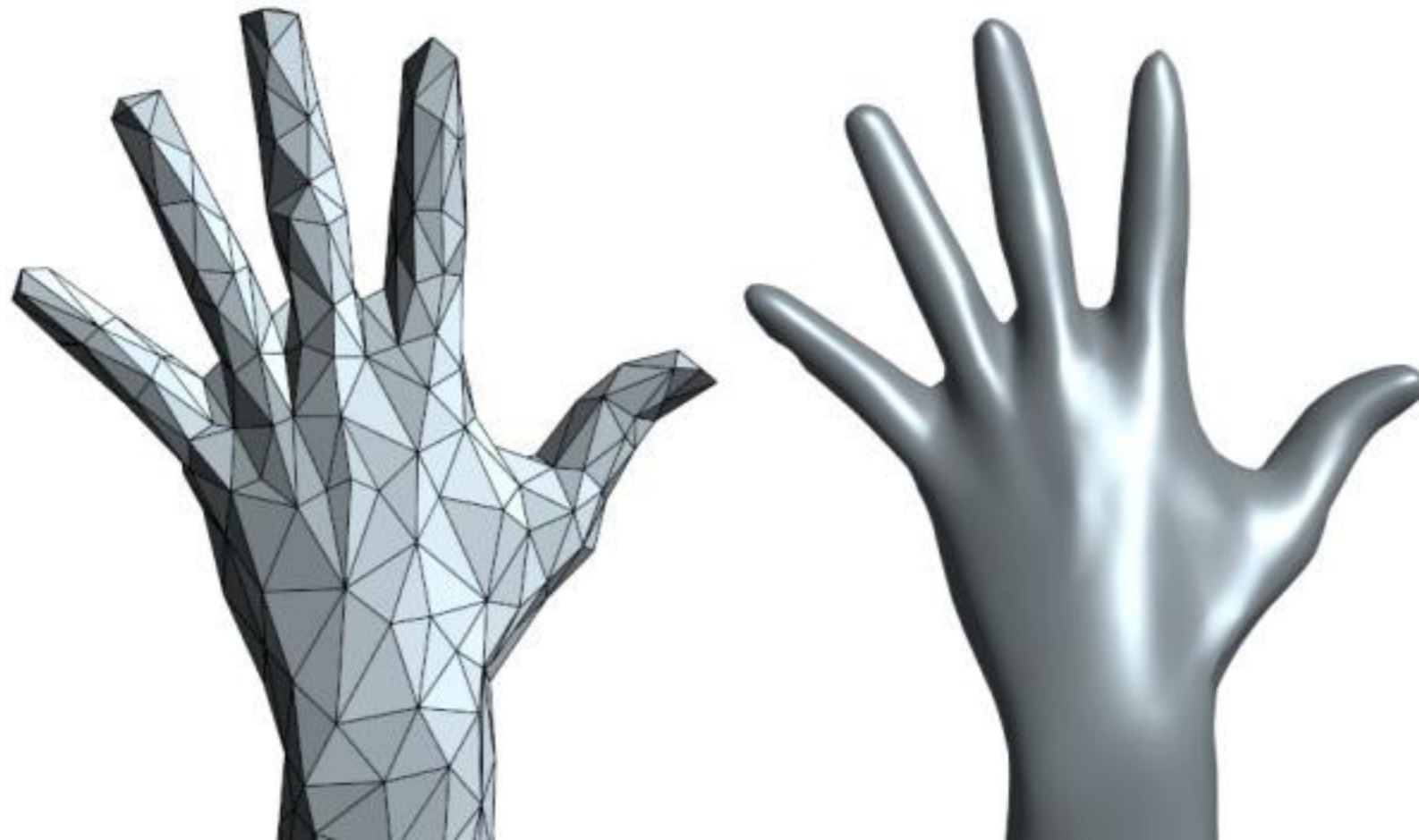


# 3D Surfaces



source:

[http://iparla.labri.fr/publications/2007/BS07b/sketch\\_teaser.jpg](http://iparla.labri.fr/publications/2007/BS07b/sketch_teaser.jpg)

# Please leave early feedback for your TAs!

Index (to access all courses):

<https://www.ugrad.cs.cmu.edu/ta/feedback>

Direct link:

<https://www.ugrad.cs.cmu.edu/ta/S13/feedback/course.cgi?course=15462>



# 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

## A point based representation:

Pauly, M., Keiser, R., Kobbelt, L. P., & Gross, M., Shape modeling with point-sampled geometry. SIGGRAPH 2003.

<http://dl.acm.org/citation.cfm?id=882319>

## Some CSG examples:

Guenter, B., & Gavrilu, M., Exact procedural CSG modeling for real time graphics. Microsoft Research Technical Report, 2005.

<http://research.microsoft.com/en-us/um/people/bgguenter/>

<http://research.microsoft.com/en-us/um/people/bgguenter/docs/template.pdf>

[http://research.microsoft.com/en-us/um/people/bgguenter/images/PGeom\\_0\\_1.mov](http://research.microsoft.com/en-us/um/people/bgguenter/images/PGeom_0_1.mov)

## Subdivision surfaces:

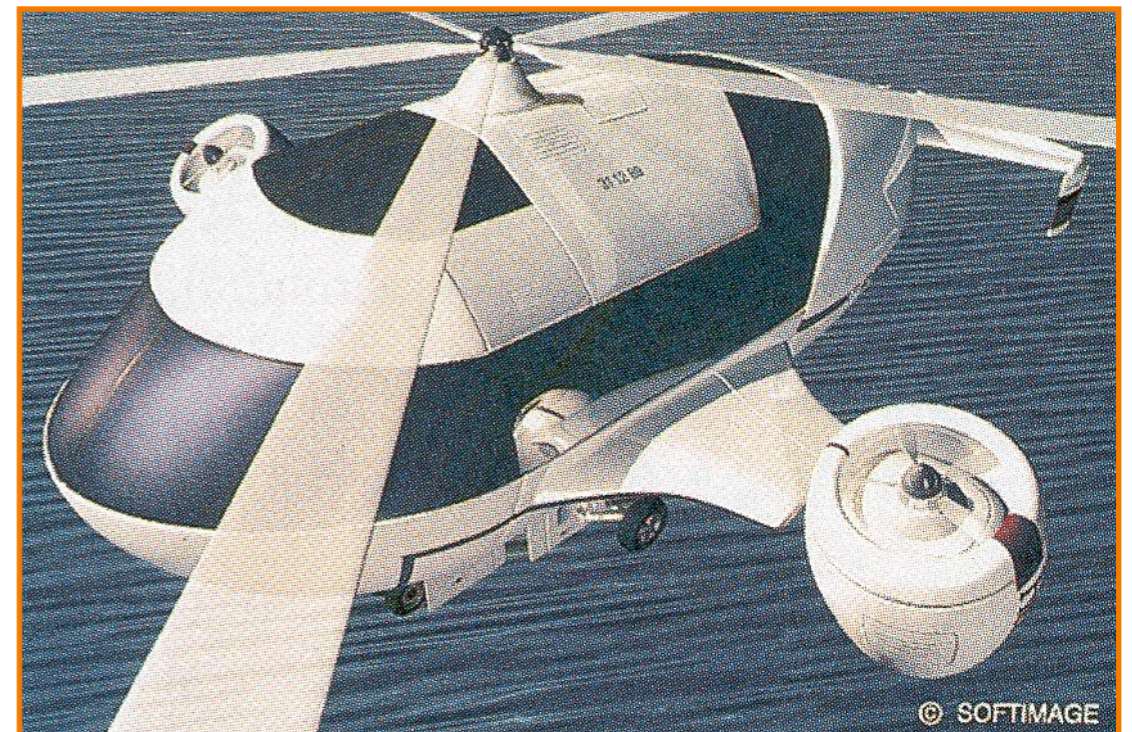
DeRose, T., Kass, M., & Truong, T. (1998). Subdivision surfaces in character animation. SIGGRAPH '98, 85–94.

<http://dl.acm.org/citation.cfm?id=280826>

# Surfaces



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



H&B Figure 10.46

## 2D Scalar Field

- $z = f(x,y)$

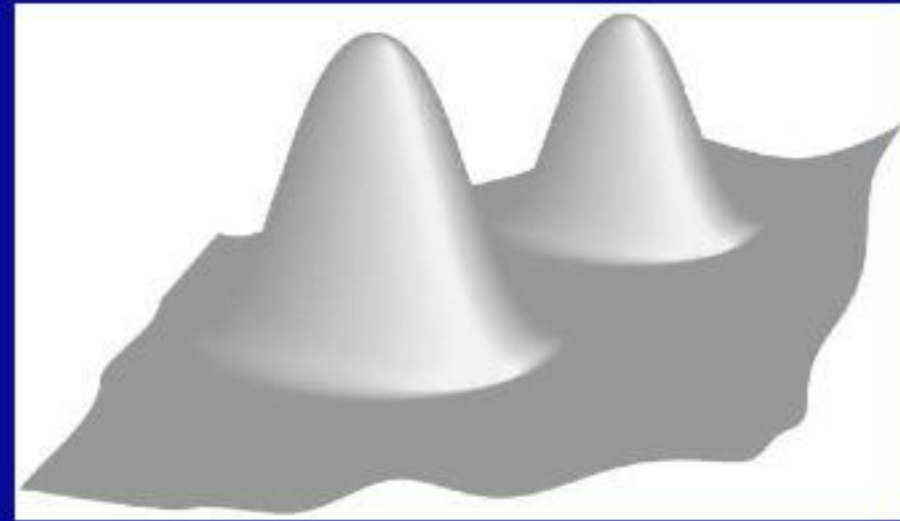
$$f(x,y) = \begin{cases} 1 - x^2 - y^2, & \text{if } x^2 + y^2 < 1 \\ 0 & \end{cases}$$

How do you visualize this function?

# Height Field

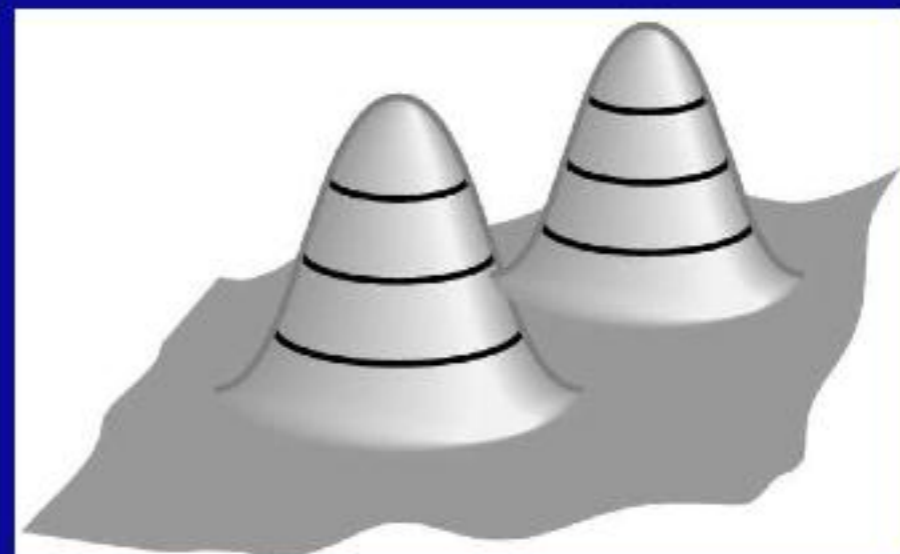
- Visualizing an explicit function

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



- Adding contour curves

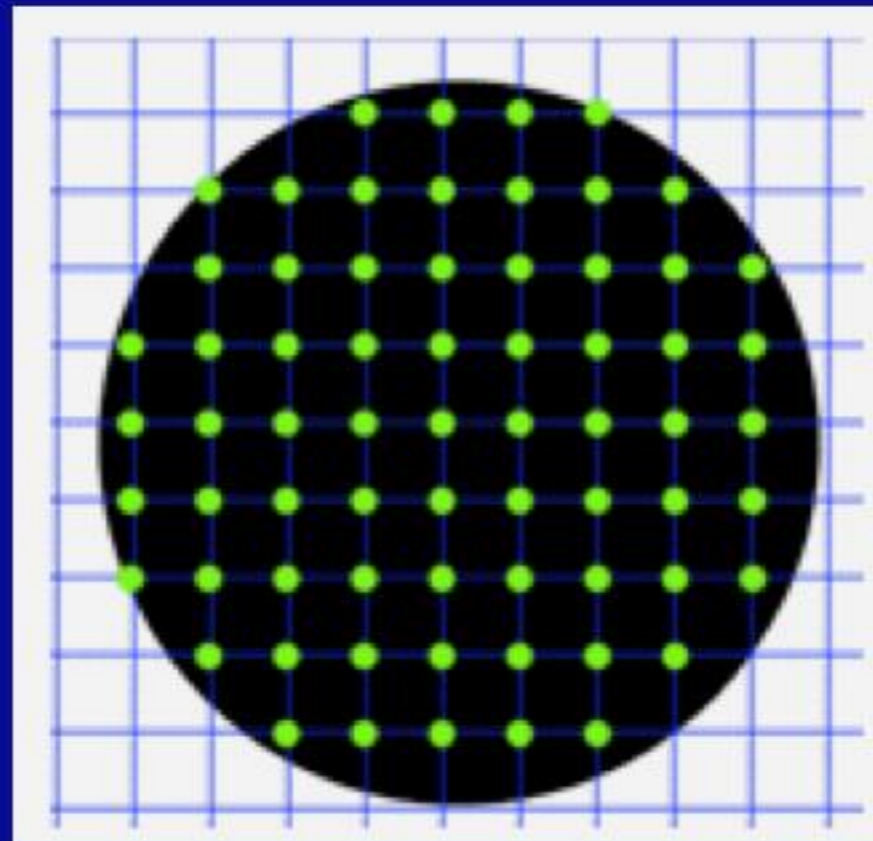
$$f(x,y) = c$$



**Implicit → Explicit 2D**  
**(Marching Squares Algorithm)**

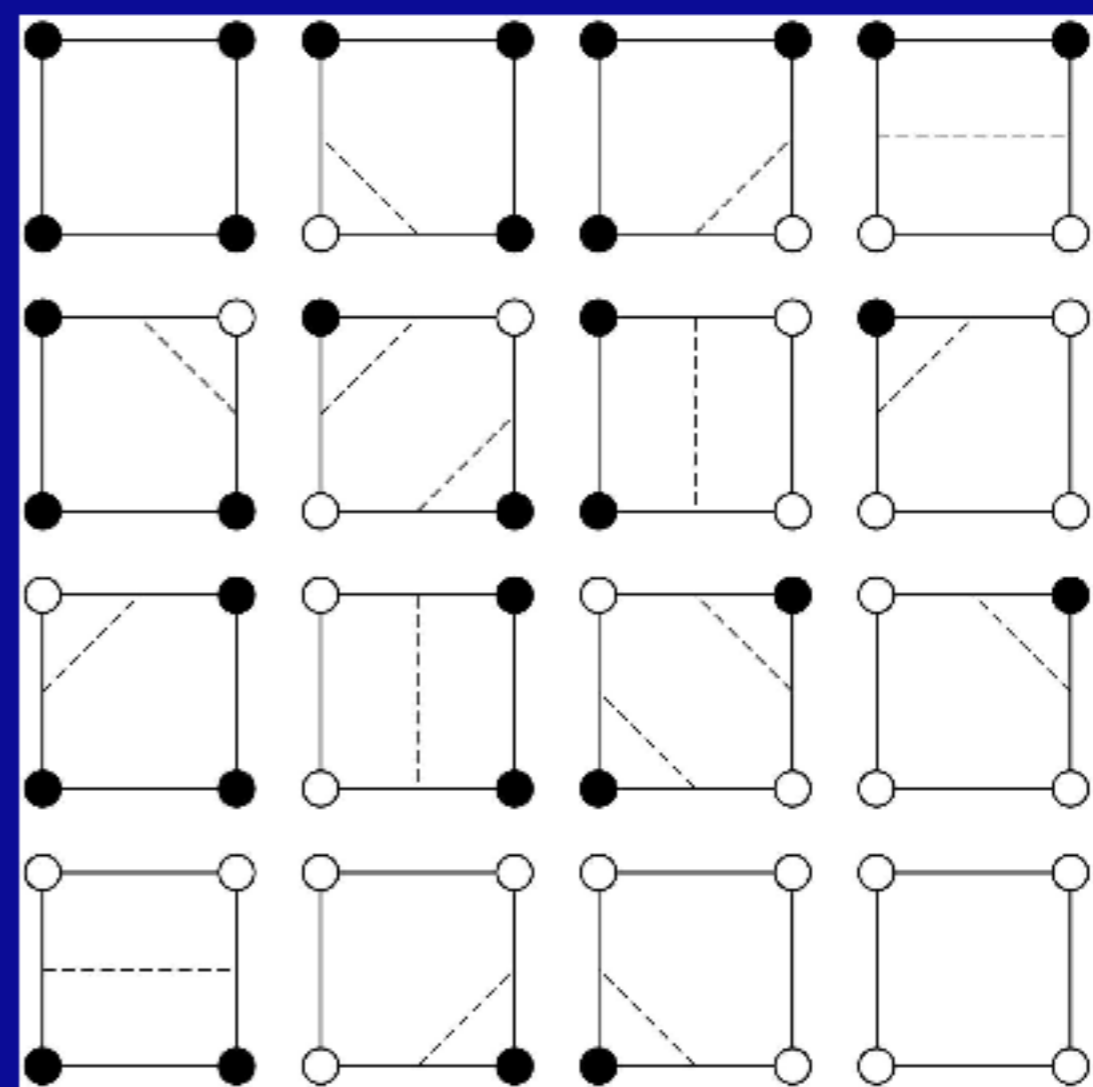
# Marching Squares

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

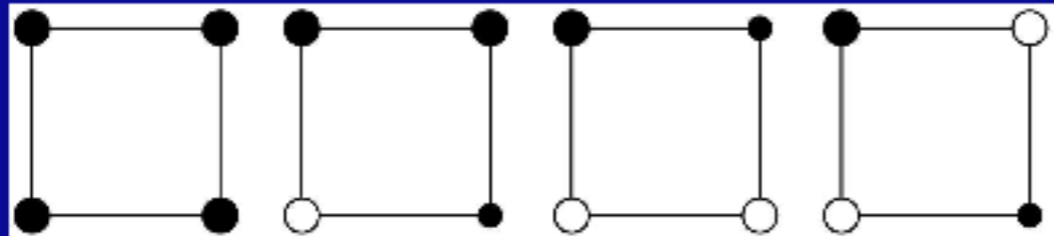


# Cases for Vertex Labels

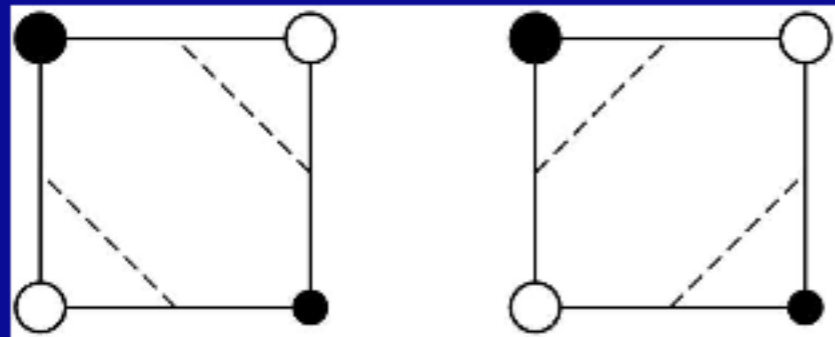
16 cases for vertex labels



4 unique mod. symmetries

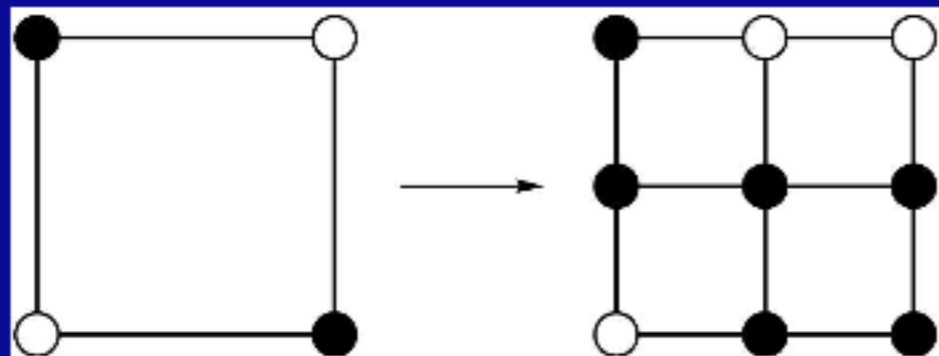
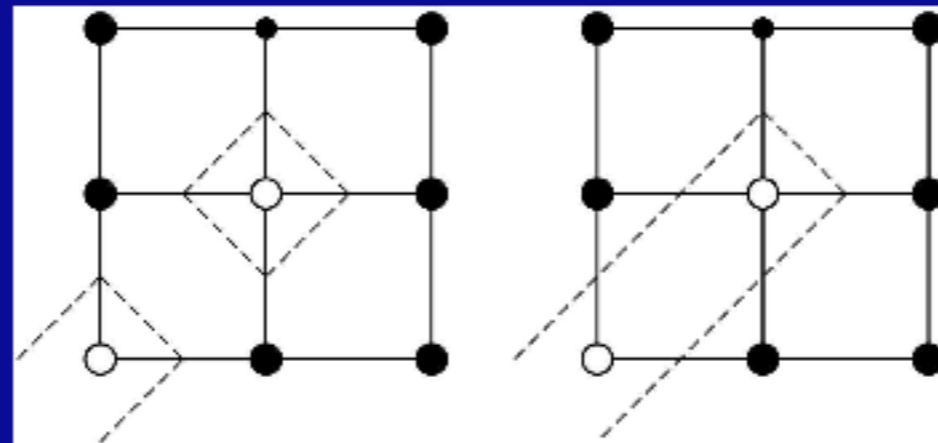


# Ambiguities of Labelings



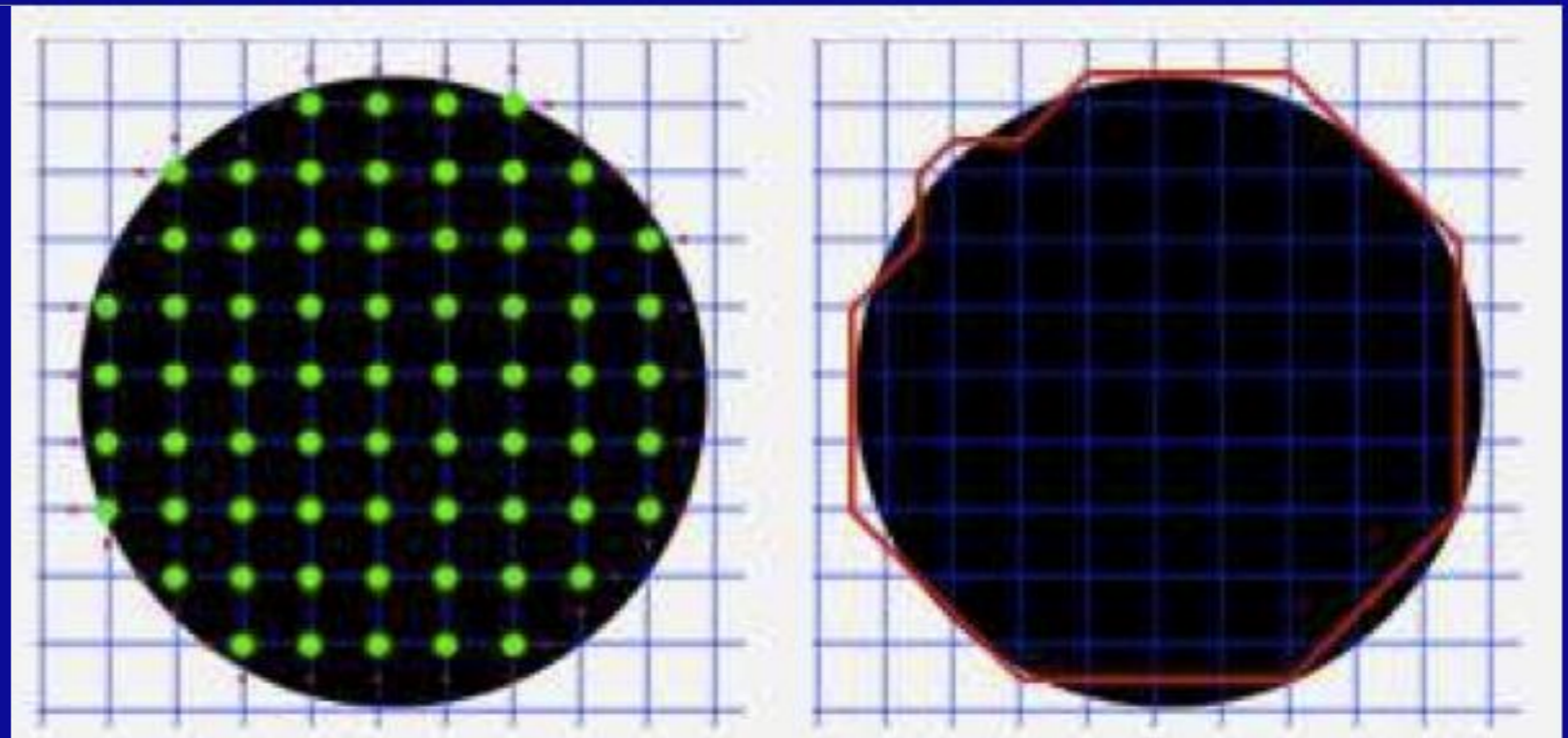
Ambiguous labels

Different resulting contours



Resolution by subdivision  
(where possible)

# Marching Squares Examples



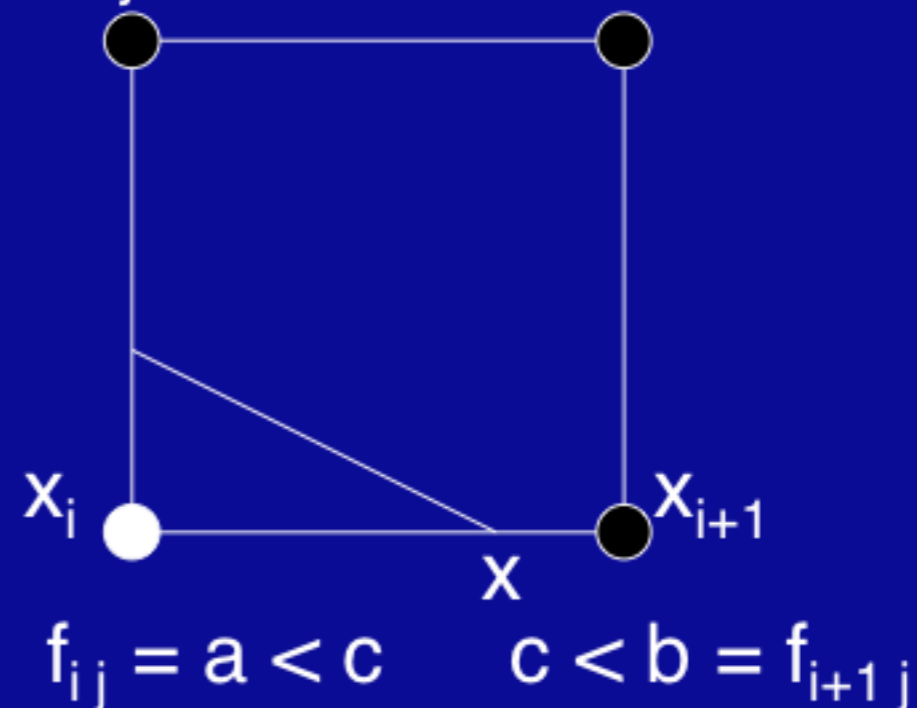
Can you do better?

# Interpolating Intersections

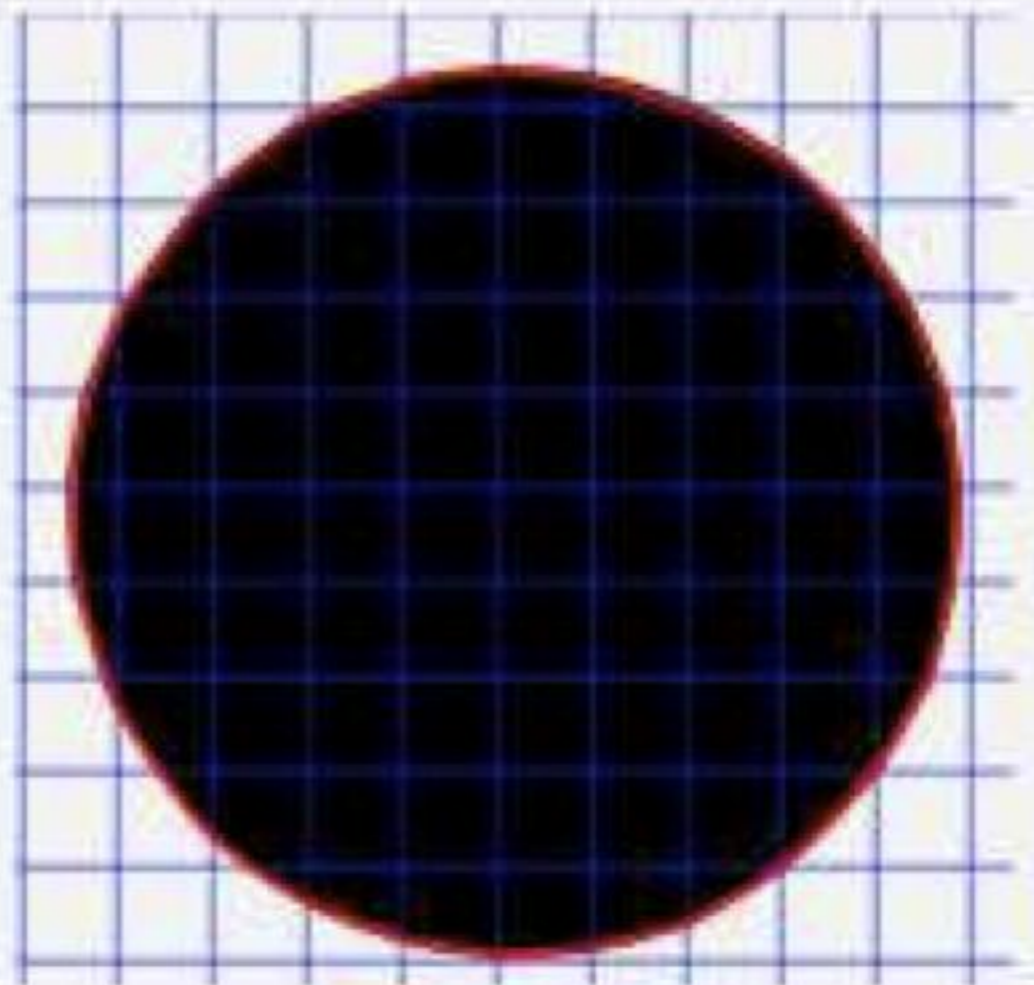
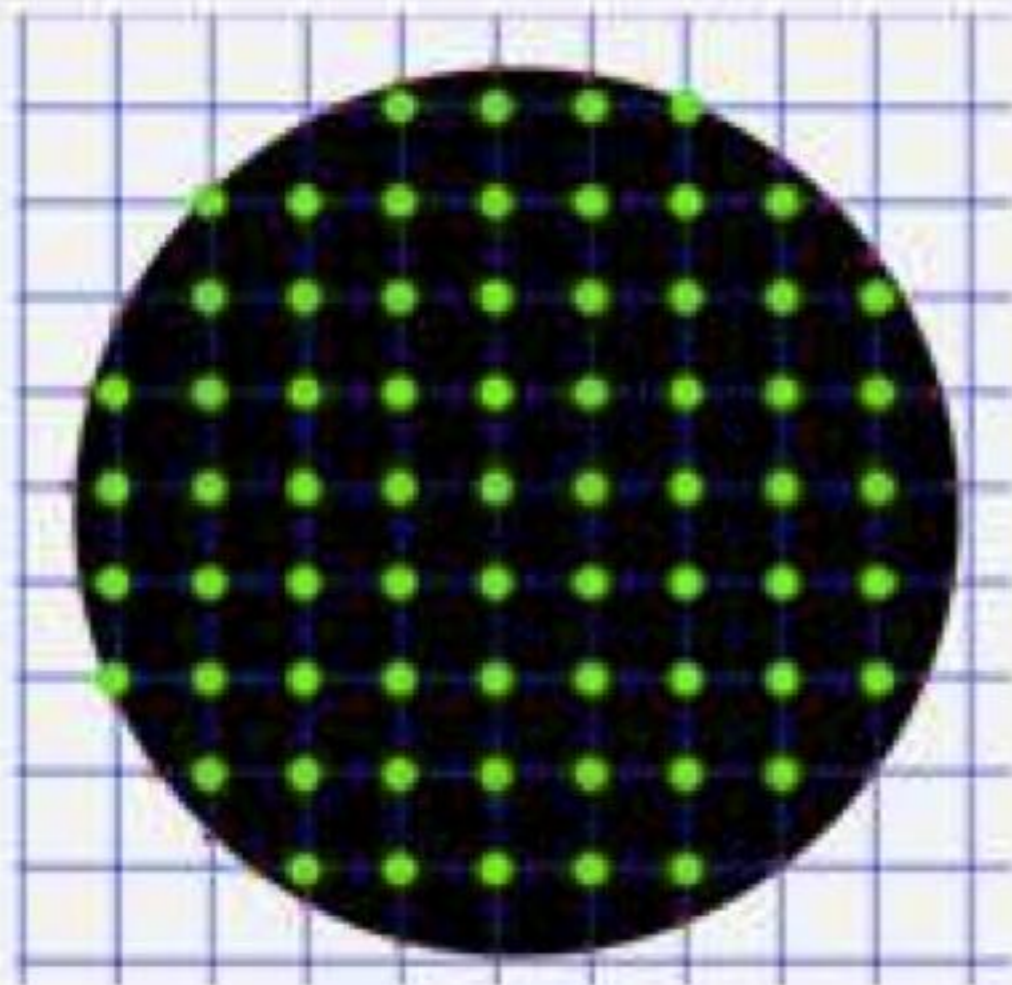
- Approximate intersection
  - Midpoint between  $x_i, x_{i+1}$  and  $y_j, y_{j+1}$
  - Better: interpolate
- If  $f_{ij} = a$  is closer to  $c$  than  $b = f_{i+1j}$  then intersection is closer to  $(x_i, y_j)$ :

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

- Analogous calculation for  $y$  direction



# Marching Squares Examples



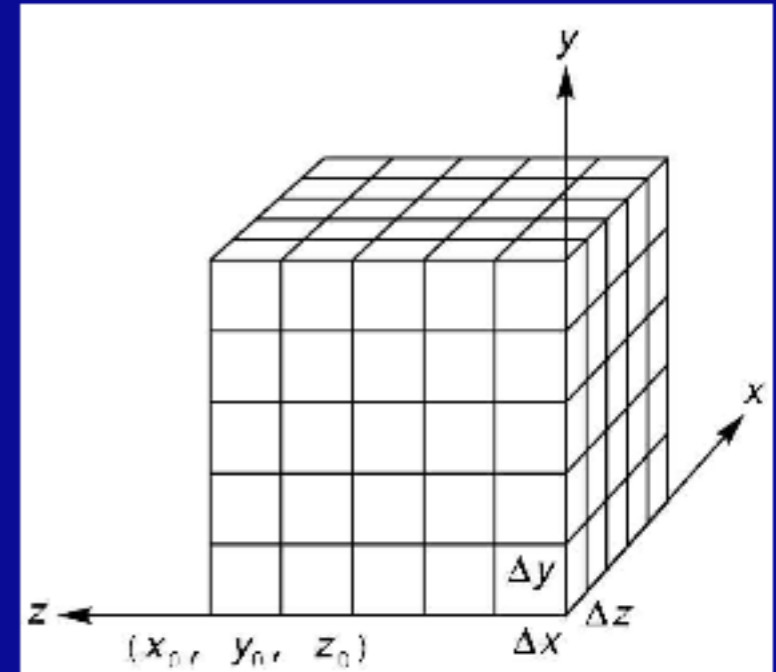
**Implicit → Explicit 3D**  
**(Marching Cubes Algorithm)**

# 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

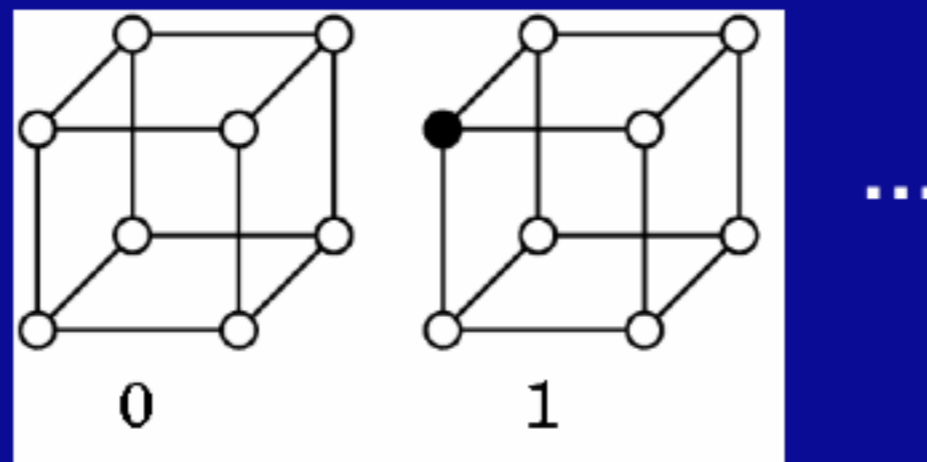


# Isosurfaces

- 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

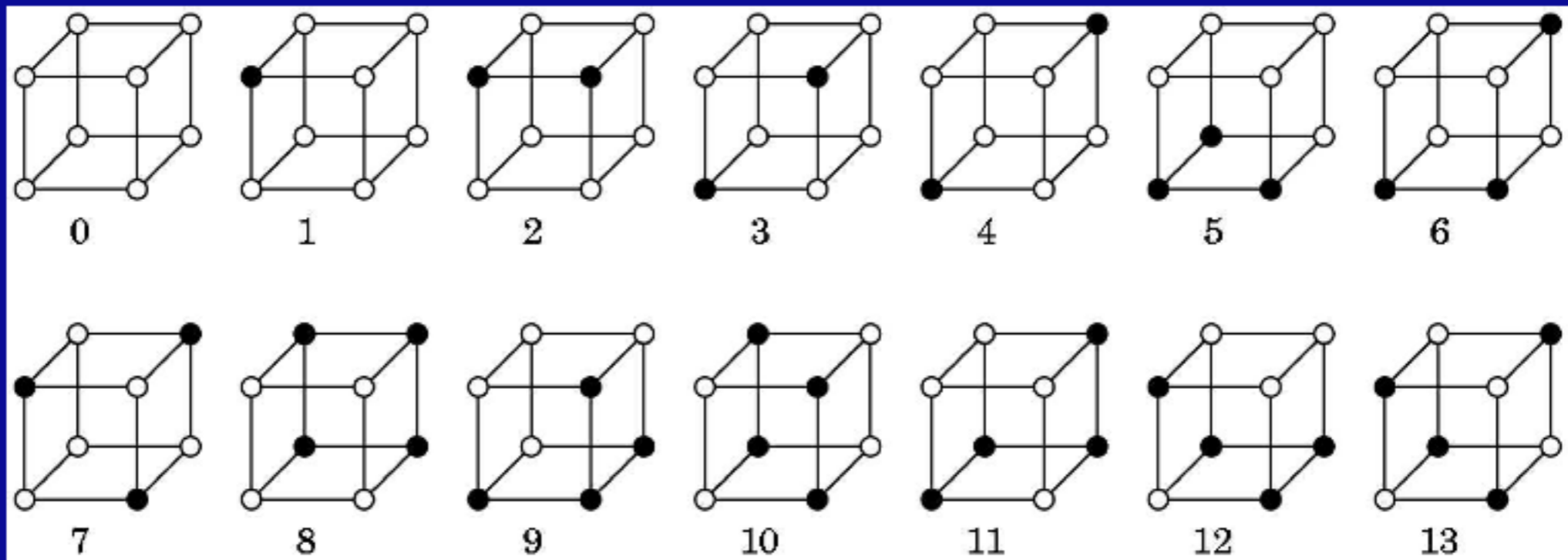
- Display technique for isosurfaces
- 3D version of marching squares
- How many possible cases?



$$2^8 = 256$$

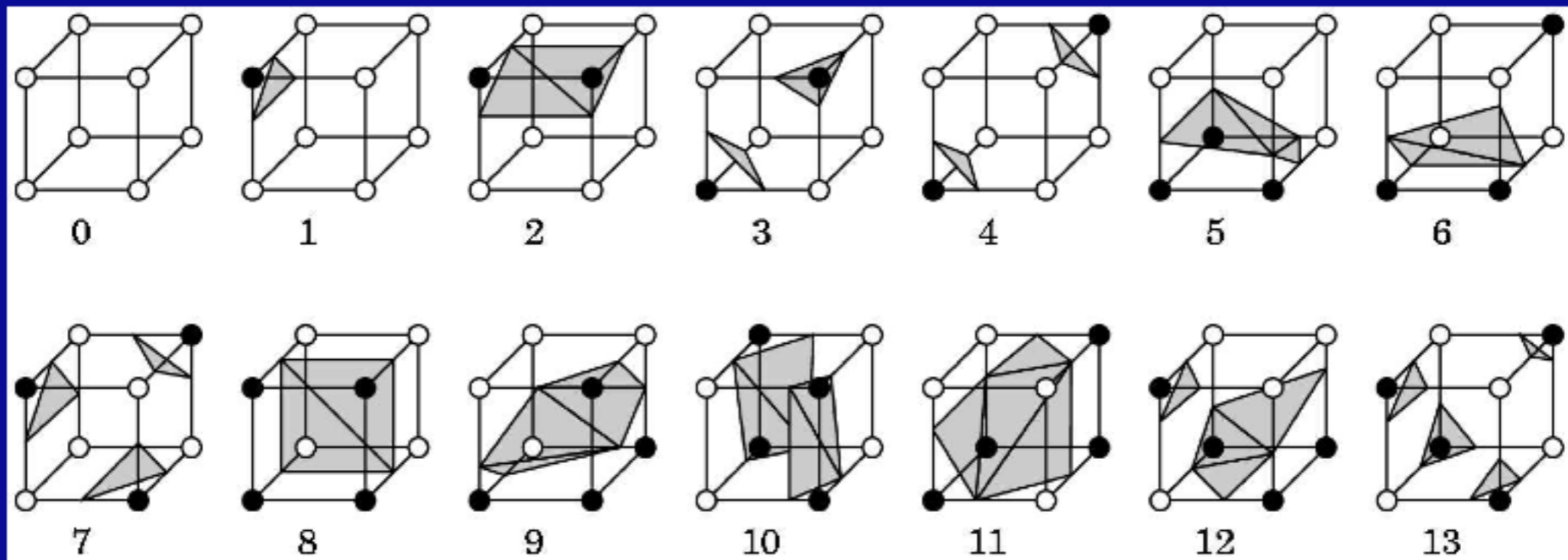
# Marching Cubes

- 14 cube labelings (after elimination symmetries)

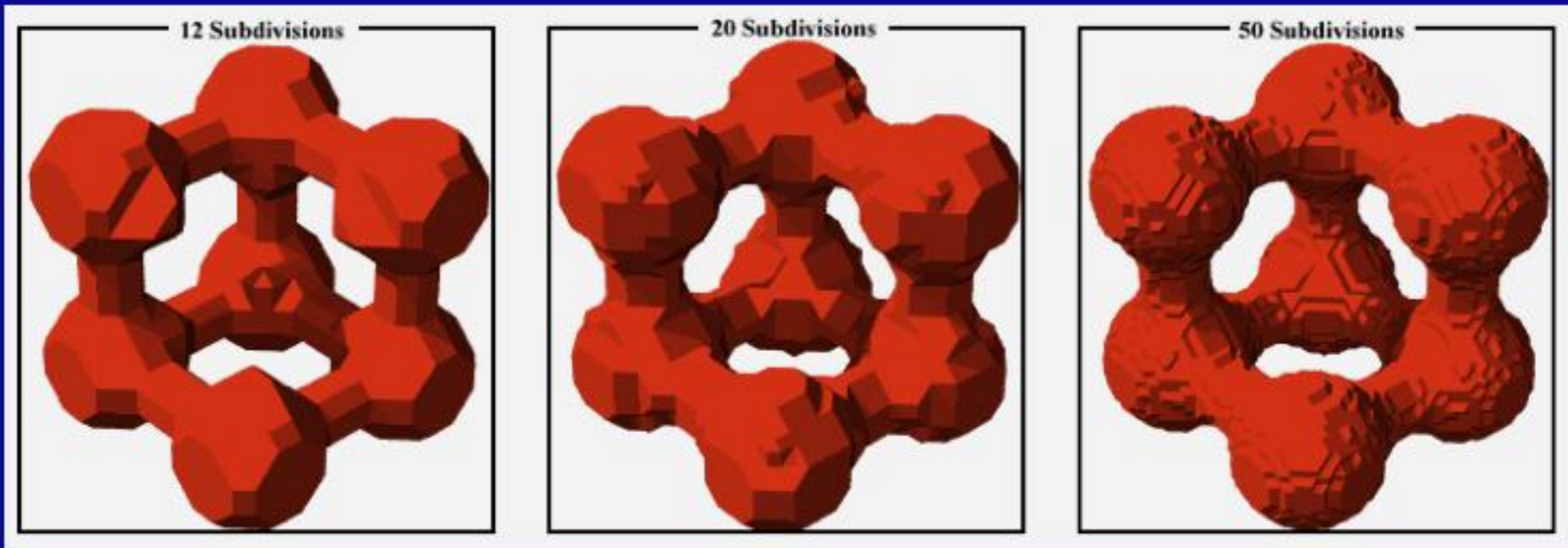
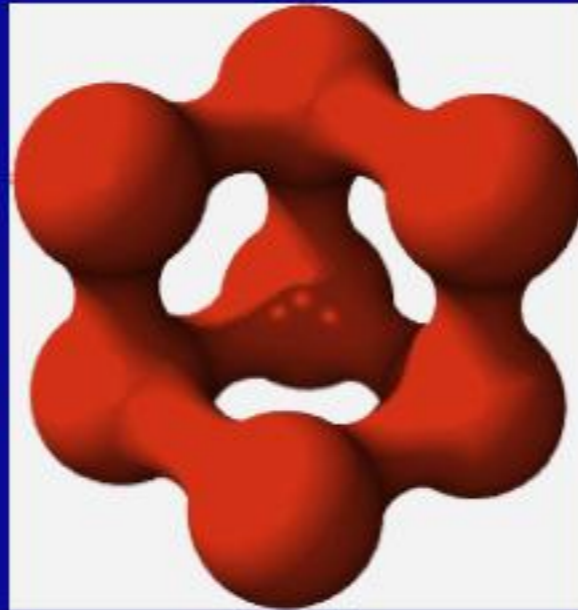


# Marching Cube Tessellations

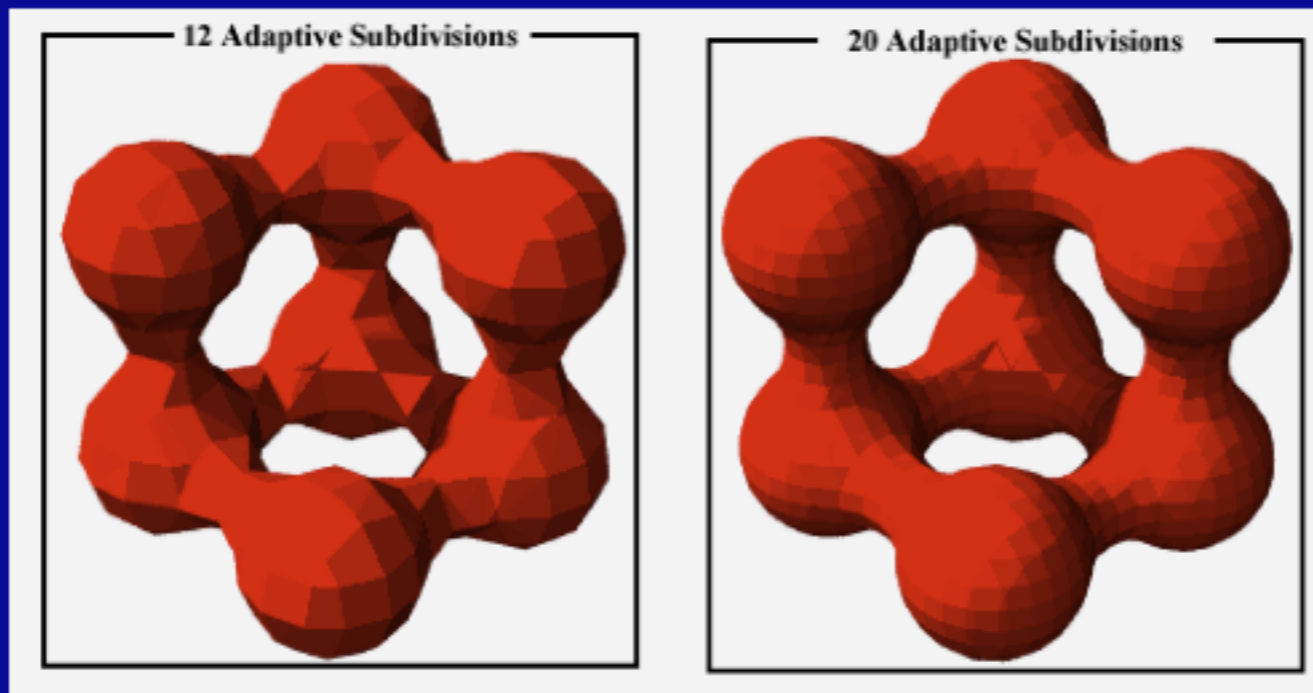
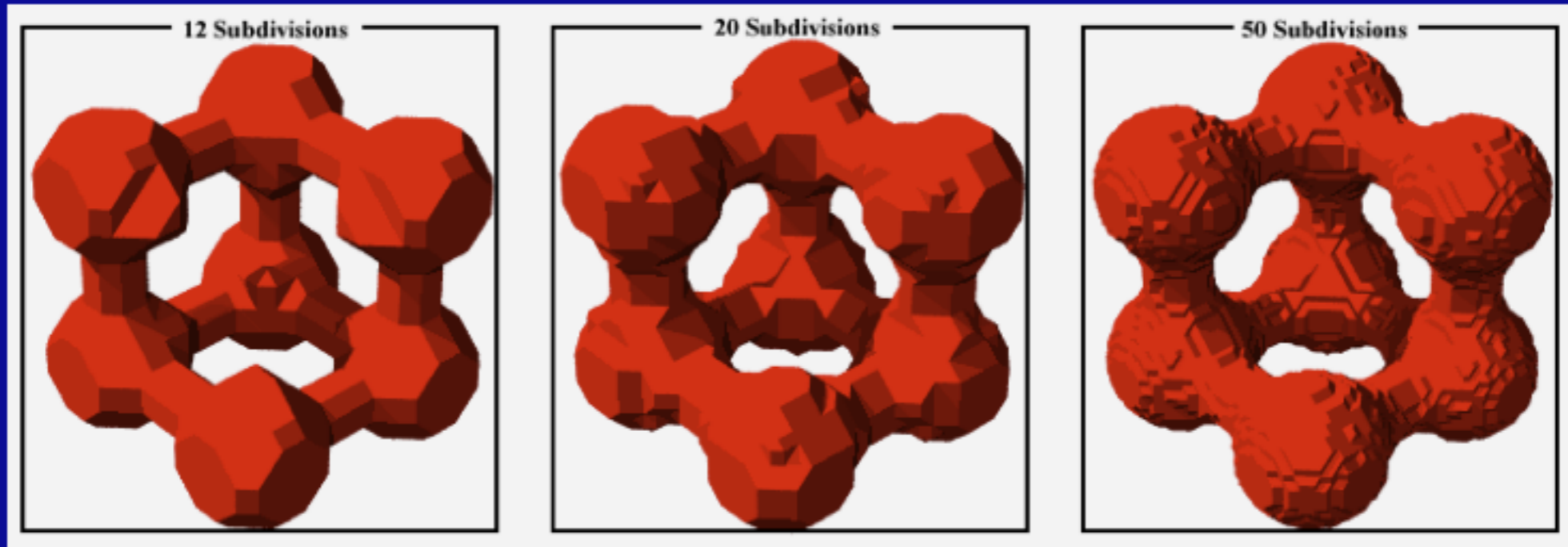
- Generalize marching squares, just more cases
- Interpolate as in 2D
- Ambiguities similar to 2D



# Marching Squares Examples



# Marching Squares Examples



# 3D Object Representations

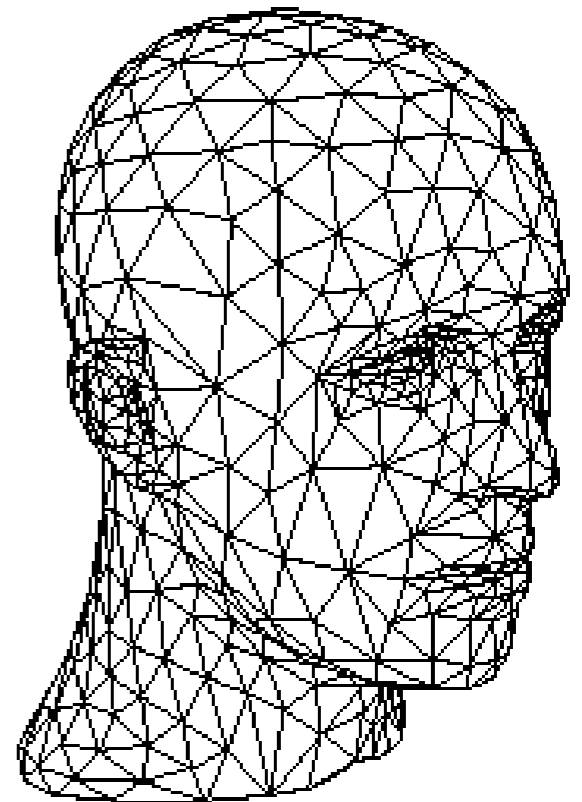


- Raw data
  - o Voxels
  - o Point cloud
  - o Range image
  - o Polygons
- Surfaces
  - o Mesh
  - o Subdivision
  - o Parametric
  - o Implicit
- Solids
  - o Octree
  - o BSP tree
  - o CSG
  - o Sweep
- High-level structures
  - o Scene graph
  - o Application specific

# Polygon Meshes

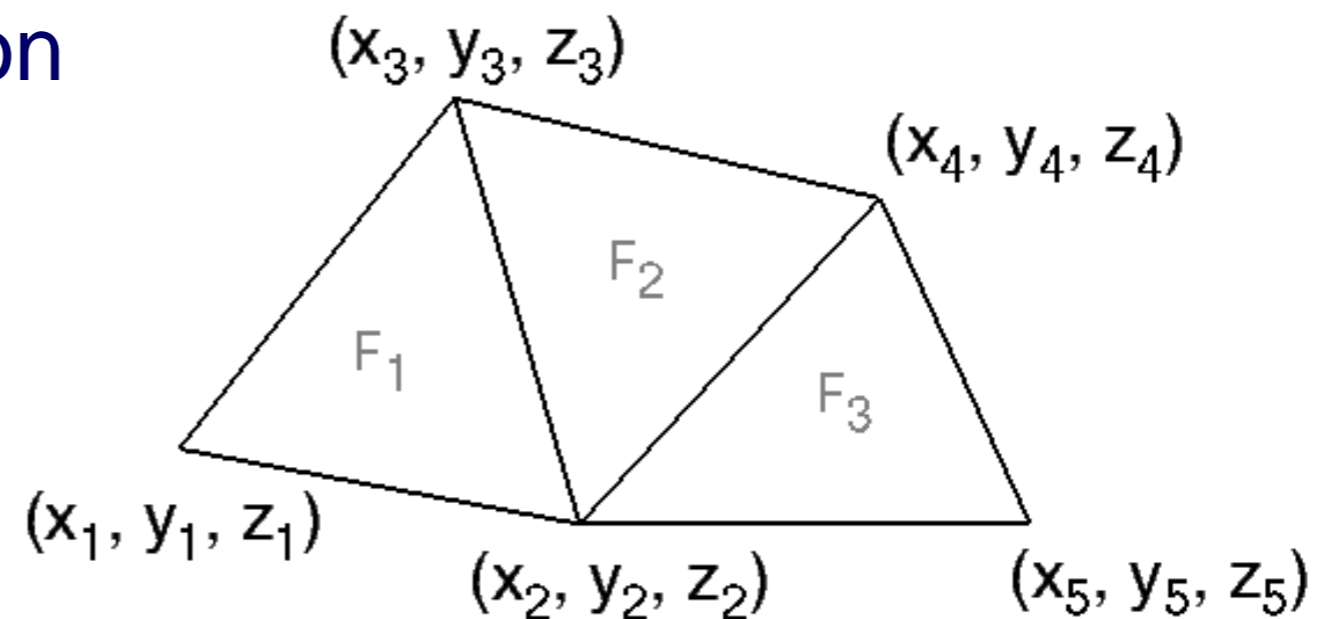
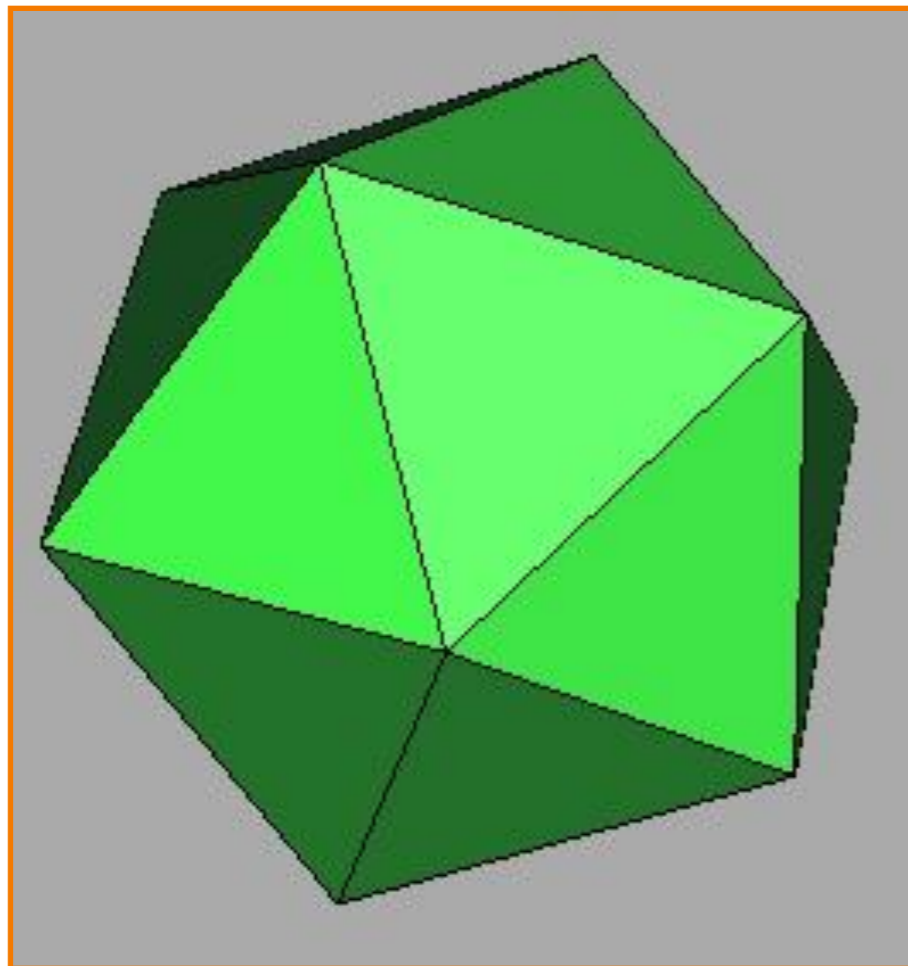


- How should we represent a mesh in a computer?
  - Efficient traversal of topology
  - Efficient use of memory
  - Efficient updates
- Mesh Representations
  - Independent faces
  - Vertex and face tables
  - Adjacency lists
  - Winged-Edge
  - Half-Edge
  - etc.



# Independent Faces

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

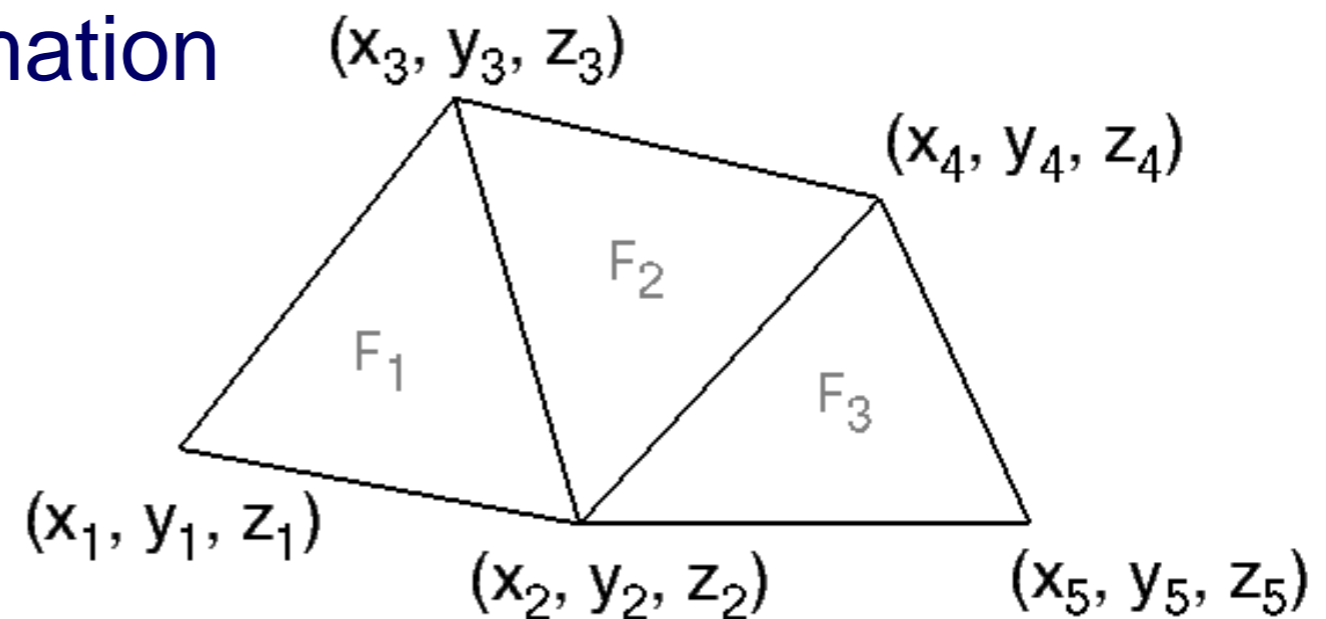
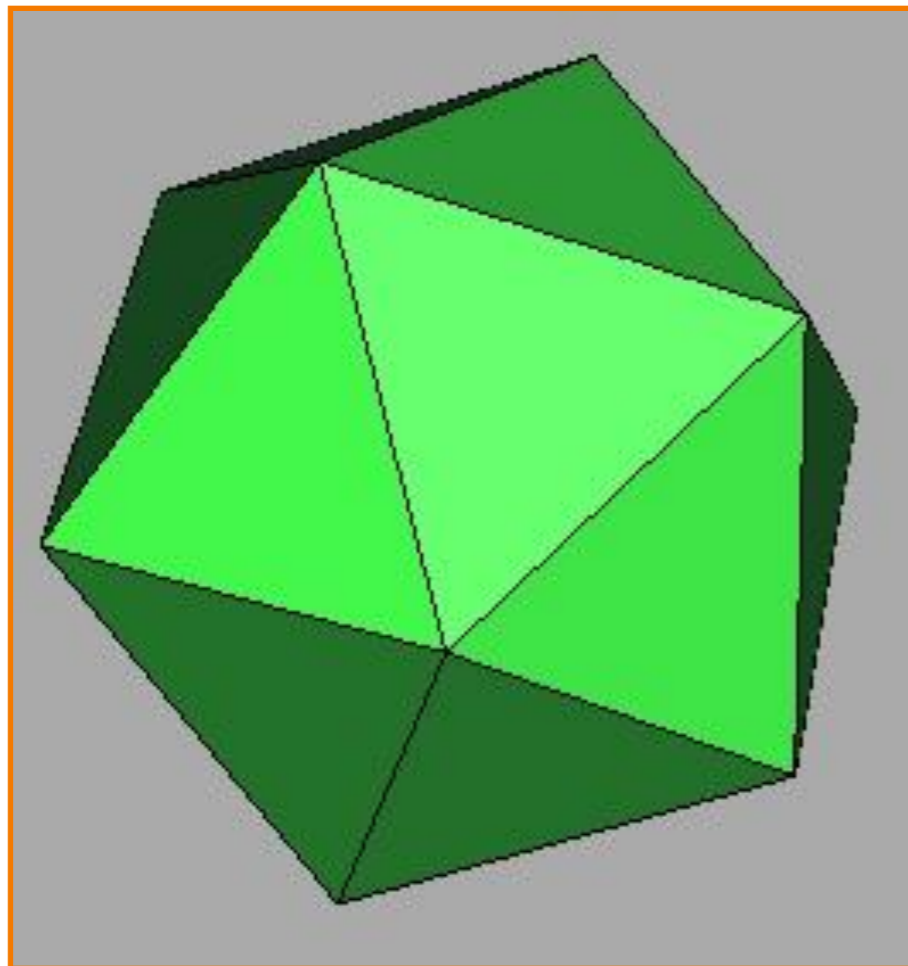


FACE TABLE

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

# Vertex and Face Tables

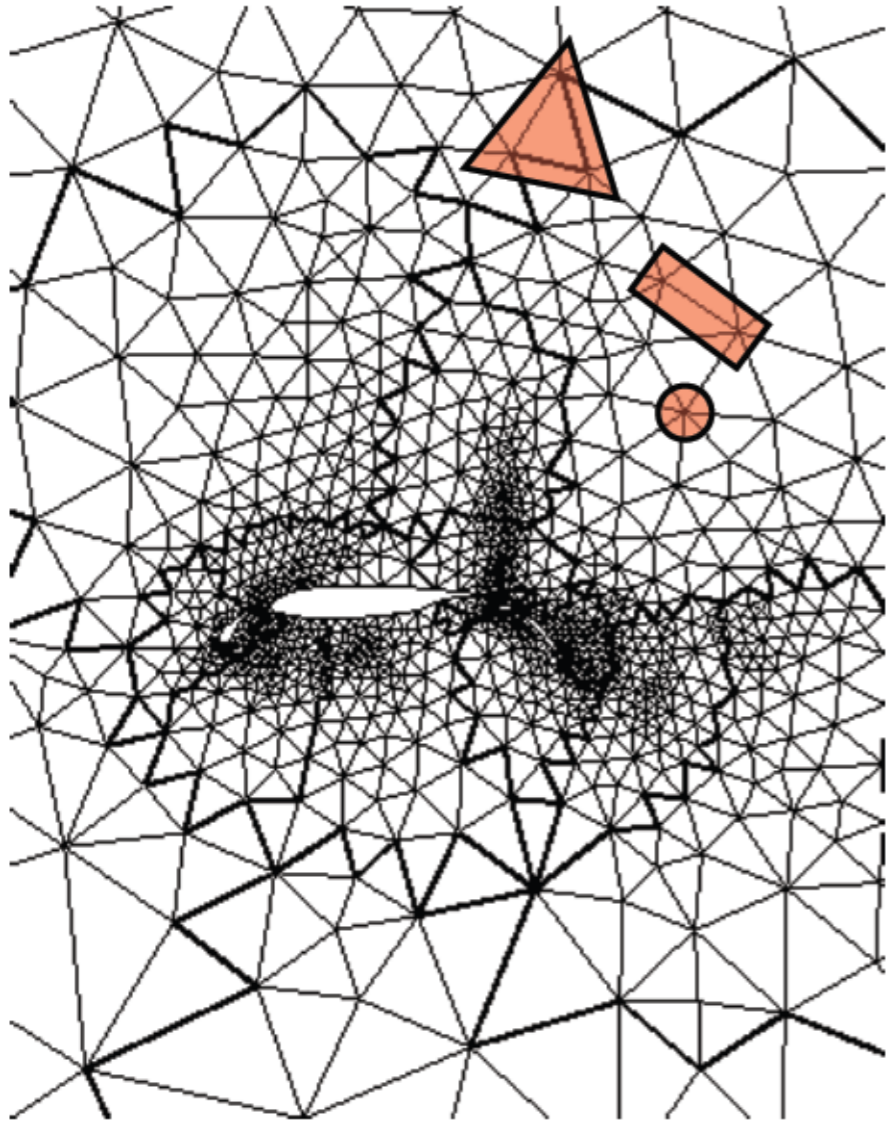
- Each face lists vertex references
  - Shared vertices
  - Still no adjacency information



VERTEX TABLE			
V <sub>1</sub>	X <sub>1</sub>	Y <sub>1</sub>	Z <sub>1</sub>
V <sub>2</sub>	X <sub>2</sub>	Y <sub>2</sub>	Z <sub>2</sub>
V <sub>3</sub>	X <sub>3</sub>	Y <sub>3</sub>	Z <sub>3</sub>
V <sub>4</sub>	X <sub>4</sub>	Y <sub>4</sub>	Z <sub>4</sub>
V <sub>5</sub>	X <sub>5</sub>	Y <sub>5</sub>	Z <sub>5</sub>

FACE TABLE			
F <sub>1</sub>	V <sub>1</sub>	V <sub>2</sub>	V <sub>3</sub>
F <sub>2</sub>	V <sub>2</sub>	V <sub>4</sub>	V <sub>3</sub>
F <sub>3</sub>	V <sub>2</sub>	V <sub>5</sub>	V <sub>4</sub>

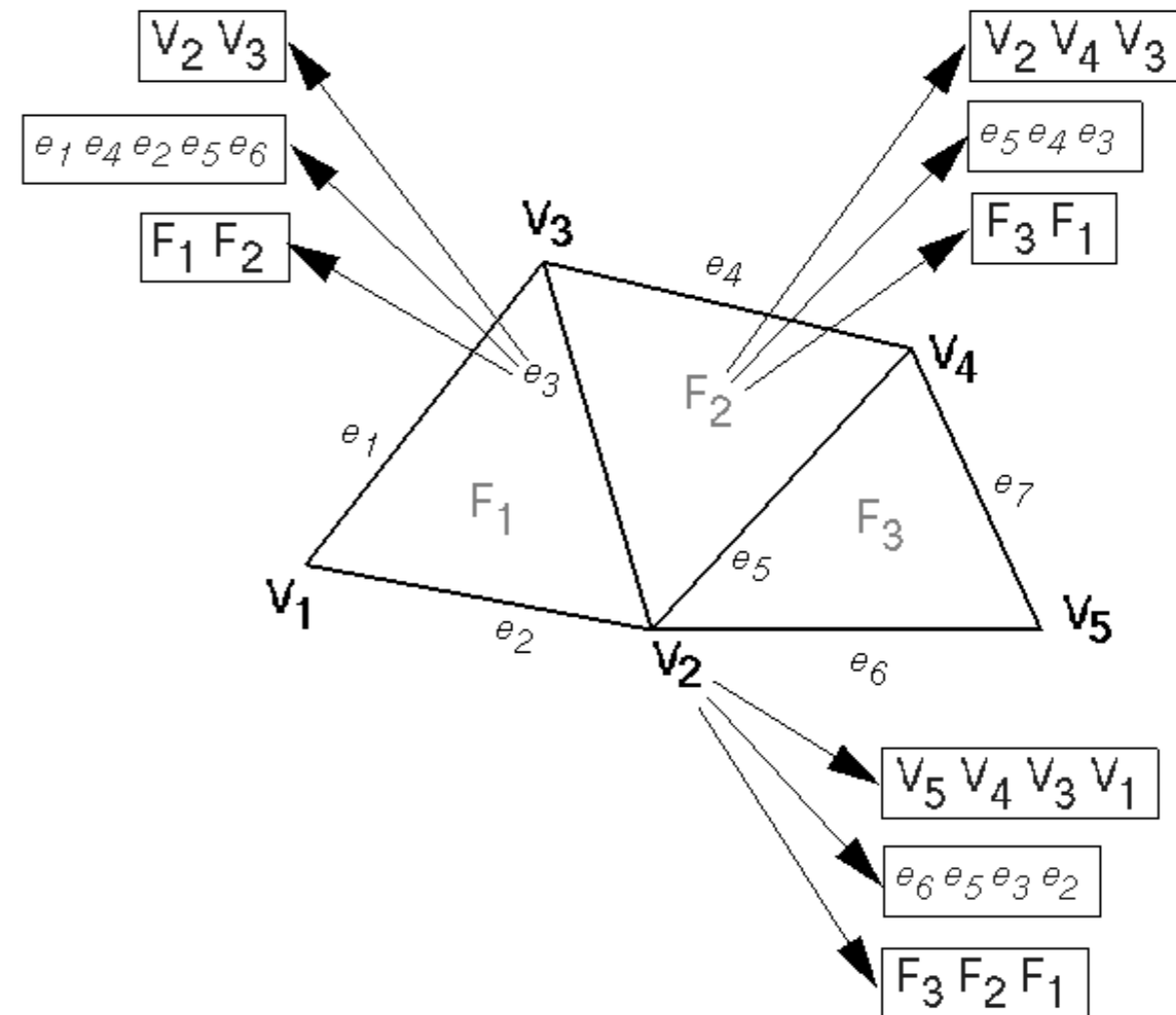
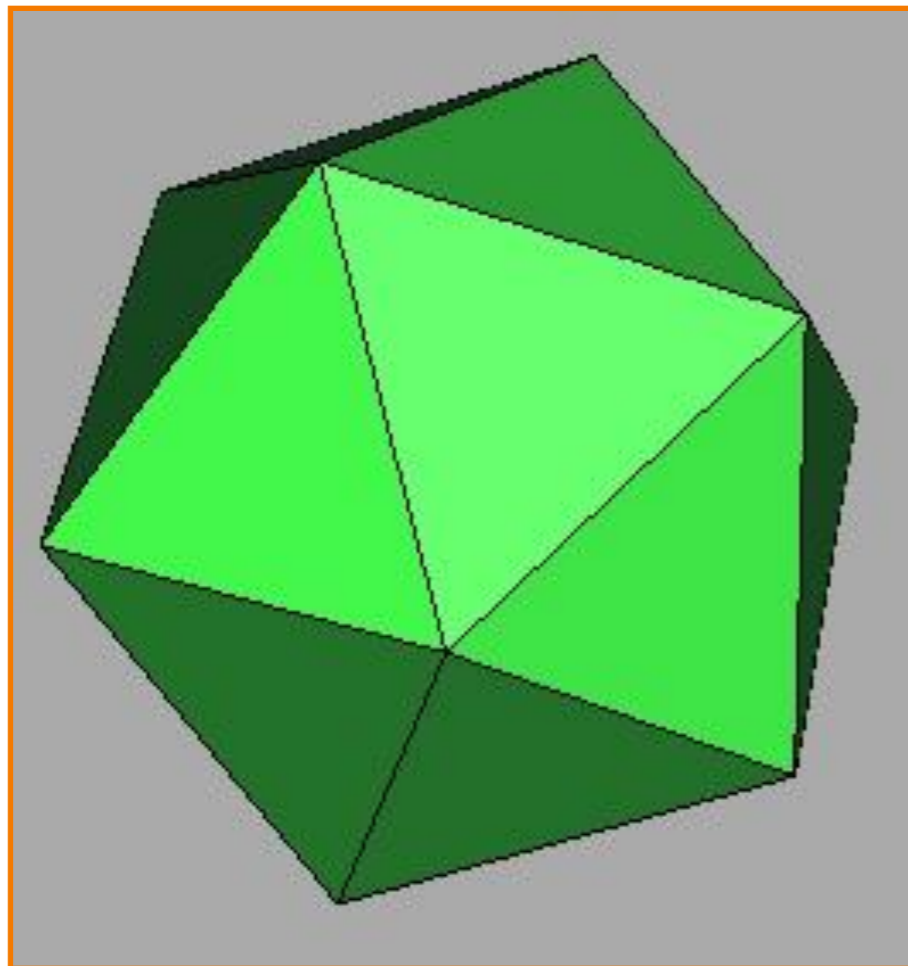
# Possible Queries



- 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?

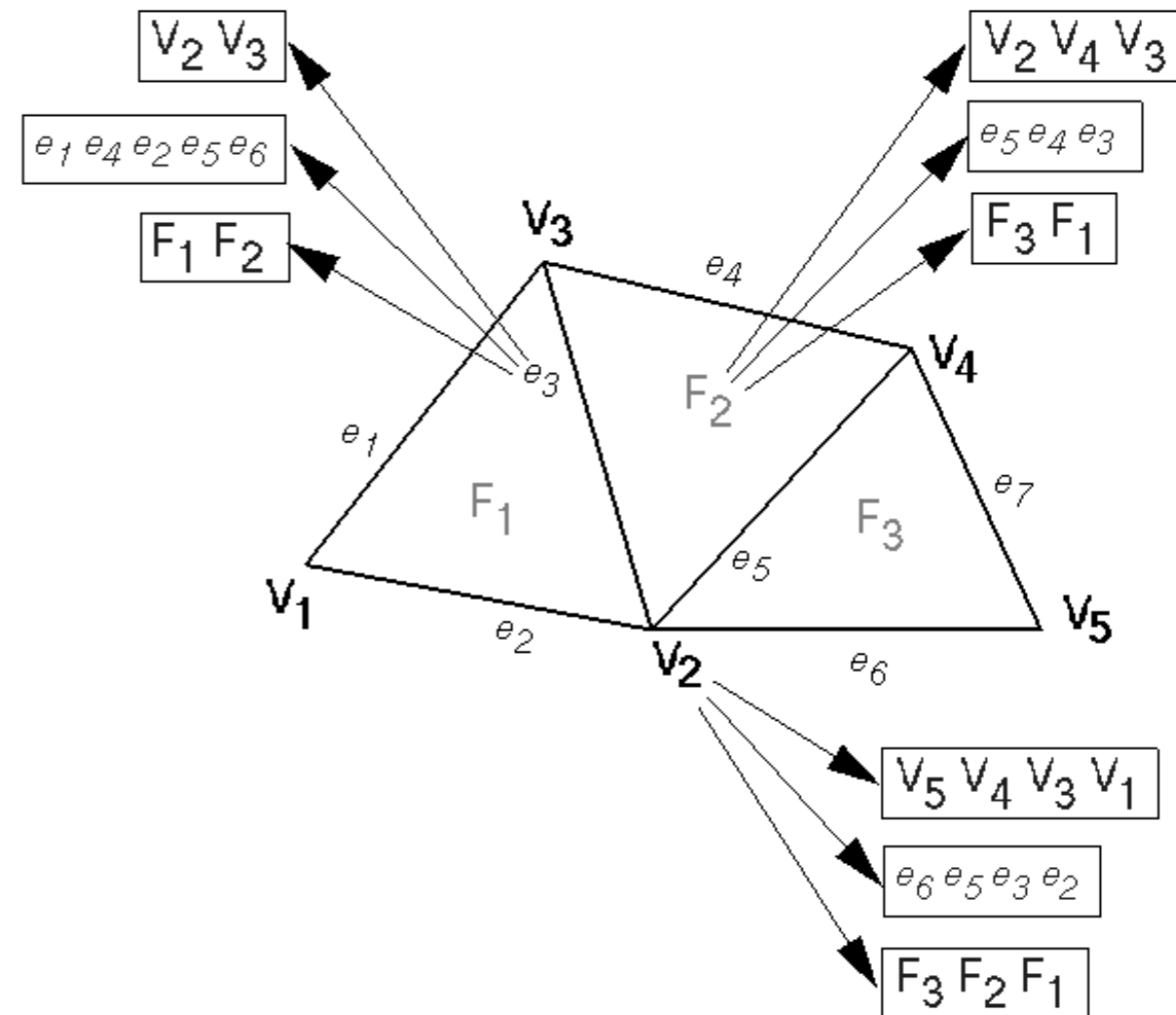
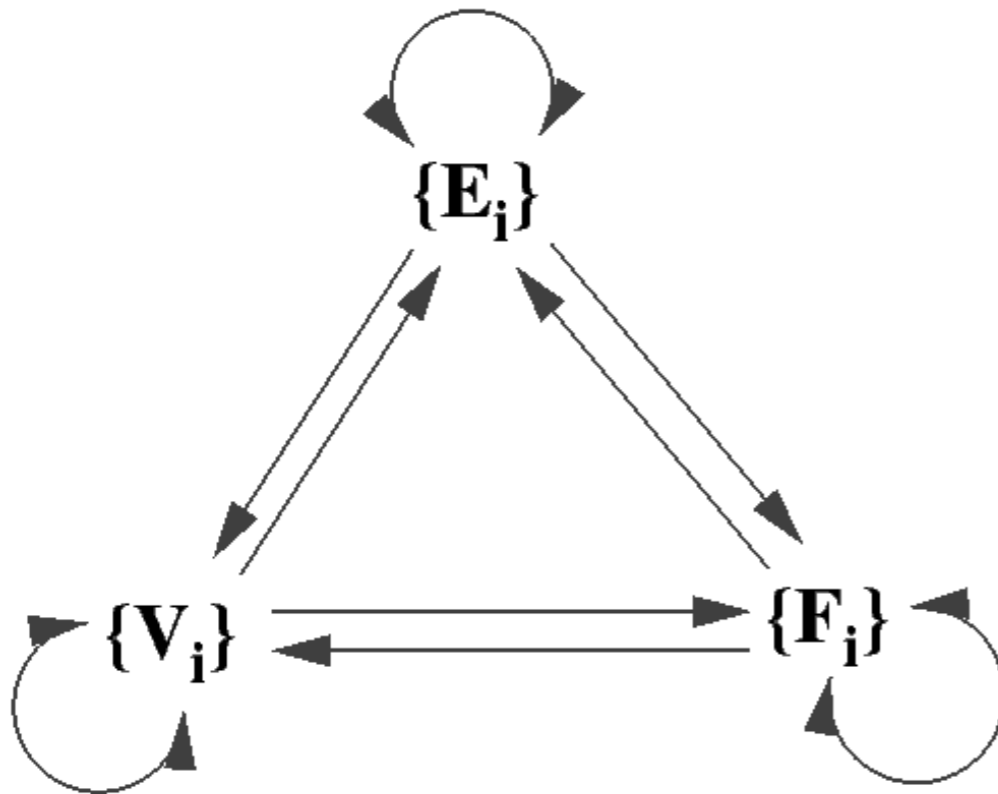
# Adjacency Lists

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



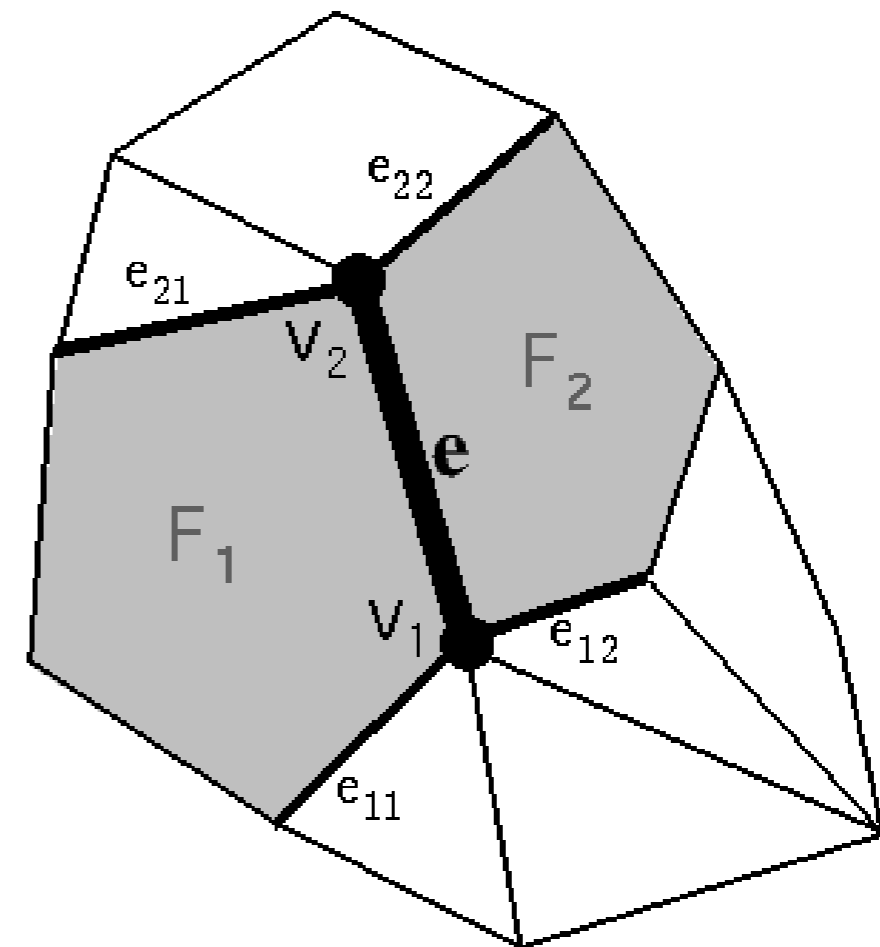
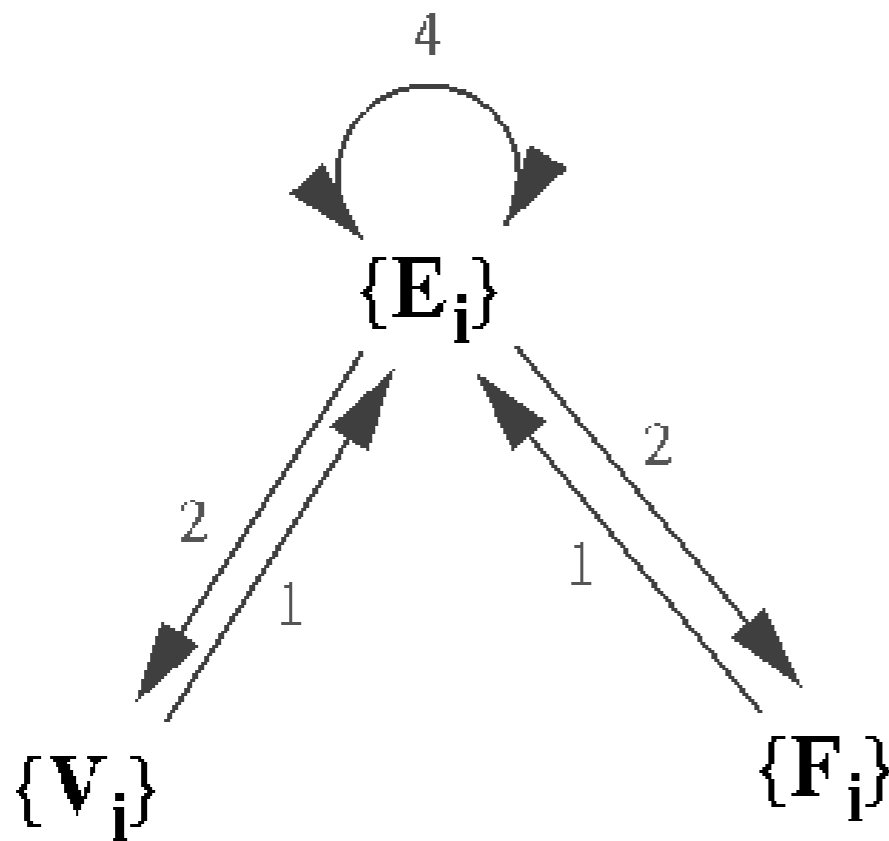
# Partial Adjacency Lists

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



# Winged Edge

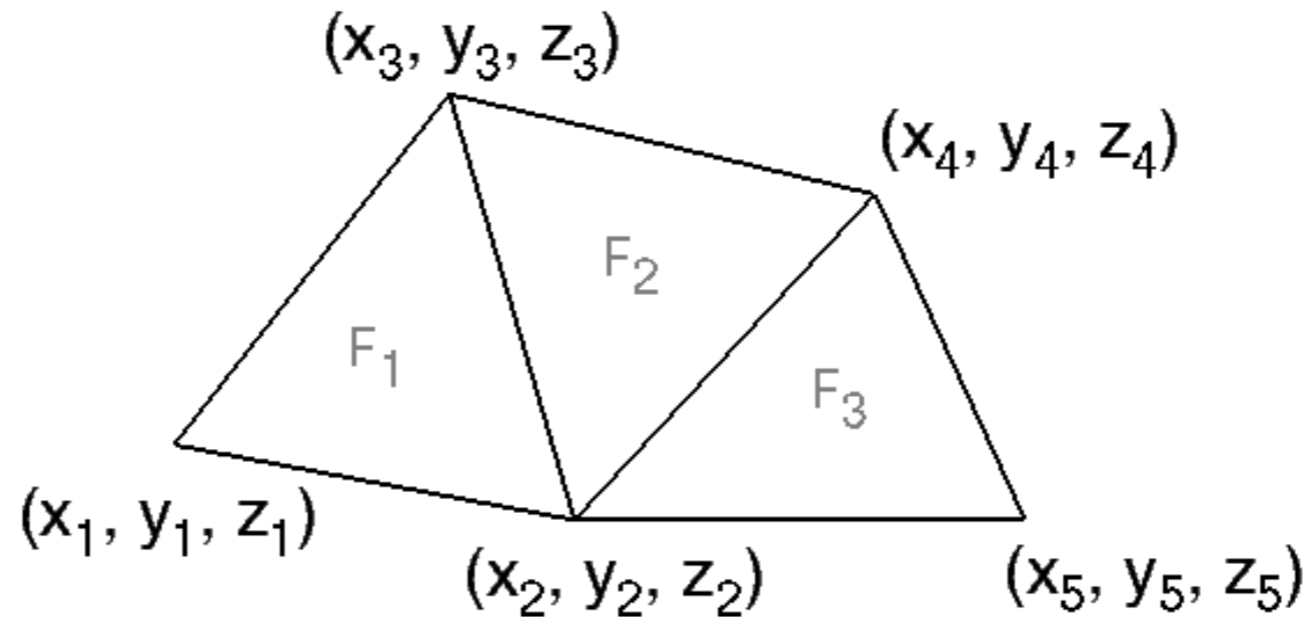
- Adjacency encoded in edges
  - All adjacencies in  $O(1)$  time
  - Little extra storage (fixed records)
  - Arbitrary polygons



# Winged Edge



- Example:



VERTEX TABLE				
V <sub>1</sub>	X <sub>1</sub>	Y <sub>1</sub>	Z <sub>1</sub>	e <sub>1</sub>
V <sub>2</sub>	X <sub>2</sub>	Y <sub>2</sub>	Z <sub>2</sub>	e <sub>6</sub>
V <sub>3</sub>	X <sub>3</sub>	Y <sub>3</sub>	Z <sub>3</sub>	e <sub>3</sub>
V <sub>4</sub>	X <sub>4</sub>	Y <sub>4</sub>	Z <sub>4</sub>	e <sub>5</sub>
V <sub>5</sub>	X <sub>5</sub>	Y <sub>5</sub>	Z <sub>5</sub>	e <sub>6</sub>

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

FACE TABLE	
F <sub>1</sub>	e <sub>1</sub>
F <sub>2</sub>	e <sub>3</sub>
F <sub>3</sub>	e <sub>5</sub>

**Think about accessing:**

**faces adjacent to the current face**

**vertices connected to the current vertex**

**More info and examples for winged edge:**

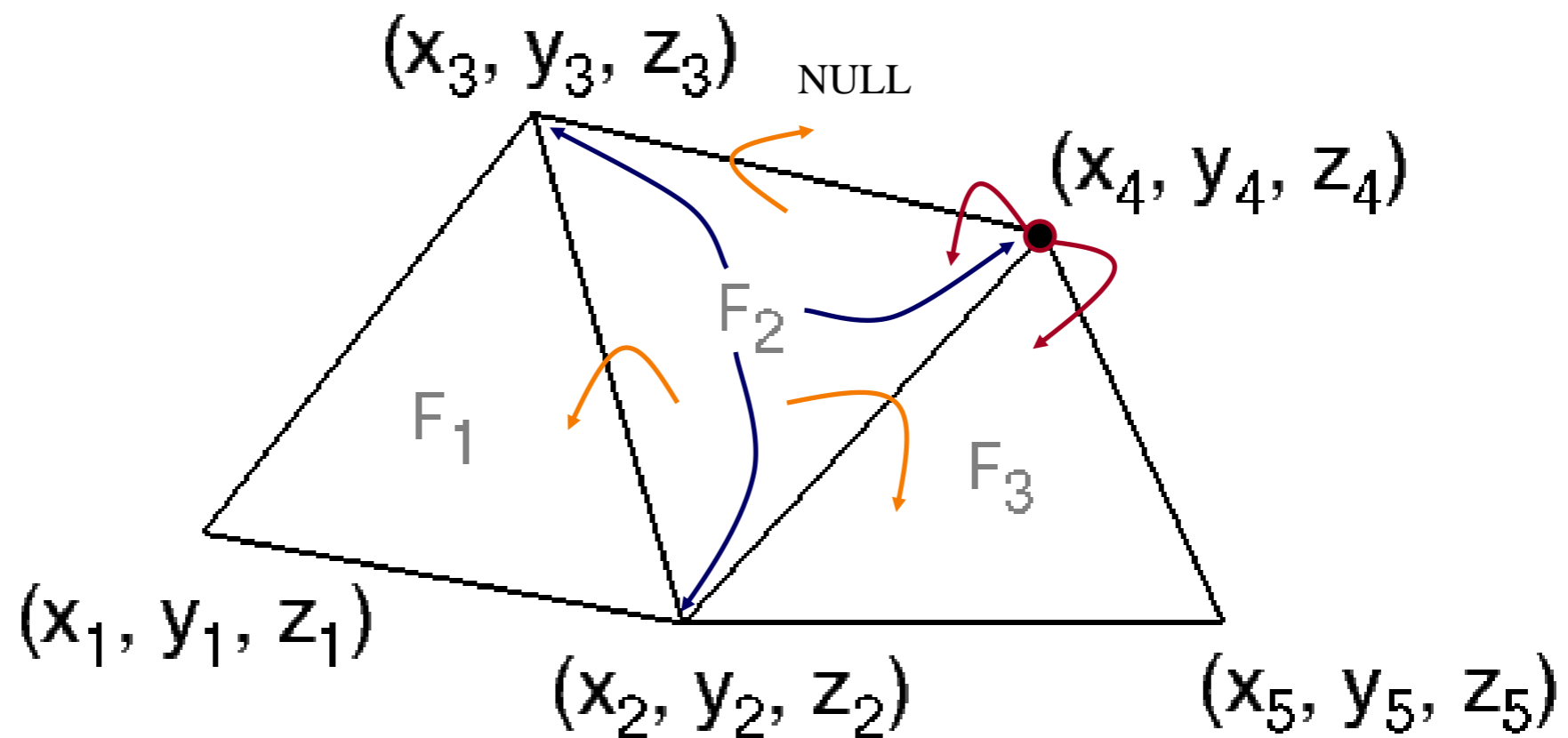
<http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/winged-e.html>

<http://ezekiel.vancouver.wsu.edu/~cs442/lectures/winged-edge/winged-edge.pdf>



# 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



**Accessing:**

**faces adjacent to the current face**

**vertices connected to the current vertex**

# 3D Object Representations

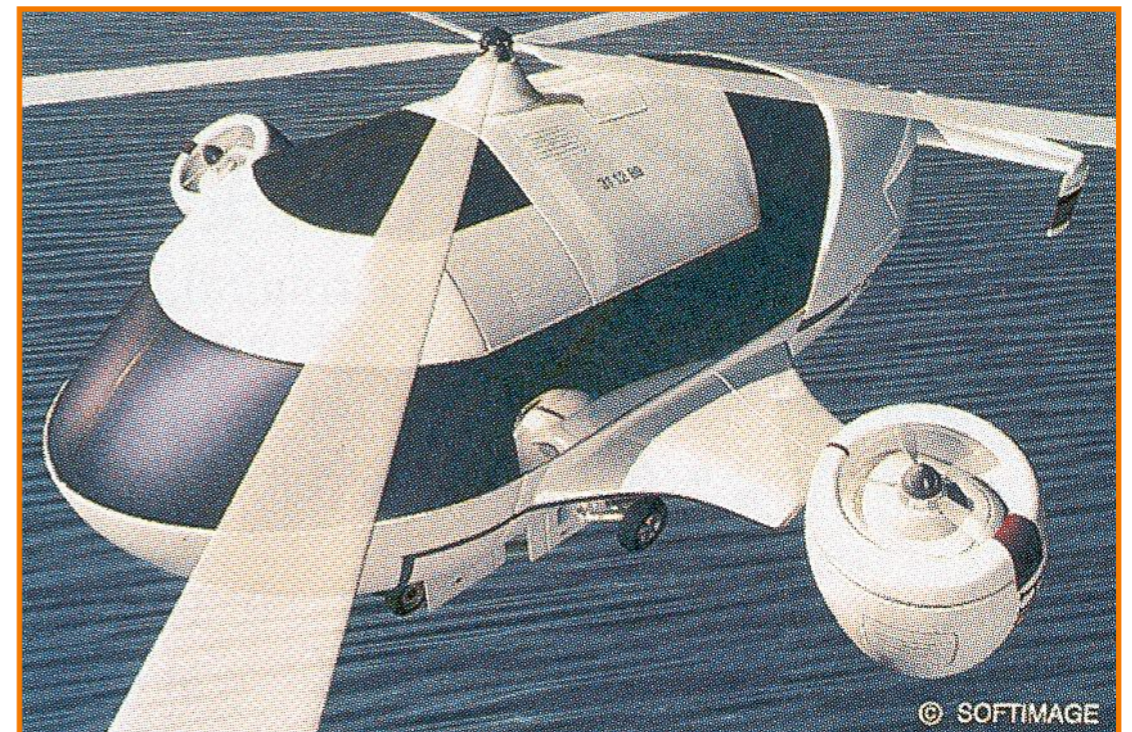


- Raw data
  - o Voxels
  - o Point cloud
  - o Range image
  - o Polygons
- Surfaces
  - o Mesh
  - o Subdivision
  - o Parametric
  - o Implicit
- Solids
  - o Octree
  - o BSP tree
  - o CSG
  - o Sweep
- High-level structures
  - o Scene graph
  - o Application specific

# 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



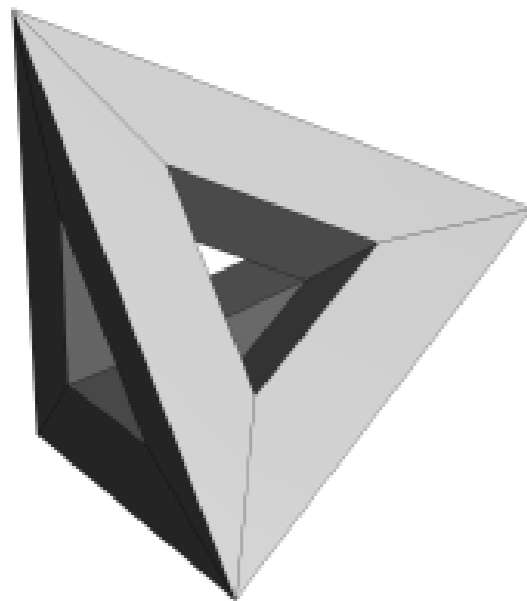
- How do you make a smooth curve?



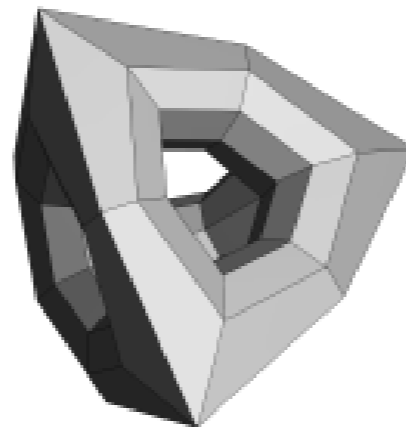
# Subdivision Surfaces



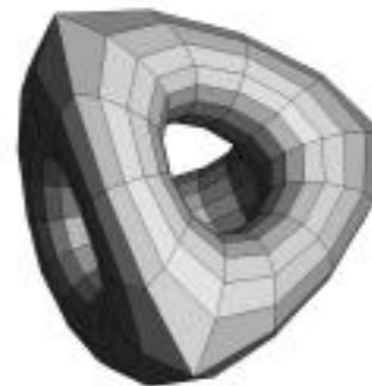
- Coarse mesh & subdivision rule
  - Define smooth surface as limit of sequence of refinements



(a)



(b)



(c)

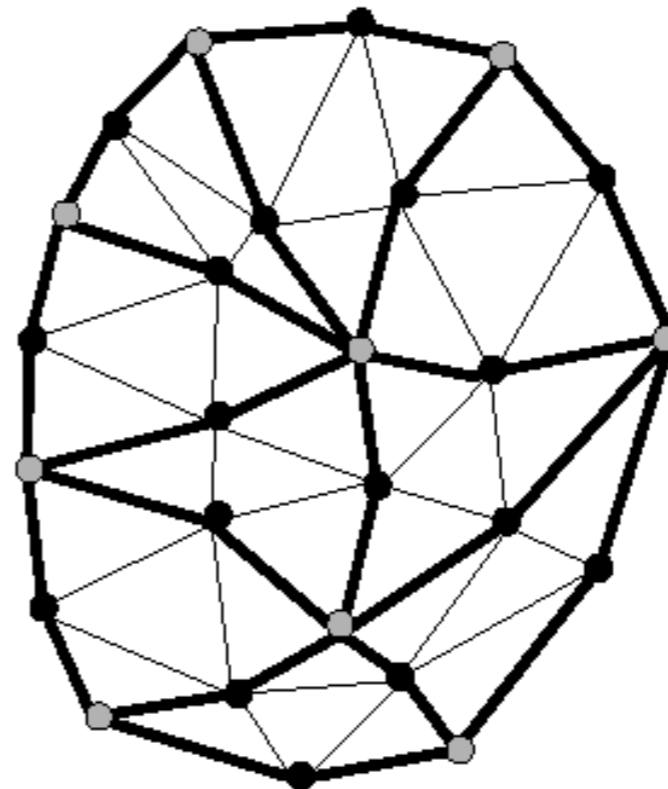
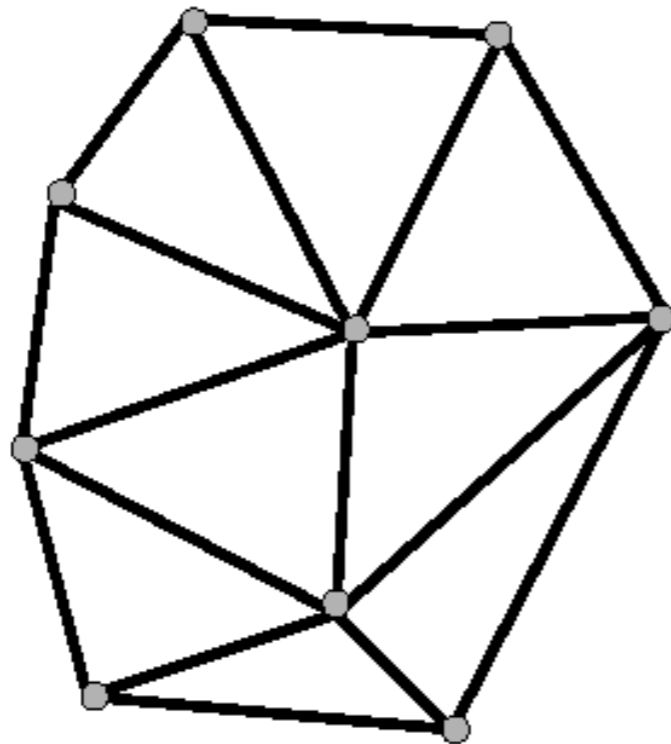


(d)

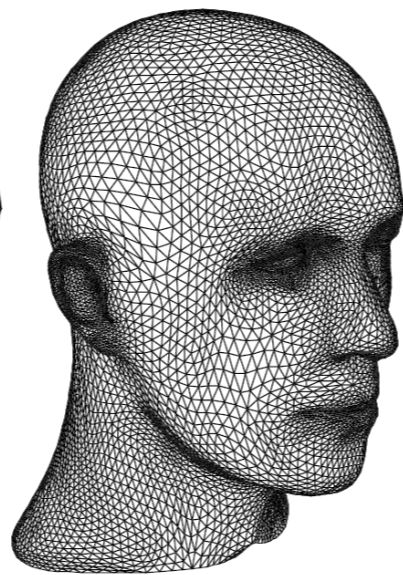
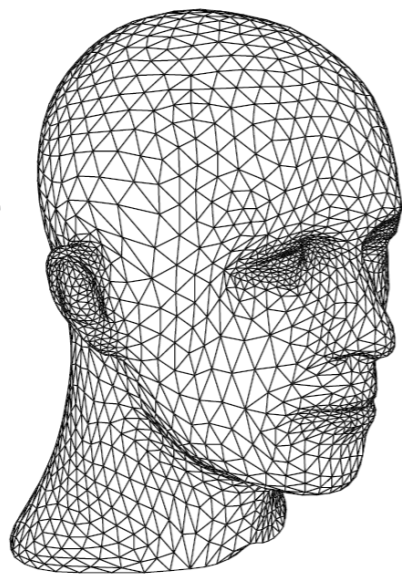
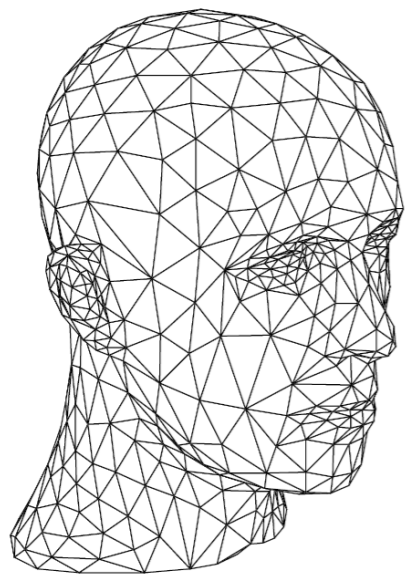
# Key Questions



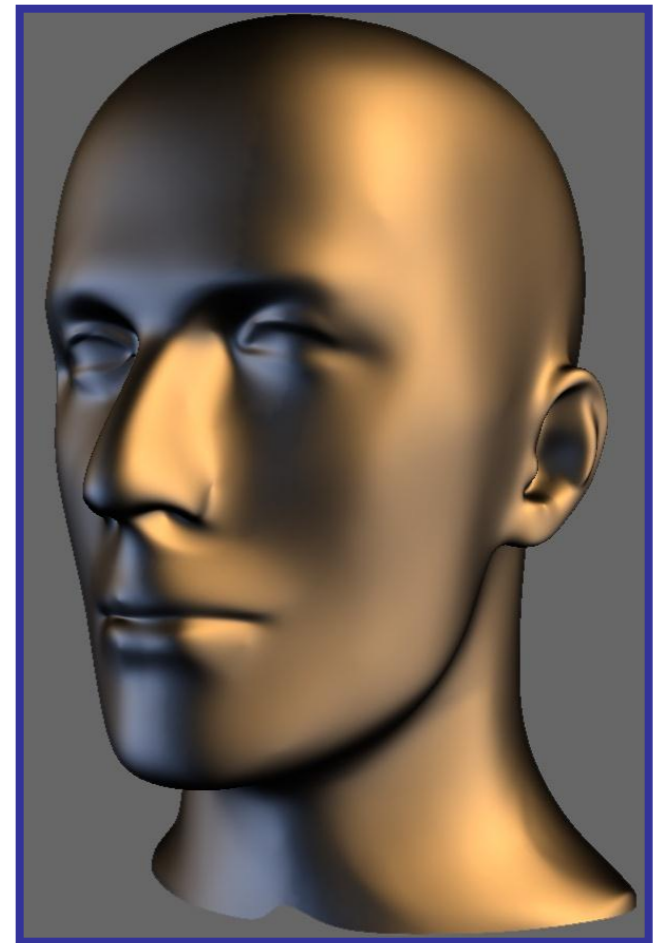
- How refine mesh?
  - Aim for properties like smoothness
- How store mesh?
  - Aim for efficiency for implementing subdivision rules



# Subdivision Surfaces – A 3D example



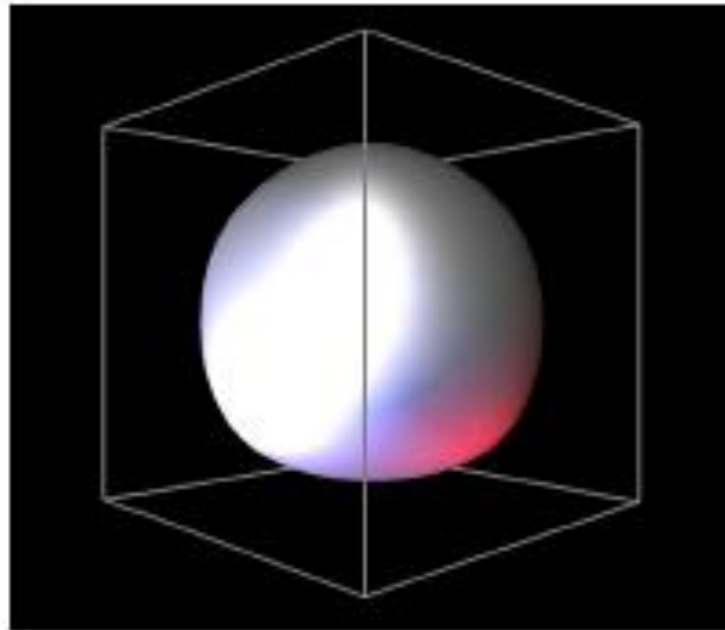
...



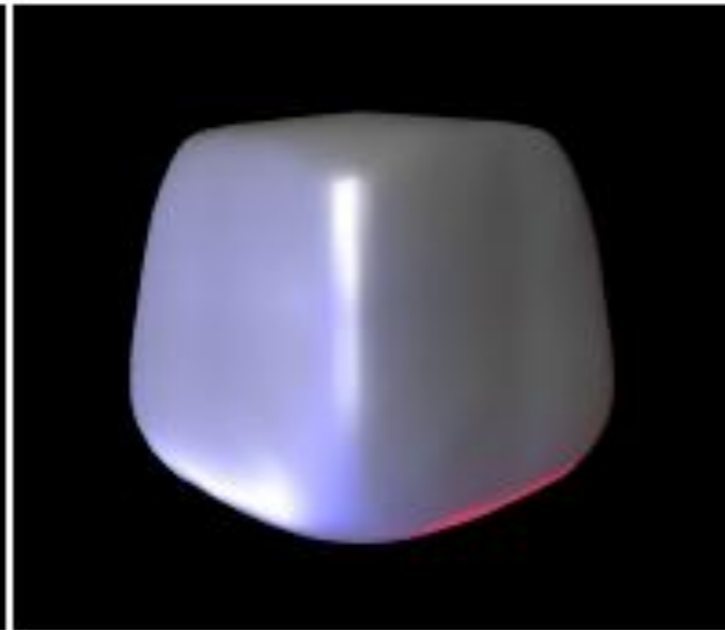
# Applications: Computer Graphics Animation



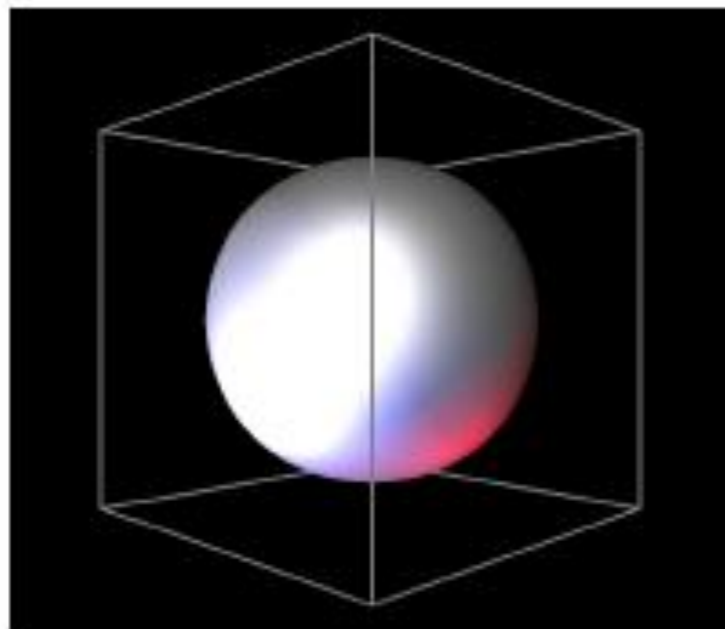
# Subdivision Schemes



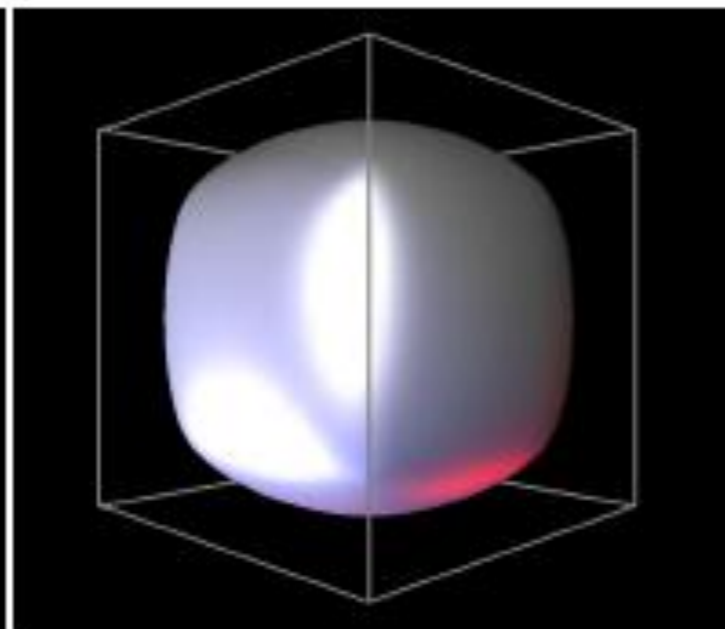
*Loop*



*Butterfly*

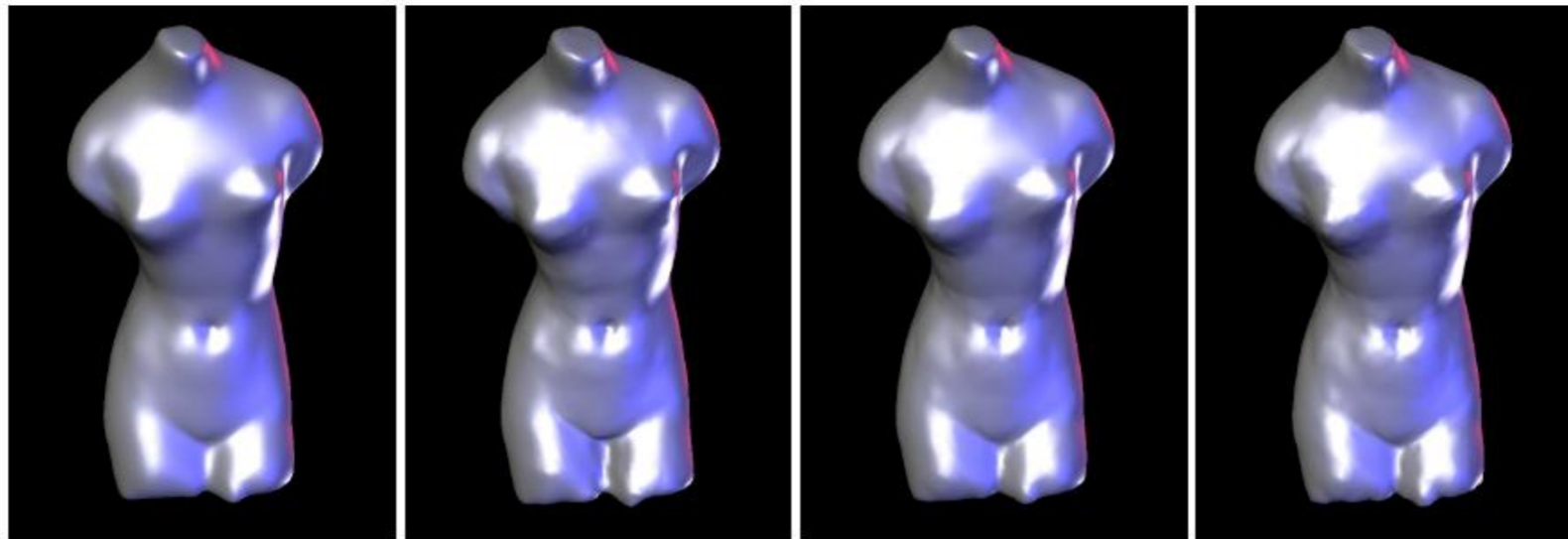


*Catmull-Clark*



*Doo-Sabin*

# Visual Comparison



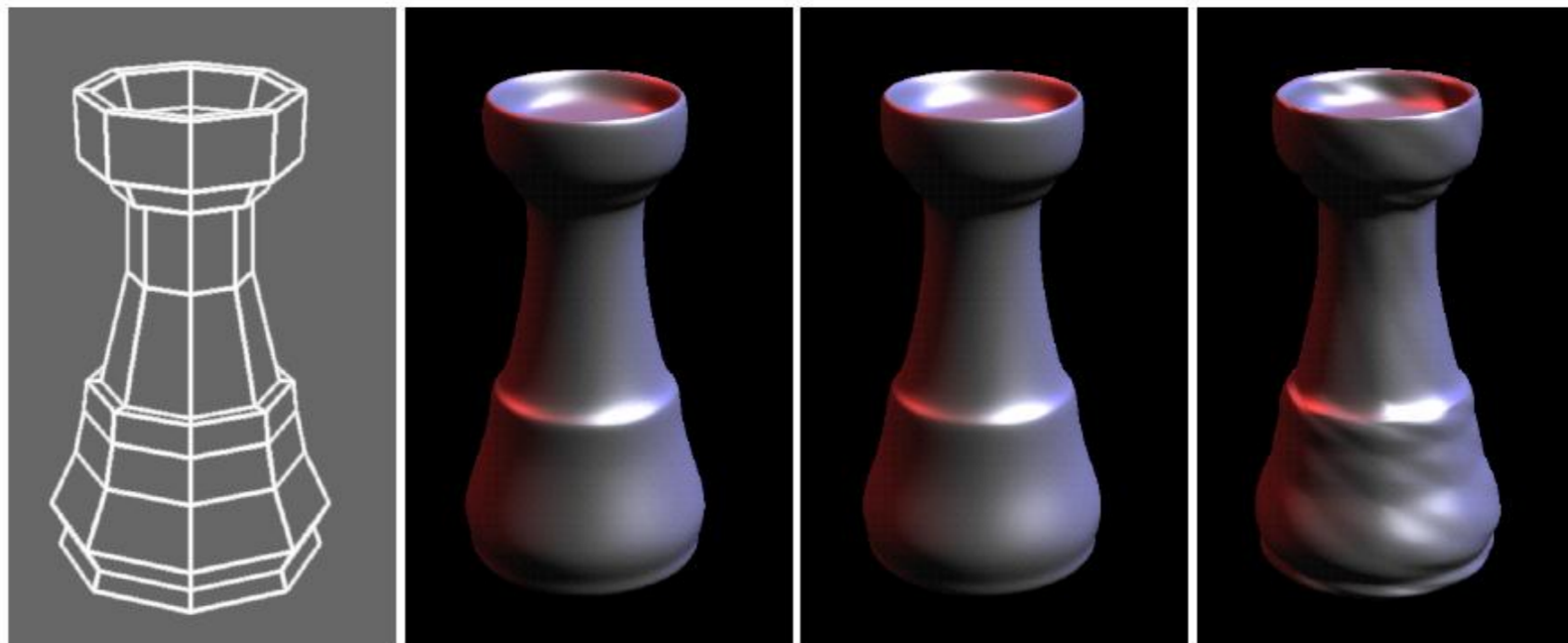
*Loop*

*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



# 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