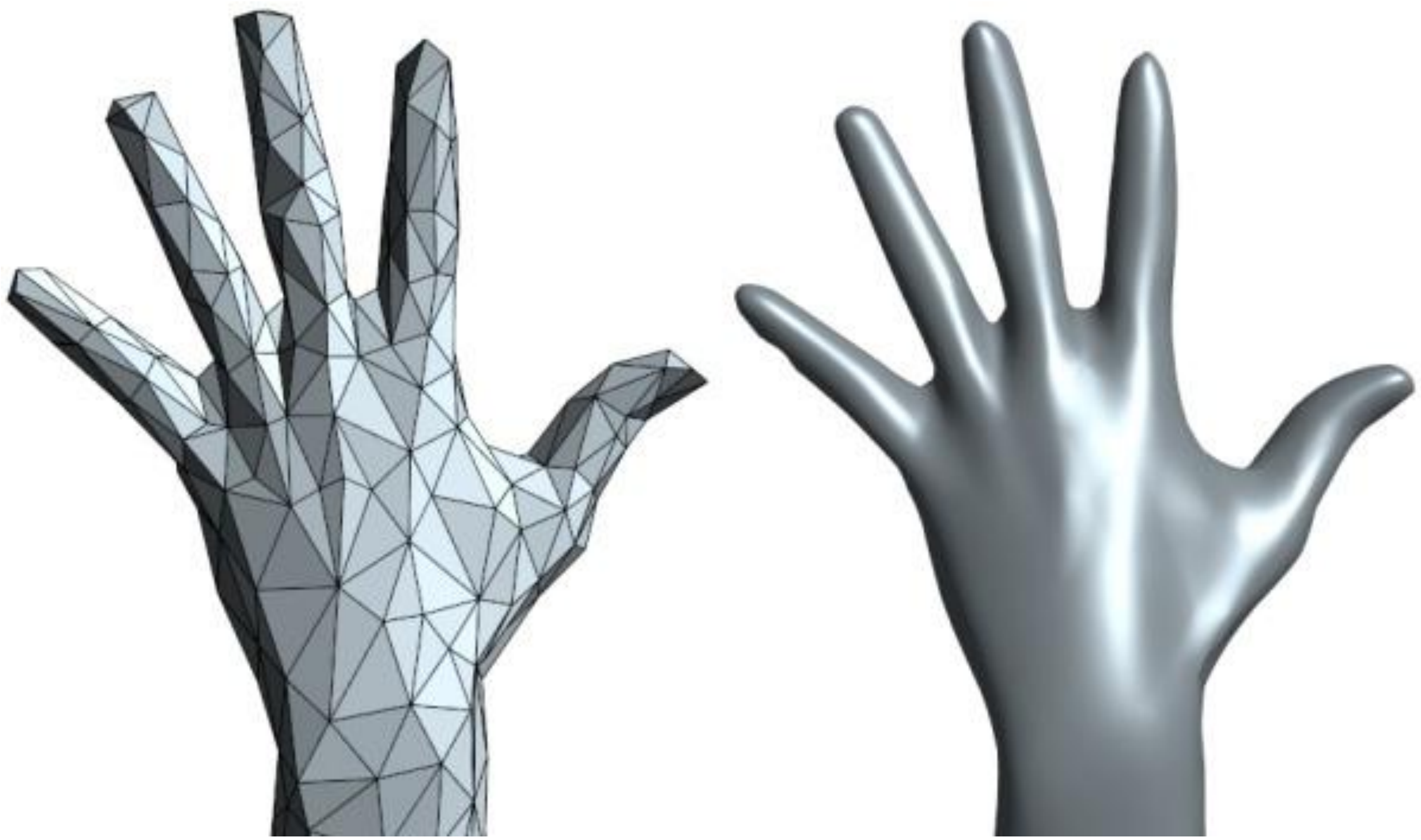
3D Surfaces



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

Mesh Representations & Subdivision Surfaces

- Tom Funkhouser
- Princeton University

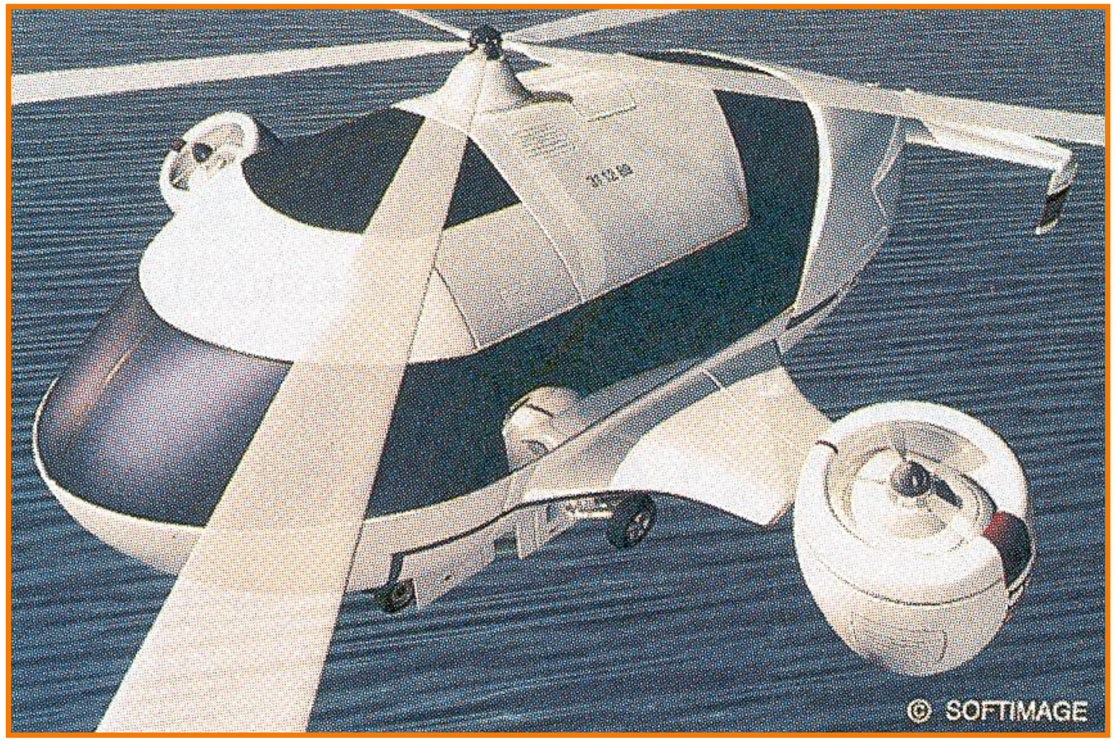
- COS 426, Spring 2007

3D Object Representations

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

Surfaces

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



H&B Figure 10.46

2D Scalar Field

- z = f(x,y)

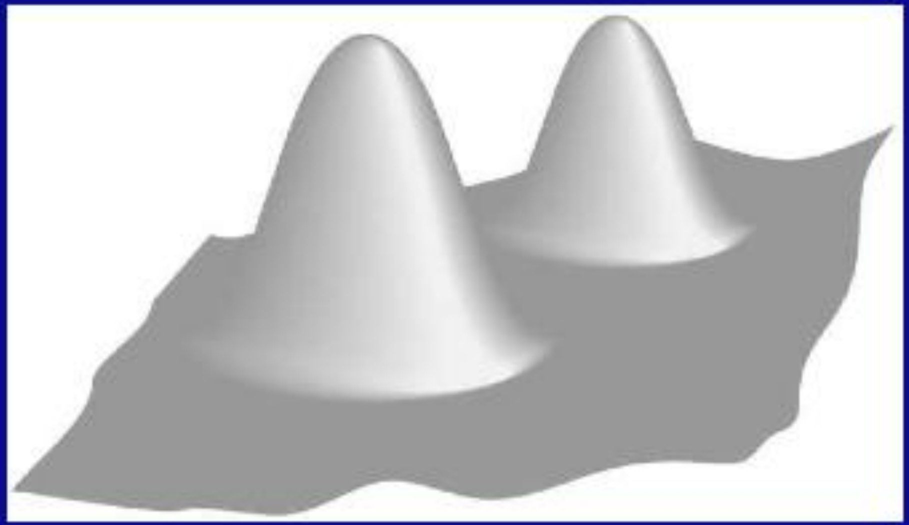
$$f(x,y) = \begin{cases} 1 - x^2 - y^2, & \text{if } x^2 + y^2 < 1 \\ 0 & \end{cases}$$

How do you visualize this function?

Height Field

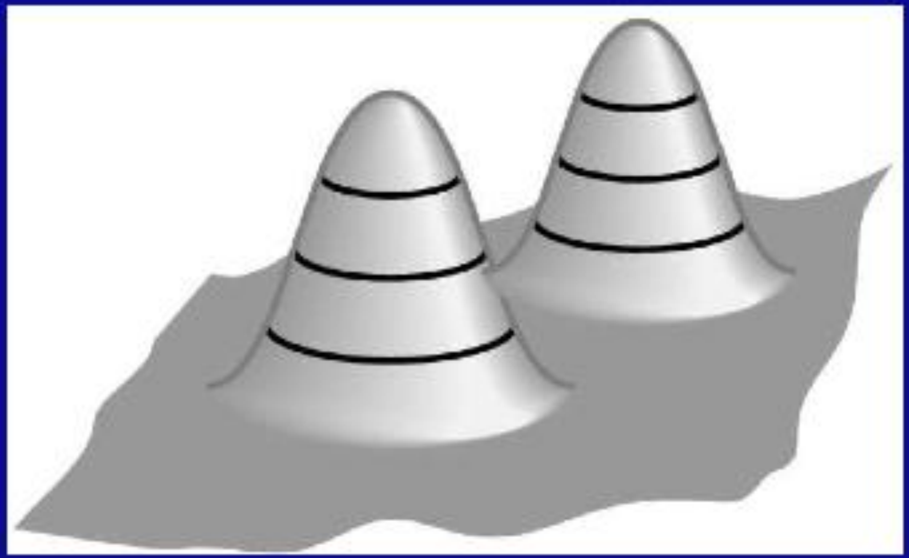
- Visualizing an explicit function

$$z = f(x,y)$$



- Adding contour curves

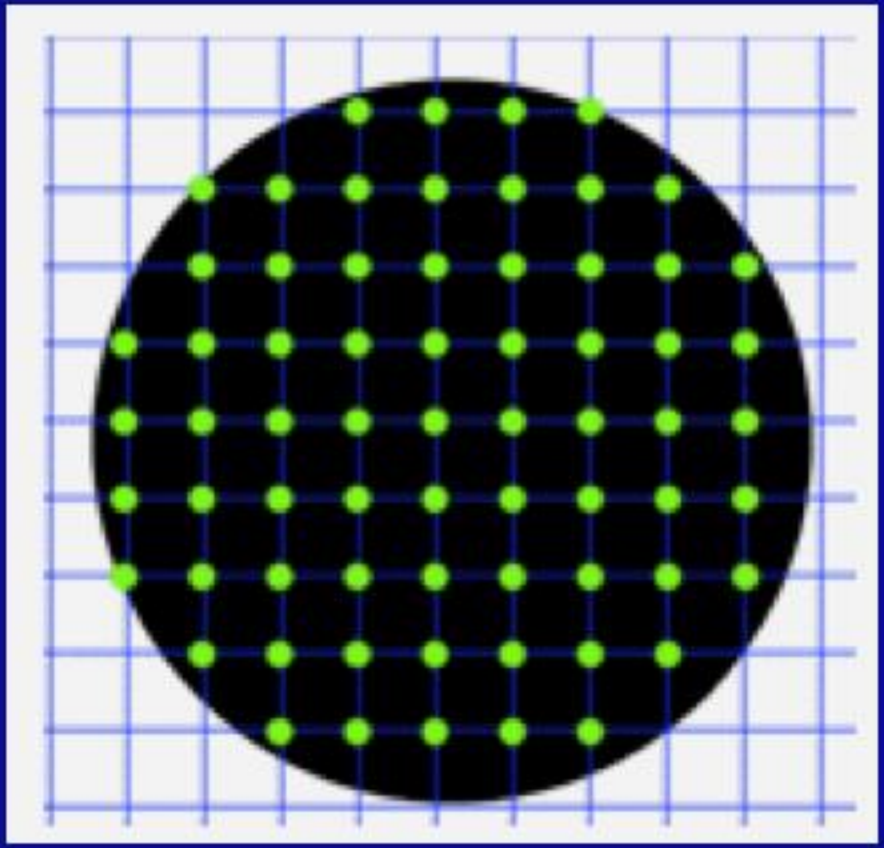
$$f(x,y) = c$$



Implicit → Explicit 2D
(Marching Squares Algorithm)

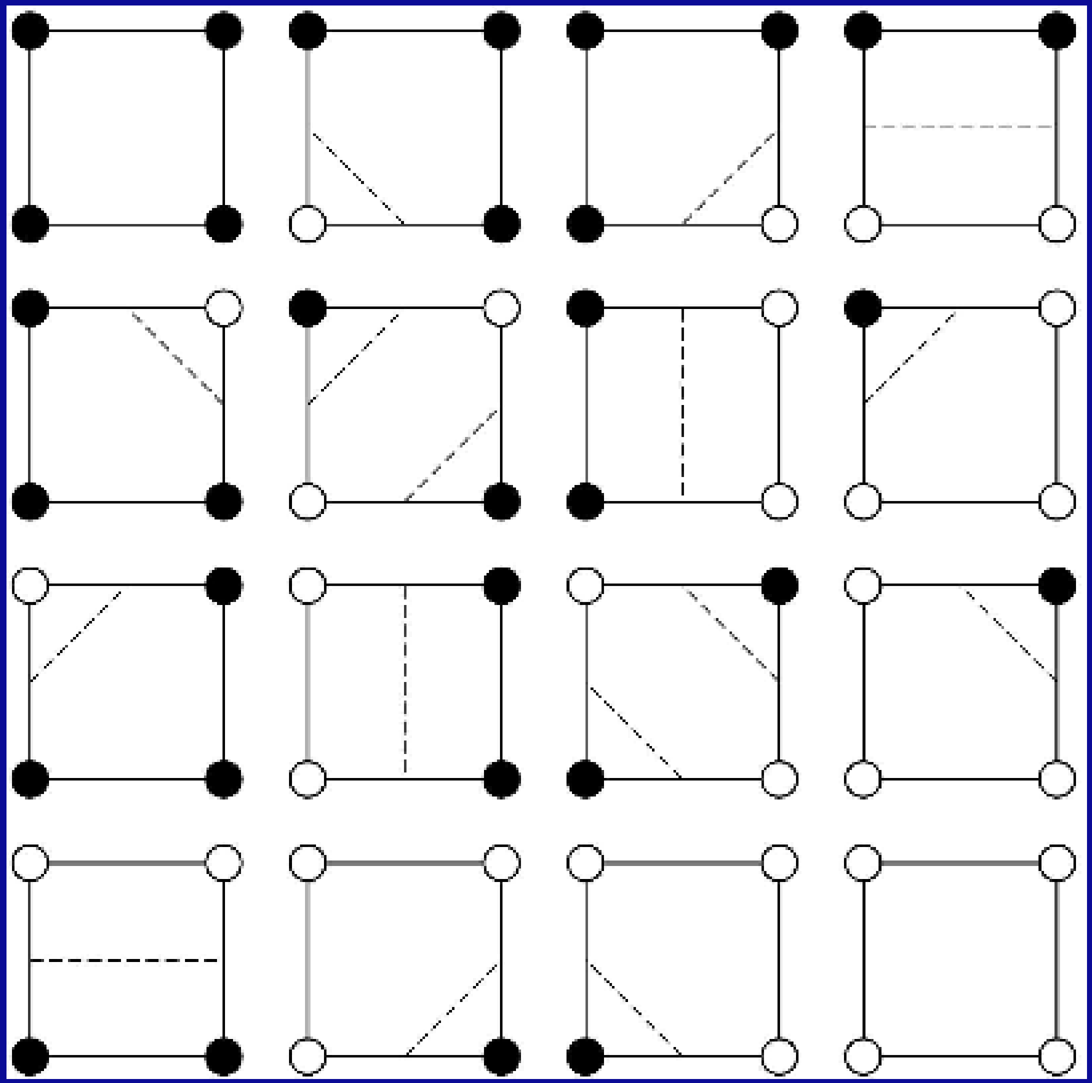
Marching Squares

- Sample function f at every grid point x_i, y_j
- For every point $f_{i\,j} = f(x_i, y_j)$ either $f_{i\,j} \leq c$ or $f_{i\,j} > c$

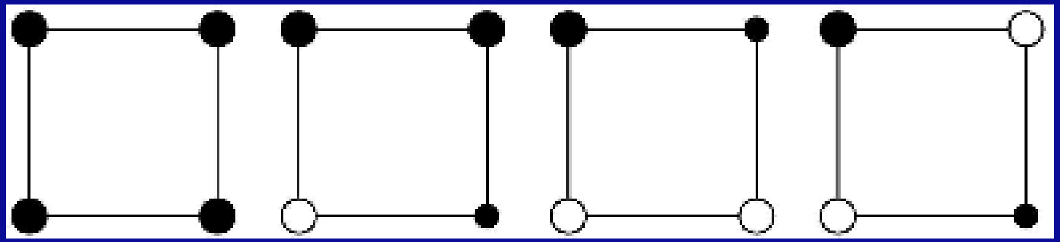


Cases for Vertex Labels

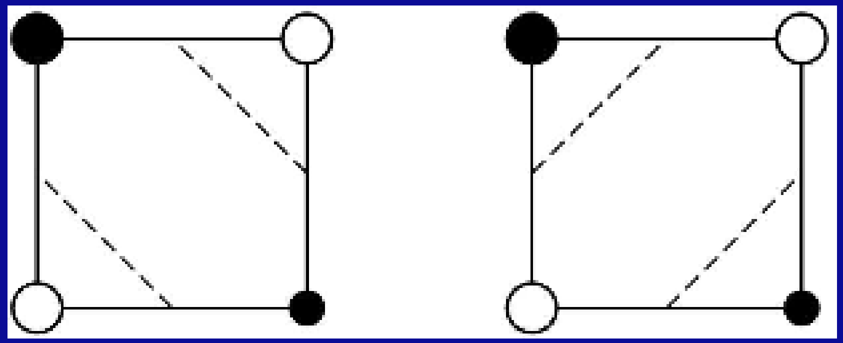
16 cases for vertex labels



4 unique mod. symmetries

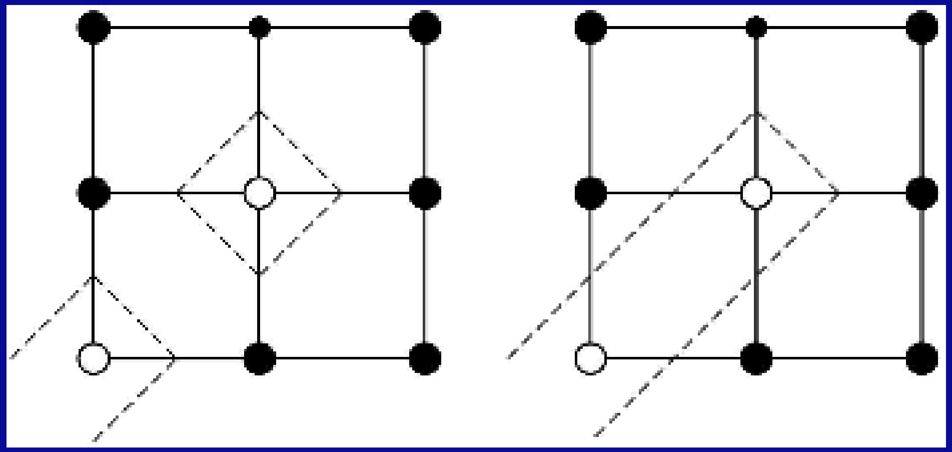


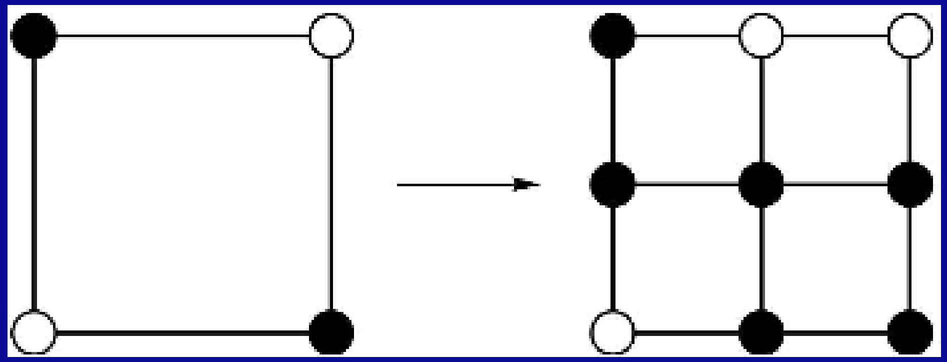
Ambiguities of Labelings



Ambiguous labels

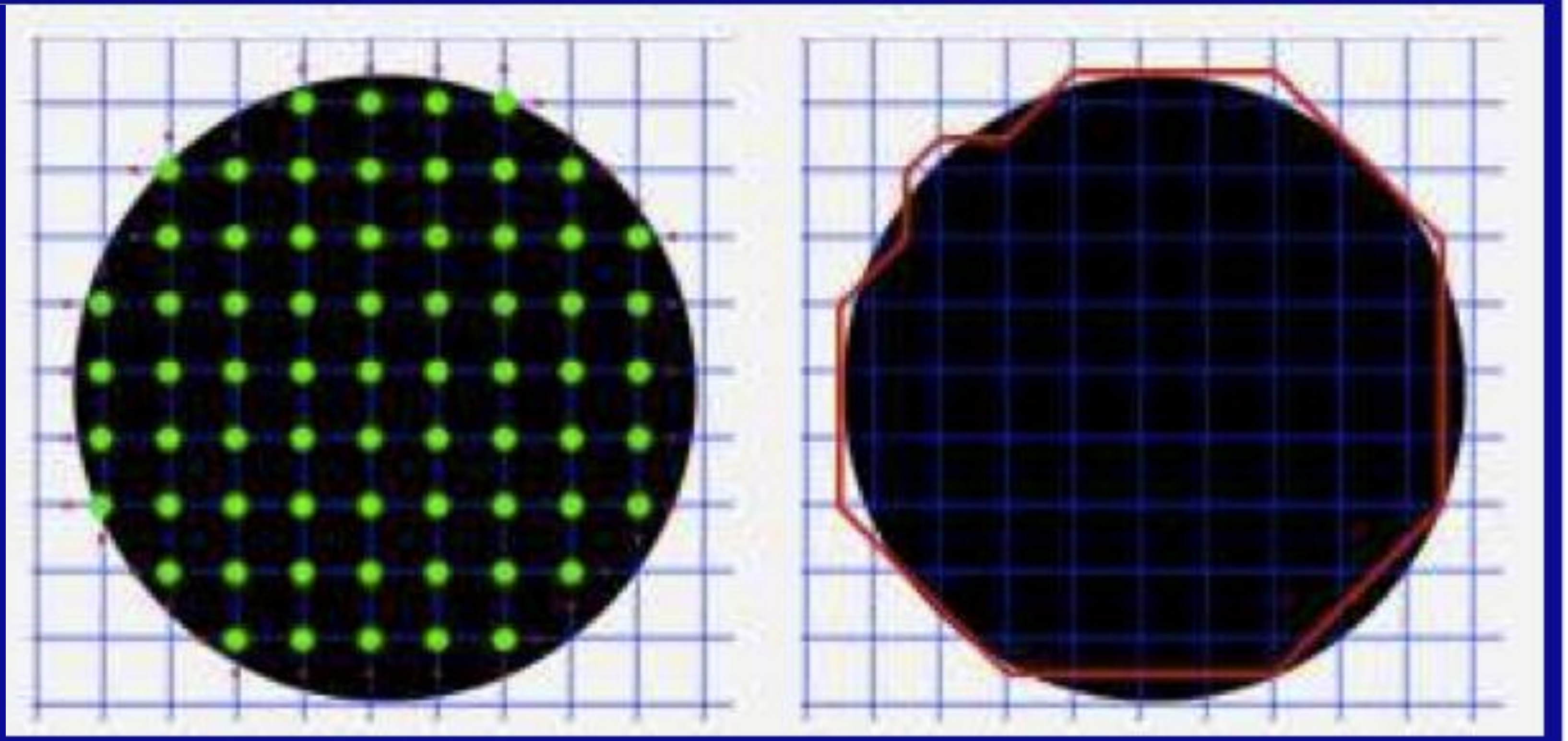
Different resulting contours





Resolution by subdivision (where possible)

Marching Squares Examples



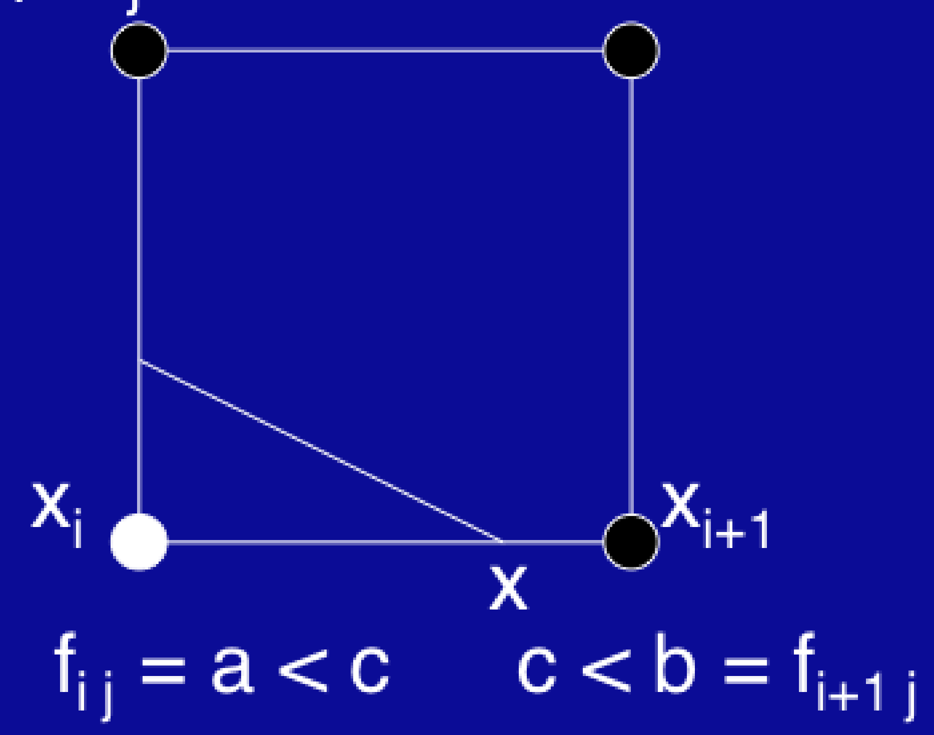
Can you do better?

Interpolating Intersections

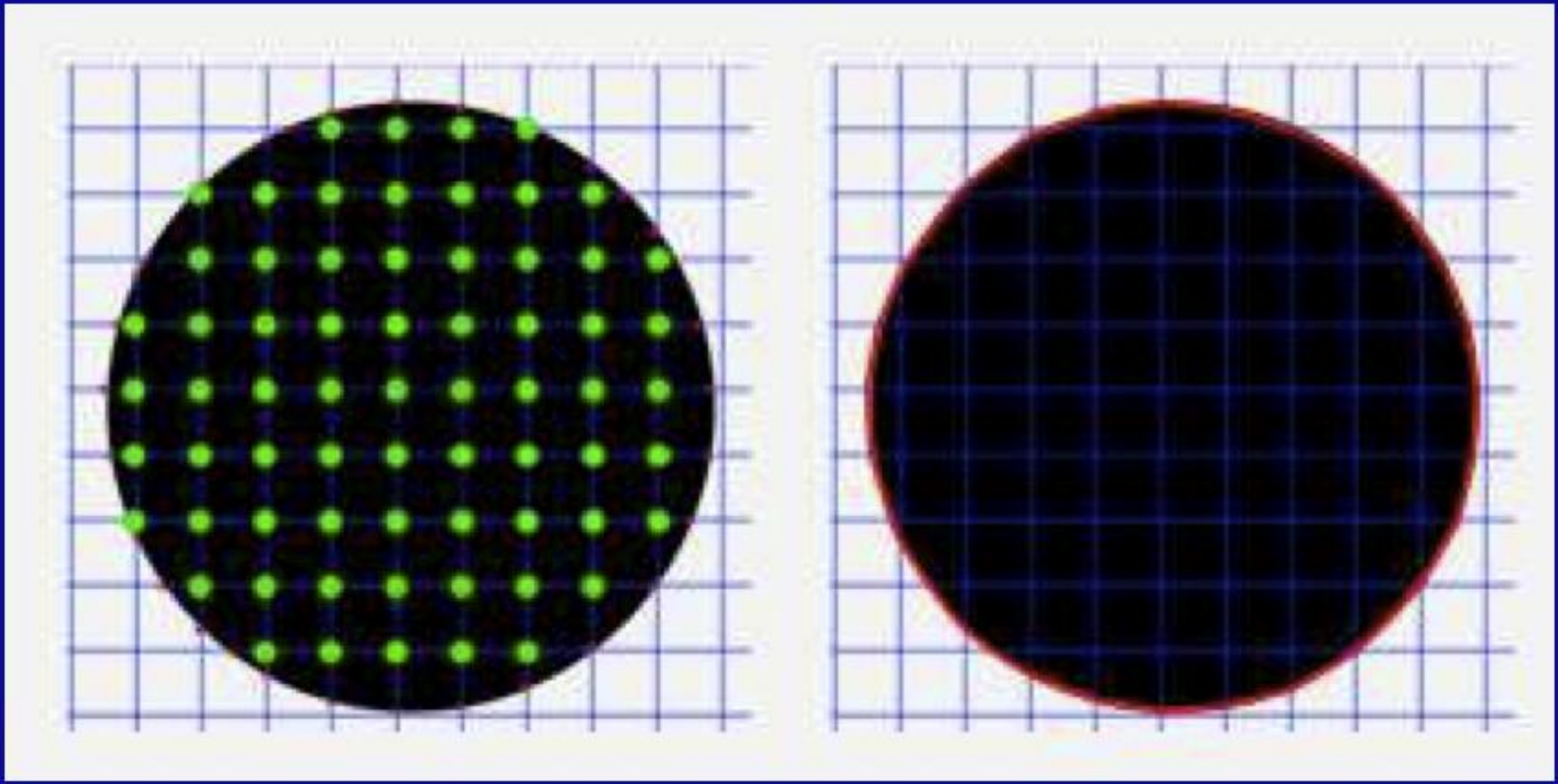
- Approximate intersection
  - Midpoint between x_i, x_{i+1} and y_j, y_{j+1}
  - Better: interpolate
- If $f_{i\,j} = a$ is closer to c than $b = f_{i+1\,j}$ then intersection is closer to (x_i, y_j):

$$\frac{x - x_i}{x_{i+1} - x} = \frac{c - a}{b - c}$$

- Analogous calculation for y direction

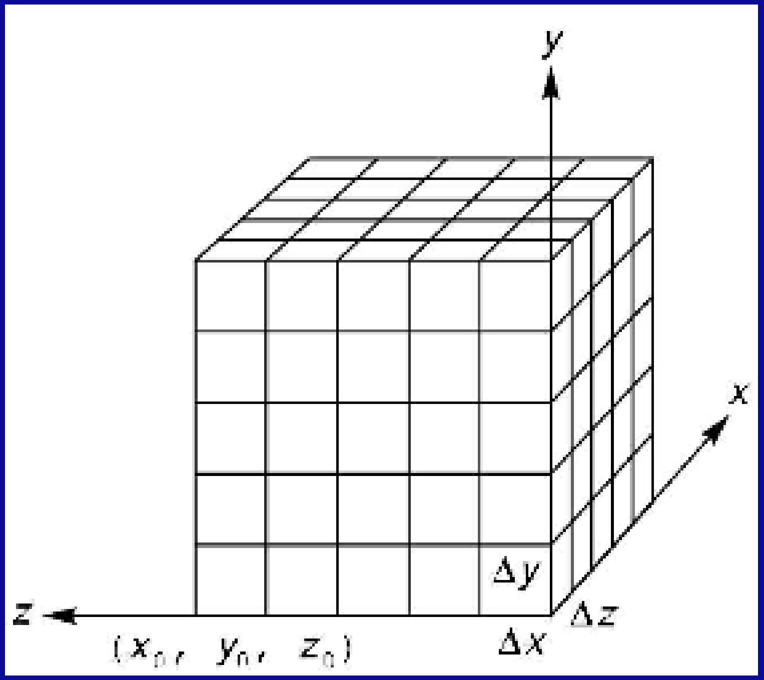


Marching Squares Examples



Implicit → Explicit 3D
(Marching Cubes Algorithm)

3D Scalar Fields



- Volumetric data sets
- Example: tissue density
- Assume again regularly sampled

$$
\begin{aligned}
x_i &= x_0 + i\Delta x \\
y_j &= y_0 + j\Delta y \\
z_k &= z_0 + k\Delta z
\end{aligned}
$$

- Represent as voxels
- Two rendering methods
  - –Isosurface rendering
  - –Direct volume rendering

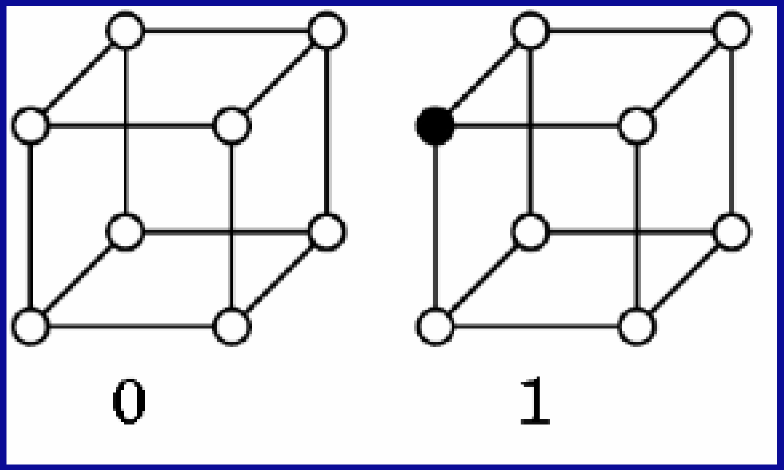
Isosurfaces

- Generalize contour curves to 3D

- Isosurface given by $f(x,y,z) = c$
  - $f(x, y, z) < c$ inside
  - $f(x, y, z) = c$ surface
  - $f(x, y, z) > c$ outside

Marching Cubes

- Display technique for isosurfaces
- 3D version of marching squares
- How many possible cases?

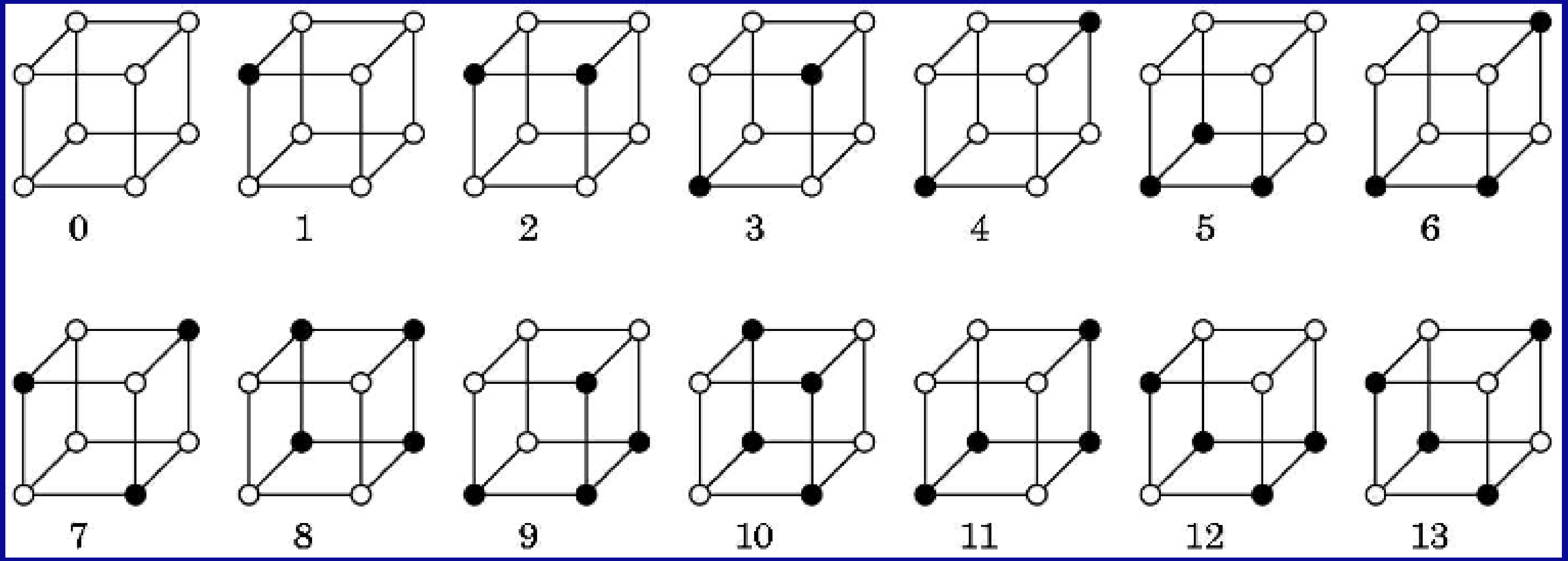


...

$2^8 = 256$

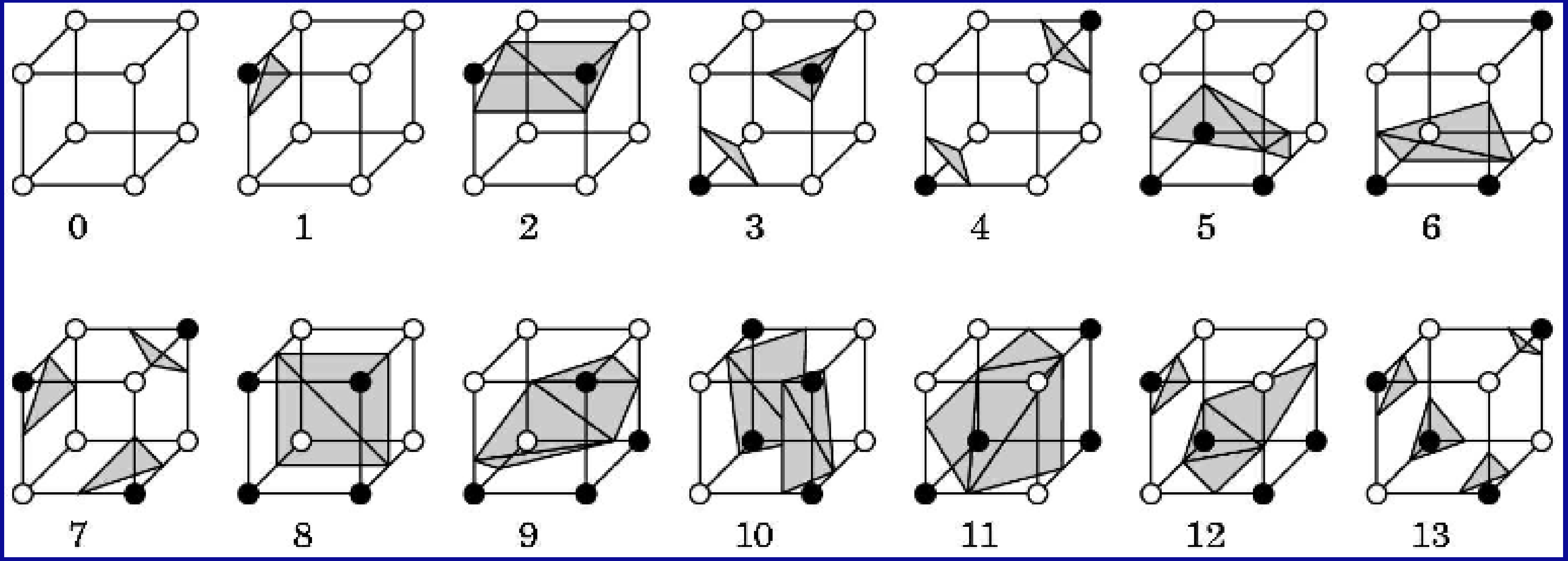
Marching Cubes

- 14 cube labelings (after elimination symmetries)

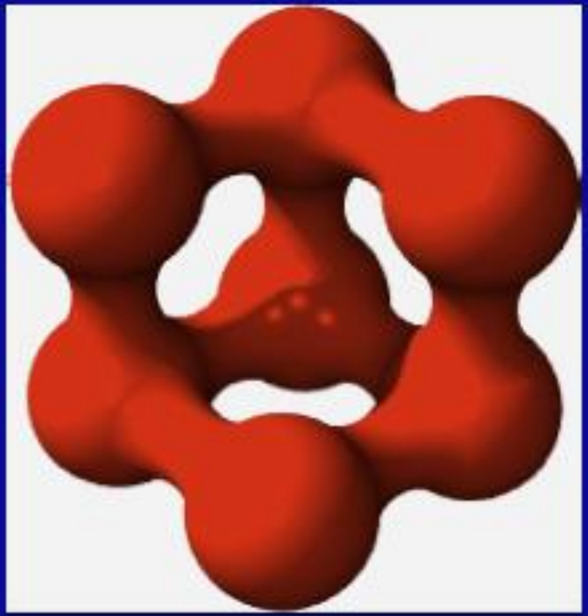


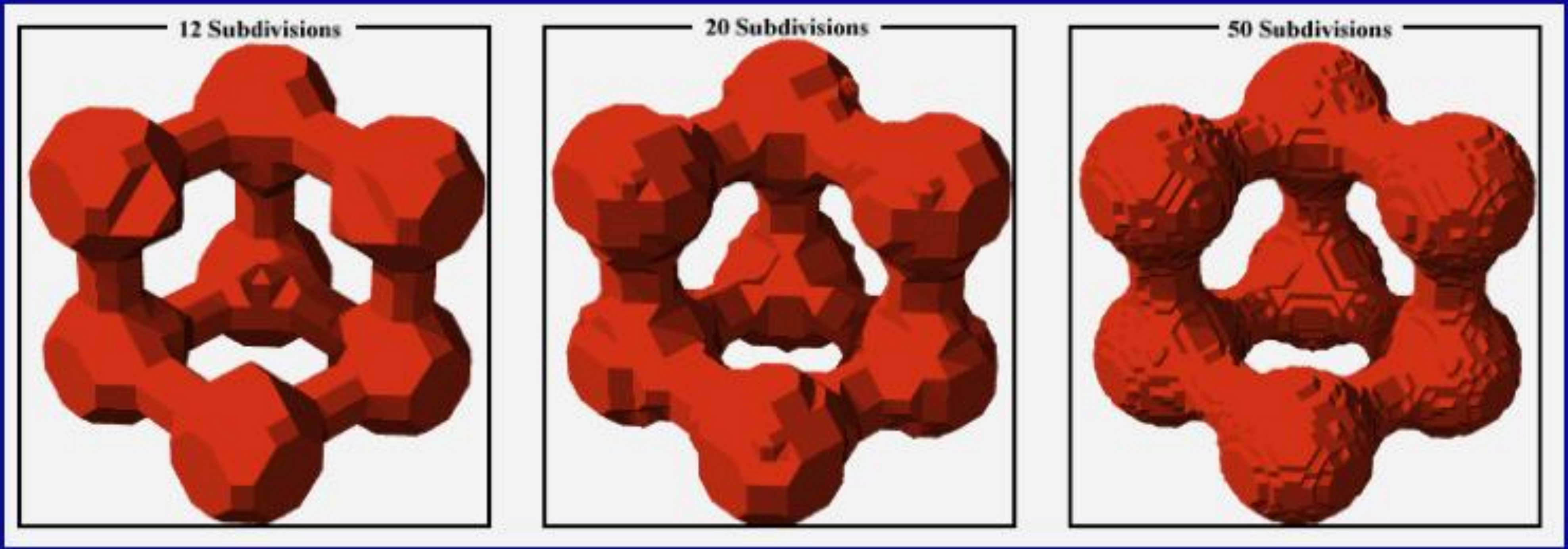
Marching Cube Tessellations

- Generalize marching squares, just more cases
- Interpolate as in 2D
- Ambiguities similar to 2D



Marching Squares Examples



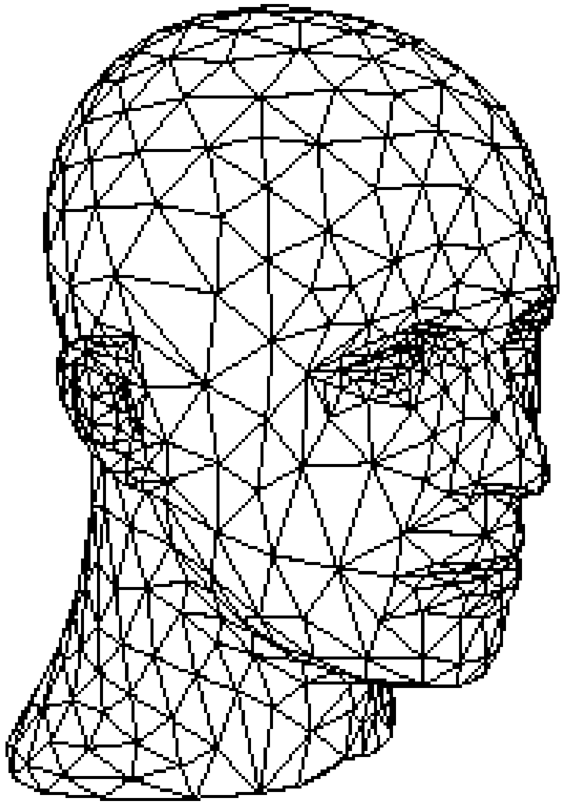


3D Object Representations

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

Polygon Meshes

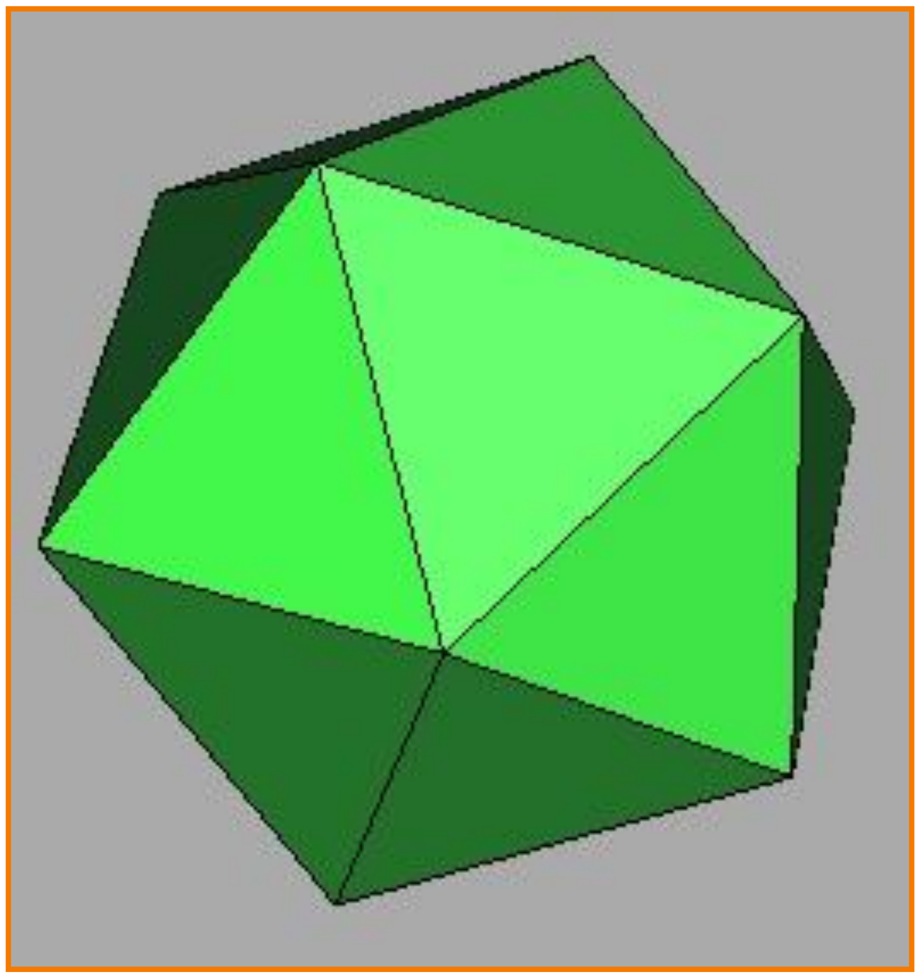
- How should we represent a mesh in a computer?
  - Efficient traversal of topology
  - Efficient use of memory
  - Efficient updates
- Mesh Representations
  - Independent faces
  - Vertex and face tables
  - Adjacency lists
  - Winged-Edge
  - Half-Edge
  - etc.

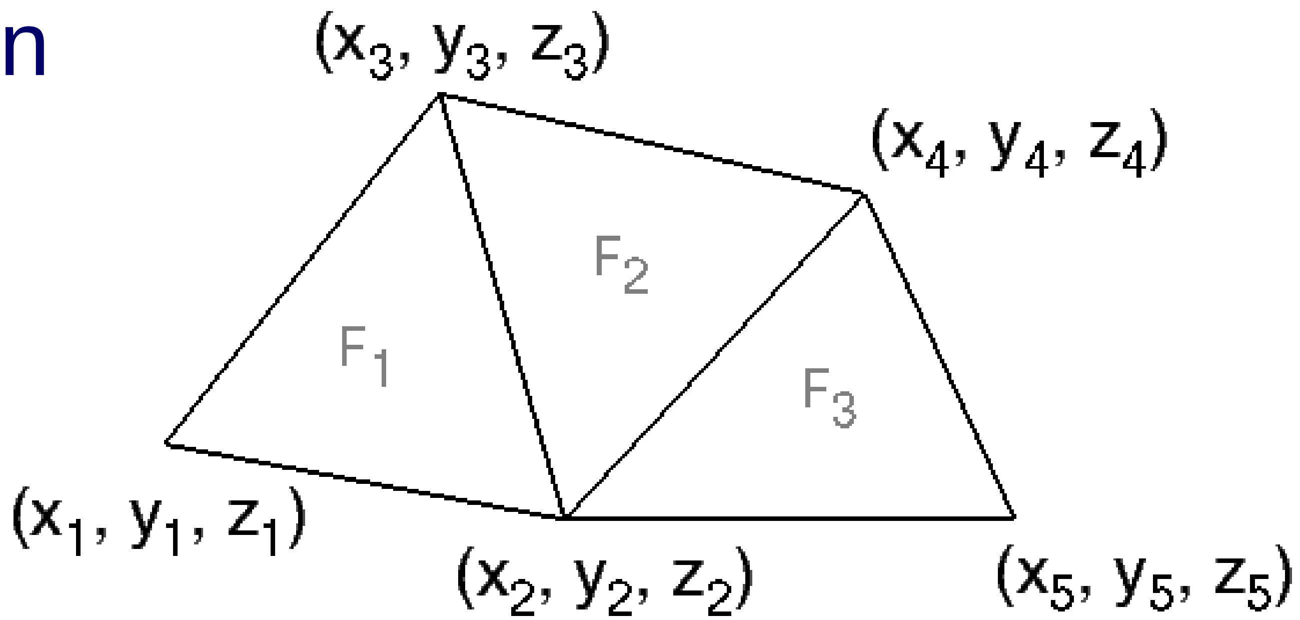


Zorin & Schroeder, SIGGRAPH 99, Course Notes

Independent Faces

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

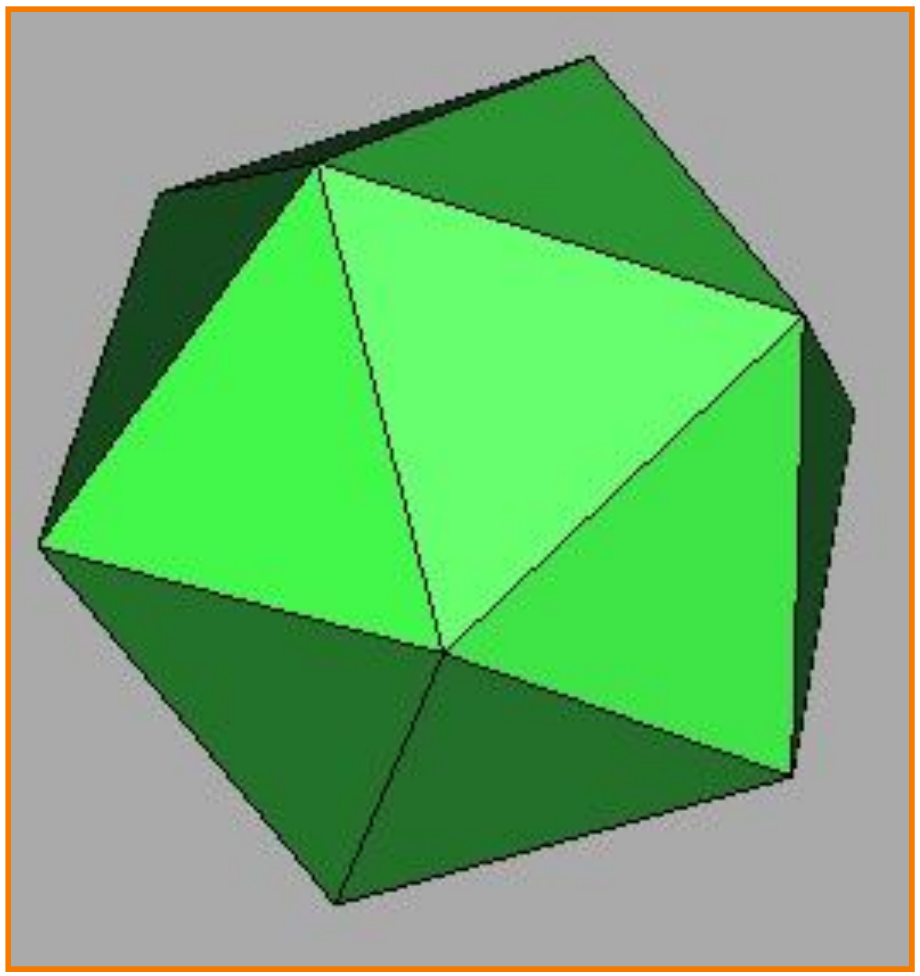


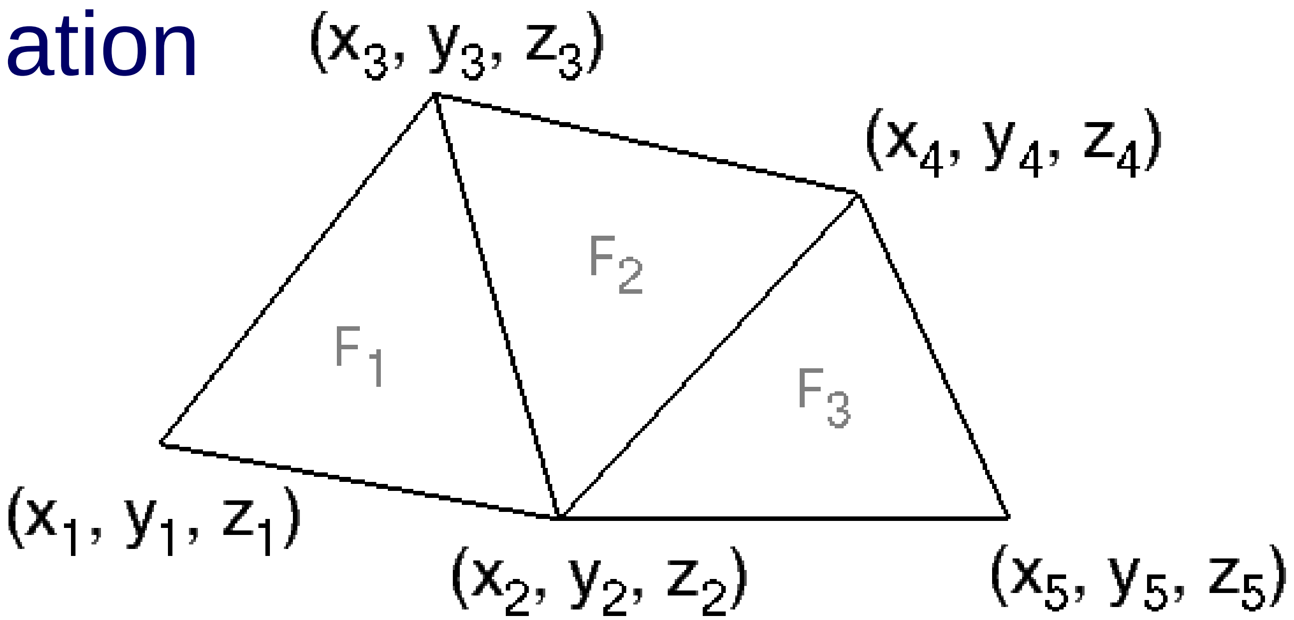


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

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

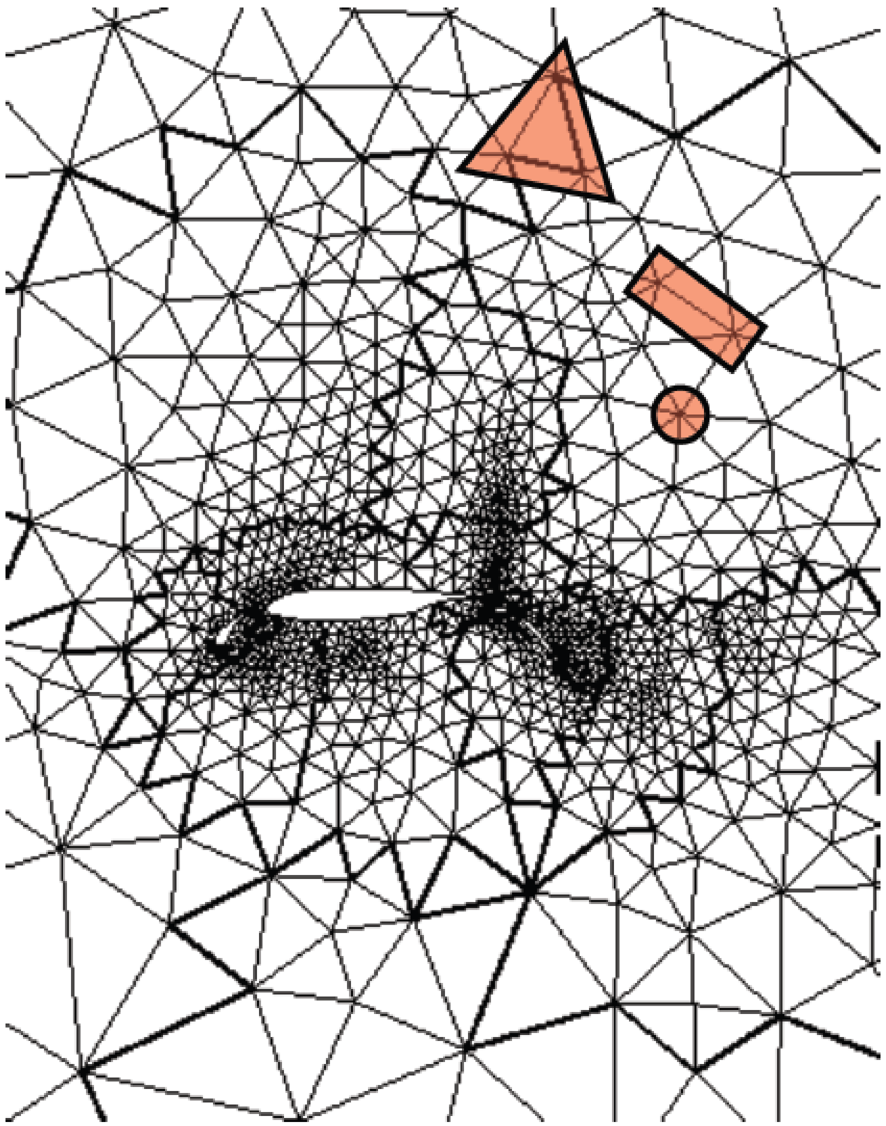




VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

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

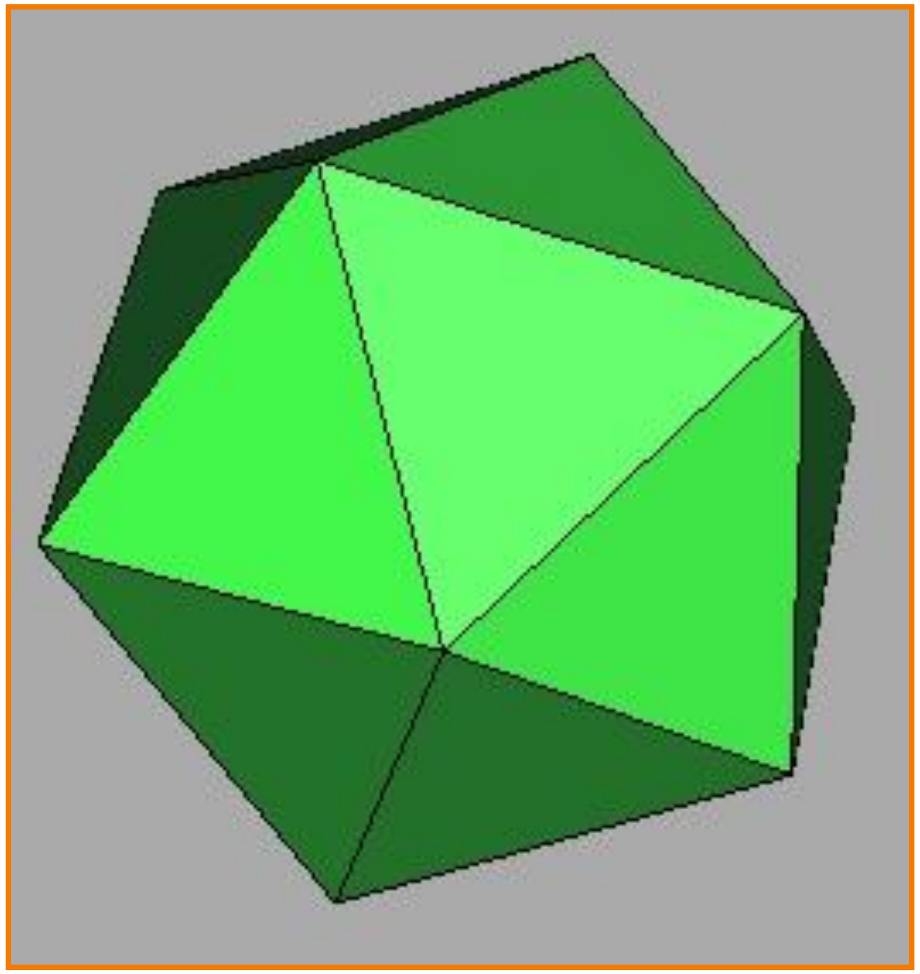
Possible Queries

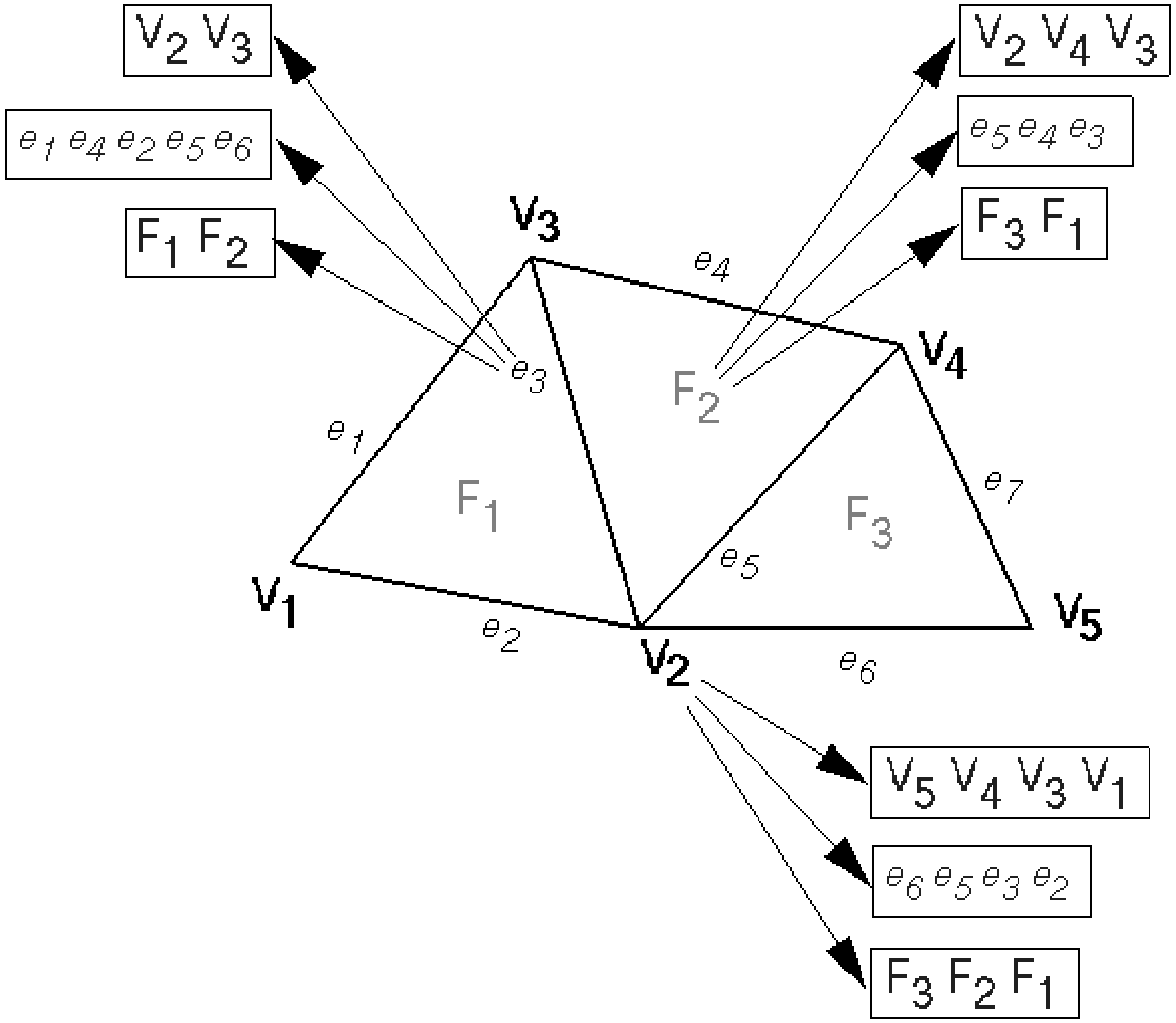


- Which faces use this vertex?
- Which edges use this vertex?
- Which faces border this edge?
- Which edges border this face?
- Which faces are adjacent to this face?

Adjacency Lists

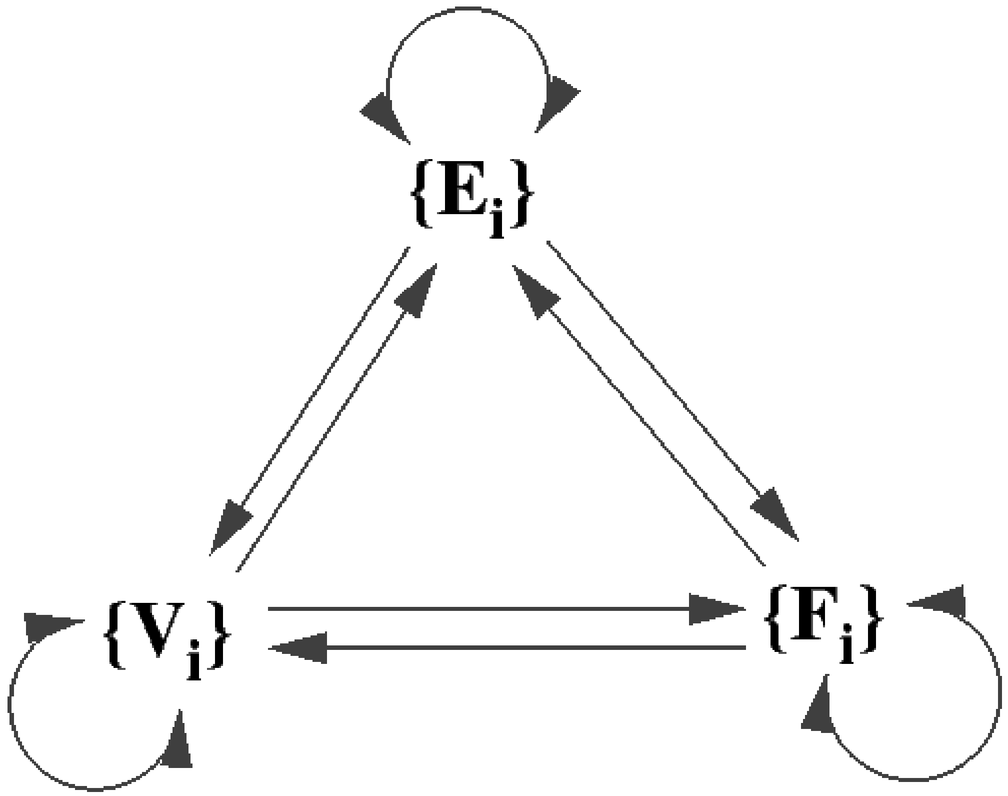
- Store all vertex, edge, and face adjacencies
  - Efficient adjacency traversal
  - Extra storage

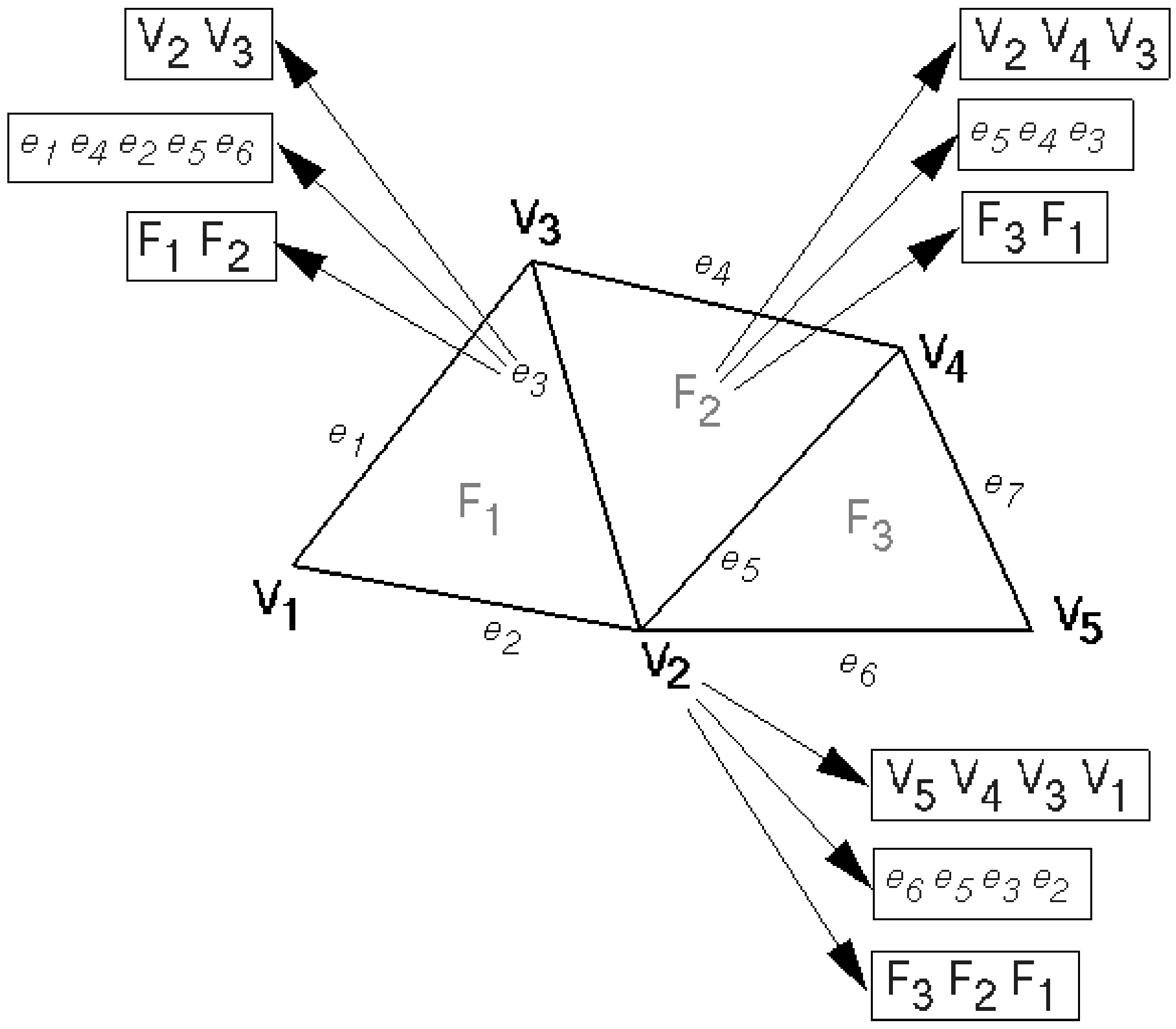




Partial Adjacency Lists

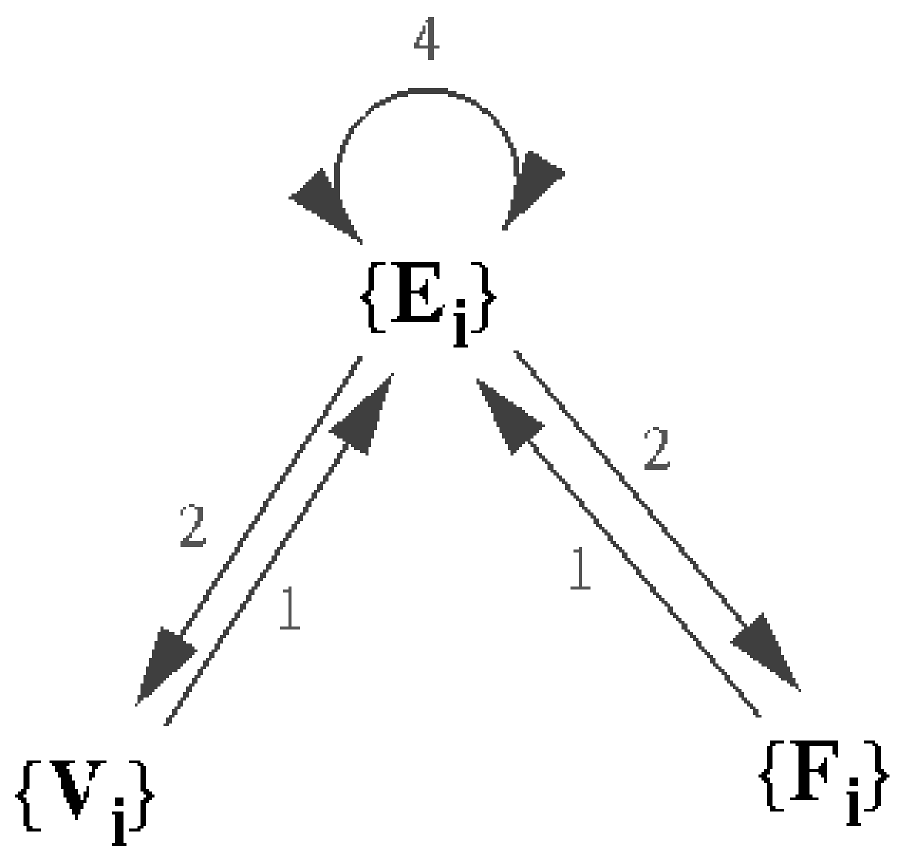
- Can we store only some adjacency relationships and derive others?

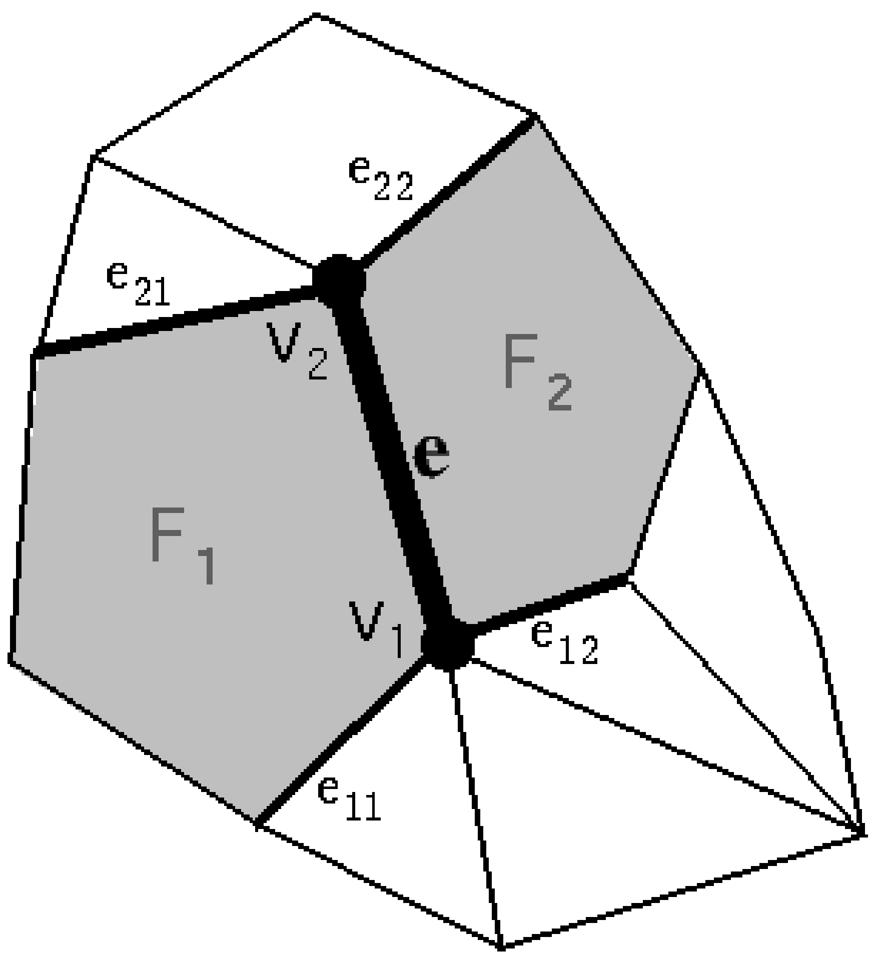




Winged Edge

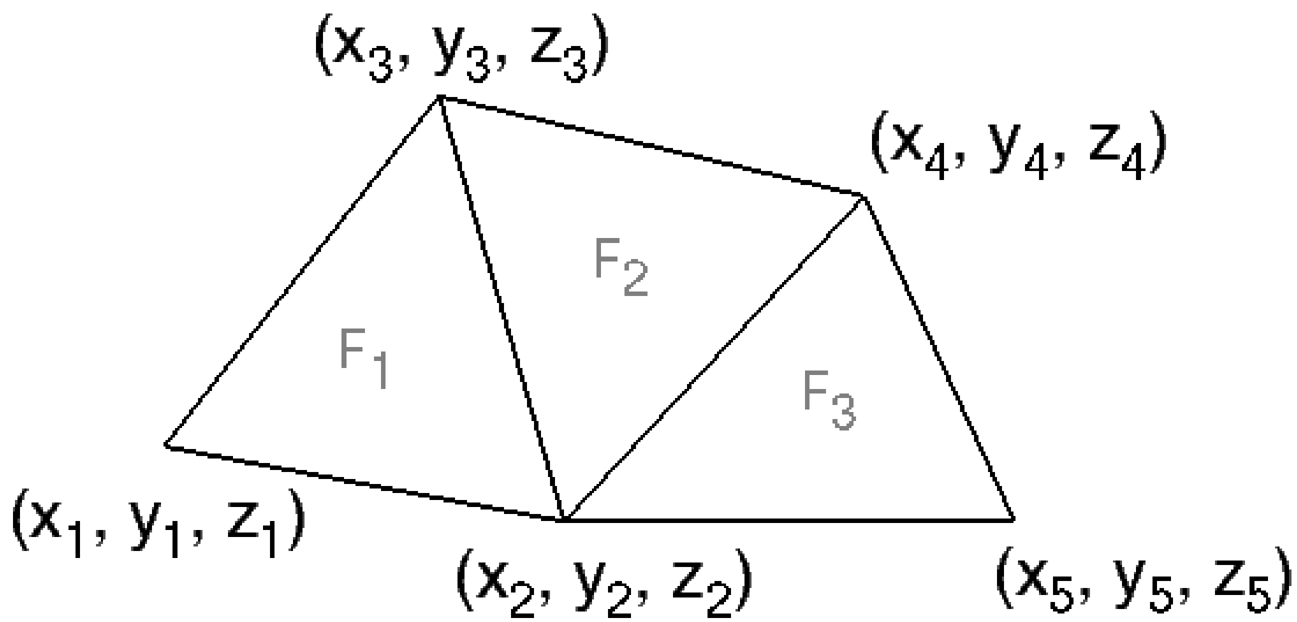
- Adjacency encoded in edges
  - All adjacencies in O(1) time
  - Little extra storage (fixed records)
  - Arbitrary polygons





Winged Edge

- Example:



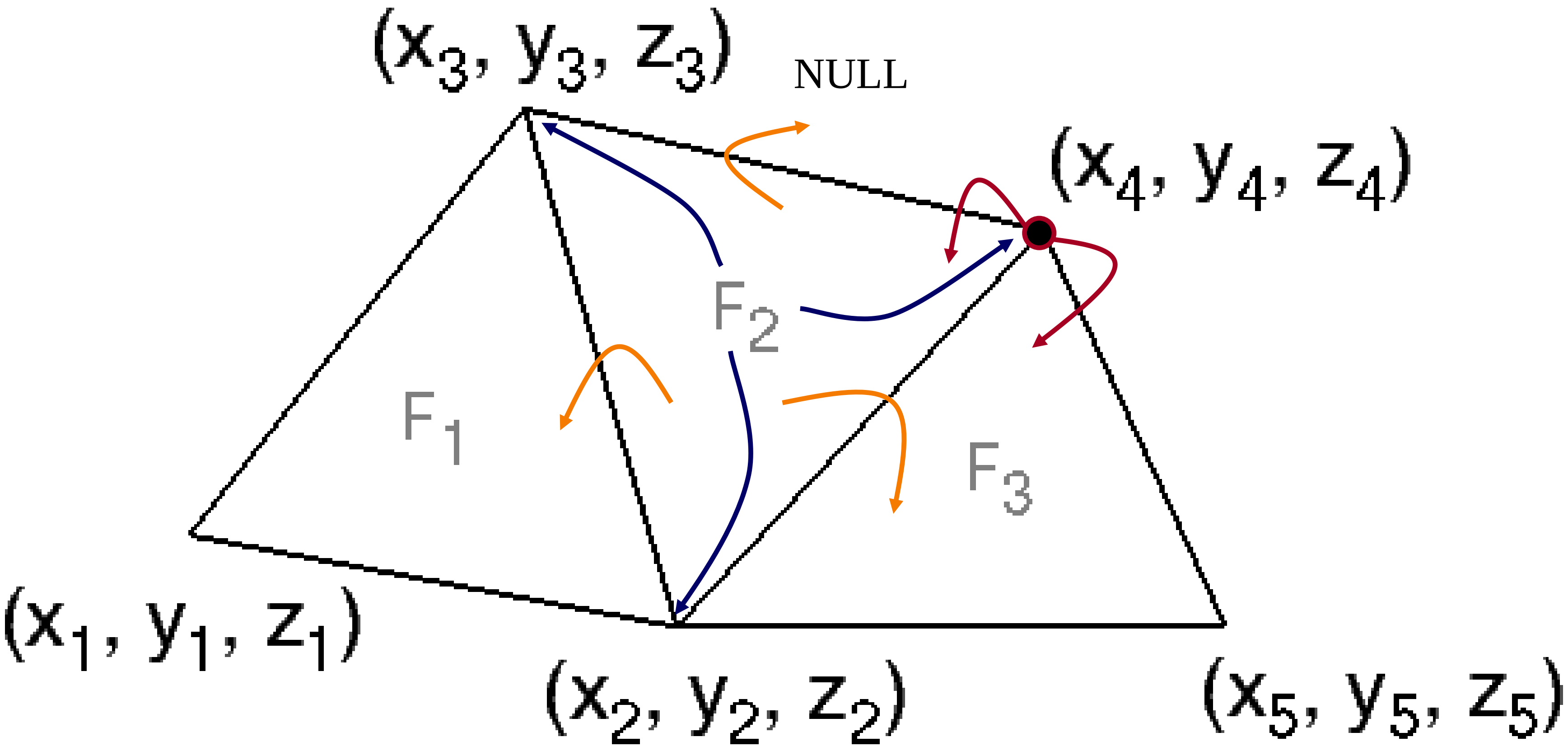
VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

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	$V3$	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

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

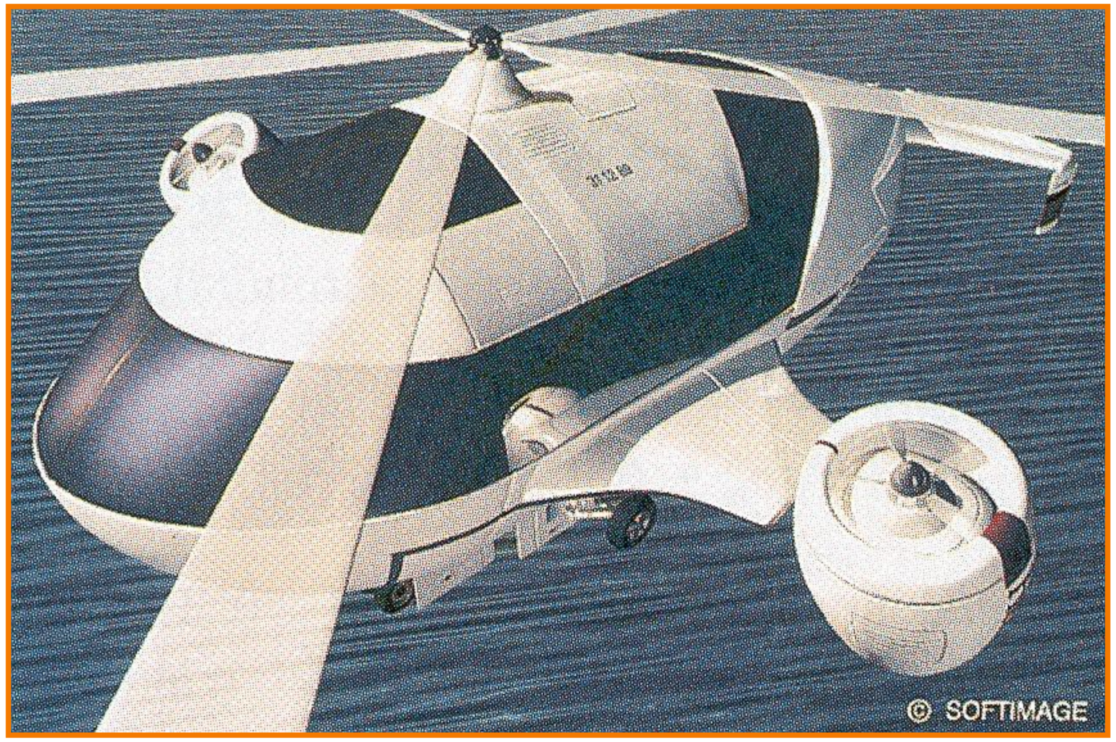


3D Object Representations

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

Surfaces

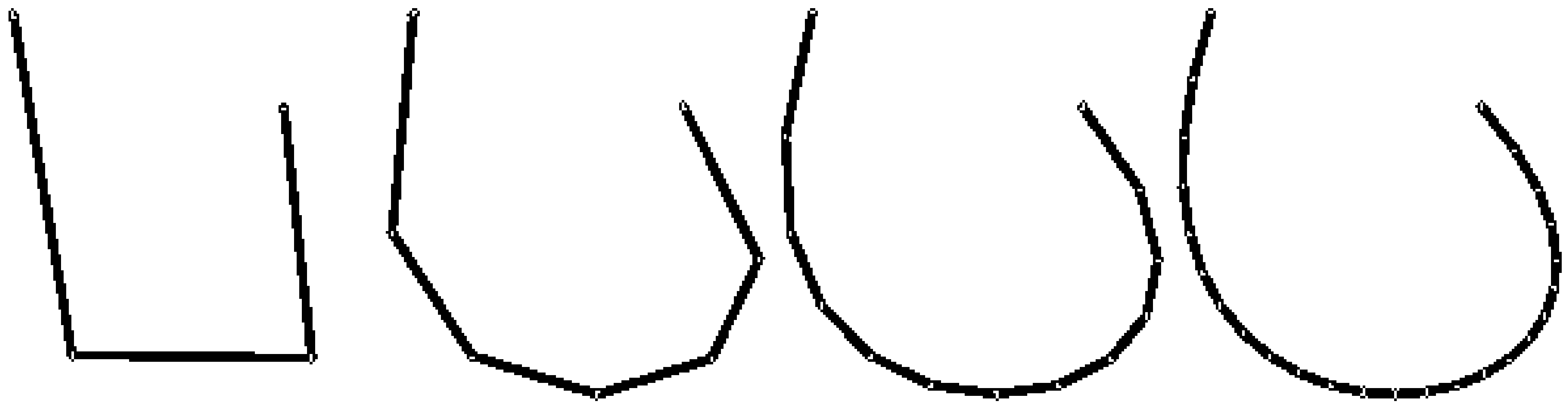
- What makes a good surface representation?
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - ➢ Guaranteed continuity
  - Natural parameterization
  - Efficient display
  - Efficient intersections



H&B Figure 10.46

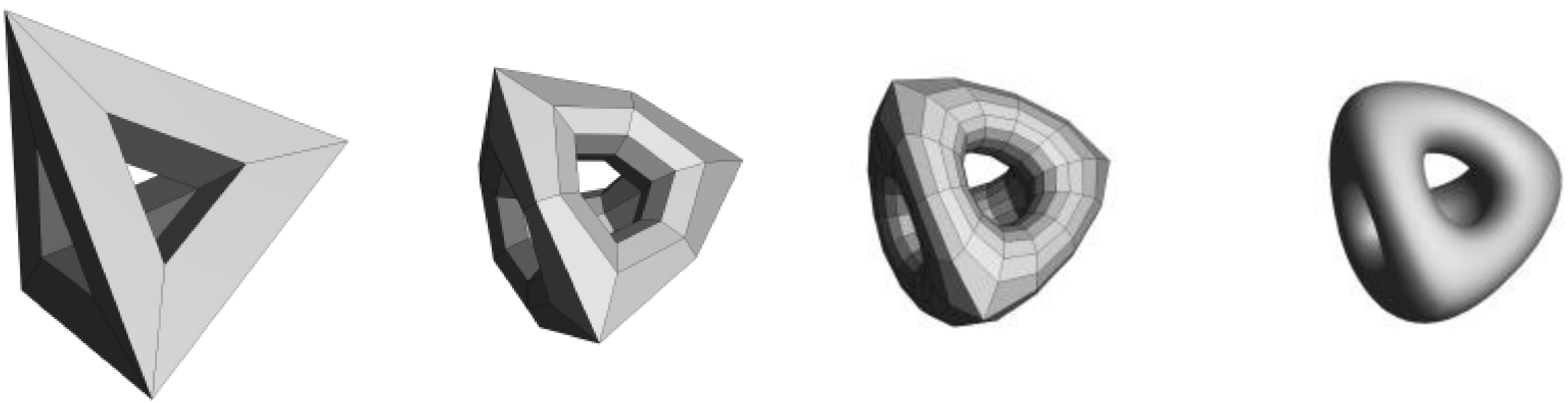
Subdivision

- How do you make a smooth curve?



Subdivision Surfaces

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

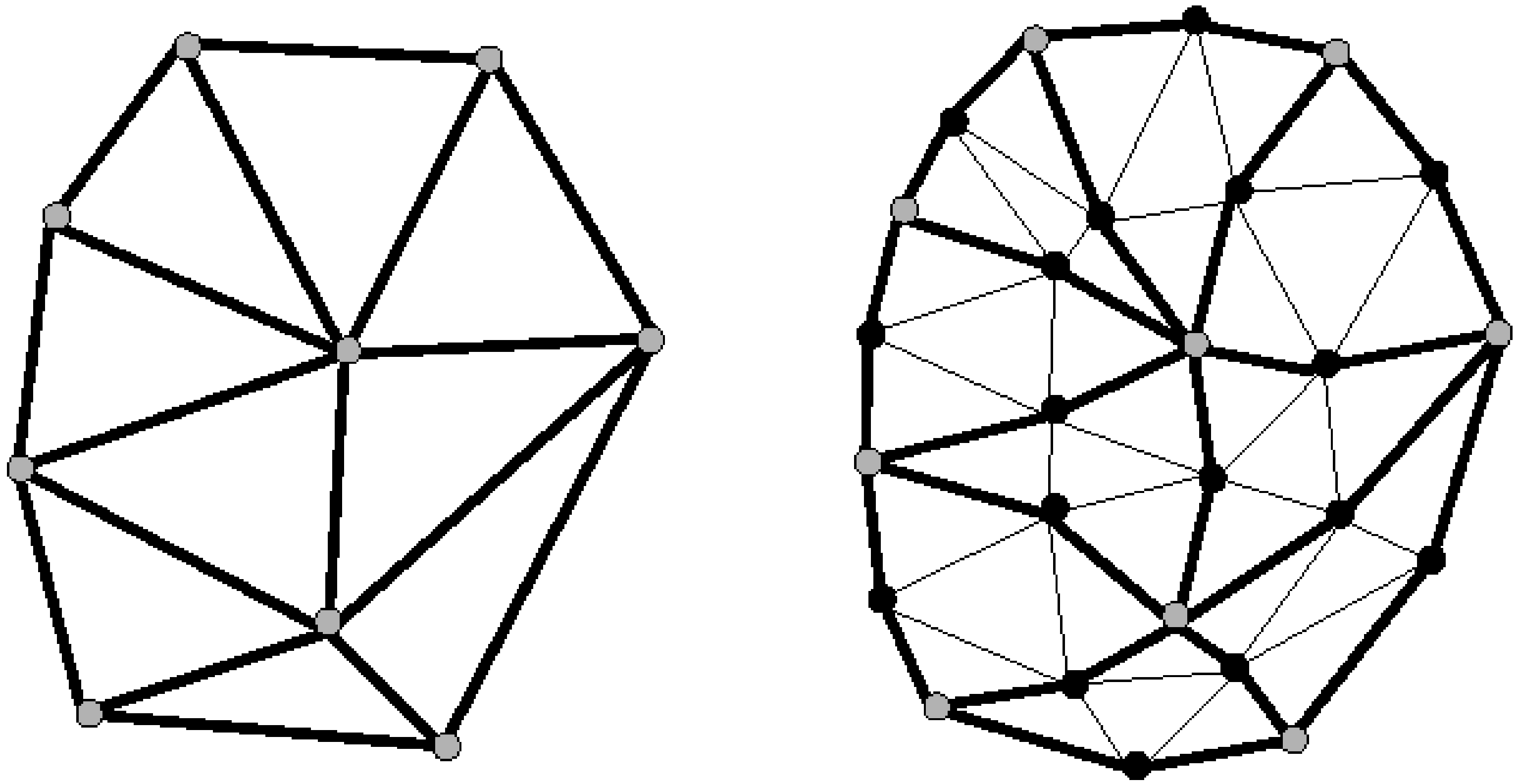


(a) (b) (c) (d)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

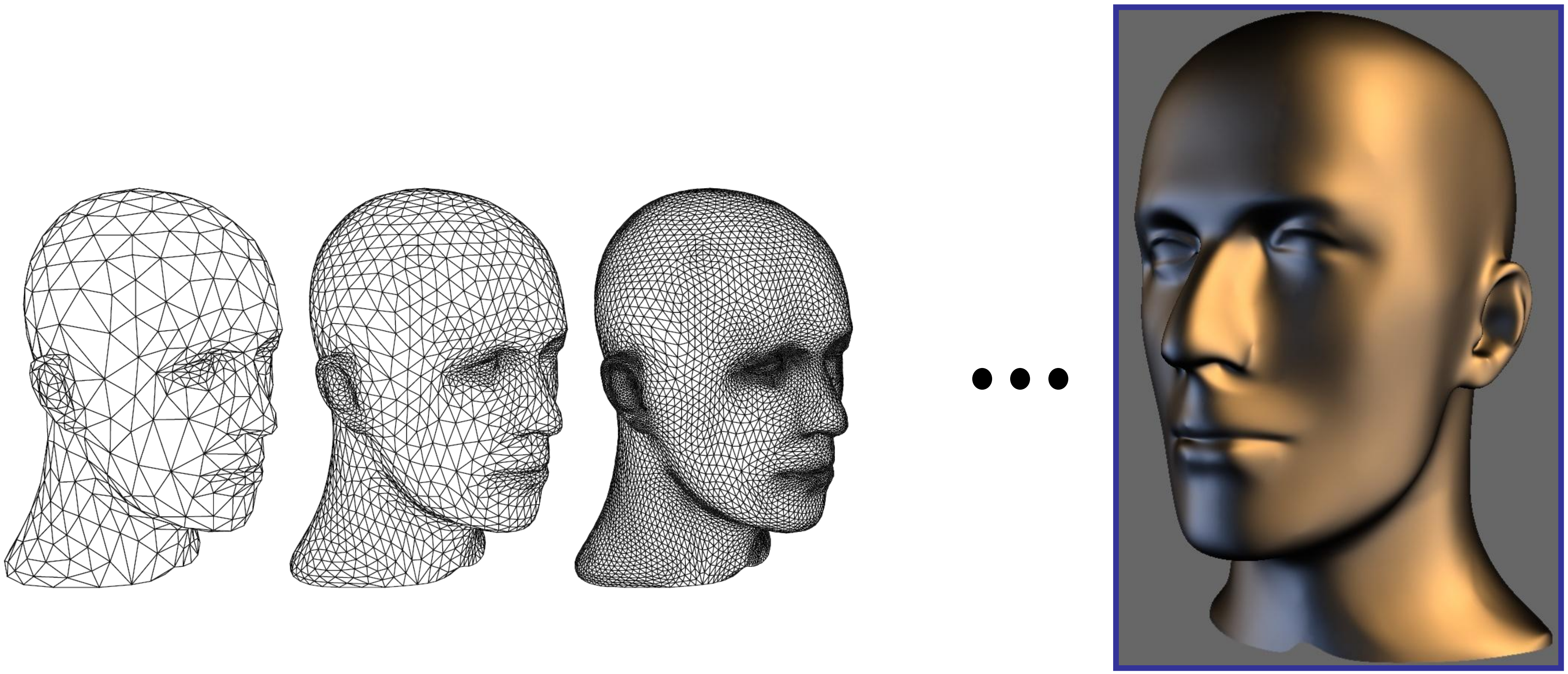
Key Questions

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



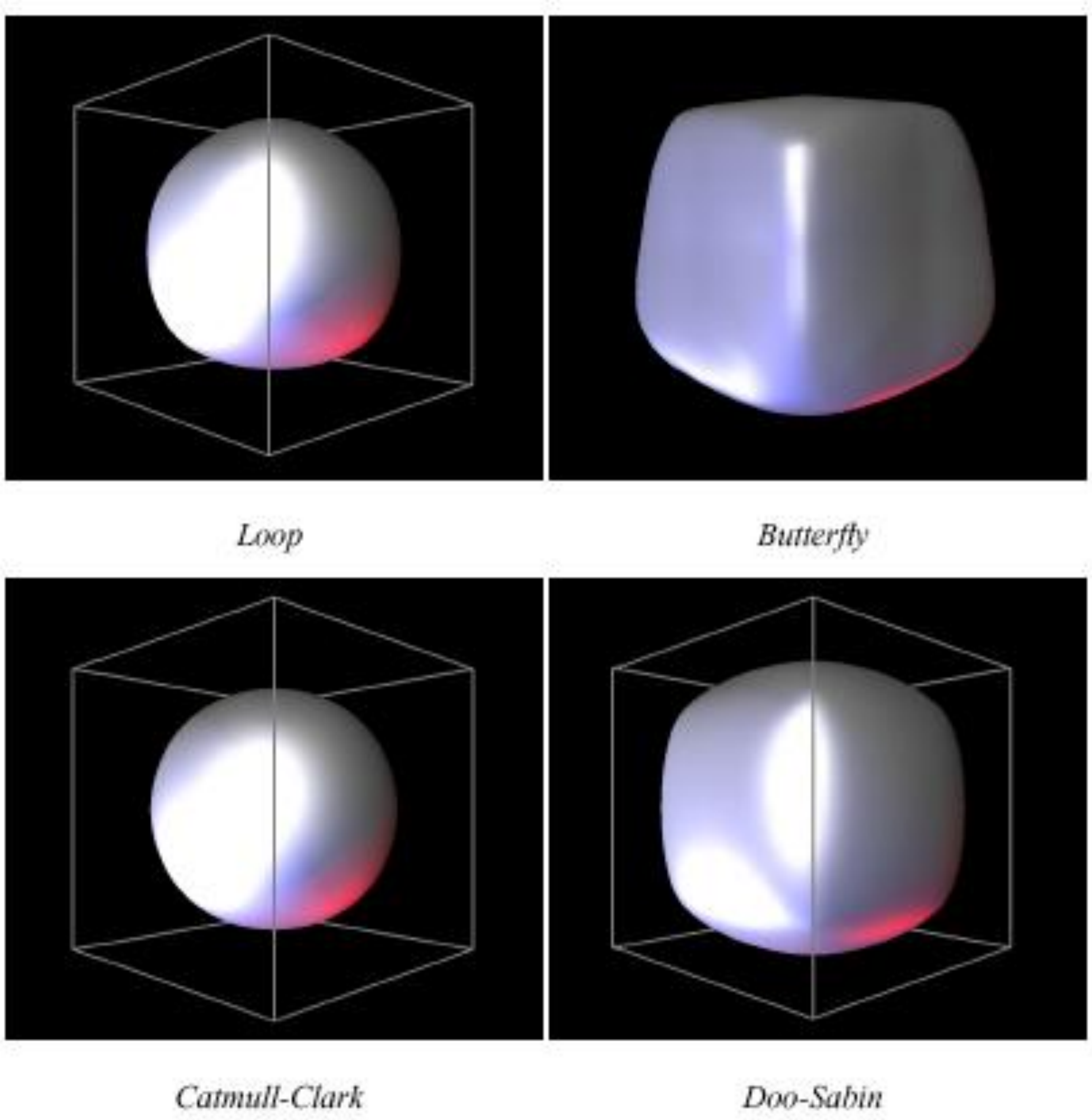
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces – A 3D example



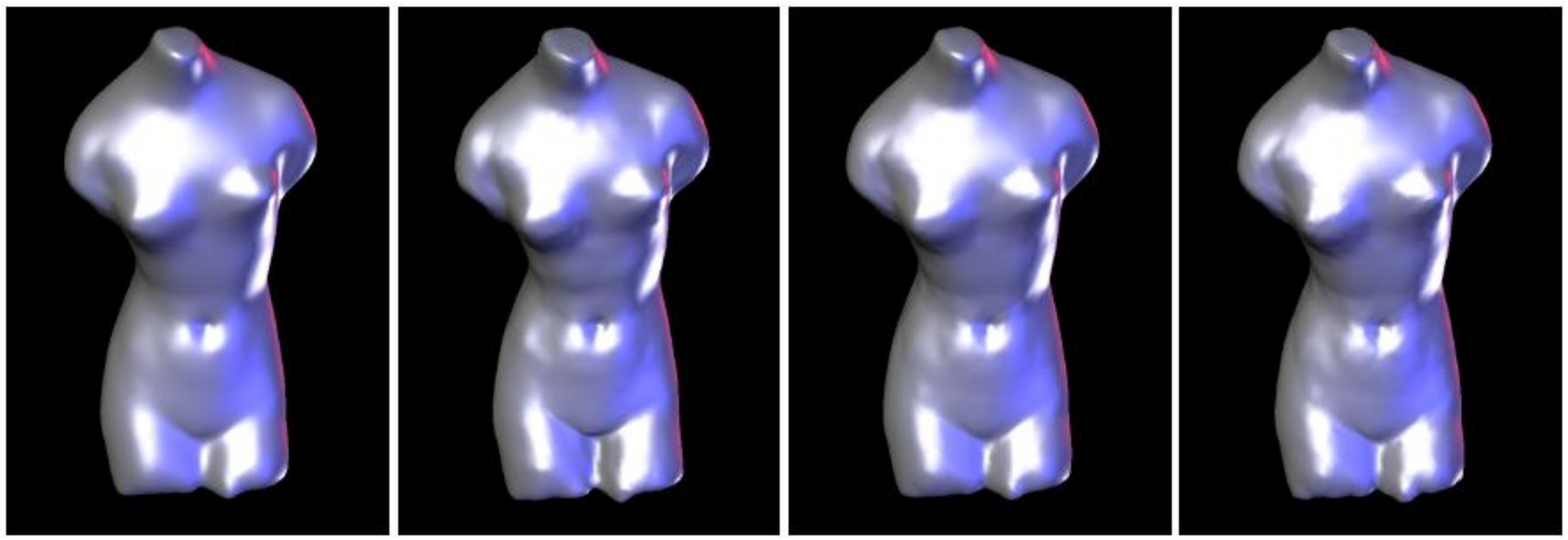
Applications: Computer Graphics Animation

Subdivision Schemes



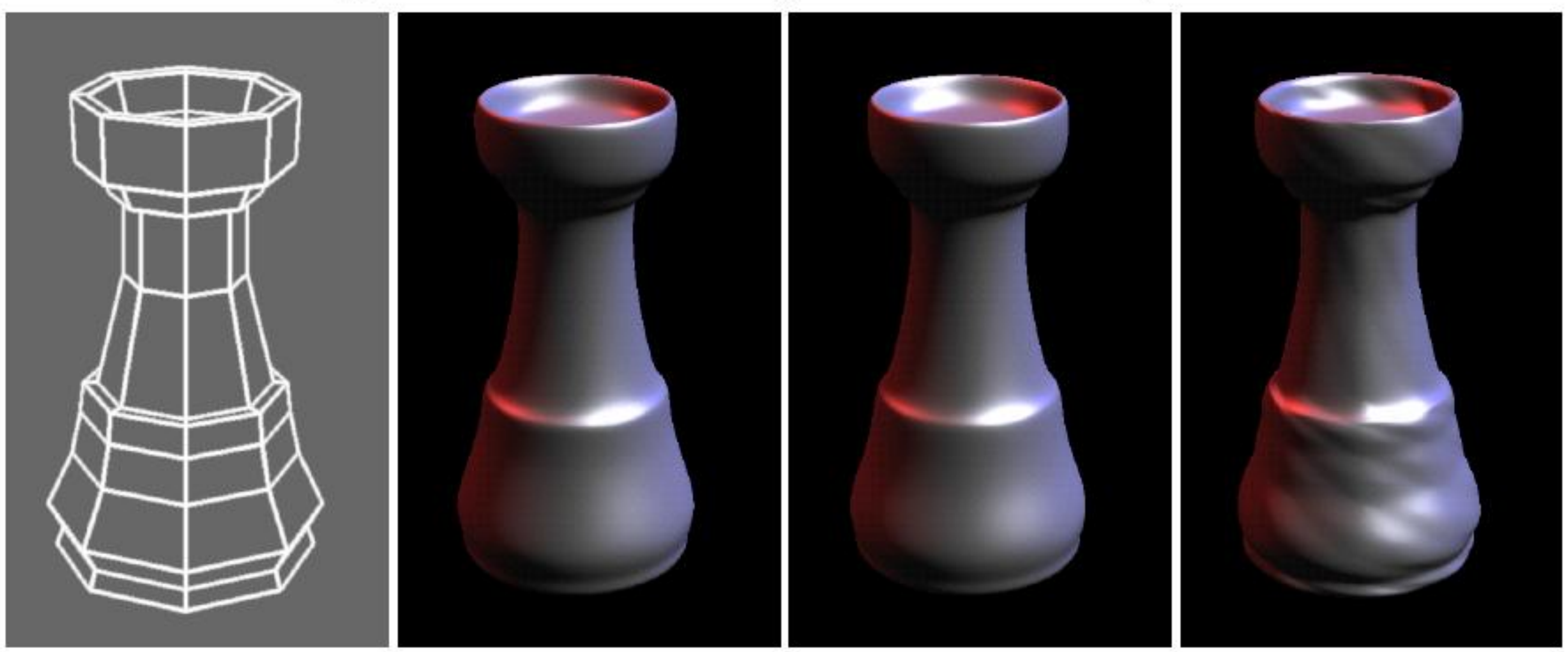
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Visual Comparison



Loop | *Butterfly* | *Catmull-Clark* | *Doo-Sabin*

Different subdivision schemes produce similar results for smooth meshes.



Initial mesh | *Loop* | *Catmull-Clark* | *Catmull-Clark, after triangulation*

Subdivision Surfaces

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

Pixar

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