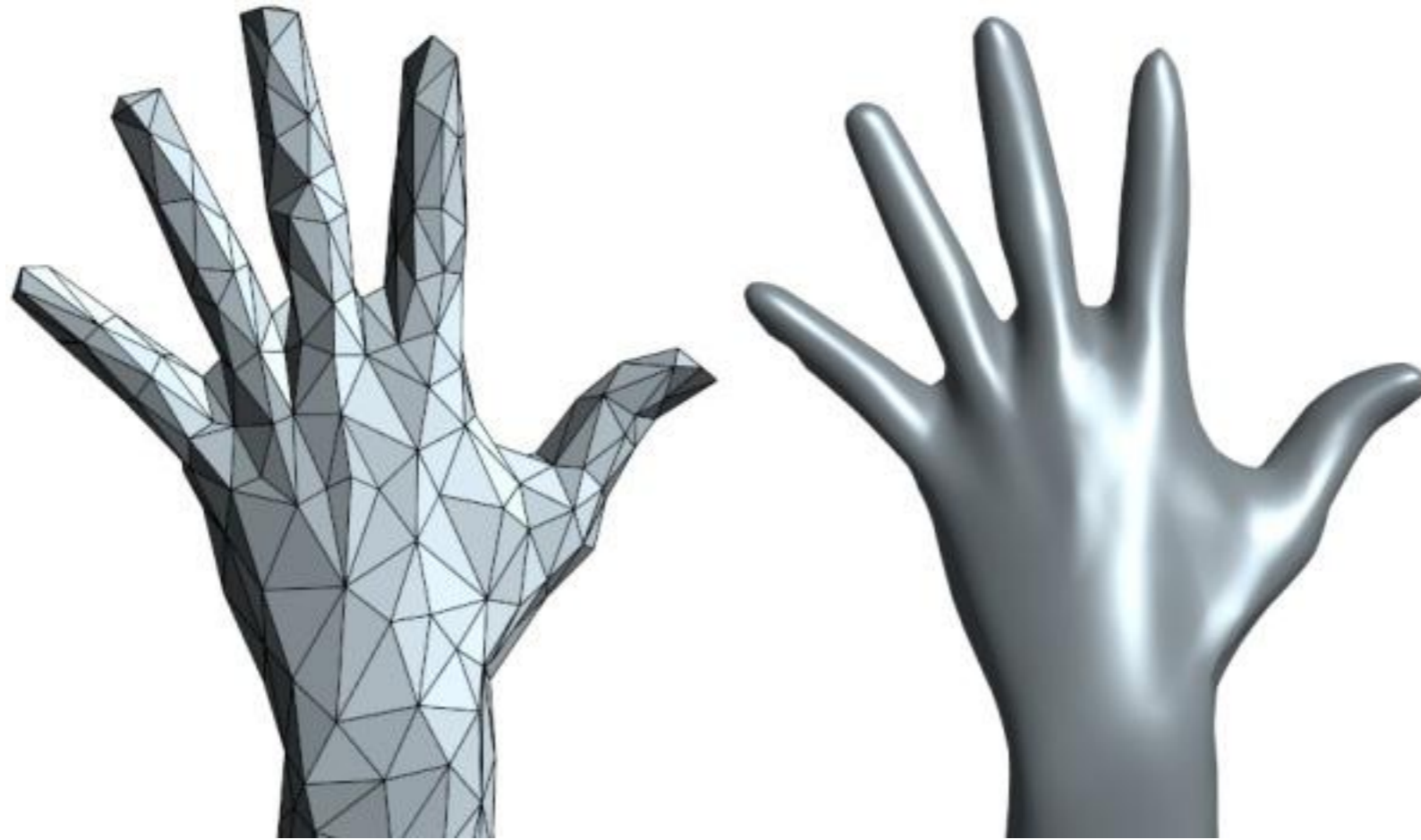


3D Surfaces



source:

http://iparla.labri.fr/publications/2007/BS07b/sketch_teaser.jpg



Mesh Representations & Subdivision Surfaces

- Tom Funkhouser
- Princeton University
- COS 426, Spring 2007

3D Object Representations



- Raw data
 - o Voxels
 - o Point cloud
 - o Range image
 - o Polygons
- Surfaces
 - o Mesh
 - o Subdivision
 - o Parametric
 - o Implicit
- Solids
 - o Octree
 - o BSP tree
 - o CSG
 - o Sweep
- High-level structures
 - o Scene graph
 - o Application specific

Examples

- Point Data – Digital Michelangelo Project: <http://graphics.stanford.edu/projects/mich/>
- Volume Data (voxels) – Stanford Volume Data Archive: <http://graphics.stanford.edu/data/voldata/>
- Polygon Data – Princeton Shape Benchmark: <http://shape.cs.princeton.edu/benchmark/>
- Modeling shapes with point-based geometry: <http://www.graphics.ethz.ch/Downloads/Publications/Papers/2003/Pau03/Pau03.pdf>
- Constructive Solid Geometry: <http://research.microsoft.com/en-us/um/people/bgquenter/> (see under Procedural Modeling)

3D Object Representations

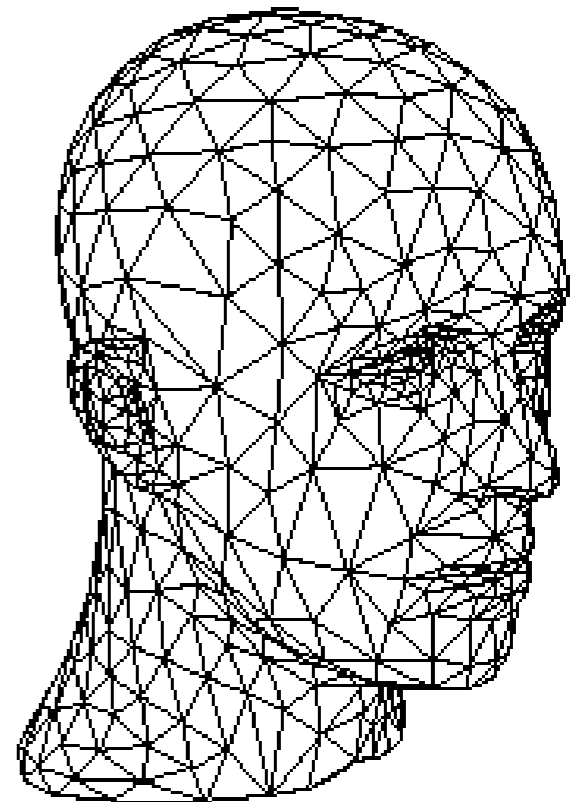


- Raw data
 - o Voxels
 - o Point cloud
 - o Range image
 - o Polygons
- Surfaces
 - o Mesh
 - o Subdivision
 - o Parametric
 - o Implicit
- Solids
 - o Octree
 - o BSP tree
 - o CSG
 - o Sweep
- High-level structures
 - o Scene graph
 - o Application specific

Polygon Meshes

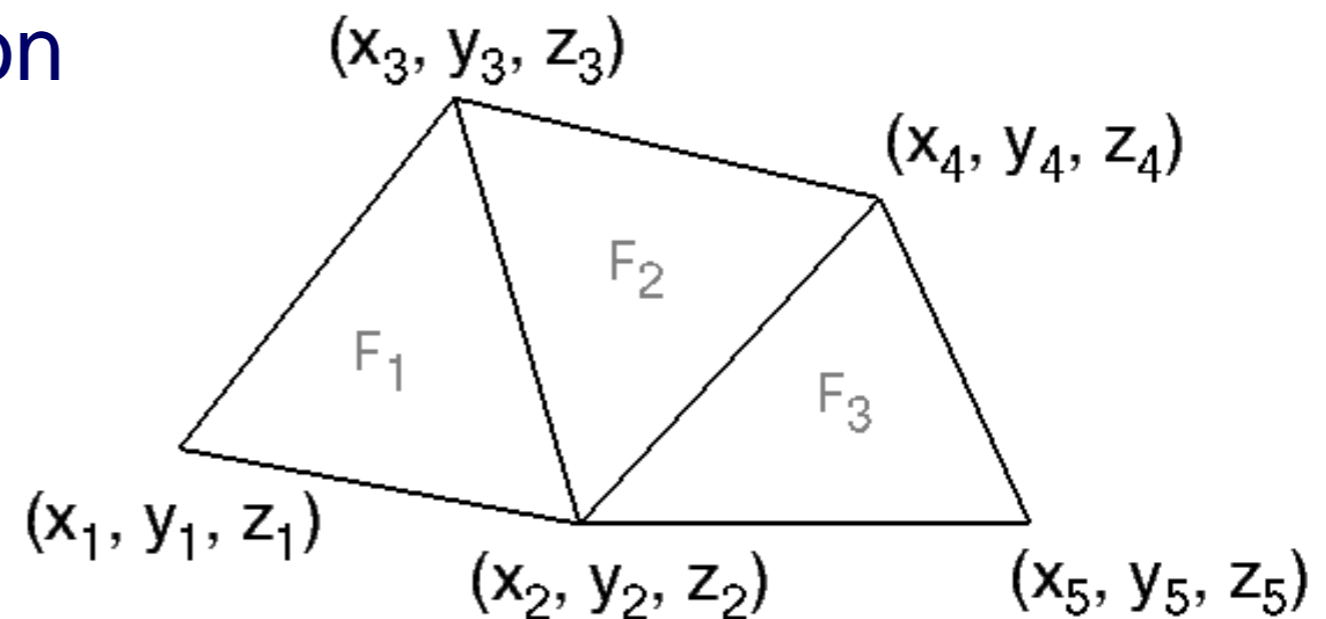
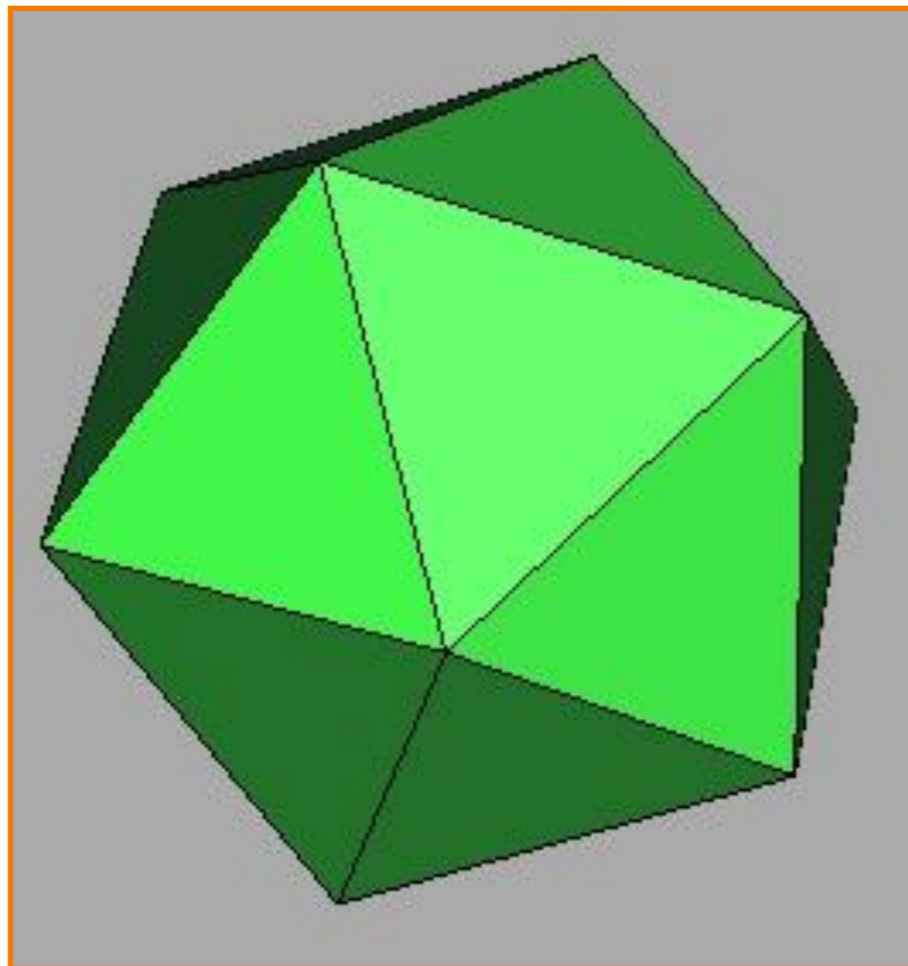


- How should we represent a mesh in a computer?
 - o Efficient traversal of topology
 - o Efficient use of memory
 - o Efficient updates
- Mesh Representations
 - o Independent faces
 - o Vertex and face tables
 - o Adjacency lists
 - o Winged-Edge
 - o Half-Edge
 - o etc.



Independent Faces

- Each face lists vertex coordinates
 - Redundant vertices
 - No adjacency information

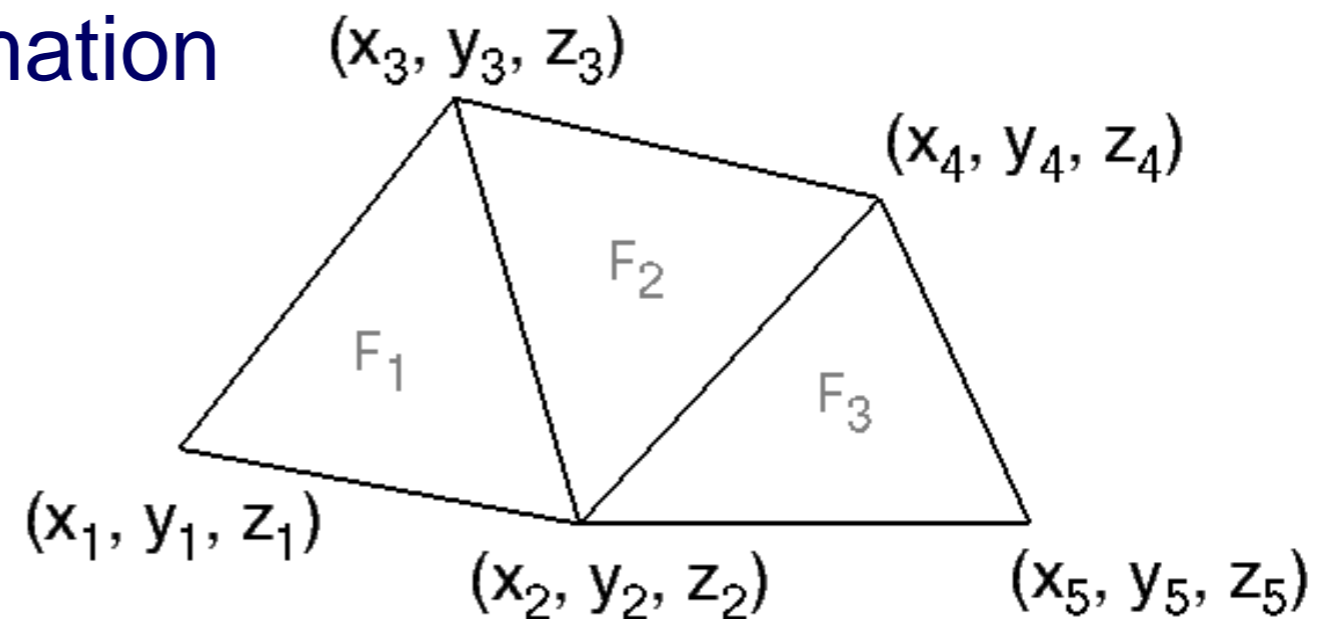
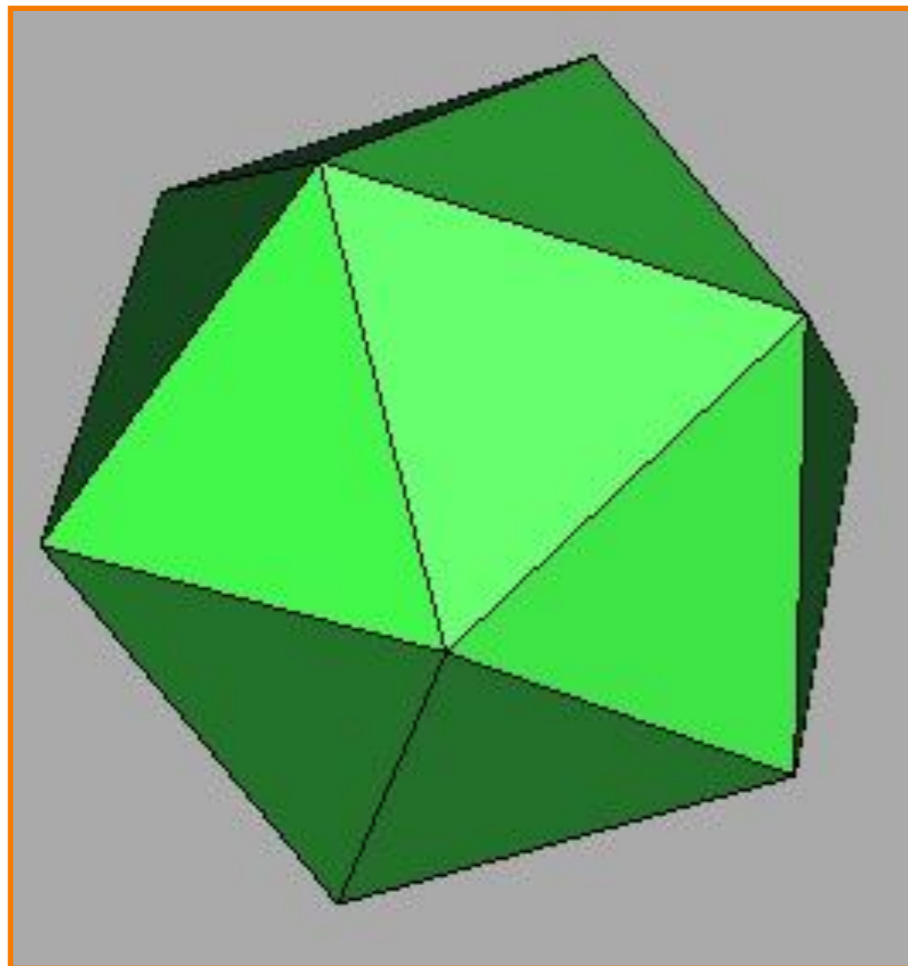


FACE TABLE

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)

Vertex and Face Tables

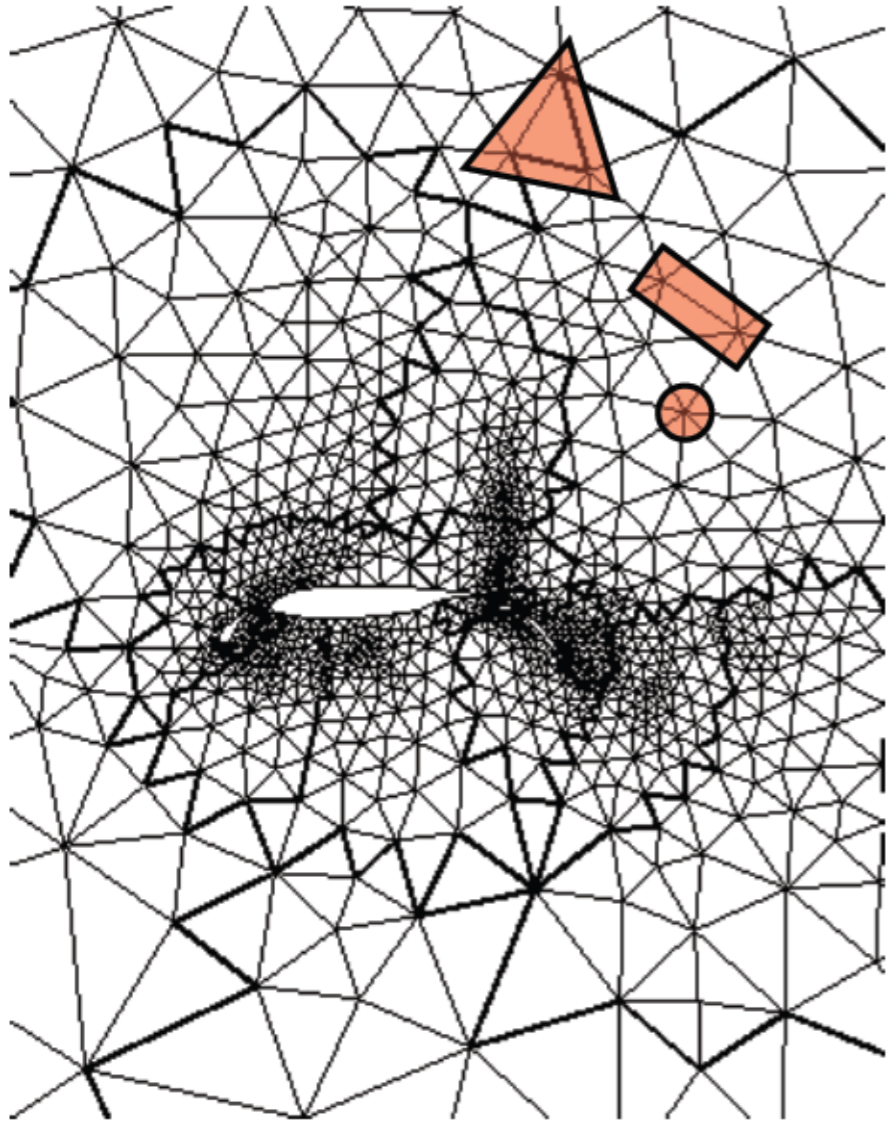
- Each face lists vertex references
 - Shared vertices
 - Still no adjacency information



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

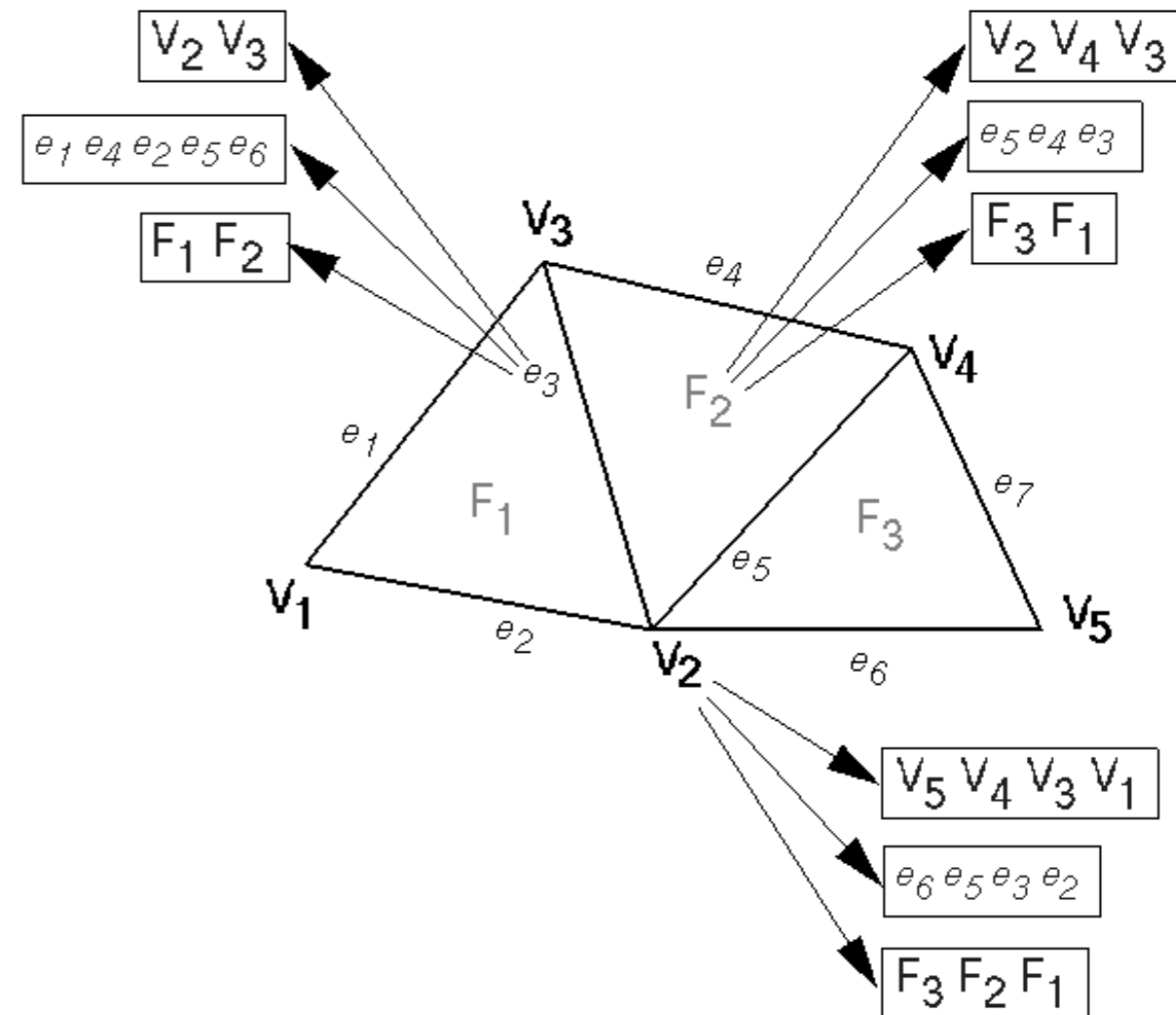
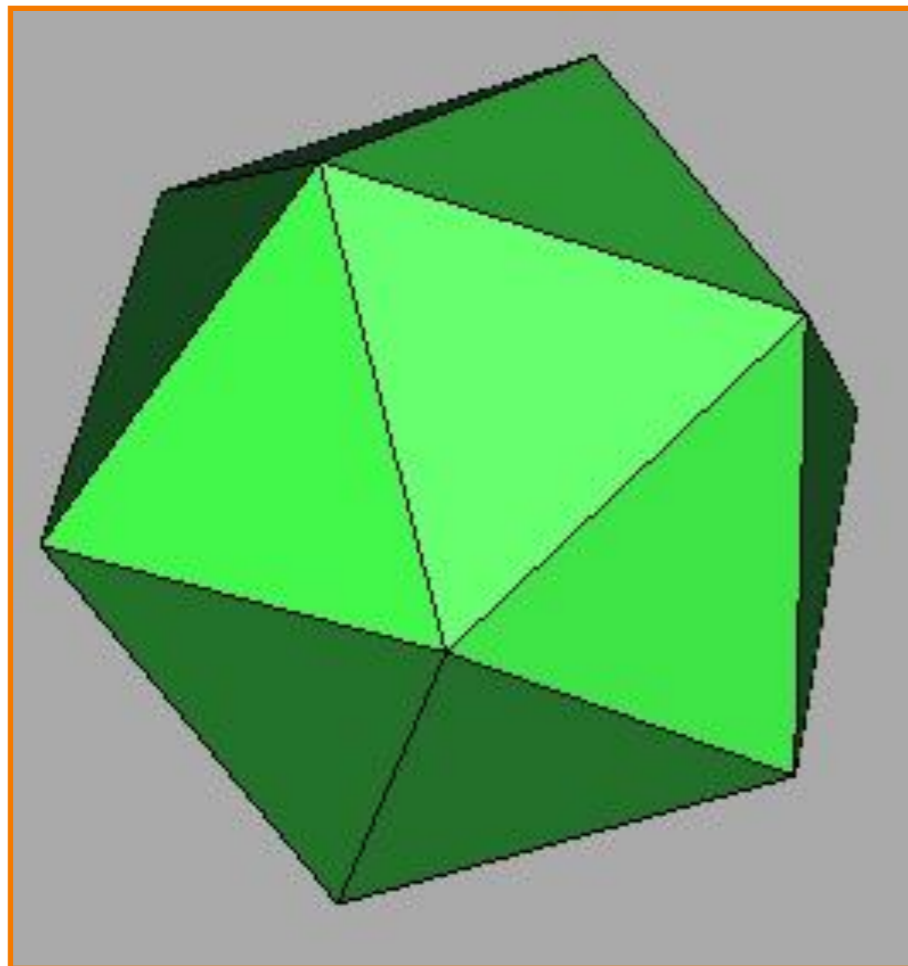
Possible Queries



- Which faces use this vertex?
- Which edges use this vertex?
- Which faces border this edge?
- Which edges border this face?
- Which faces are adjacent to this face?

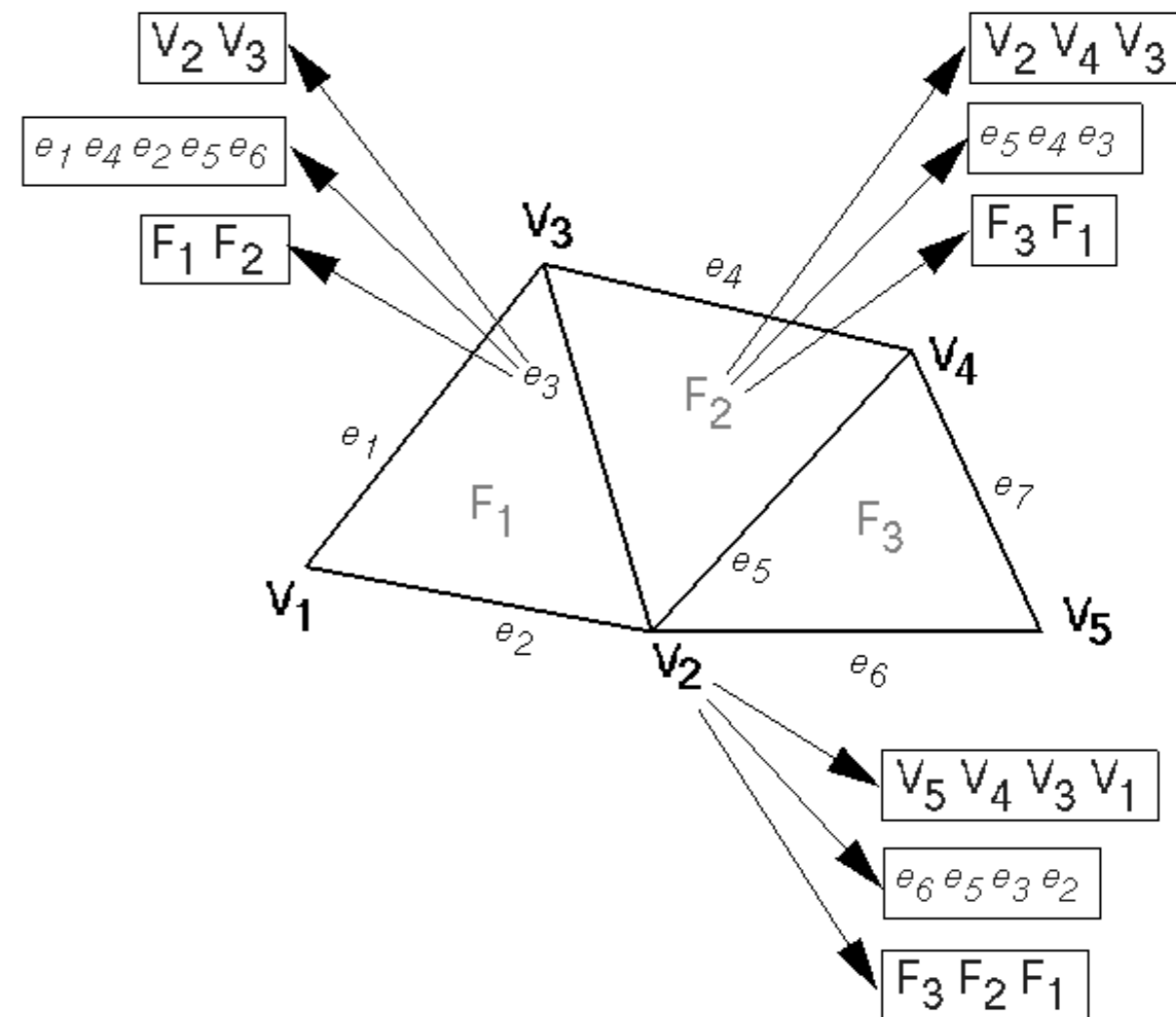
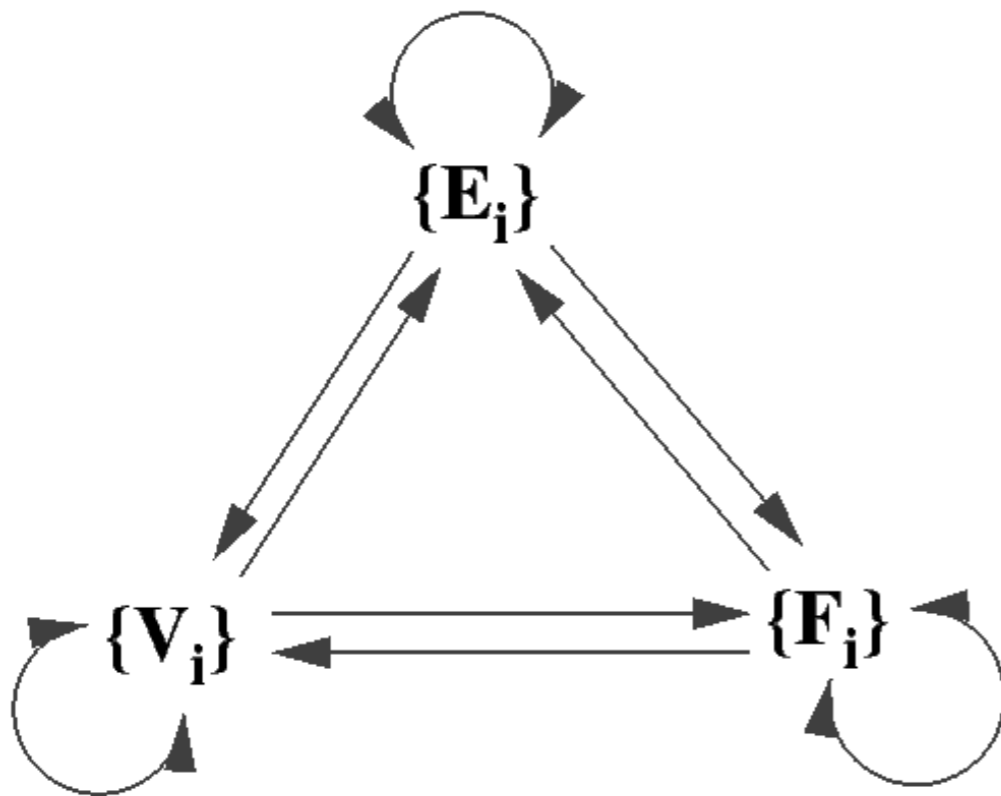
Adjacency Lists

- Store all vertex, edge, and face adjacencies
 - Efficient adjacency traversal
 - Extra storage



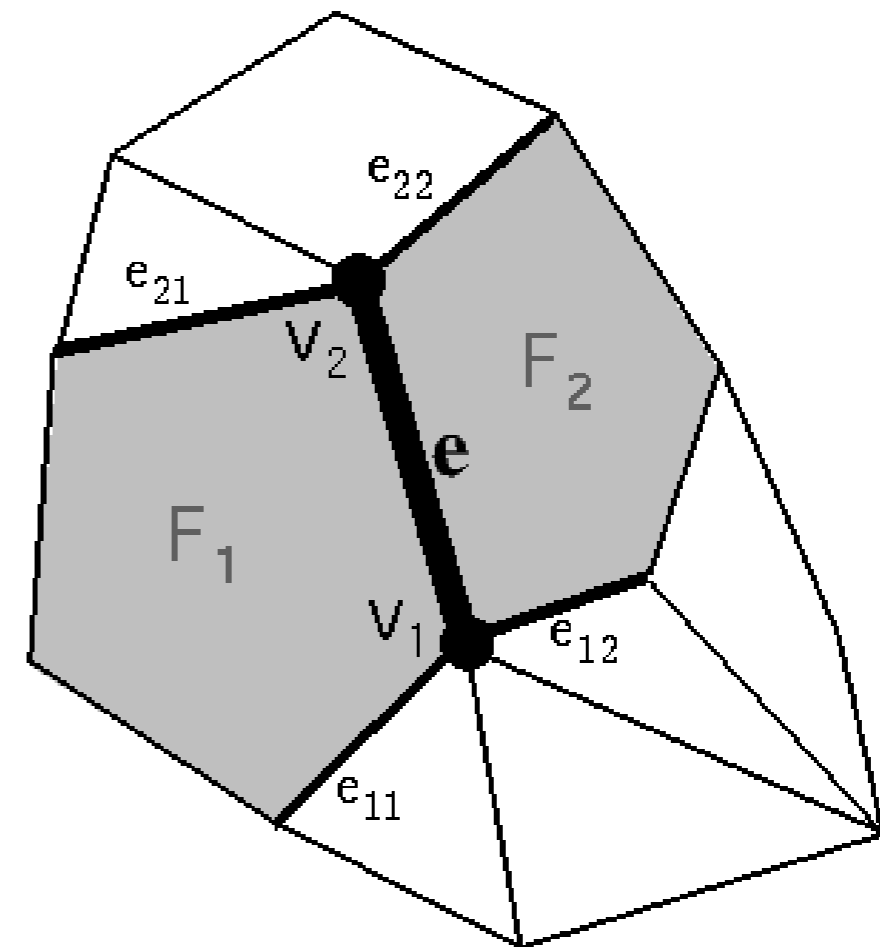
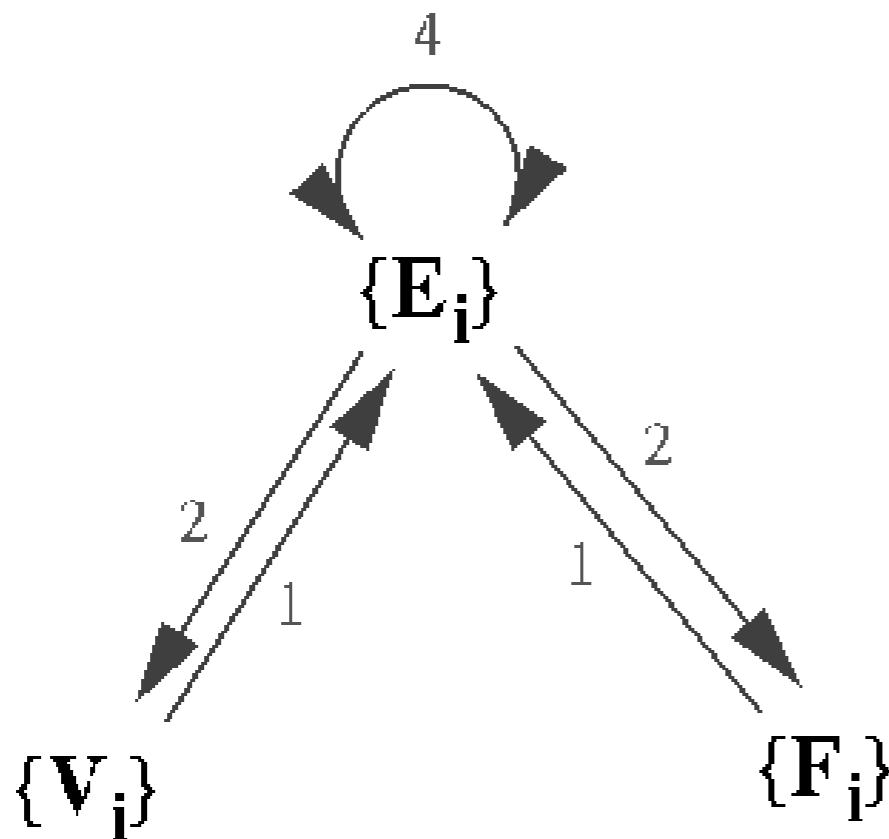
Partial Adjacency Lists

- Can we store only some adjacency relationships and derive others?



Winged Edge

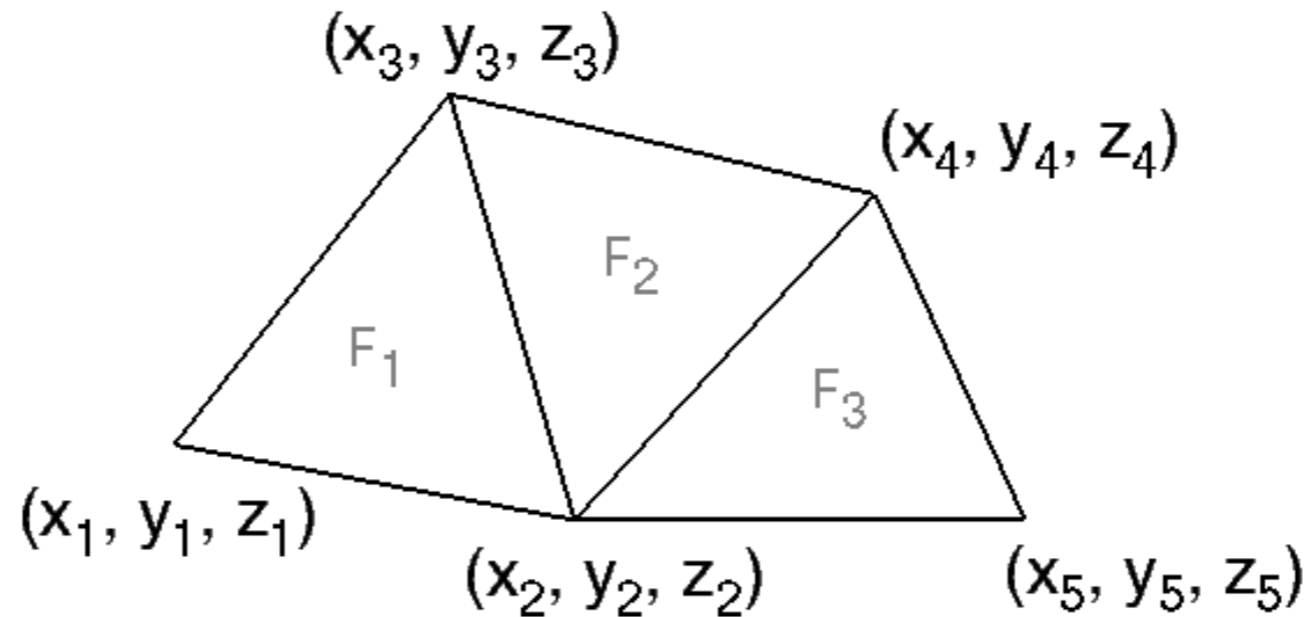
- Adjacency encoded in edges
 - All adjacencies in $O(1)$ time
 - Little extra storage (fixed records)
 - Arbitrary polygons



Winged Edge



- Example:



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

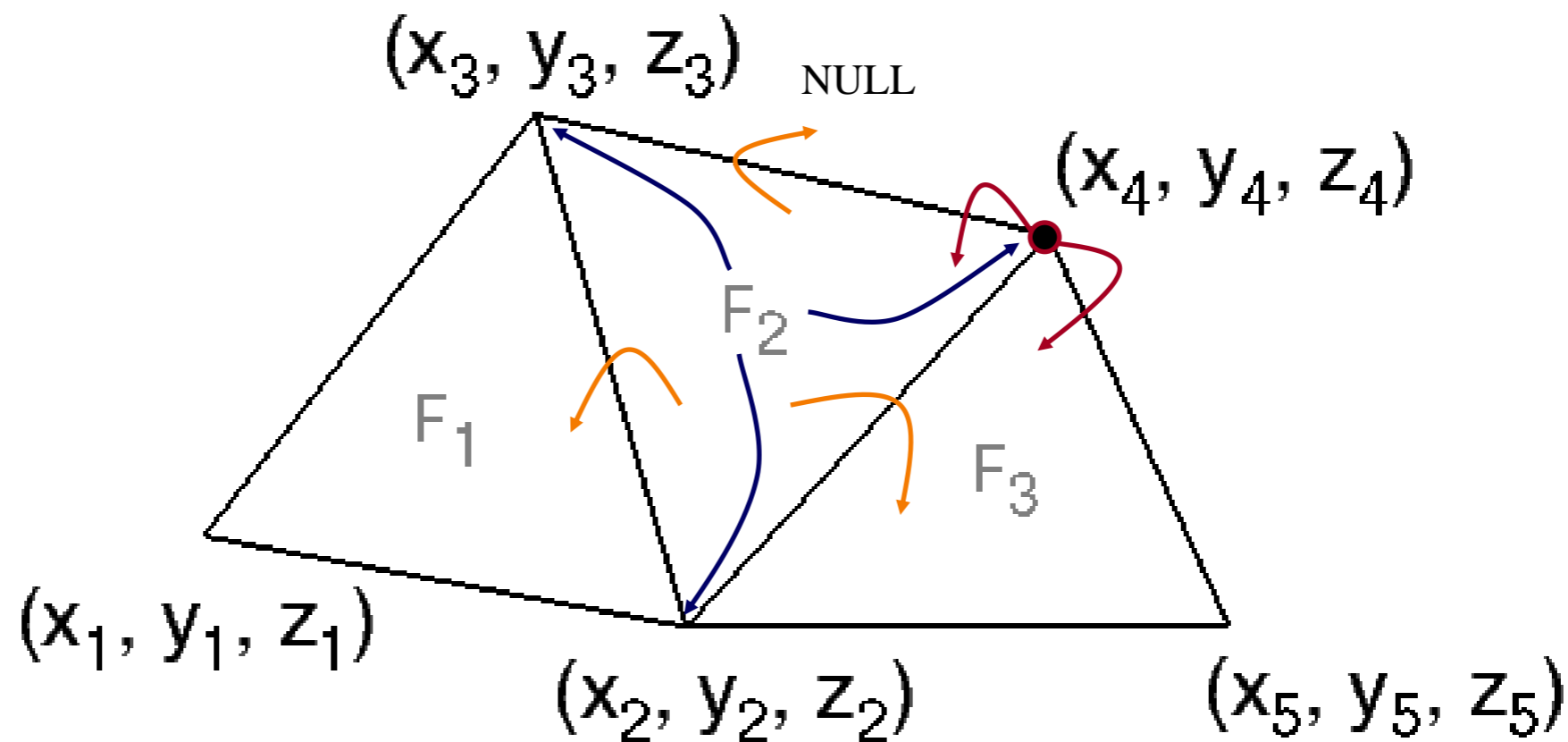
EDGE TABLE					11	12	21	22
e ₁	V ₁	V ₃	F ₁		e ₂	e ₂	e ₄	e ₃
e ₂	V ₁	V ₂	F ₁		e ₁	e ₁	e ₃	e ₆
e ₃	V ₂	V ₃	F ₁	F ₂	e ₂	e ₅	e ₁	e ₄
e ₄	V ₃	V ₄		F ₂	e ₁	e ₃	e ₇	e ₅
e ₅	V ₂	V ₄	F ₂	F ₃	e ₃	e ₆	e ₄	e ₇
e ₆	V ₂	V ₅	F ₃		e ₅	e ₂	e ₇	e ₇
e ₇	V ₄	V ₅		F ₃	e ₄	e ₅	e ₆	e ₆

FACE TABLE	
F ₁	e ₁
F ₂	e ₃
F ₃	e ₅



Simple Triangle Mesh

- Do not store edges at all
 - All faces have 3 vertices and 3 neighbors
- Store adjacency in vertices and faces
 - For each face: 3 vertices and 3 faces
 - For each vertex: N faces



3D Object Representations

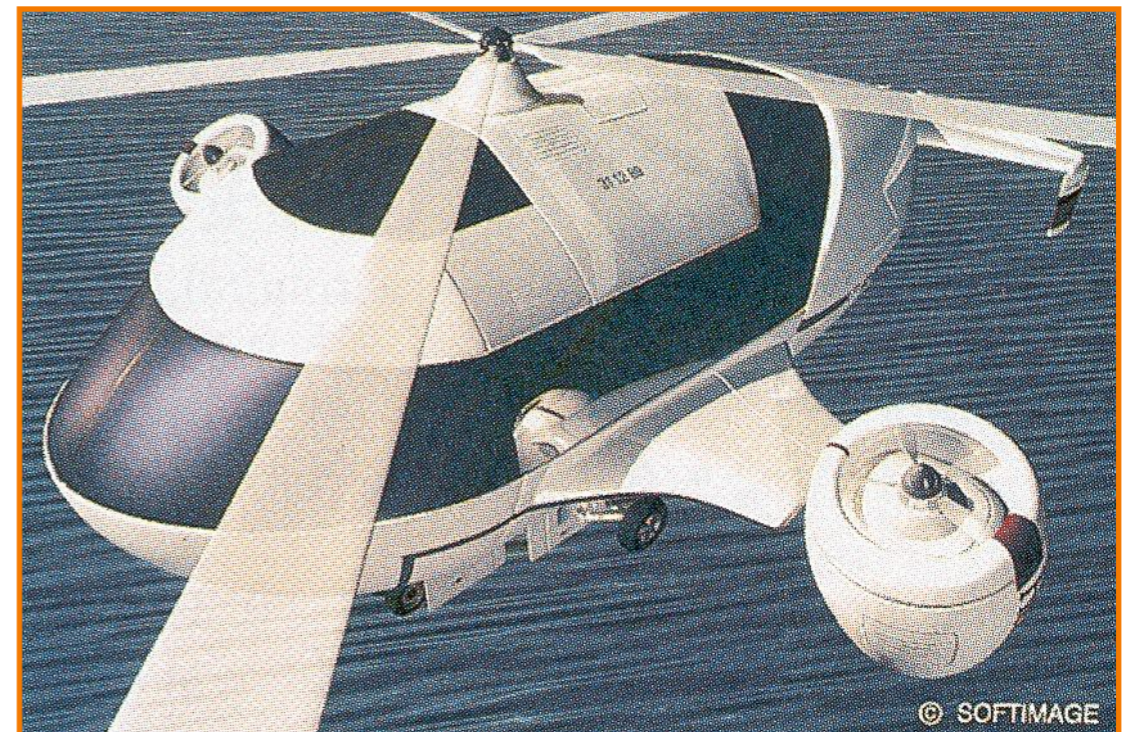


- Raw data
 - o Voxels
 - o Point cloud
 - o Range image
 - o Polygons
- Surfaces
 - o Mesh
 - o Subdivision
 - o Parametric
 - o Implicit
- Solids
 - o Octree
 - o BSP tree
 - o CSG
 - o Sweep
- High-level structures
 - o Scene graph
 - o Application specific

Surfaces



- What makes a good surface representation?
 - o Accurate
 - o Concise
 - o Intuitive specification
 - o Local support
 - o Affine invariant
 - o Arbitrary topology
 - **Guaranteed continuity**
 - o Natural parameterization
 - o Efficient display
 - o Efficient intersections



H&B Figure 10.46

Subdivision



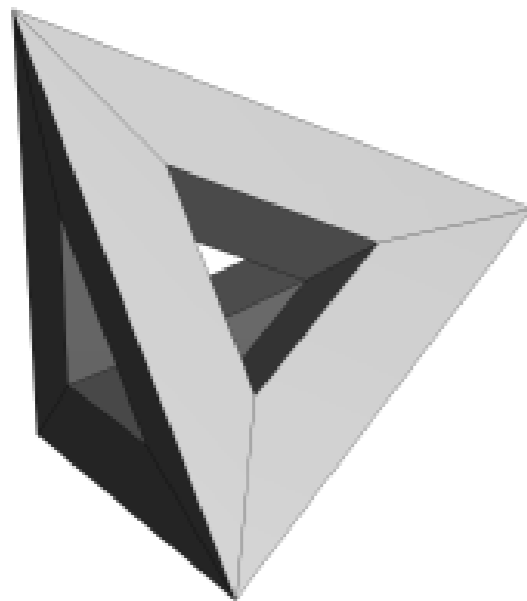
- How do you make a smooth curve?



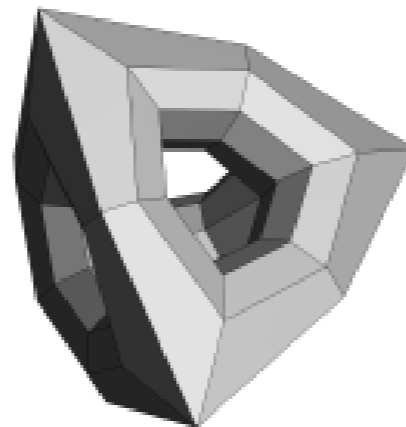
Subdivision Surfaces



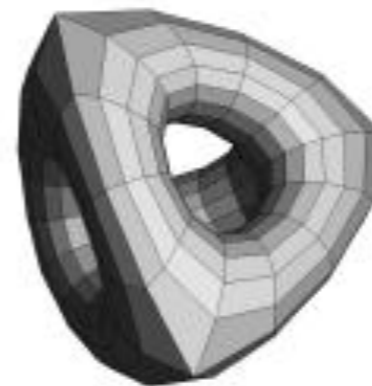
- Coarse mesh & subdivision rule
 - Define smooth surface as limit of sequence of refinements



(a)



(b)



(c)

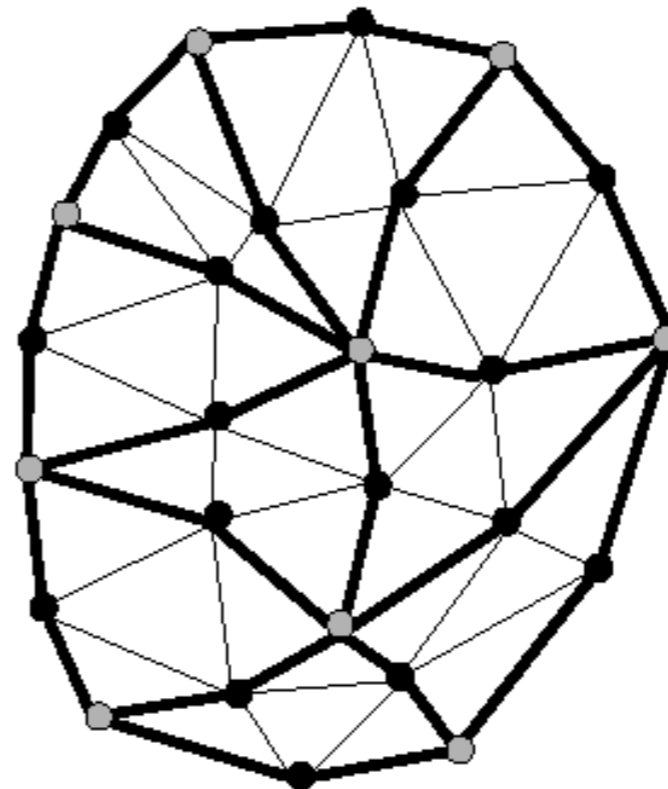
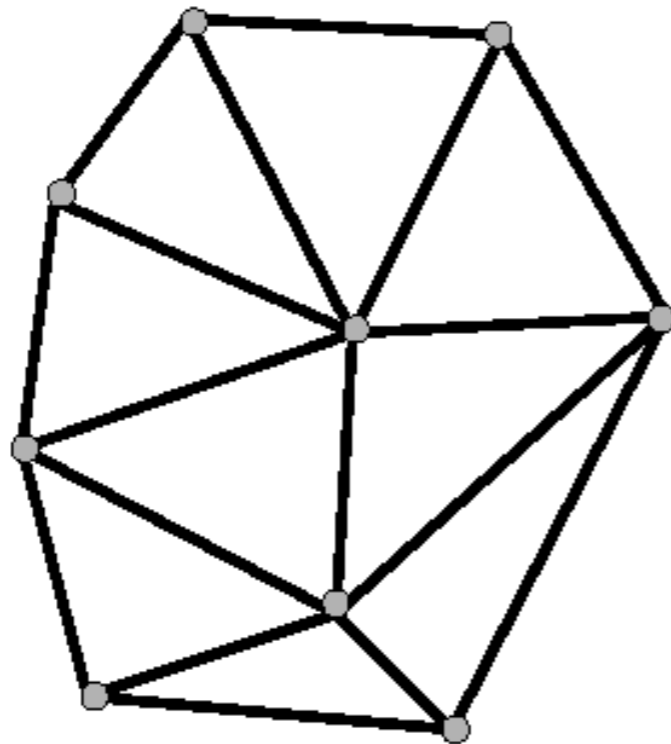


(d)

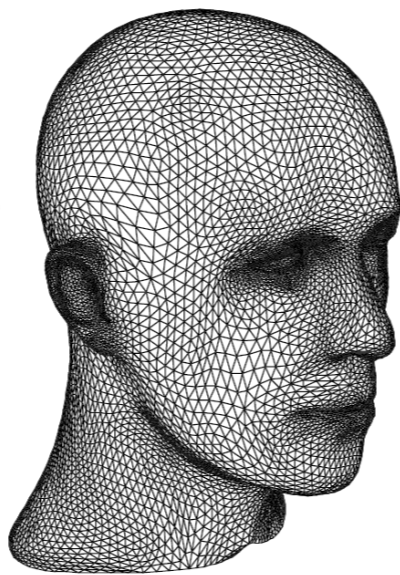
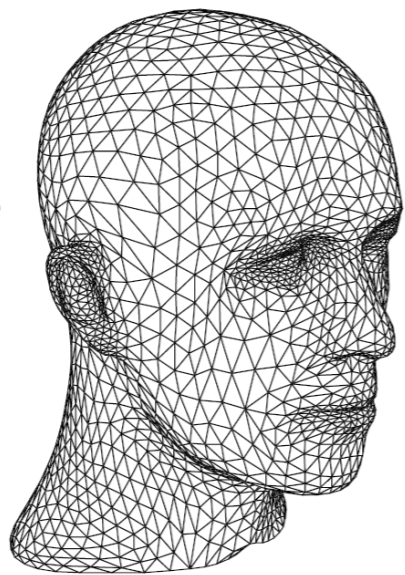
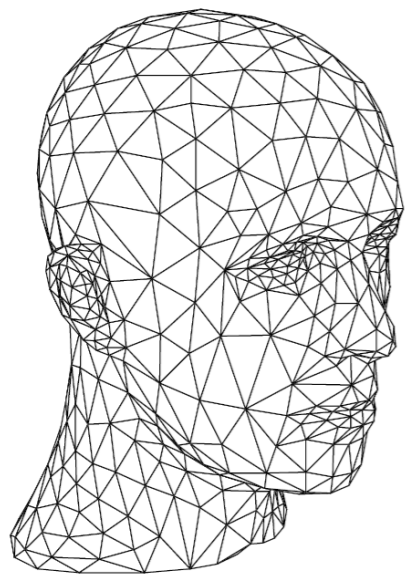
Key Questions



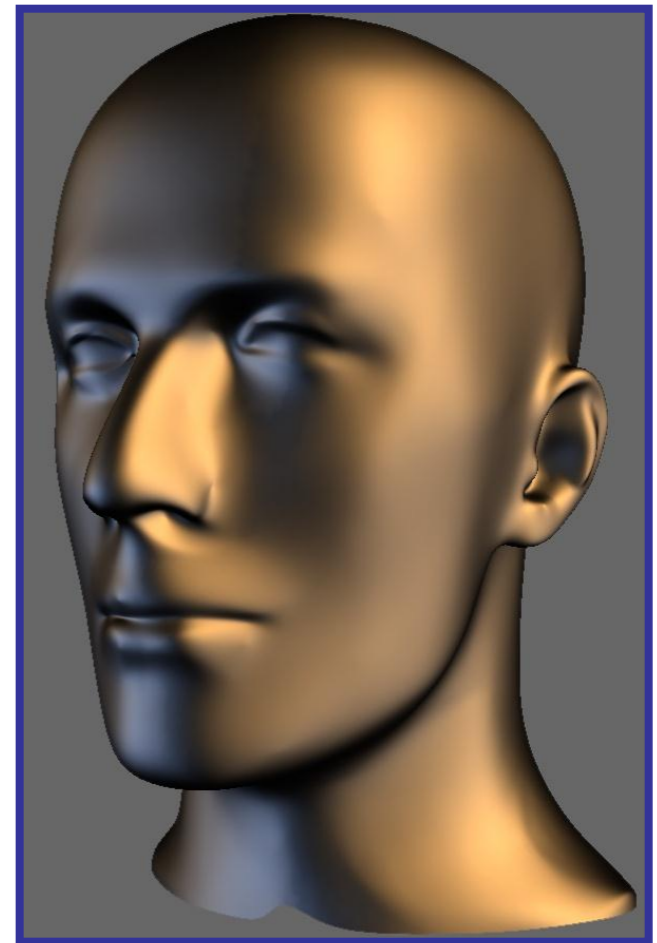
- How refine mesh?
 - Aim for properties like smoothness
- How store mesh?
 - Aim for efficiency for implementing subdivision rules



Subdivision Surfaces – A 3D example



...



Applications: Computer Graphics Animation



Subdivision Surfaces



- Advantages:
 - o Simple method for describing complex surfaces
 - o Relatively easy to implement
 - o Arbitrary topology
 - o Local support
 - o Guaranteed continuity
 - o Multiresolution
- Difficulties:
 - o Intuitive specification
 - o Parameterization
 - o Intersections



Summary



Feature	Polygonal Mesh	Subdivision Surface
Accurate	No	Yes
Concise	No	Yes
Intuitive specification	No	No
Local support	Yes	Yes
Affine invariant	Yes	Yes
Arbitrary topology	Yes	Yes
Guaranteed continuity	No	Yes
Natural parameterization	No	No
Efficient display	Yes	Yes
Efficient intersections	No	No