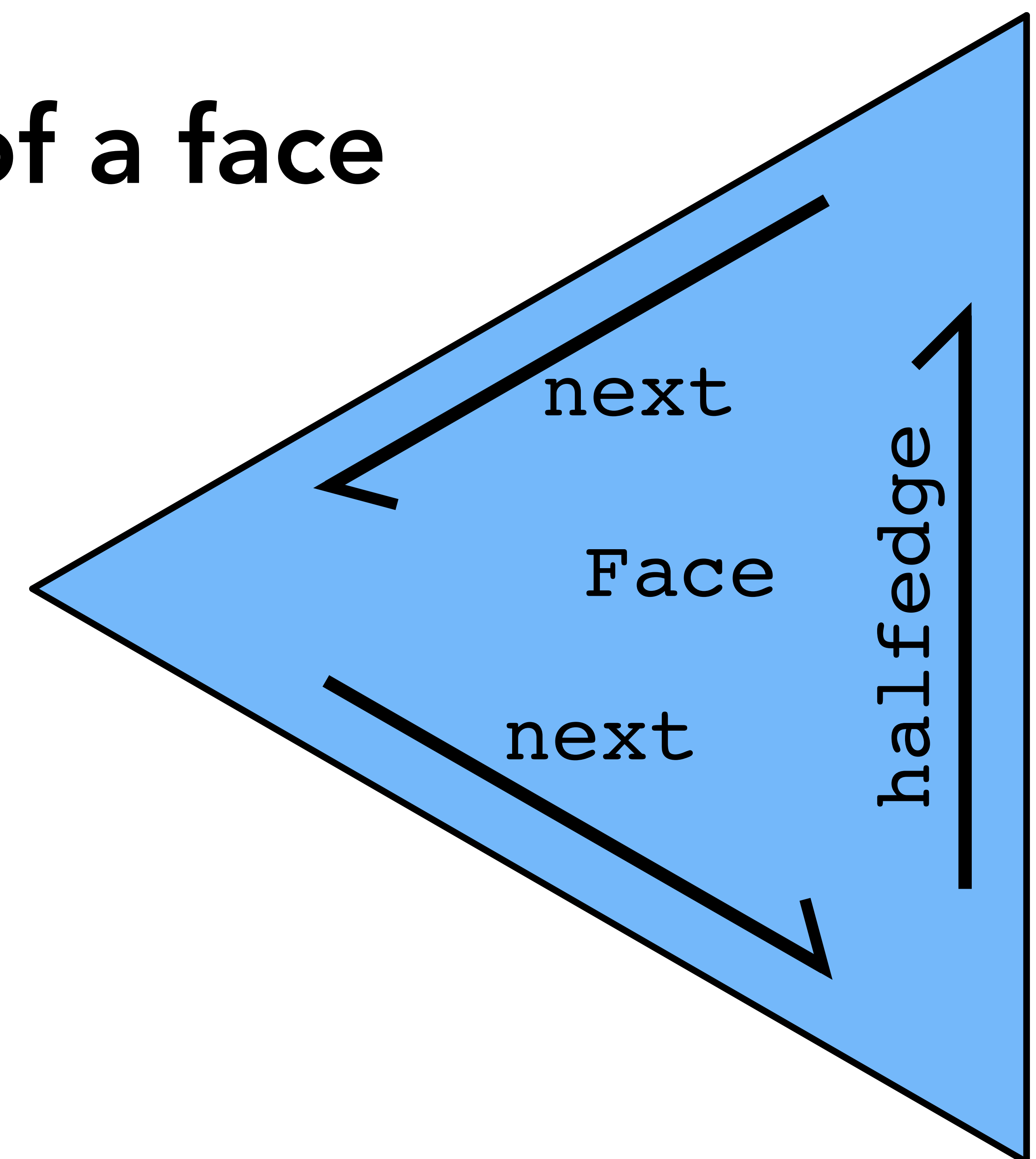
Half-Edge Facilitates Mesh Traversal

Use twin and next pointers to move around mesh

Process vertex, edge and/or face pointers

Example 1: process all vertices of a face

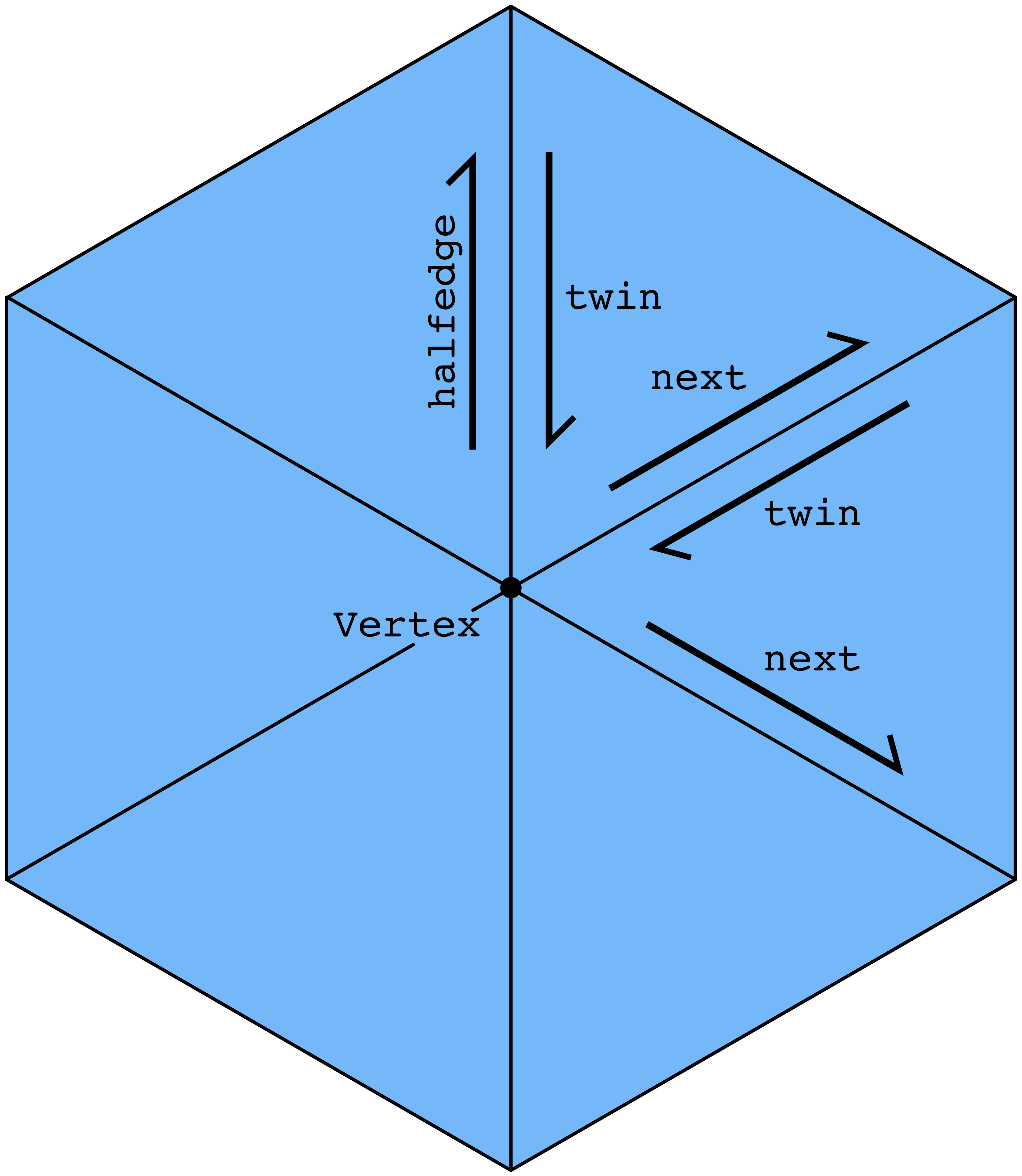
```
Halfedge* h = f->halfedge;
do {
  process(h->vertex);
  h = h->next;
}
```



Half-Edge Facilitates Mesh Traversal

Example 2: process all edges around a vertex

```
Halfedge* h = v->halfedge;
do {
  process(h->edge);
  h = h->twin->next;
}
```

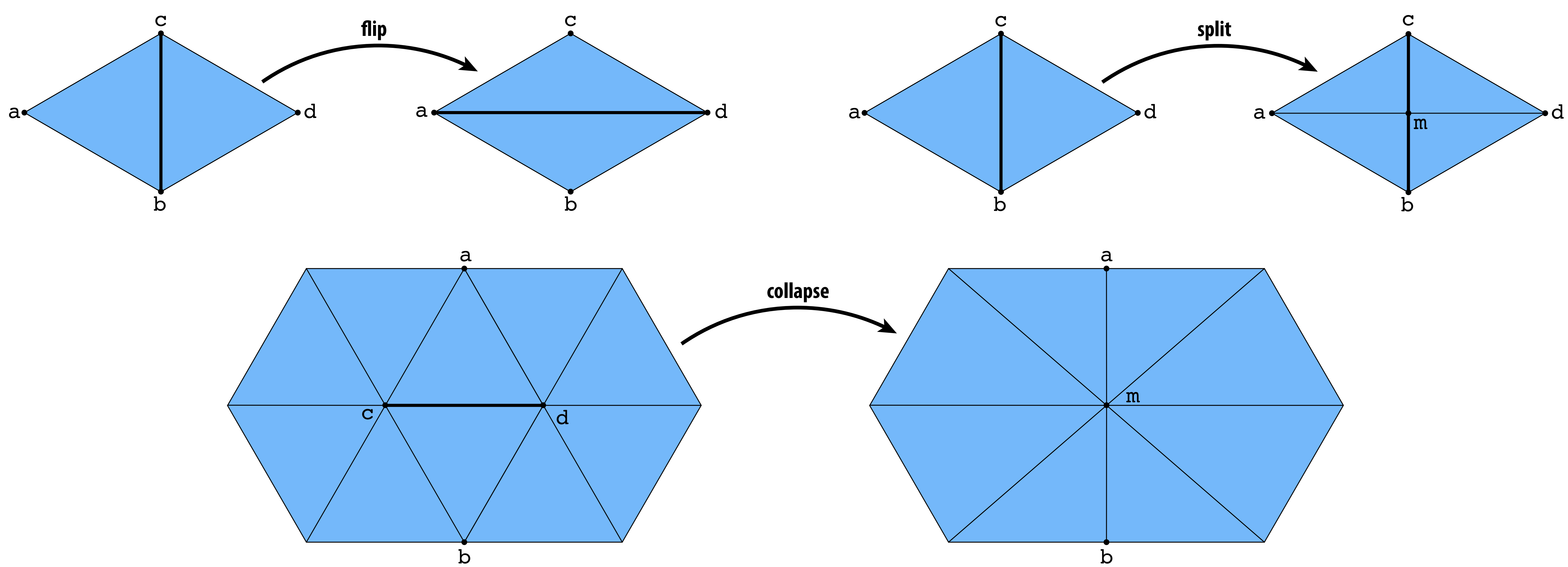


Local Mesh Operations

Half-Edge – Local Mesh Editing

Basic operations for linked list: insert, delete

Basic ops for half-edge mesh: flip, split, collapse edges

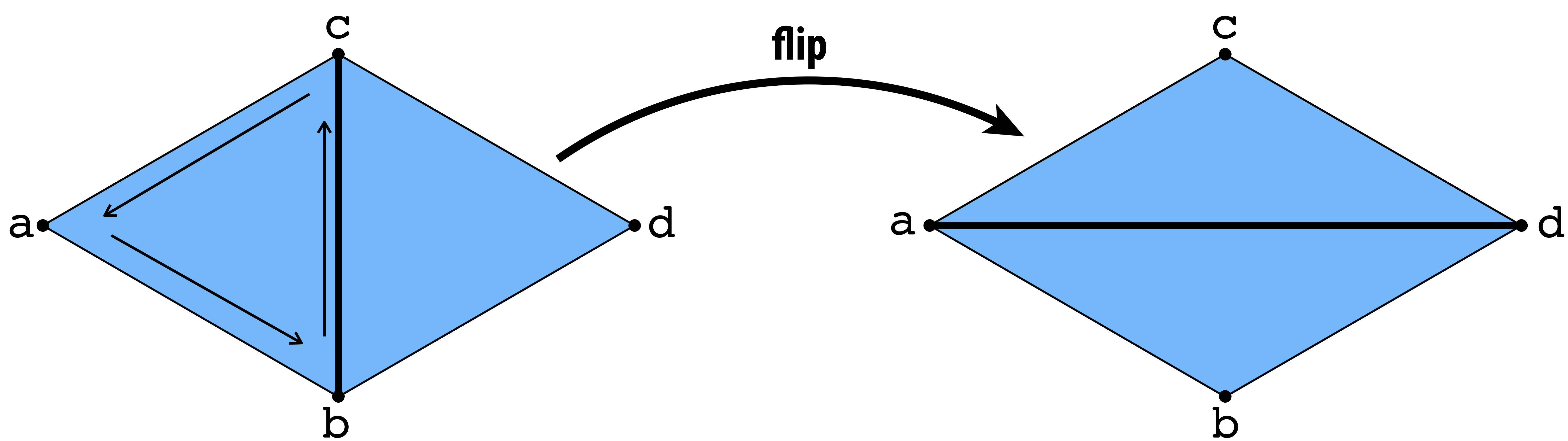


Allocate / delete elements; reassign pointers

(Care needed to preserve mesh manifold property)

Half-Edge – Edge Flip

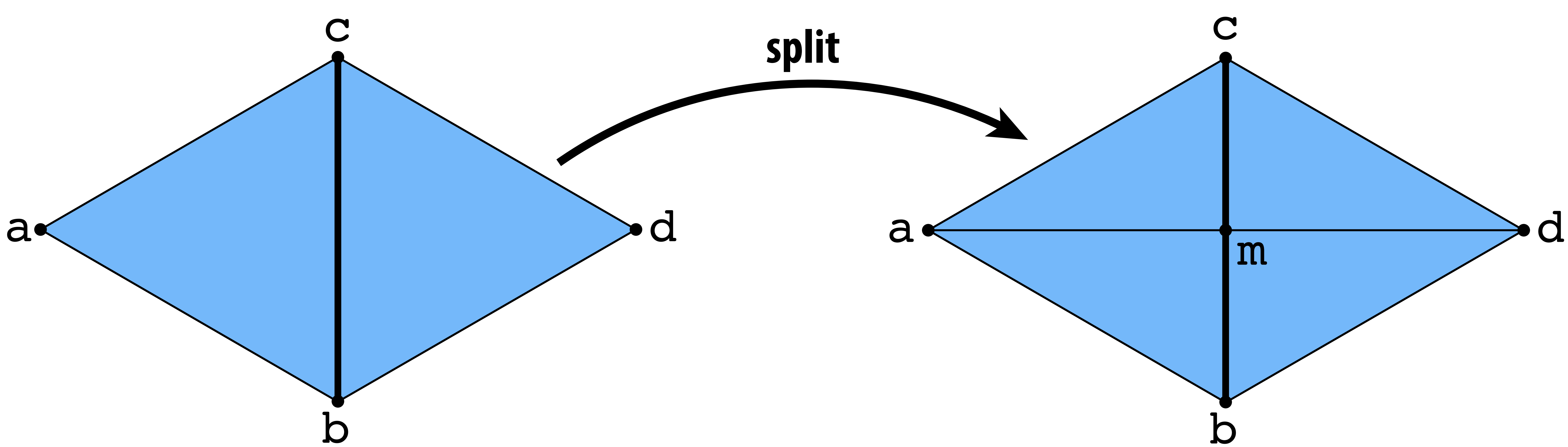
- Triangles (a,b,c), (b,d,c) become (a,d,c), (a,b,d):



- Long list of pointer reassignments
- However, no elements created/destroyed.

Half-Edge – Edge Split

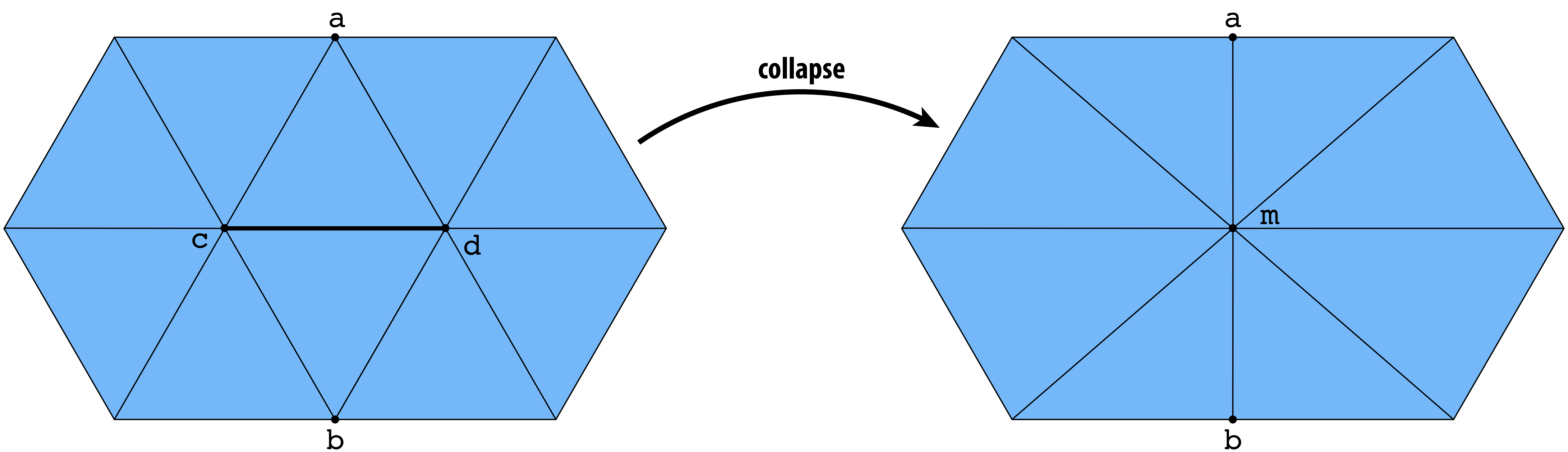
- Insert midpoint m of edge (c,b), connect to get four triangles:



- This time have to add elements
- Again, many pointer reassignments

Half-Edge – Edge Collapse

- Replace edge (c,d) with a single vertex m:

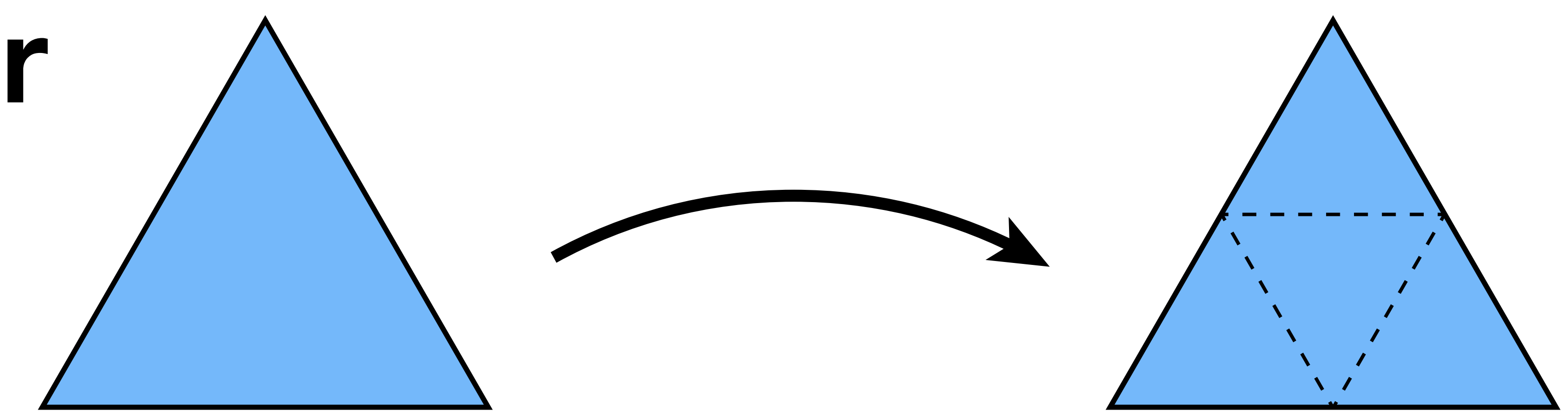


- This time have to delete elements
- Again, many pointer reassignments

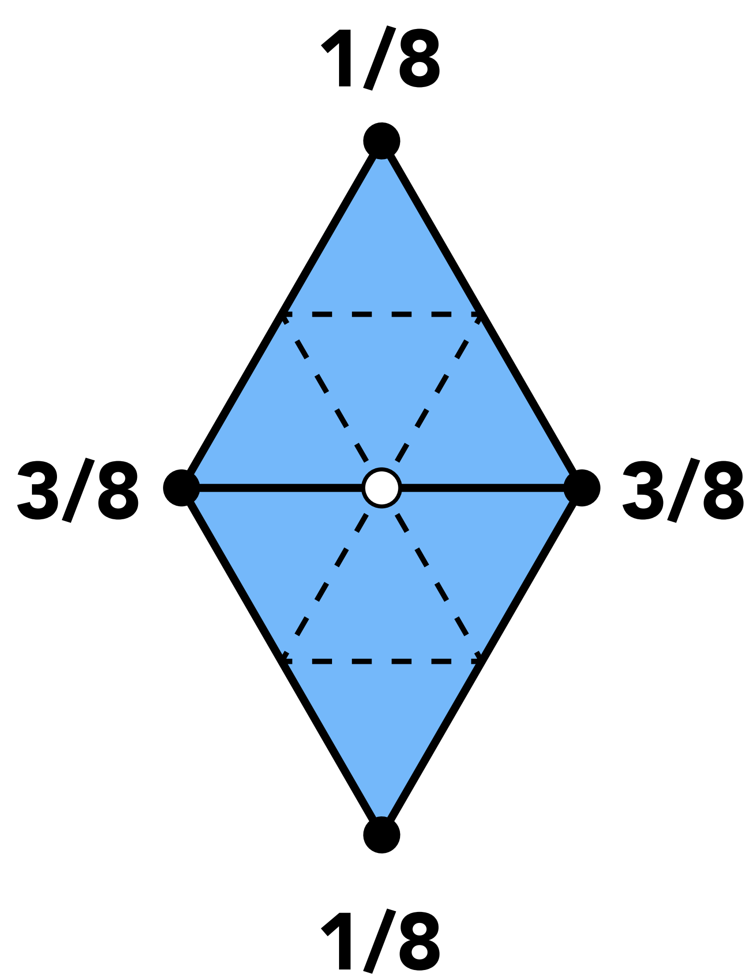
Loop Subdivision

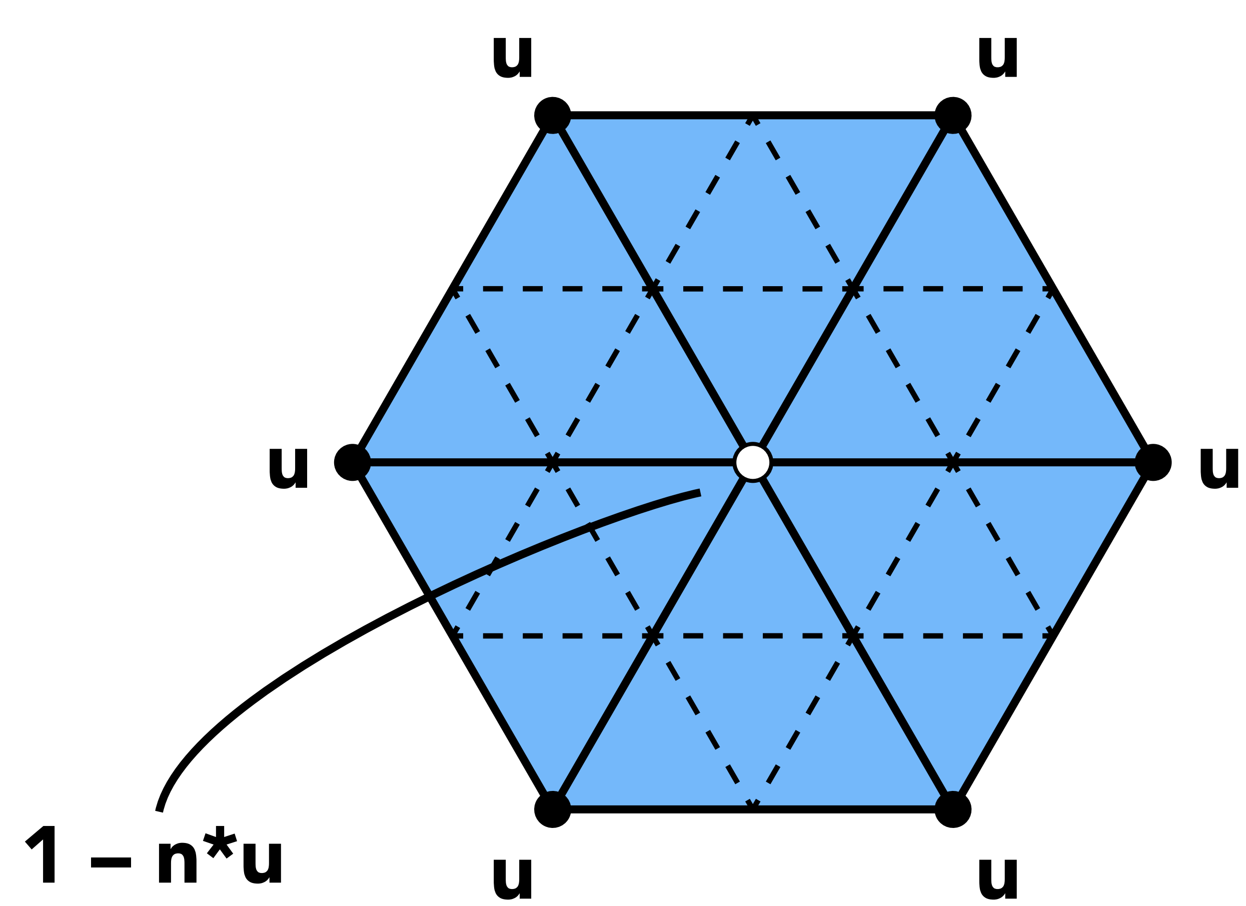
Loop Subdivision Algorithm

- Split each triangle into four



- Assign new vertex positions according to weights:





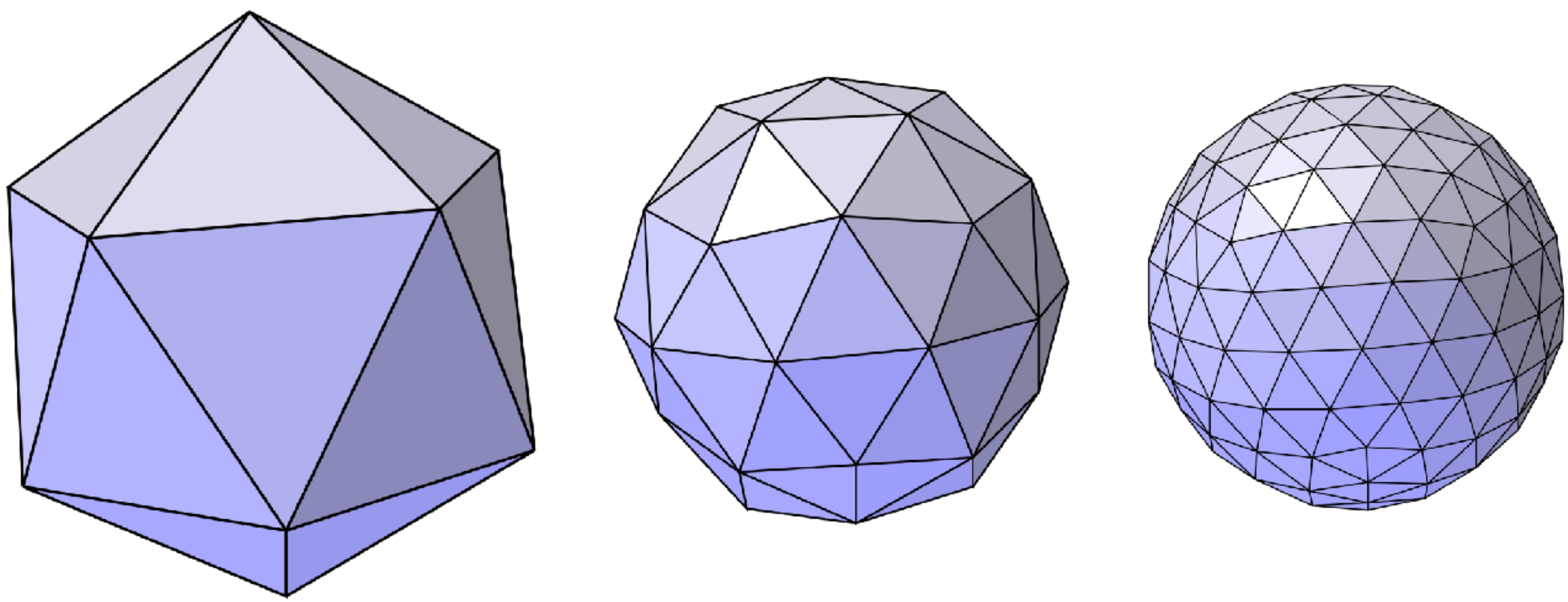
n: vertex degree

u: 3/16 if n=3, 3/(8n) otherwise

New vertices

Old vertices

Loop Subdivision Algorithm



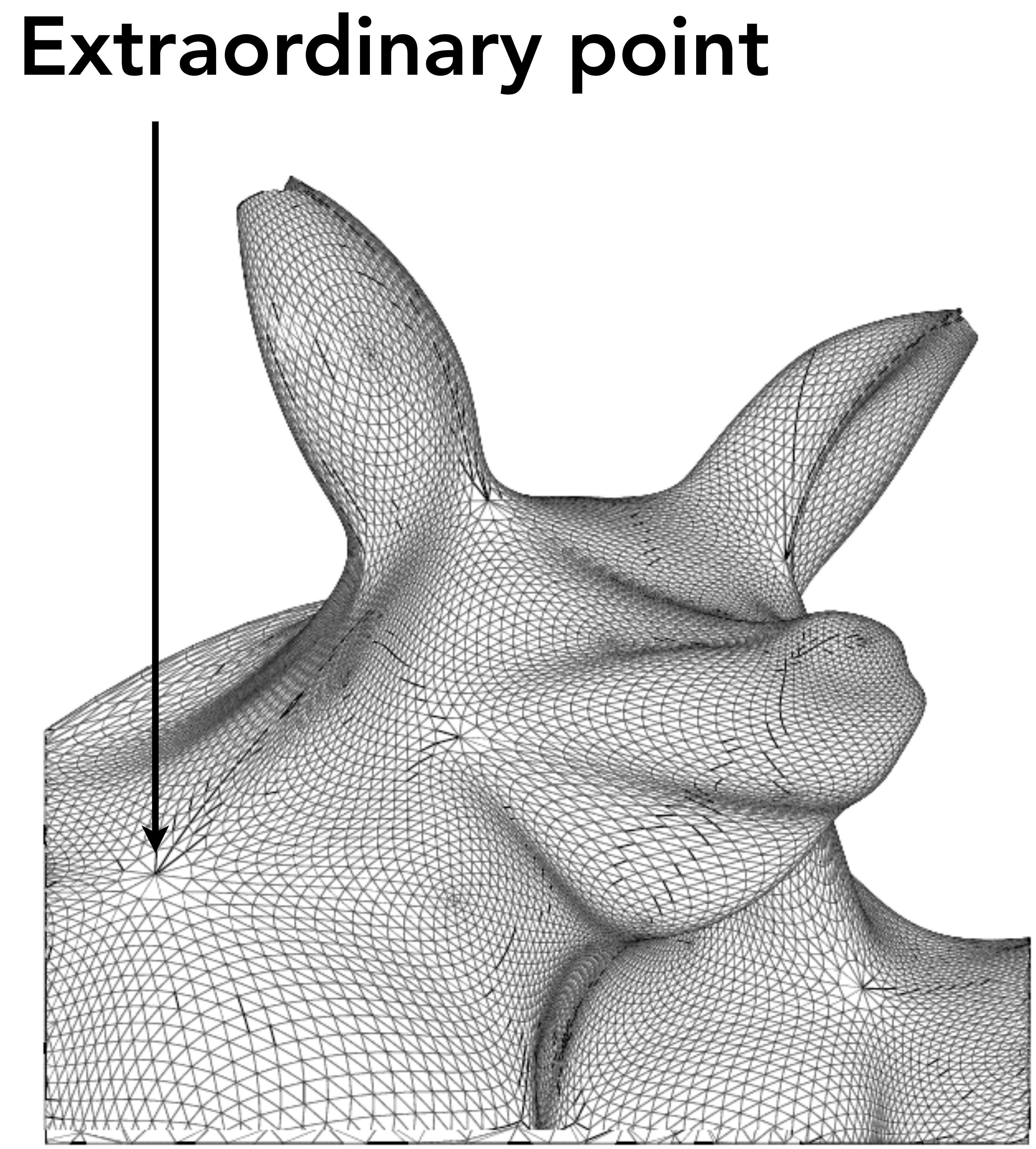
Simon Fuhrman

Semi-Regular Meshes

Most of the mesh has vertices with degree 6

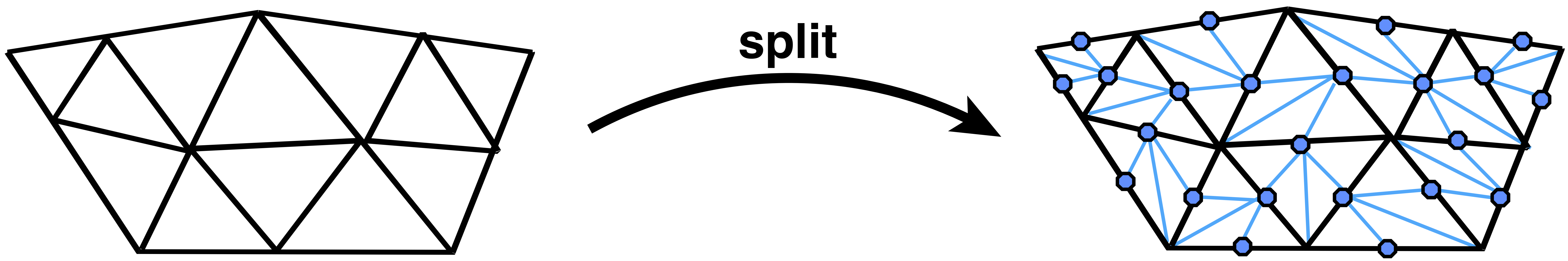
But if the mesh is topologically equivalent to a sphere, then not all the vertices can have degree 6

Must have a few extraordinary points (degree not equal to 6)

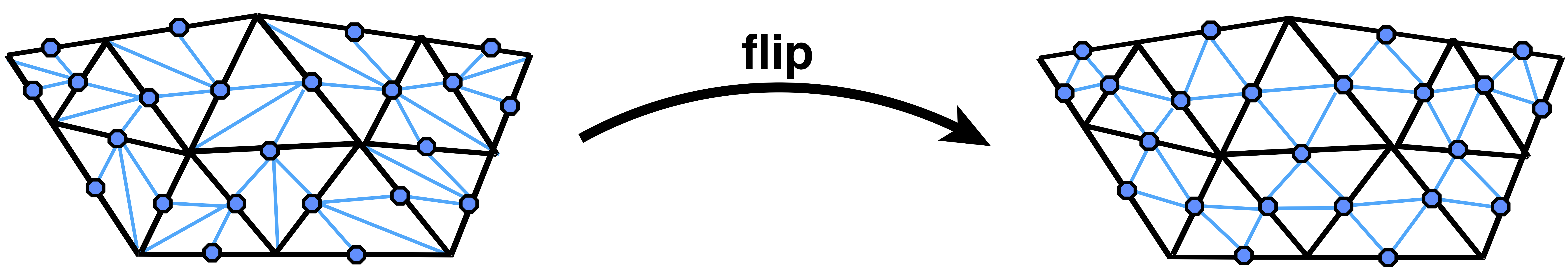


Loop Subdivision via Edge Operations

First, split edges of original mesh in any order:



Next, flip new edges that touch a new & old vertex:



(Don't forget to update vertex positions!)

Images cribbed from Keenan Crane, cribbed from Denis Zorin

What About Sharp Creases?

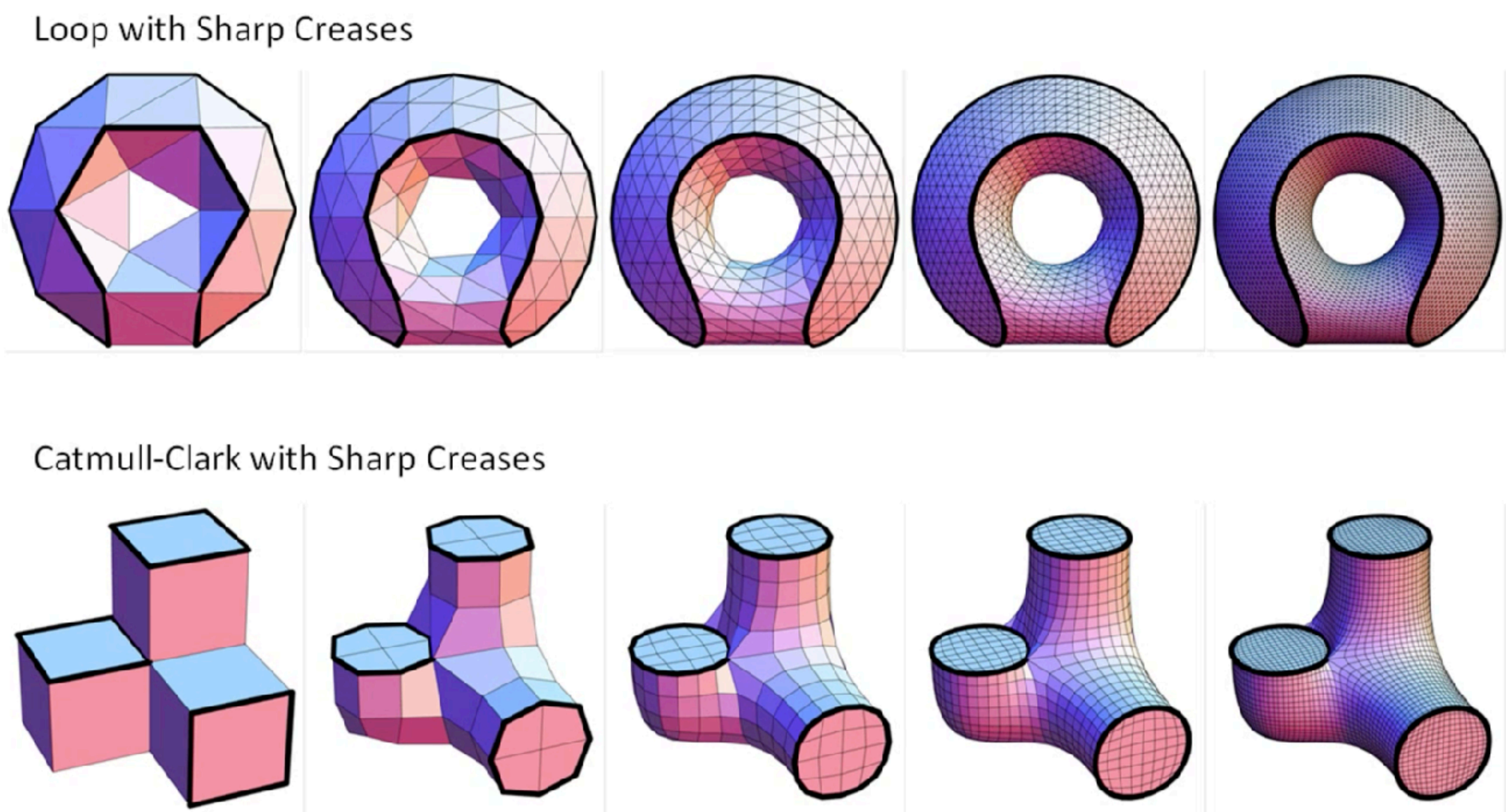
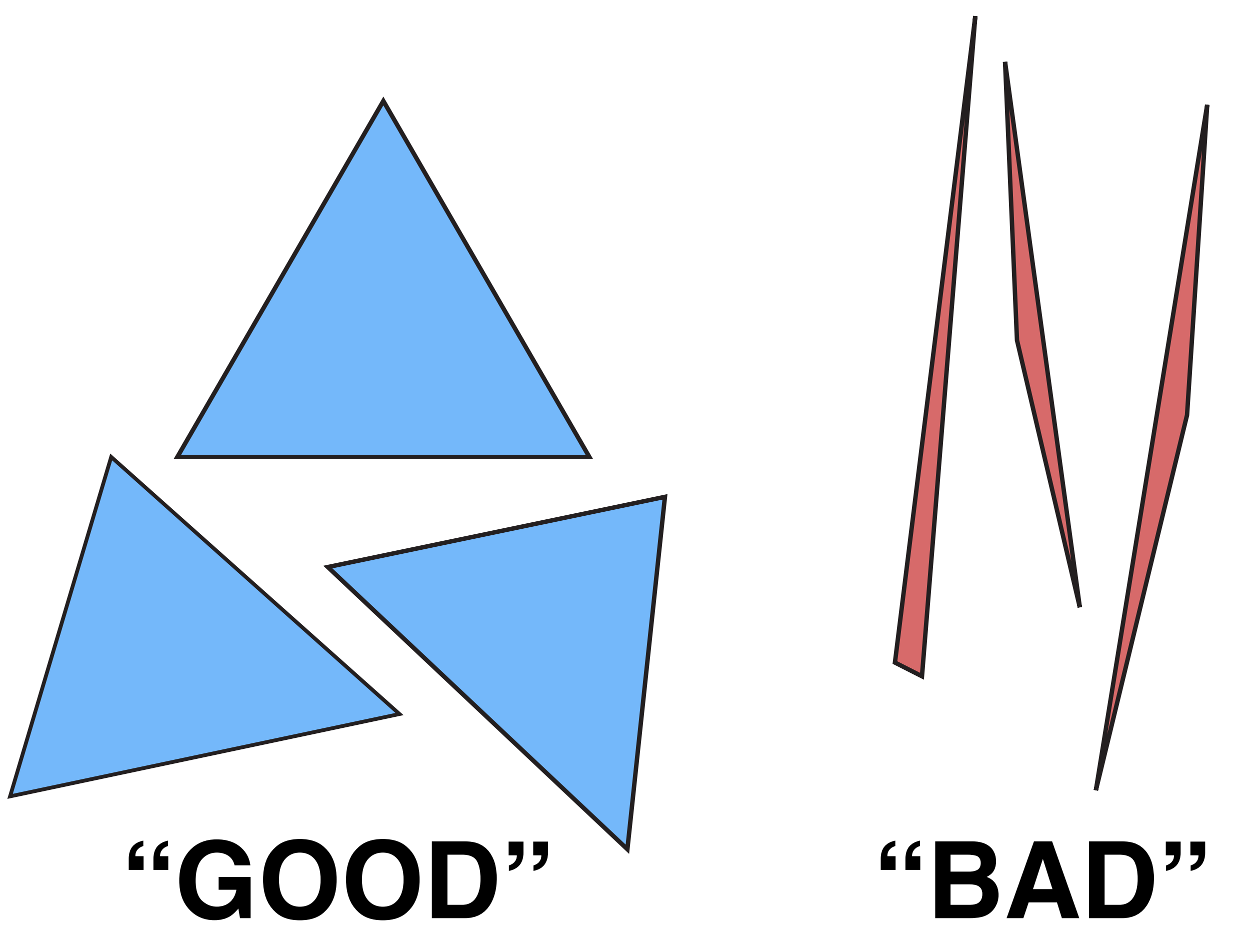


Figure from: Hakenberg et al. Volume Enclosed by Subdivision Surfaces with Sharp Creases

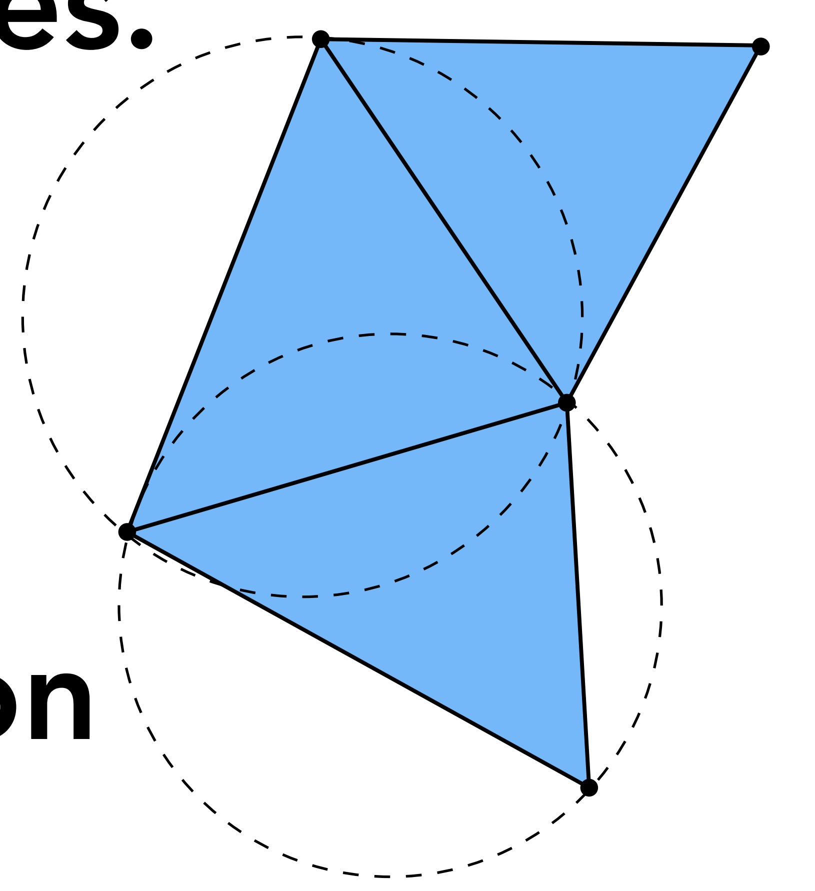
What Makes a "Good" Triangle Mesh?

One rule of thumb: triangle shape



More specific condition: Delaunay

- "Circumcircle interiors contain no vertices."



Not always a good condition, but often*

- Good for simulation
- Not always best for shape approximation

*See Shewchuk, "What is a Good Linear Element"

Acknowledgments

Thanks to Keenan Crane, Pat Hanrahan, and James O'Brien for presentation resources.

Many thanks to Ren Ng for use of lecture slides.