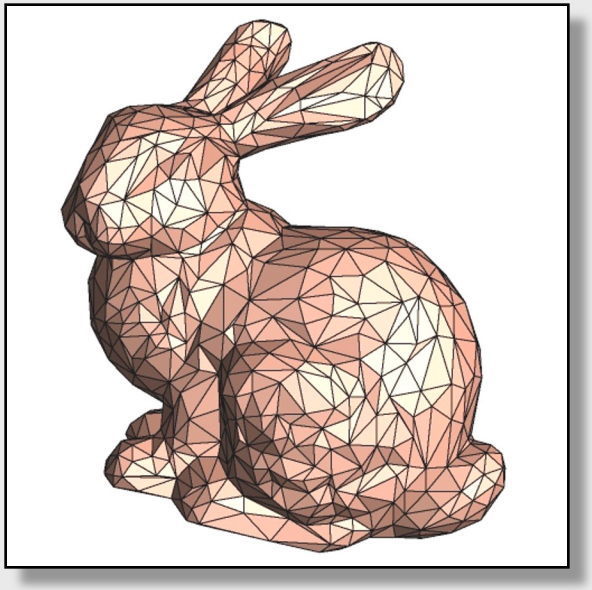
Comp 410/510

Computer Graphics
Spring 2023

Building 3D Models

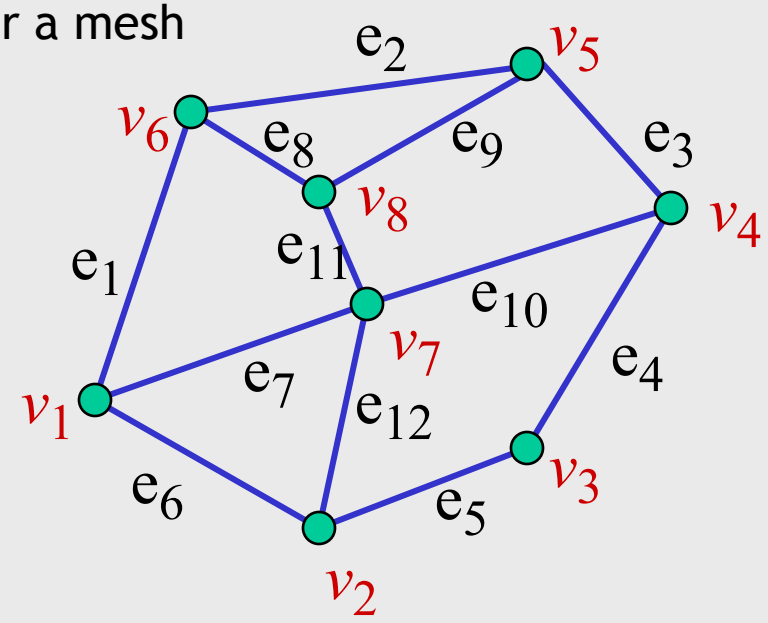
Objectives

- Introduce simple data structures for building polygonal models
  - Vertex lists
  - Polygon (face) lists
  - Edge lists



Representing a Mesh

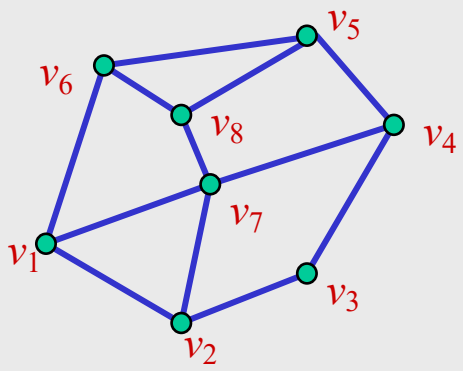
- Consider a mesh



- There are 8 nodes (vertices) and 12 edges
  - 5 interior polygons
  - 6 interior (shared) edges
- Each vertex has a location $v_i = (x_i\ y_i\ z_i)$

Simple Representation

- Define each polygon by geometric locations of its vertices



- Leads to an OpenGL code such as

```
vertices[i] = vec3(x1, y1, z1);
vertices[i+1] = vec3(x6, y6, z6);
vertices[i+2] = vec3(x7, y7, z7);
i+=3;
```

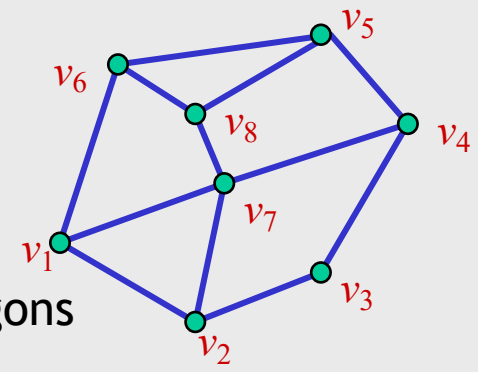
} a triangle

- Inefficient and unstructured
  - Consider moving a vertex to a new location
  - Must search for all occurrences of the same vertex

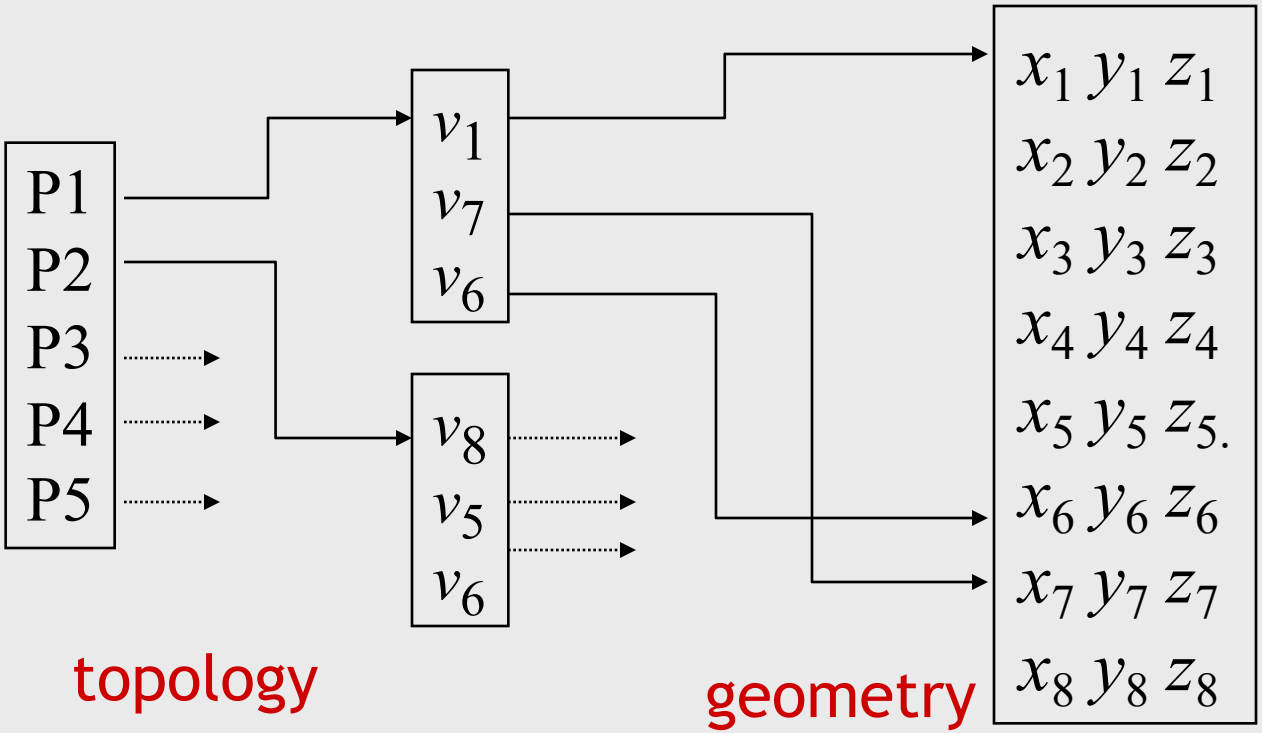
Geometry *vs.* Topology

- Generally it is a good idea to use data structures that separate geometry from topology
- Geometry: Locations of the vertices
- Topology/Connectivity: Organization of the vertices and edges
  - **Example:** A polygon is an ordered list of vertices, with edges connecting successive pairs of vertices, and the last to the first
  - Topology holds even if geometry changes

Polygon (Face) Lists

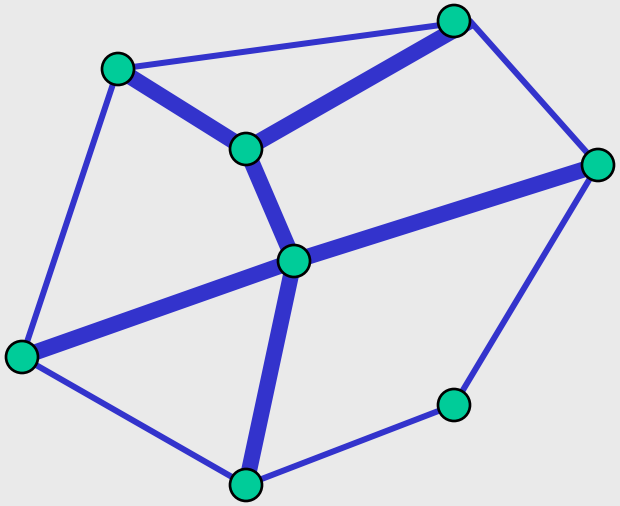


- Put the geometry in an array
- Use pointers (or indices) to associate vertices and polygons
- Introduce a polygon list



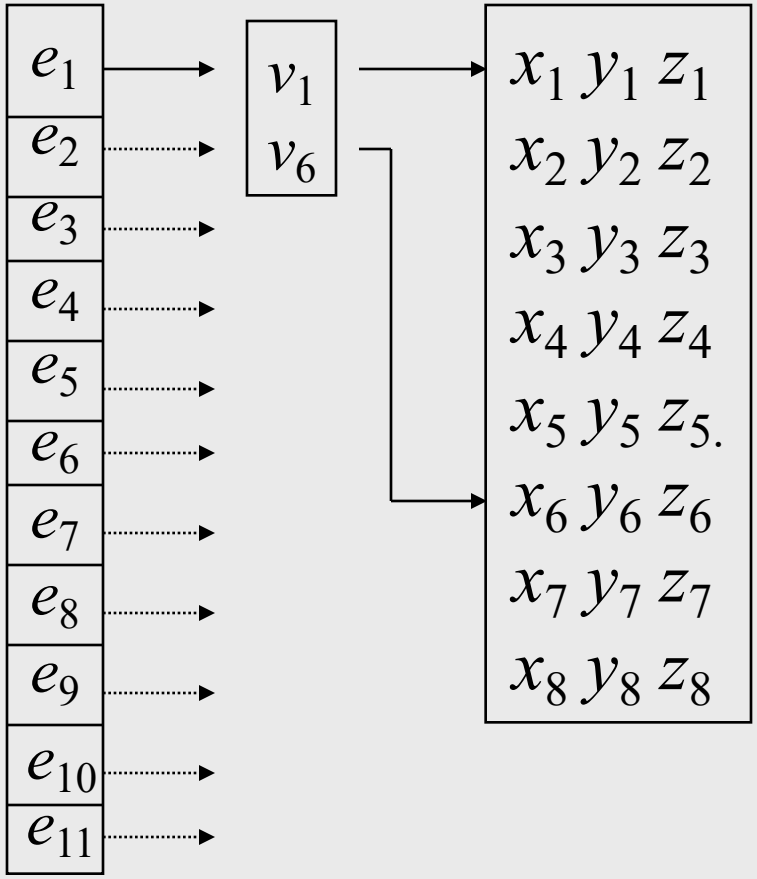
Shared Edges

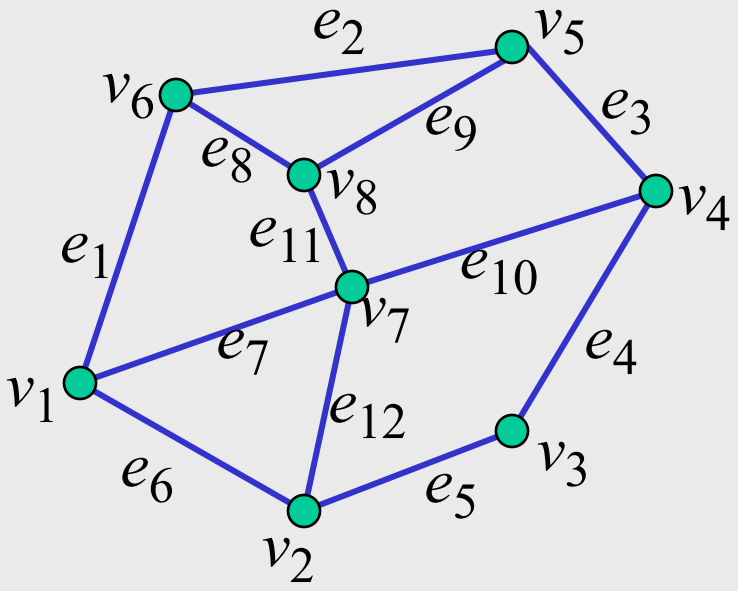
- Polygon lists will draw filled polygons correctly
- But if we draw the polygon by its edges, shared edges are drawn twice



- Can instead store a mesh by edge list

Edge List





Note that polygons are not explicitly represented

OpenGL

- Use `glDrawElements(GL_TRIANGLES,…)` for polygon lists with an index buffer
- Cube example:

```
//init
GLuint cube_indices[] = {0, 1, 2, 2, 3, 0, 1, 5, 6,6, 2, 1,7, 6, 5,5, 4, 7,4, 0, 3,3, 7,
4,4, 5, 1,1, 0, 4,3, 2, 6,6, 7, 3};

// Create and initialize an index buffer object
GLuint index_buffer;
glGenBuffers(1, &index_buffer);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, index_buffer);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(cube_indices), cube_indices, GL_STATIC_DRAW);

……

//in display callback
glDrawElements(GL_TRIANGLES, NumVertices, GL_UNSIGNED_INT, 0);
```

- Use `glDrawElements(GL_LINES,…)` or `glDrawElements(GL_LINE_STRIP,…)` for edge lists

See **spinCube (with index buffer)** example

Inward and Outward Facing Polygons

- The vertex order $\{v_1, v_2, v_7\}$ and $\{v_2, v_7, v_1\}$ are equivalent so that the same polygon will be rendered by OpenGL but the order $\{v_1, v_7, v_2\}$ is different.
- The first two describe outwardly facing polygons.
- Use the right-hand rule = counter-clockwise encirclement of outward-pointing normal.
- OpenGL can treat inward and outward facing polygons differently.