Modeling:
Mesh Representations

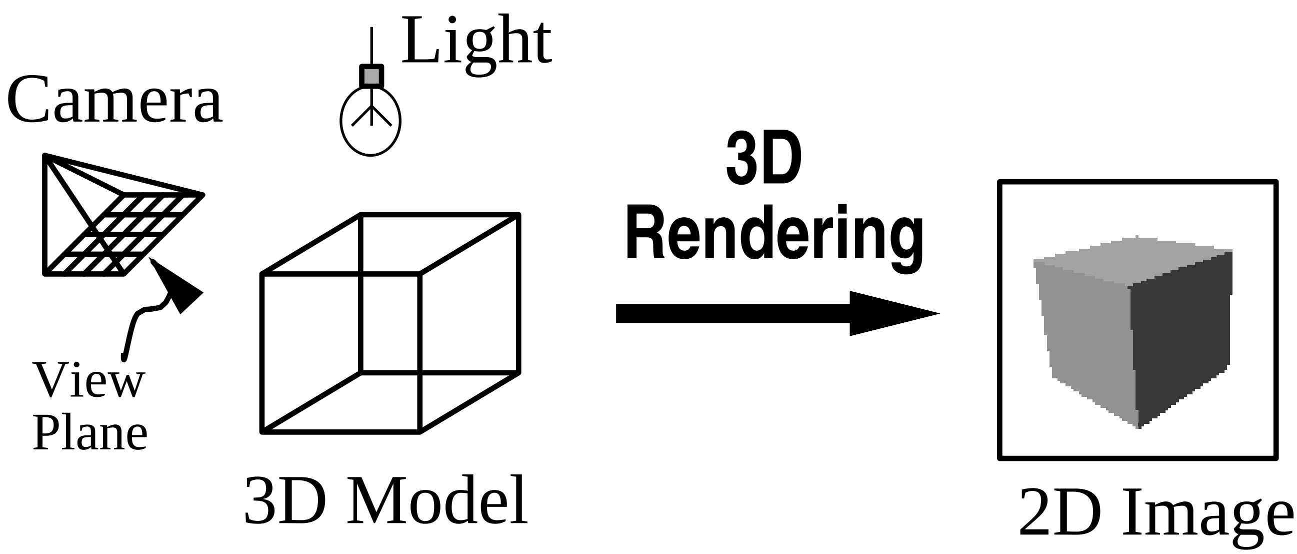
Thomas A. Funkhouser

Where Are We Now?

- **Image Processing**
- **Rendering**
  - Direct illumination
  - Global illumination
- **Modeling**
- **Animation**

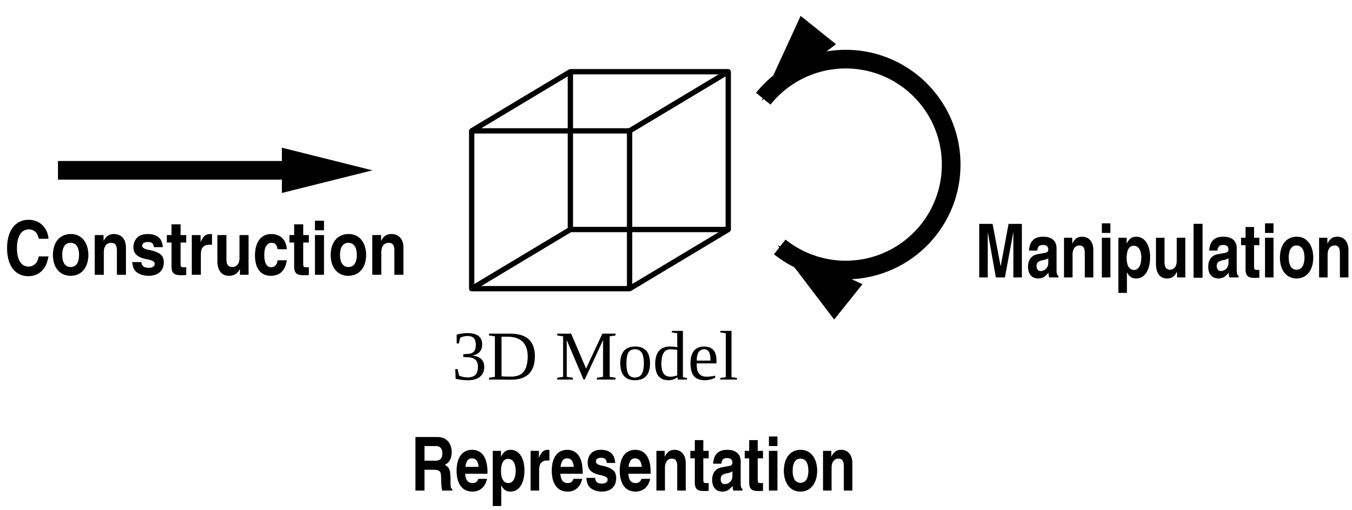
3D Rendering

- **How do we ...**
  - draw 3D objects with a computer?



3D Modeling

- **How do we ...**
  - represent 3D objects in a computer?
  - construct such representations quickly and/or automatically with a computer?
  - manipulate, analyze, verify, ... 3D geometrical objects with a computer?



3D Representations

- **Boundary representations**
  - Mesh representations
  - Parametric surfaces
  - Subdivision surfaces
- **Solid representations**
  - Voxels & Octrees
  - BSP trees
  - Constructive solid geometry
  - Algebraic surfaces
- **Image−based representations**
  - Images & panoramas
  - Light field & lumigraph
- **Composite representations**
  - Scene graphs

3D Representations

- **Accuracy**
  - How well does the representation approximate the object?
- **Computational Efficiency**
  - How quickly can we generate images from the representation?
  - How quickly can we compute intersections with the rep?
- **Storage Efficiency**
  - How much data is required to store the representation?
- **Construction Efficiency**
  - How easy is it to construct the repesentation from available input data?

Today’s Lecture

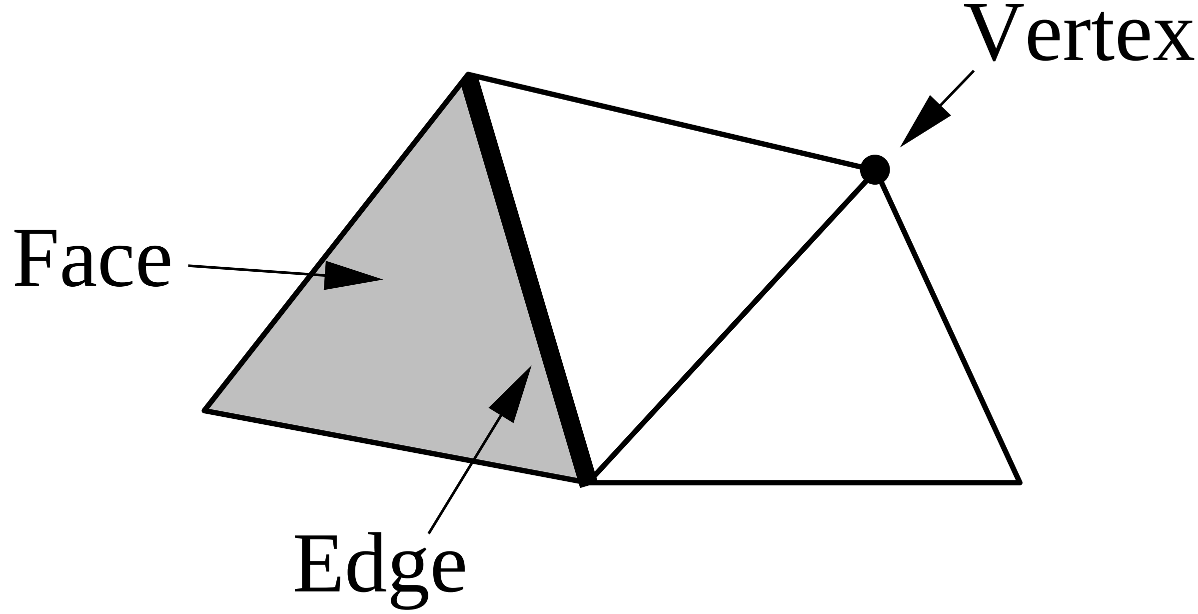
- **Mesh Representations**
  - Set of Faces
  - Triangle strips
  - Vertex tables
  - Adjacency lists
  - Winged−edge

- **Mesh Operations**
  - Traversal operations
  - Euler operations
  - Compound operations

Mesh Representations

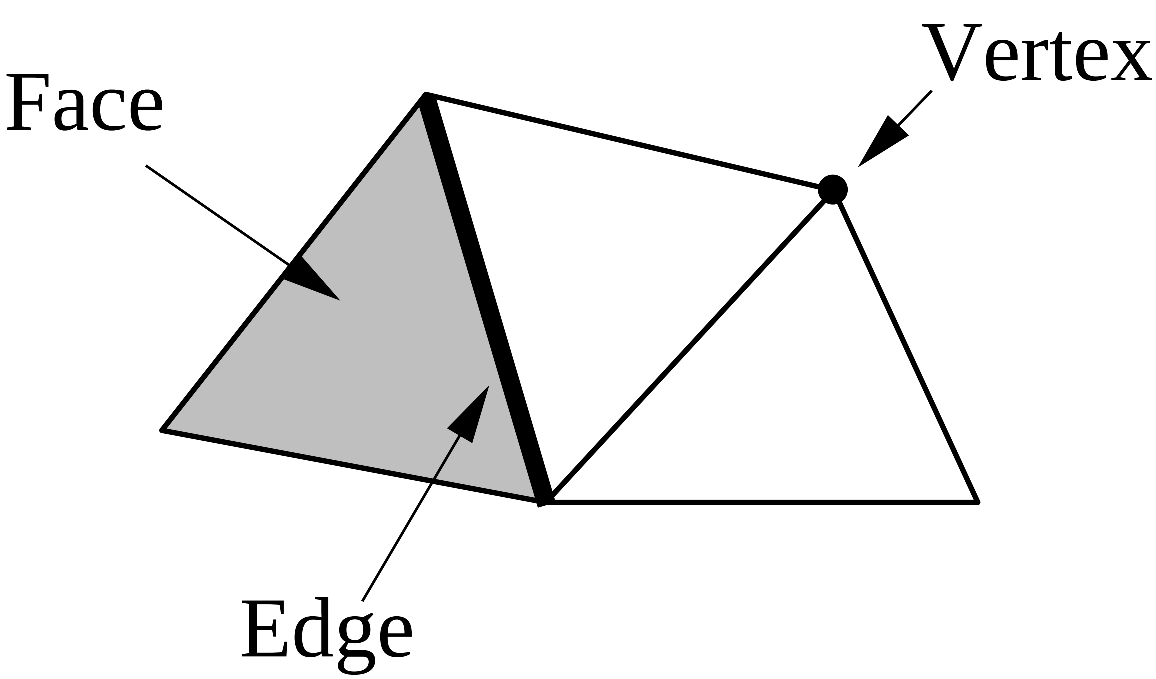
- **Properties:**
  - Boundary representation, so solids not explicitly represented
  - Piecewise linear, so approximate curved surfaces

- **Mesh Descriptions**
  - Vertex and Face tables
  - Triangle strips
  - Adjacency lists
  - Winged−edge
  - Multiresolution



Mesh Representations

- **Boundary representation**
  - Describes surface of object
  - Solids not explicitly represented
- **Piecewise linear**
  - Set of polygons
  - Approximate curved surfaces

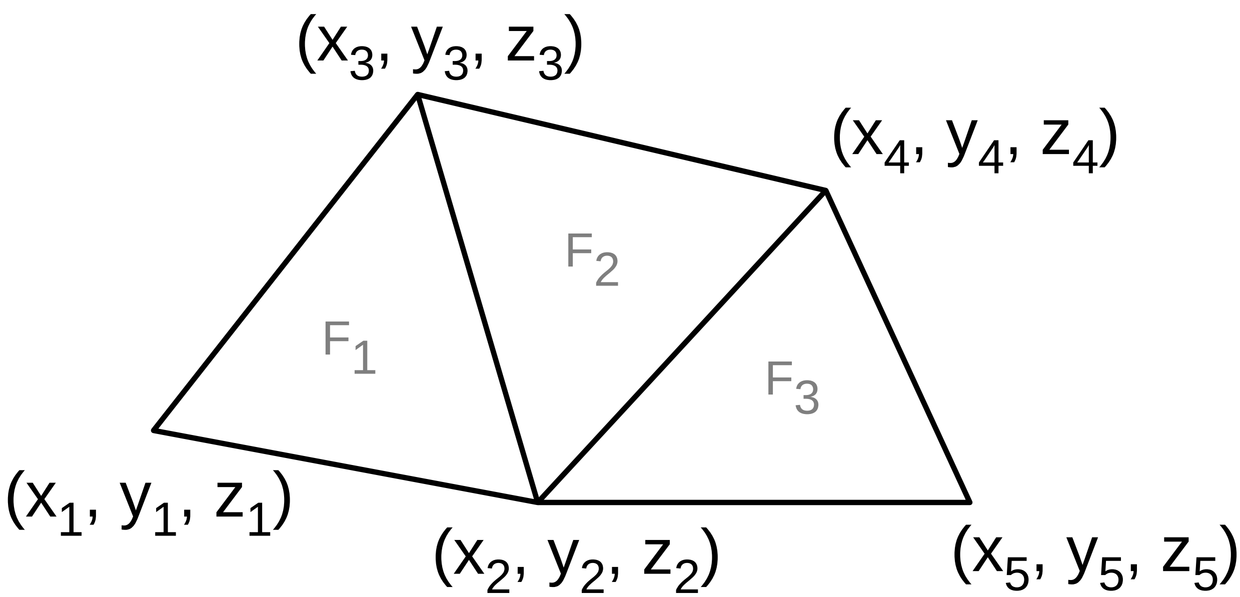


Mesh Representations

- **Possible representations**
  - List of Faces
  - Triangle strips
  - Vertex tables
  - Adjacency lists
  - Winged−edge

List of Faces

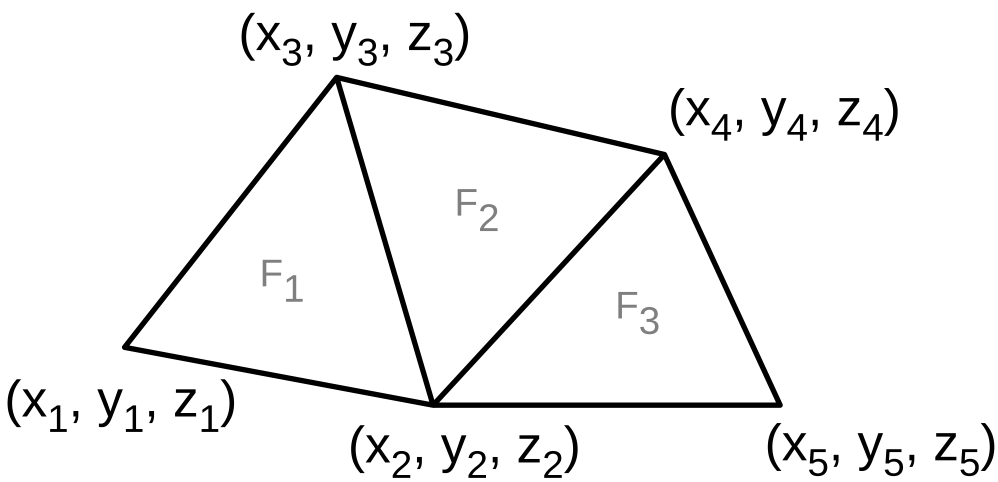
- **Each face lists vertex coordinates ...**
  - Redundant vertices
  - No topology information
  - Not hierarchical or multiresolution



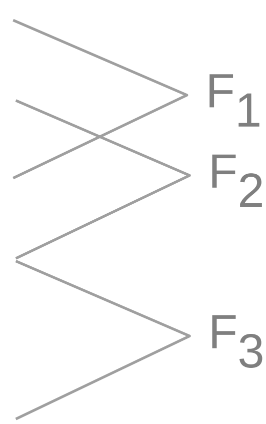
FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Triangle Strips

- **F_i is triangle (V_i, V_{i+1}, V_{i+2})**
  - Faces are implicit
  - Only k−sided faces
  - Limited vertex sharing
  - Limited adjacency information
  - Not hierarchical or multi−resolution

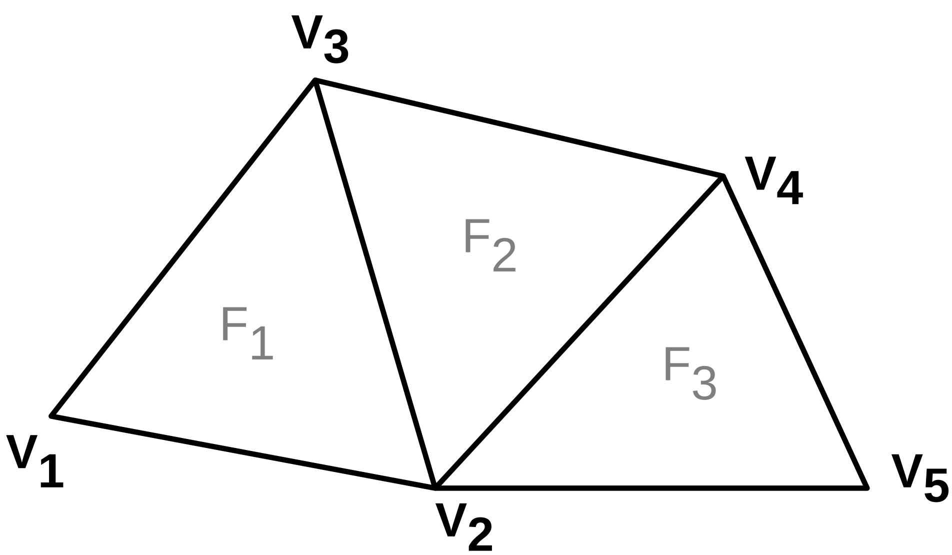


TRI STRIP
(x_1, y_1, z_1)
(x_2, y_2, z_2)
(x_3, y_3, z_3)
(x_4, y_4, z_4)
(x_2, y_2, z_2)
(x_5, y_5, z_5)



Vertex Tables

- **Each face lists vertex references ...**
  - \+ Shared vertices
  - − No adjacency information
  - − Not hierarchical or multiresolution
  - − Adjacency in O(n) time, generally

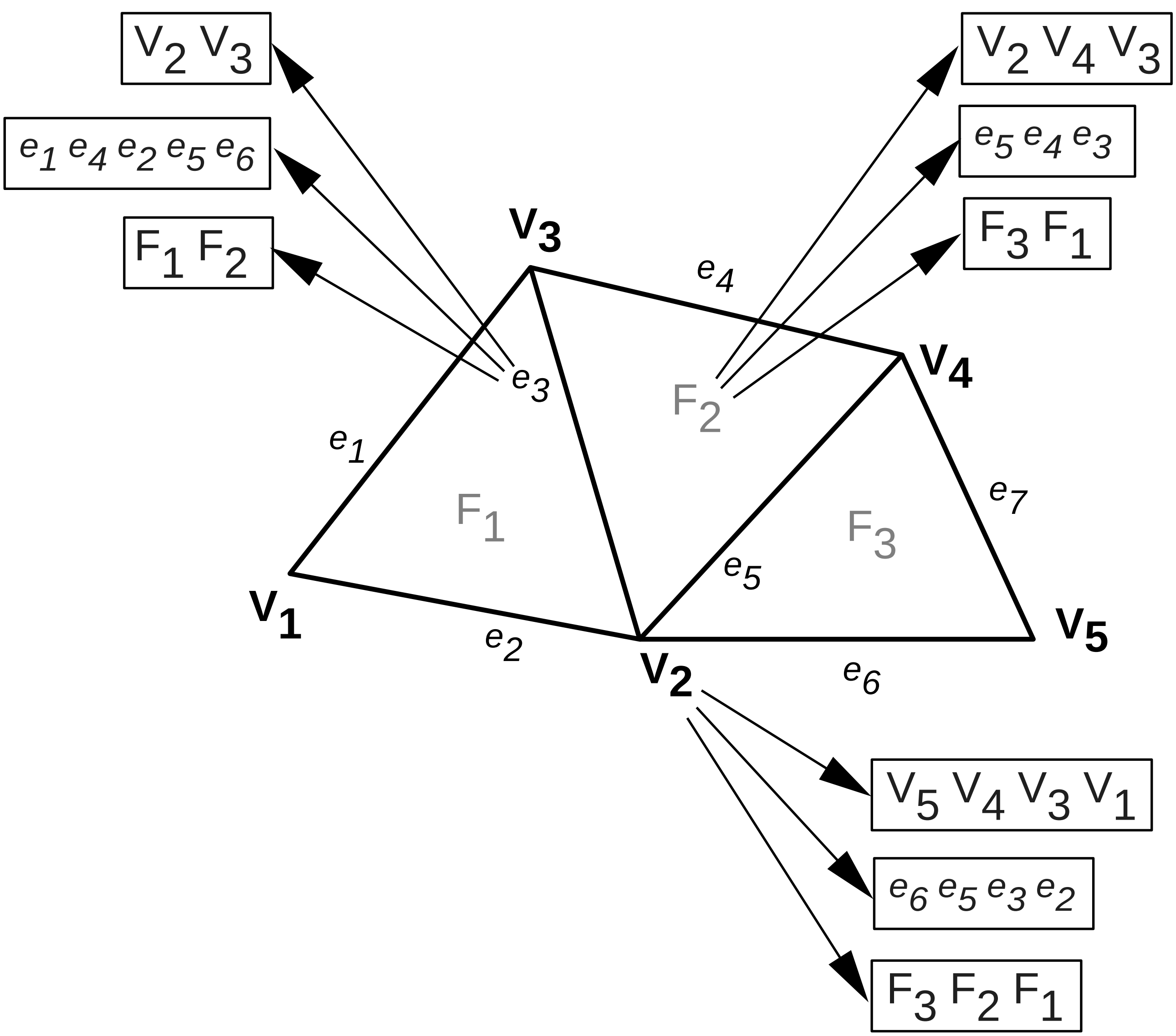


VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

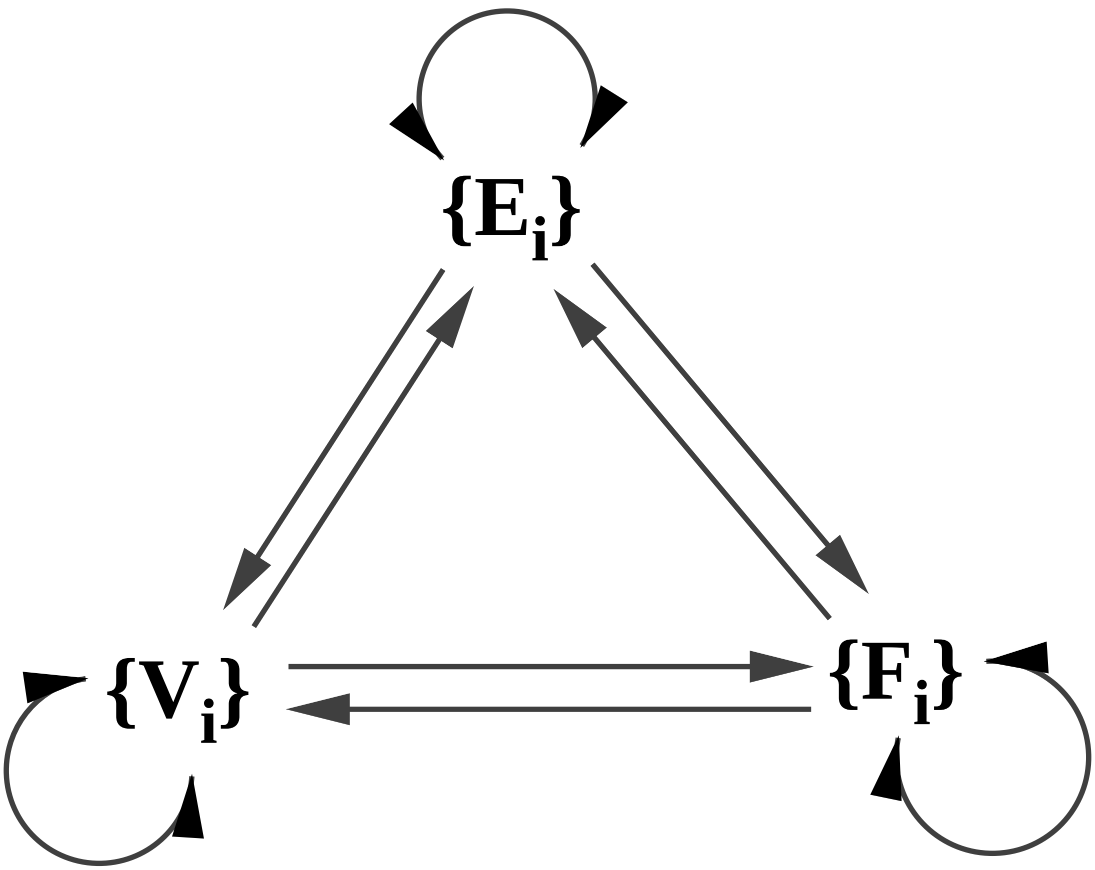
Adjacency Lists

- **Store all adjacency relationships**
  - Adjacency in one lookup
  - Efficient topology traversal
  - Extra storage



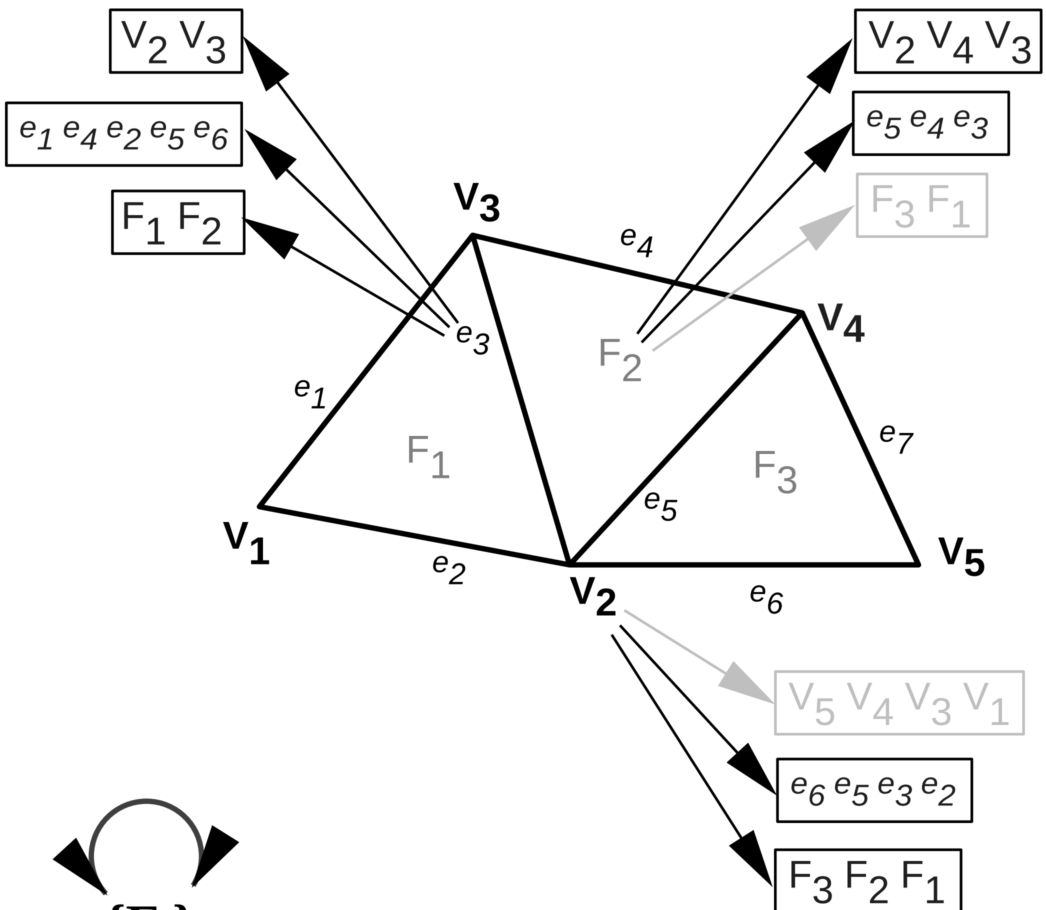
Adjacency Lists

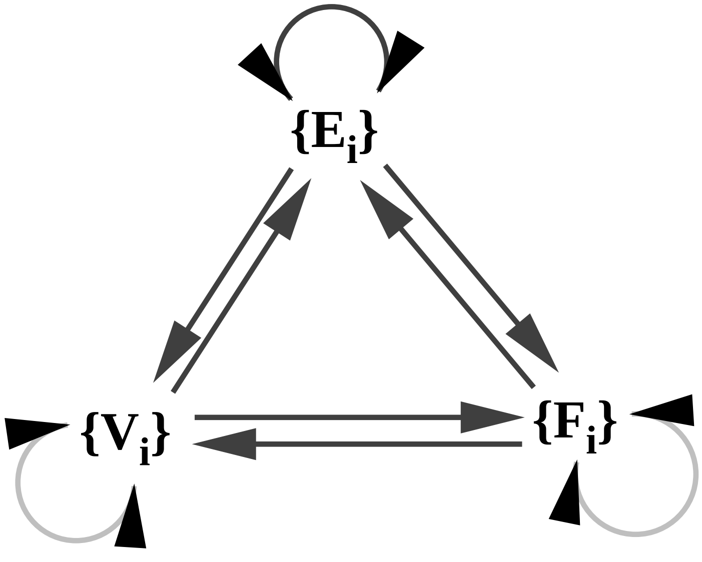
- **Directed Graph Schematic** (Woo ‘85)
  - Directed edge = explicit relation
  - Directed path = implicit relation



Partial Adjacency Lists

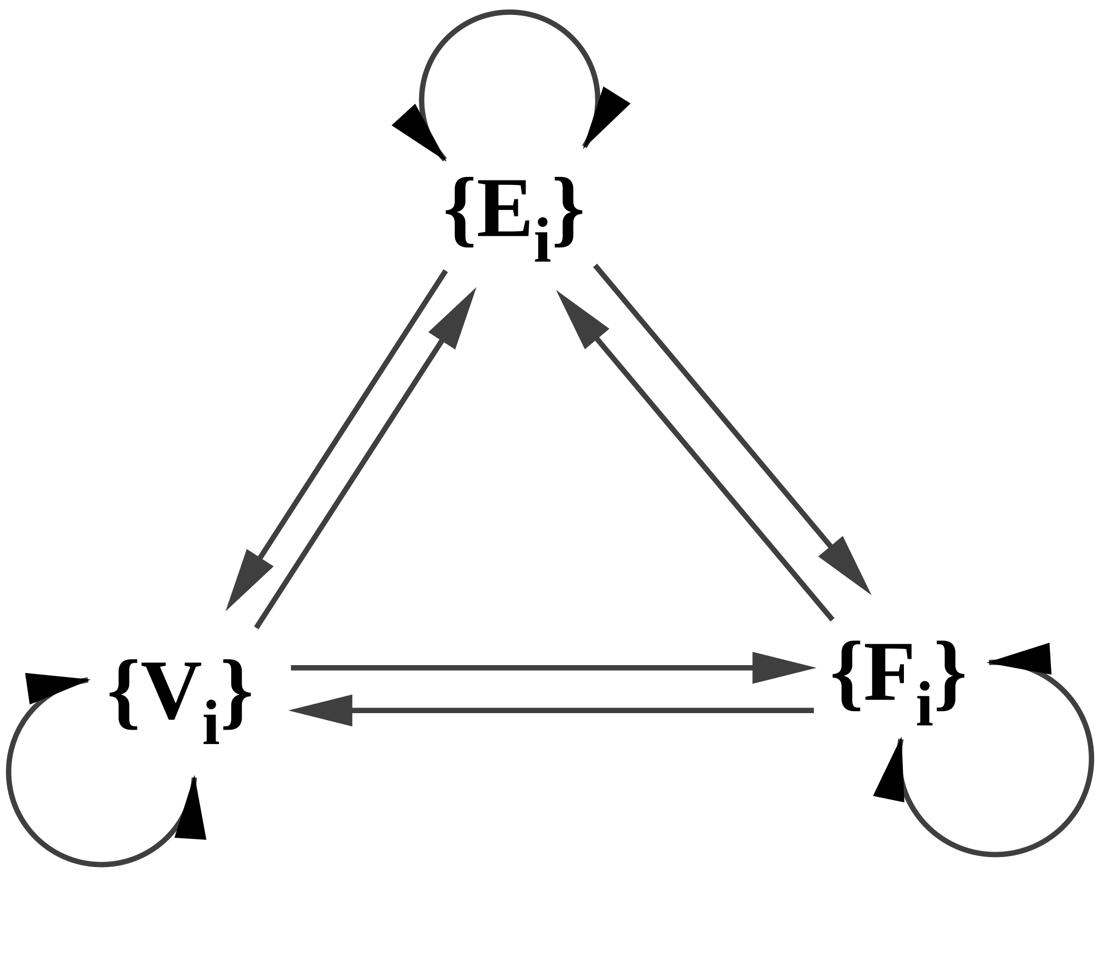
- Store some adjacency relationships and derive others





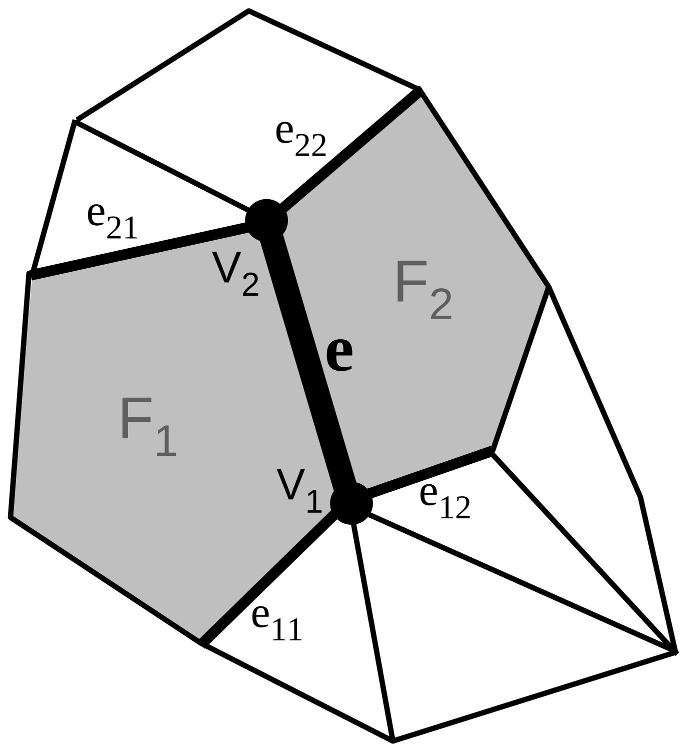
Partial Adjacency Lists

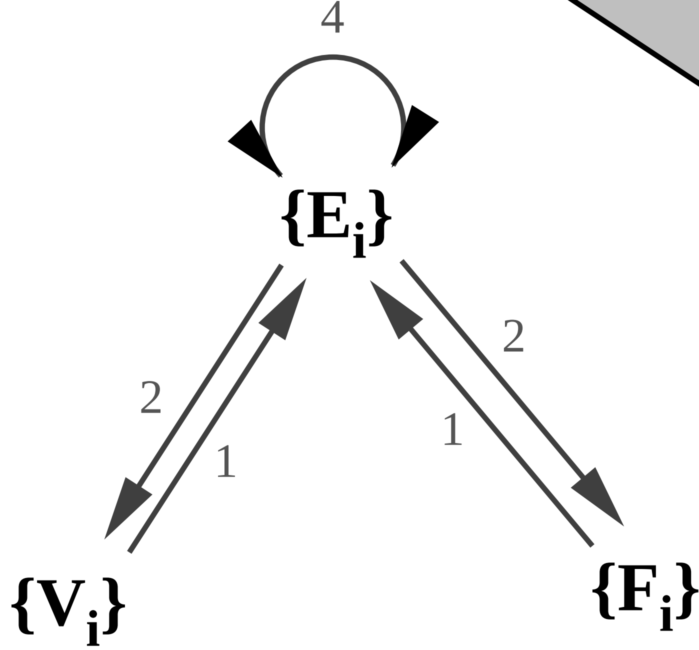
- **Which relations should be stored?**
  - Maintain fast adjacency queries
  - Use least storage



Winged Edge

- **Adjacency encoded in edges** (Baumgart ‘72)
  - Adjacency in O(1) time
  - Little extra storage (fixed records)



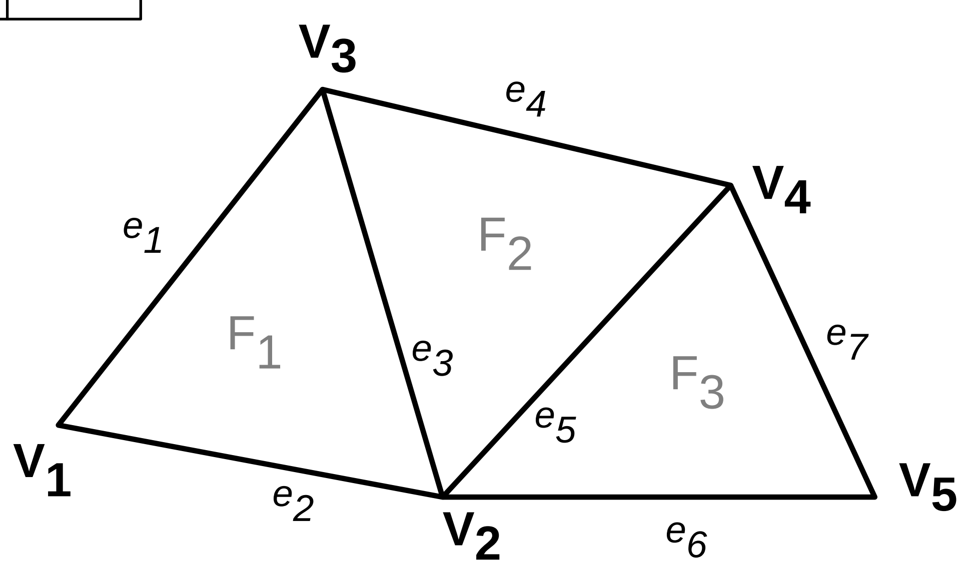


Winged Edge

EDGE TABLE					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5



Winged Edge

- **Deriving adjacency relationships**

F -> e_1:

return F.Edge();

F, e_i -> e_{i+1}:

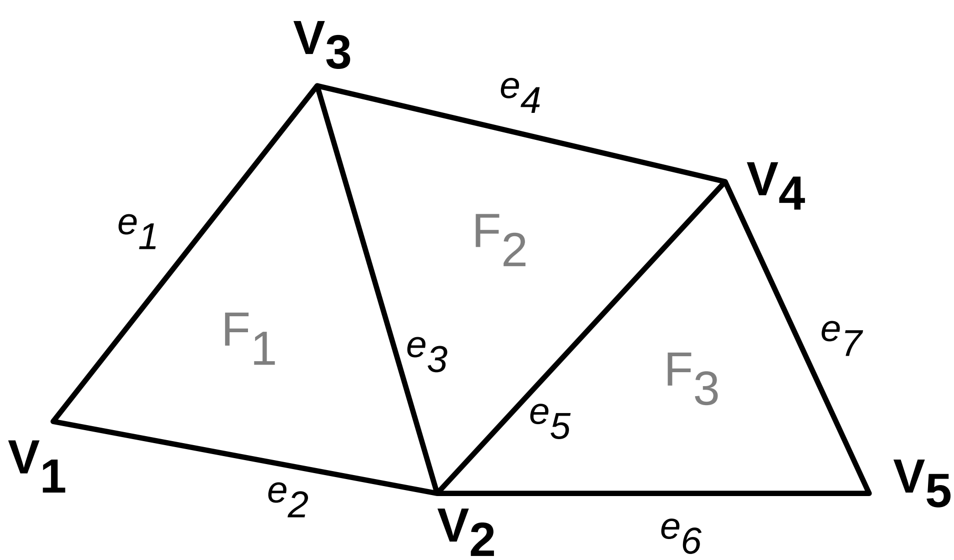
if (e_i.Face(1) == F) return e_i.Edge(2,1);

else return e_i.Edge(1,2);

F, e_i -> V_i:

if (e_i.Face(1) == F) return e_i.Vertex(1);

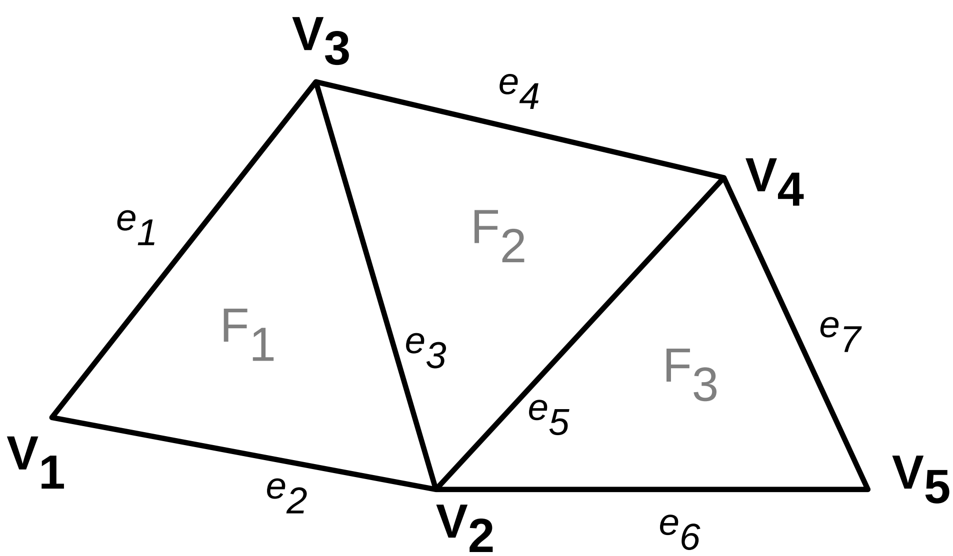
else return e_i.Vertex(2);



Mesh Operations

- **Topological traversal**
  - For each face, enumerate vertices
  - For a face, find adjacent faces
  - For an edge, find adjacent edges
  - For a vertex, find adjacent faces

- **Topological surgery**
  - Insert vertex on edge
  - Insert edge splitting a face



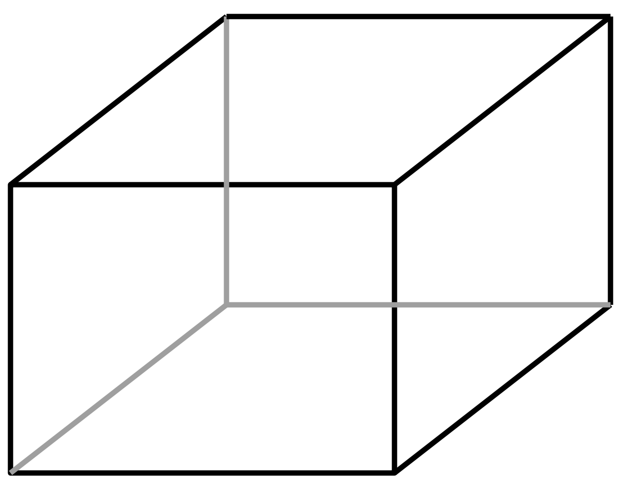
Euler Operations

- **Maintain topological integrity**
  - Insure Euler−Pontcare formula for 3D polyhedra

$$V - E + F - H = 2(M - G)$$

V = # Vertices   *E = # Edges*
F = # Faces   *H = # Hole loops*
G = # Handles   *M = # Objects*

Example:



V = 8
E = 12
F = 6
H = 0
G = 0
M = 1

$$8 - 12 + 6 = 2$$

Euler Operations

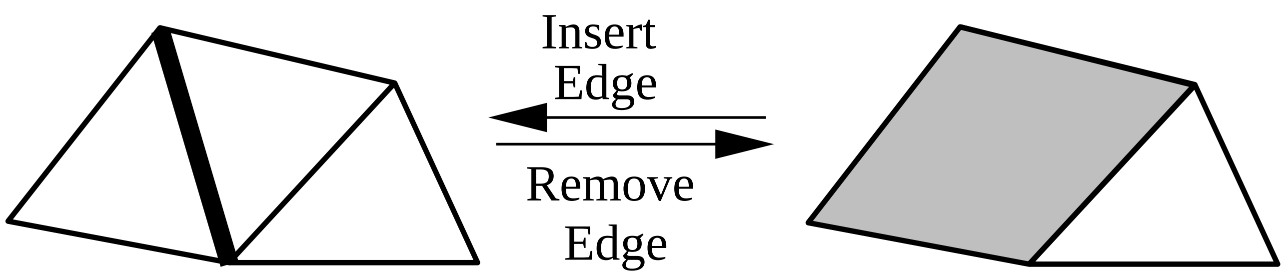
mbfv:	Makes object, face, and vertex
mev:	Creates vertex and edge
splite:	Splits edge into two by inserting vertex
mfe:	Makes face and edge
kbfv:	Removes object, face, and vertex
kev:	Removes vertex and edge
joine:	Joins two edges by removing vertex
kfe:	Removes face and edge
me−kh:	Makes edge, kills hole loop
me−kbf:	Makes edge, kills object and face
mhr−kf:	Makes hole loop and handle, kills face
mh−kbf:	Makes hole loop, kills object and face

$$V - E + F - H = 2(M - G)$$

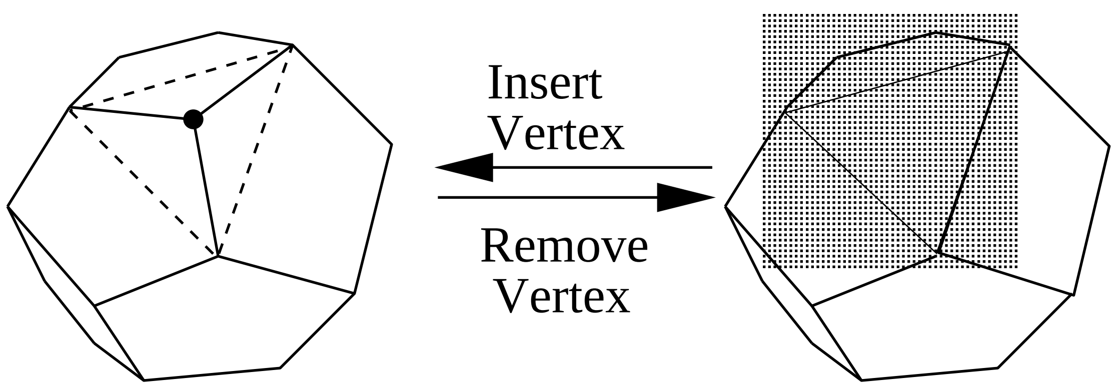
V = # Vertices   *E = # Edges*
F = # Faces   *H = # Hole loops*
G = # Handles   *M = # Objects*

Compound Operations

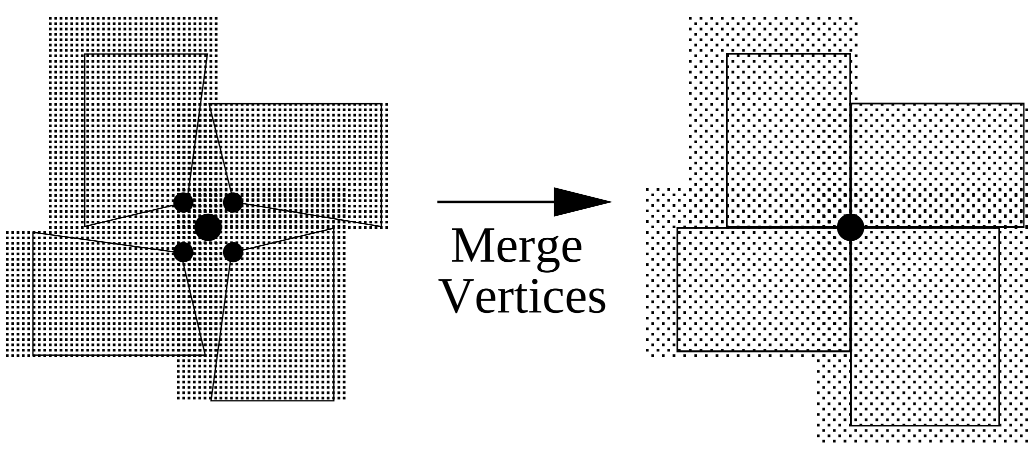
Insert edge: Inserts edge across face
Remove edge: Removes edge while joining two faces



Insert vertex on face: Splits face into many
Remove vertex: Creates face, or joins adjacent faces

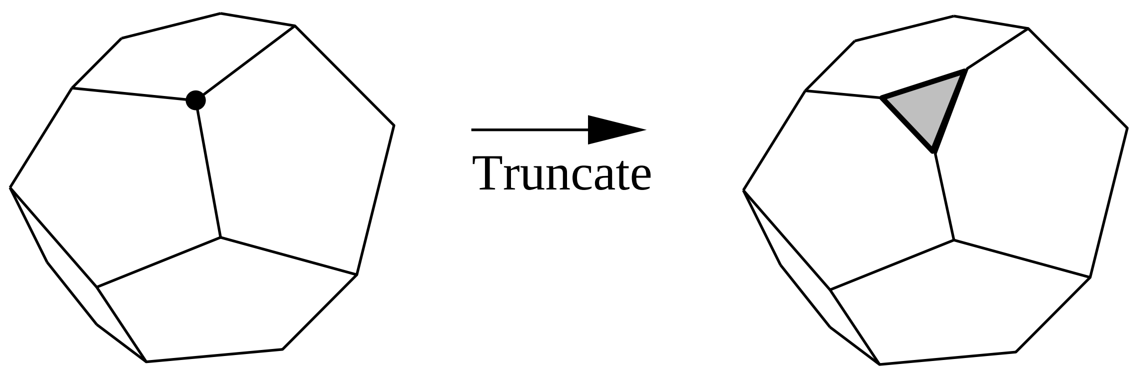


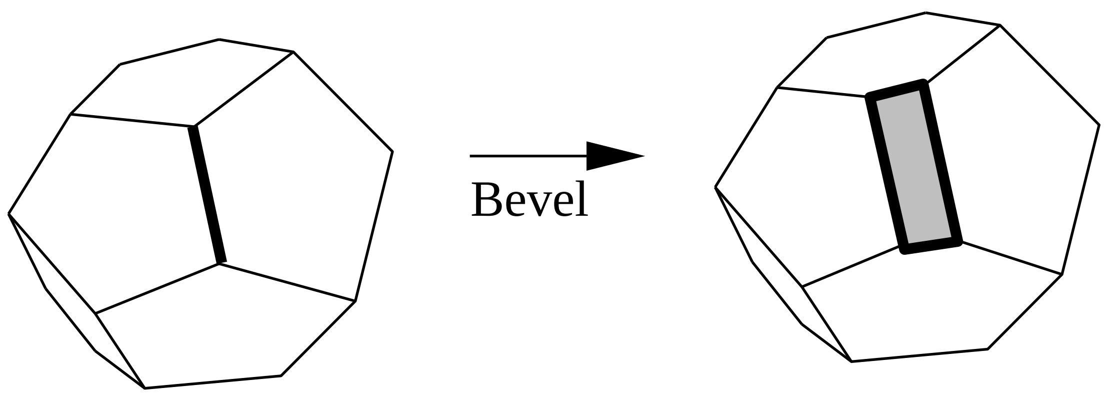
Merge vertices: Collapses n vertices into one



High−Level Operations

Truncate: Replaces vertex by face
Bevel: Replaces edge by face





Summary

- **Mesh Considerations**
  - Storage requirements
  - Computational efficiency
  - Ease of specification
- **Mesh Representations**
  - List of Faces
  - Triangle strips
  - Vertex tables
  - Adjacency lists
  - Winged−edge
- **Mesh Operations**
  - Traversal operations
  - Euler operations
  - Compound operations