polygon meshes

polygon mesh representation

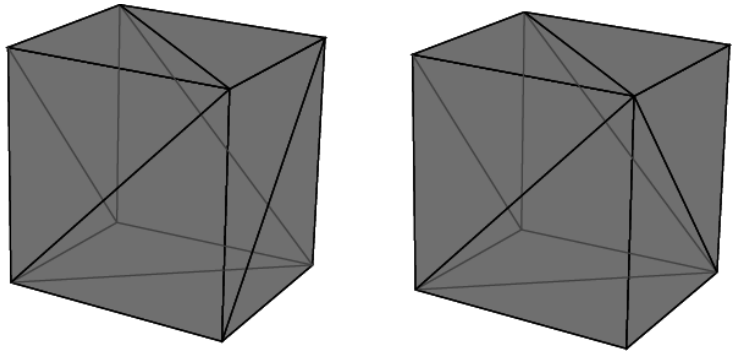
- which representation is good?
  - often triangles/quads only – will work on triangles
- compact
- efficient for rendering
  - fast enumeration of all faces
- efficient for geometry algorithms
  - finding adjacency (what is close to what)

vertices, edges, faces

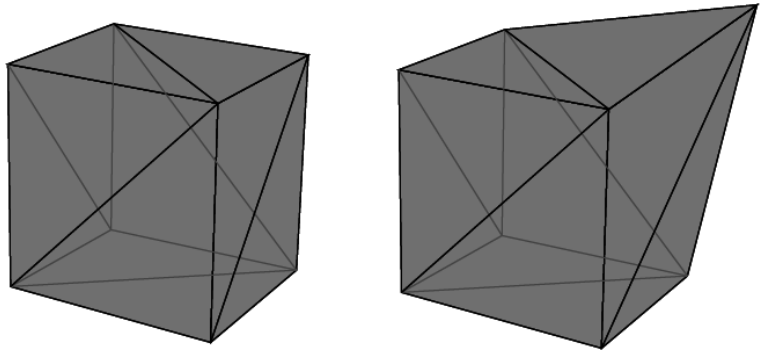
- fundamental entities
  - n_v vertices
  - n_e edges
  - n_f faces
  - simple closed surface: $n_v - n_e + n_f = 2$
- fundamental properties:
  - topology: how faces are connected
  - geometry: where faces are in space
  - algorithms mostly care about topology

topology vs. geometry

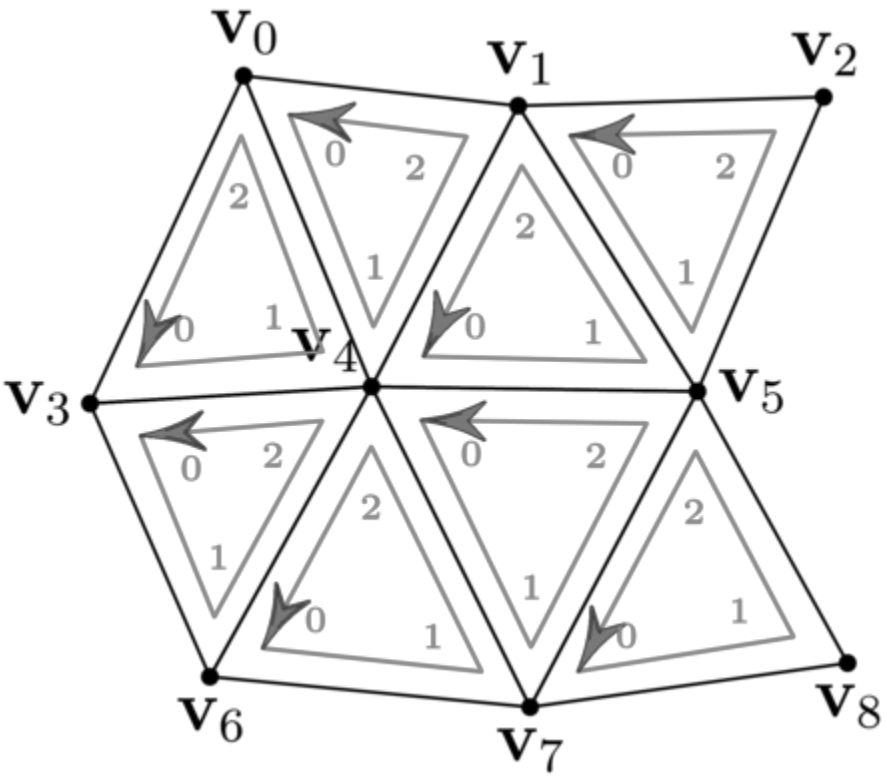
- same geometry, different topology



- same topology, different geometry



triangles

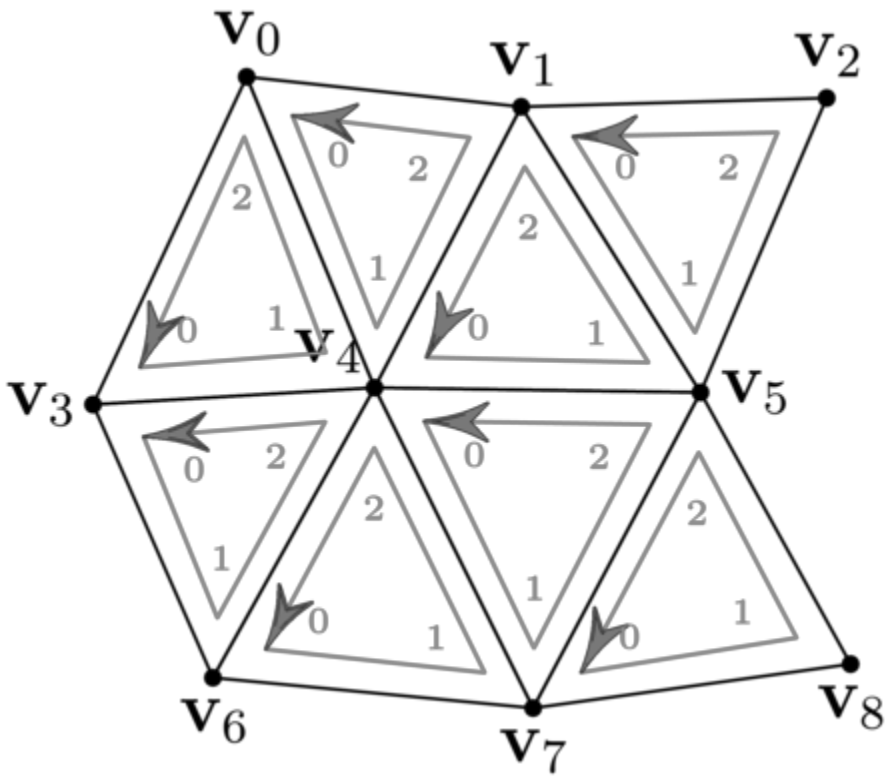


triangle[0]	(x0, y0, z0)	(x1, y1, z1)	(x2, y2, z2)
triangle[1]	(x1, y1, z1)	(x2, y2, z2)	(x3, y3, z3)
...	...	...	...

triangles

- array of vertex data
  - data type: vertex[nf][3]
  - vertex stores position and optional data (normal, uvs)
  - ~72 bytes per triangle with vertex position only
- redundant
- adjacency is not well defined
  - floating point errors in comparing vertices

indexed triangles



triangle[0]	0,1,2
triangle[1]	2,1,3
...	...

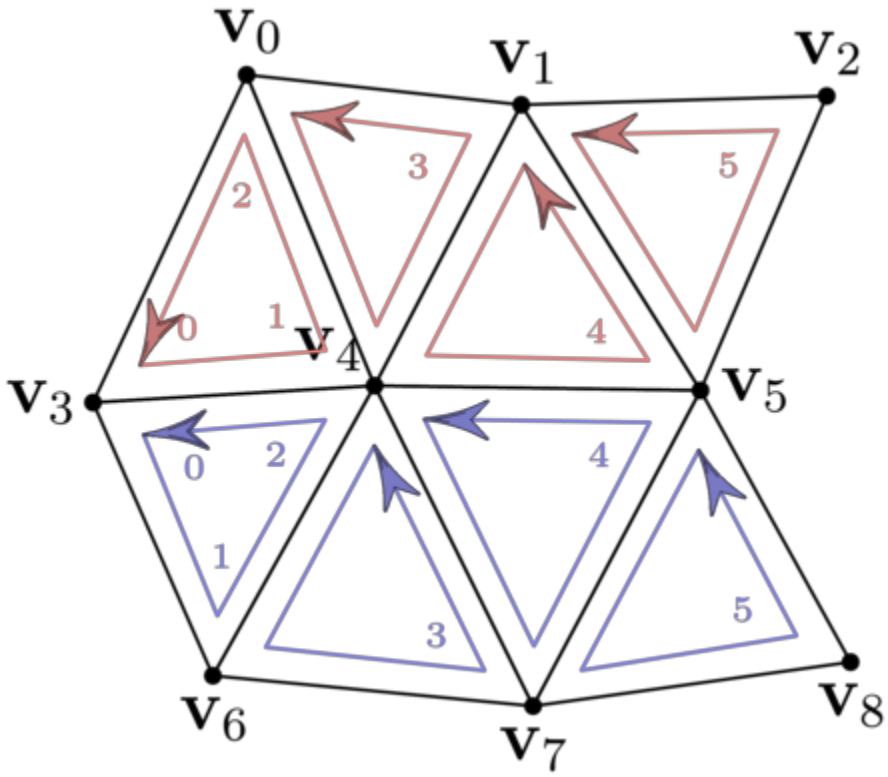
vertex[0]	(x0, y0, z0)
vertex[1]	(x1, y1, z1)
...	...

indexed triangles

- array of vertex data
  - data type: vertex[nv]
  - 12 bytes per vertex with position only
- array of vertex indices (3 per triangle)
  - data type: int[nf][3], often flattened in a single array
  - 24 bytes per triangle
- total storage: ~ 36 bytes (50% memory)
- topology/geometry stored separately/explicitly
  - adjacency queries are well defined

triangles strips

- since triangle share edges, reuse vertices in index list
- requires multiple strips for general case



strip[0]	0,1,2,3,4,5
strip[1]	...
...	...

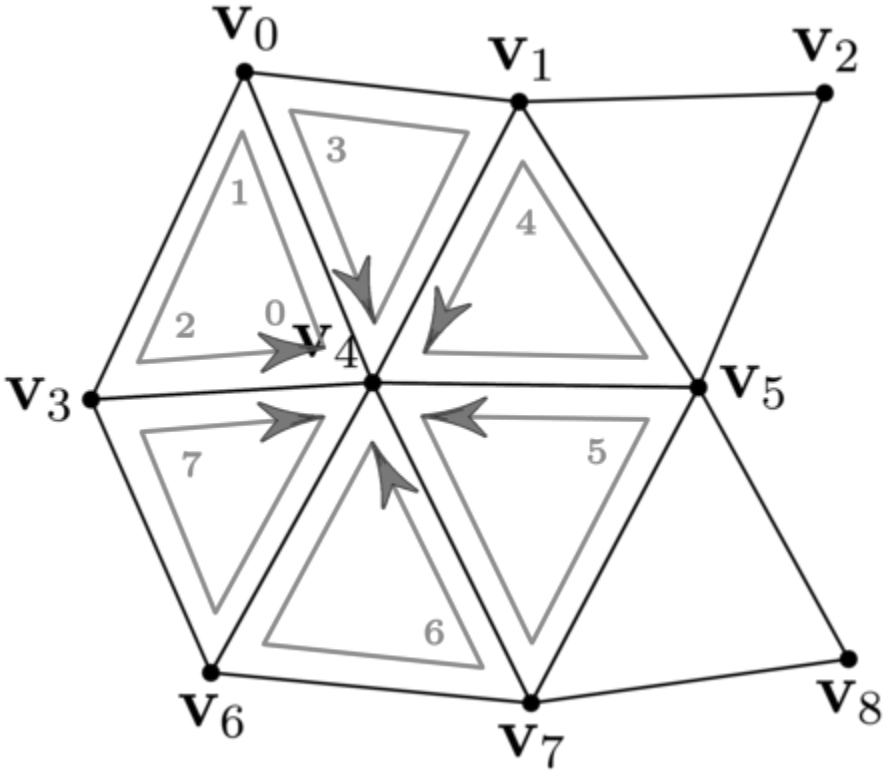
vertex[0]	(x0, y0, z0)
vertex[1]	(x1, y1, z1)
...	...

triangles strips

- array of vertex data
  - vertex[nv]
  - 12 bytes per vertex with position only
- array of lists of vertex indices
  - int[nf][varyingLength]
- for long lists saves about 1/3 index memory

triangle fans

- similar to triangle strips, but different arrangement
- requires many fans, so used little

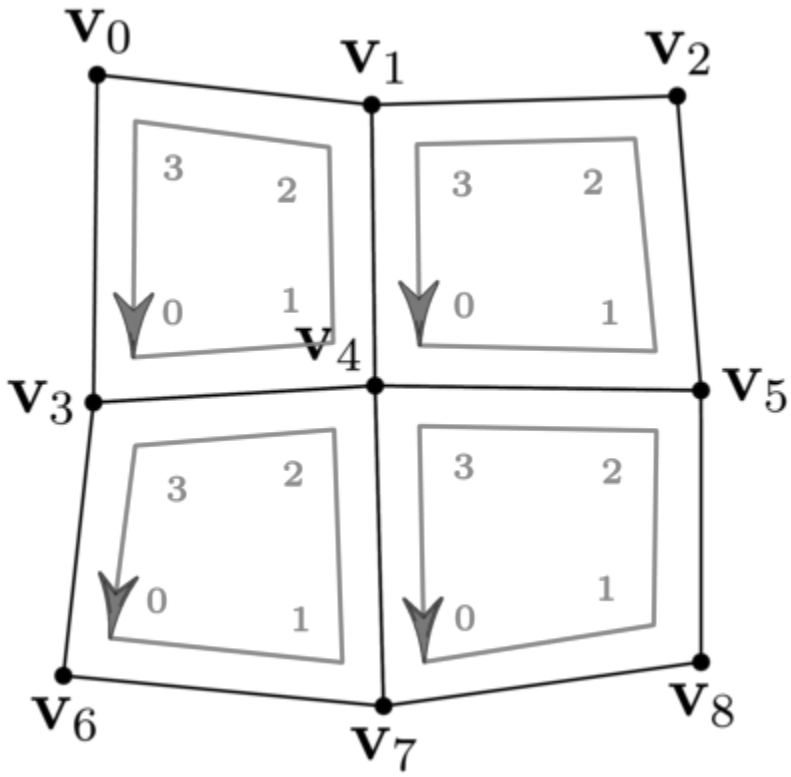


fan[0]	1,2,2,3,4,7,6,0,2
fan[1]	...
...	...

vertex[0]	(x0, y0, z0)
vertex[1]	(x1, y1, z1)
...	...

quad meshes

- similar options as for storing triangles
  - flat quads, indexed quad meshes, quad strips, no fans

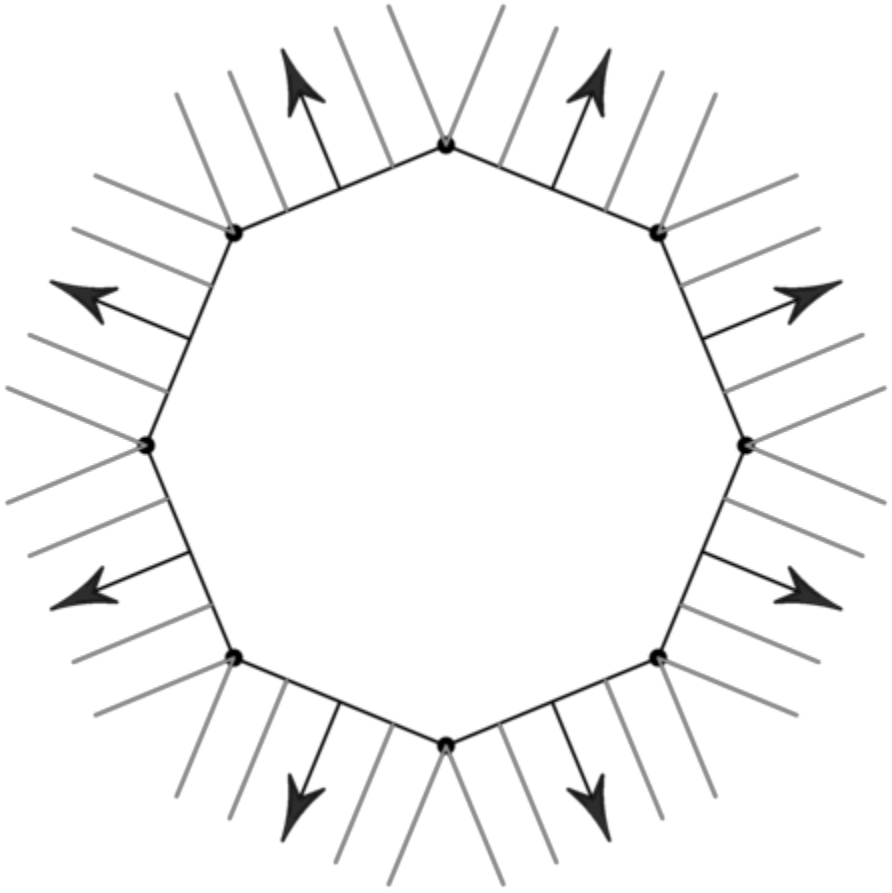


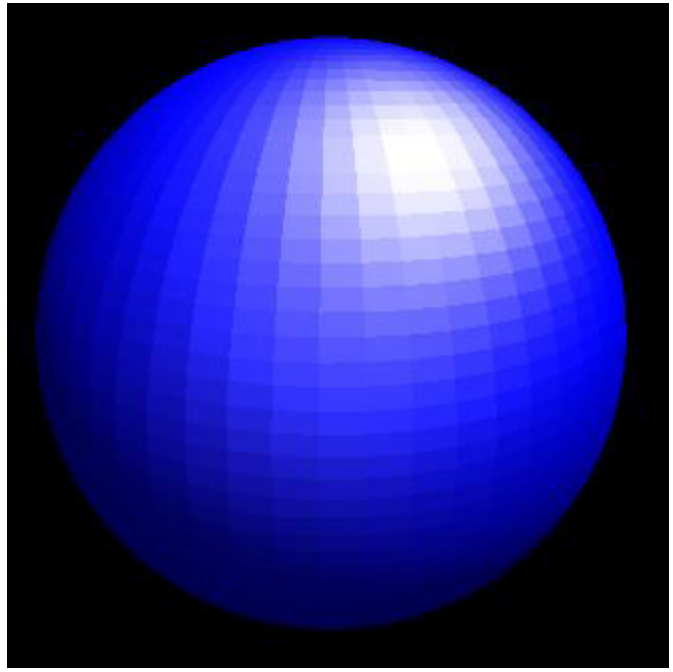
triangle-vs-quad meshes

- triangle meshes
  - well defined: always planar and convex
  - irregular arrangement: hard to manipulate for artists
  - used in rendering as low-level representation
- quad meshes
  - not well defined: can be non-planar and concave
  - rebural arrangement: convenient for modeling
  - converted to triangles for rendering

defining normals

- face normals: same normal for all points in face
  - geometrically correct, but faceted look





defining normals

- vertex normals: store normal at vertices, interpolate in face
  - geometrically “inconsistent”, but smooth look