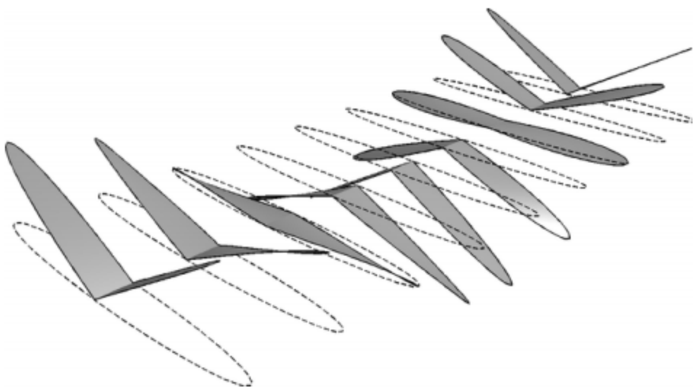
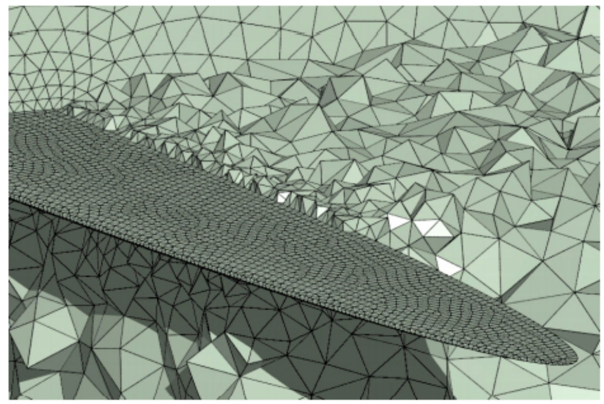
Triangle Mesh

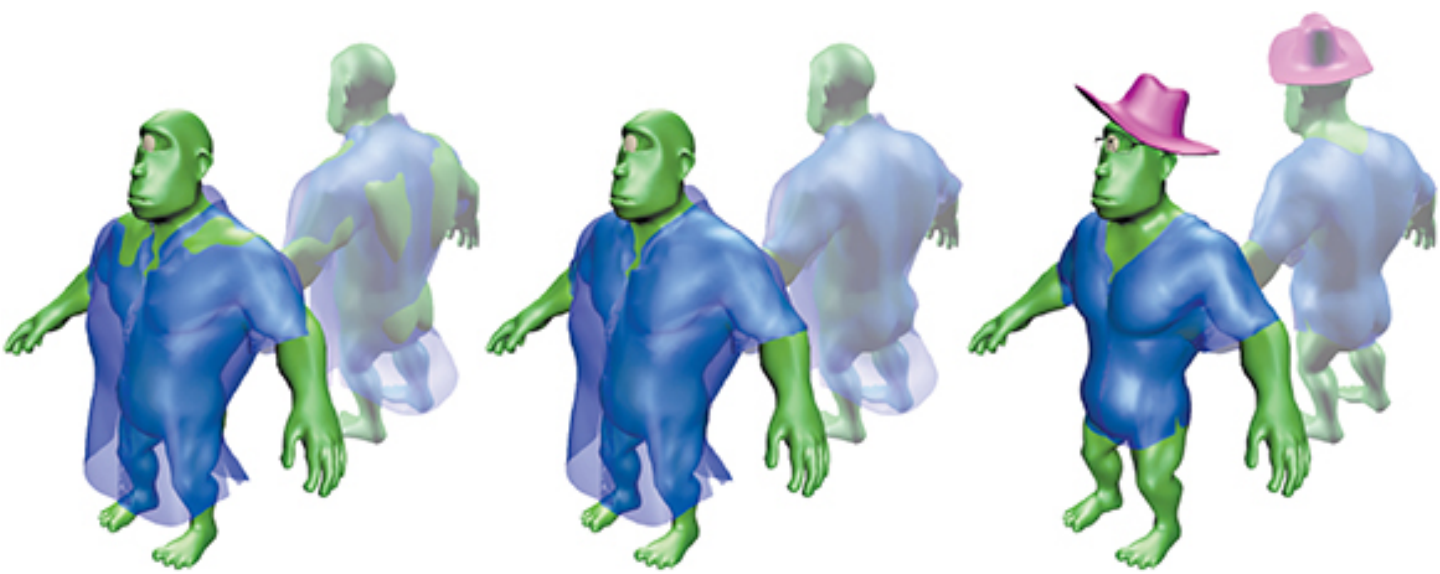
Readings: Chapter 12 (12.1)

Why mesh?



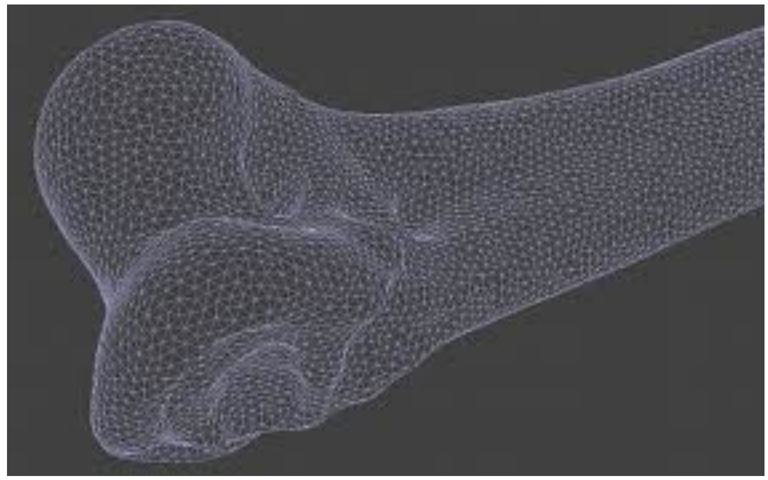


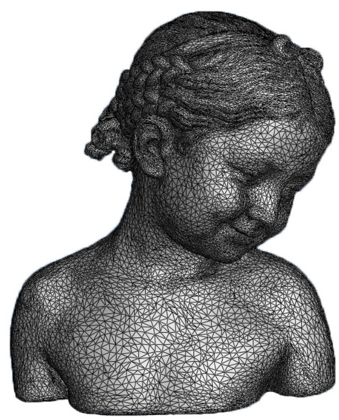
Numerical simulation of flapping wings
Persson, Willis, & Peraire 2011

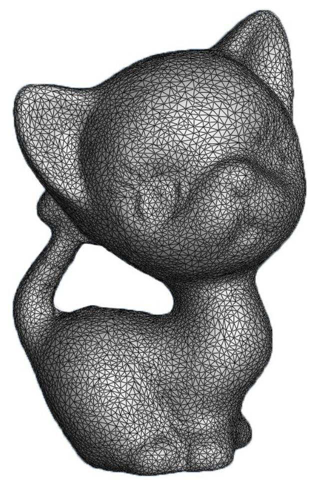


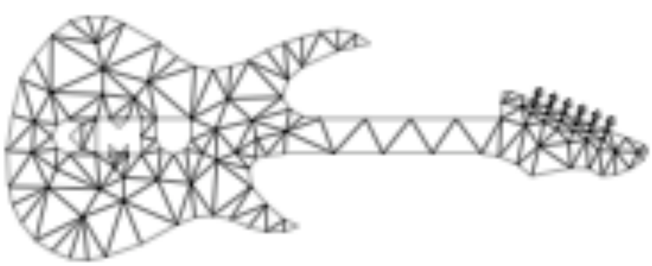
ETH

Why mesh?









Notation

- Nt = # triangles; Nv = # of vertices; Ne = # of edges
- Euler: Nv – Ne + Nt = 2 for a simple closed surface

Representations for triangle meshes

Objectives

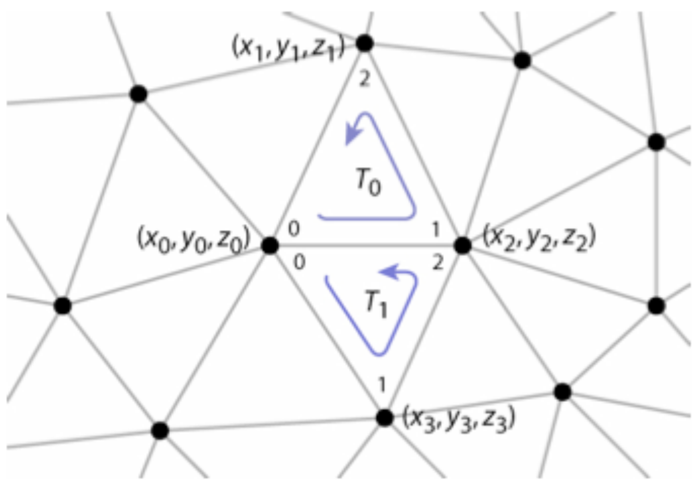
- Compactness
- Efficiency for rendering
- Efficiency of queries
  - All vertices of a triangle
  - All triangles around a vertex
  - Neighboring triangles of a triangle
  - Applications:
    - Finding triangle strips; computing subdivision surfaces; Mesh editing

Methods

- Separate triangles
- Indexed triangle set
  - Shared vertices
- Triangle strips and triangle fans
  - Compression schemes for transmission to hardware
- Triangle-neighbor data structure
  - Supports adjacency queries
- Winged-edge data structure
  - Supports general polygon meshes

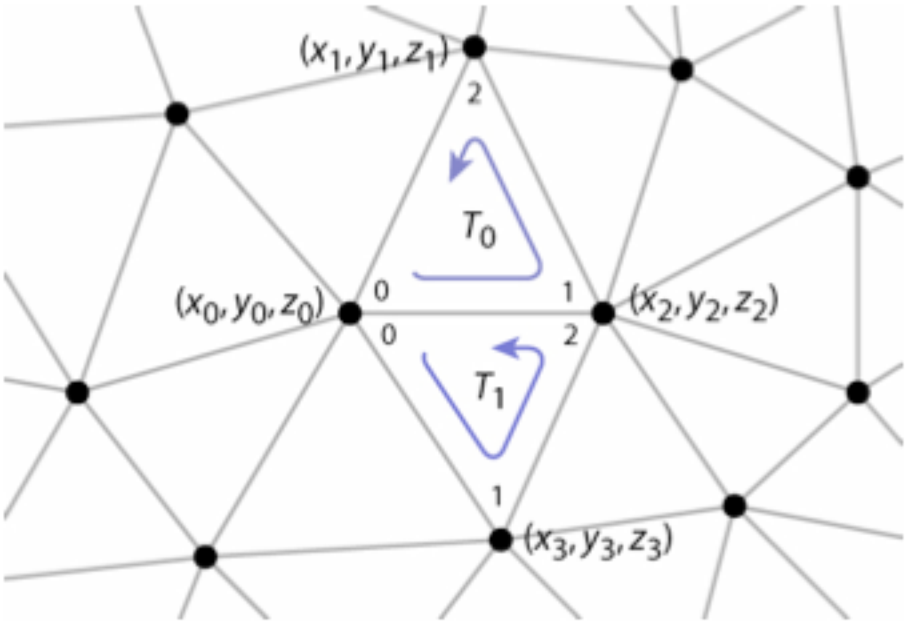
Separate triangles

- Array of triples of points
  - Float [Nt][3][3]: about 72 bytes per vertex
    - 2 triangles per vertex (on average)
    - 3 vertices per triangle
    - 3 coordinates per vertex
    - 4 bytes per coordinate (float)



- Any problems?

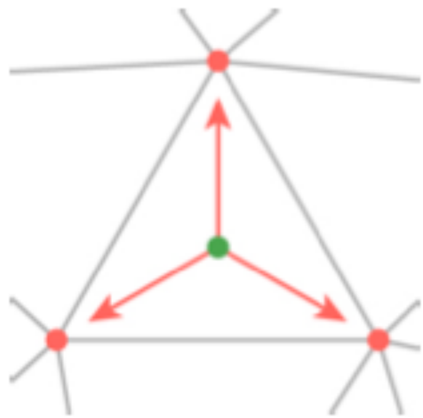
Separate triangles



What is the representation?

Indexed triangle set

- Store each vertex once
- Each triangle points to its three vertices

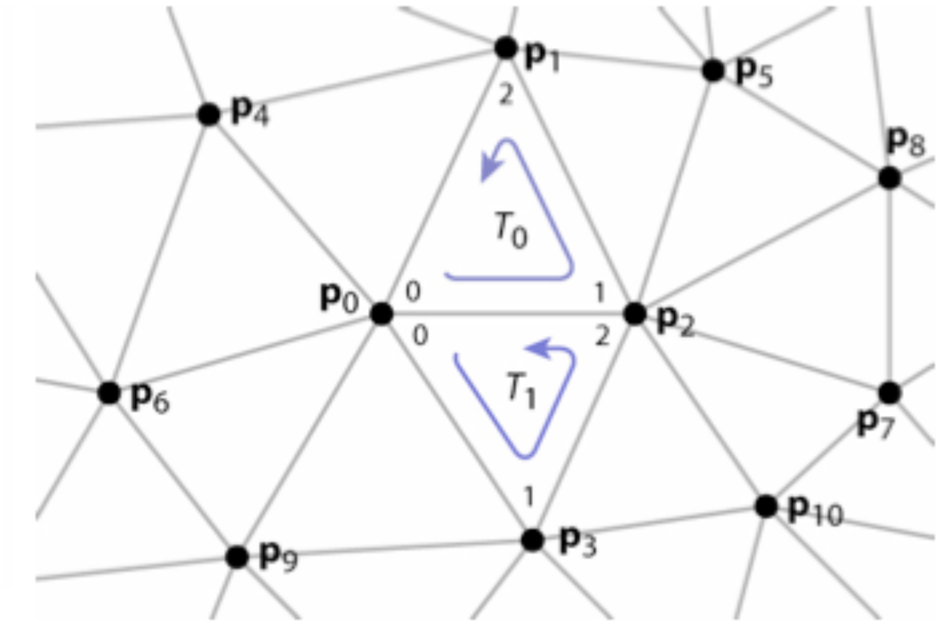


```
Triangle {
 Vertex ver[3];
}

Vertex{
 float pos[3]; // or other data
}

Mesh  {
  float verts[nv][3];
  int tInd[nt][3];
}
```

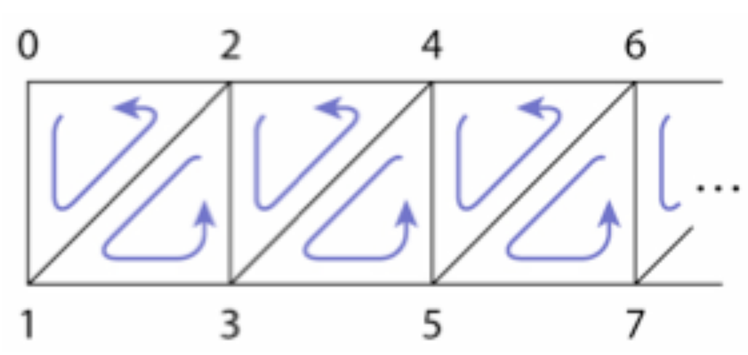
Indexed triangle set



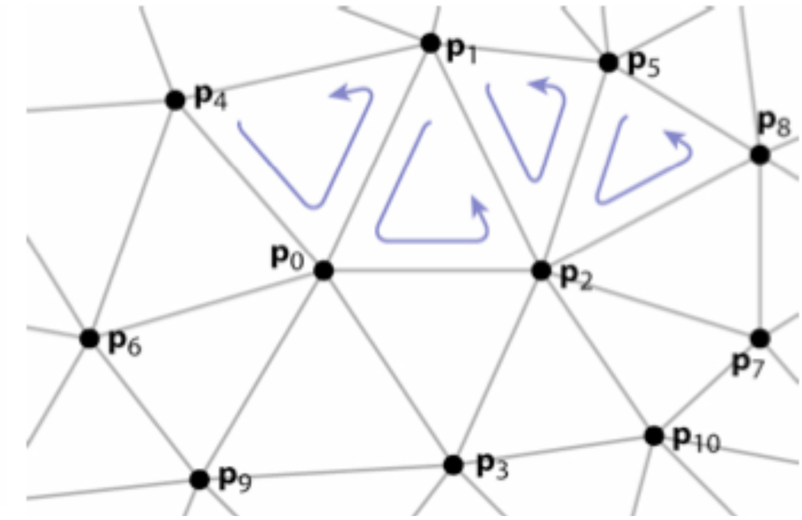
What is the representation?

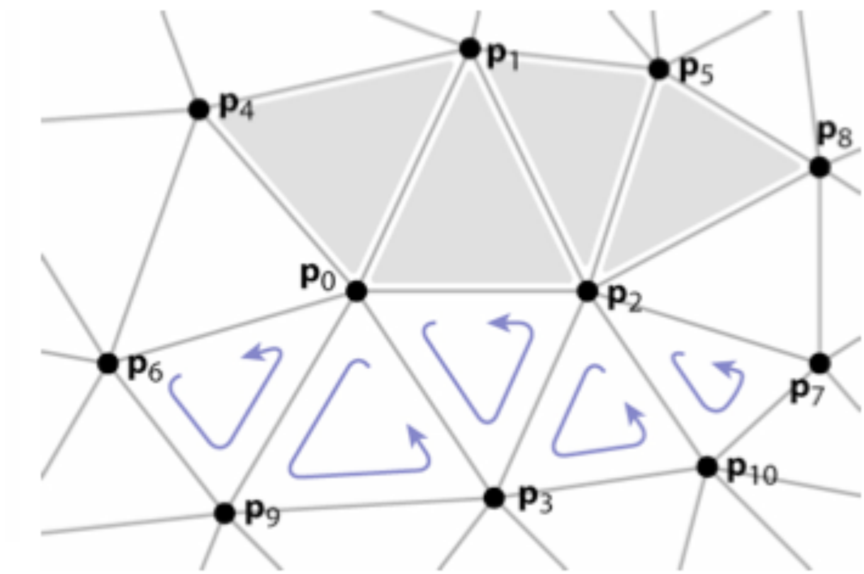
Triangle strips

- Take advantage of the mesh property
  - Each triangle is usually adjacent to the previous
  - Let every vertex create a triangle by reusing the second and third vertices of the previous triangle
  - Every sequence of three vertices produces a triangle
  - E.g., 0, 1, 2, 3, 4 5, 6, 7, .. Leads to
  - (0 1 2), (2 1 3), (2 3 4), (4 3 5), (4 5 6), (6 5 7)



Triangle strips

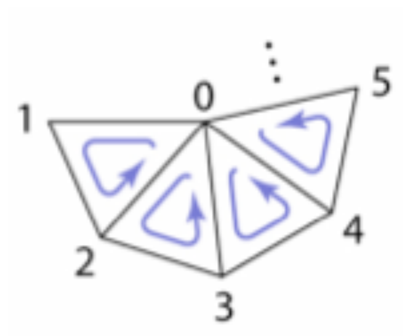




What is the representation?
P4, p0 p1

Triangle fans

- Same idea as triangle strips, but keep oldest rather than newest
  - Every sequence of three vertices produces a triangle
  - E.g., 0, 1, 2, 3, 4, 5, .. Lead to
  - (0 1 2), (0 2 3), (0 3 4), (0 4 5)



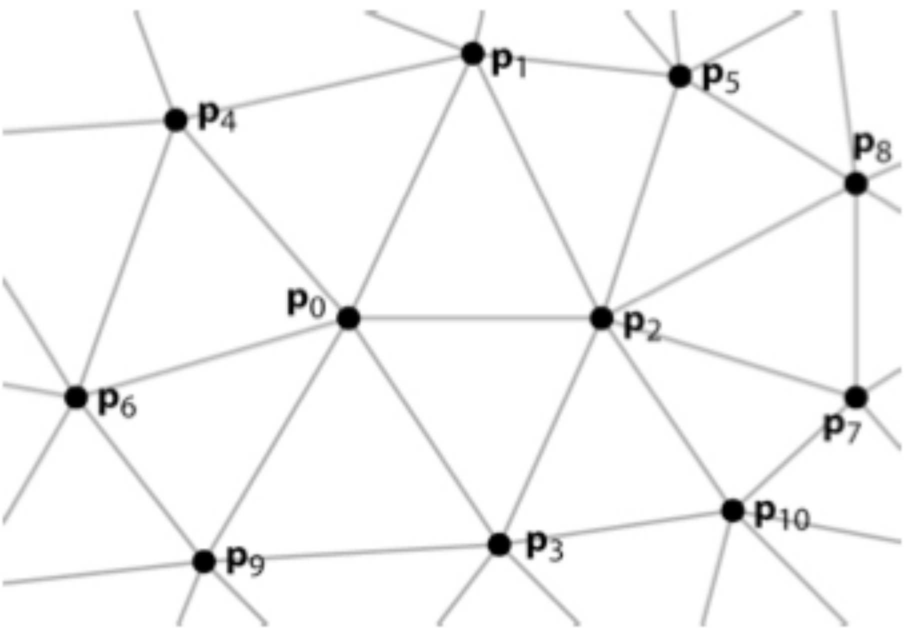
Data structures for mesh connectivity
and
Triangle neighbor structure

Why data structures?

- Given a triangle, what are the three adjacent triangles?
- Given an edge, which two triangles share it?
- Given a vertex, which faces share it?
- Given a vertex, which edges share it?

Half-edge data structure to traverse a mesh

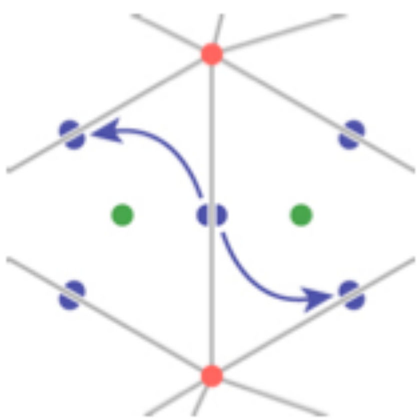
Edge-centric data structure rather than face-centric



(See lecture notes)

Half-edge structure

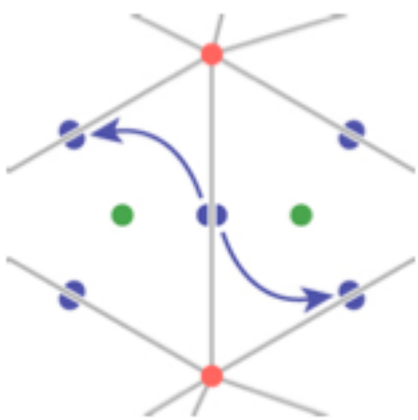
- Each half-edge points to:
  - Next edge (left forward)

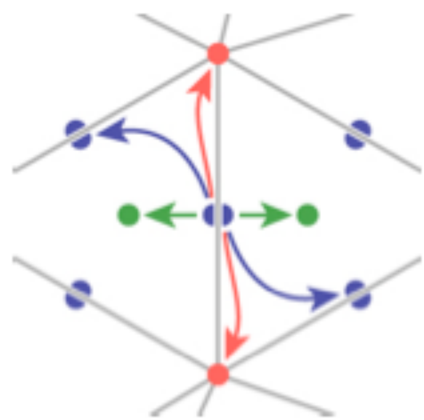


(See lecture notes)

Half-edge structure

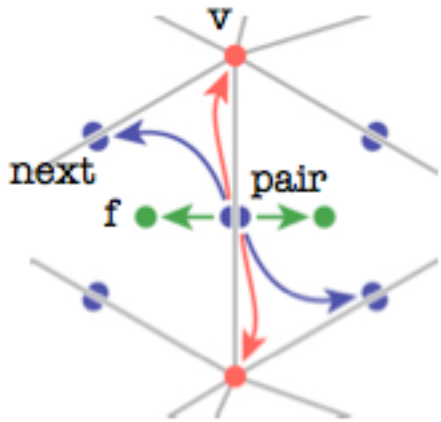
- Each half-edge points to:
  - Next edge (left forward)
  - Next vertex (front)
  - The face (left)
  - The opposite half-edge
- Each face or vertex points to one half-edge





(See lecture notes)

Half-edge structure




```
Hedge {
  Hedge pair, next;
  Vertex v;
  Face f;
}

Face {
  // per-face data
  Hedge h; // any adjacent h-edge
}

Vertex {
  // per-vertex data
  Hedge h; // any incident h-edge
}
```

Half-edge structure