

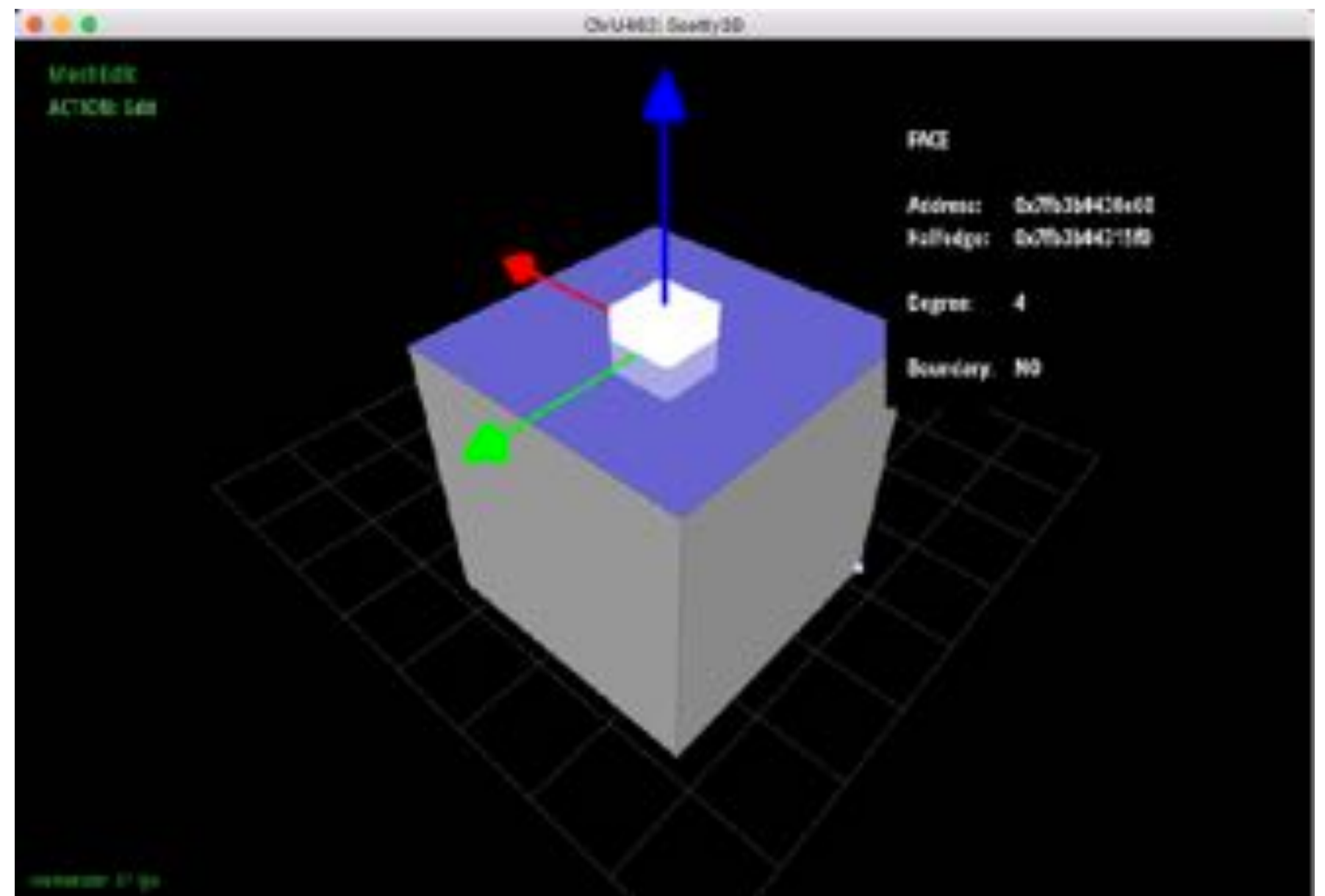
# Introduction to Geometry

---

**Computer Graphics**  
**CMU 15-462/15-662**

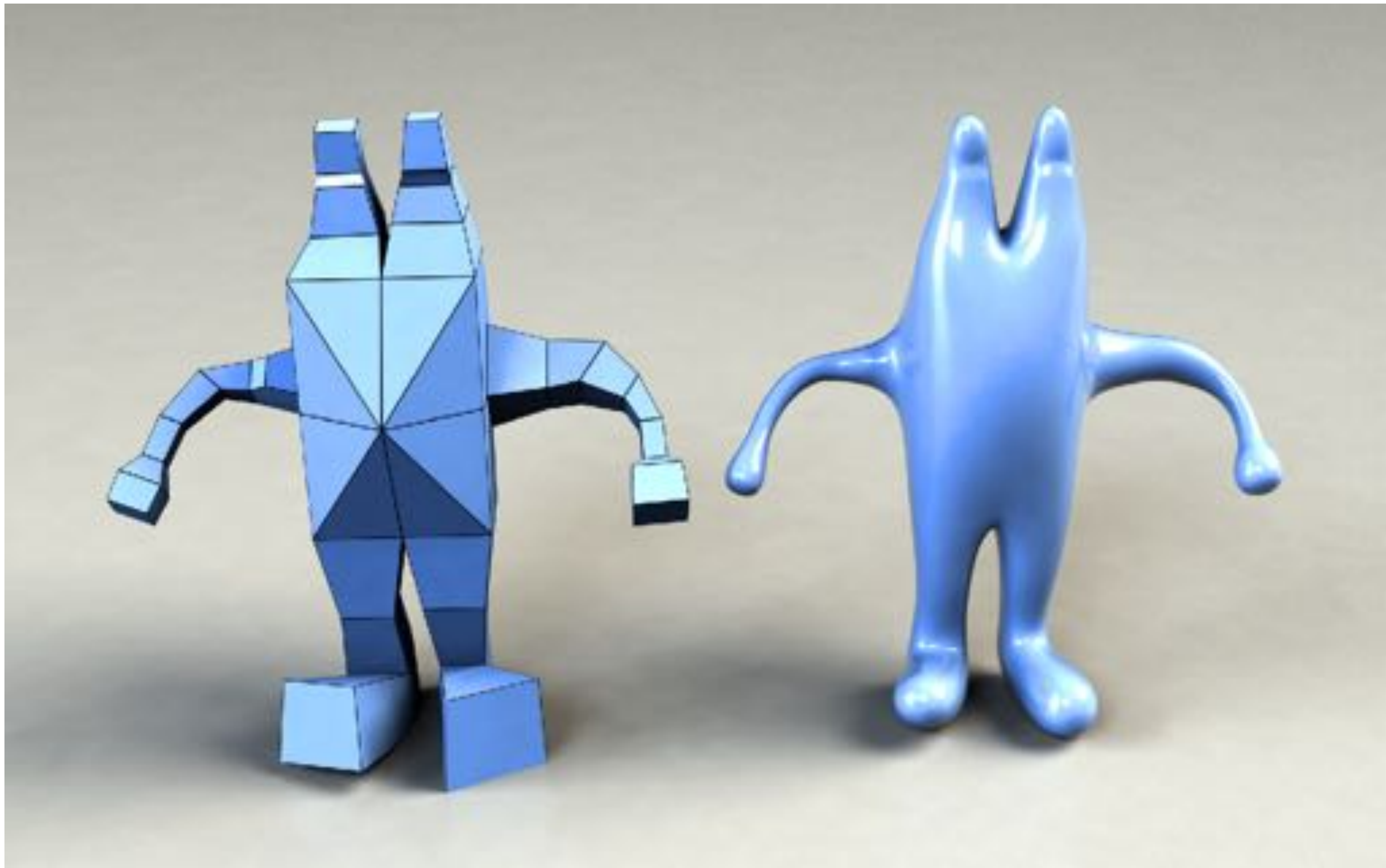
# Assignment 2

- Start building up “Scotty3D”; first part is 3D modeling



**(Start from the cube you described in Lecture 1!)**

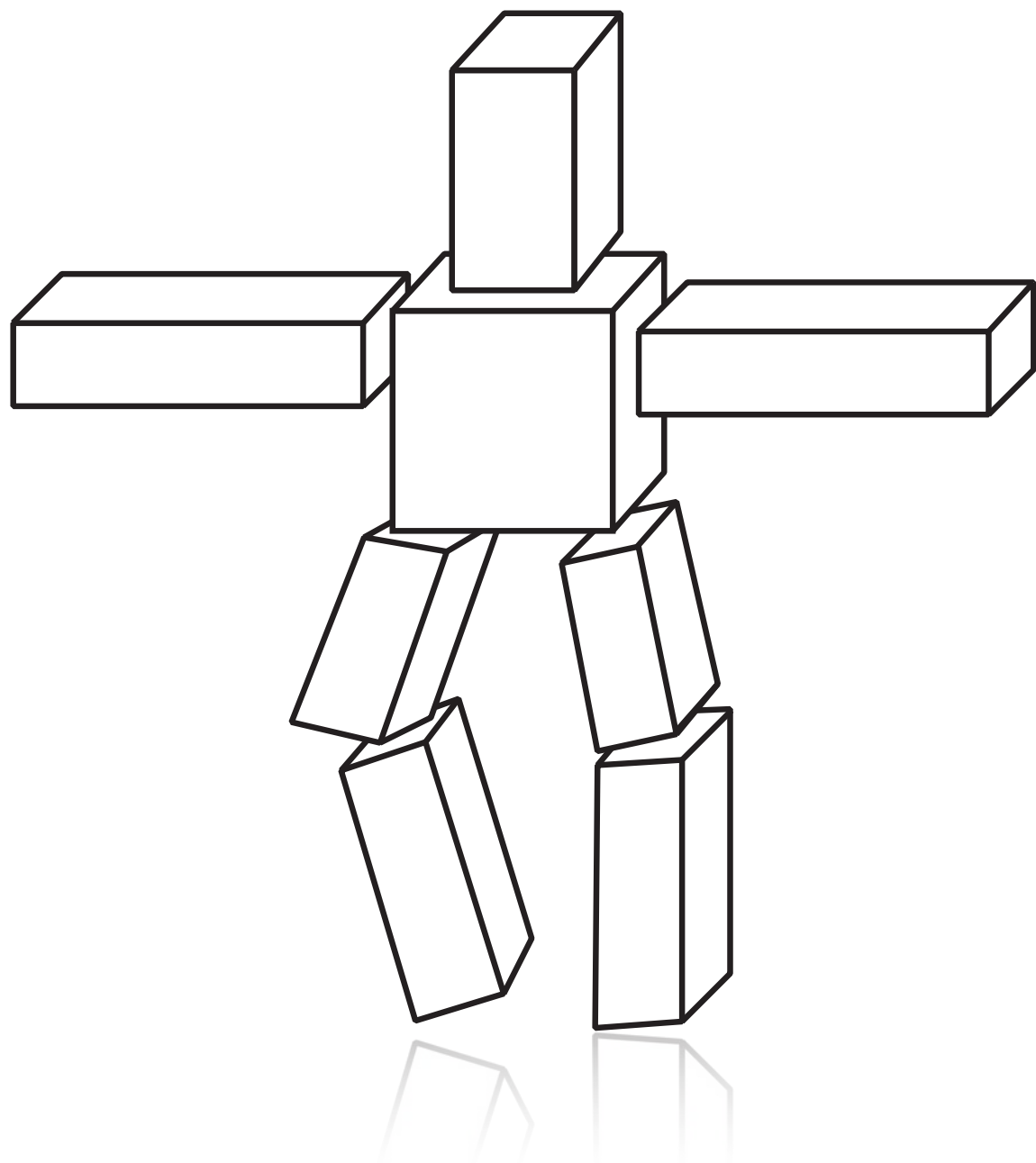
# 3D Modeling Competition



**(This mesh was created in Scotty3D in about 5 minutes... you can do much better!)**

# Increasing the complexity of our models

**Transformations**



**Geometry**



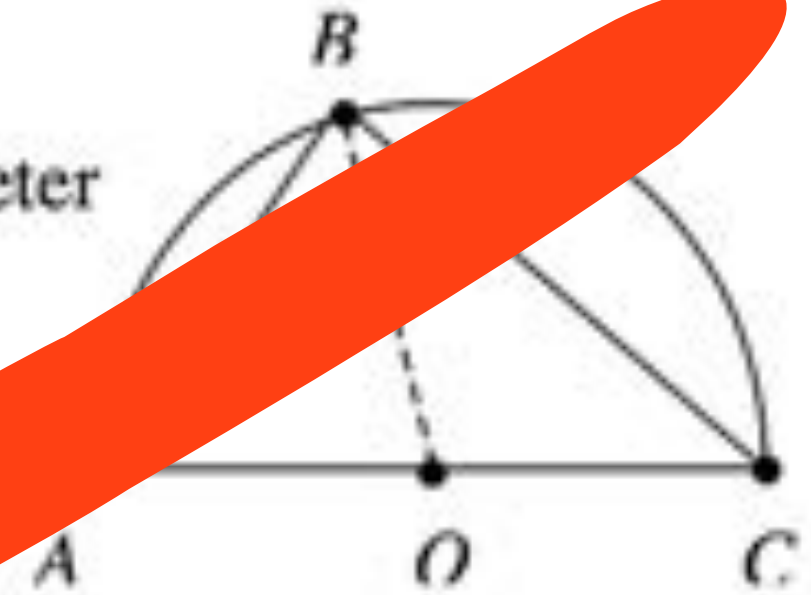
**Materials, lighting, ...**



# Q: What is geometry?

A: Geometry is the study of two-column proofs.

THEOREM 9.5. Let  $\triangle ABC$  be inscribed in a semicircle with diameter  $\overline{AC}$ . Then  $\angle ABC$  is a right angle.



*Proof:*

Statement

1. Draw radius  $OB$ . Then  $OB = OC = OA$ .
2.  $m\angle OBC = m\angle BCA$   
 $m\angle OBA = m\angle BAC$
3.  $m\angle ABC = m\angle OBA + m\angle OBC$
4.  $m\angle ABC + m\angle BCA + m\angle BAC = 180$
5.  $m\angle ABC + m\angle OBA + m\angle OBC = 180$
6.  $2m\angle ABC = 180$
7.  $m\angle ABC = 90$
8.  $\angle ABC$  is a right angle

Given

1. Isosceles Triangle Theorem
2. Angle Addition Postulate
3. The sum of the angles of a triangle is 180
4. Substitution (line 1)
5. Substitution (line 3)
6. Substitution (line 3)
7. Division Property of Equality
8. Definition of Right Angle

**Ceci n'est pas géométrie.**

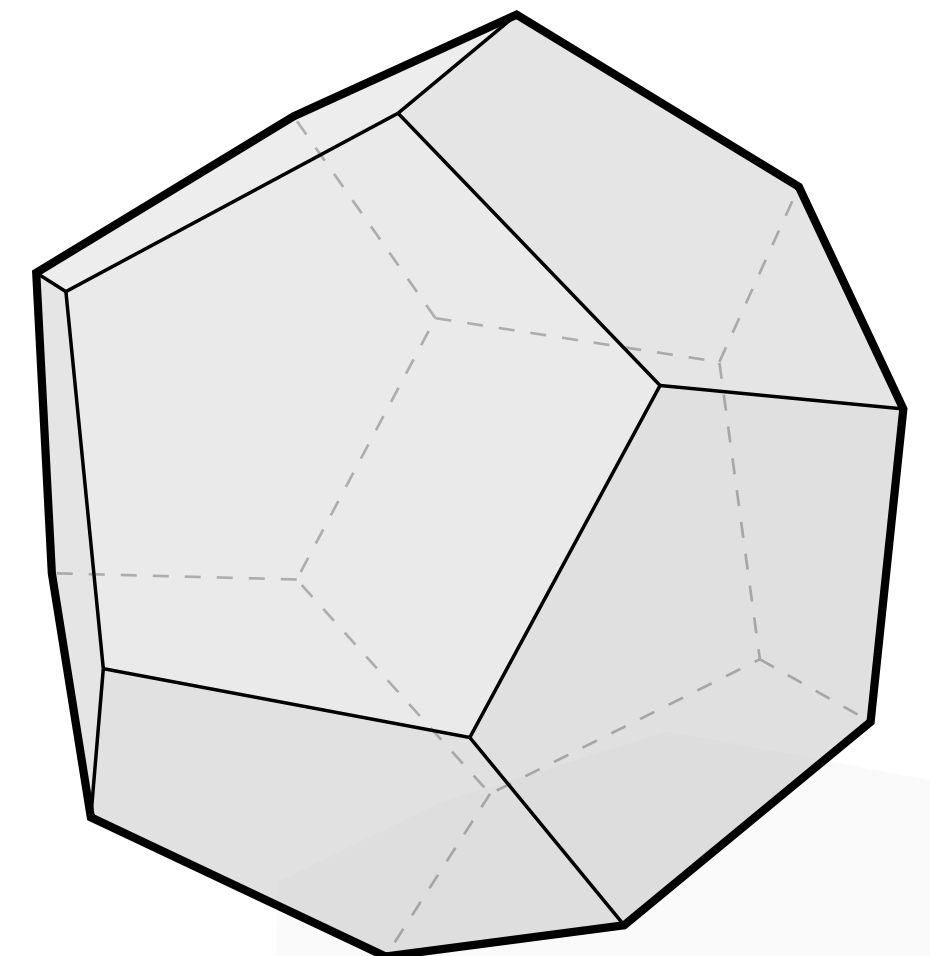
# What is geometry?

“Earth”

“measure”

**ge • om • et • ry** /jē'ämətrē/ *n.*

1. The study of shapes, sizes, patterns, and positions.
2. The study of spaces where some quantity (lengths, angles, etc.) can be *measured*.



**Plato: “...the earth is in appearance like one of those balls which have leather coverings in twelve pieces...”**

# How can we describe geometry?

**IMPLICIT**

$$x^2 + y^2 = 0$$

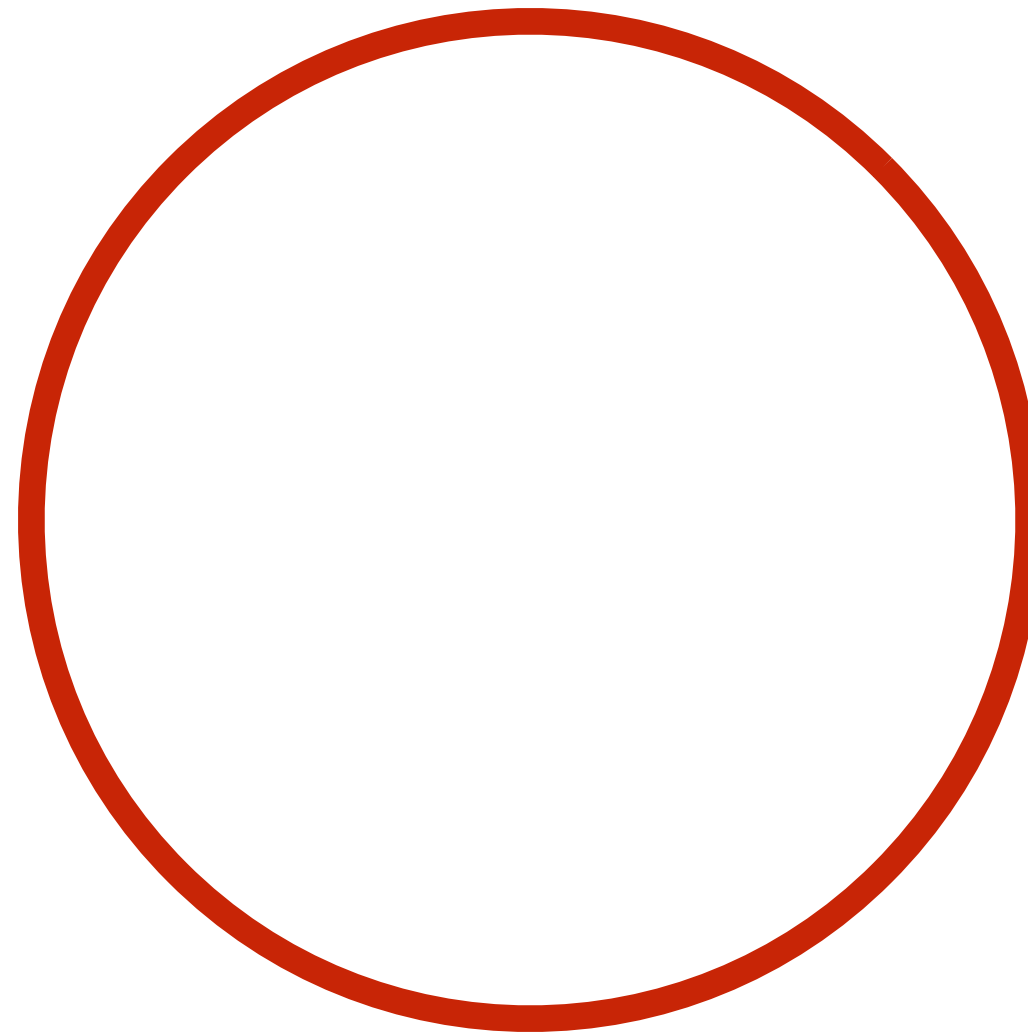
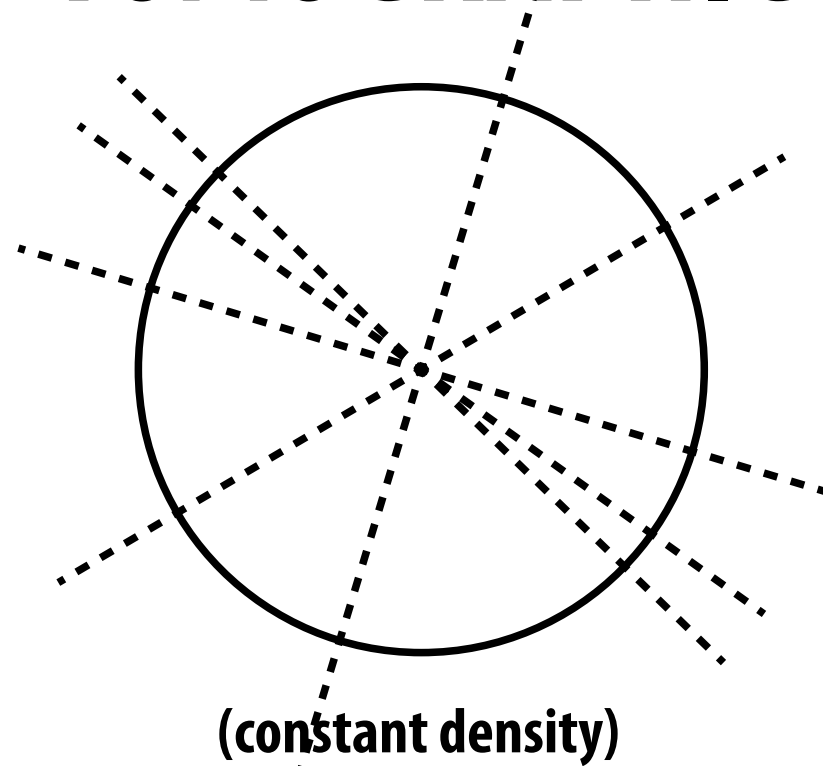
**LINGUISTIC**

“unit circle”

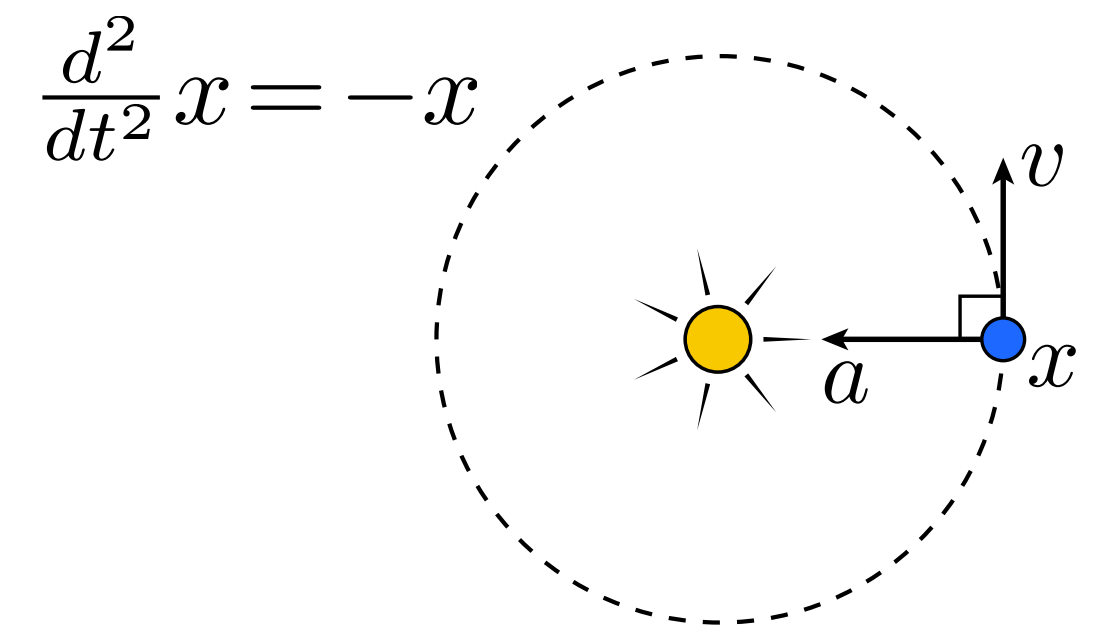
**EXPLICIT**

$$\underbrace{(\cos \theta)}_x, \underbrace{(\sin \theta)}_y$$

**TOMOGRAPHIC**



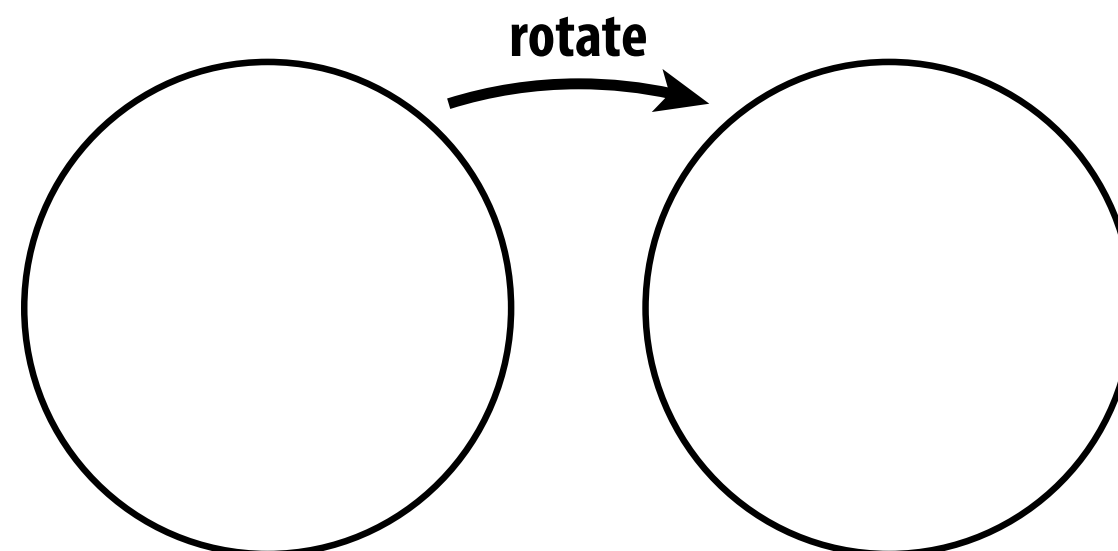
**DYNAMIC**



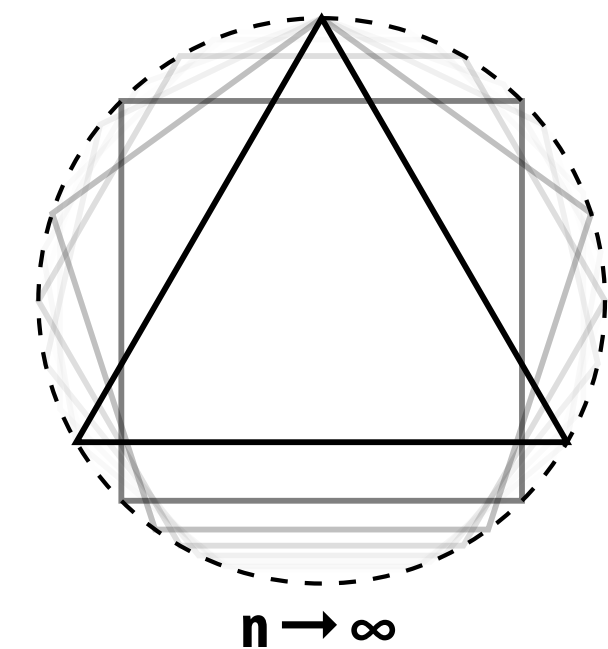
**CURVATURE**

$$\kappa = 1$$

**SYMMETRIC**



**DISCRETE**



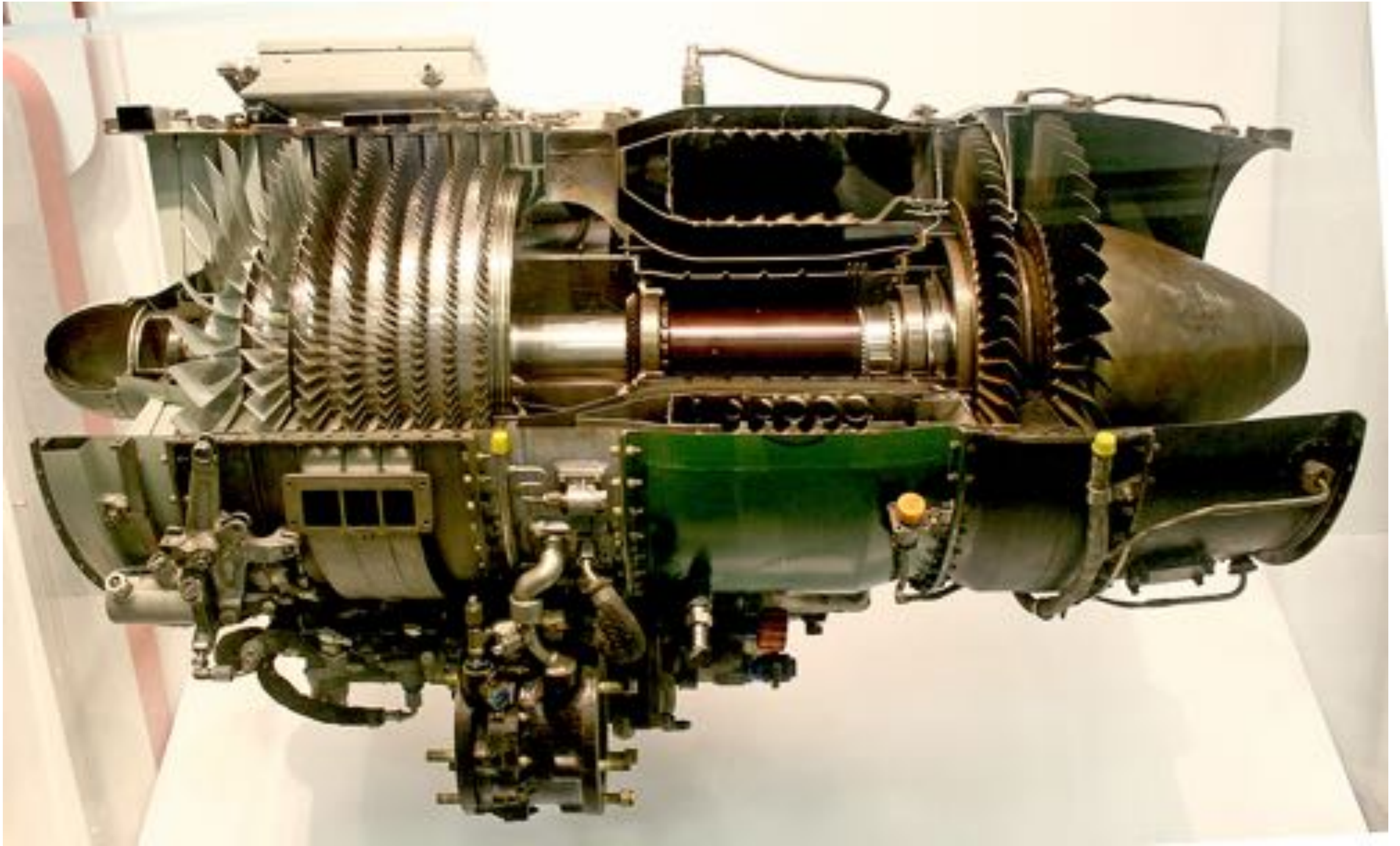
**Given all these options, what's the best way to encode geometry on a computer?**



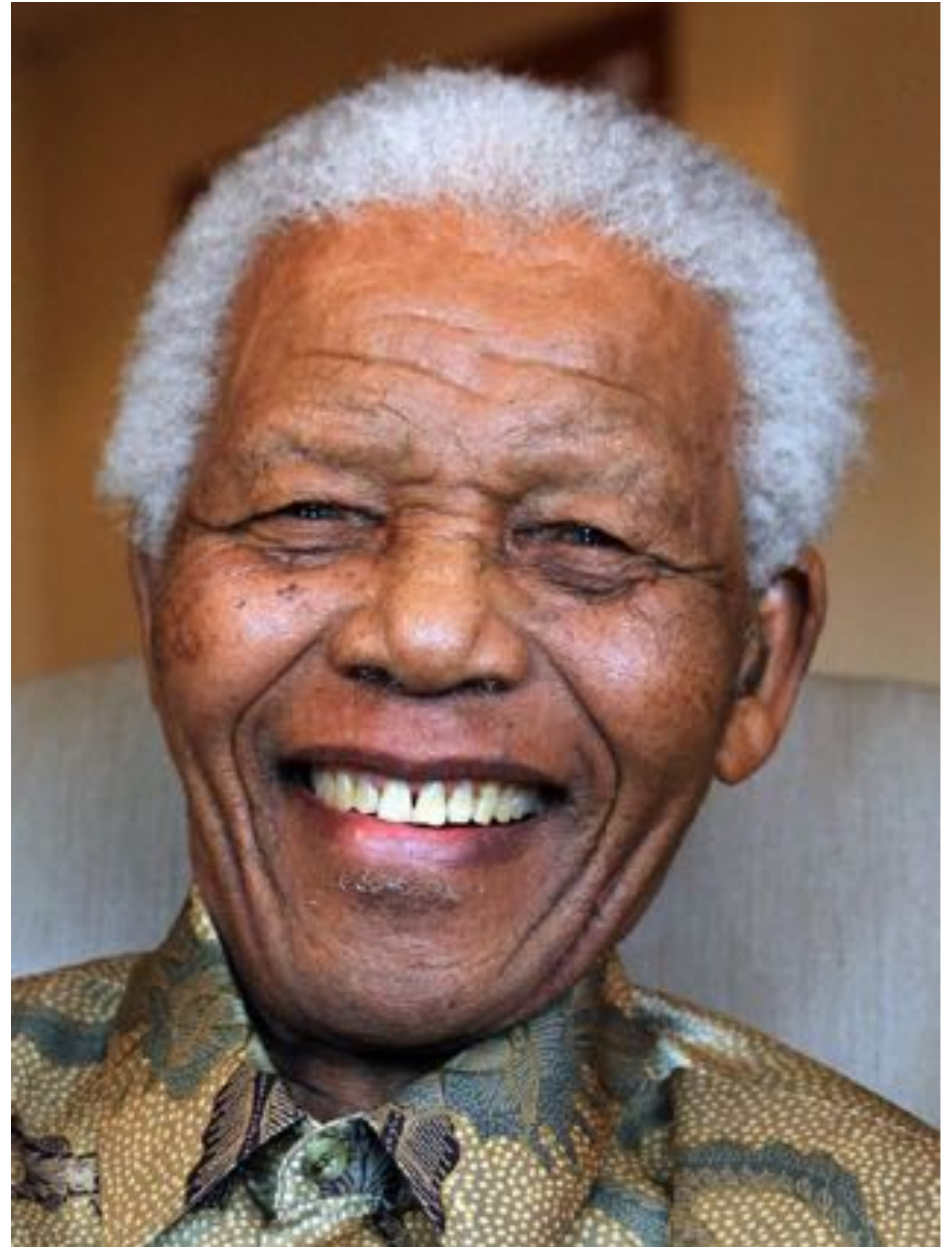
# Examples of geometry



# Examples of geometry



# Examples of geometry



# Examples of geometry



# Examples of geometry



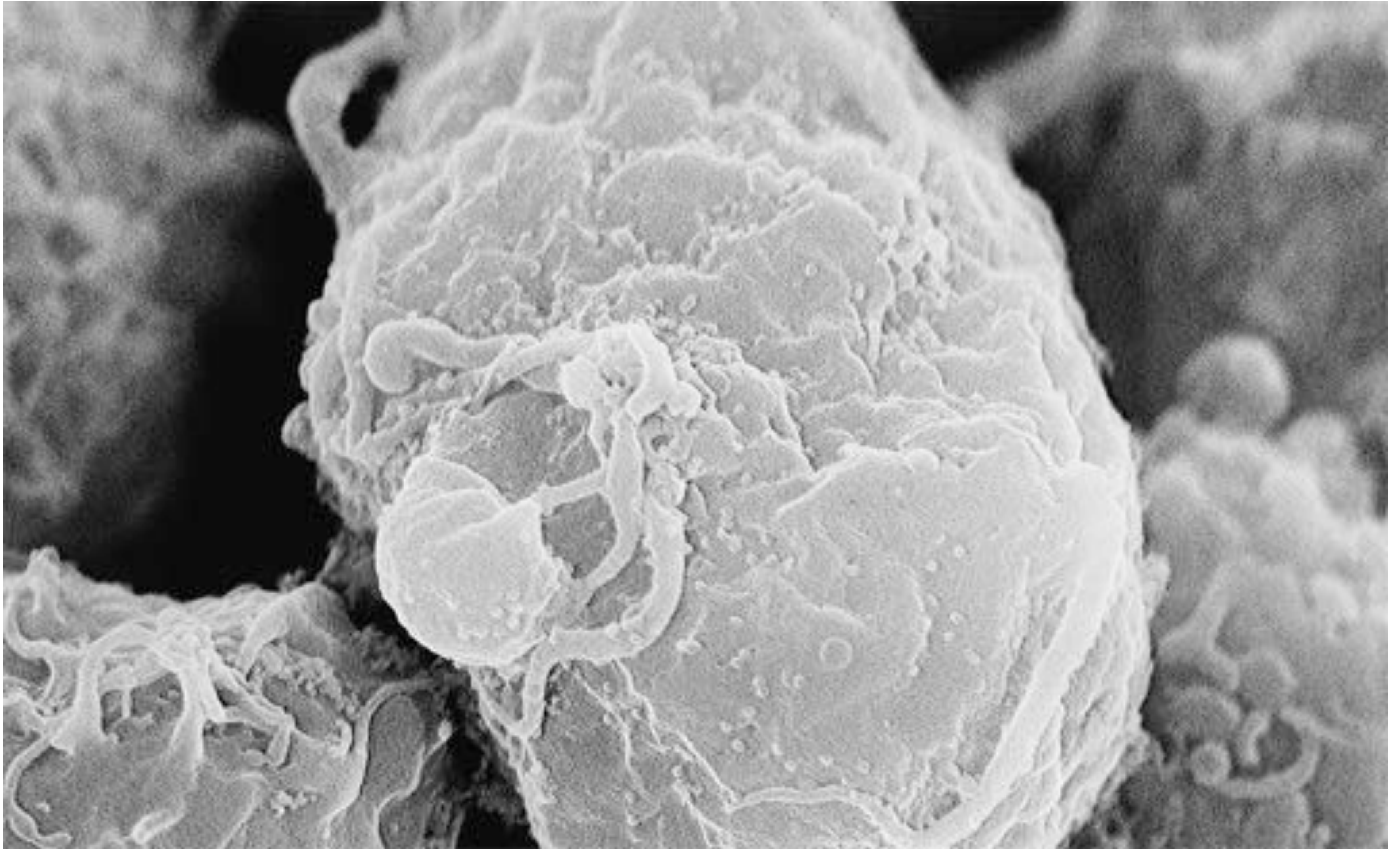
# Examples of geometry



# Examples of geometry

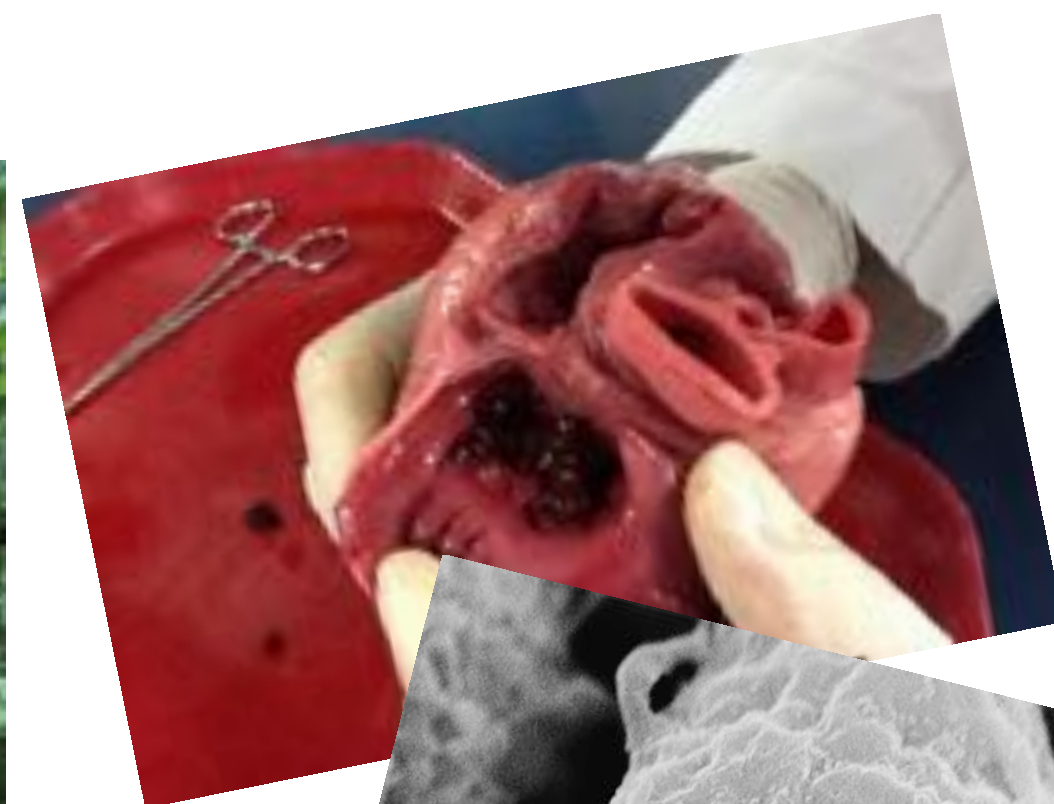


# Examples of geometry





# It's a Jungle Out There!



# **No one “best” choice—geometry is hard!**

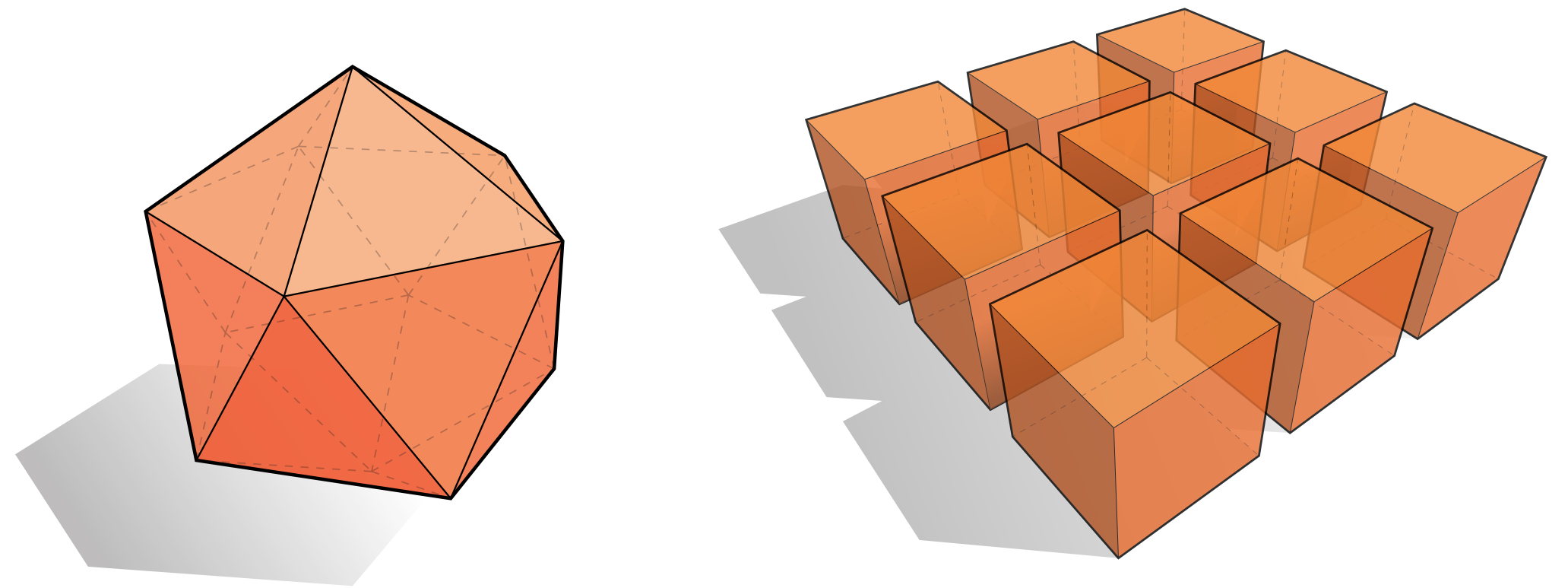
**“I hate meshes.  
I cannot believe how hard this is.  
Geometry is hard.”**

**—David Baraff**  
**Senior Research Scientist**  
**Pixar Animation Studios**

# Many ways to digitally encode geometry

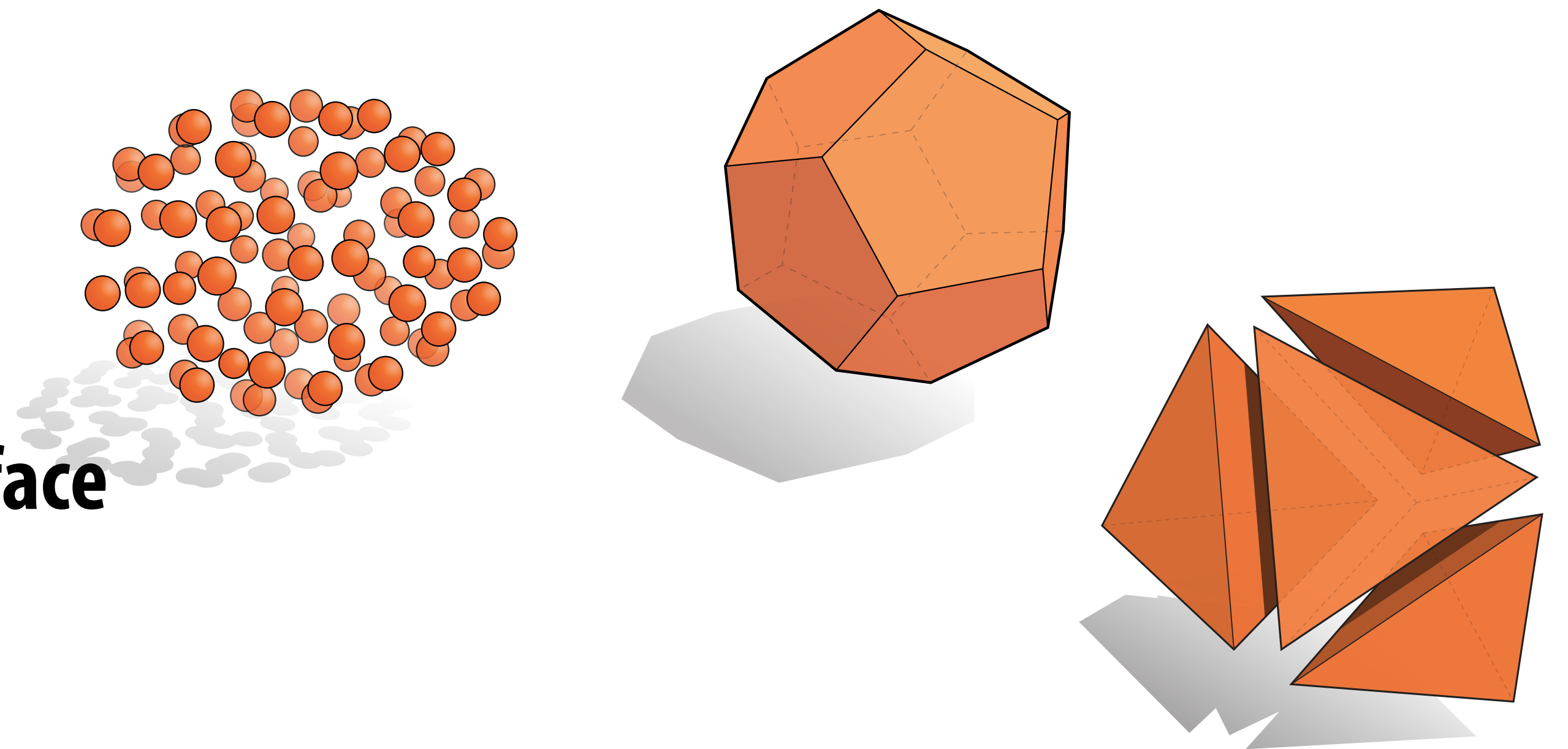
## ■ EXPLICIT

- point cloud
- polygon mesh
- subdivision, NURBS
- ...



## ■ IMPLICIT

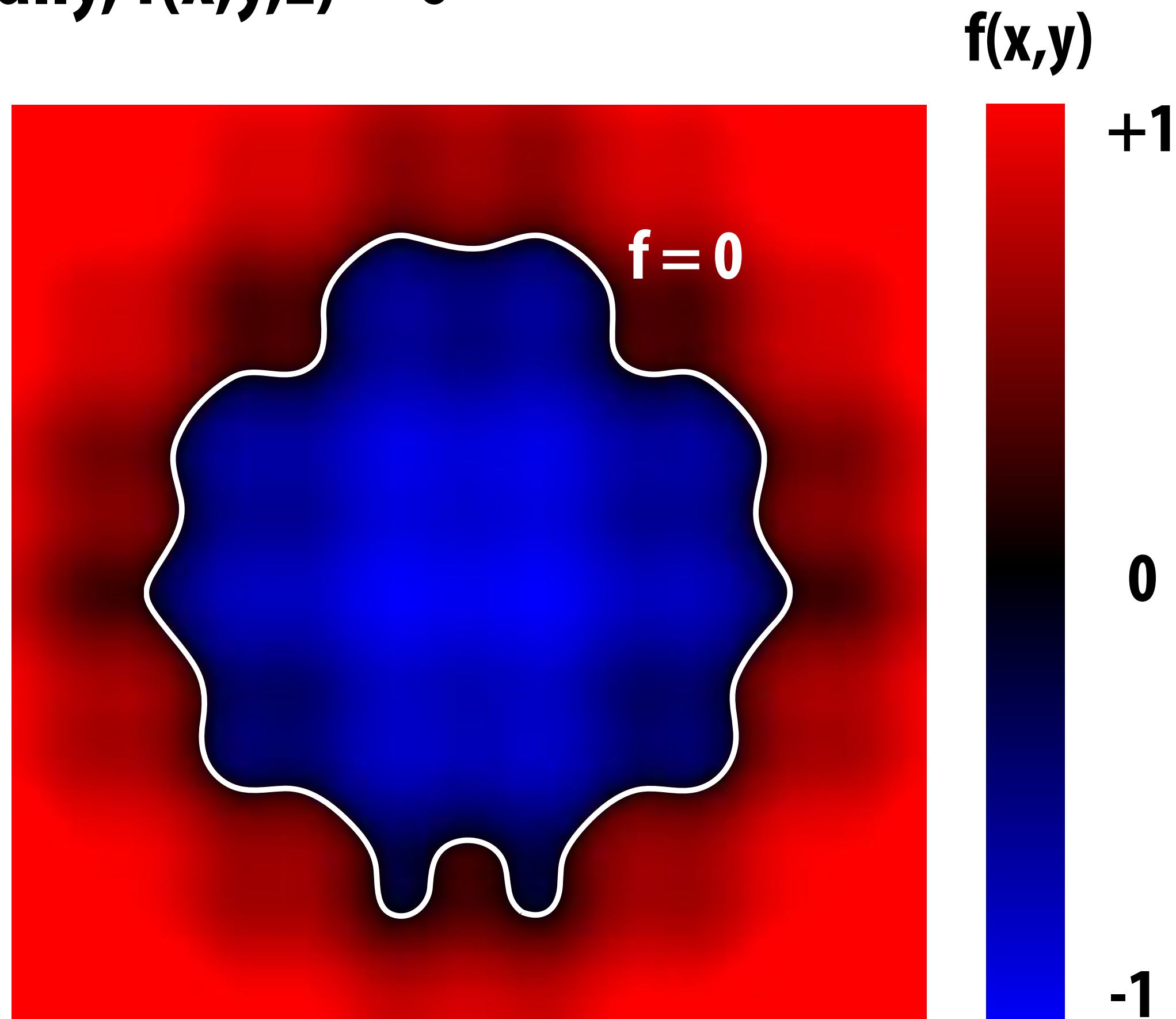
- level set
- algebraic surface
- L-systems
- ...



■ Each choice best suited to a different task/type of geometry

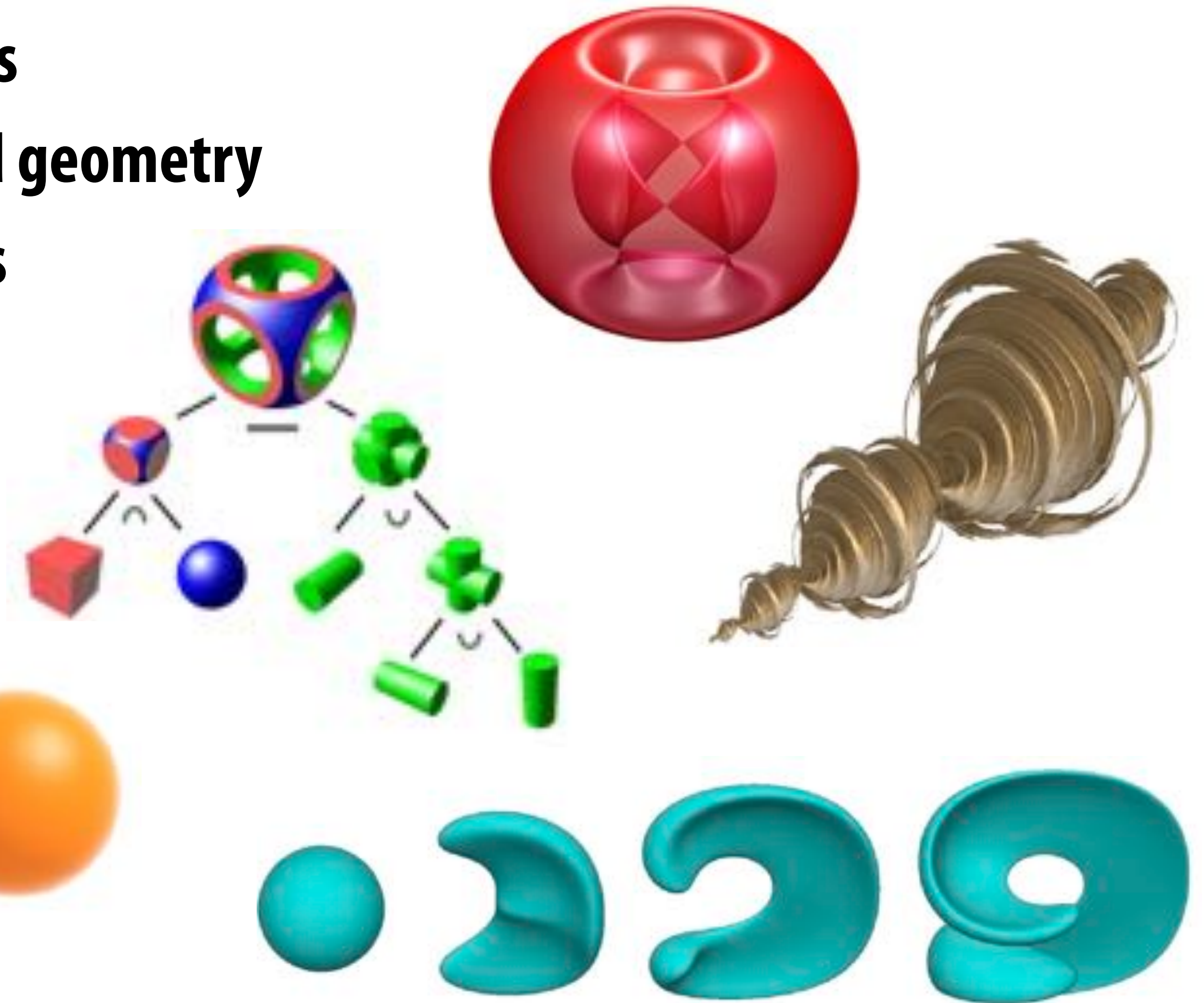
# “Implicit” Representations of Geometry

- Points aren't known directly, but satisfy some relationship
- E.g., unit sphere is all points such that  $x^2+y^2+z^2=1$
- More generally,  $f(x,y,z) = 0$



# Many implicit representations in graphics

- algebraic surfaces
- constructive solid geometry
- level set methods
- blobby surfaces
- fractals
- ...



(Will see some of these a bit later.)

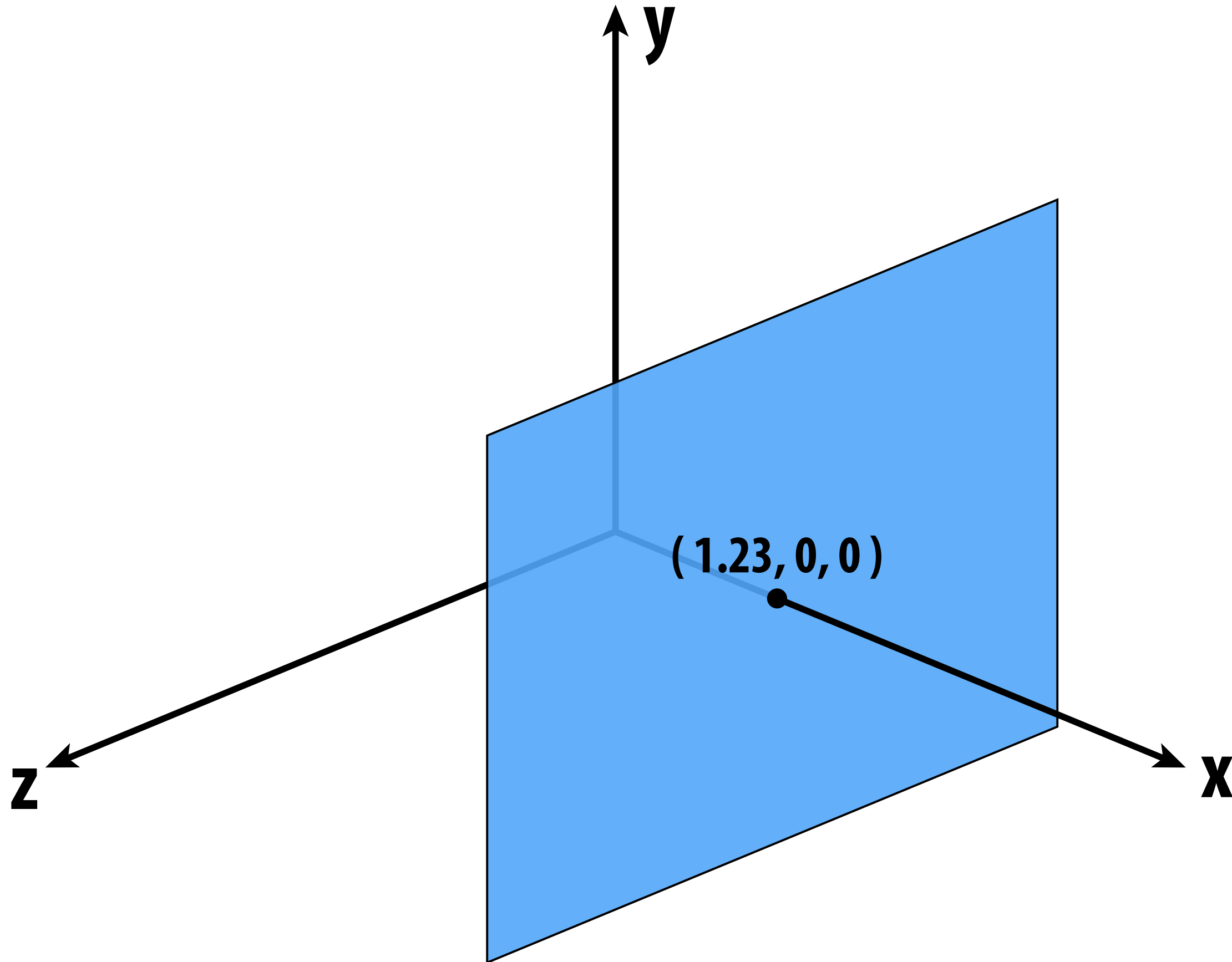
**But first, let's play a game:**

**I'm thinking of an implicit surface  $f(x,y,z)=0$ .**

**Find any point on it.**

# Give up?

My function was  $f(x,y,z) = x - 1.23$  (a plane):



**Implicit surfaces make some tasks hard (like sampling).**

**Let's play another game.**

**I have a new surface  $f(x,y,z) = x^2 + y^2 + z^2 - 1$**

**I want to see if a point is inside it.**



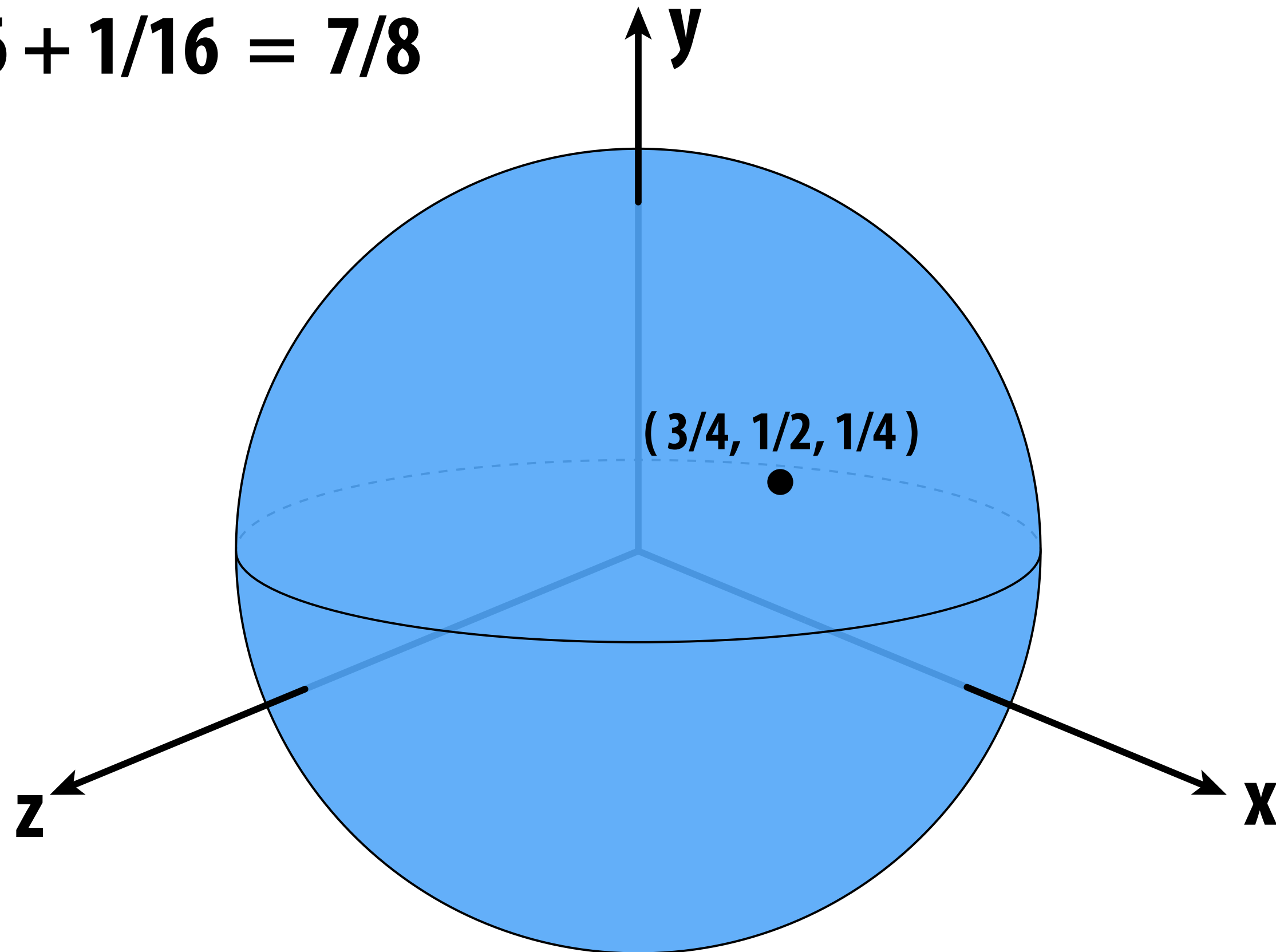
# Check if this point is inside the unit sphere

How about the point  $(3/4, 1/2, 1/4)$ ?

$$9/16 + 4/16 + 1/16 = 7/8$$

$$7/8 < 1$$

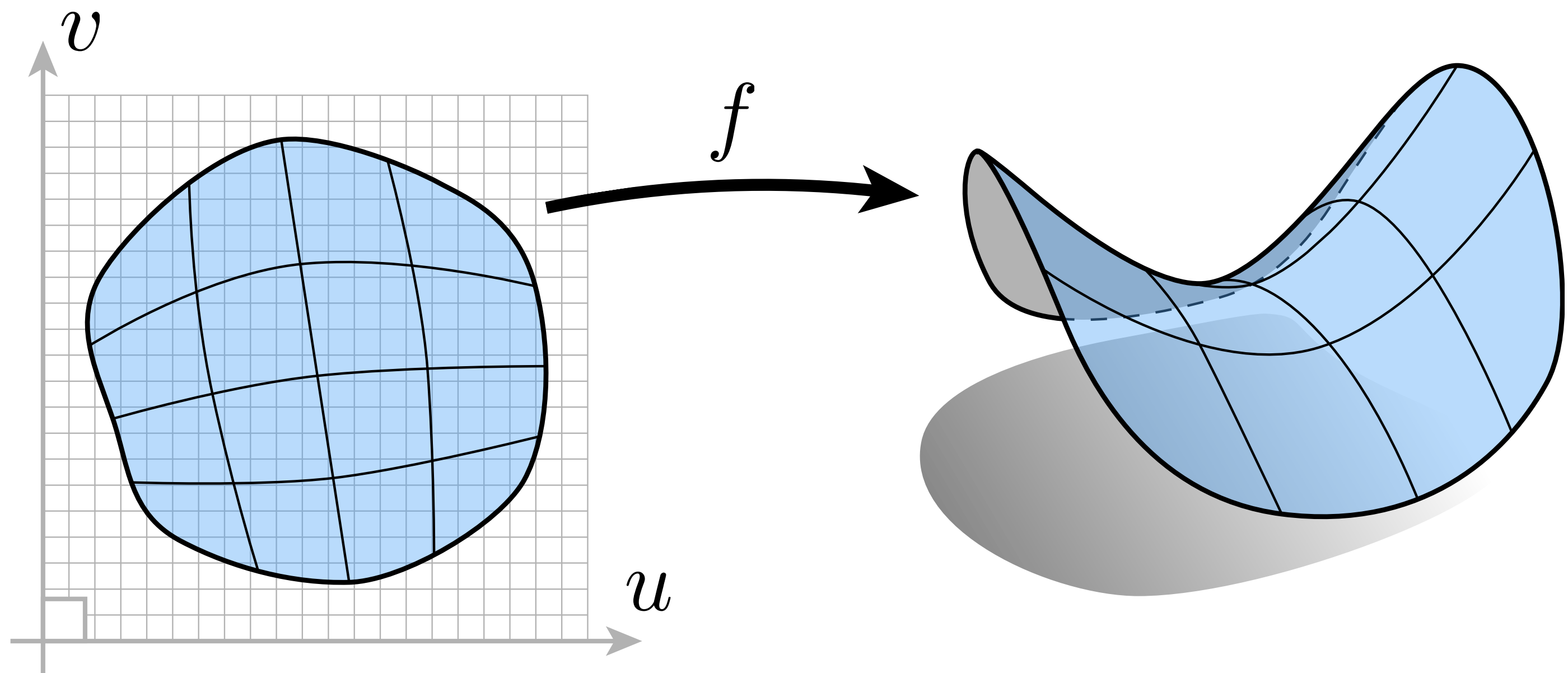
**YES.**



**Implicit surfaces make other tasks easy (like inside/outside tests).**

# “Explicit” Representations of Geometry

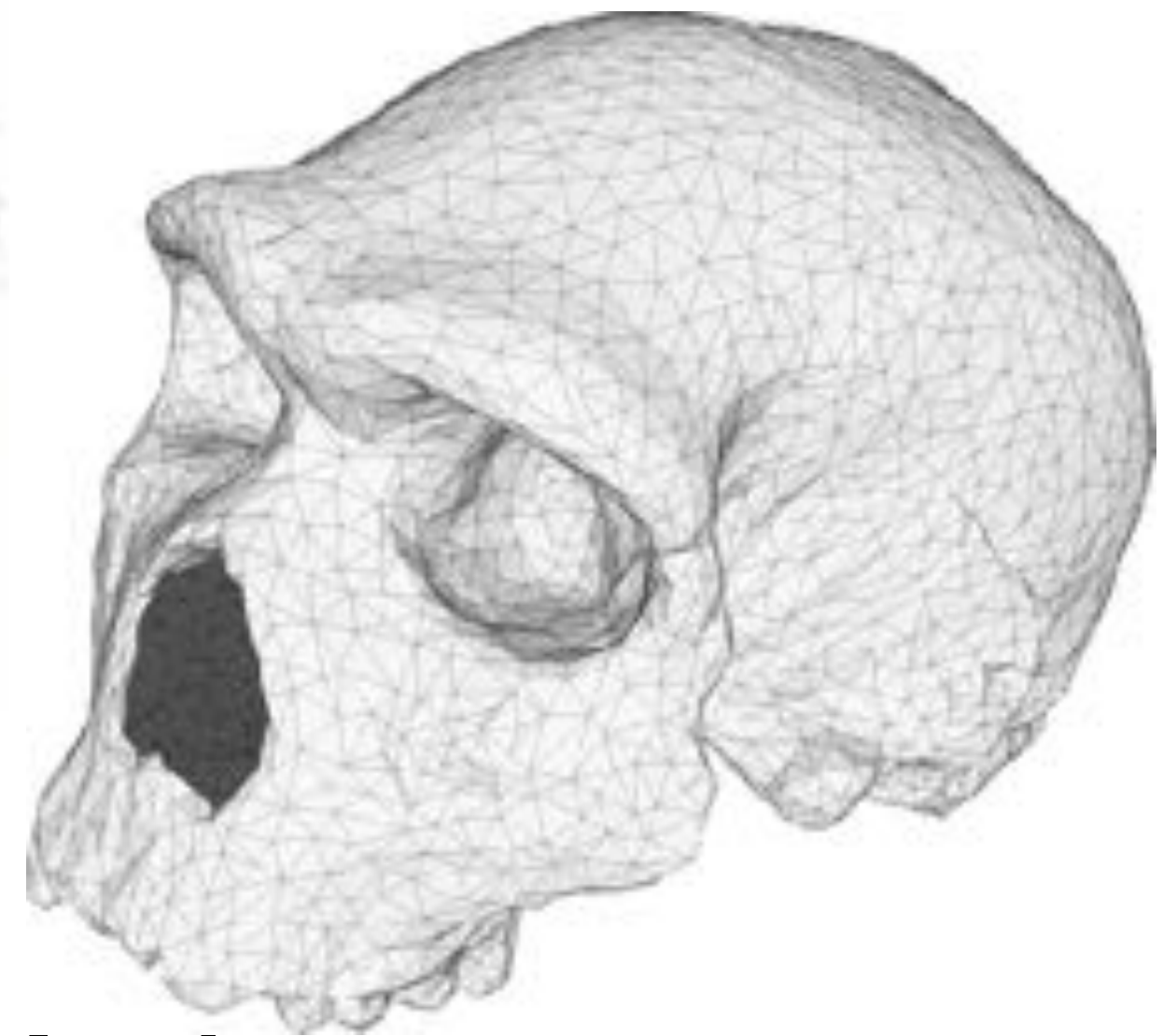
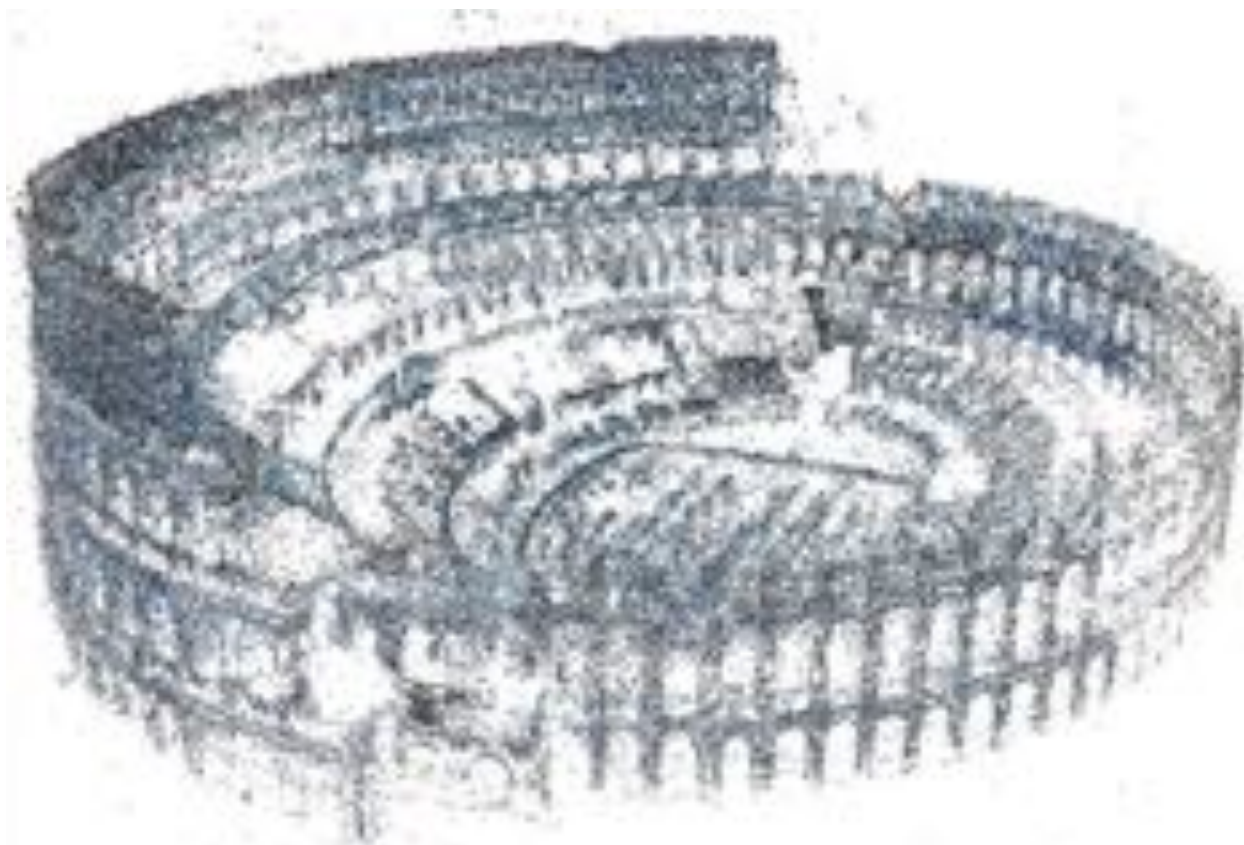
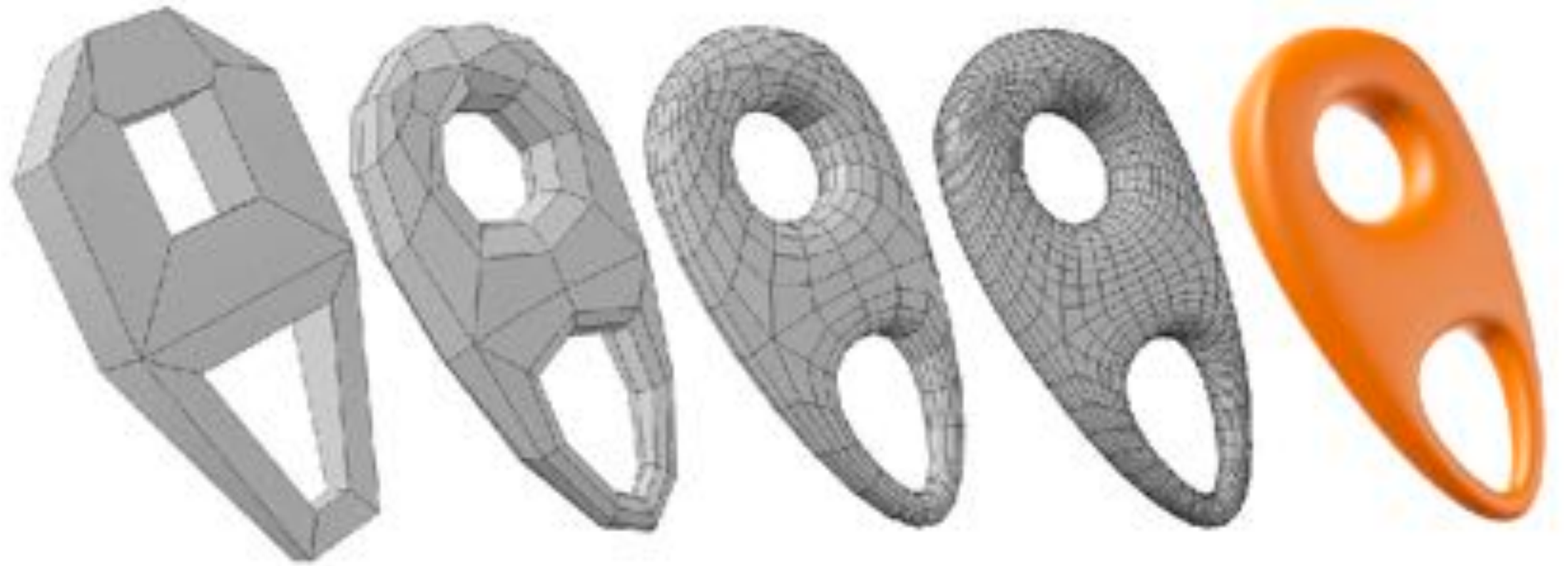
- All points are given directly
- E.g., points on sphere are  $(\cos(u) \sin(v), \sin(u) \sin(v), \cos(v))$ ,  
for  $0 \leq u < 2\pi$  and  $0 \leq v \leq \pi$
- More generally:  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



- (Might have a bunch of these maps, e.g., one per triangle!)

# Many explicit representations in graphics

- triangle meshes
- polygon meshes
- subdivision surfaces
- NURBS
- point clouds
- ...



**(Will see some of these a bit later.)**

**But first, let's play a game:**

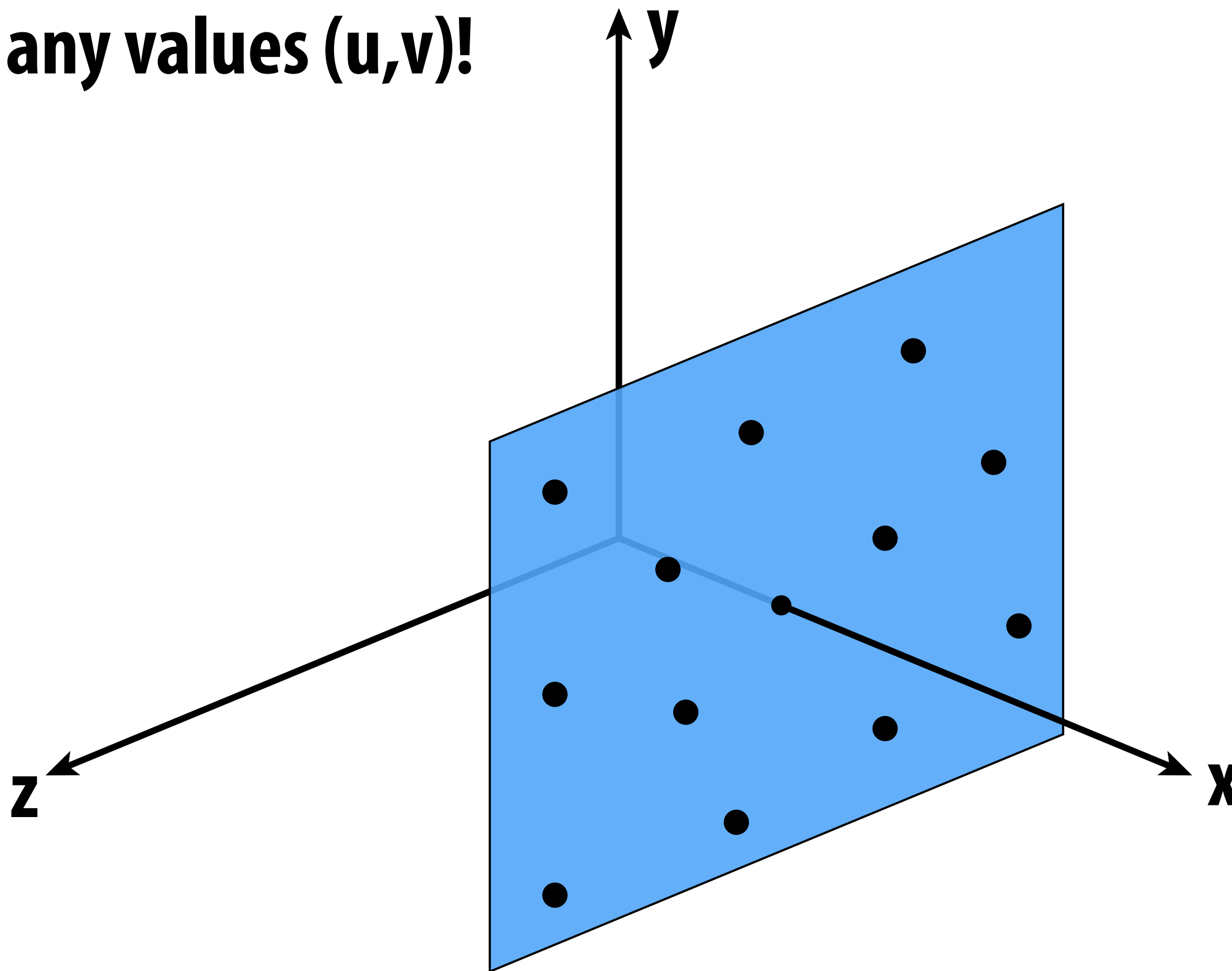
**I'll give you an explicit surface.**

**You give me some points on it.**

# Sampling an explicit surface

My surface is  $f(u, v) = (1.23, u, v)$ .

Just plug in any values  $(u, v)$ !



Explicit surfaces make some tasks easy (like sampling).

**Let's play another game.**

**I have a new surface  $f(u,v)$ .**

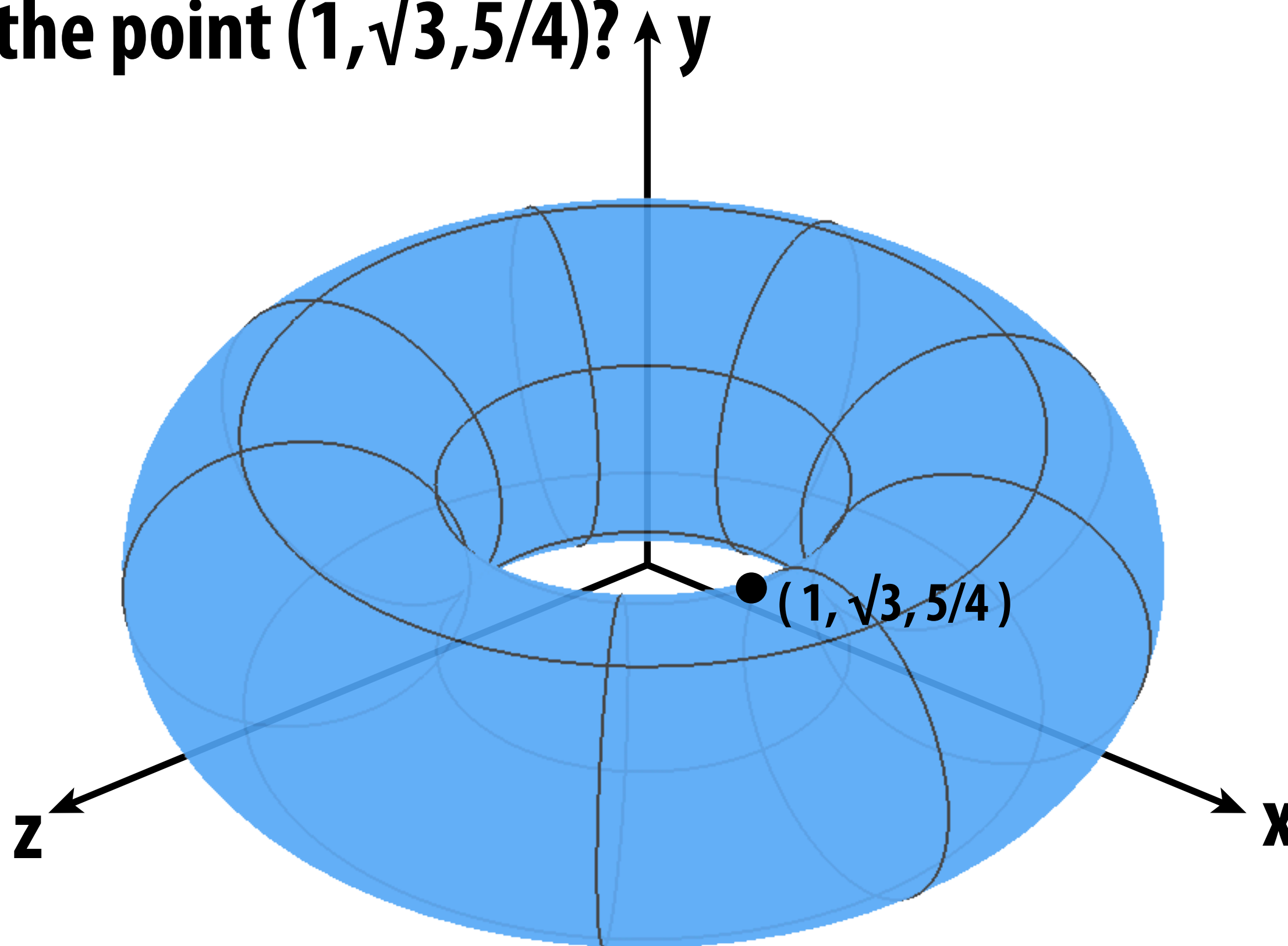
**I want to see if a point is inside it.**

# Check if this point is inside the torus

My surface is  $f(u,v) = (2 + \cos(u))\cos(v), 2 + \cos(u))\sin(v), \sin(u)$

How about the point  $(1, \sqrt{3}, 5/4)$ ?

**...NO!**



**Explicit surfaces make other tasks hard (like inside/outside tests).**

## **CONCLUSION:**

**Some representations work better than others—depends on the task!**



**Different representations will also be better suited to different types of geometry.**

**Let's take a look at some common representations used in computer graphics.**

# Algebraic Surfaces (Implicit)

- Surface is zero set of a polynomial in  $x, y, z$  (“algebraic variety”)

- Examples:



$$x^2 + y^2 + z^2 = 1$$



$$(R - \sqrt{x^2 + y^2})^2 + z^2 = r^2$$



$$(x^2 + \frac{9y^2}{4} + z^2 - 1)^3 = x^2 z^3 + \frac{9y^2 z^3}{80}$$

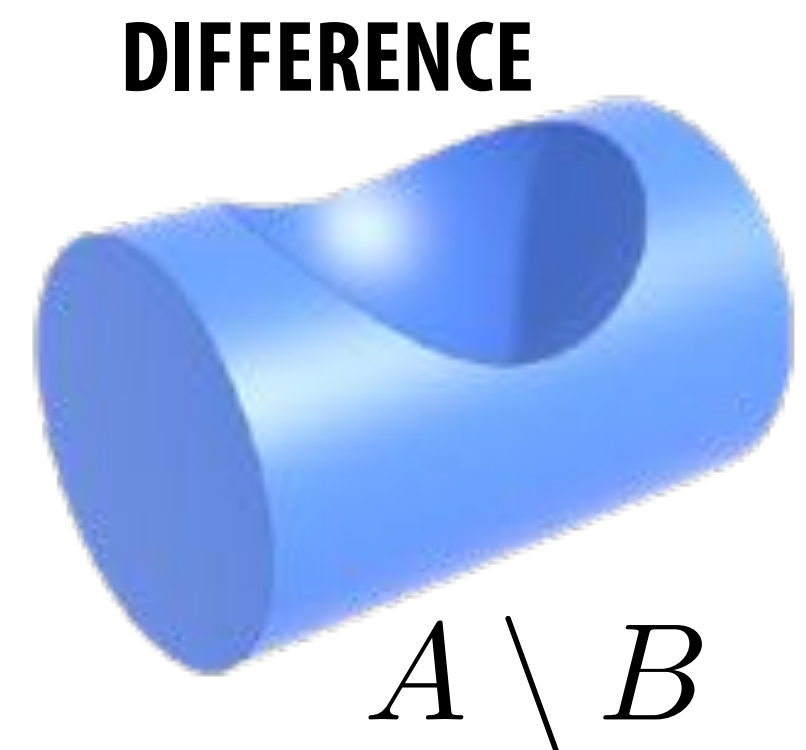
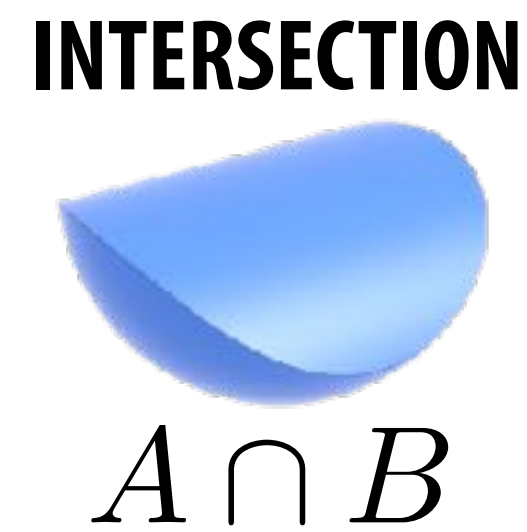
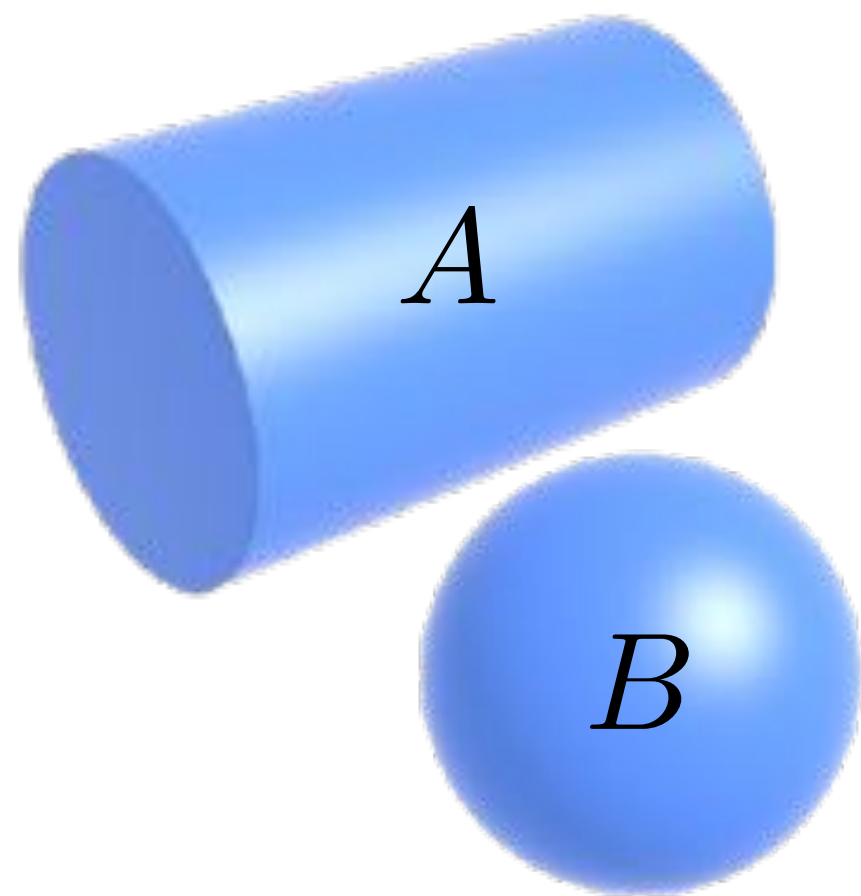
- What about more complicated shapes?



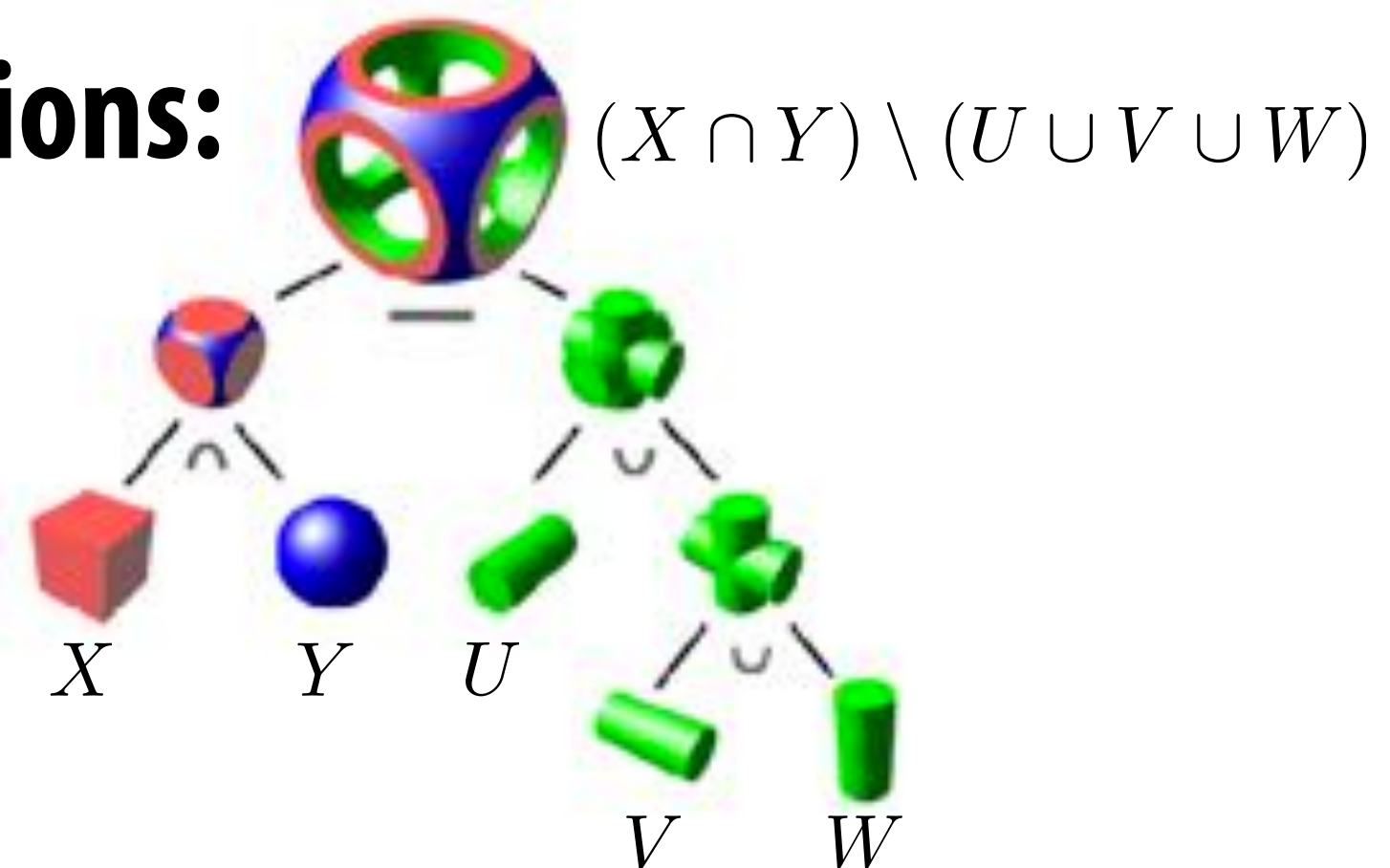
- Very hard to come up with polynomials!

# Constructive Solid Geometry (Implicit)

- Build more complicated shapes via Boolean operations
- Basic operations:

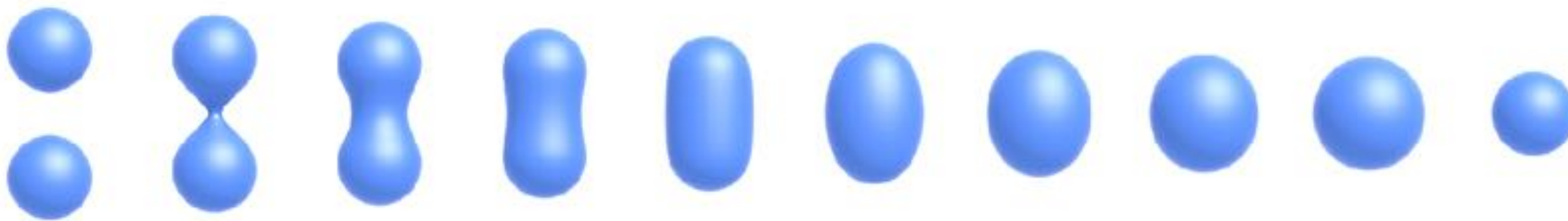


- Then chain together expressions:



# Bloppy Surfaces (Implicit)

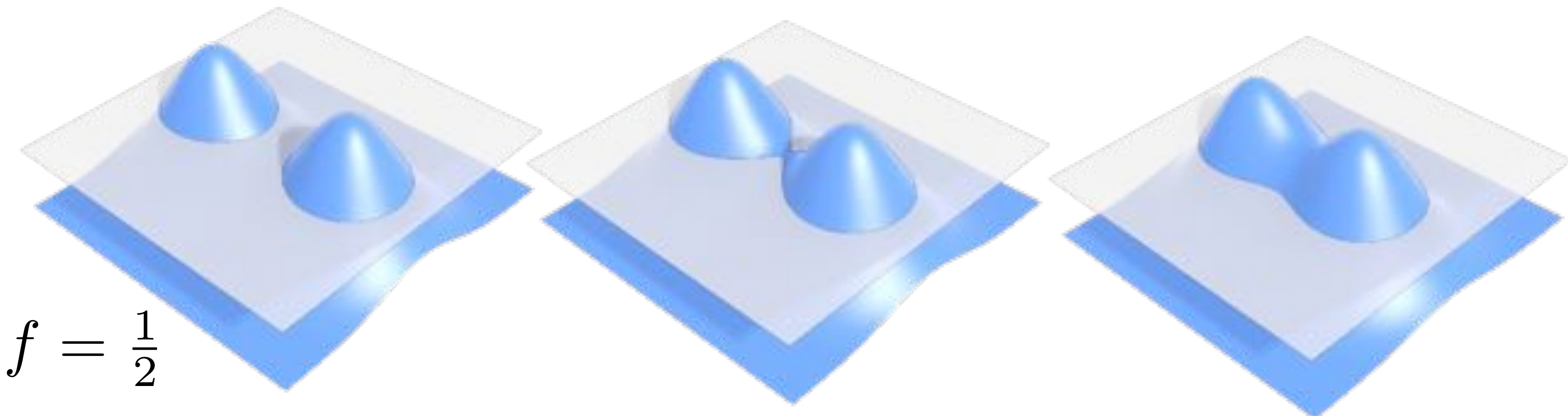
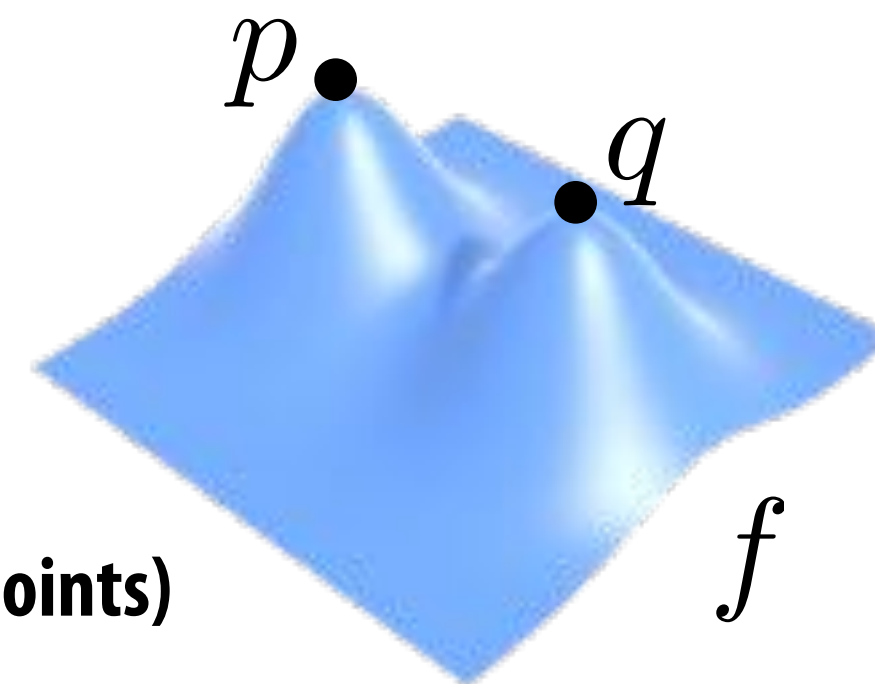
- Instead of Booleans, gradually blend surfaces together:



- Easier to understand in 2D:

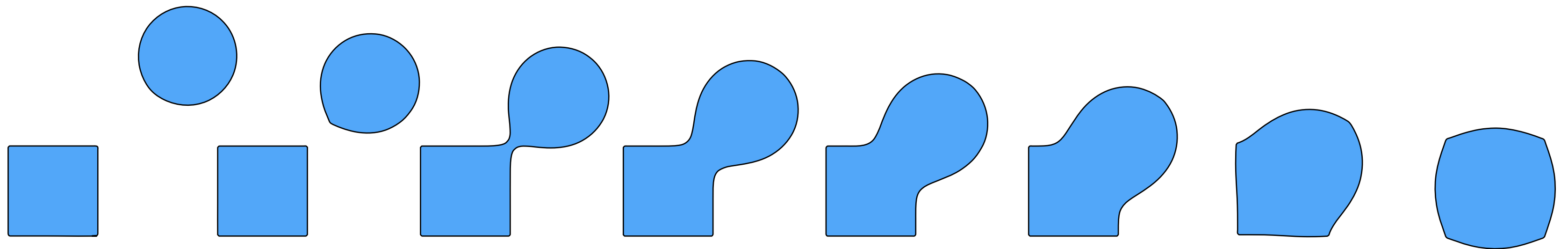
$$\phi_p(x) := e^{-|x-p|^2} \quad \text{(Gaussian centered at } p\text{)}$$

$$f := \phi_p + \phi_q \quad \text{(Sum of Gaussians centered at different points)}$$



# Blending Distance Functions (Implicit)

- A distance function gives distance to closest point on object
- Can blend any two distance functions  $d_1, d_2$ :



- Similar strategy to points, though many possibilities. E.g.,

$$f(x) := e^{d_1(x)^2} + e^{d_2(x)^2} - \frac{1}{2}$$

- Appearance depends on exactly how we combine functions
- Q: How do we implement a simple Boolean union?
- A: Just take the minimum:  $f(x) := \min(d_1(x)) + \min(d_2(x))$

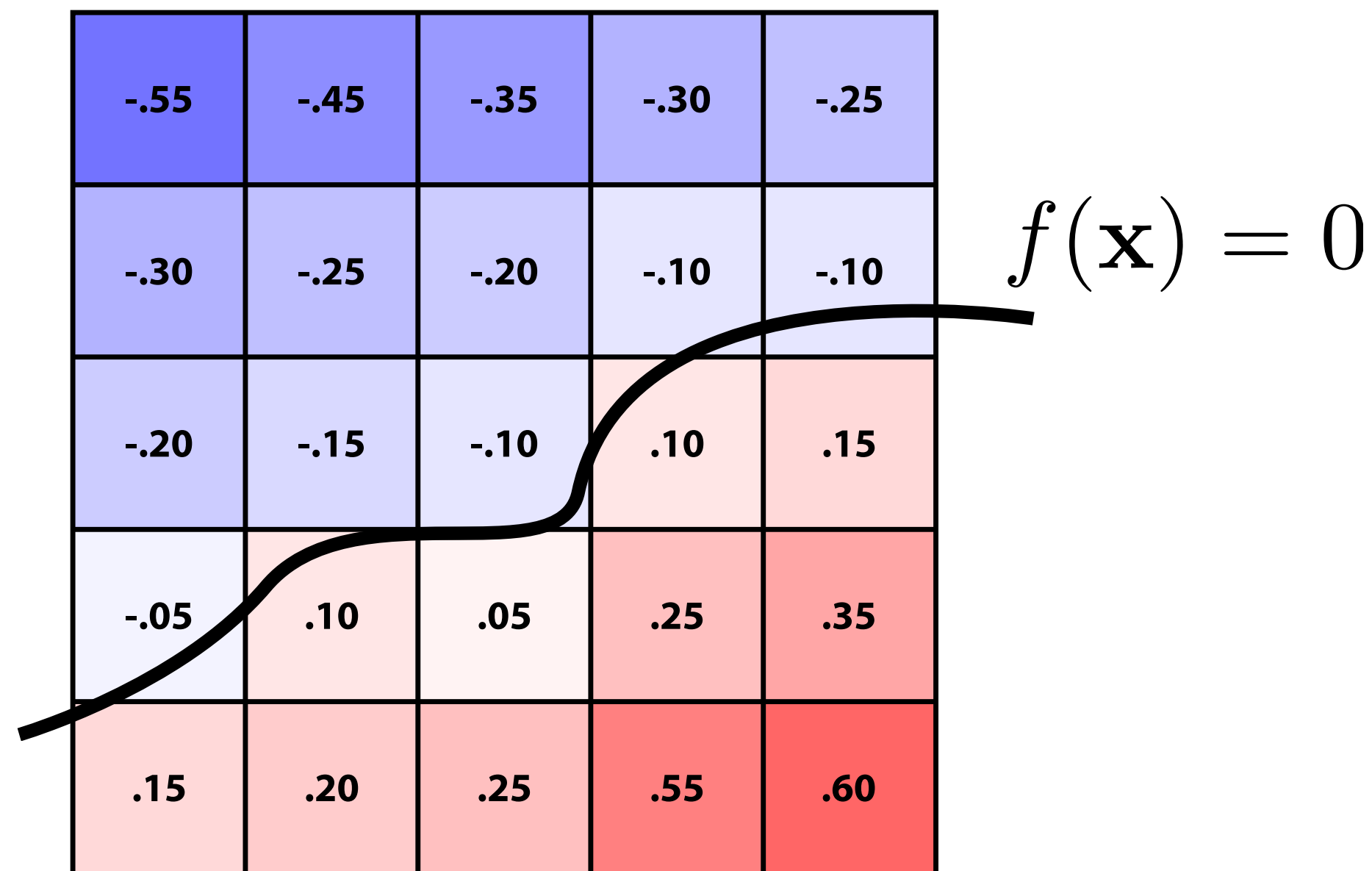
# Scene of pure distance functions (not easy!)



See <http://iquilezles.org/www/material/nvscene2008/nvscene2008.htm>

# Level Set Methods (Implicit)

- Implicit surfaces have some nice features (e.g., merging/splitting)
- But, hard to describe complex shapes in closed form
- Alternative: store a grid of values approximating function



- Surface is found where interpolated values equal zero
- Provides much more explicit control over shape (like a texture)
- Often demands sophisticated filtering (trilinear, tricubic...)

# Level Sets from Medical Data (CT, MRI, etc.)

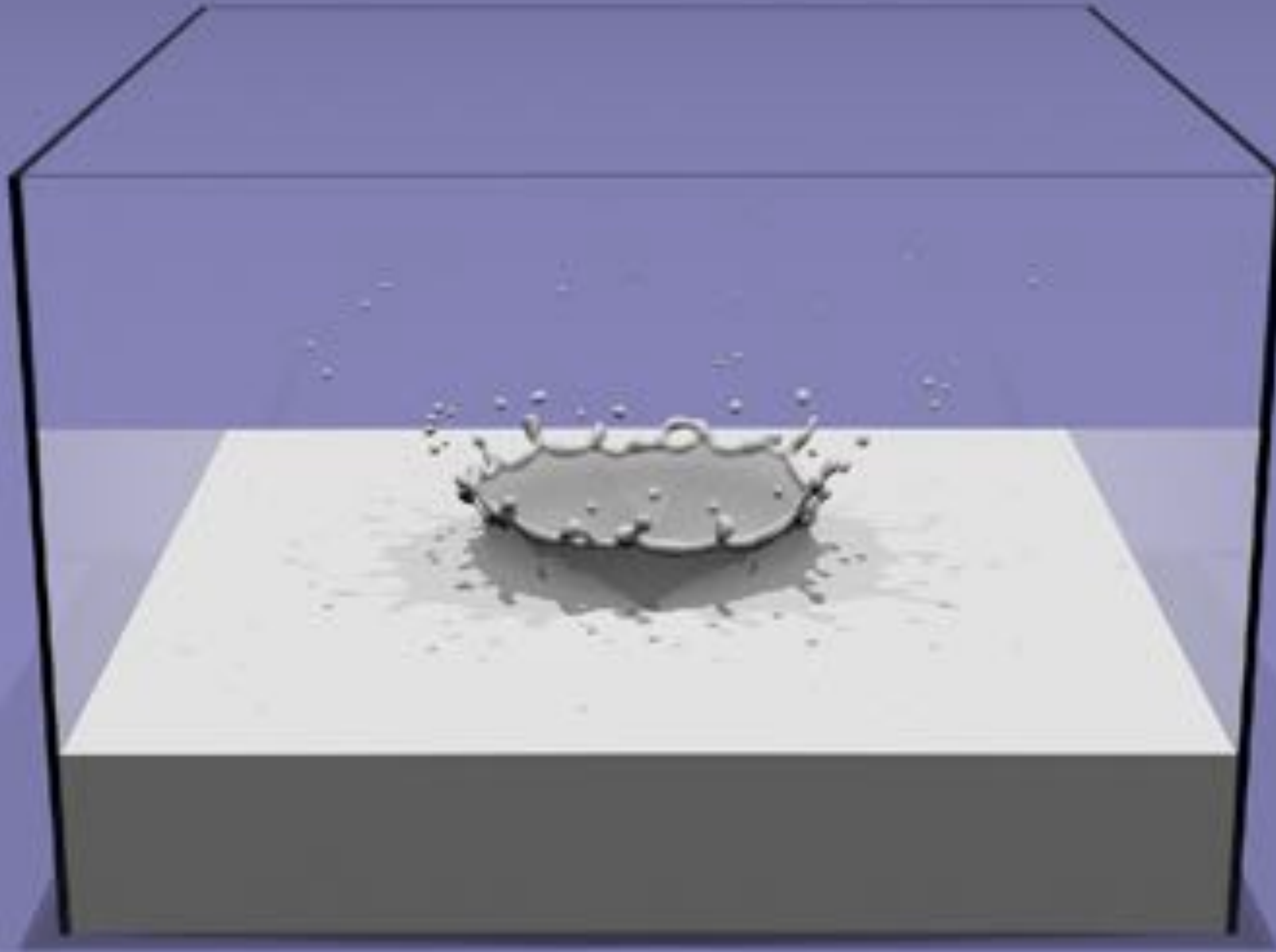
- Level sets encode, e.g., constant tissue density





# Level Sets in Physical Simulation

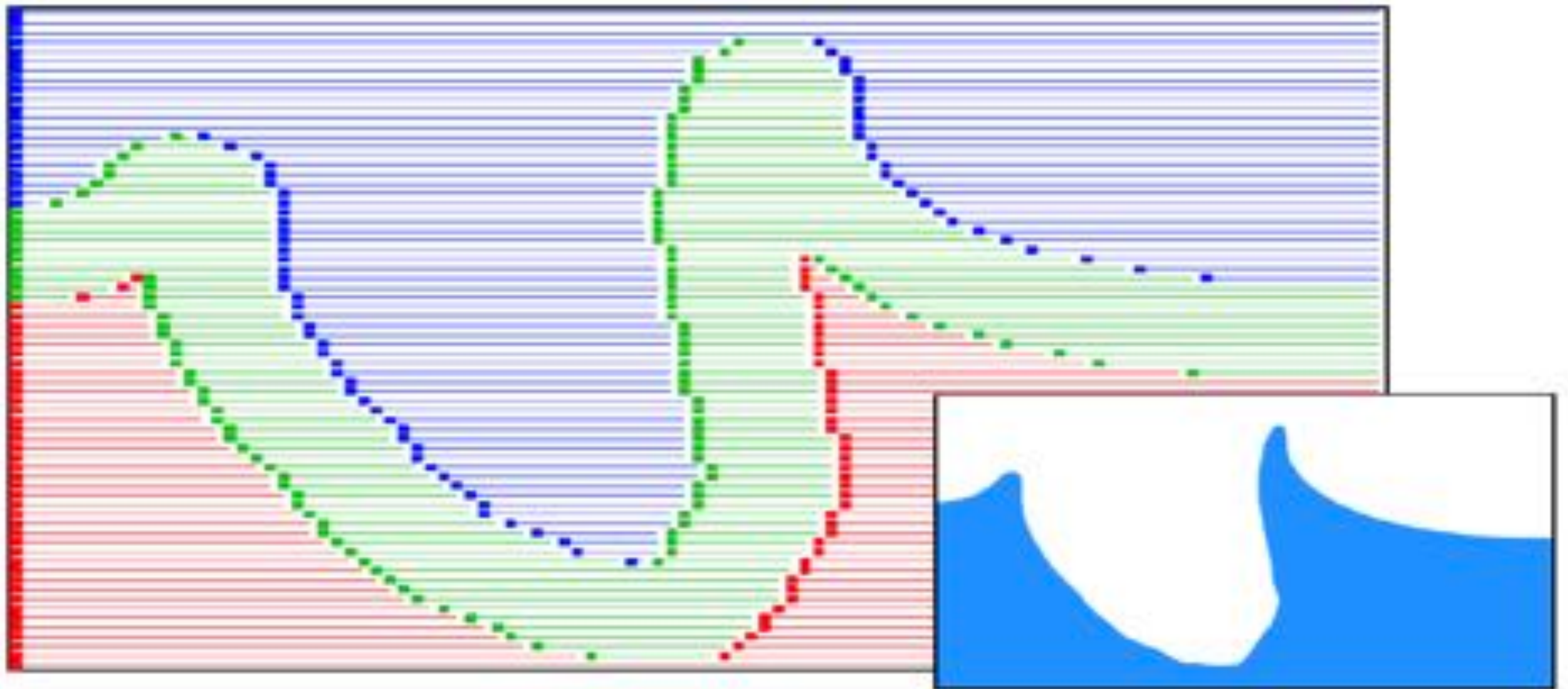
- Level set encodes distance to air-liquid boundary



See <http://physbam.stanford.edu>

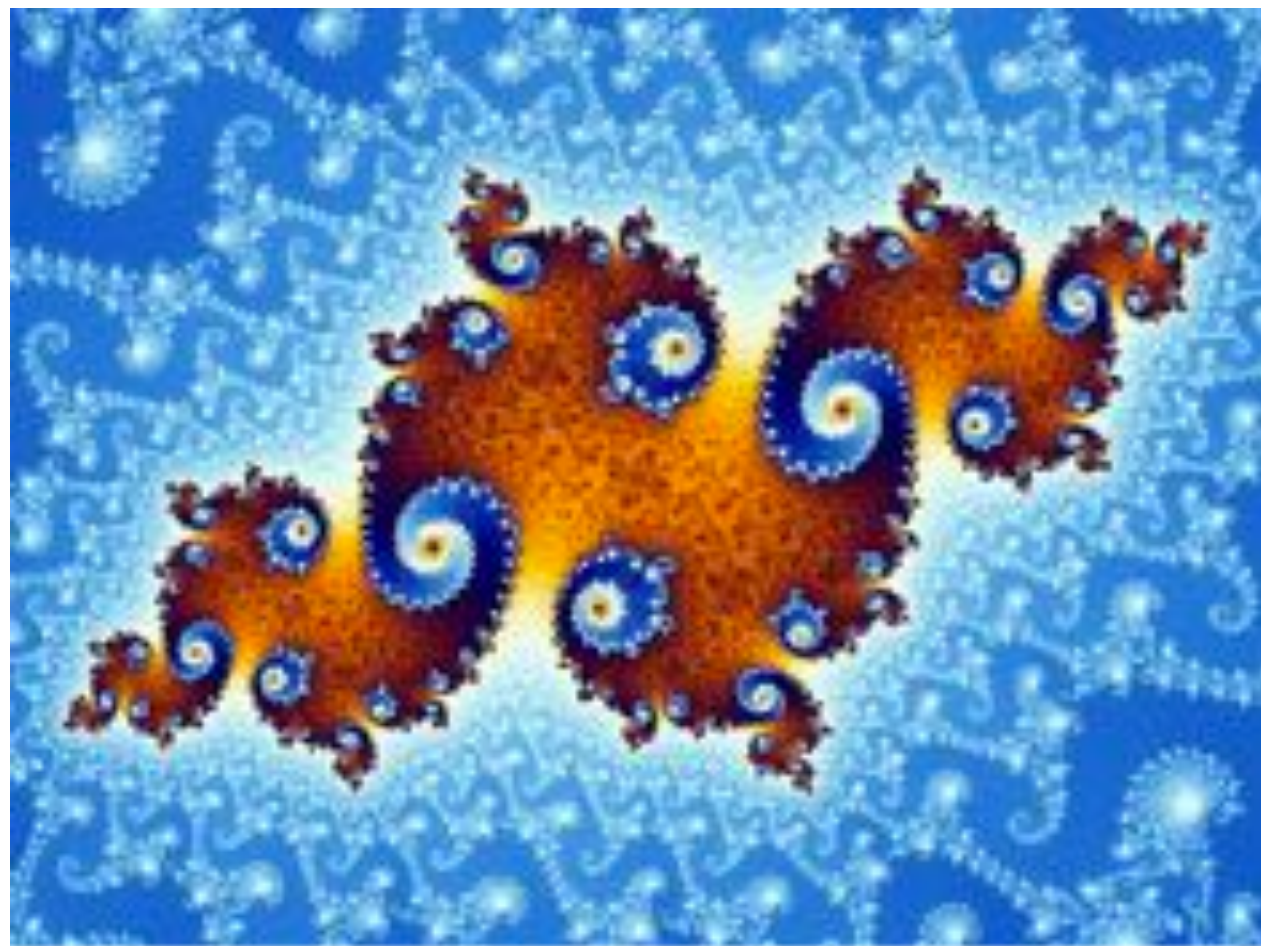
# Level Set Storage

- **Drawback: storage for 2D surface is now  $O(n^3)$**
- **Can reduce cost by storing only a narrow band around surface:**



# Fractals (Implicit)

- No precise definition; exhibit self-similarity, detail at all scales
- New “language” for describing natural phenomena
- Hard to control shape!



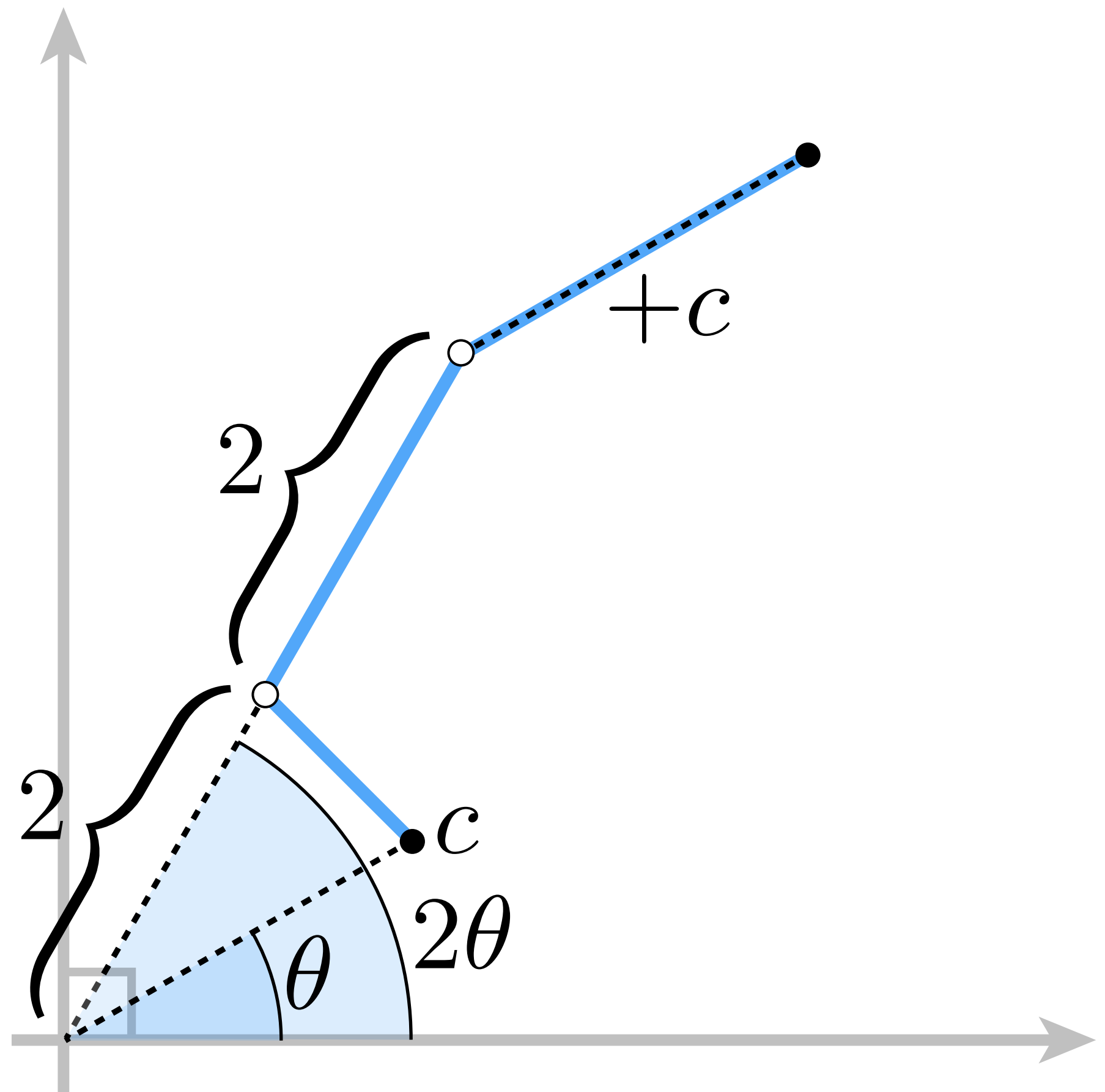
# Mandelbrot Set - Definition

## ■ For each point $c$ in the plane:

- double the angle
- square the magnitude
- add the original point  $c$
- repeat

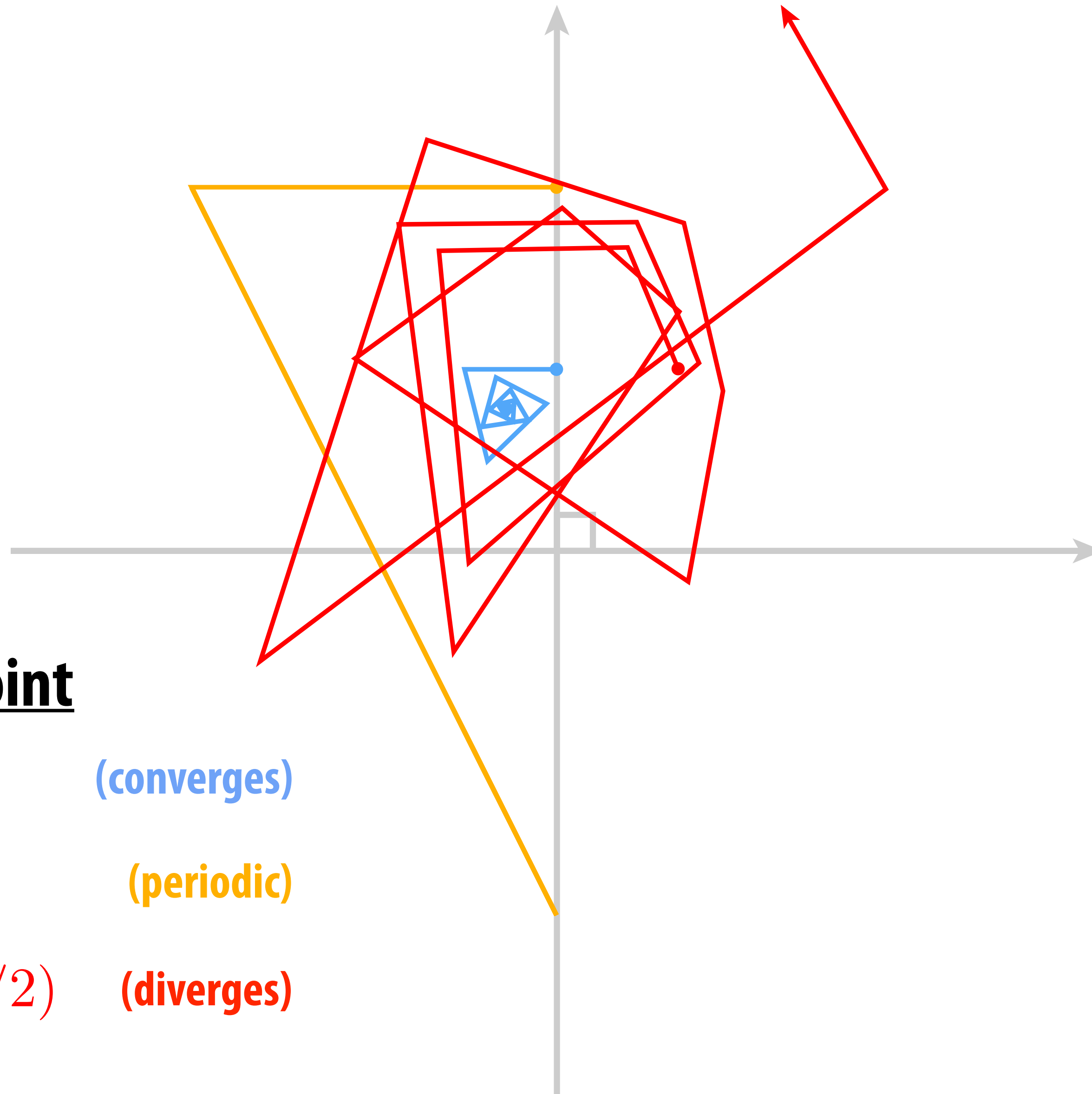
## ■ Complex version:

- Replace  $z$  with  $z^2+c$
- repeat



**If the point remains bounded (never goes to  $\infty$ ), it's in the set.**

# Mandelbrot Set - Examples



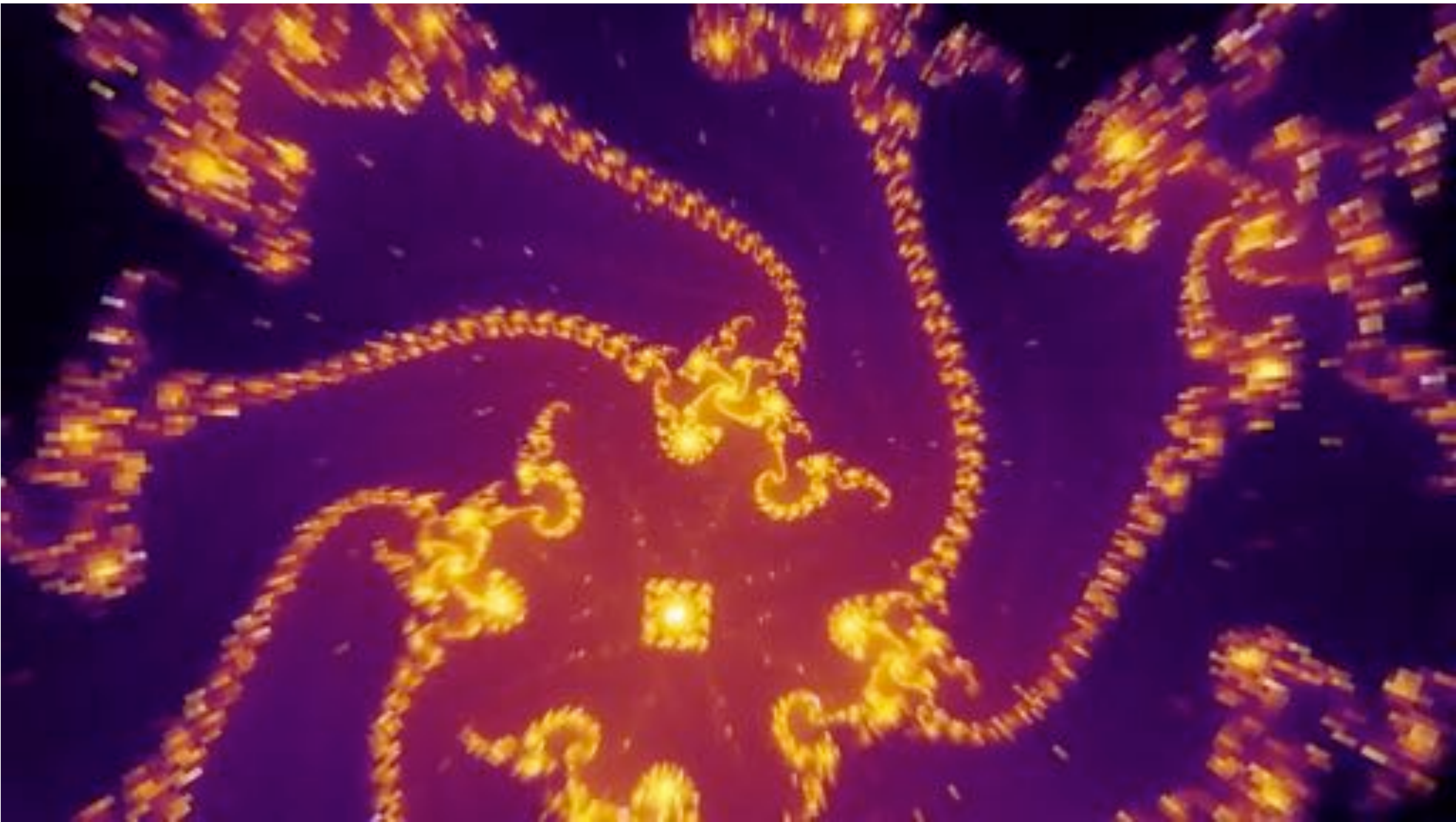
## starting point

■  $(0, 1/2)$  (converges)

■  $(0, 1)$  (periodic)

■  $(1/3, 1/2)$  (diverges)

# Mandelbrot Set - Zooming In



**(Colored according to how quickly each point diverges/converges.)**

# Iterated Function Systems



**Scott Draves (CMU Alumnus) - see <http://electricssheep.org>**

# Implicit Representations - Pros & Cons

## ■ Pros:

- **description can be very compact (e.g., a polynomial)**
- **easy to determine if a point is in our shape (just plug it in!)**
- **other queries may also be easy (e.g., distance to surface)**
- **for simple shapes, exact description/no sampling error**
- **easy to handle changes in topology (e.g., fluid)**

## ■ Cons:

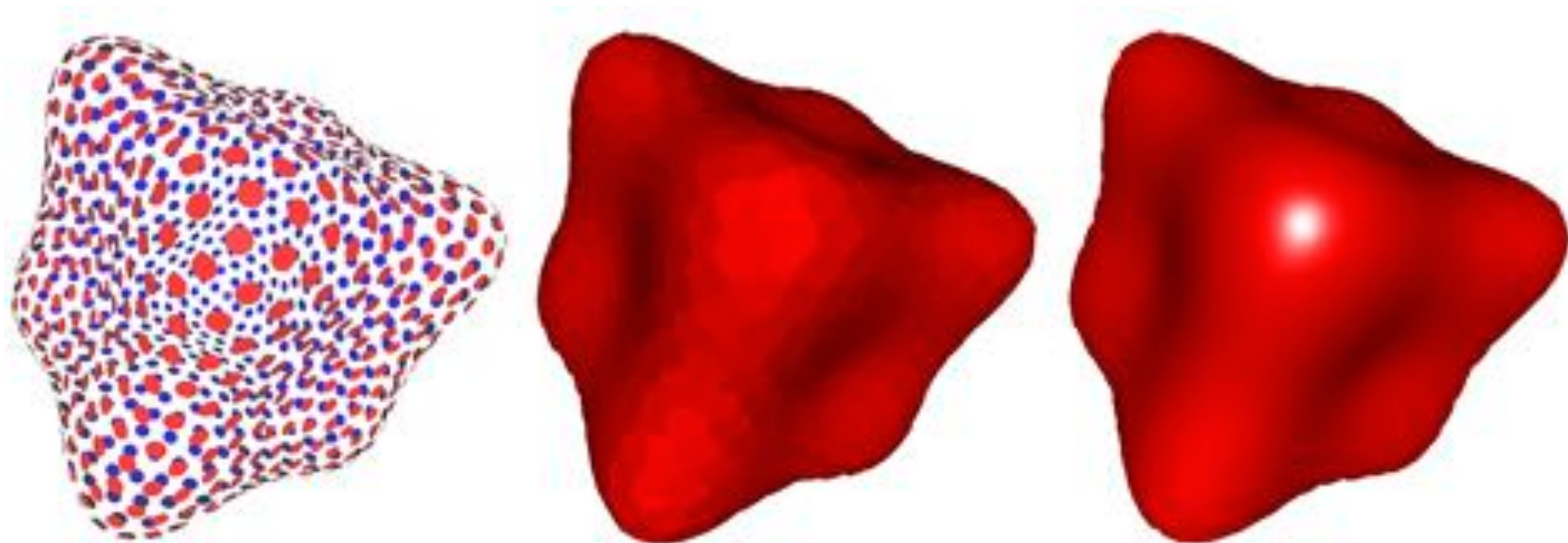
- **expensive to find all points in the shape (e.g., for drawing)**
- **very difficult to model complex shapes**



**What about explicit representations?**

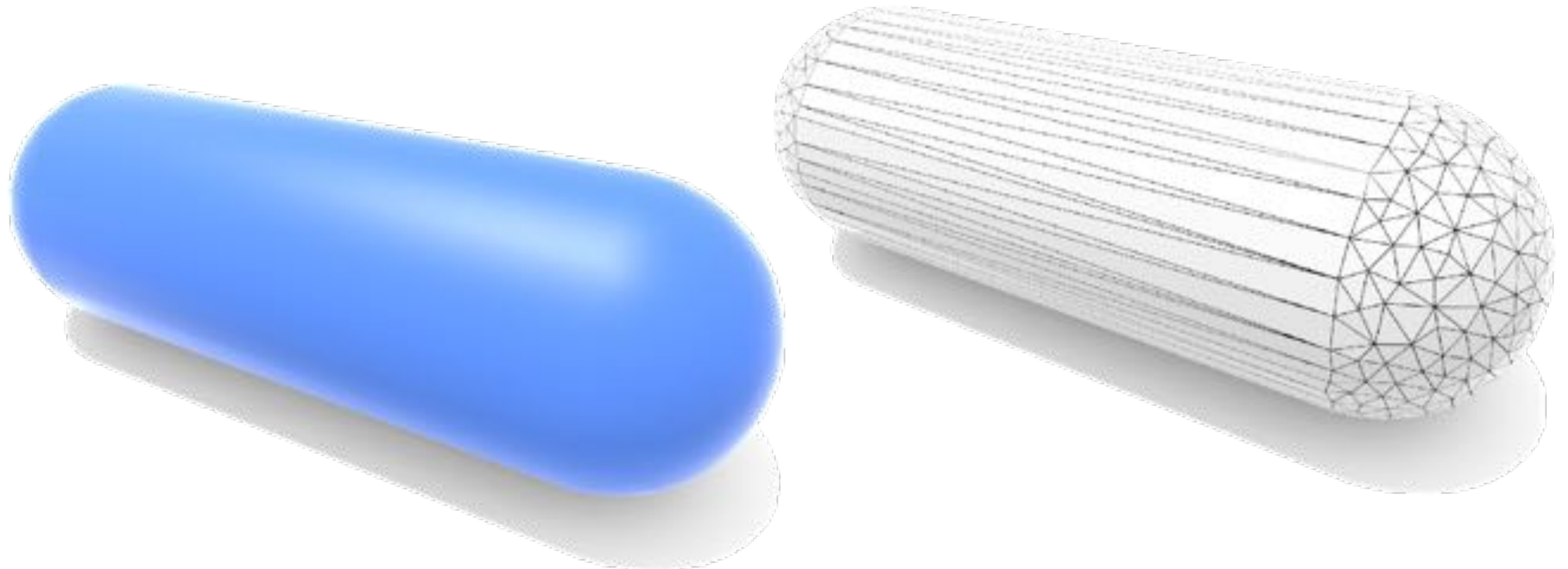
# Point Cloud (Explicit)

- Easiest representation: list of points  $(x,y,z)$
- Often augmented with normals
- Easily represent any kind of geometry
- Useful for LARGE datasets ( $\gg 1$  point/pixel)
- Hard to interpolate undersampled regions
- Hard to do processing / simulation / ...



# Polygon Mesh (Explicit)

- **Store vertices and polygons (most often triangles or quads)**
- **Easier to do processing/simulation, adaptive sampling**
- **More complicated data structures**
- **Perhaps most common representation in graphics**



**(Much more about polygon meshes in upcoming lectures!)**

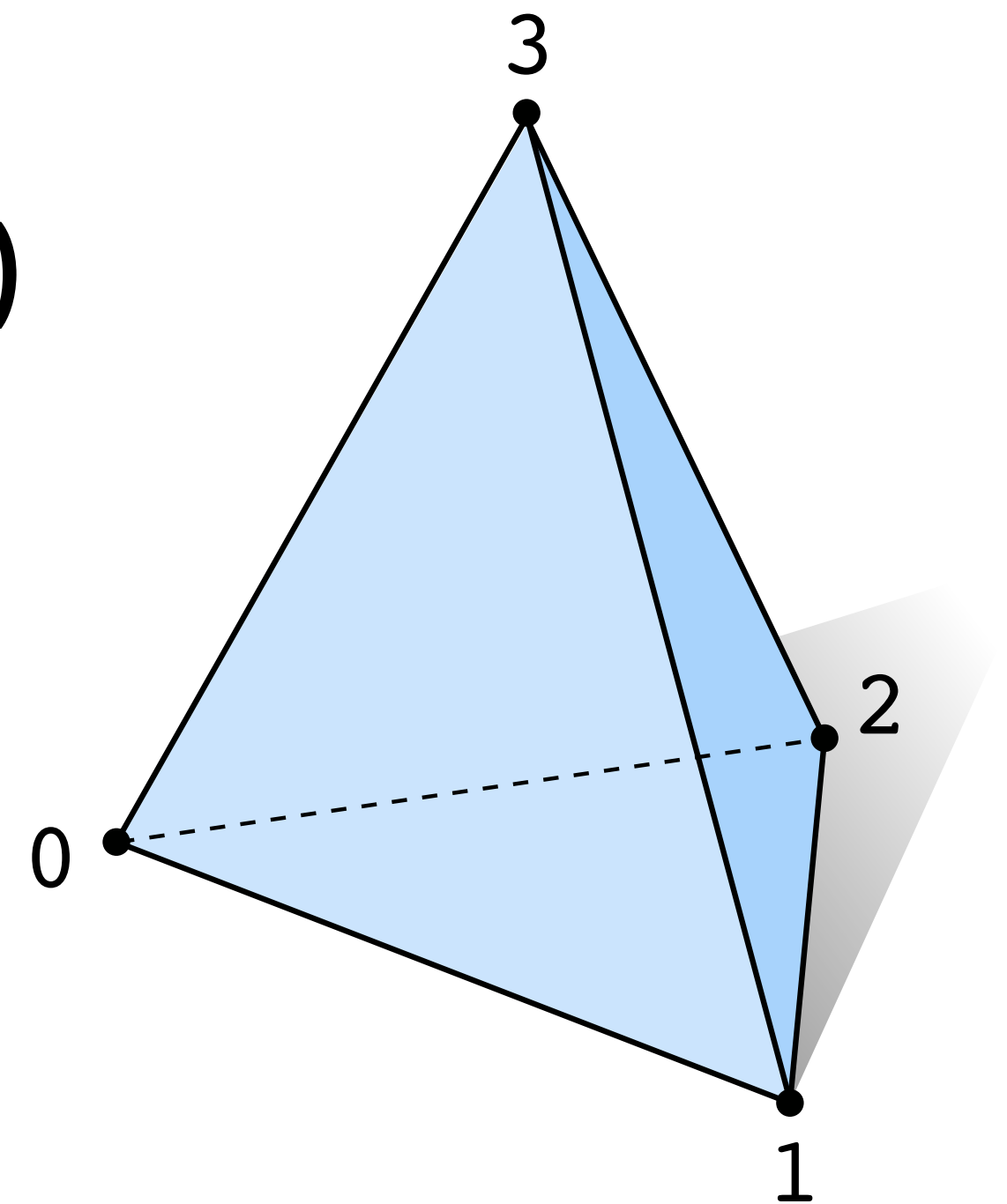
# Triangle Mesh (Explicit)

- Store vertices as triples of coordinates  $(x,y,z)$

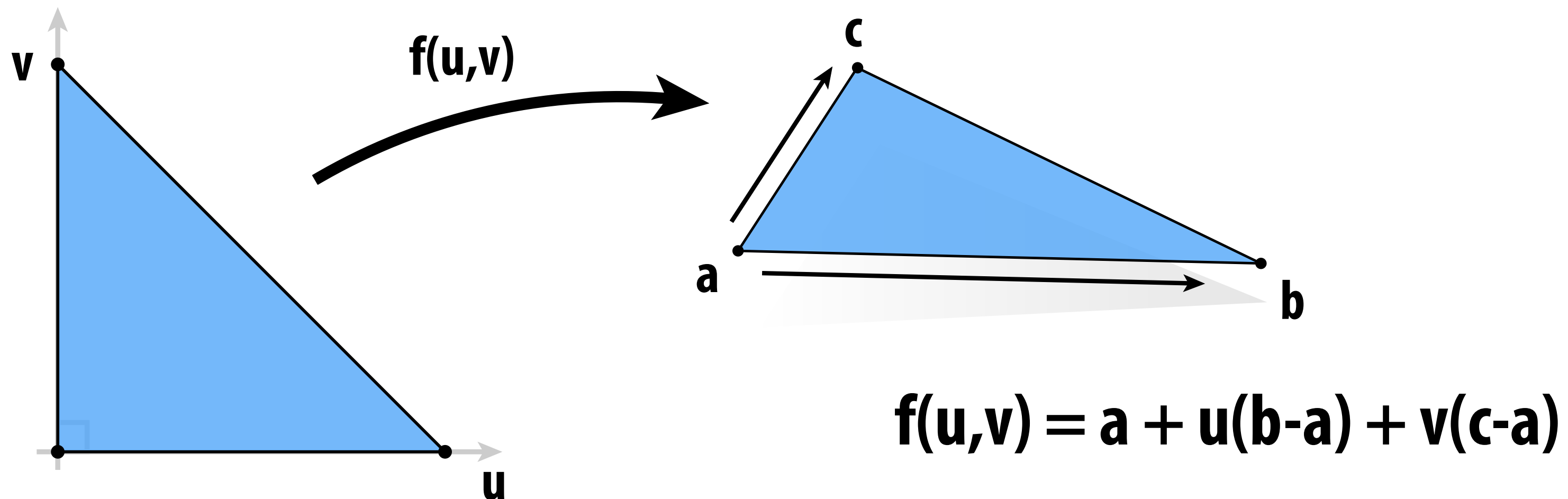
- Store triangles as triples of indices  $(i,j,k)$

- E.g., tetrahedron:

	VERTICES			TRIANGLES		
	x	y	z	i	j	k
0:	-1	-1	-1	0	2	1
1:	1	-1	1	0	3	2
2:	1	1	-1	3	0	1
3:	-1	1	1	3	1	2

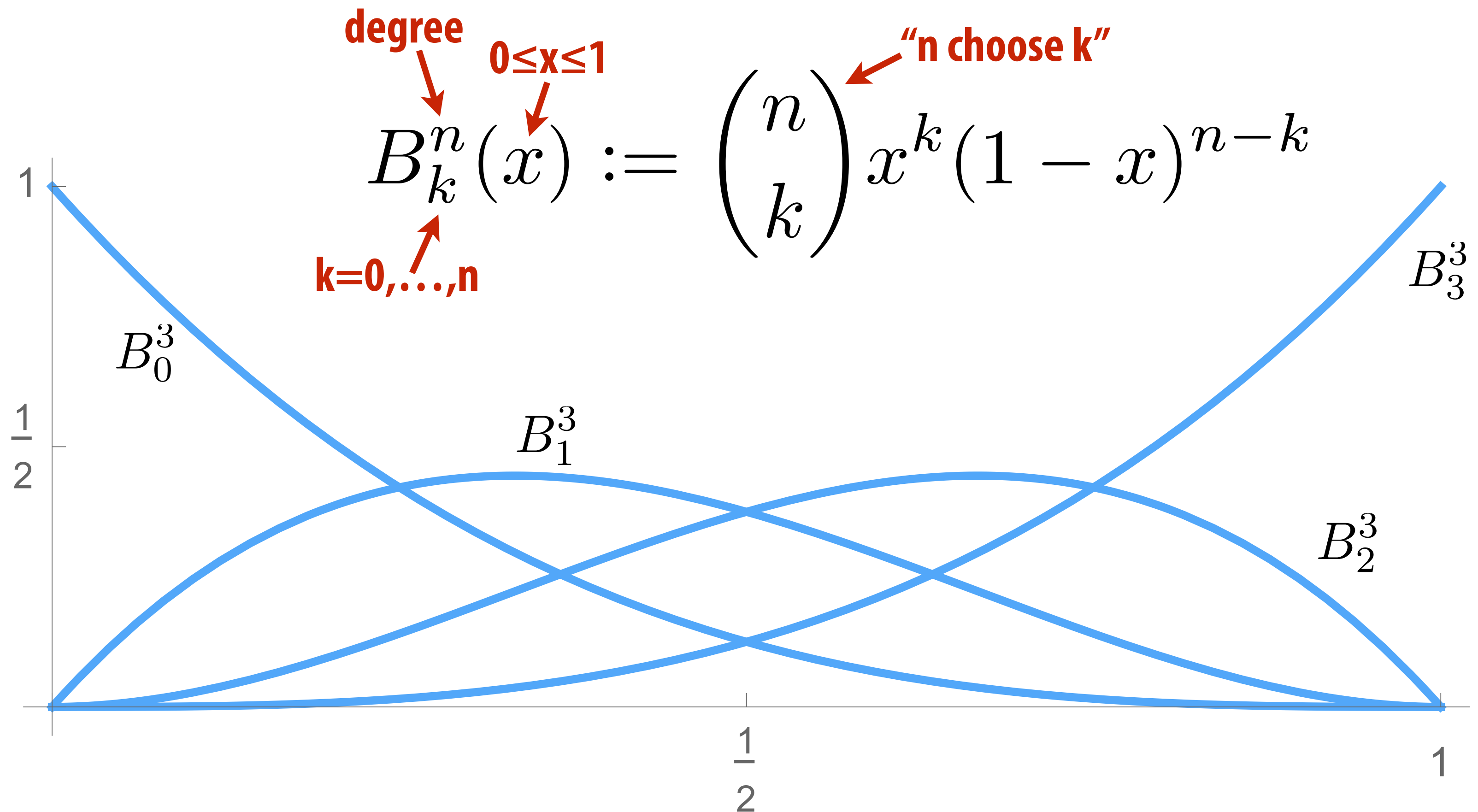


- Use linear interpolation to define points inside triangles:



# Bernstein Basis

- Why limit ourselves to just linear interpolation?
- More flexibility by using higher-order polynomials
- Instead of usual basis  $(1, x, x^2, x^3, \dots)$ , use Bernstein basis:

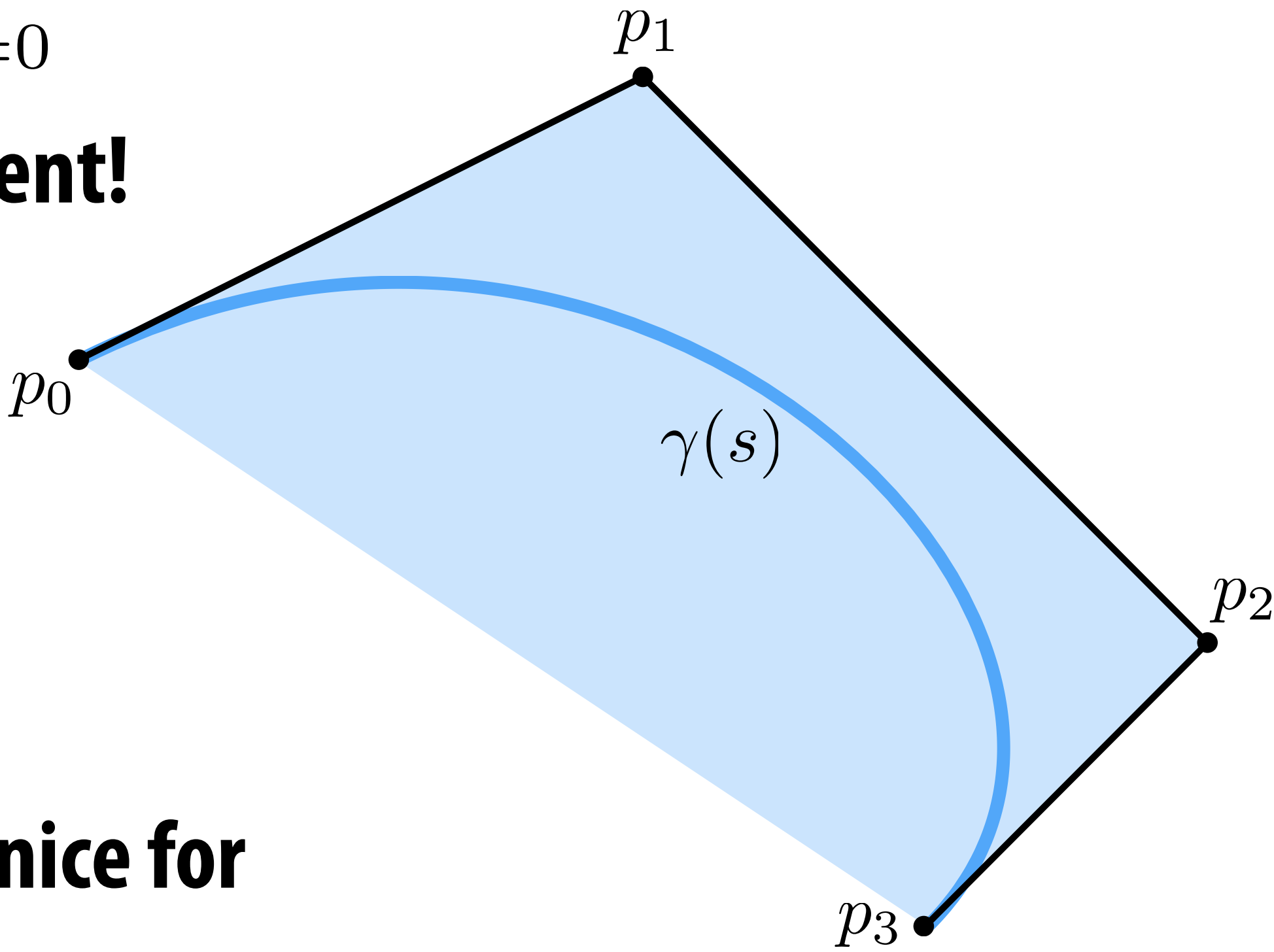


# Bézier Curves (Explicit)

- A Bézier curve is a curve expressed in the Bernstein basis:

$$\gamma(s) := \sum_{k=0}^n B_{n,k}(s) p_k$$

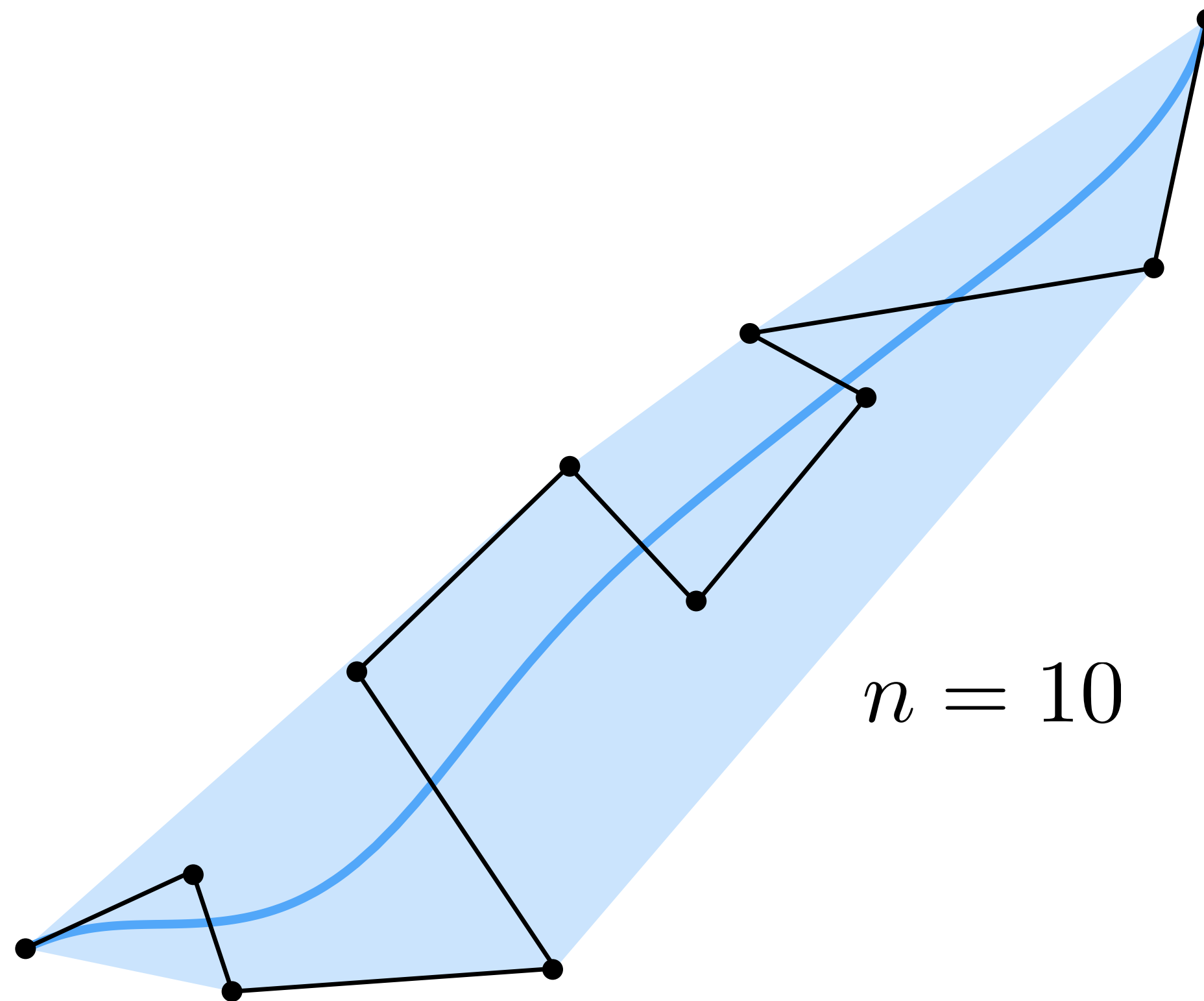
control points



- For  $n=1$ , just get a line segment!
- For  $n=3$ , get “cubic Bézier”:
- Important features:
  1. interpolates endpoints
  2. tangent to end segments
  3. contained in convex hull (nice for rasterization)

# Higher-order Bézier Curves?

- What if we want a more interesting curve?
- High-degree Bernstein polynomials don't interpolate well:



**Very hard to control!**

# Piecewise Bézier Curves (Explicit)

- More interesting shapes: piece together many Bézier curves
- Widely-used technique (Illustrator, fonts, SVG, etc.)



- Formally, piecewise Bézier curve:

piecewise Bézier

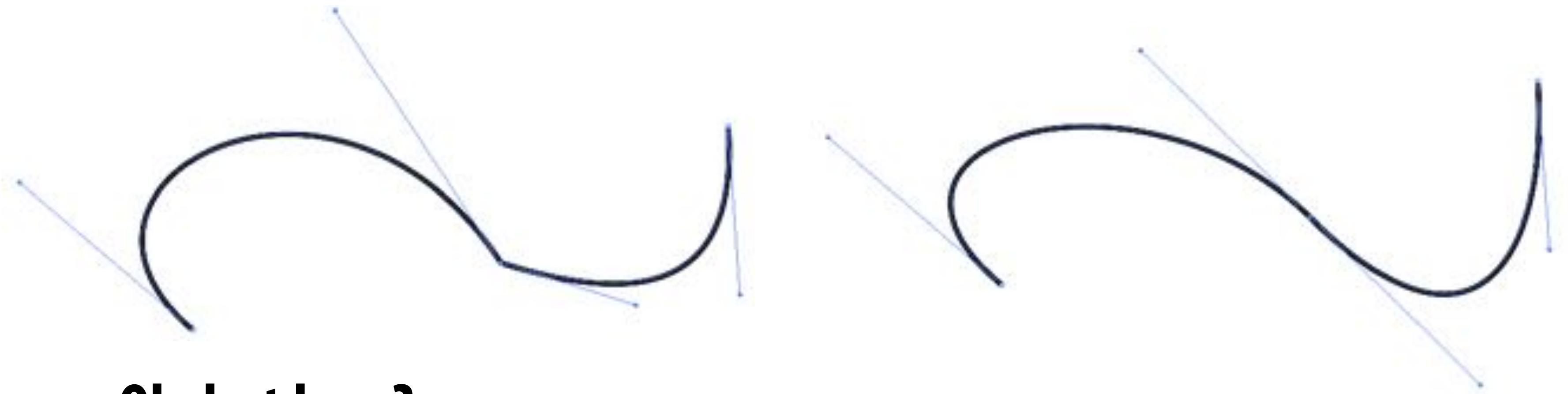
$$\gamma(u) := \gamma_i \left( \frac{u - u_i}{u_{i+1} - u_i} \right), \quad u_i \leq u < u_{i+1}$$

single Bézier



# Bézier Curves — tangent continuity

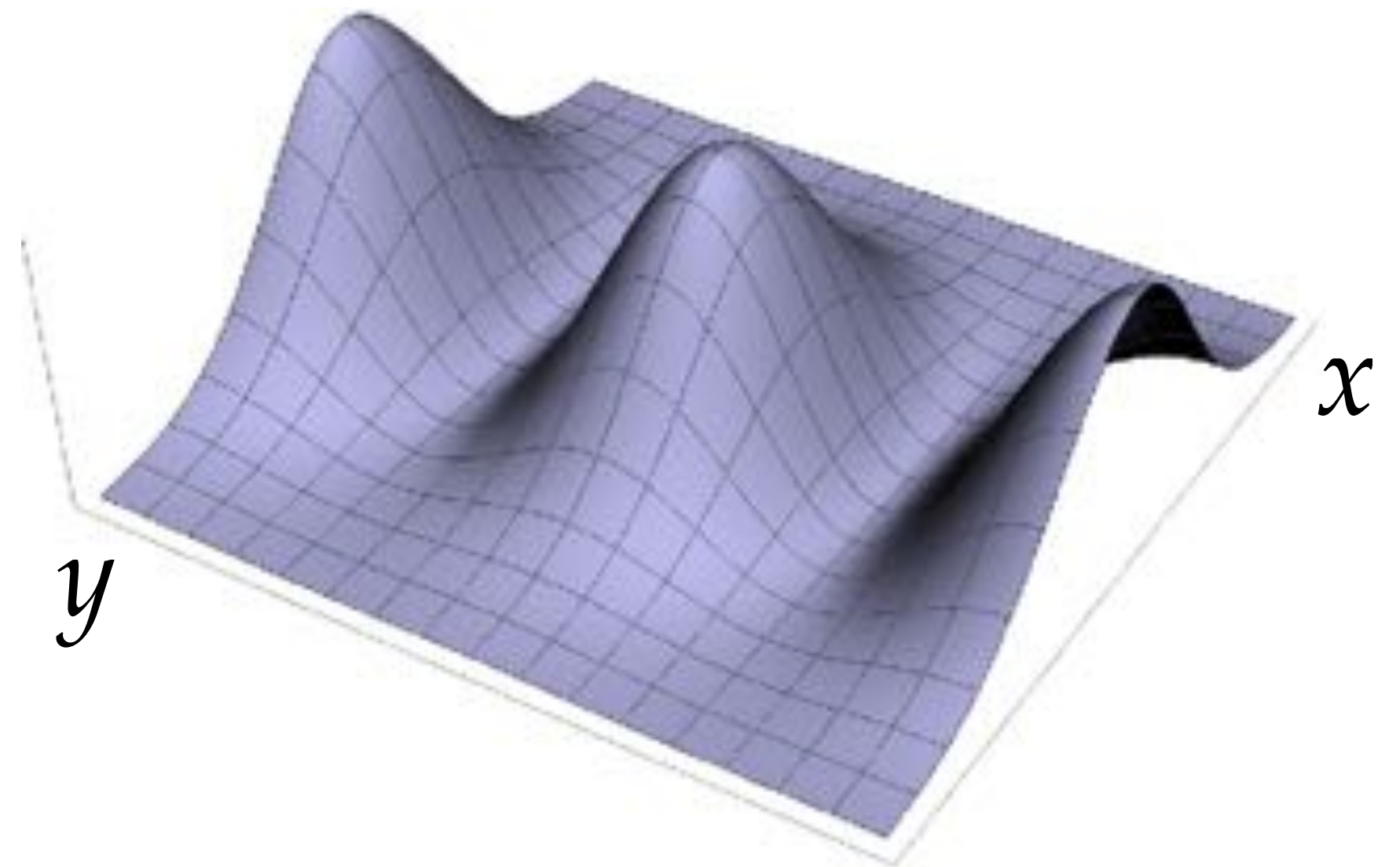
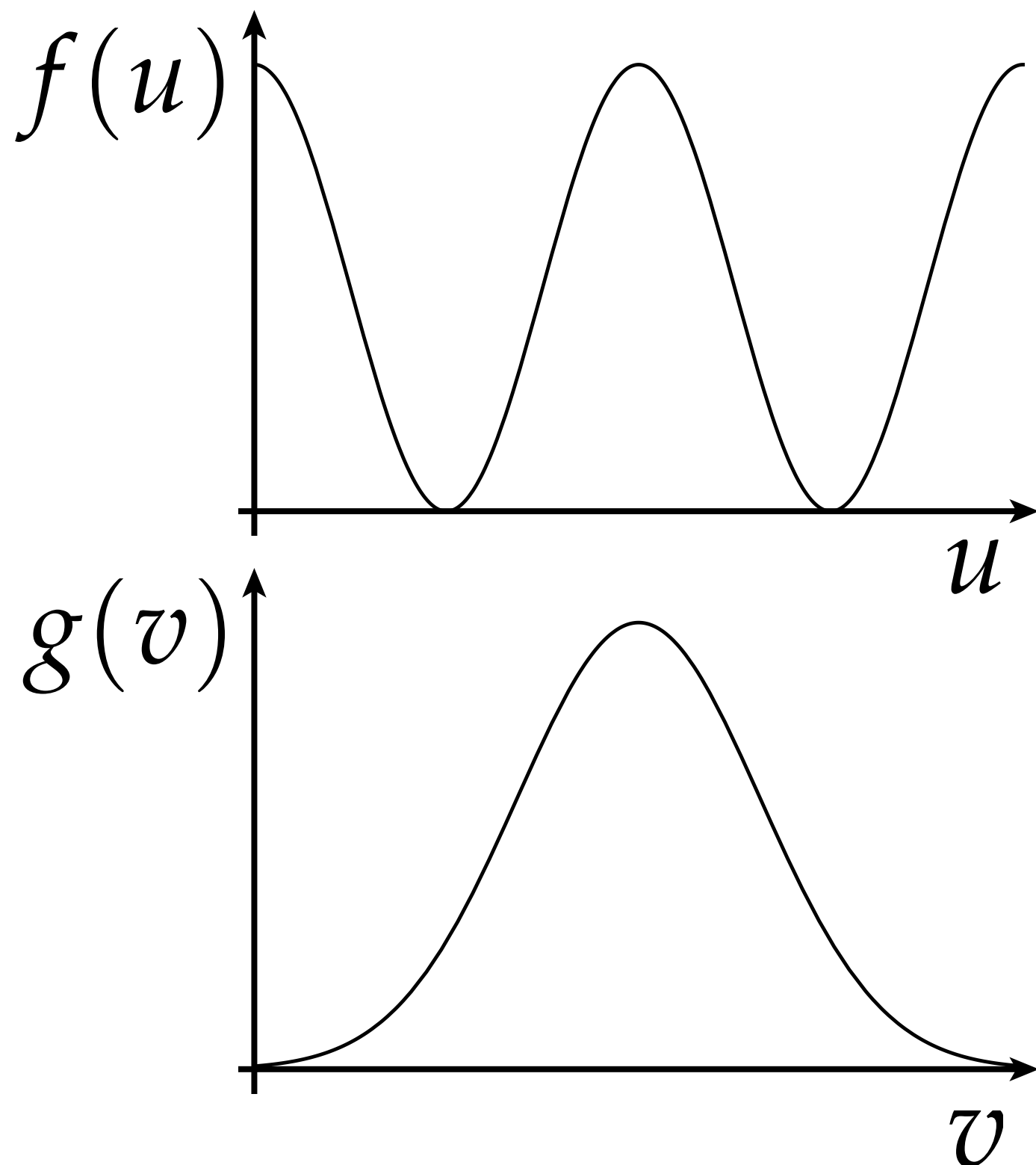
- To get “seamless” curves, want tangents to line up:



- Ok, but how?
- Each curve is cubic:  $u^3p_0 + 3u^2(1-u)p_1 + 3u(1-u)^2p_2 + (1-u)^3p_3$
- Want endpoints of each segment to meet
- Want tangents at endpoints to meet
- Q: How many constraints vs. degrees of freedom?
- Q: Could you do this with quadratic Bézier? Linear Bézier?

# Tensor Product

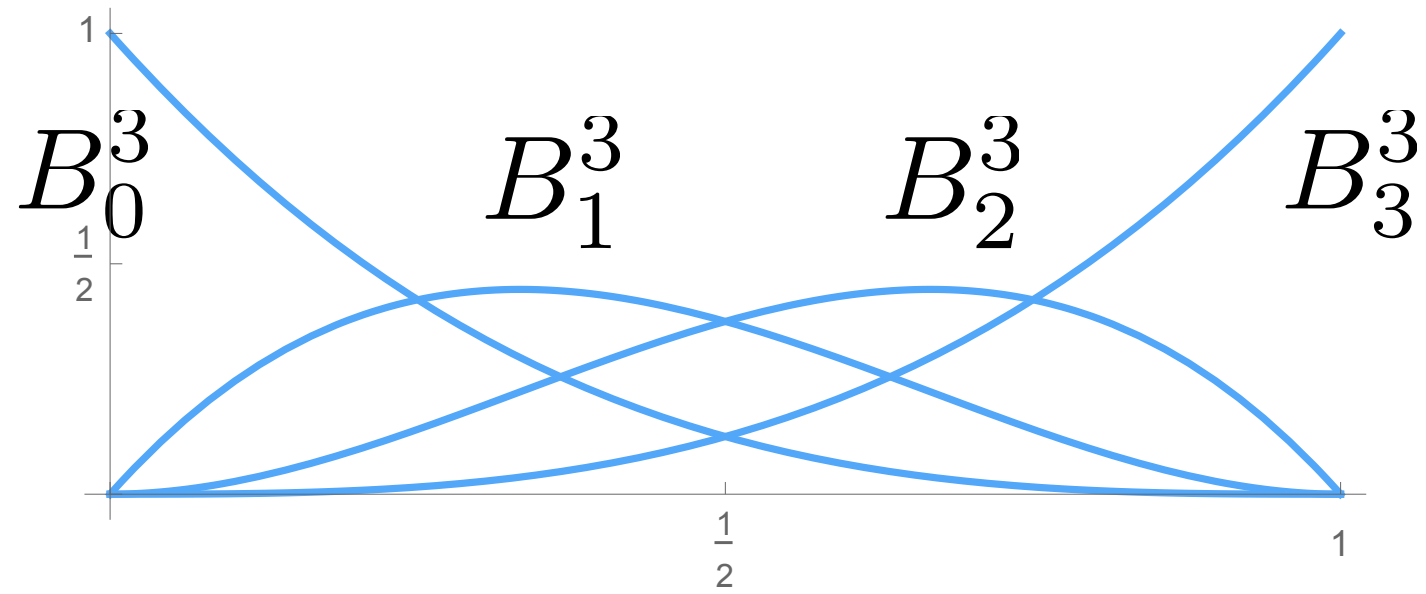
- Can use a pair of curves to get a surface
- Value at any point  $(u,v)$  given by product of a curve  $f$  at  $u$  and a curve  $g$  at  $v$  (sometimes called the “tensor product”):



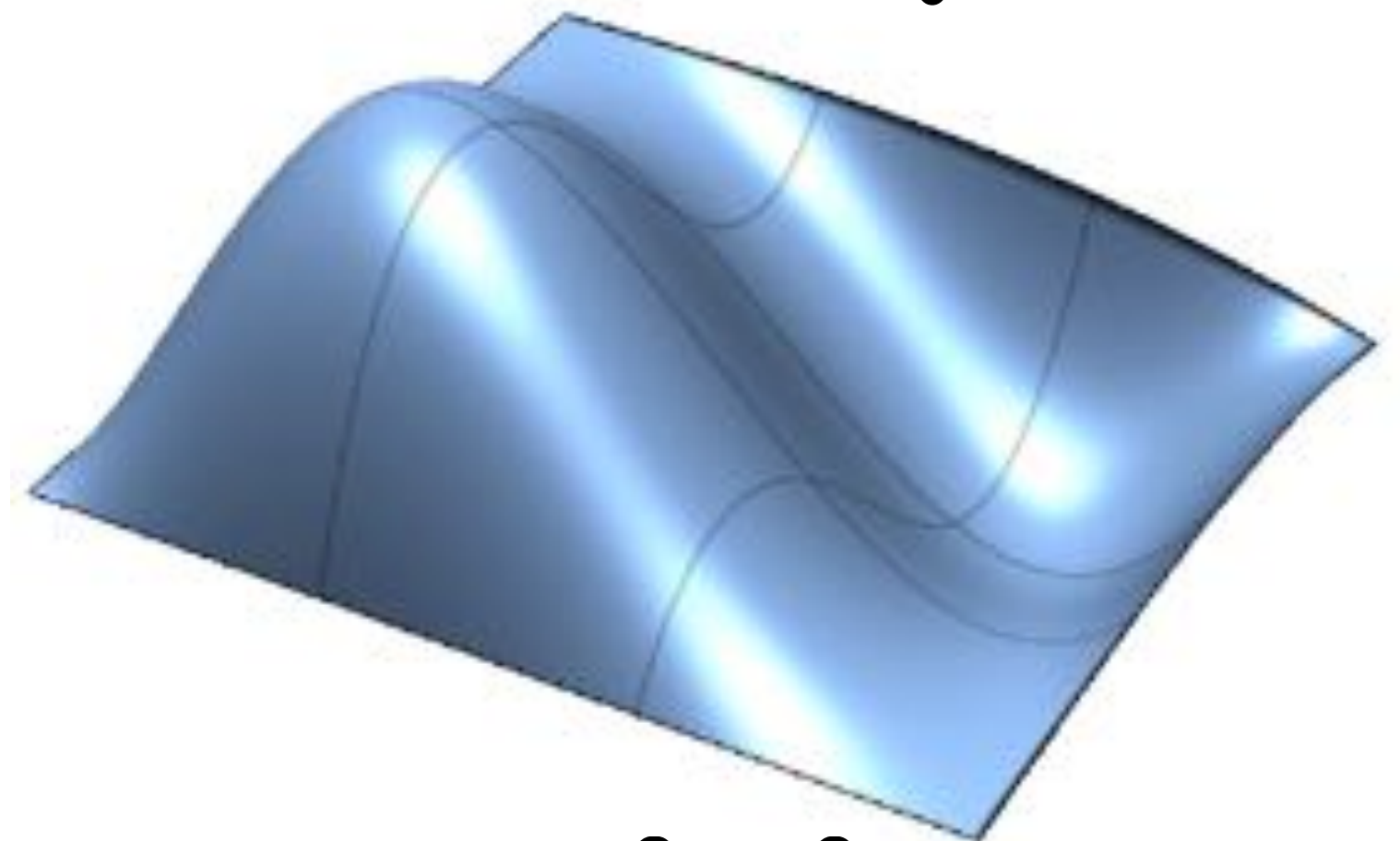
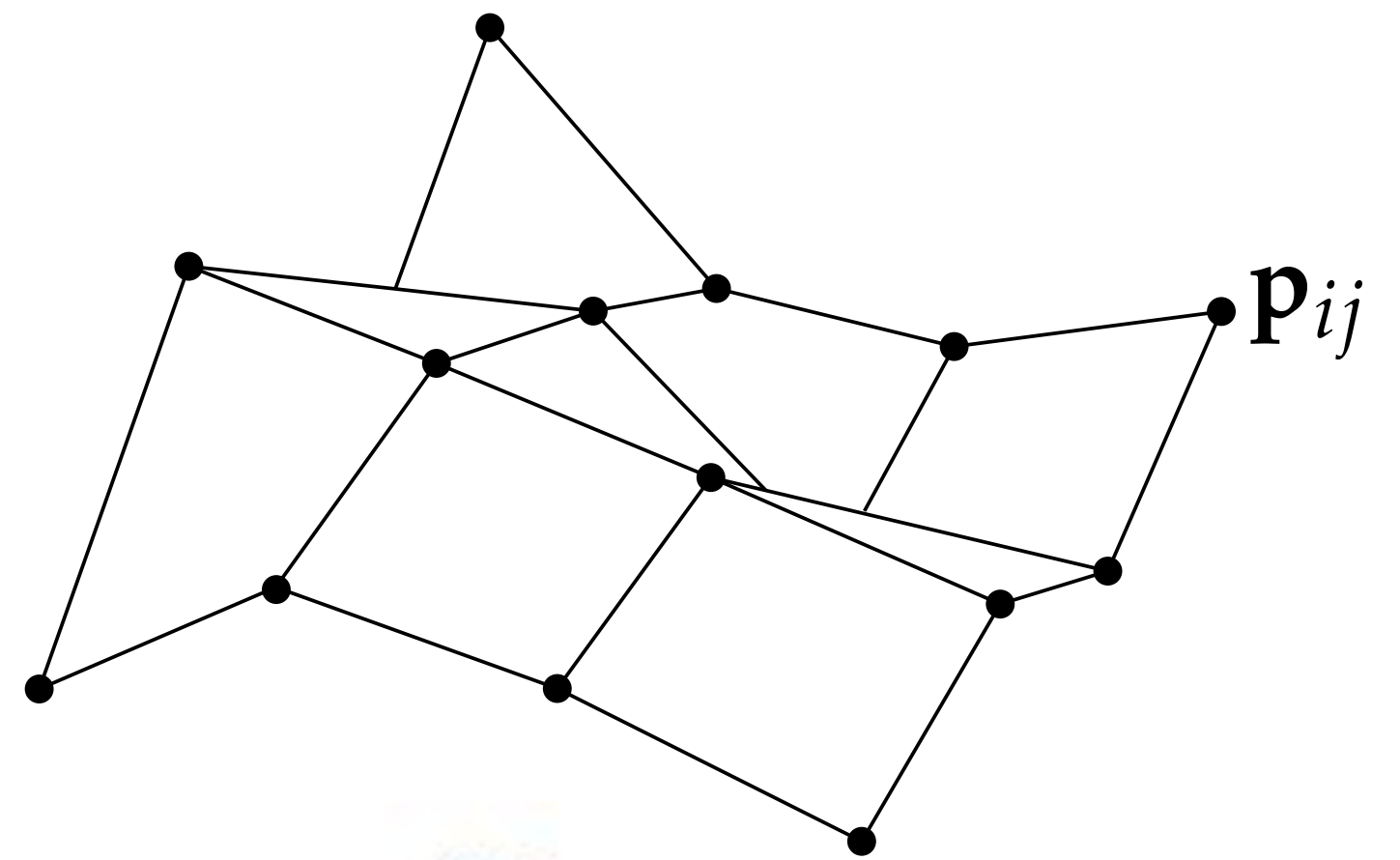
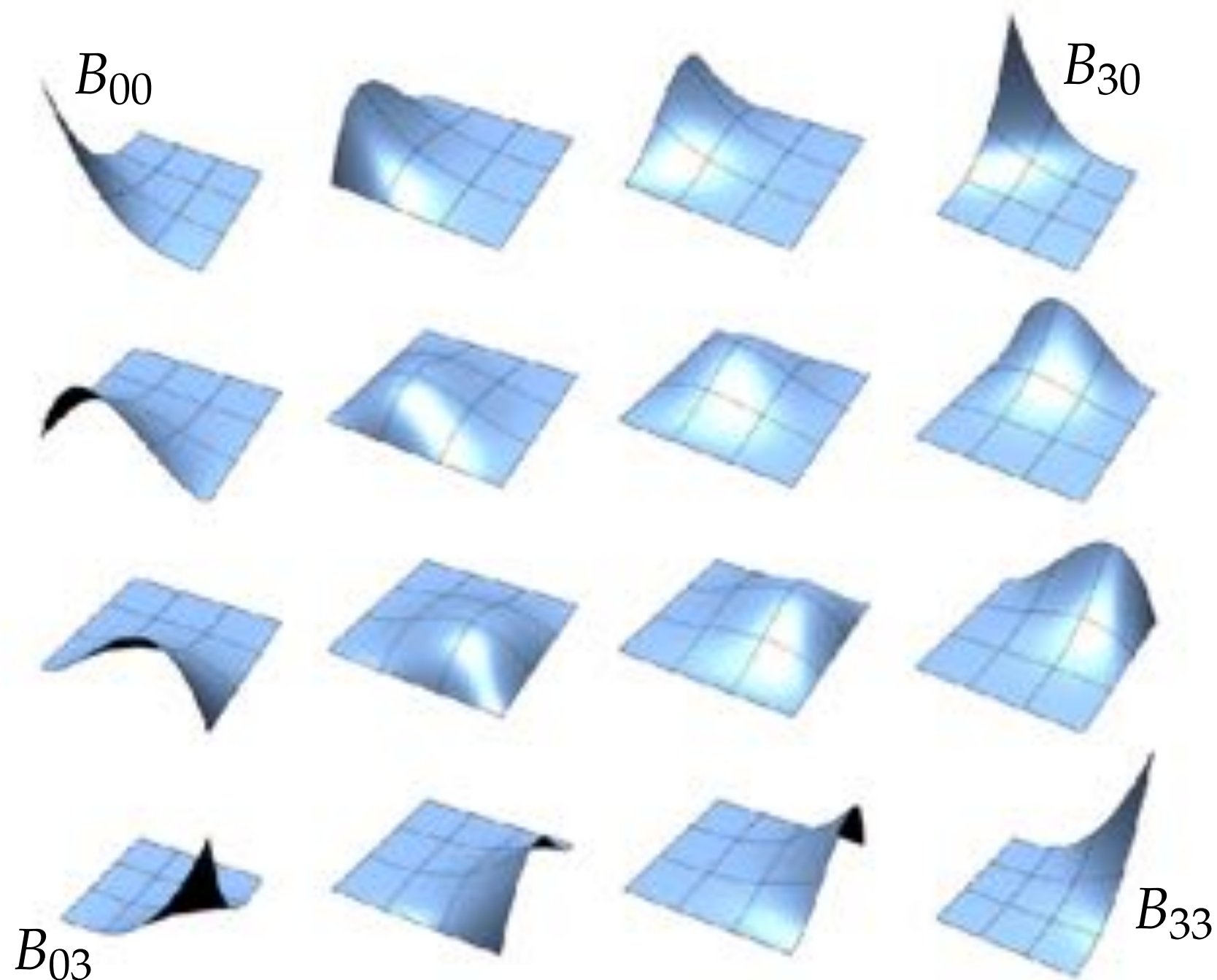
$$(f \otimes g)(u, v) := f(u)g(v)$$

# Bézier Patches

- Bézier patch is sum of (tensor) products of Bernstein bases



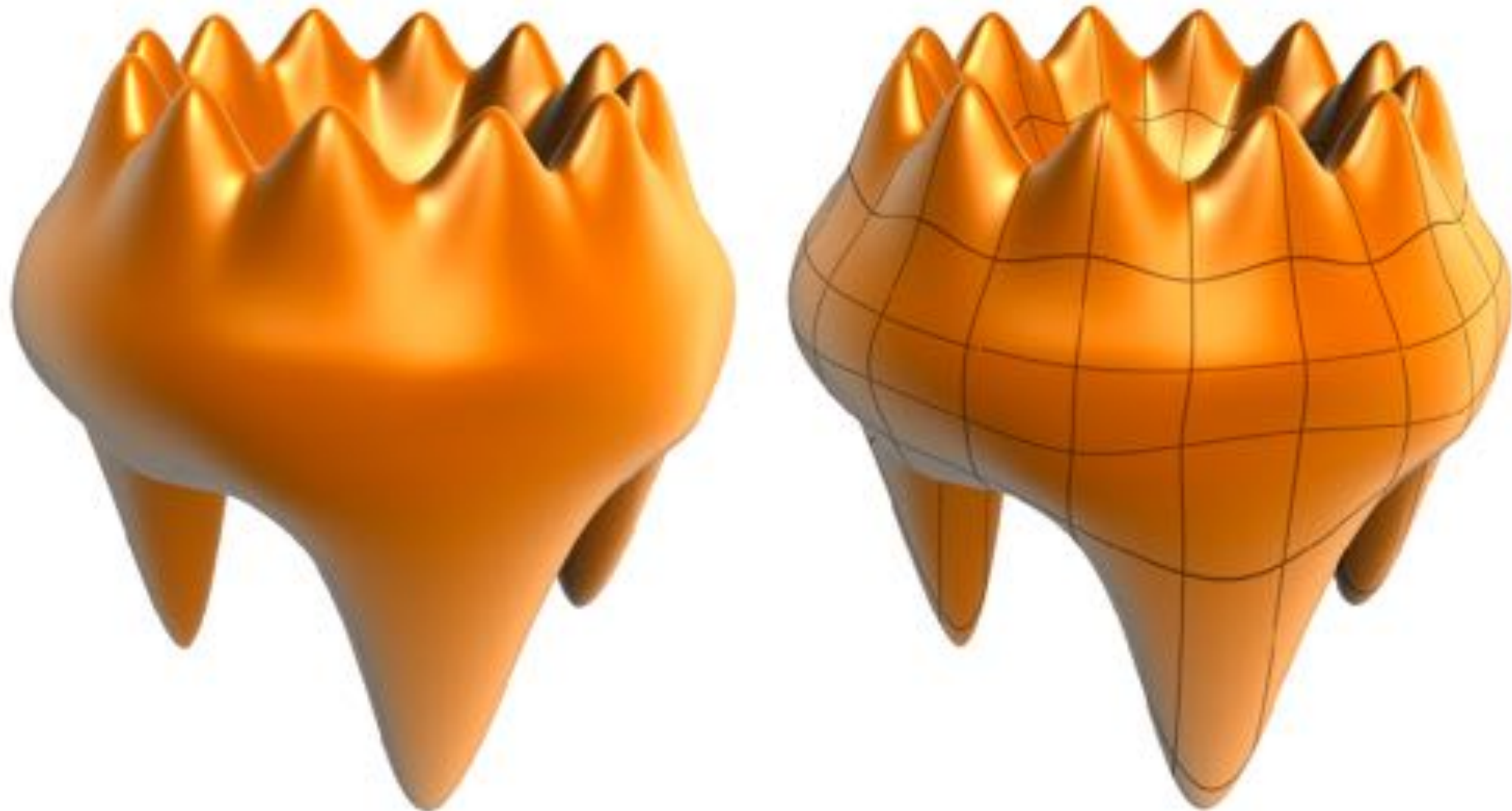
$$B_{i,j}^3(u, v) := B_i^3(u) B_j^3(v)$$



$$S(u, v) := \sum_{i=0}^3 \sum_{j=0}^3 B_{i,j}^3(u, v) \mathbf{p}_{ij}$$

# Bézier Surface

- Just as we connected Bézier curves, can connect Bézier patches to get a surface:



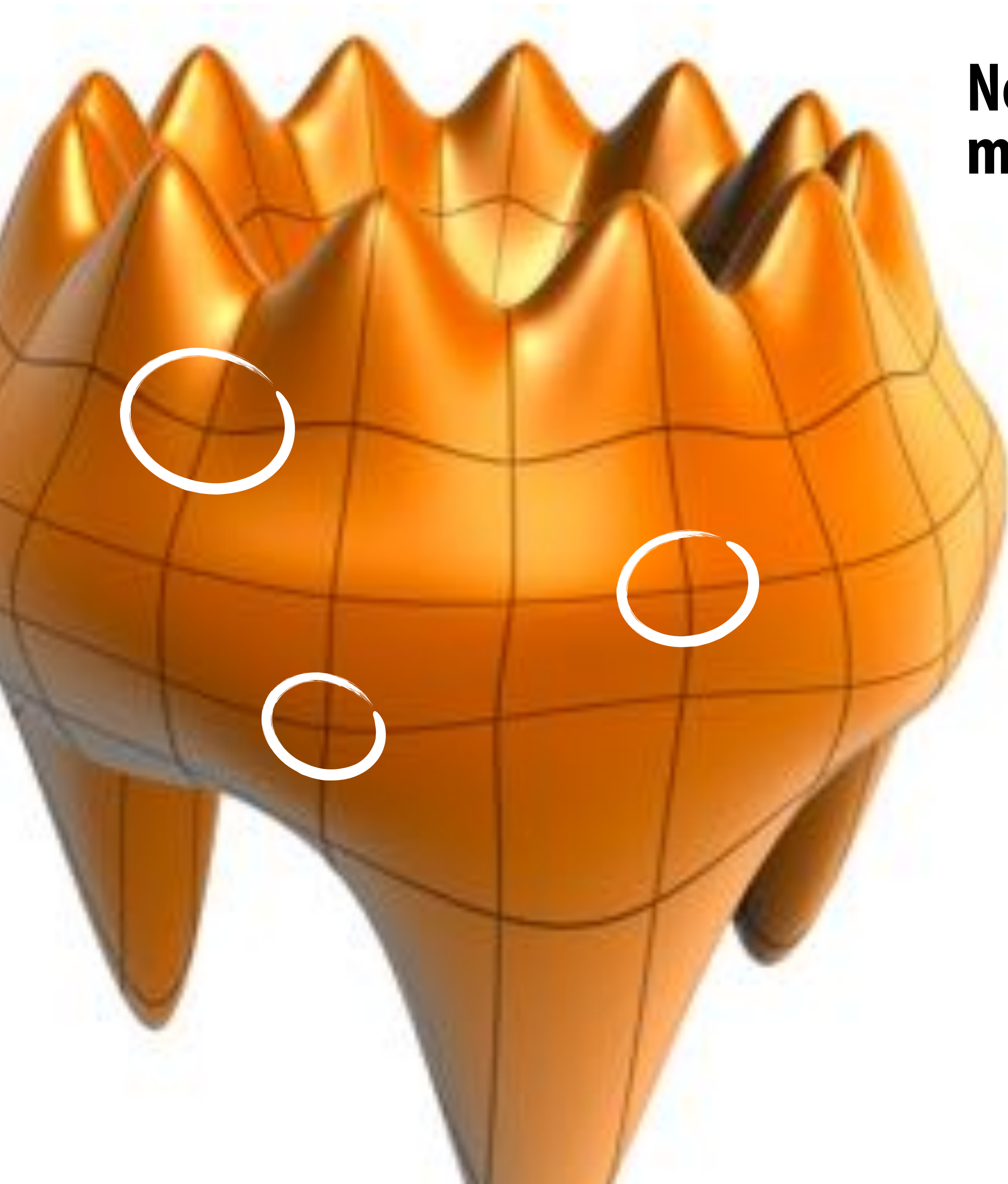
- Very easy to draw: just dice each patch into regular  $(u,v)$  grid!

**Q: Can we always get tangent continuity?**

**(Think: how many constraints? How many degrees of freedom?)**

**Notice anything fishy  
about the last picture?**

# Bézier Patches are Too Simple



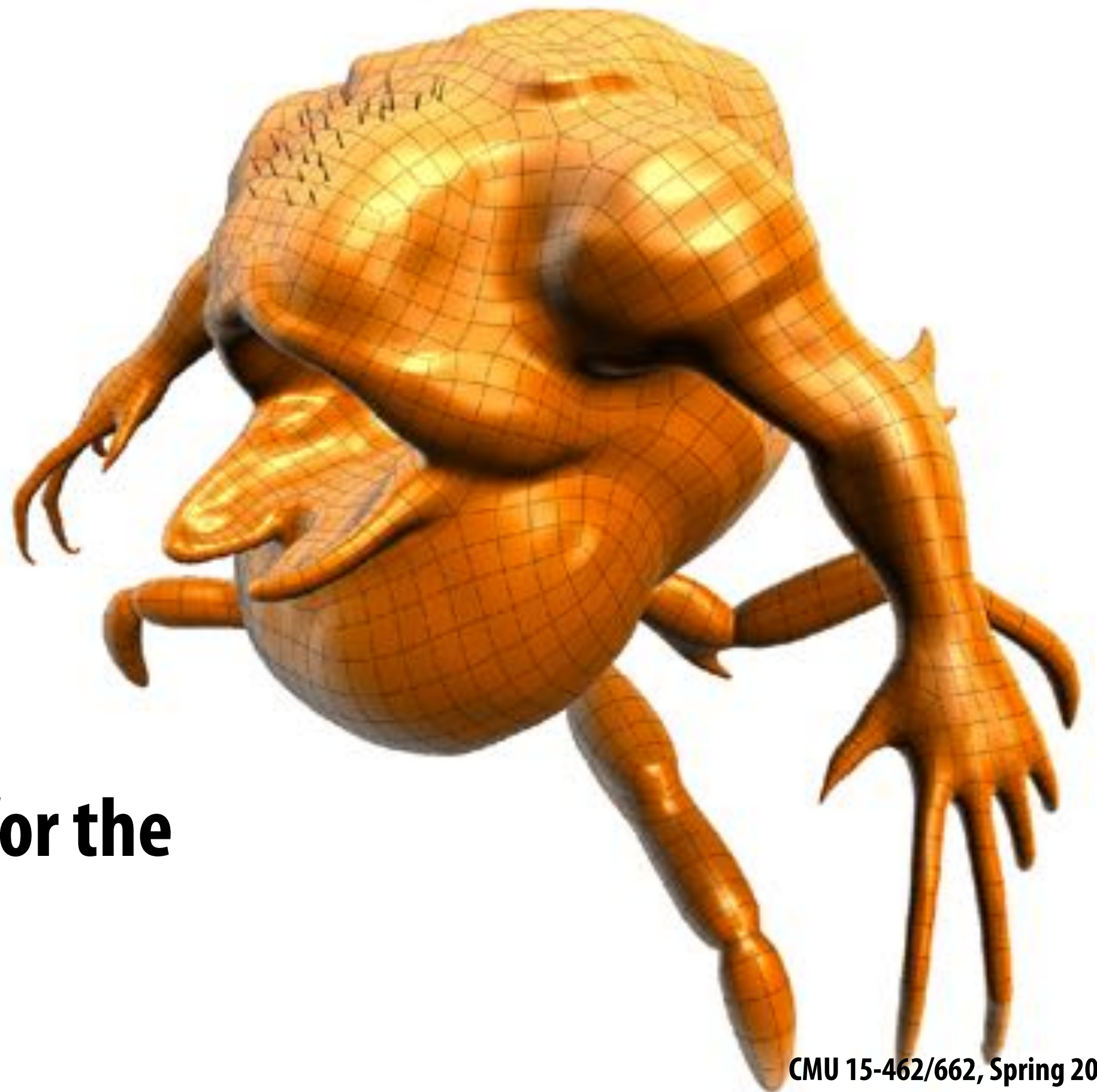
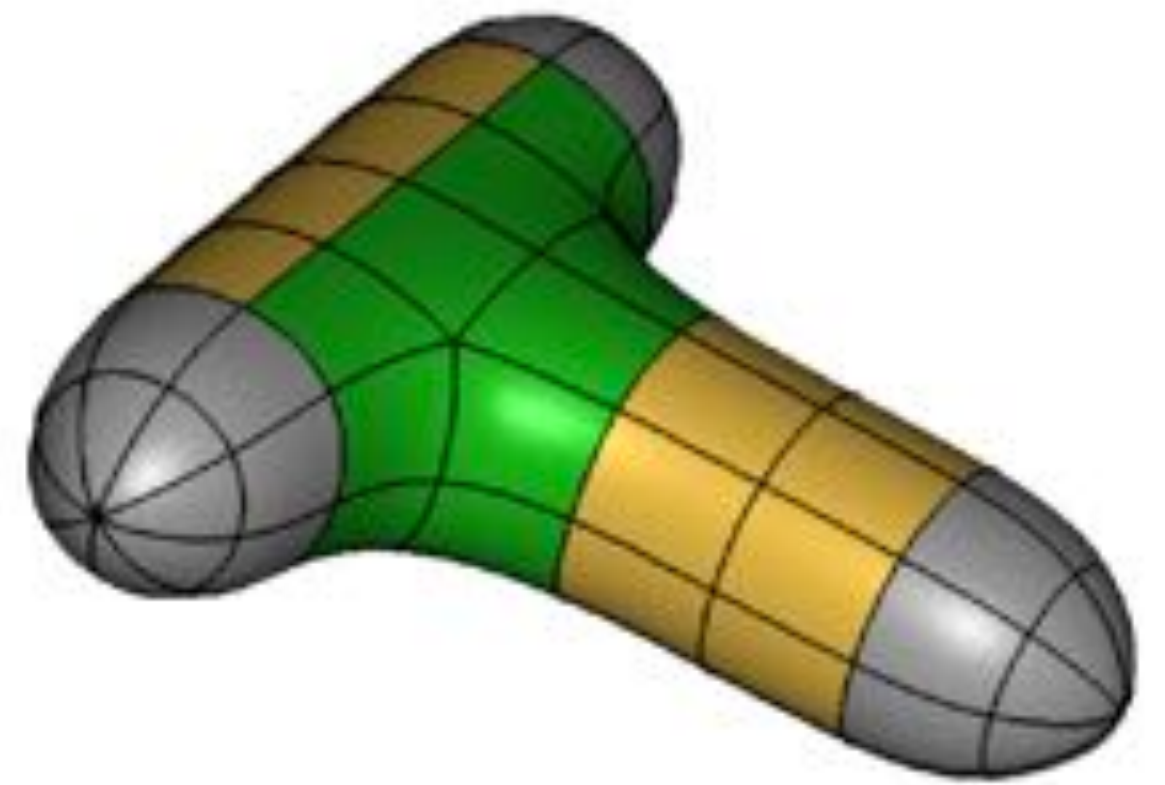
**Notice that exactly four patches meet around every vertex!**

**In practice, far too constrained.**

**To make interesting shapes (with good continuity), we need patches that allow more interesting connectivity...**

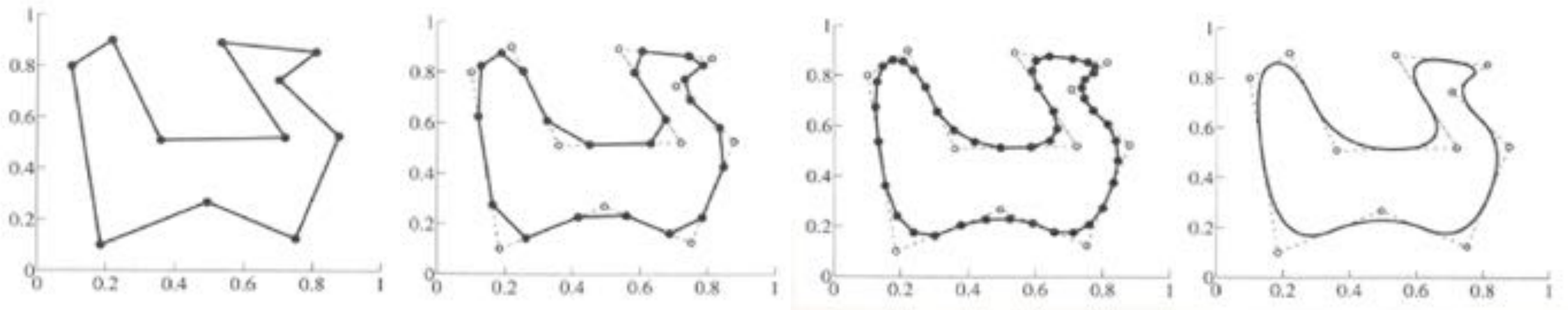
# Spline patch schemes

- There are many alternatives!
- NURBS, Gregory, Pm, polar...
- Tradeoffs:
  - degrees of freedom
  - continuity
  - difficulty of editing
  - cost of evaluation
  - generality
  - ...
- As usual: pick the right tool for the job!



# Subdivision (Explicit or Implicit?)

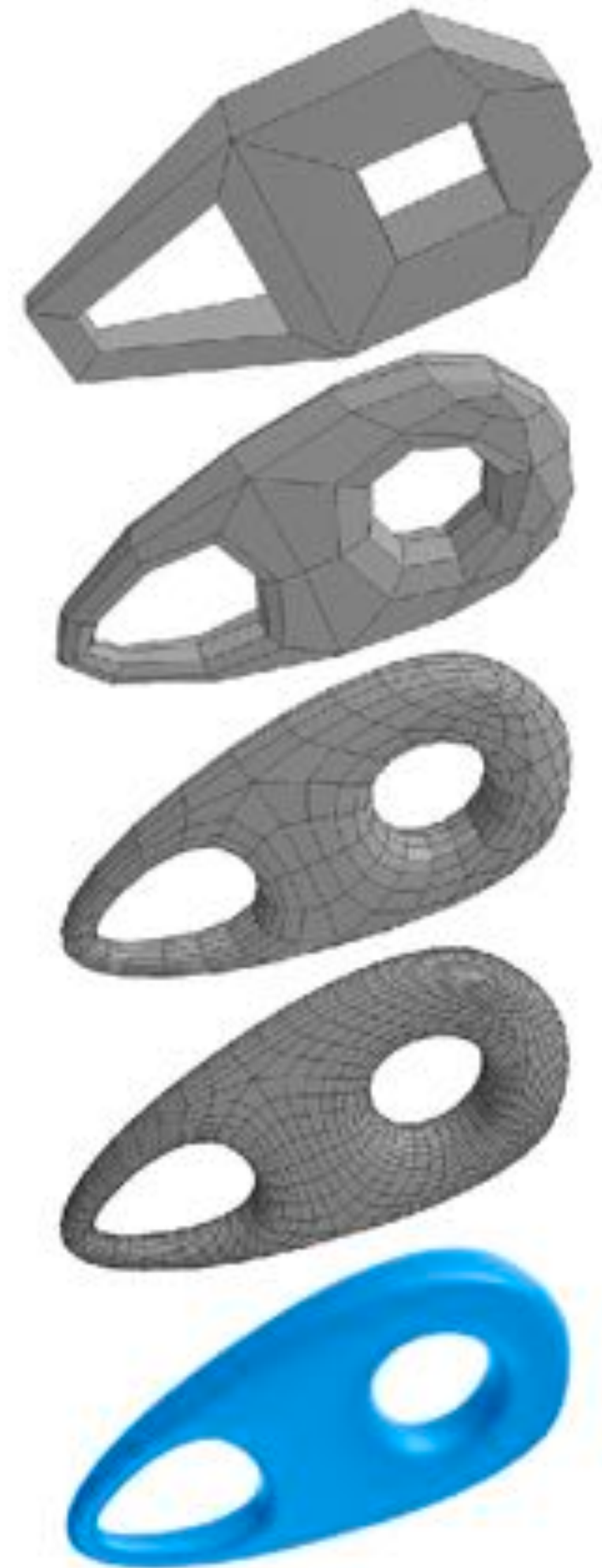
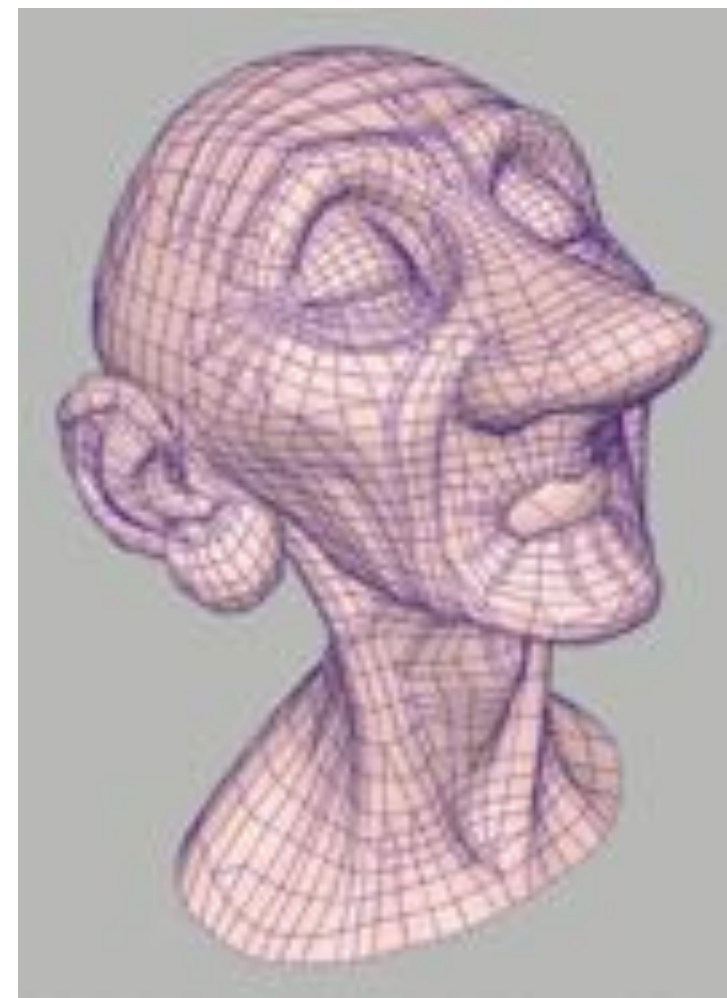
- **Alternative starting point for curves/surfaces: subdivision**
- **Start with control curve**
- **Insert new vertex at each edge midpoint**
- **Update vertex positions according to fixed rule**
- **For careful choice of averaging rule, yields smooth curve**
  - **Some subdivision schemes correspond to well-known spline schemes!**





# Subdivision Surfaces (Explicit)

- Start with coarse polygon mesh (“control cage”)
- Subdivide each element
- Update vertices via local averaging
- Many possible rule:
  - Catmull-Clark (quads)
  - Loop (triangles)
  - ...
- Common issues:
  - interpolating or approximating?
  - continuity at vertices?
- Easier than splines for modeling; harder to evaluate pointwise



# Subdivision in Action (Pixar's "Geri's Game")

# Next time: Curves, Surfaces, & Meshes

