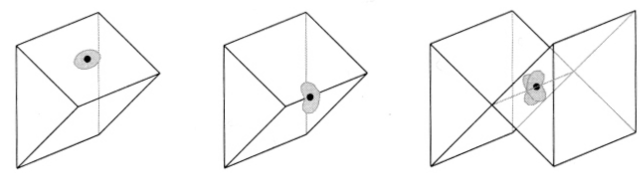
Cornell University CS 569: Interactive Computer Graphics

Triangle meshes

Lecture 14

Topological validity

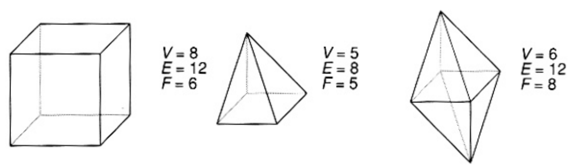
- strongest property, and most simple: be a manifold
  - this means that no points should be "special"
  - manifold:
    - edge points: each edge should have exactly 2 triangles
    - vertex points: each vertex should have one loop of triangles
  - manifold with boundary (looser):
    - edge points: each edge should have at most 2 triangles
    - vertex points: each vertex should have one connected fan of triangles



[Foley et al.]

Notation

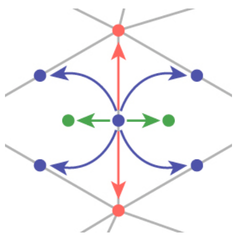
- n_T = #tris; n_V = #verts; n_E = #edges
- Euler: $n_V - n_E + n_T = 2$ for a simple closed surface
  - and in general sums to small integer
  - argument for implication that n_T:n_E:n_V is about 2:3:1



[Foley et al.]

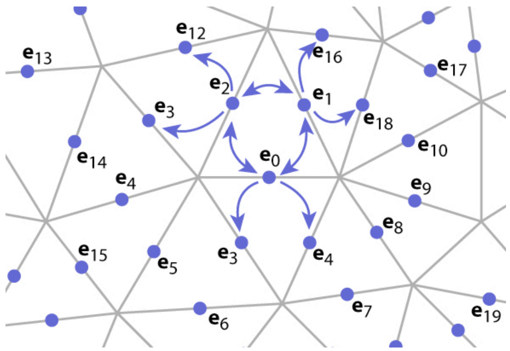
Winged-edge structure

- Edge-centric rather than face-centric
  - therefore also works for polygon meshes
- Each (oriented) edge points to:
  - left and right forward edges
  - left and right backward edges
  - front and back vertices
  - left and right faces
- Each face or vertex points to one edge



Winged-edge structure

	hl	hr	tl	tr
edge[0]	1	4	2	3
edge[1]	18	0	16	2
edge[2]	12	1	3	0
	⋮			

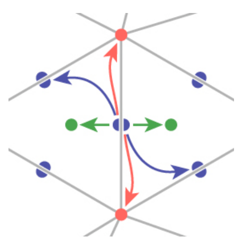


Winged-edge structure

- array of vertex positions: 12 bytes/vertex
- array of 8-tuples of indices (per edge)
  - head/tail left/right edges + head/tail verts + left/right tris
  - int[n_E][8]: about 96 bytes per vertex
    - 3 edges per vertex (on average)
    - (8 indices x 4 bytes) per edge
- add a representative edge per vertex
  - int[n_V]: 4 bytes per vertex
- total storage: 112 bytes per vertex

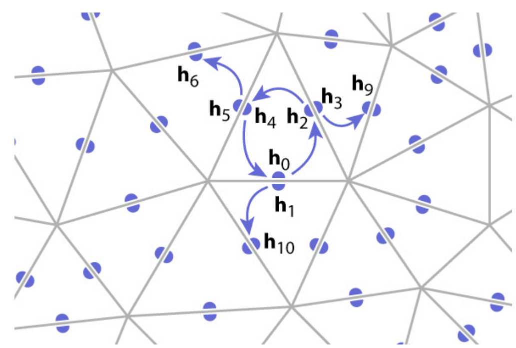
Half-edge structure

- Simplifies, cleans up winged edge
  - still works for polygon meshes
- Each half-edge points to:
  - next edge (left forward)
  - next vertex (front)
  - the face (left)
  - the opposite half-edge
- Each face or vertex points to one half-edge



Half-edge structure

	pair	next
hedge[0]	1	2
hedge[1]	0	10
hedge[2]	3	4
hedge[3]	2	9
hedge[4]	5	0
hedge[5]	4	6
	⋮	



Half-edge structure

- array of vertex positions: 12 bytes/vert
- array of 4-tuples of indices (per h-edge)

  next, pair h-edges + head vert + left tri

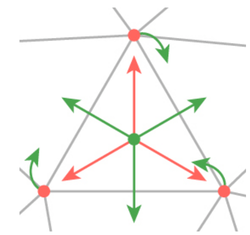
  int[2n_E][4]: about 96 bytes per vertex

  - 6 h. edges per vertex (on average)
  - (4 indices x 4 bytes) per h-edge
- add a representative hedge per vertex

  int[n_V]: 4 bytes per vertex
- total storage: 112 bytes per vertex

Triangle neighbor structure

- Extension to indexed triangle set
- Triangle points to its three neighboring triangles
- Vertex points to a single neighboring triangle
- Can now enumerate triangles around a vertex

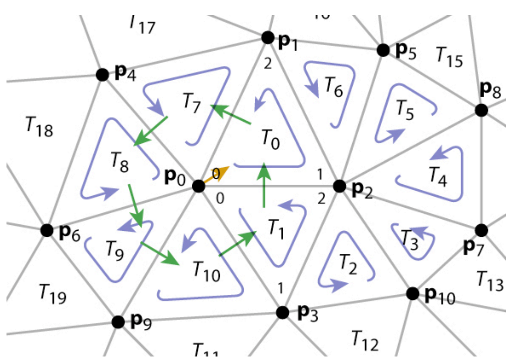


Triangle neighbor structure

vTri[0]	0
vTri[1]	6
vTri[2]	1
vTri[3]	1
	⋮

tNbr[0]	1, 6, 7
tNbr[1]	10, 2, 0
tNbr[2]	3, 1, 12
tNbr[3]	2, 13, 4
	⋮

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
tInd[2]	10, 2, 3
tInd[3]	2, 10, 7
	⋮



Triangle neighbor structure

- indexed mesh is 36 bytes per vertex
- add an array of triples of indices (per triangle)

  int[n_T][3]: about 24 bytes per vertex

  - 2 triangles per vertex (on average)
  - (3 indices x 4 bytes) per triangle
- add an array of representative triangle per vertex

  int[n_V]: 4 bytes per vertex
- total storage: 64 bytes per vertex