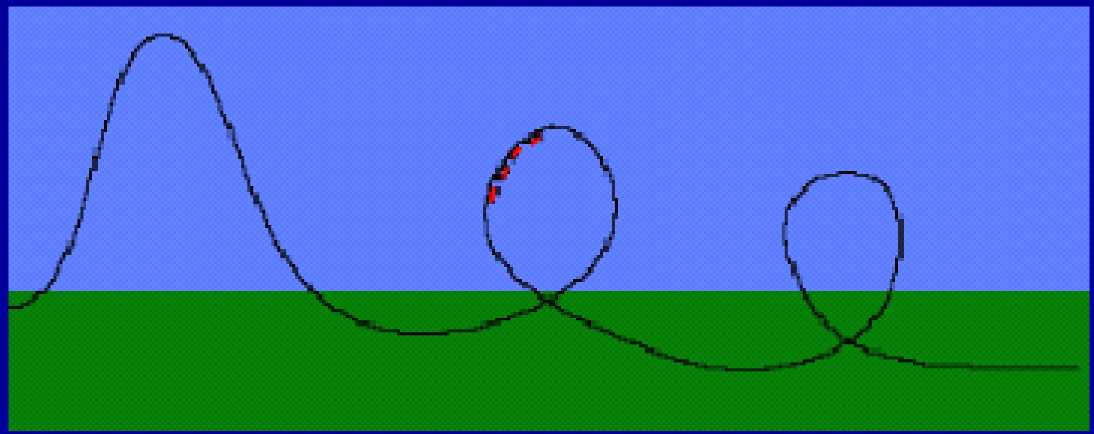
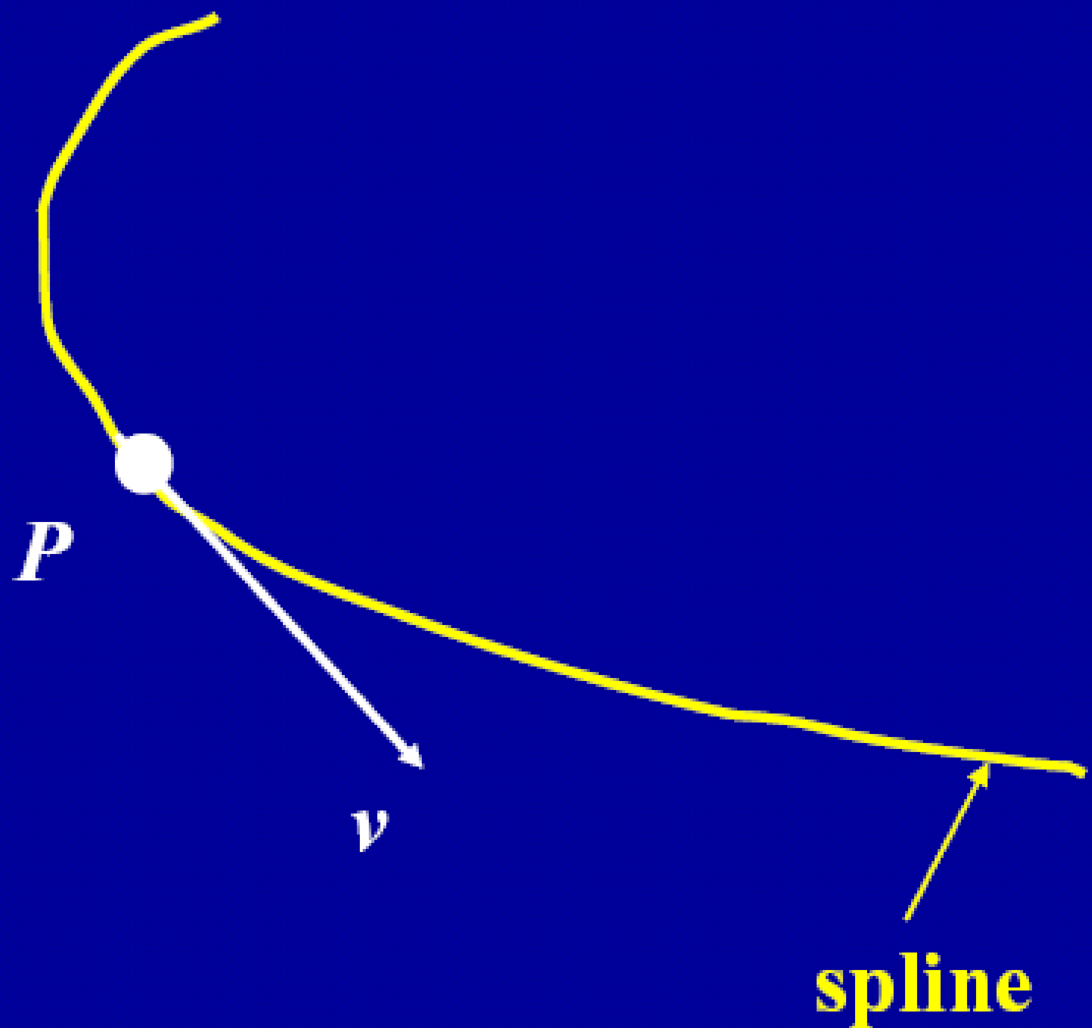
Physics of a Mass Point

Roller coaster

- Next programming assignment involves creating a 3D roller coaster animation
- We must model the 3D curve describing the roller coaster, but how?
- How to make the simulation obey the laws of gravity?



Back to the physics of the roller-coaster: mass point moving on a spline



frictionless model, with gravity

- Velocity vector always points in the tangential direction of the curve

Mass point on a spline (contd.)
frictionless model, with gravity

- **Our assumption is : no friction among the point and the spline**
- **Use the conservation of energy law to get the current velocity**

chalkboard

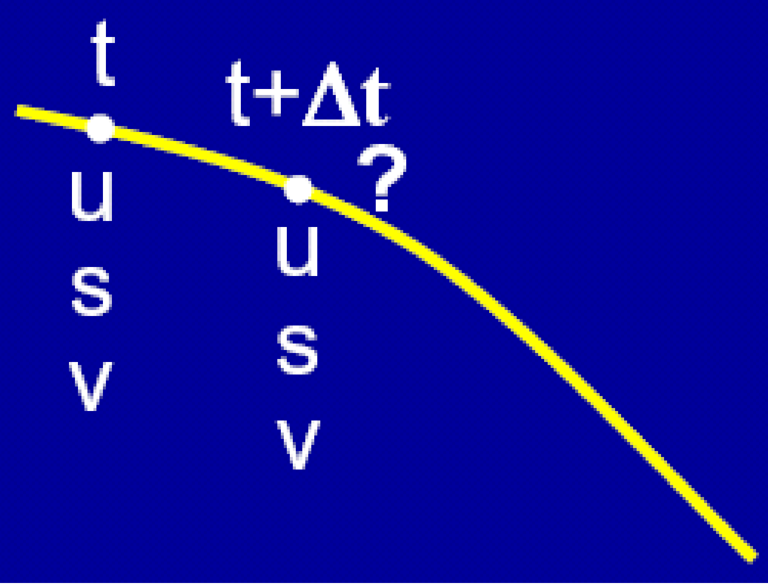
Mass point on a spline (contd.)
frictionless model, with gravity

- Given current h, we can always compute the corresponding |v|:

$$|v| = \sqrt{2g(h_{\max} - h)}$$

Simulating mass point on a spline

- Time step Δt
- We have: $\Delta s = |v| * \Delta t$ and $s = s + \Delta s$.
- We want the new value of u, so that can compute new point location
- Therefore:
  We know s, need to determine u
  Here we use the bisection routine to compute u=u(s).



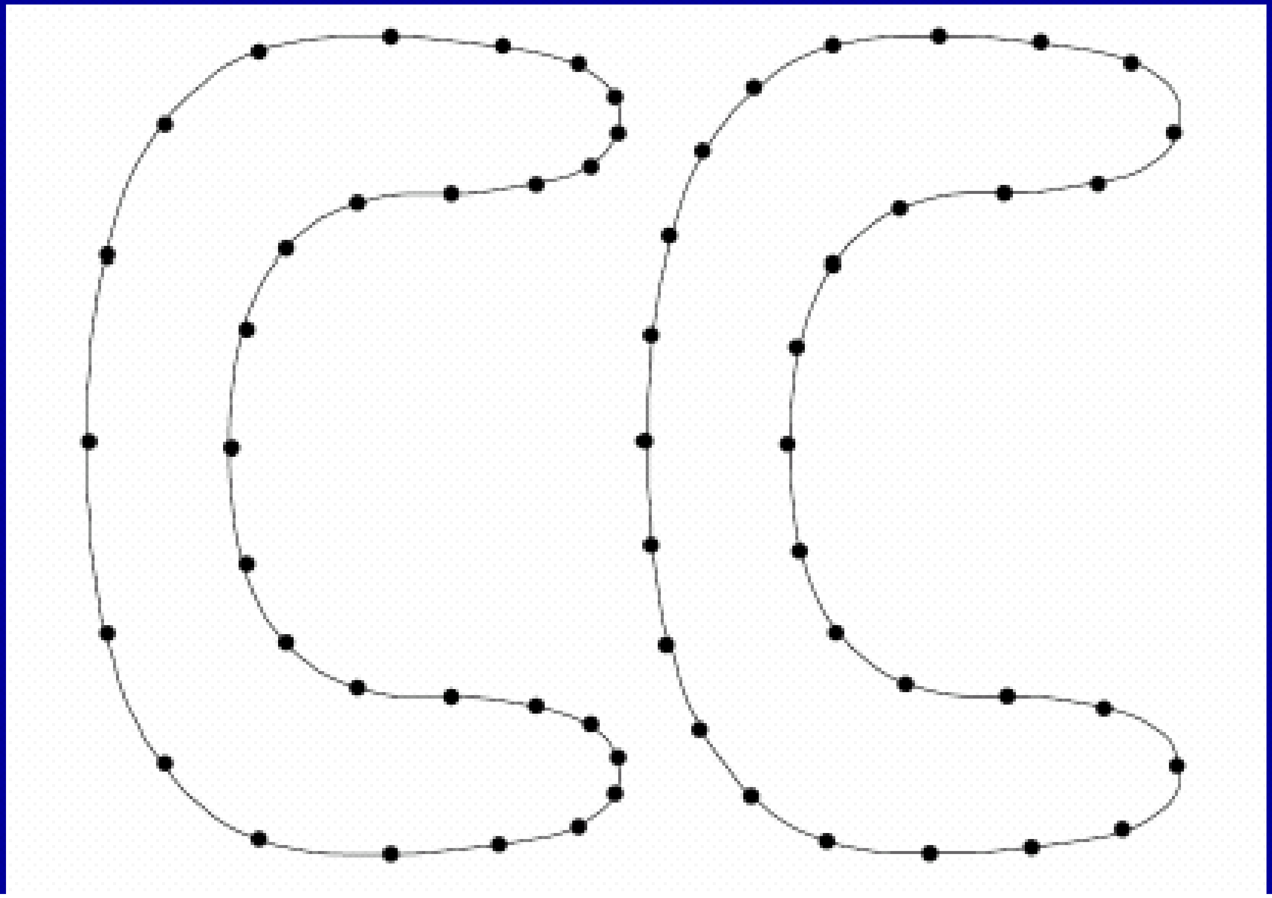
Mass point simulation

- Assume we have a 32-piece spline, with a general parameterization of $u \in [0,31]$

```
MassPoint(tmax) // tmax = final time
/* assume initially, we have t=0 and point is located at
u=0 */
    u = 0;
    s = 0;
    t = 0;
    While t < tmax
    {
        Assert u < 31;  // if not, end of spline reached
        Determine current velocity |v| using physics;
        s = s + |v| * Δt; // compute new arclength
        u = Bisection(u,u + delta,s); // solve for t
        p = p(u); // p = new mass point location
        Do some stuff with p, i.e. render point location, etc.
        t = t + Δt; // proceed to next time step
    }
```

Arclength Parametrization

- There are an infinite number of parameterizations of a given curve. Slow, fast, speed continuous or discontinuous, clockwise (CW) or CCW…
- A special one: arc-length-parameterization: u=s is arc length. We care about these for animation.



Arclength Parametrization

chalkboard

Arclength Parametrization Summary

- Arclength parameter s=s(u) is the distance from p(0) to p(u) along the curve

$$s(u) = \int_0^u \sqrt{x'(v)^2 + y'(v)^2 + z'(v)^2}\, dv$$

- The integral for s(u) usually cannot be evaluated analytically
- Has to evaluate the integral numerically
- Simpson's integration rule

$$\int_a^b f(x)dx = \sum_{k=1}^{(n-1)/2} \frac{h}{3}[f(x_{2k-1}) + 4f(x_{2k}) + f(x_{2k+1})] + O(h^5)$$

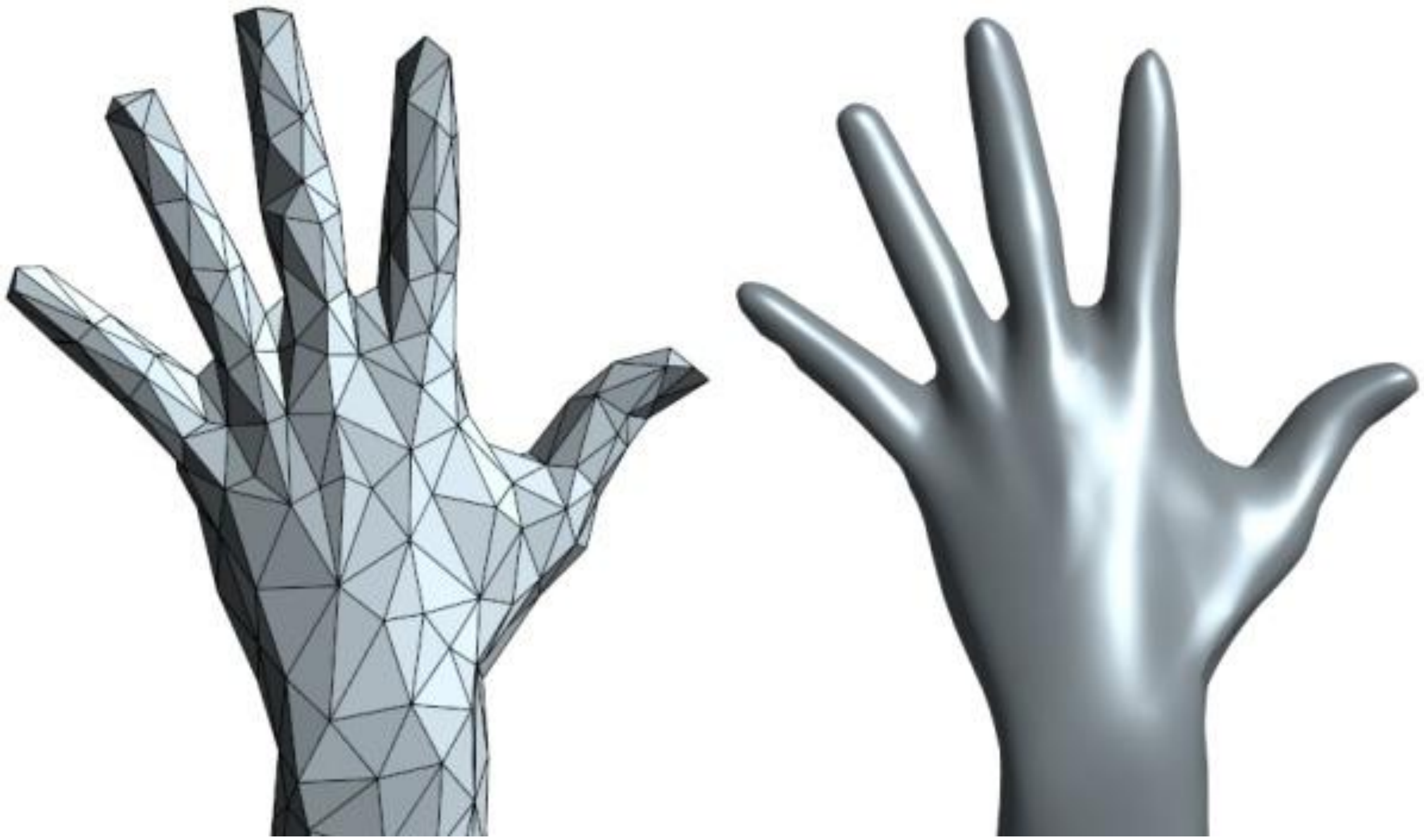
- Use bisection (next slide) to compute universe: u=u(s)

Computing inverse u=u(s)

- Must have initial guess for the interval containing u

```
Bisection(umin,umax,s)
/* umin = min value of u
   umax = max value of u; umin <= u <= umax
    s = target value */
    Forever // but not really forever
    {
        u = (umin + umax) / 2; // u = candidate for solution
        If |s(u)-s| < epsilon
            Return u;
        If s(u) > s  // u too big, jump into left interval
            umax = u;
        Else  // t too small, jump into right interval
            umin = u;
    }
```

3D Surfaces



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

2D Scalar Field

- z = f(x,y)

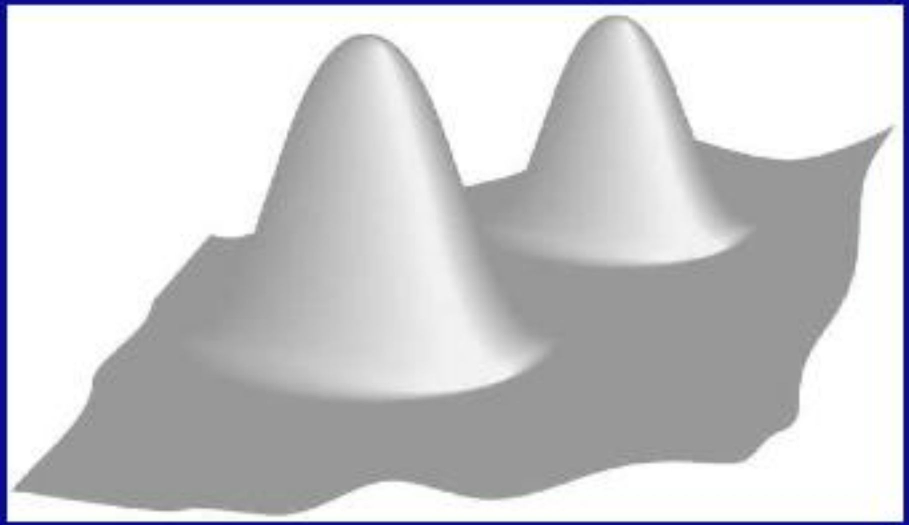
$$f(x,y) = \begin{cases} 1 - x^2 - y^2, & \text{if } x^2 + y^2 < 1 \\ 0 & \end{cases}$$

How do you visualize this function?

Height Field

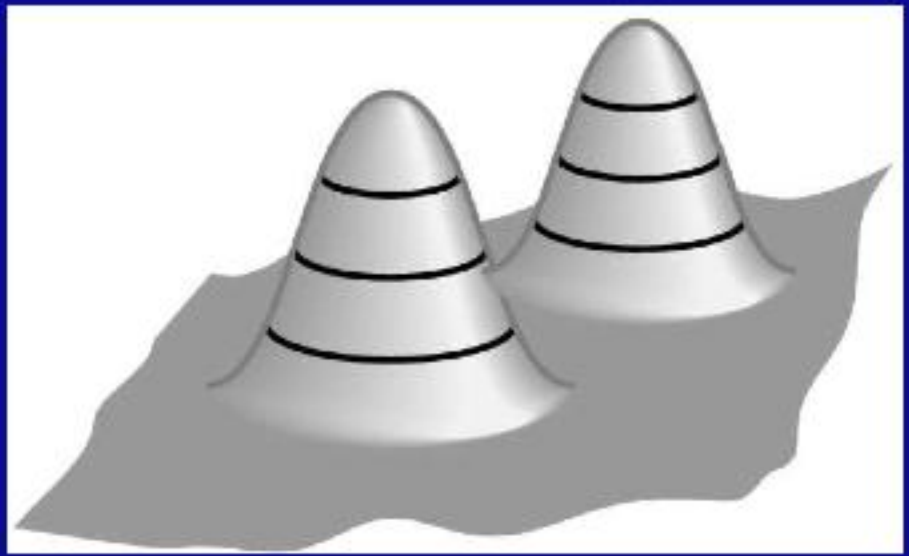
- Visualizing an explicit function

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



- Adding contour curves

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

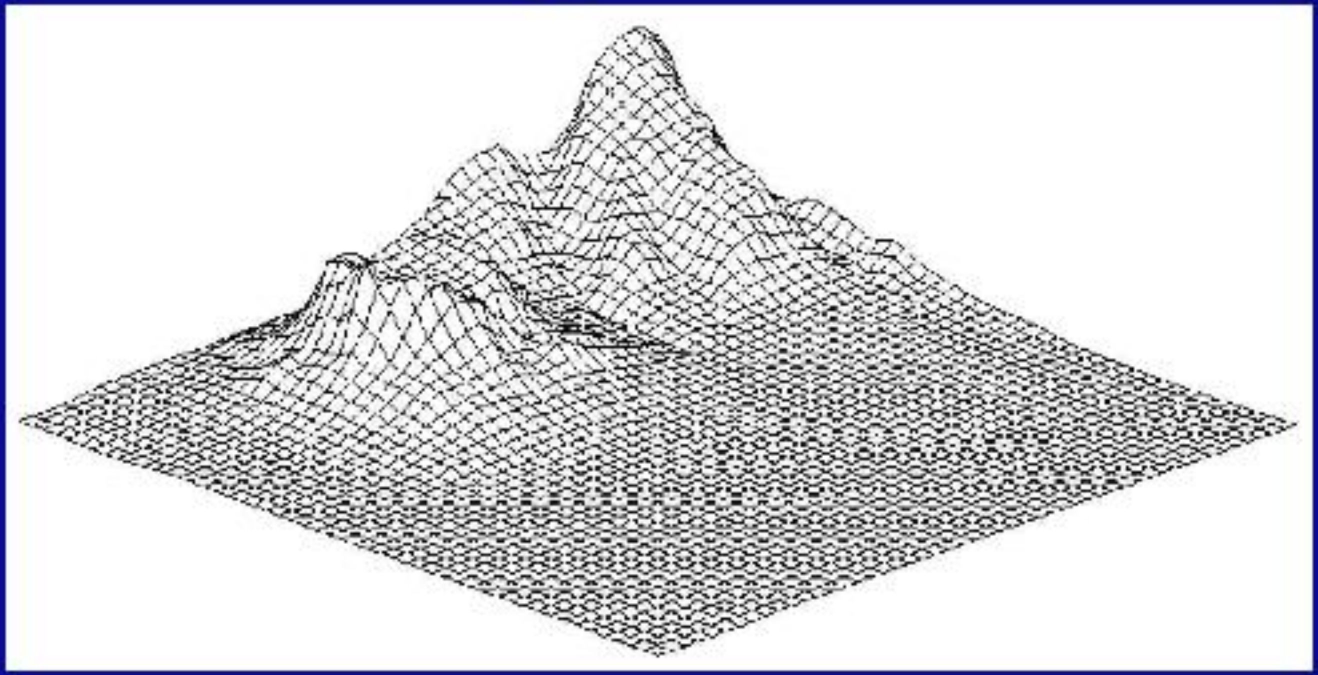


Meshes

- Function is sampled (given) at x_i, y_i, $0 \leq i, j \leq n$
- Assume equally spaced

$$
\begin{aligned}
x_i &= x_0 + i\Delta x \\
y_j &= y_0 + j\Delta y
\end{aligned}
\qquad z_{ij} = f(x_i, y_j)
$$

- Generate quadrilateral or triangular mesh
- [Asst 1]

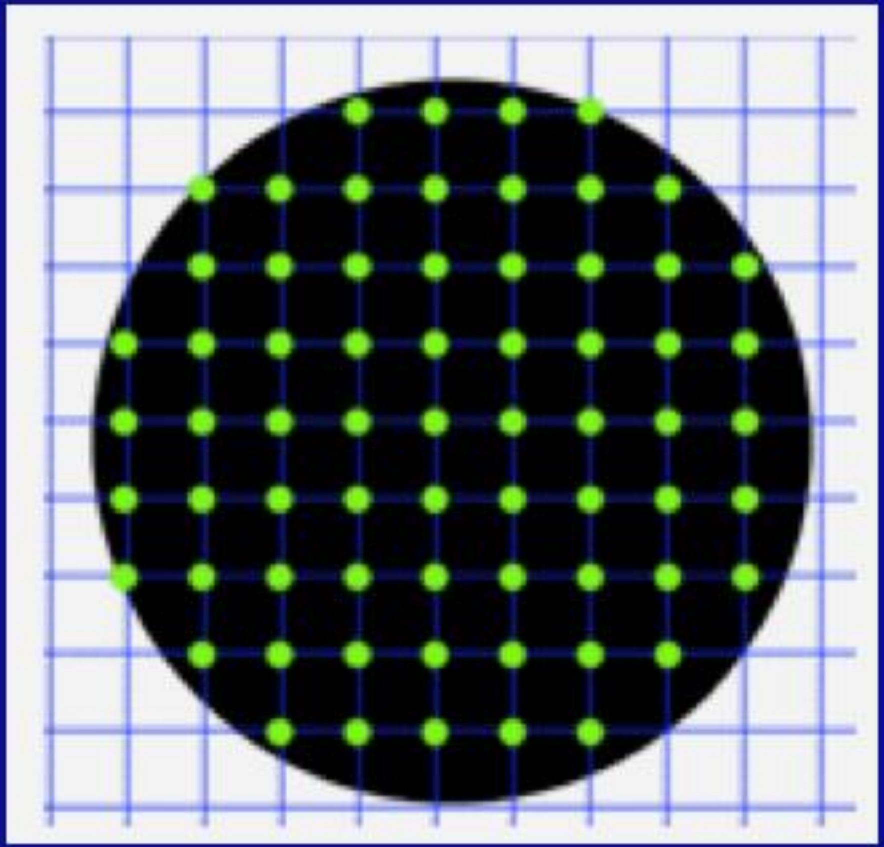


Implicit → Explicit 2D

(Marching Squares Algorithm)

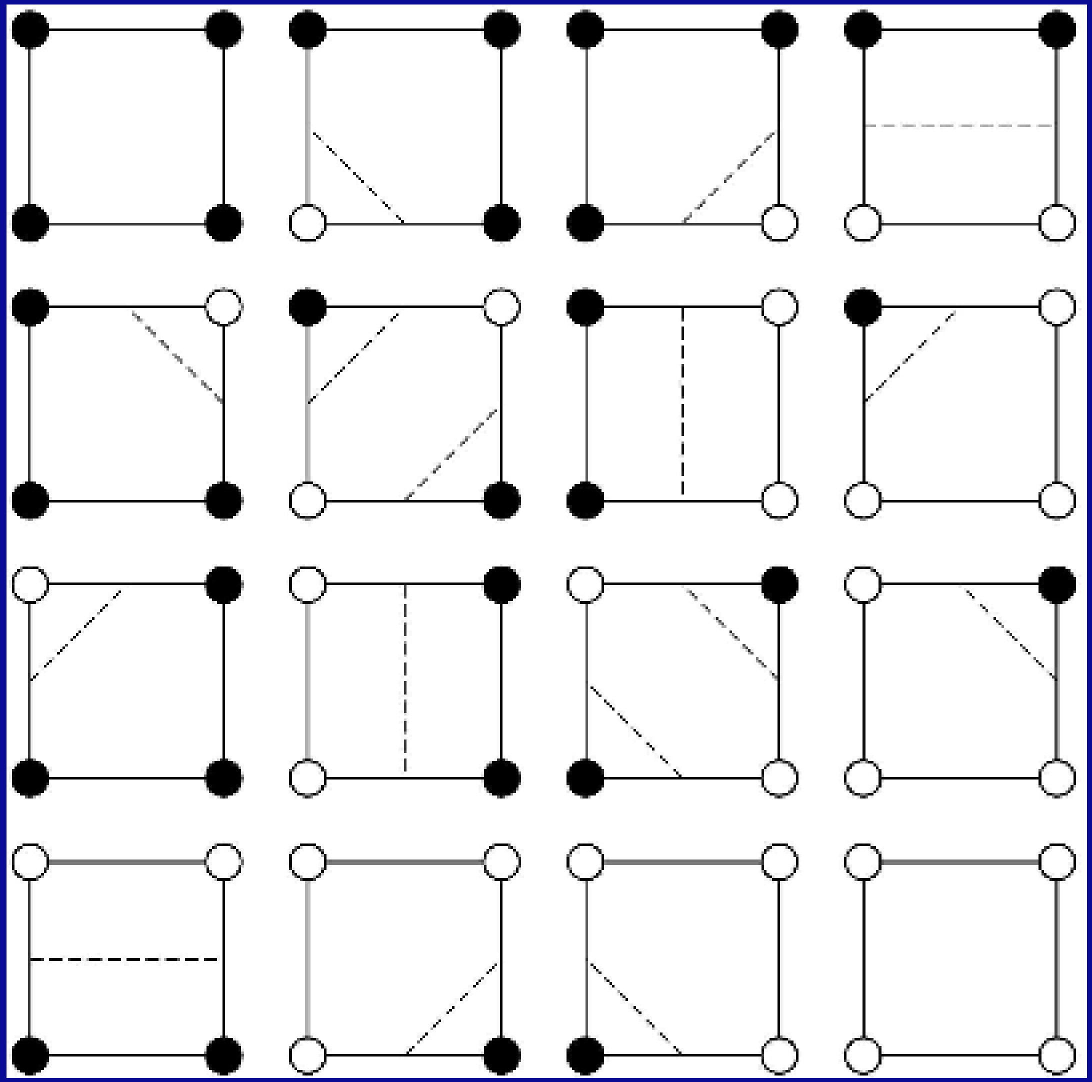
Marching Squares

- 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 $f_{ij} > c$

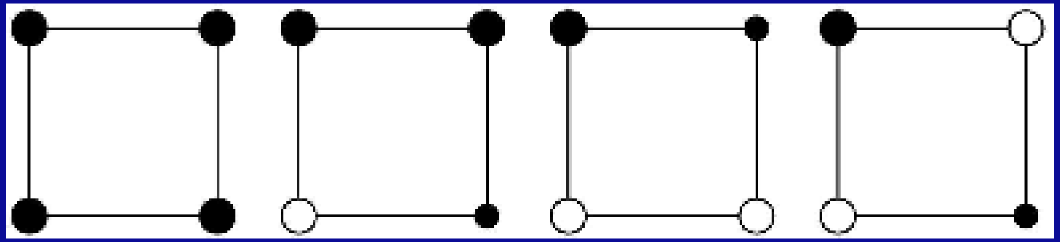


Cases for Vertex Labels

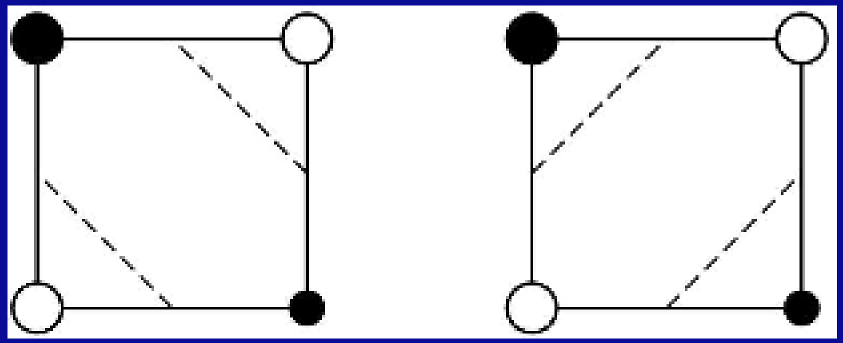
16 cases for vertex labels



4 unique mod. symmetries

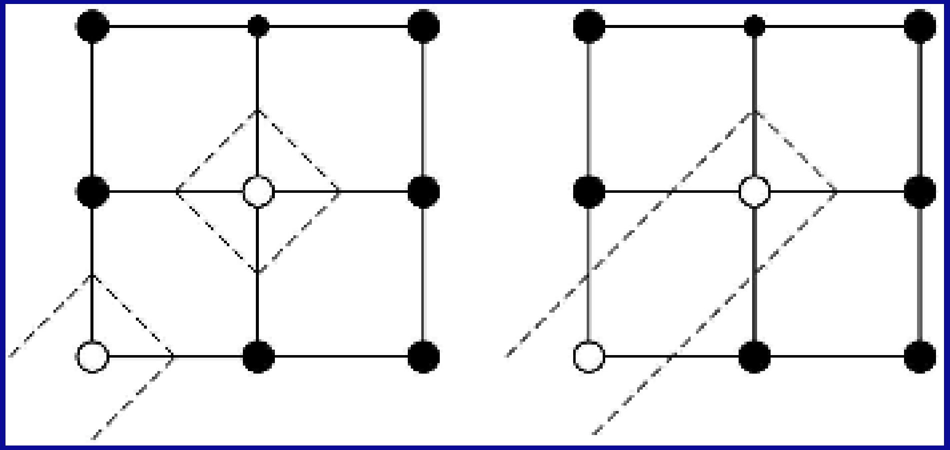


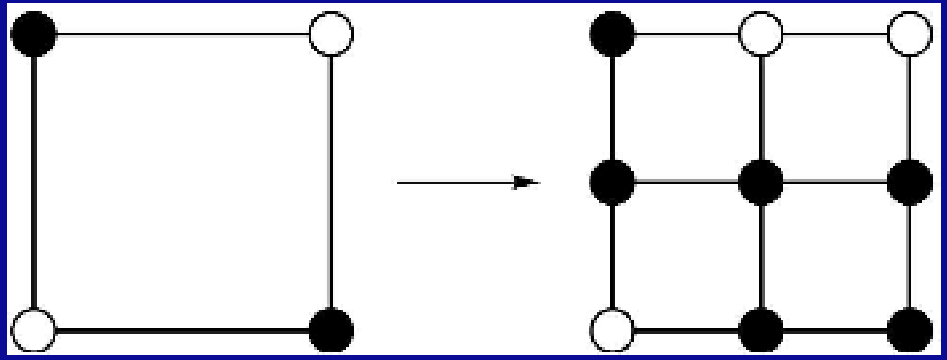
Ambiguities of Labelings



Ambiguous labels

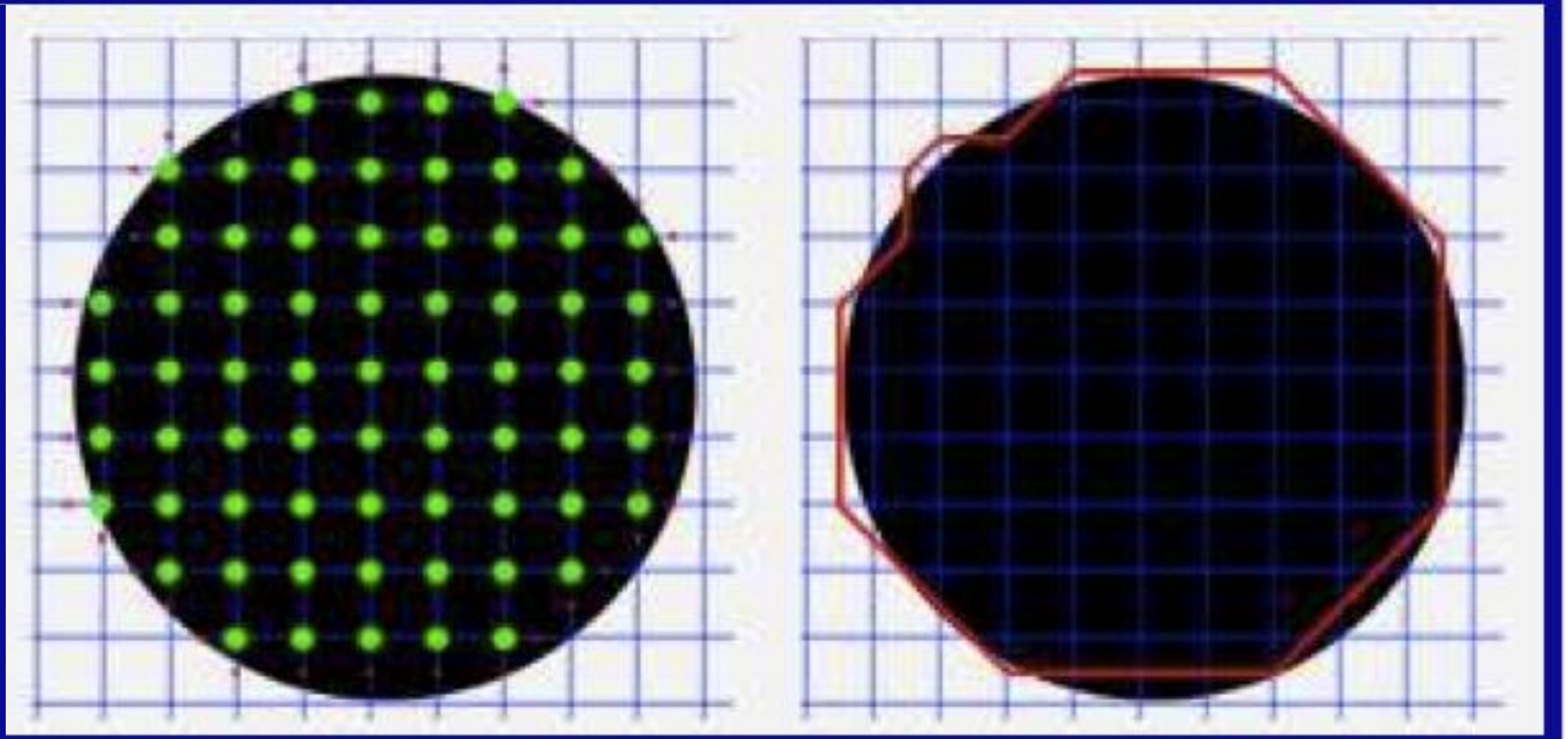
Different resulting contours





Resolution by subdivision (where possible)

Marching Squares Examples



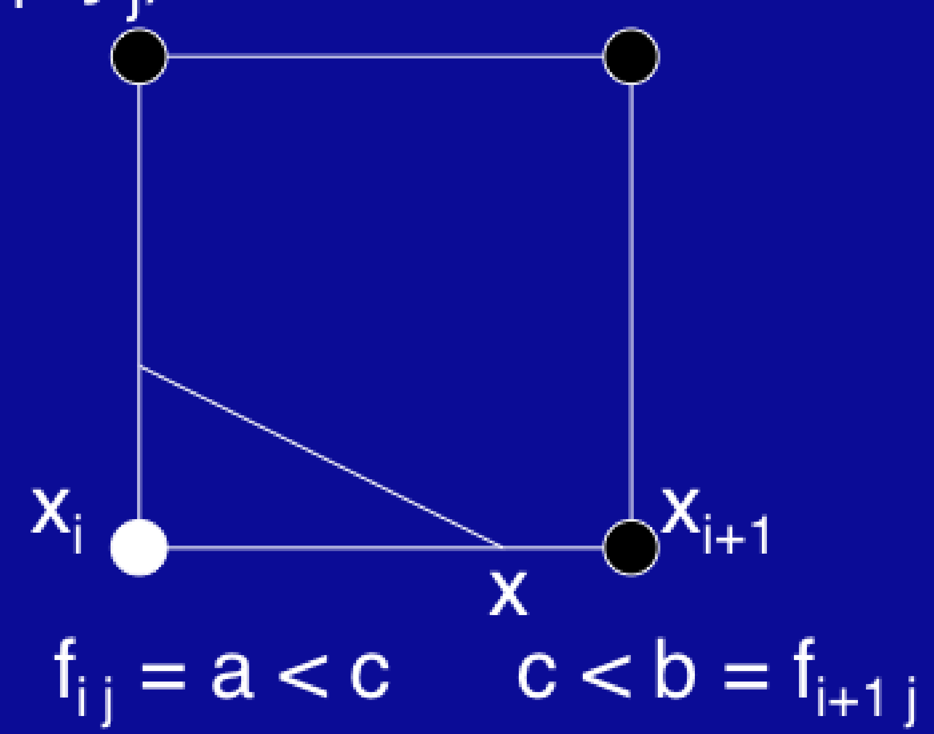
Can you do better?

Interpolating Intersections

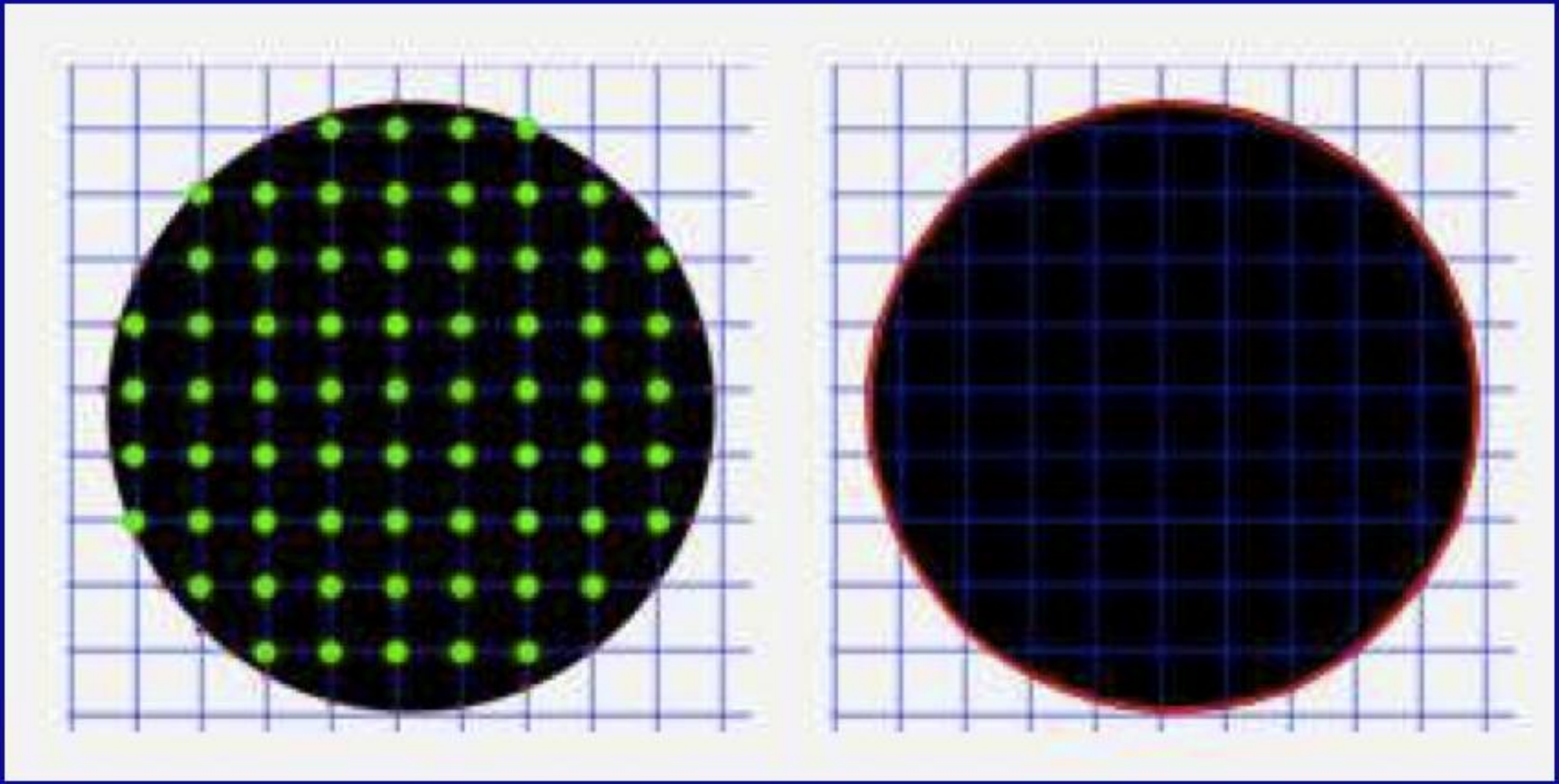
- Approximate intersection
  - Midpoint between x_i, x_{i+1} and y_j, y_{j+1}
  - Better: interpolate
- If $f_{i\,j} = a$ is closer to c than $b = f_{i+1\,j}$ then intersection is closer to (x_i, y_j):

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

- Analogous calculation for y direction

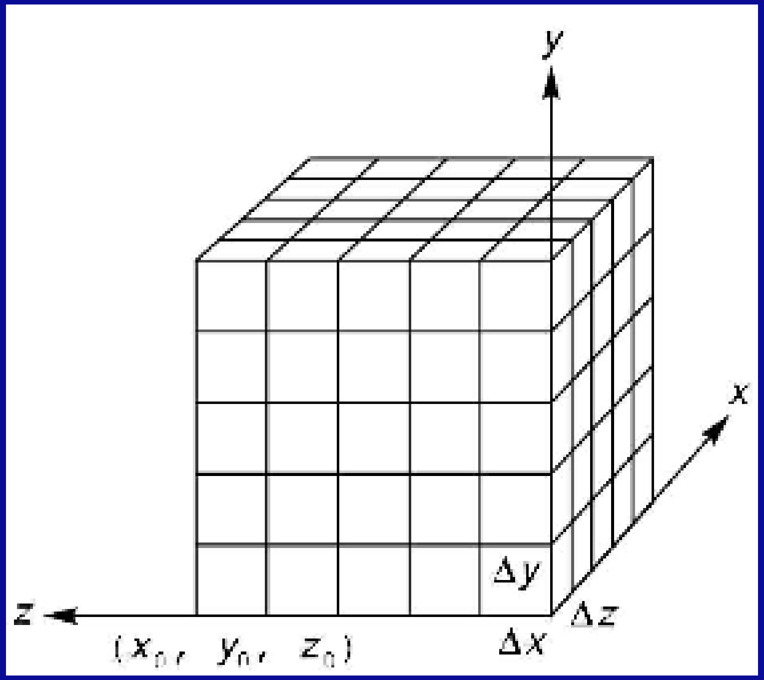


Marching Squares Examples



Implicit $\rightarrow$ 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

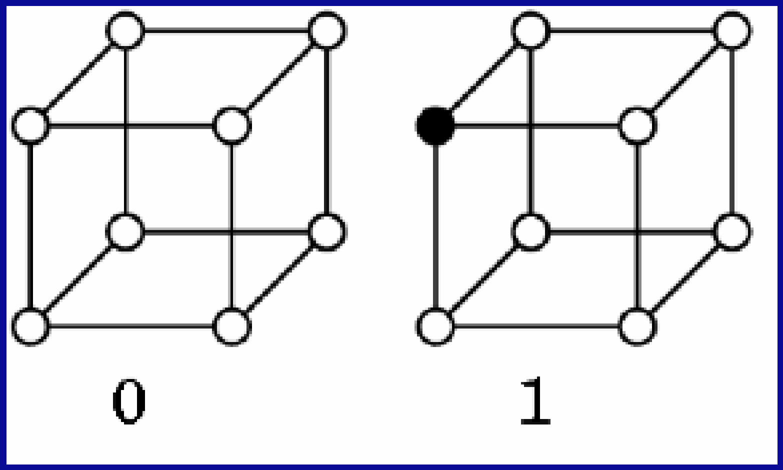
Isosurfaces

- 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?

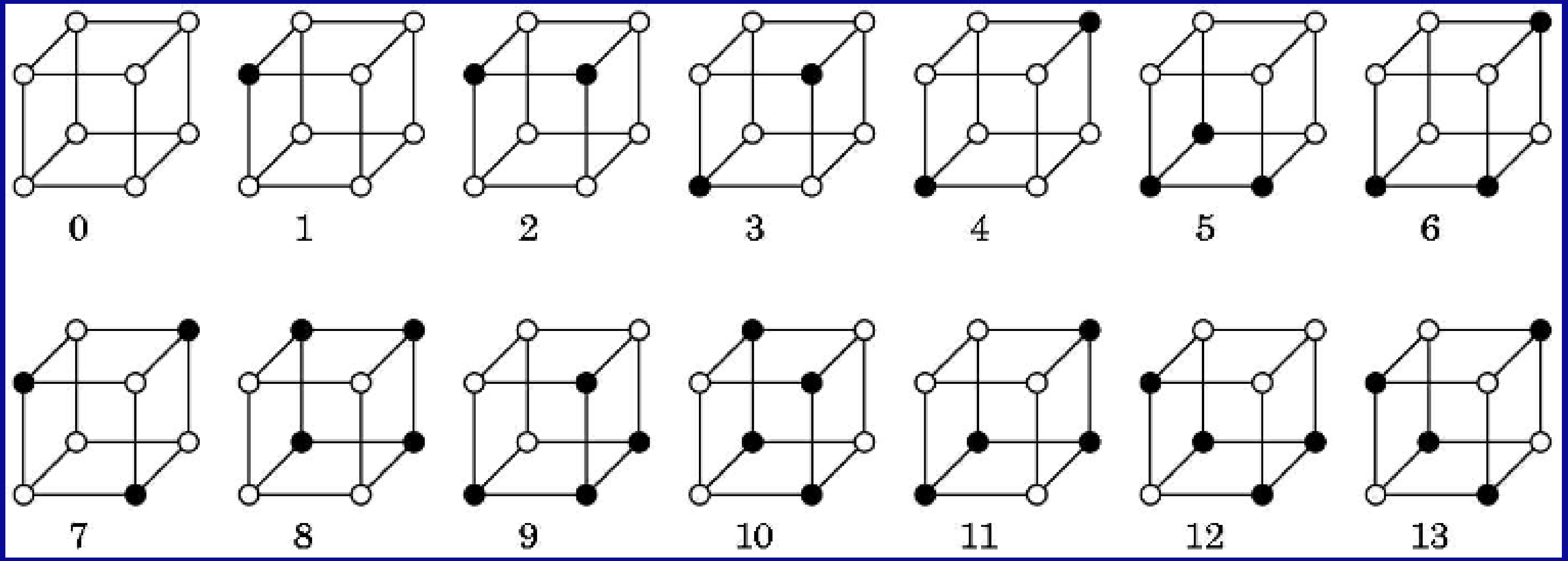


...

$2^8 = 256$

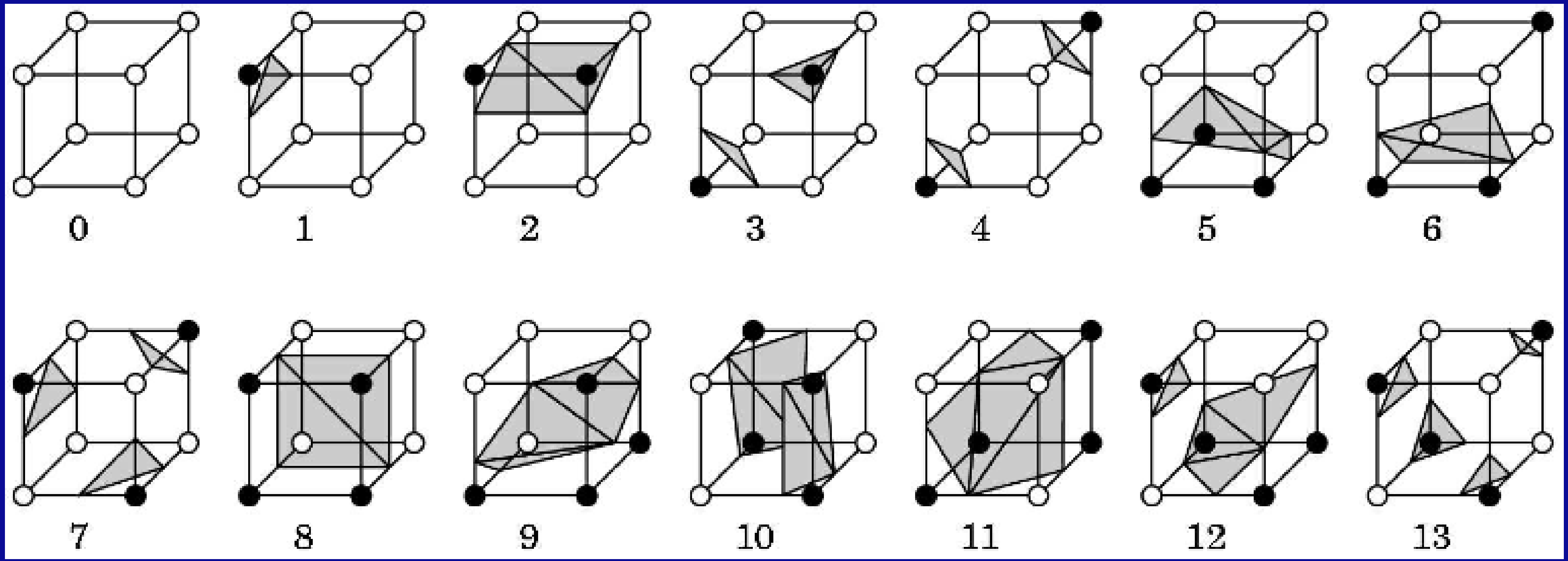
Marching Cubes

- 14 cube labelings (after elimination symmetries)

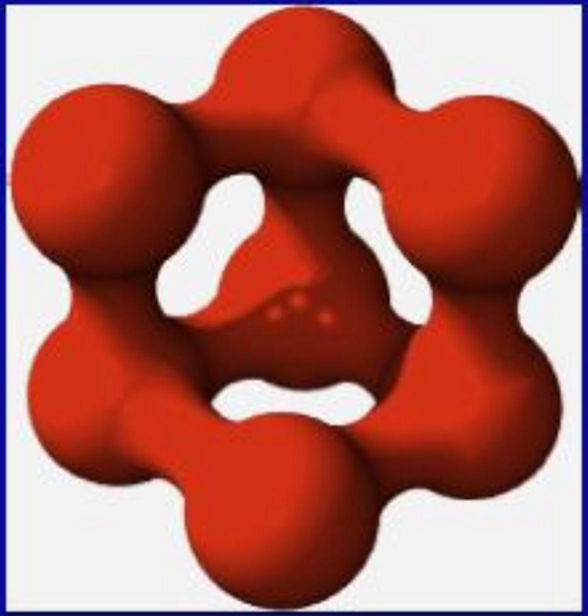


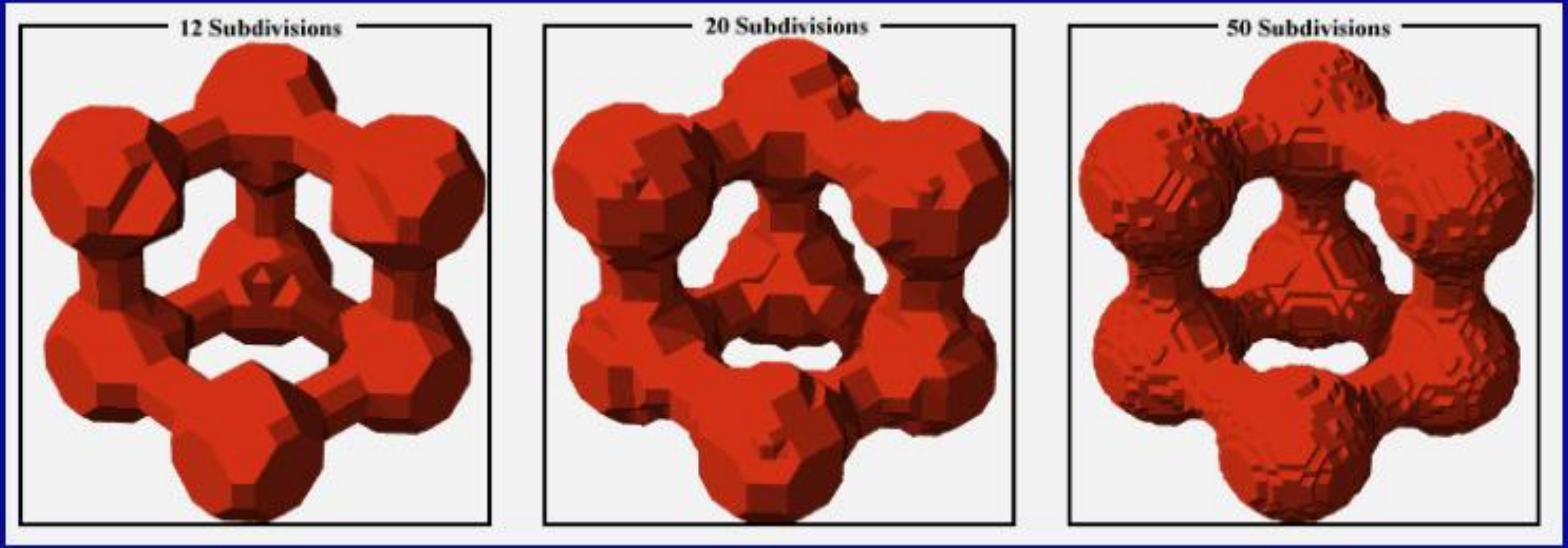
Marching Cube Tessellations

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

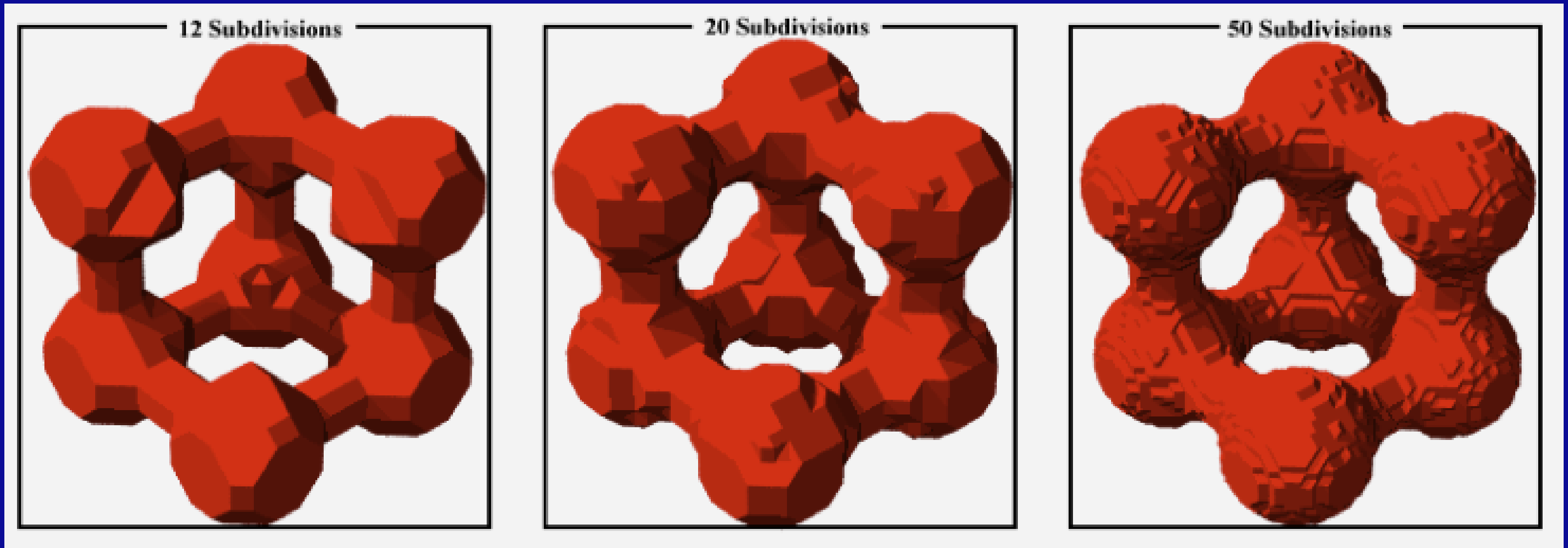


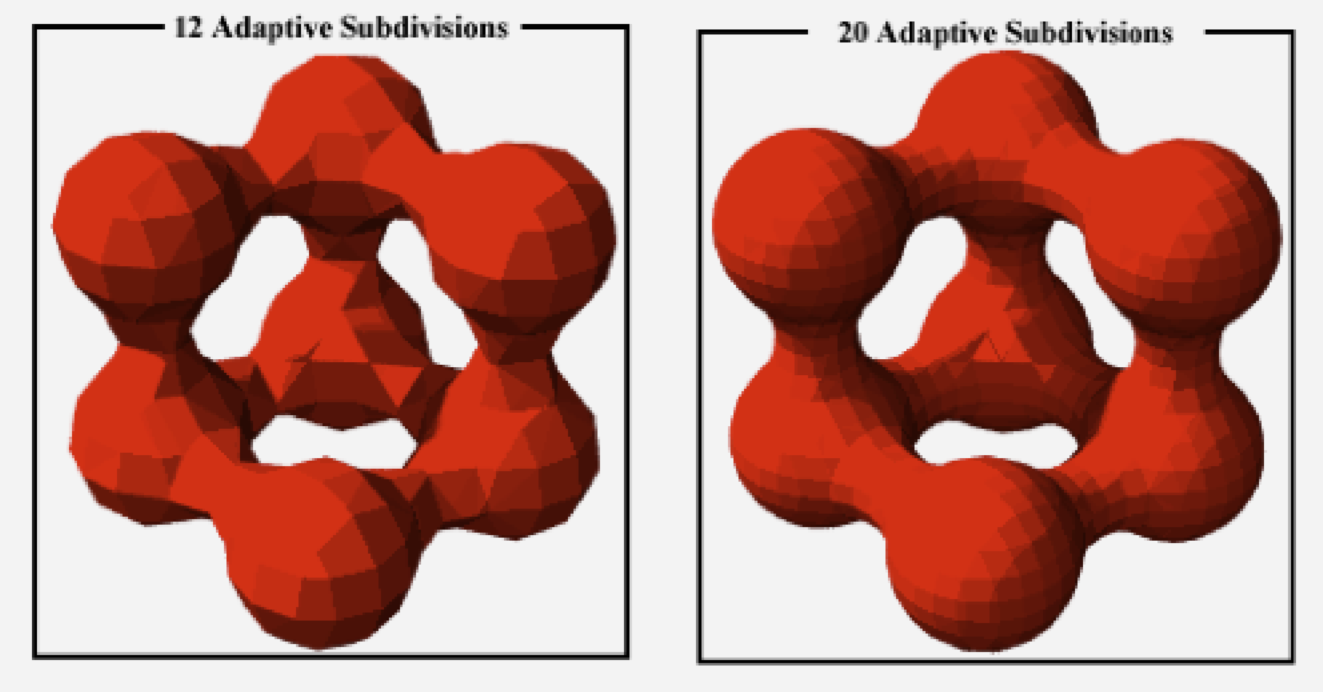
Marching Squares Examples





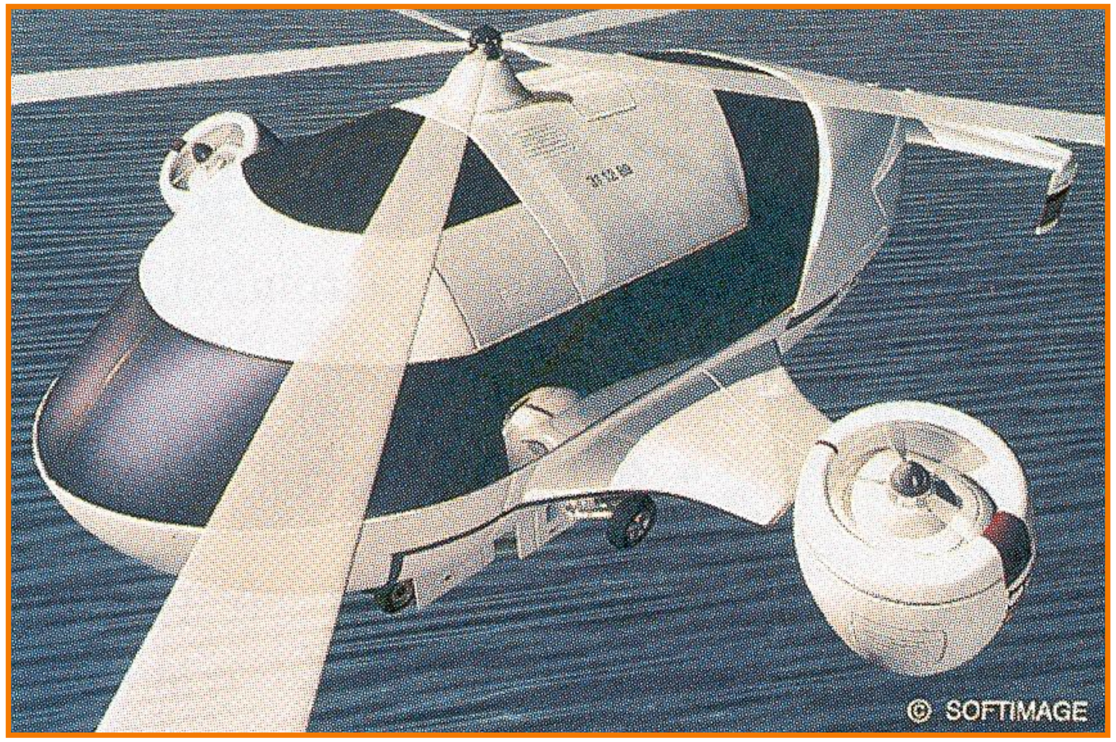
Marching Squares Examples





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

3D Object Representations

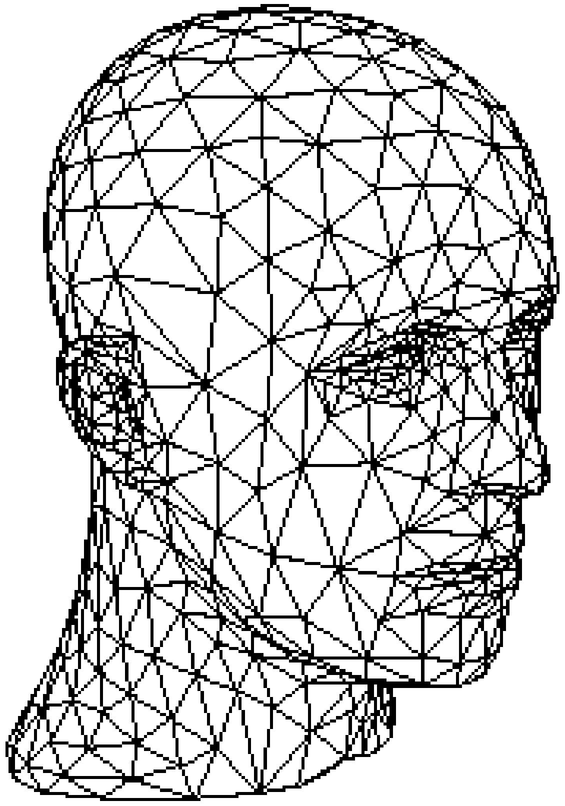
- Raw data
  - Voxels
  - Point cloud
  - Range image
  - Polygons
- Surfaces
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Octree
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

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

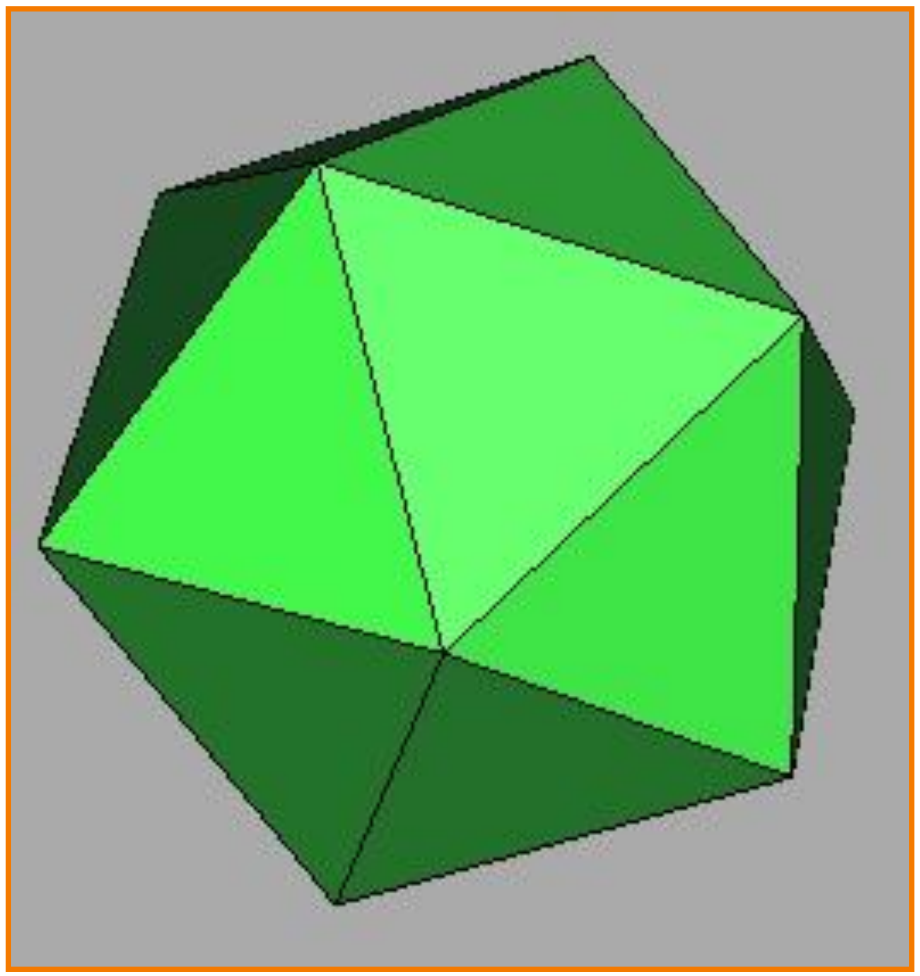
- How should we represent a mesh in a computer?
  - Efficient traversal of topology
  - Efficient use of memory
  - Efficient updates
- Mesh Representations
  - Independent faces
  - Vertex and face tables
  - Adjacency lists
  - Winged-Edge
  - Half-Edge
  - etc.

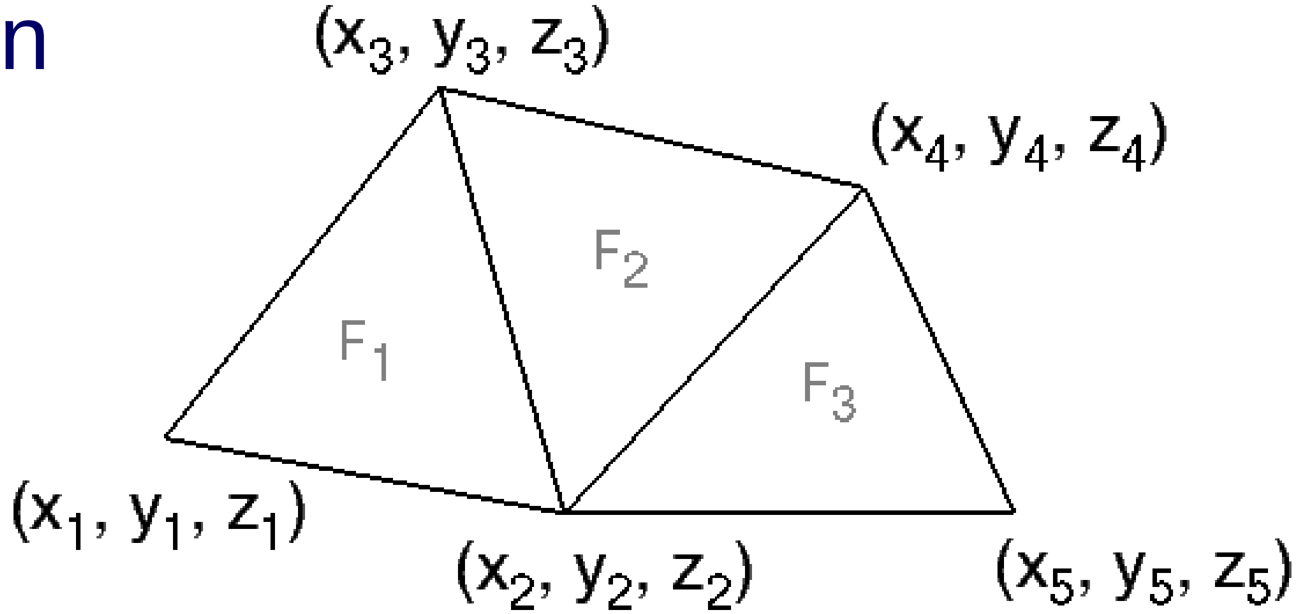


Zorin & Schroeder, SIGGRAPH 99, Course Notes

Independent Faces

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

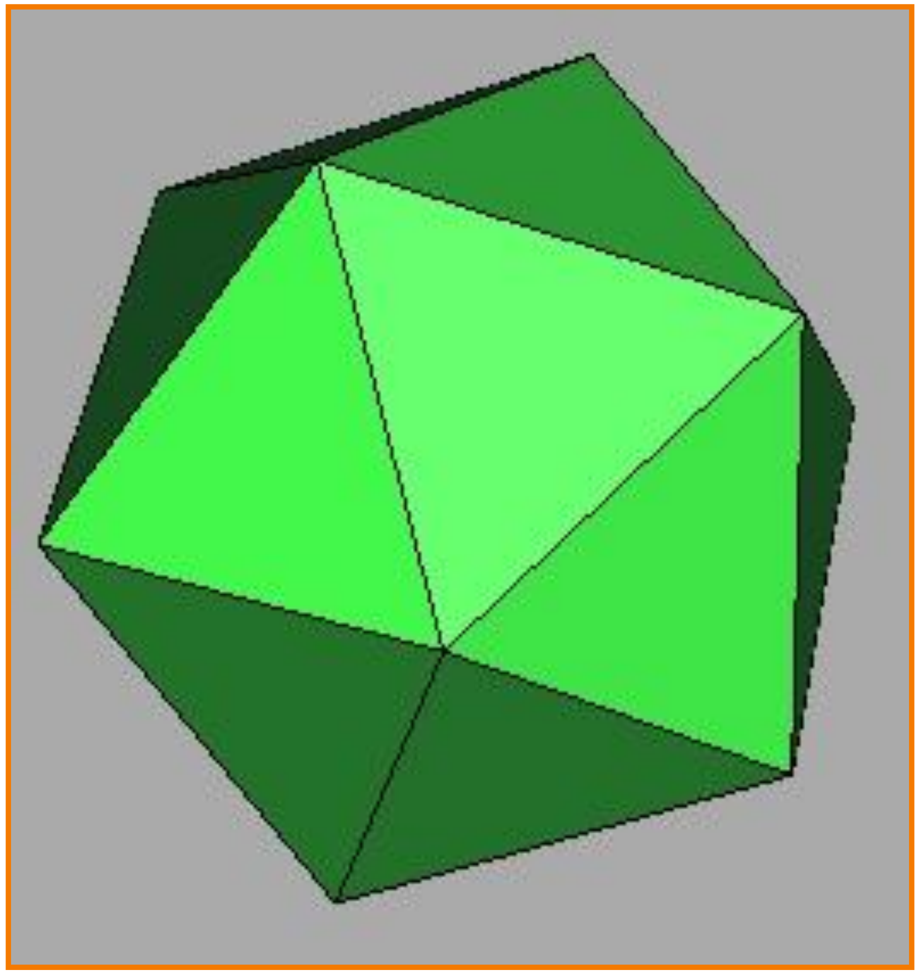


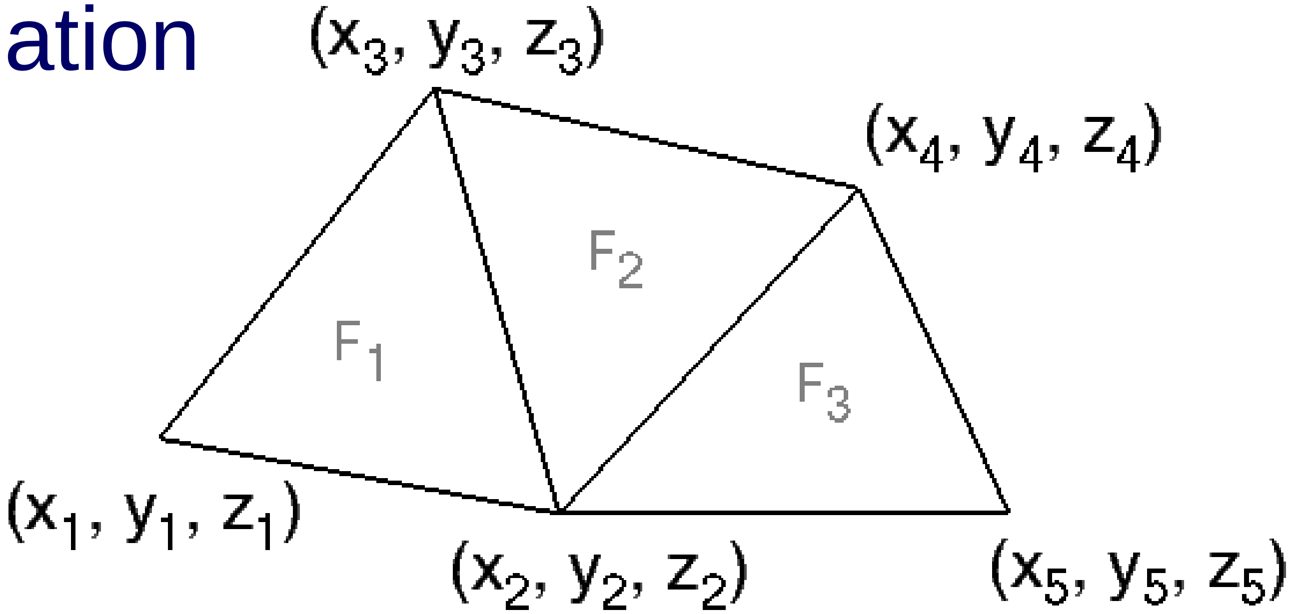


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

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





VERTEX TABLE

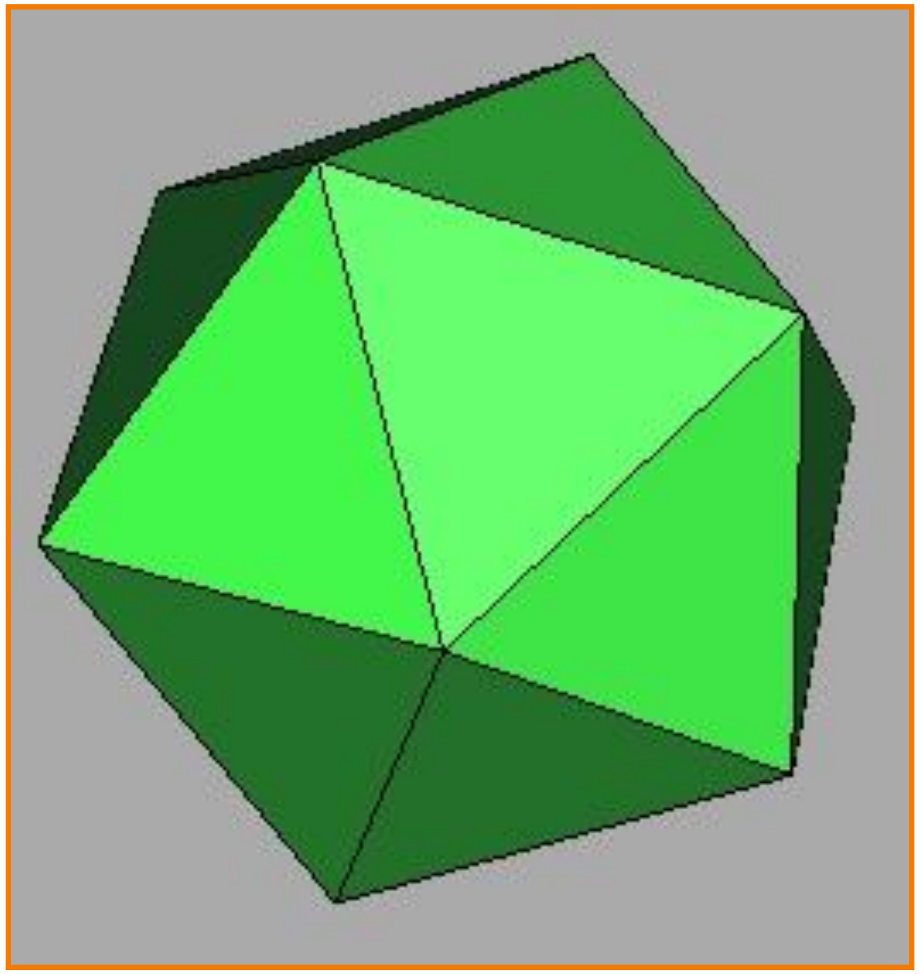
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

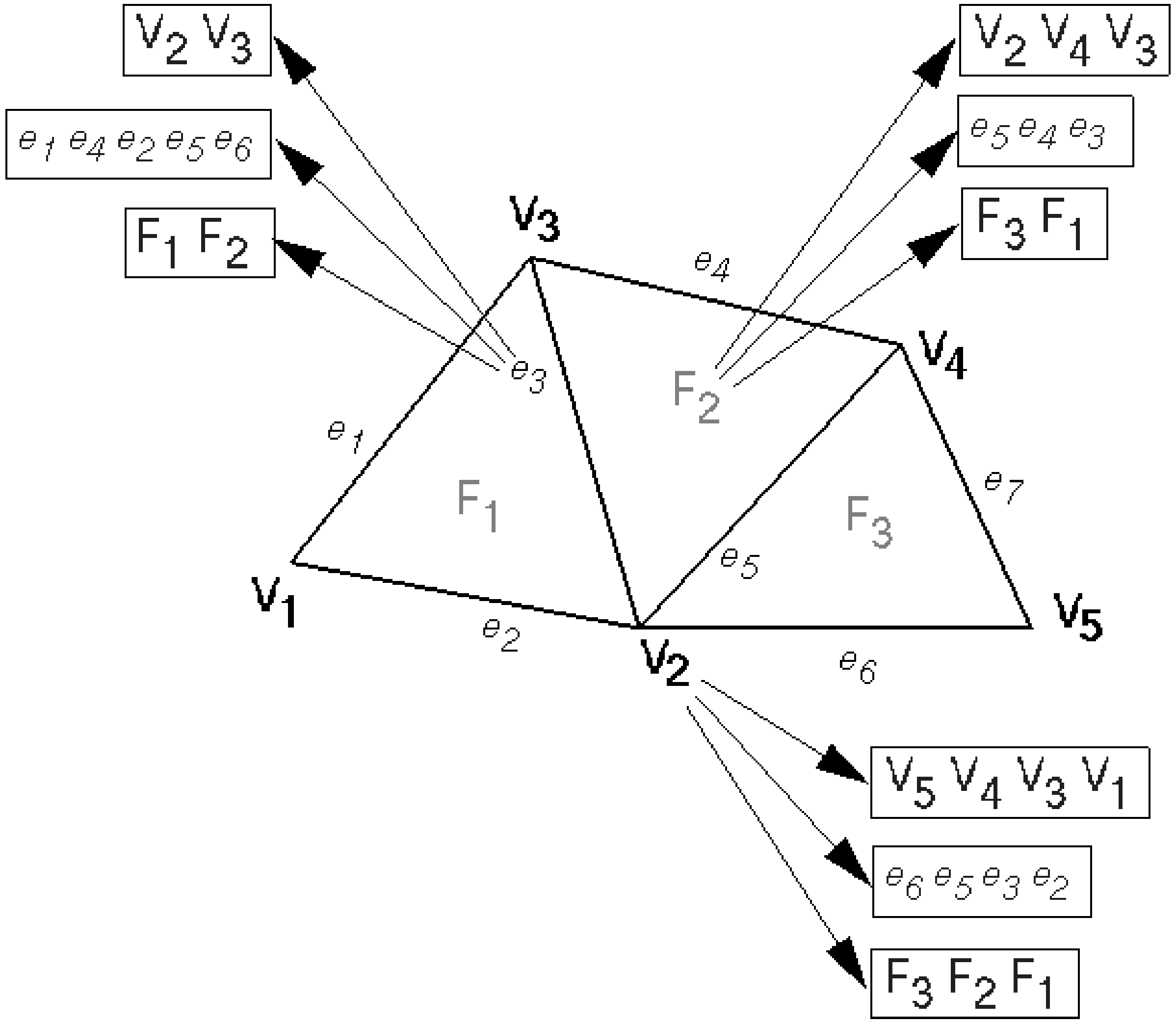
FACE TABLE

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

Adjacency Lists

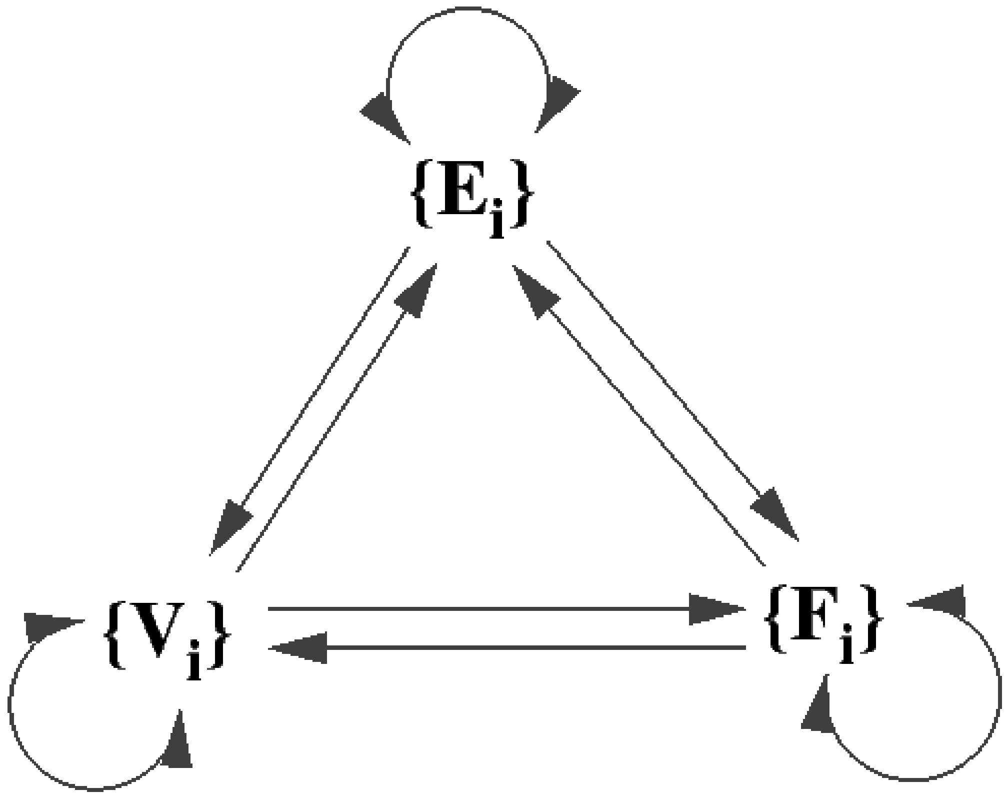
- Store all vertex, edge, and face adjacencies
  - Efficient adjacency traversal
  - Extra storage

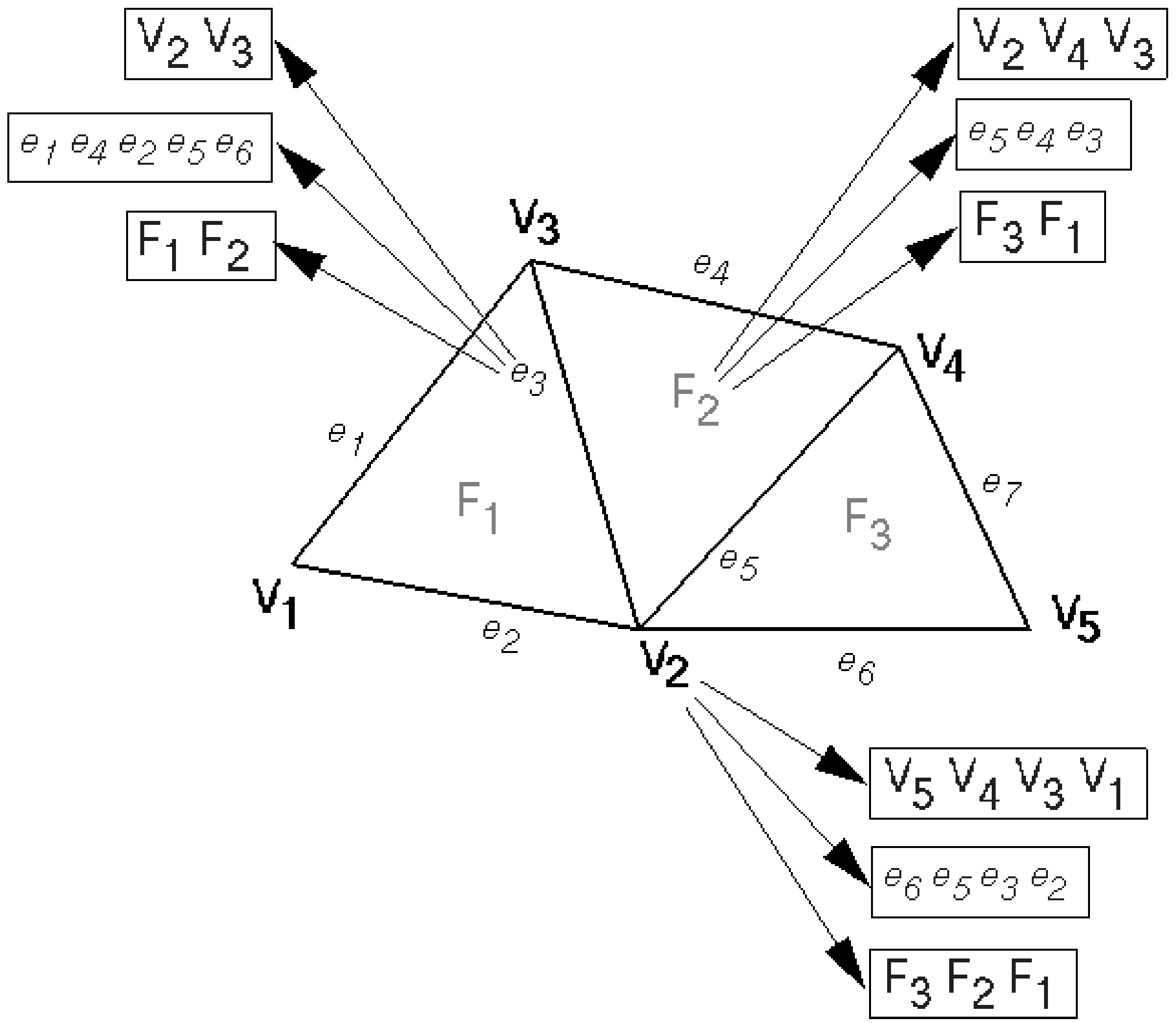




Partial Adjacency Lists

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



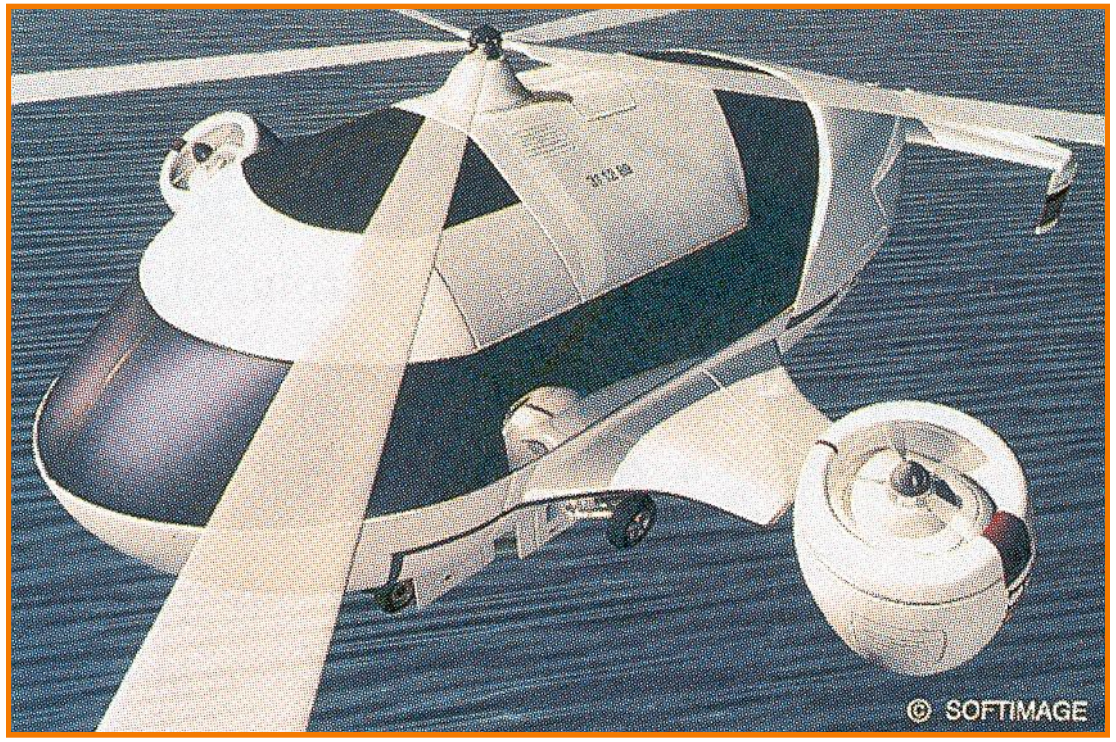


3D Object Representations

- 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

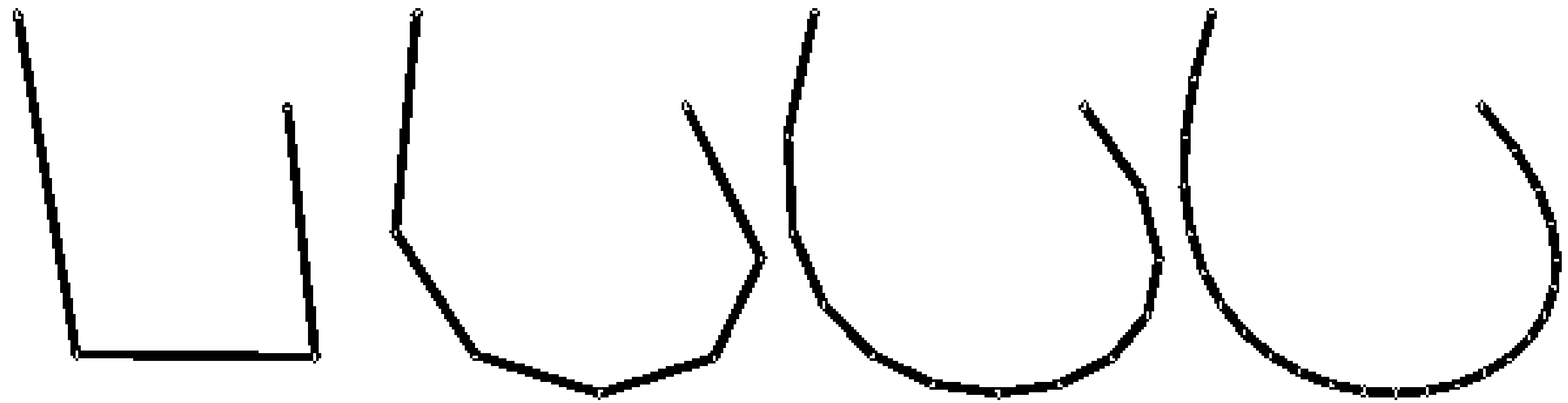
- What makes a good surface representation?
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - ➢ Guaranteed continuity
  - Natural parameterization
  - Efficient display
  - Efficient intersections



H&B Figure 10.46

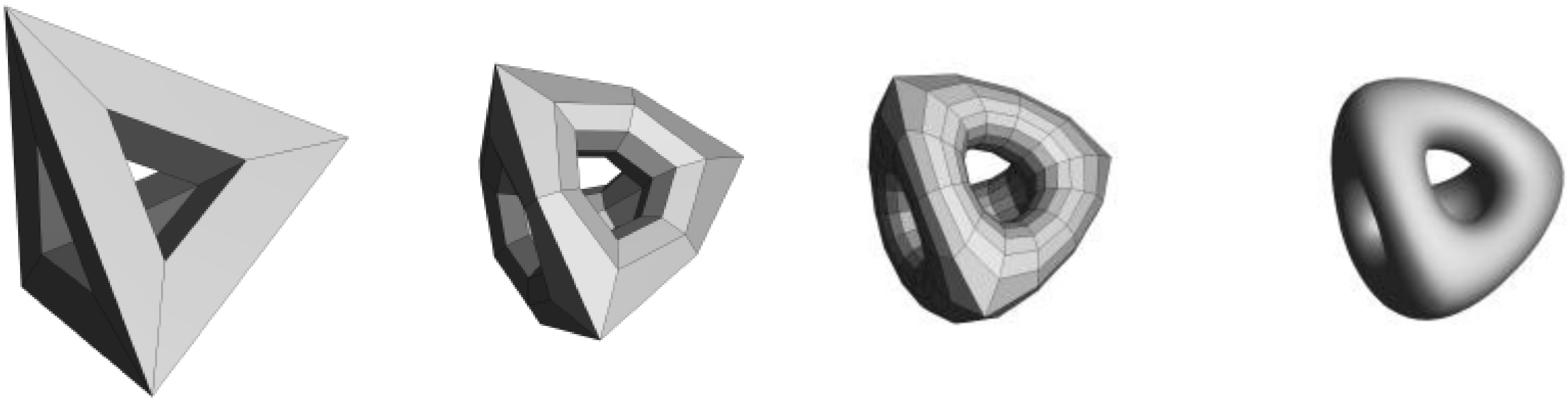
Subdivision

- How do you make a smooth curve?



Subdivision Surfaces

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

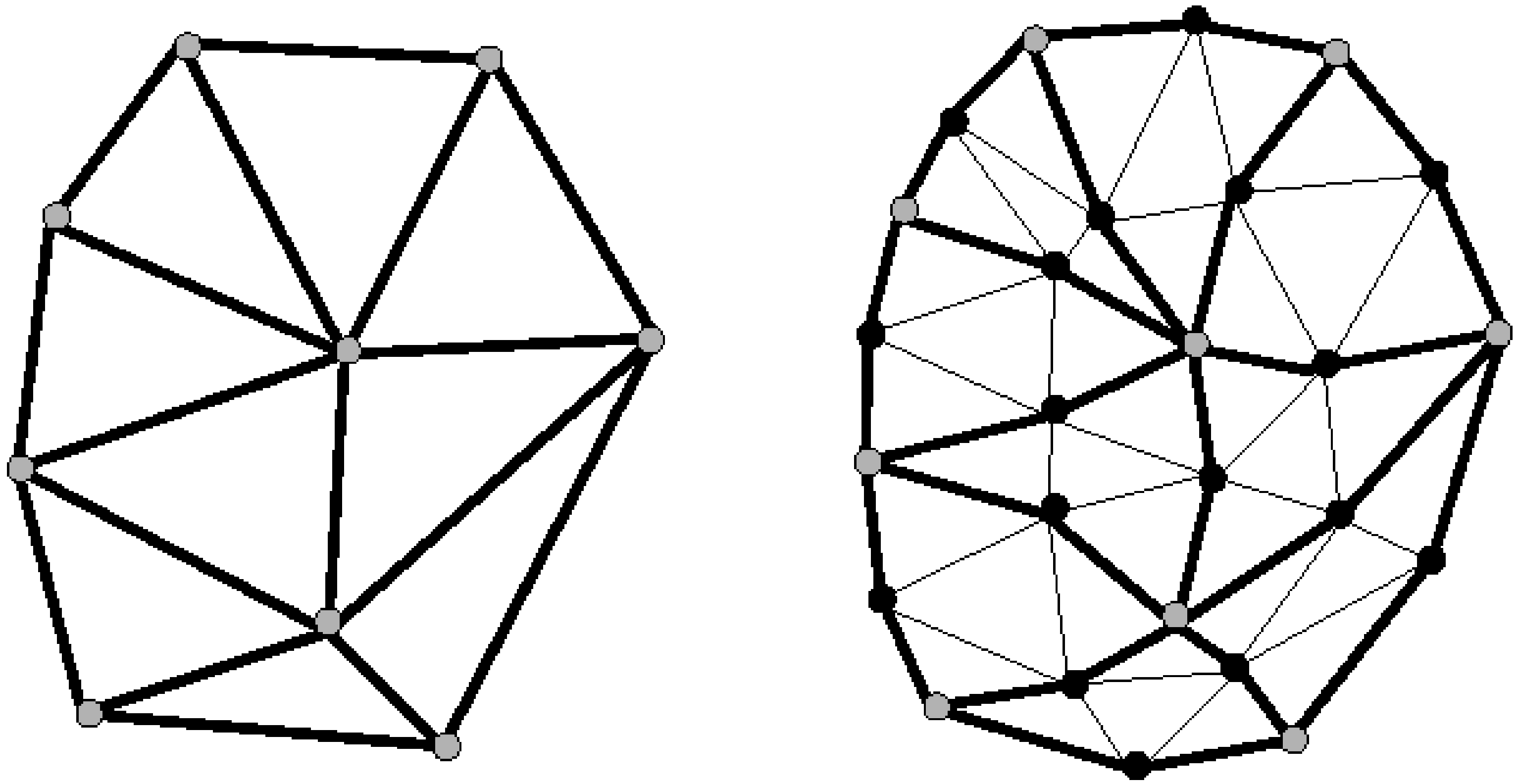


(a) (b) (c) (d)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

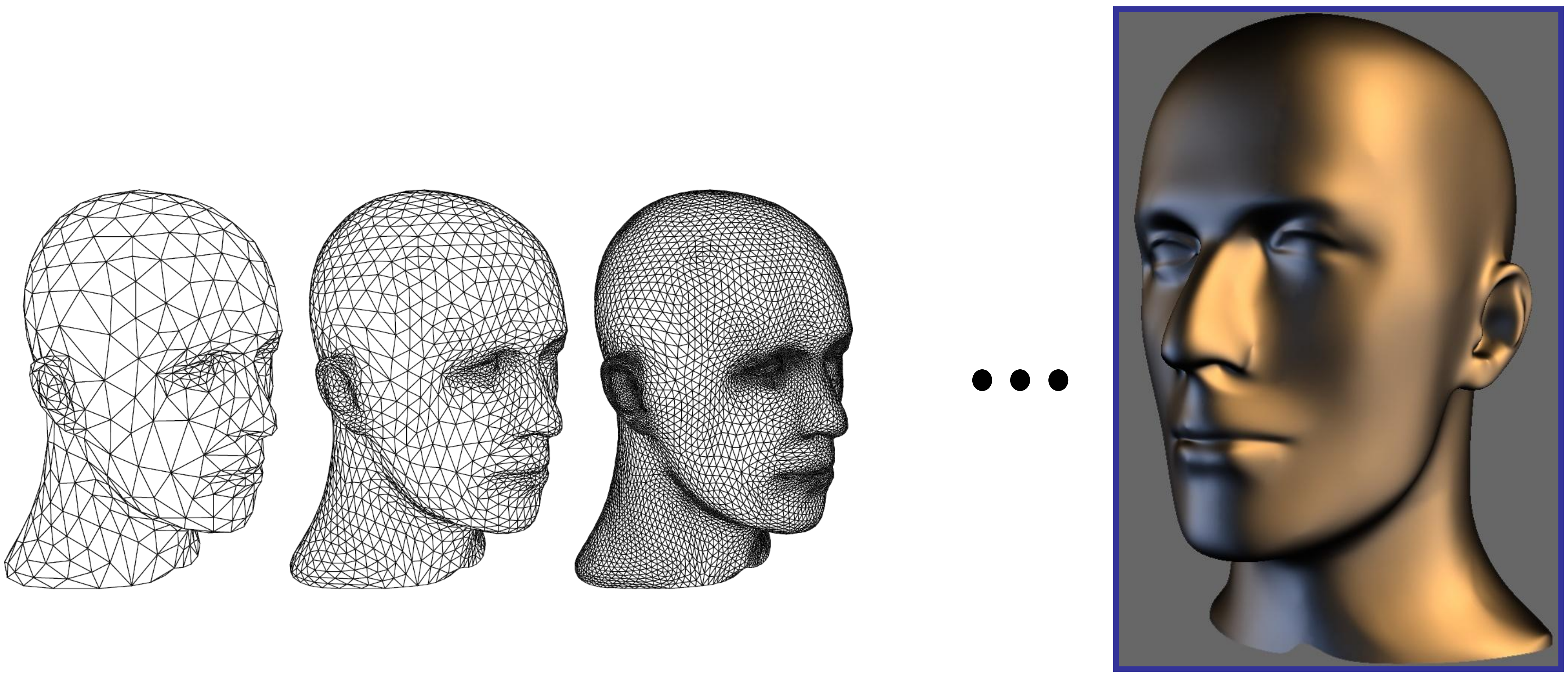
Key Questions

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



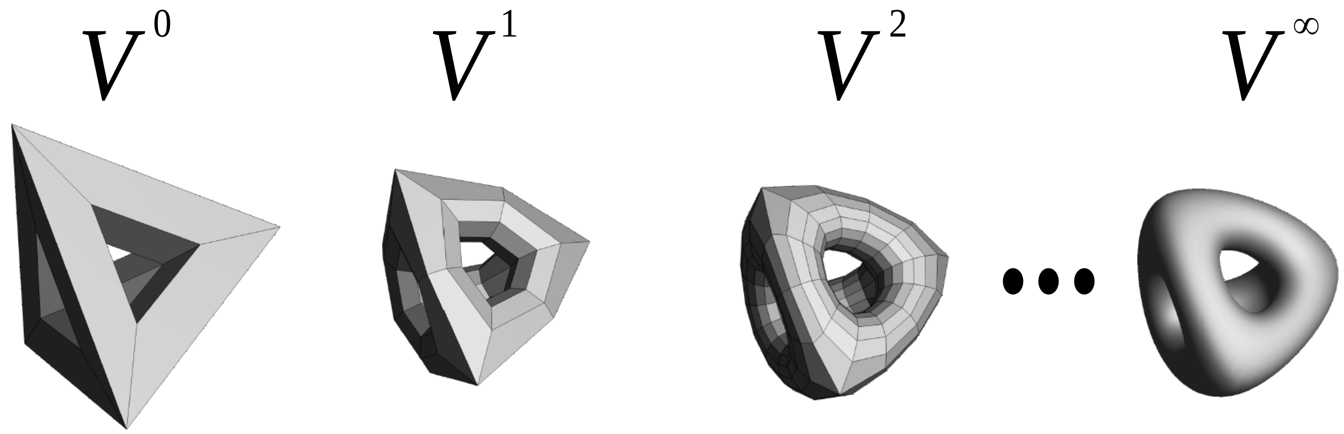
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces – A 3D example



Applications: Mainly Computer Graphics / animation

The basic idea



- In each iteration
  - Refine a control net (mesh)
  - Increases the number of vertices / faces
- The mesh vertices converges to a limit surface
- Each subdivision scheme has:
  - Rules to calculate the locations of new vertices.
  - A method to generate the new net topology.

Subdivision schemes

- Catmul Clark
- Doo Sabin
- Loop
- Butterfly – Nira Dyn
- …many more

Classification:

- Mesh types: tris, quads, hex…, combination
- Face / vertex split
- Interpolating / Approximating
- Smoothness
- (Non)Linear
- …

Catmull-Clark scheme ’78

- **Face Point**

$$f = \frac{1}{m}\sum_{i=1}^{m} p_i$$

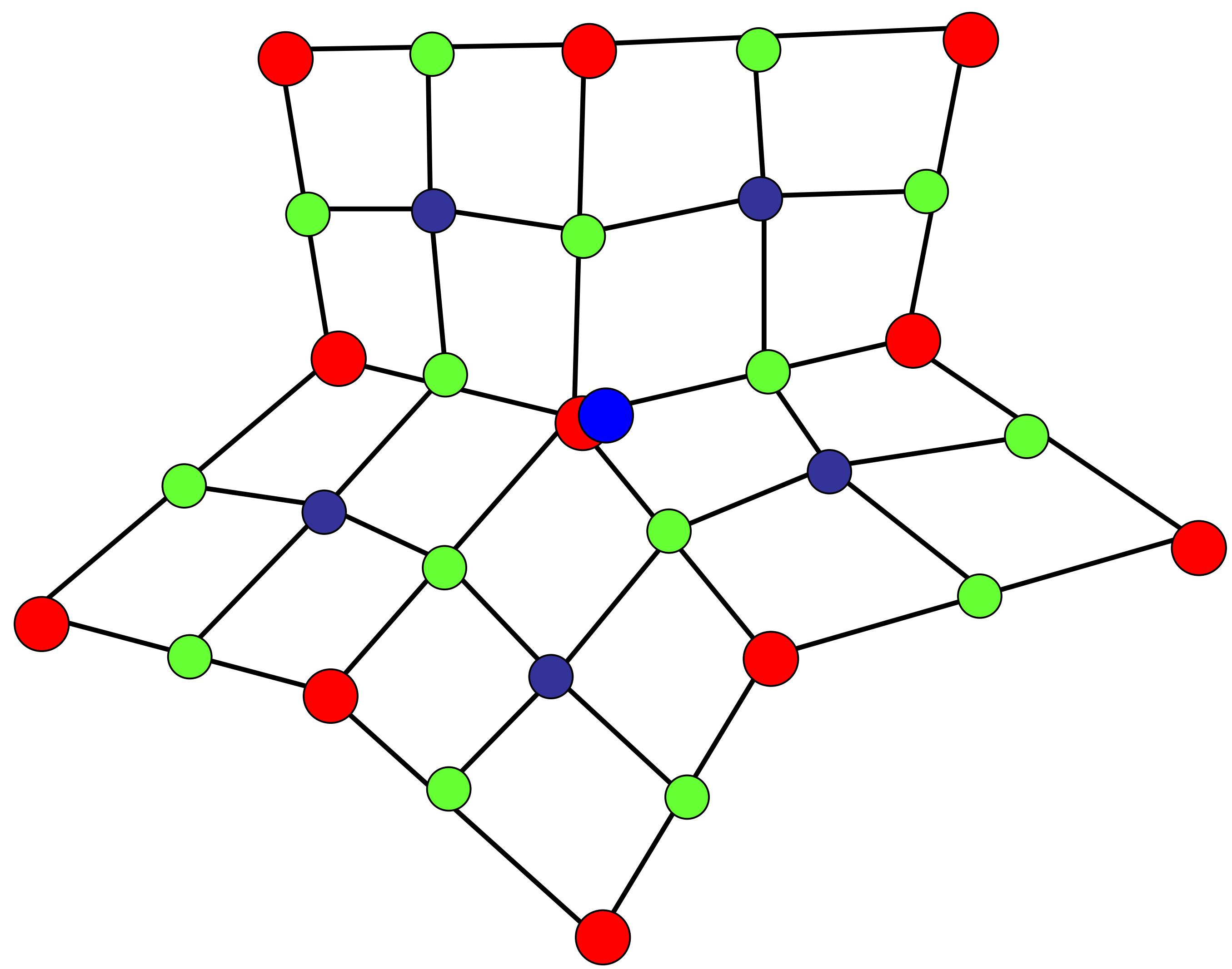
- **Edge Point**

$$e = \frac{p_1 + p_2 + f_1 + f_2}{4}$$

- **Vertex Point**

$$v = \frac{Q}{n} + \frac{2R}{n} + \frac{p\ (n-3)}{n}$$

$$v = \frac{1}{n^2}\sum_{i=1}^{n} f_i + \frac{1}{n^2}\sum_{i=1}^{n} e_i + \frac{n-2}{n} p$$



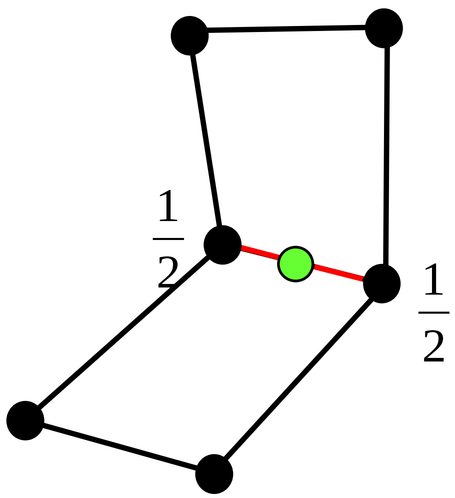
- ❑ Q – Average of face points
- ❑ R – Average of midpoints
- ❑ P – old vertex

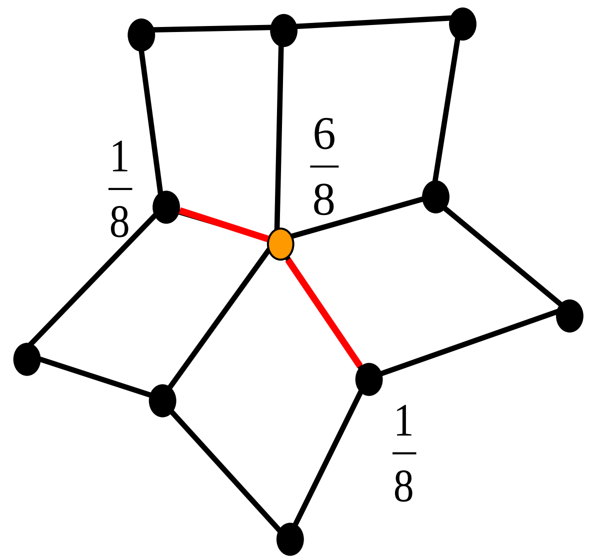
Keep subdividing…

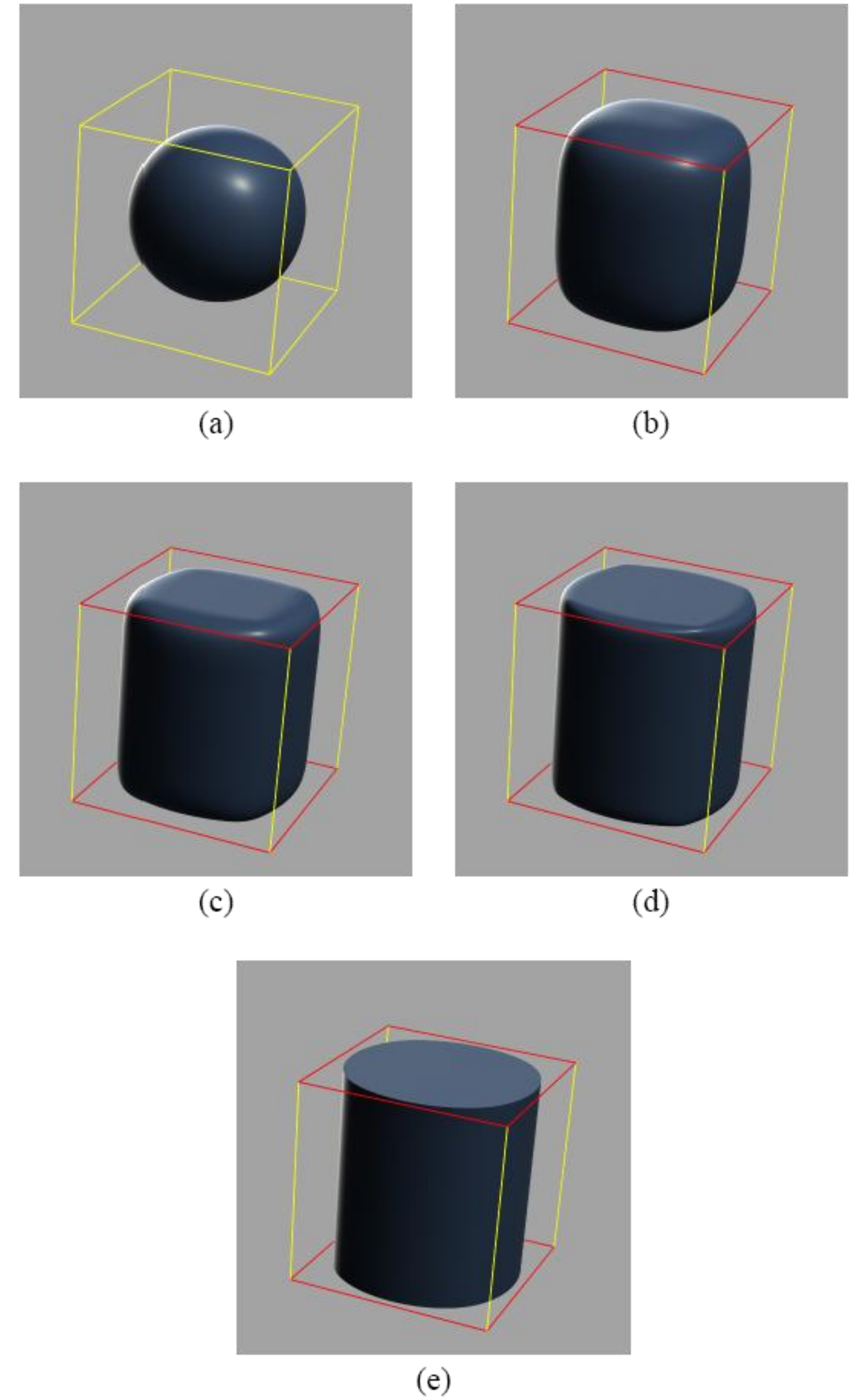
- After 1 iteration, Every new face is a rectangular.
- Extraordinary vertices are forever. Valence is retained.
- Ultimately, at the limit, the surface will be a standard bicubic B-spline surface at every point except at these “extraordinary points”, and therefore $C^{(2)}$ at all but extraordinary points.
- Catmull & Clark did not prove or guarantee continuity at the extraordinary points but note that trials indicate this much.
- Later on it was proven to be C1

Catmull-Clark, special rules

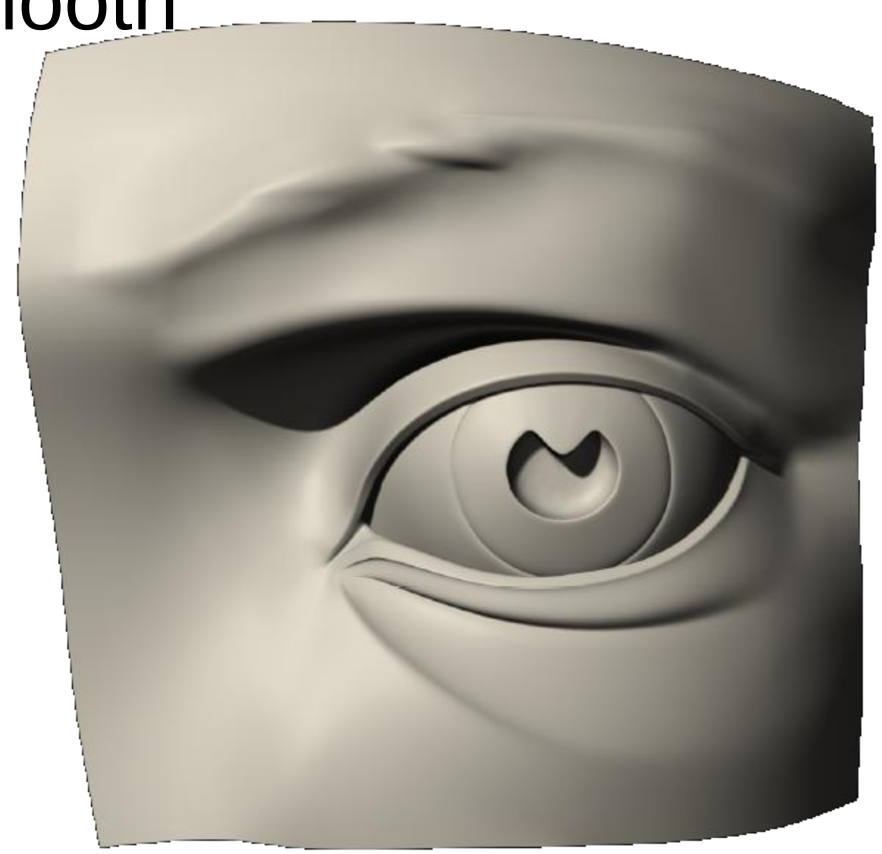
Crease/body masks



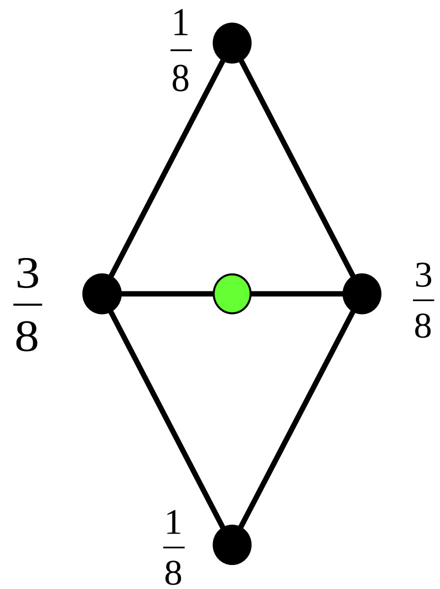


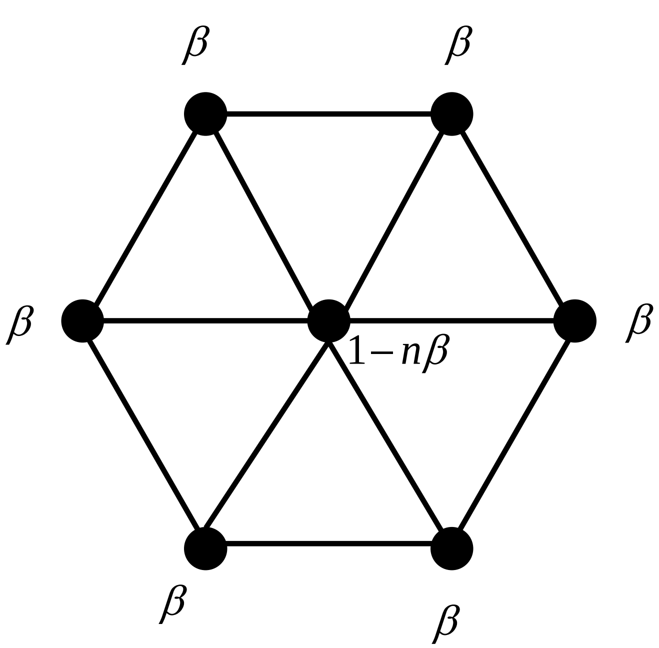


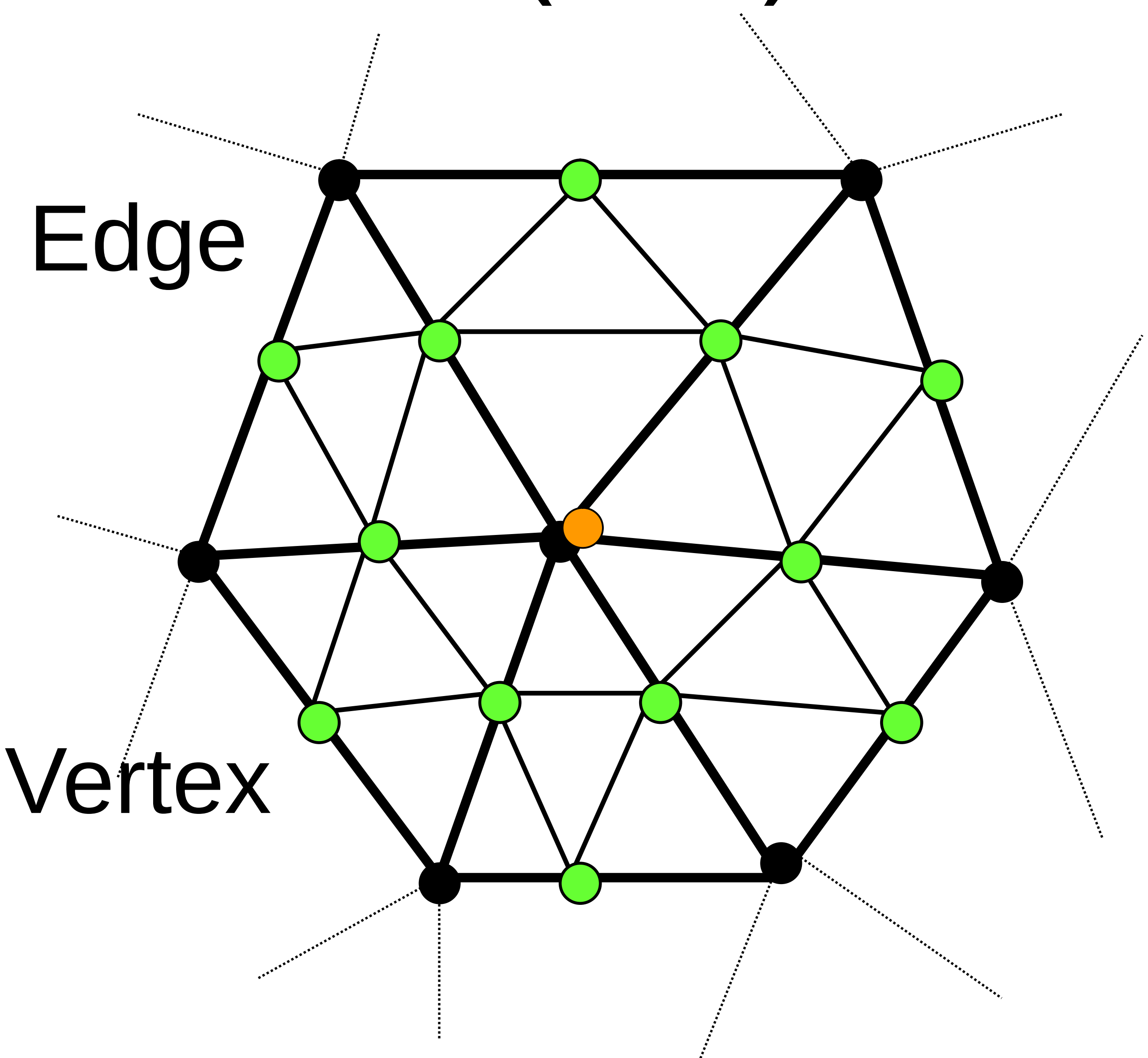
Piecewise smooth surface



Loop's scheme ('87)

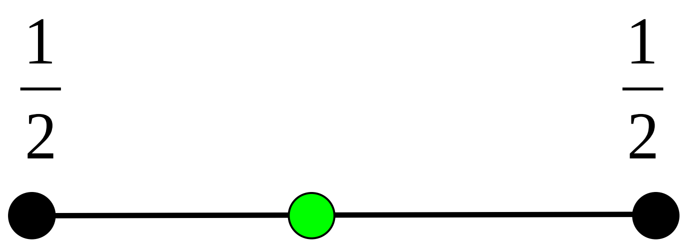


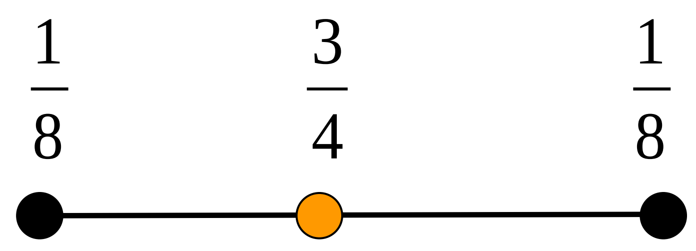




$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{2}{8}\cos\left(\frac{2\pi}{n}\right)\right)^2\right)$$

Loop’s scheme - Boundary



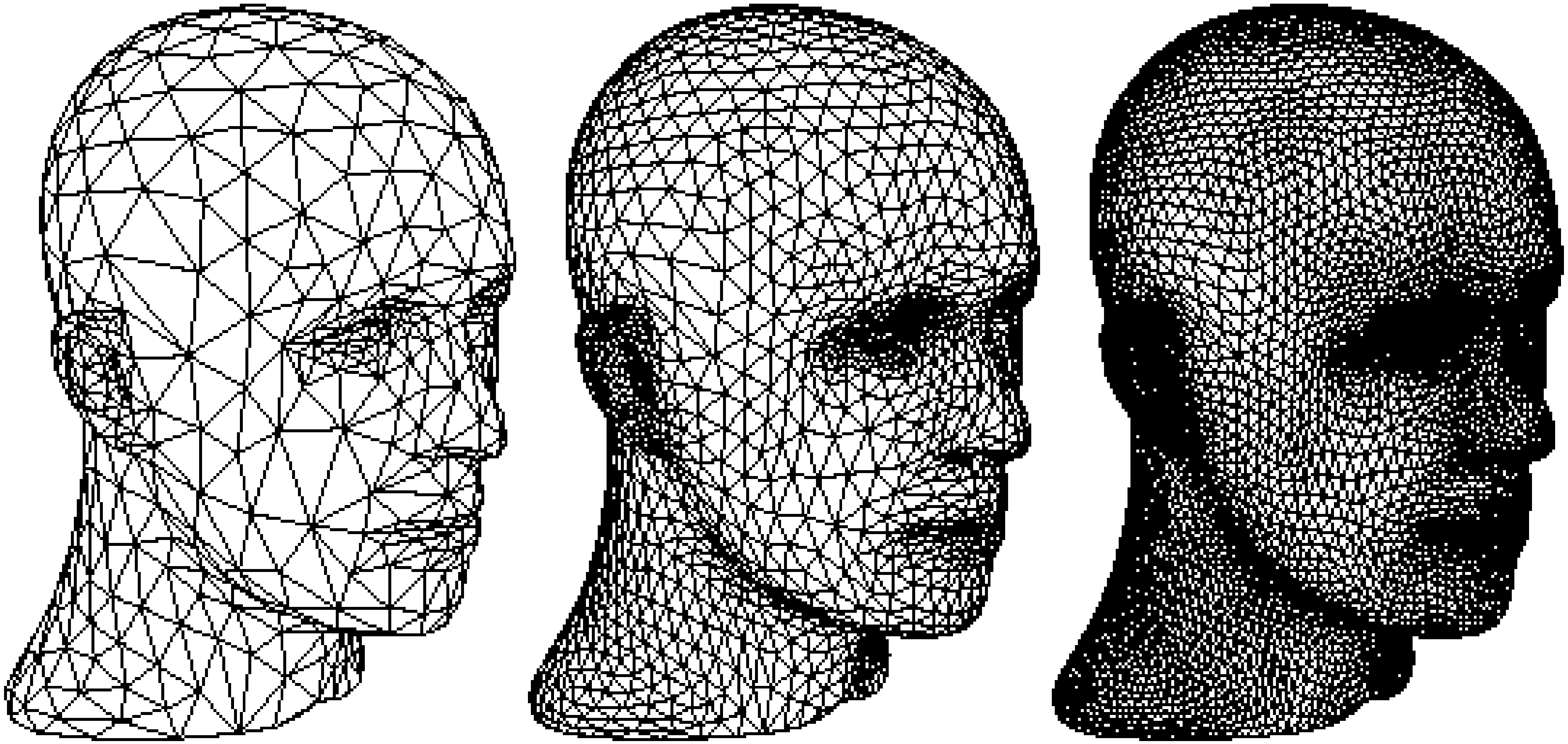


Edge point

Vertex point

- The boundary is a cubic B-Spline curve
- The curve only depends on the control points on the boundary
- Good for connecting 2 meshes
- C0 for irregularities near boundary

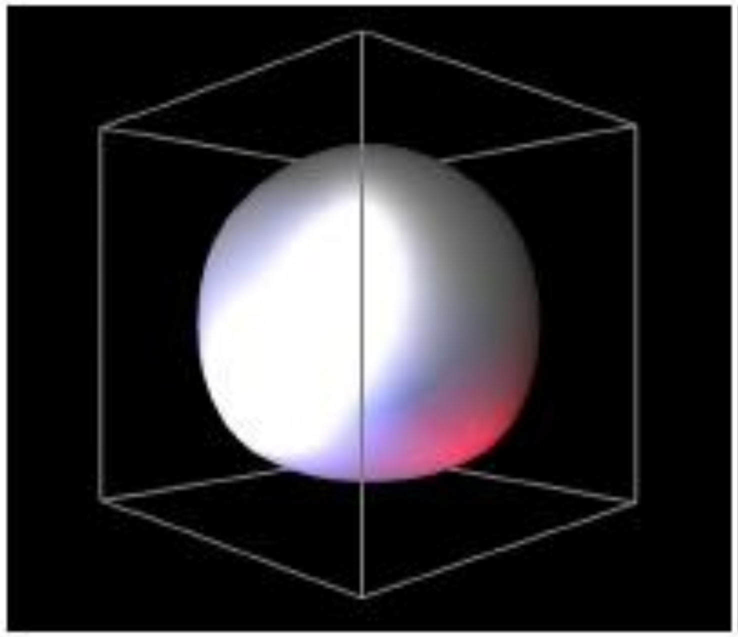
Loop Subdivision Scheme



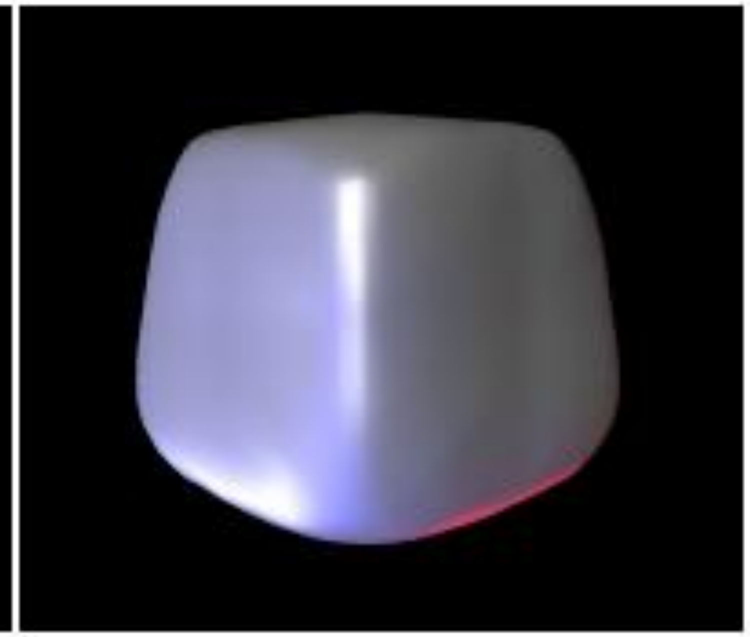
Limit surface has provable smoothness properties!

Zorin & Schroeder
SIGGRAPH 99
Course Notes

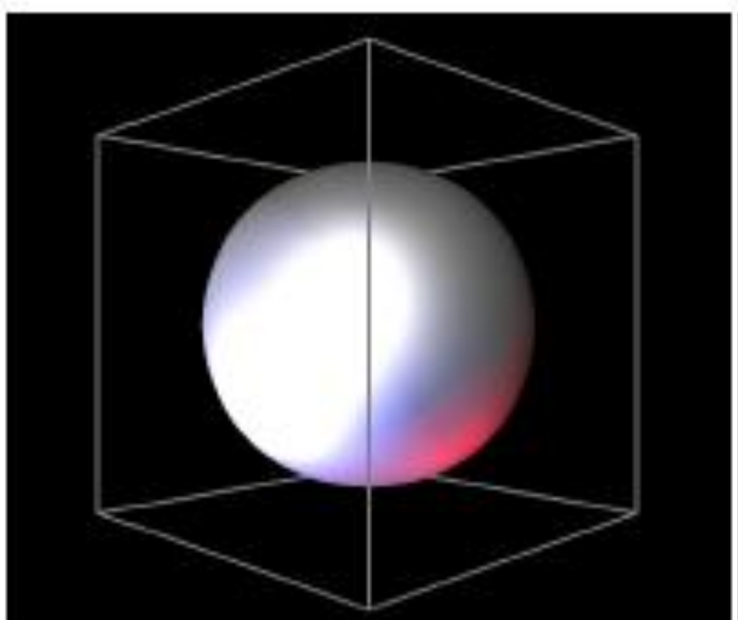
Subdivision Schemes



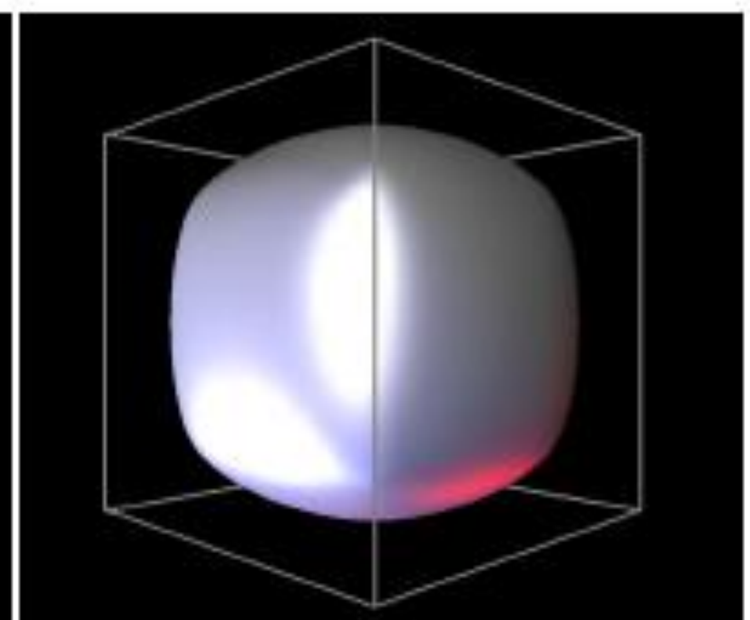
Loop



Butterfly



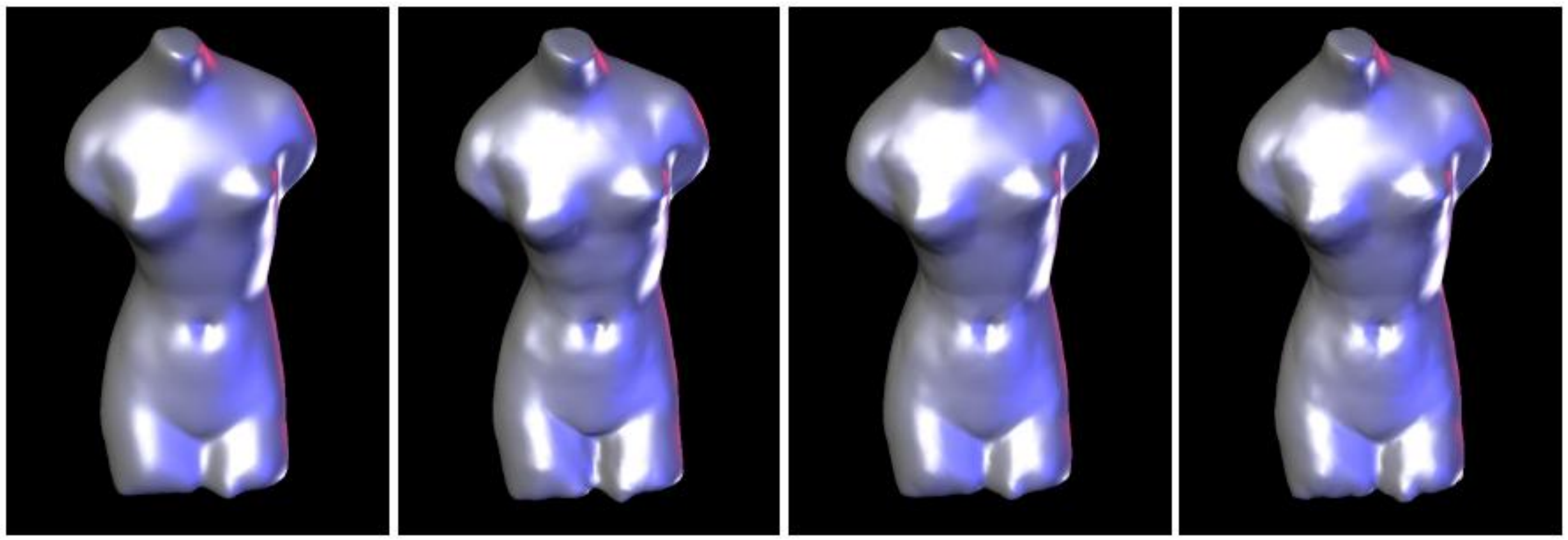
Catmull-Clark



Doo-Sabin

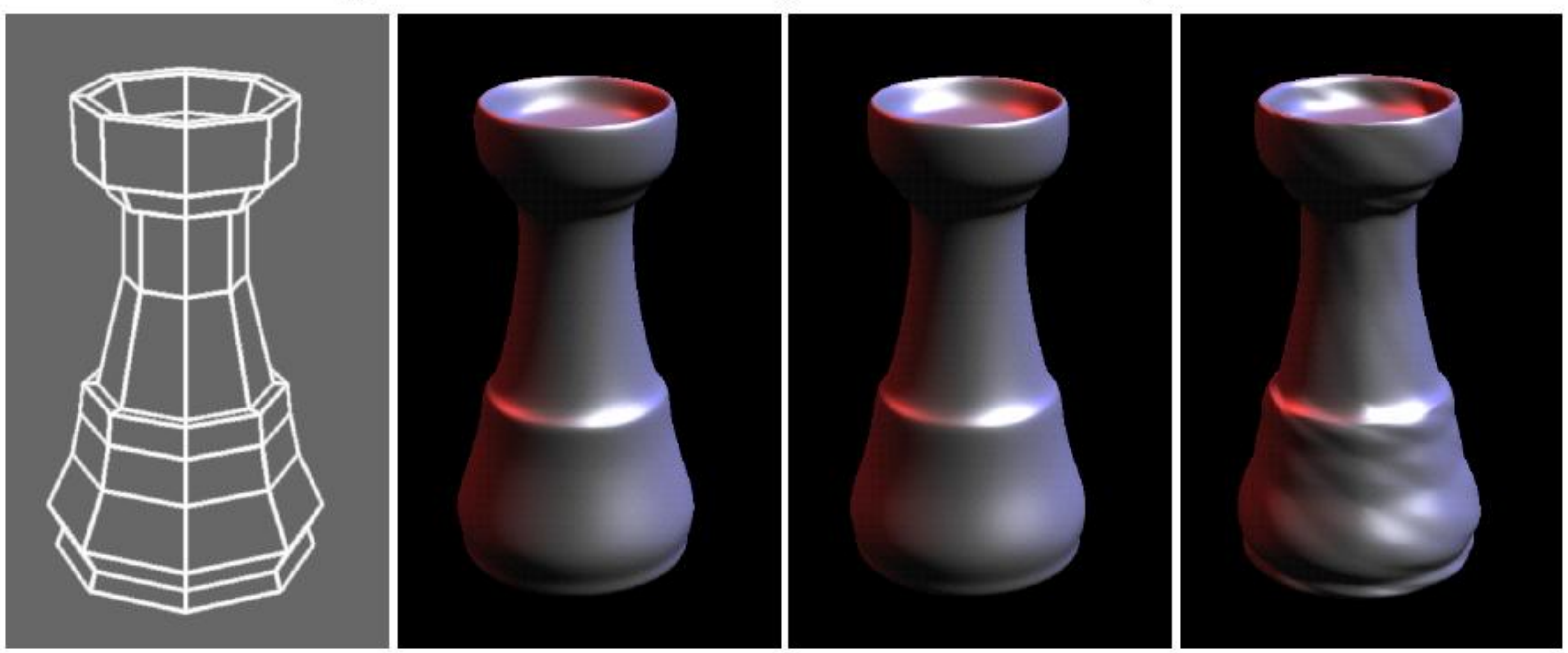
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Visual Comparison – Cont’



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

Different subdivision schemes produce similar results for smooth meshes.



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

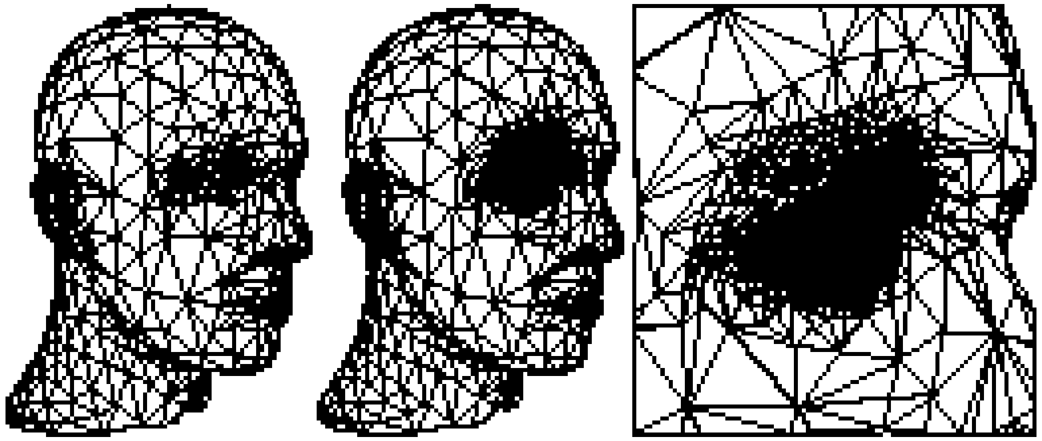
Subdivision Surfaces

- Properties:
  - 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

Pixar

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