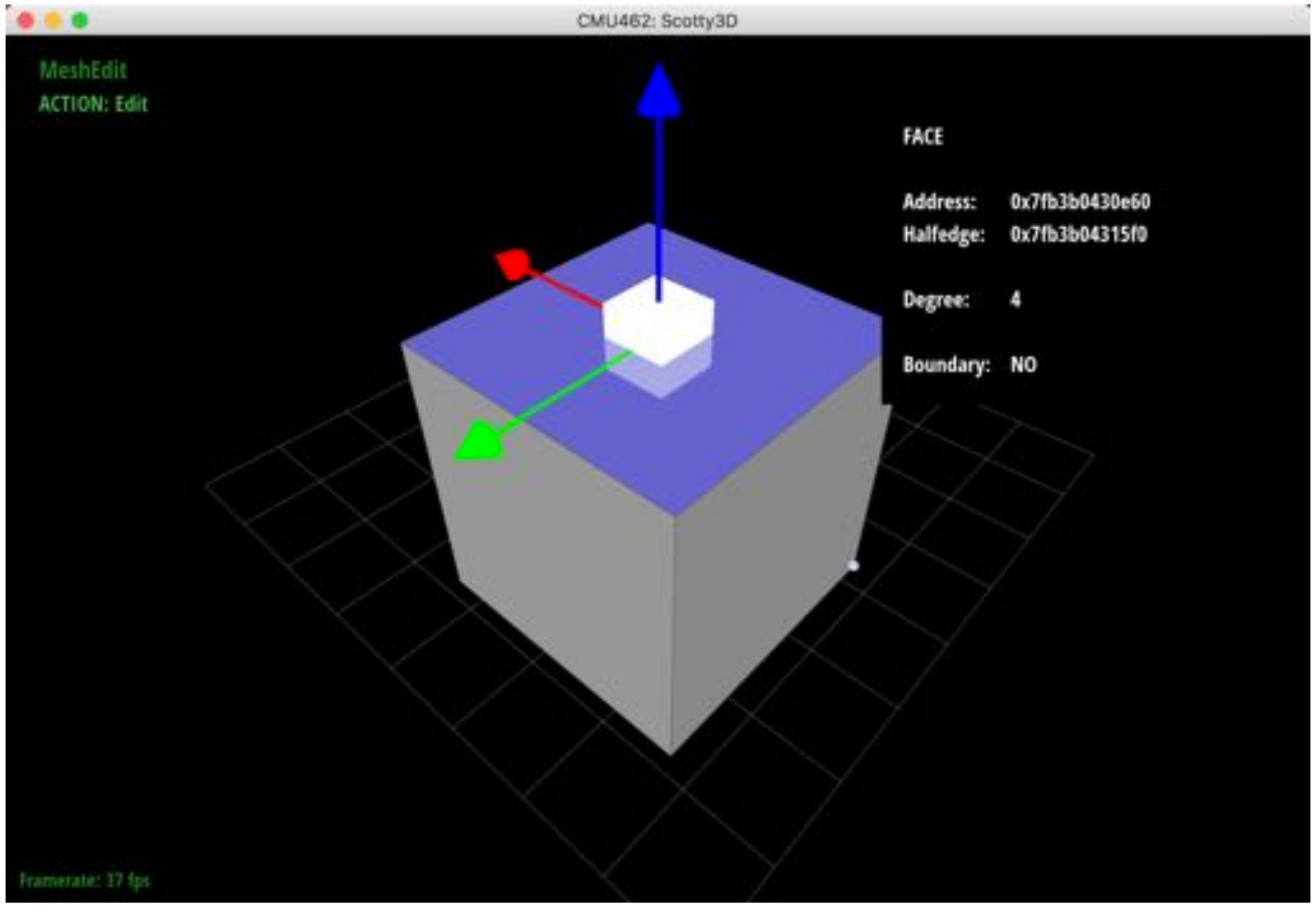
Introduction to Geometry

Computer Graphics
CMU 15-462/15-662

Assignment 2

- **Start building up "Scotty3D"; first part is 3D modeling**

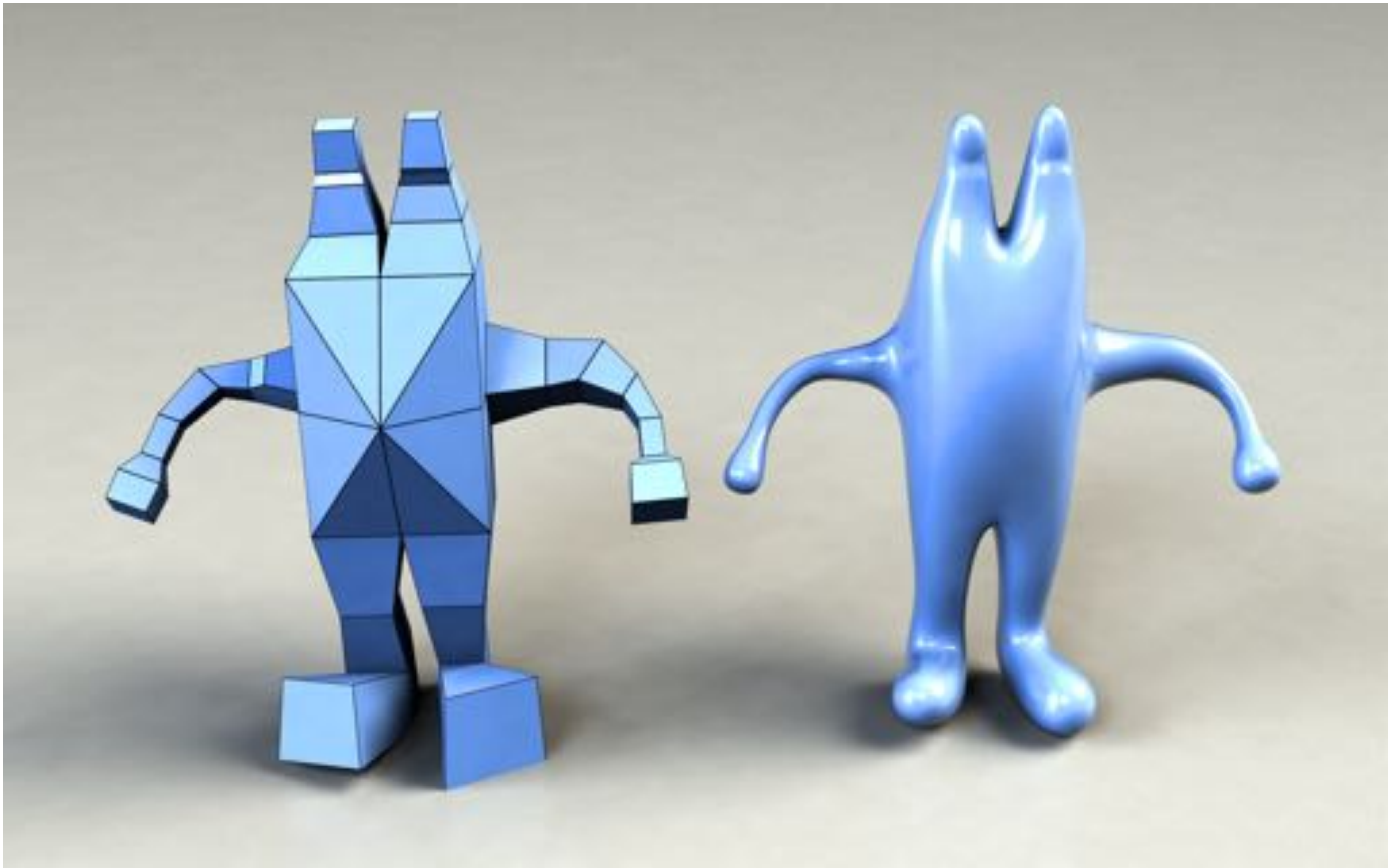
Scotty3D



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

3D Modeling Competition

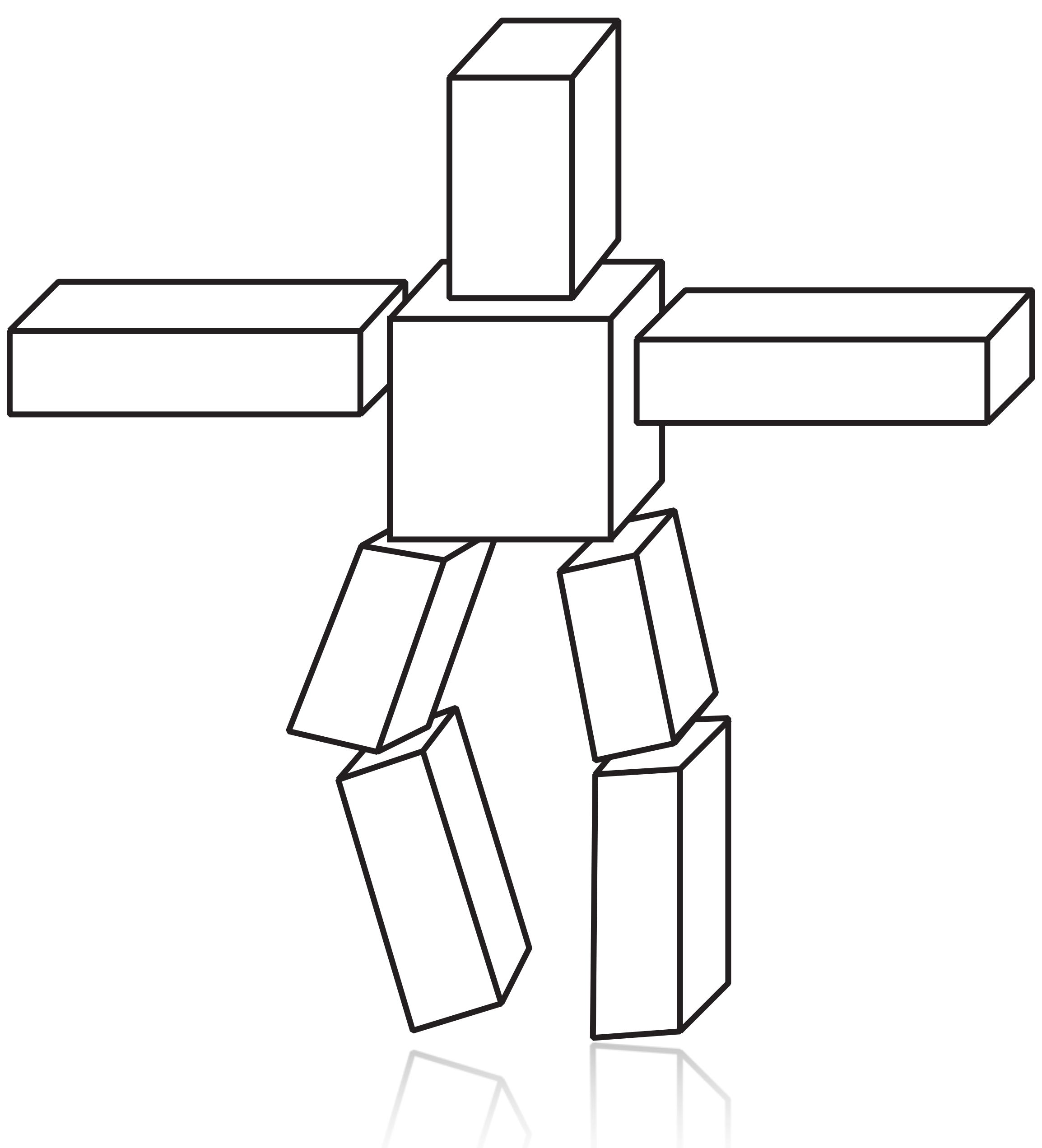
- **Don't just make great software… make great art! :-)**



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

Increasing the complexity of our models

Transformations

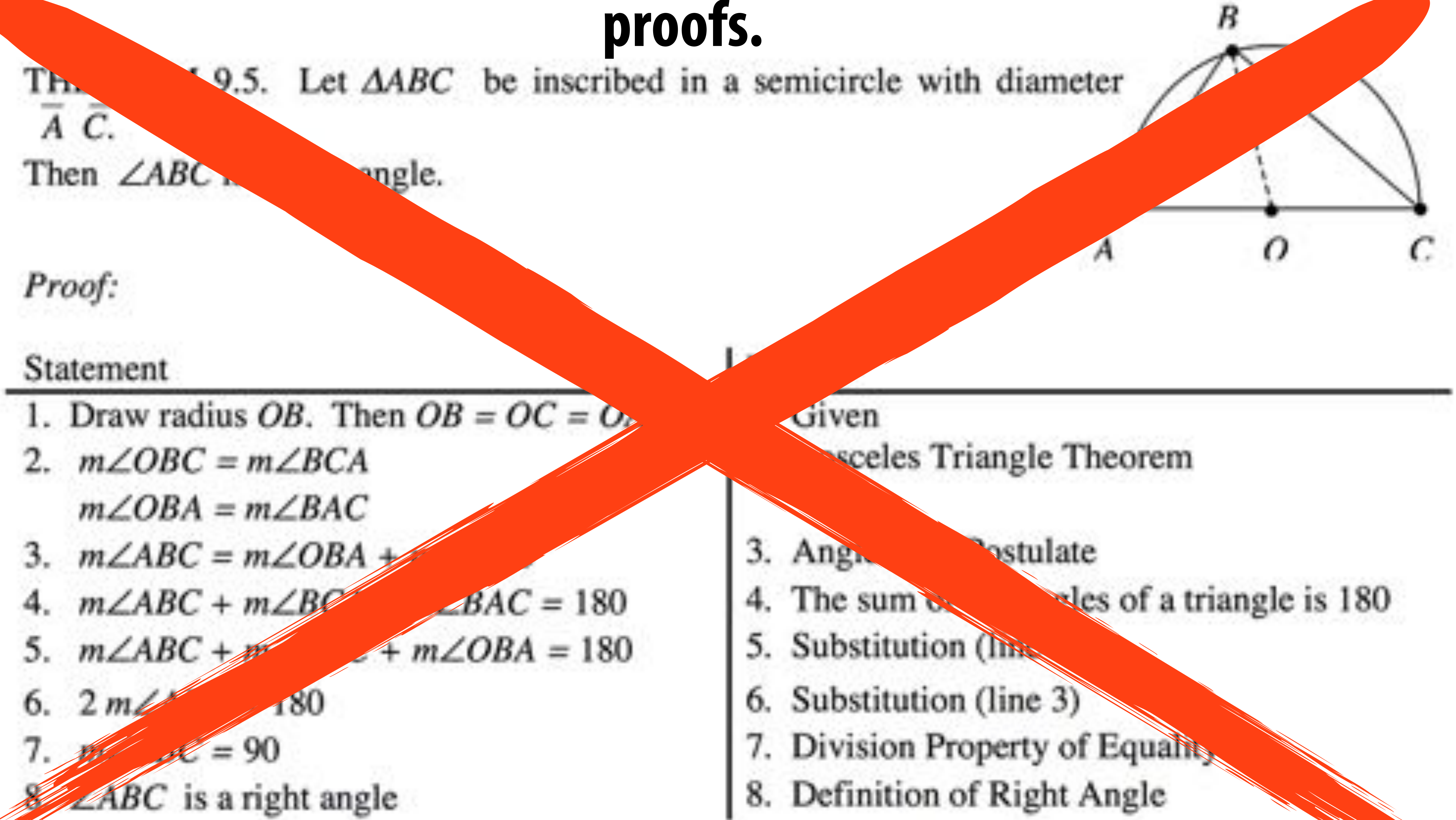


Geometry

Materials, lighting, ...

Q: What is geometry?

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



Ceci n'est pas géométrie.

See: Paul Lockhart, "A Mathematician's Lament"

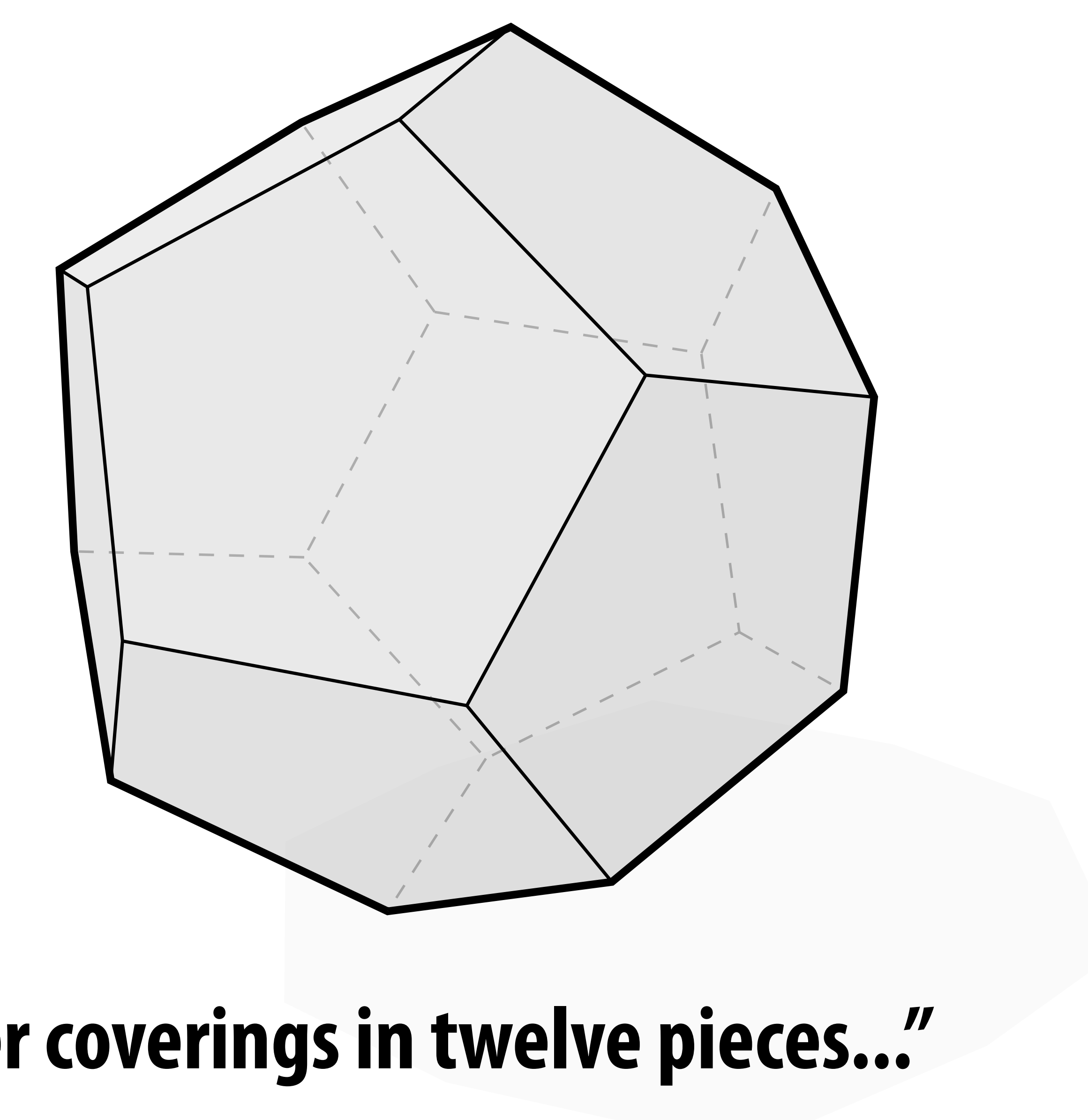
What is geometry?

"Earth" "measure"

ge•om•et•ry /jēˈämətrē/ *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..."

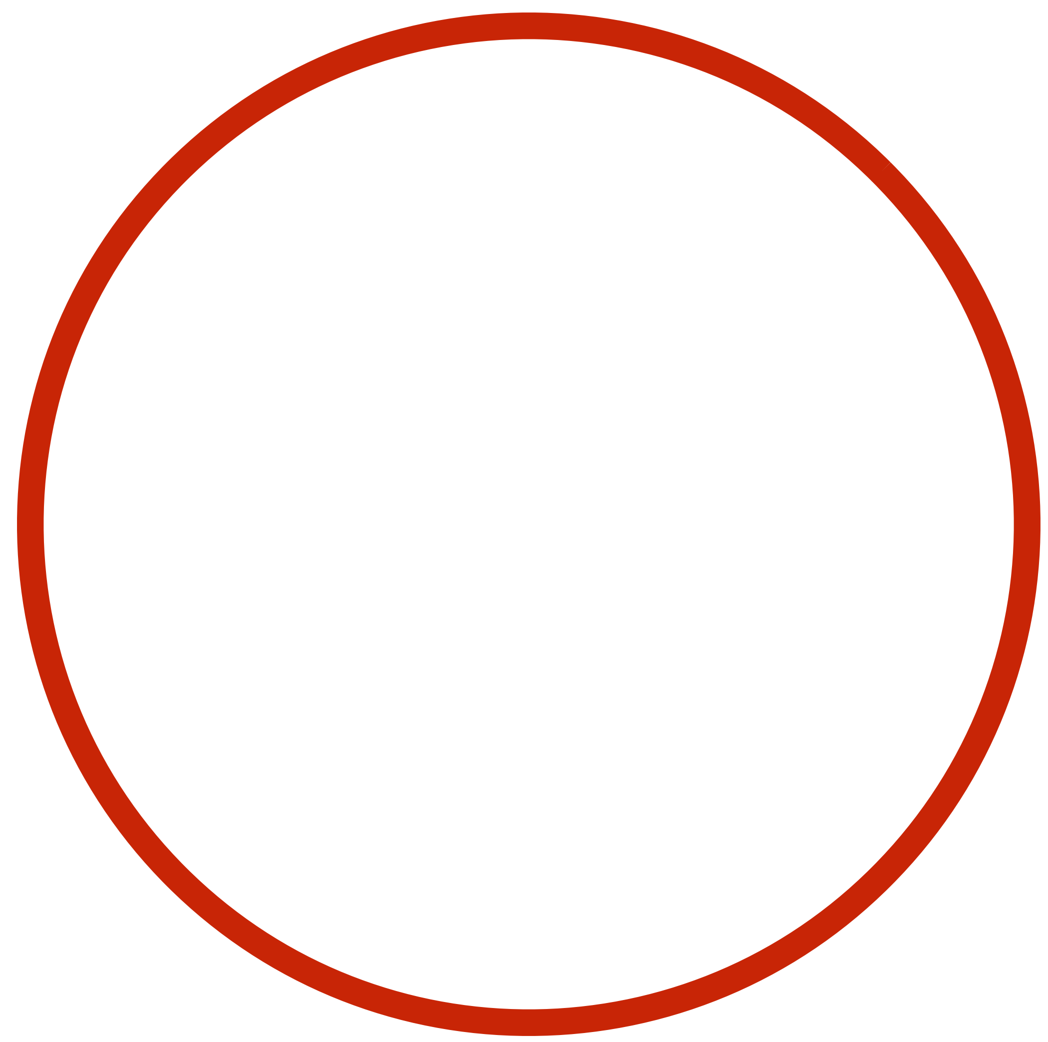
How can we describe geometry?

IMPLICIT

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

LINGUISTIC

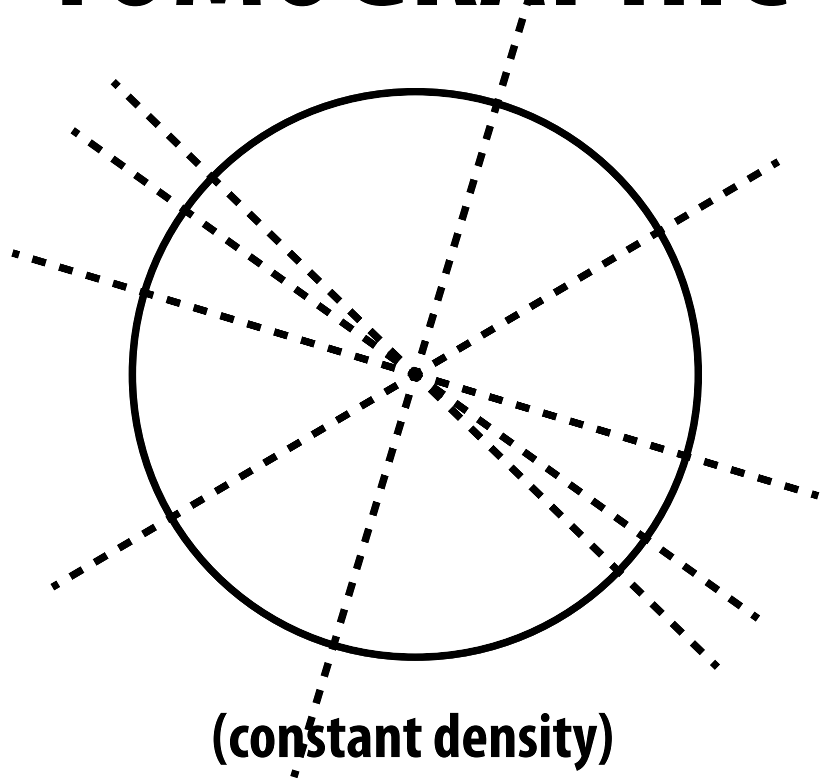
“unit circle”



EXPLICIT

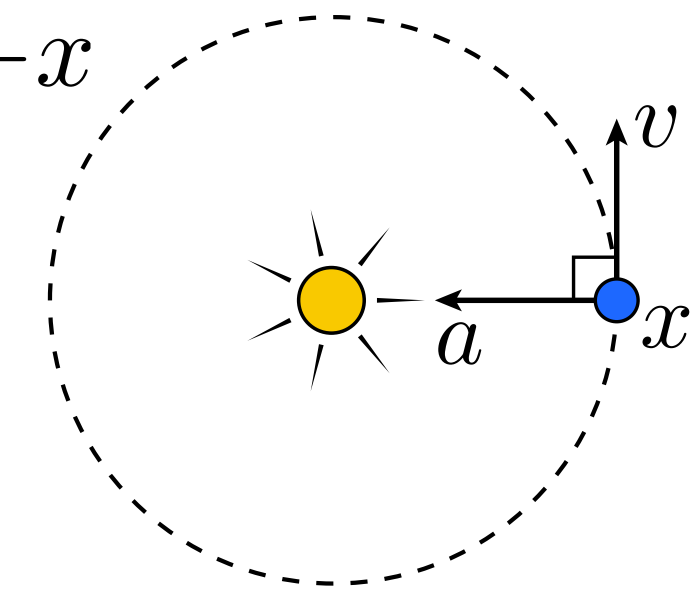
$$(\underbrace{\cos\theta}_{x},\underbrace{\sin\theta}_{y})$$

TOMOGRAPHIC

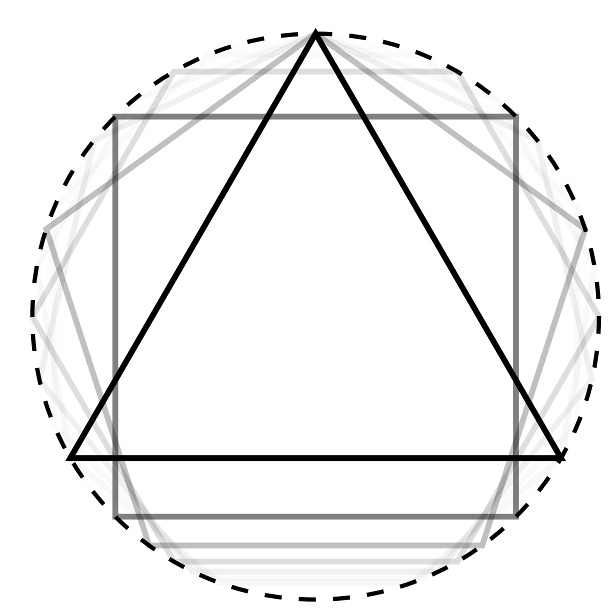


DYNAMIC

$$\frac{d^2}{dt^2}x=-x$$



DISCRETE

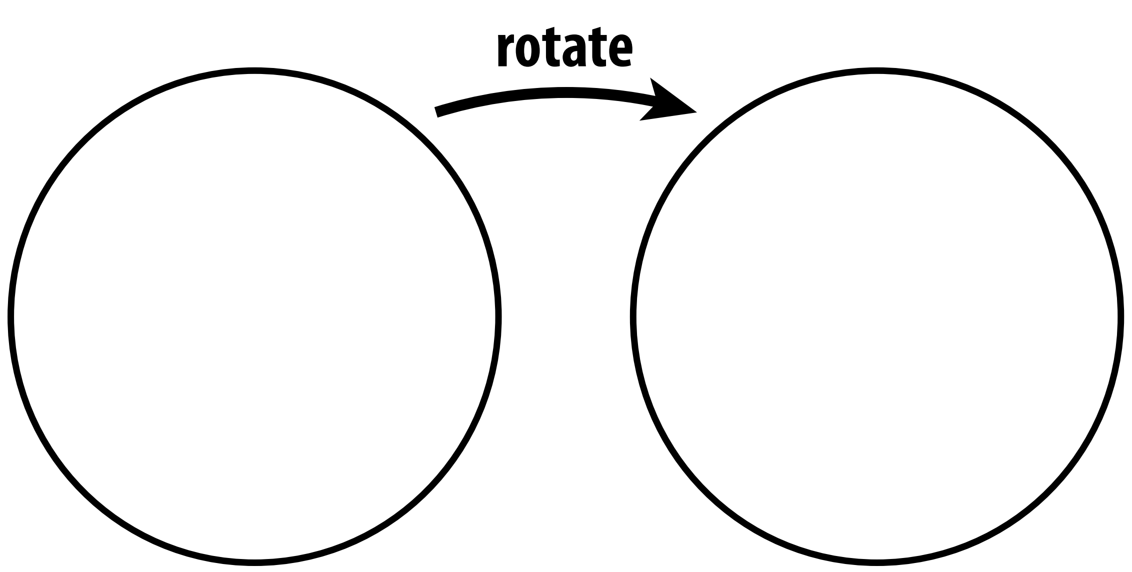


$n \to \infty$

CURVATURE

$$\kappa=1$$

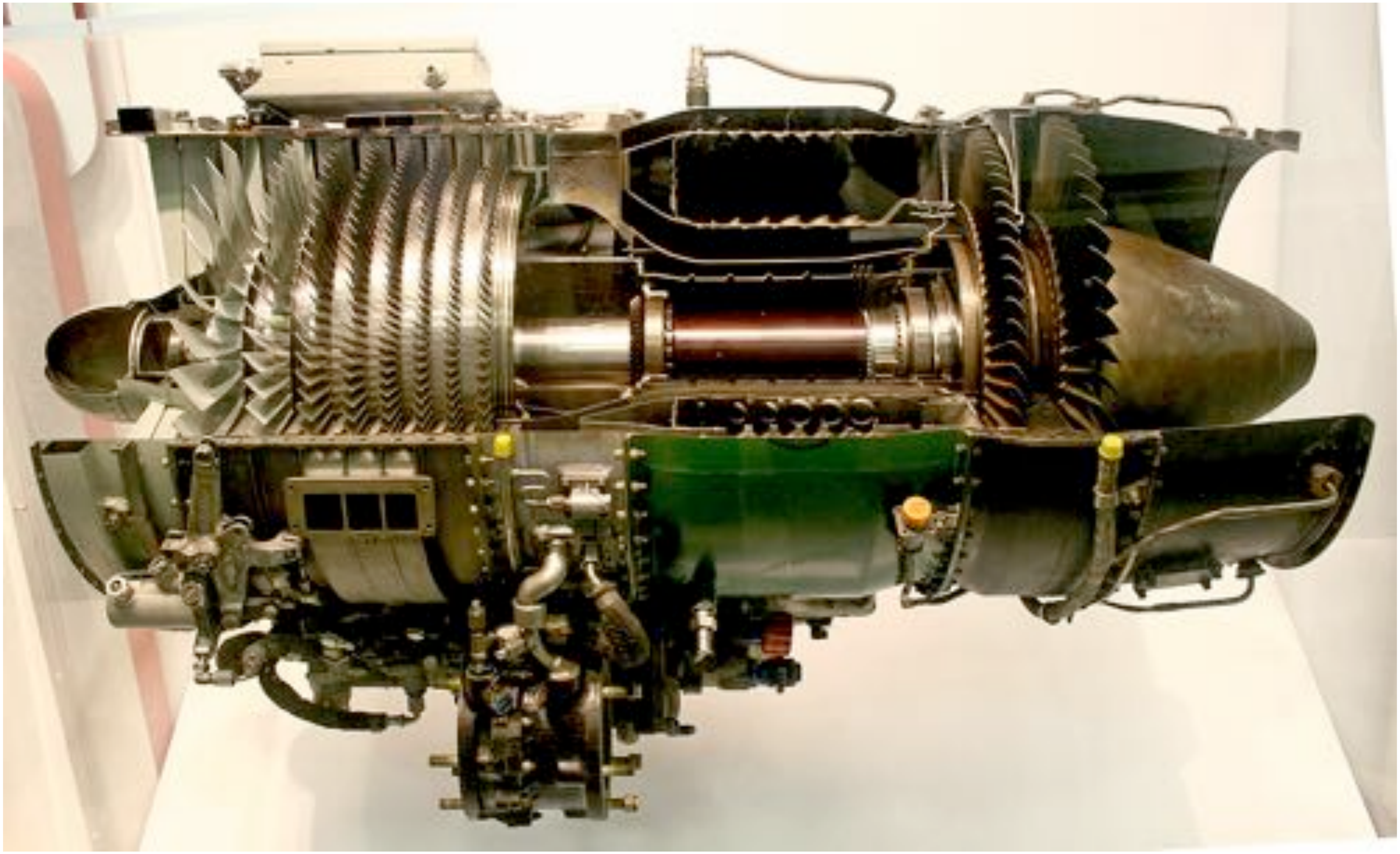
SYMMETRIC



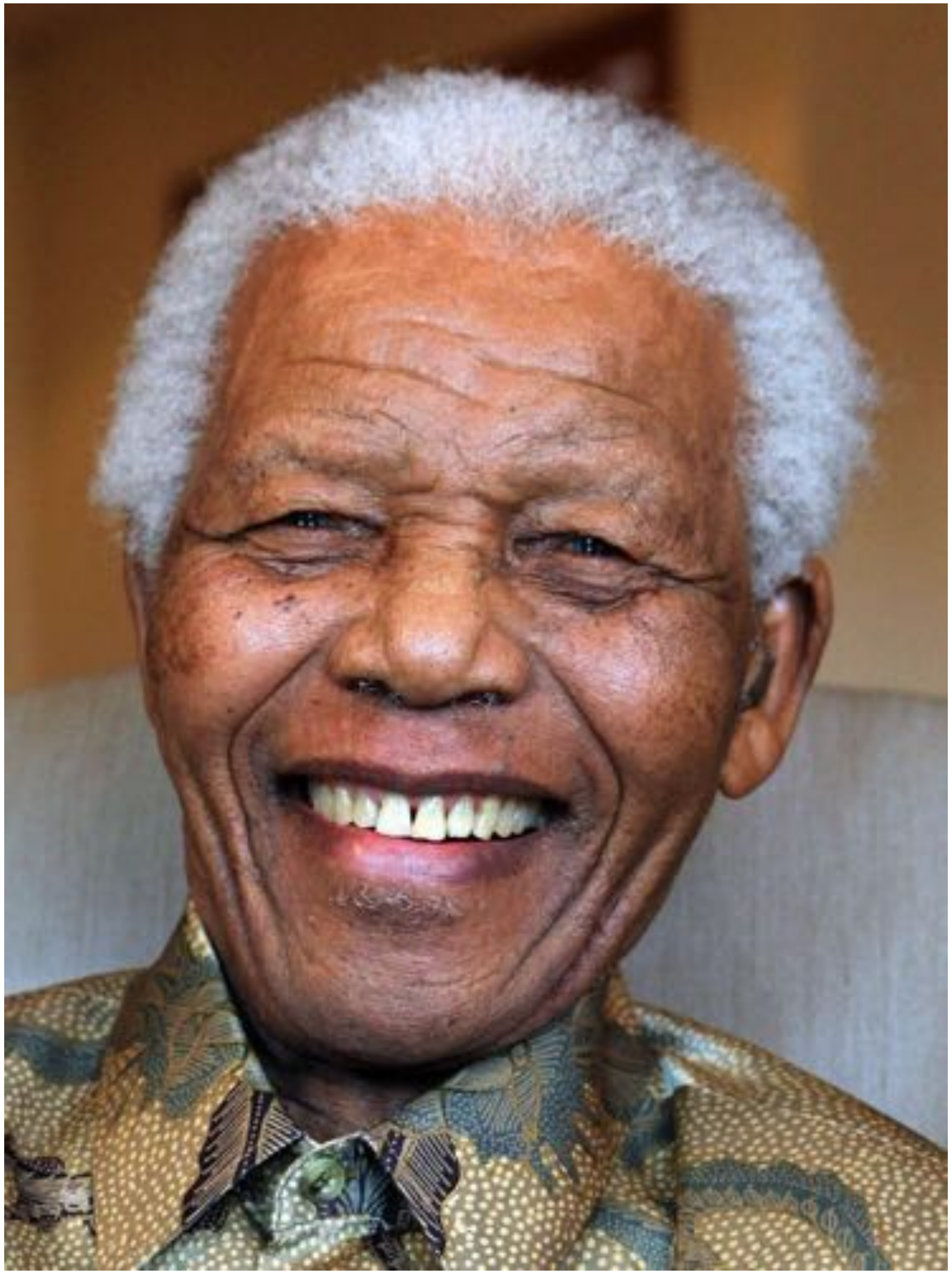
Given all these options, what's the best way to encode geometry on a computer?

Examples of geometry

Examples of geometry



Examples of geometry



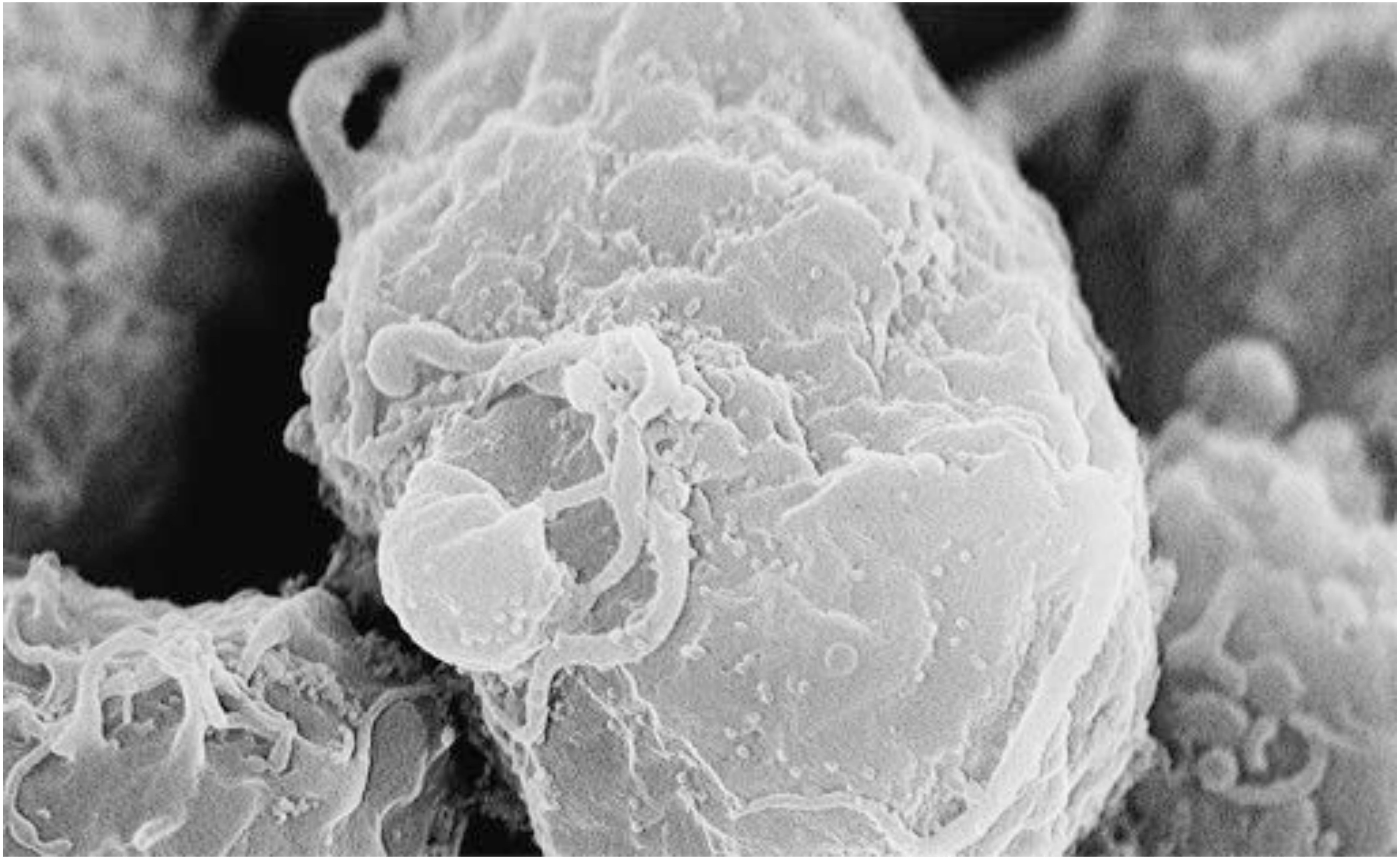
Examples of geometry

Examples of geometry

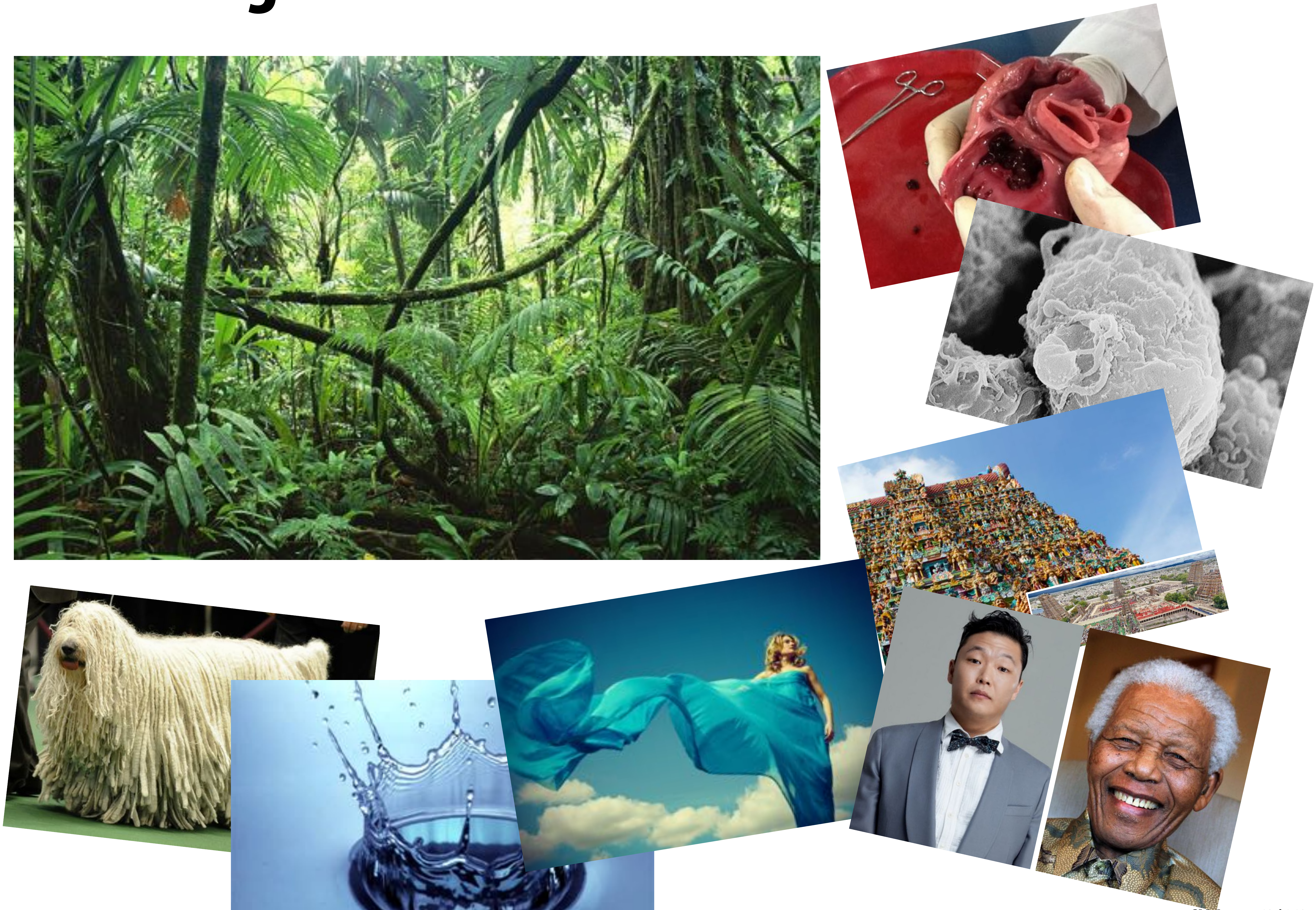
Examples of geometry

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."

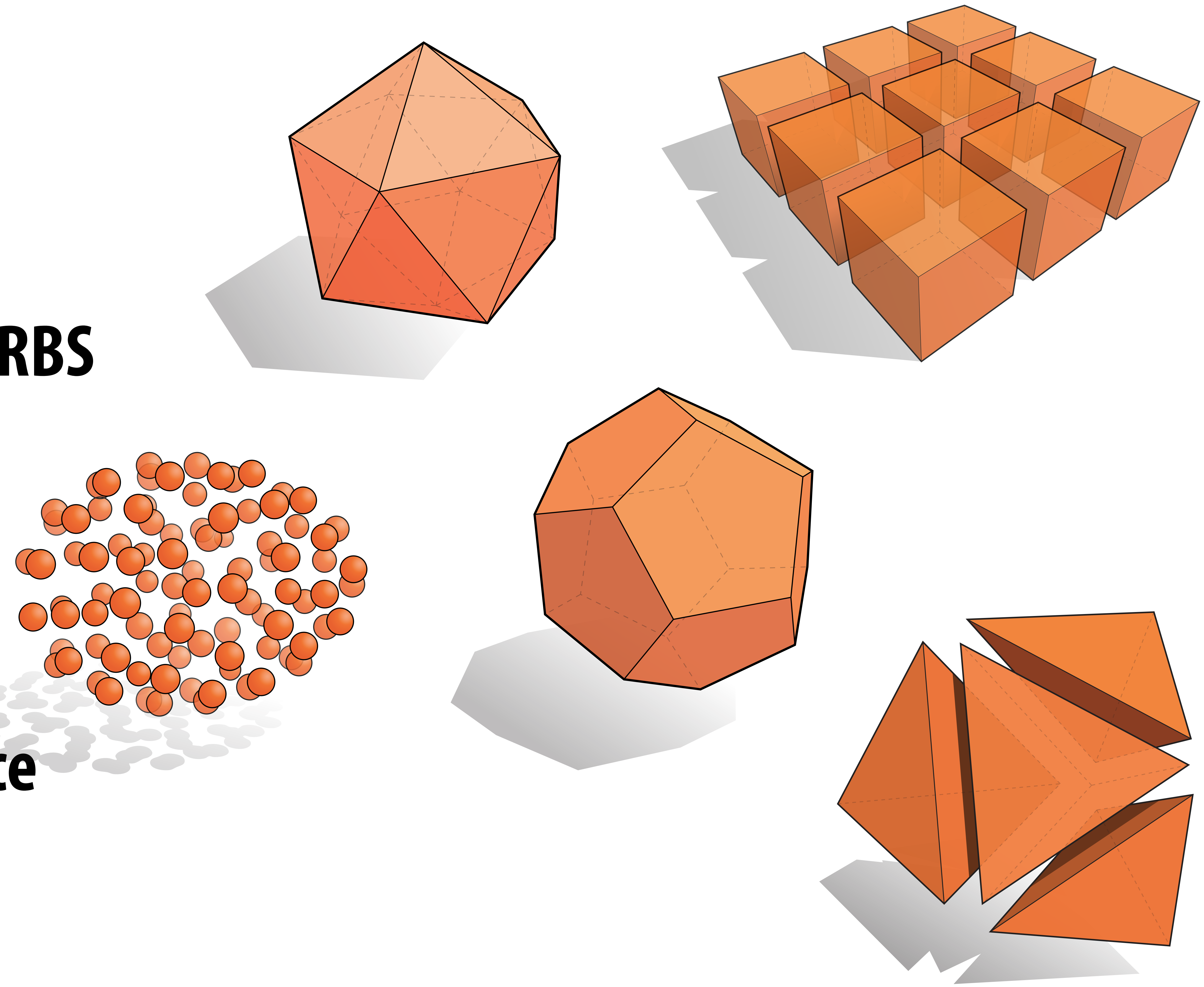
—David Baraff

Senior Research Scientist
Pixar Animation Studios

Slide cribbed from Jeff Erickson.

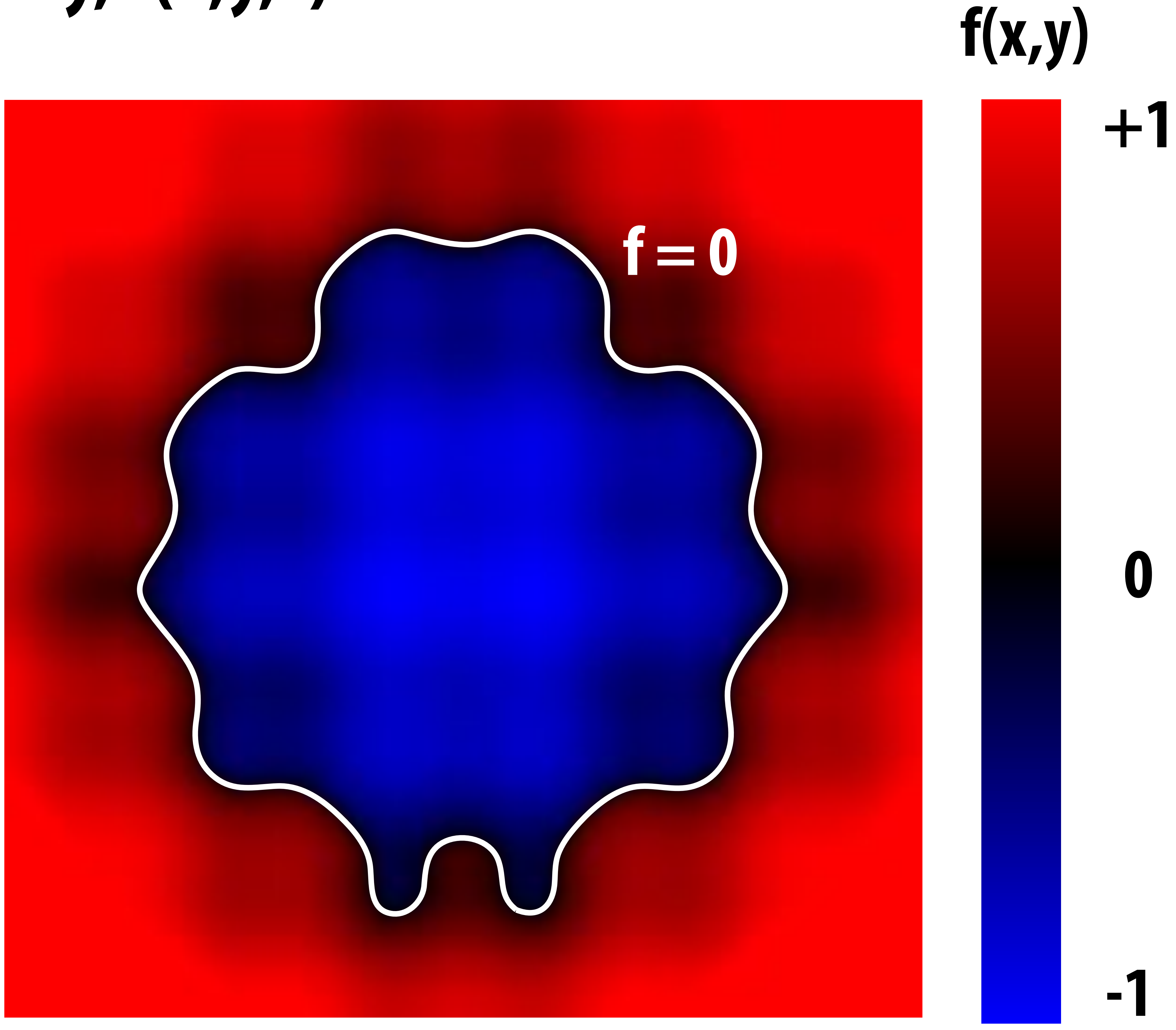
Many ways to digitally encode geometry

- **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**



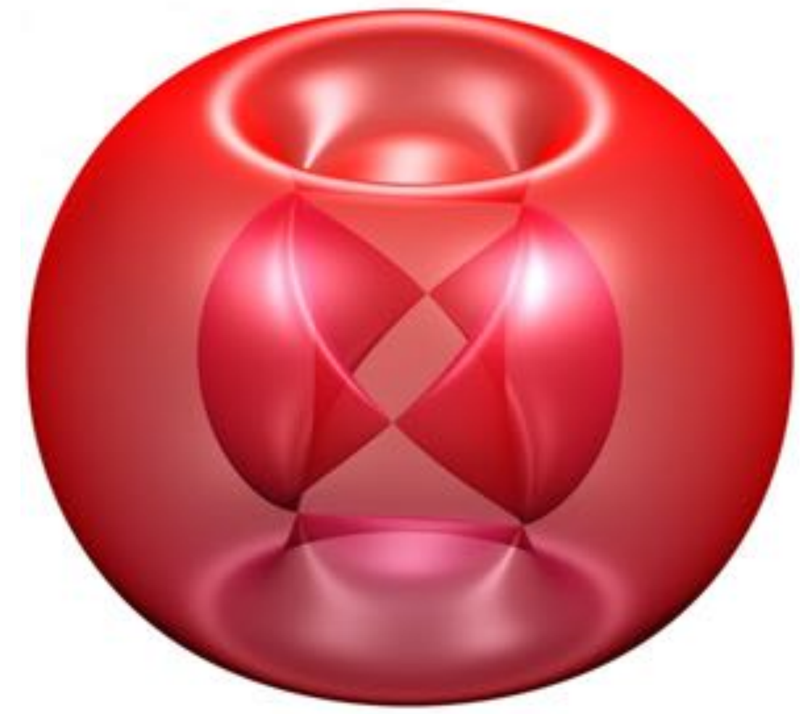
"Implicit" Representations of Geometry

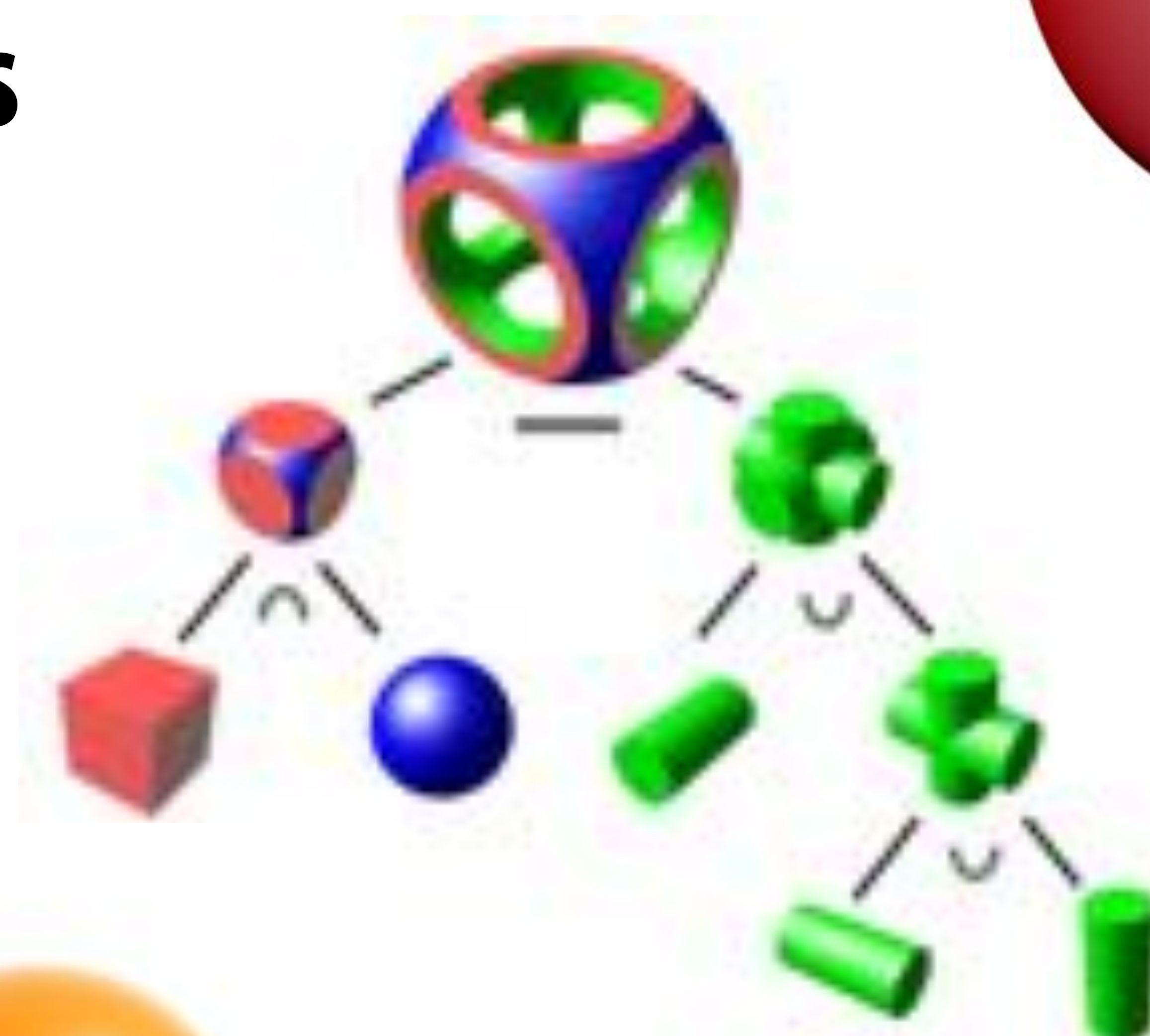
- Points aren't known directly, but satisfy some relationship
- E.g., unit sphere is all points such that $x^2+y^2+z^2=1$
- More generally, $f(x,y,z) = 0$



Many implicit representations in graphics

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





(Will see some of these a bit later.)

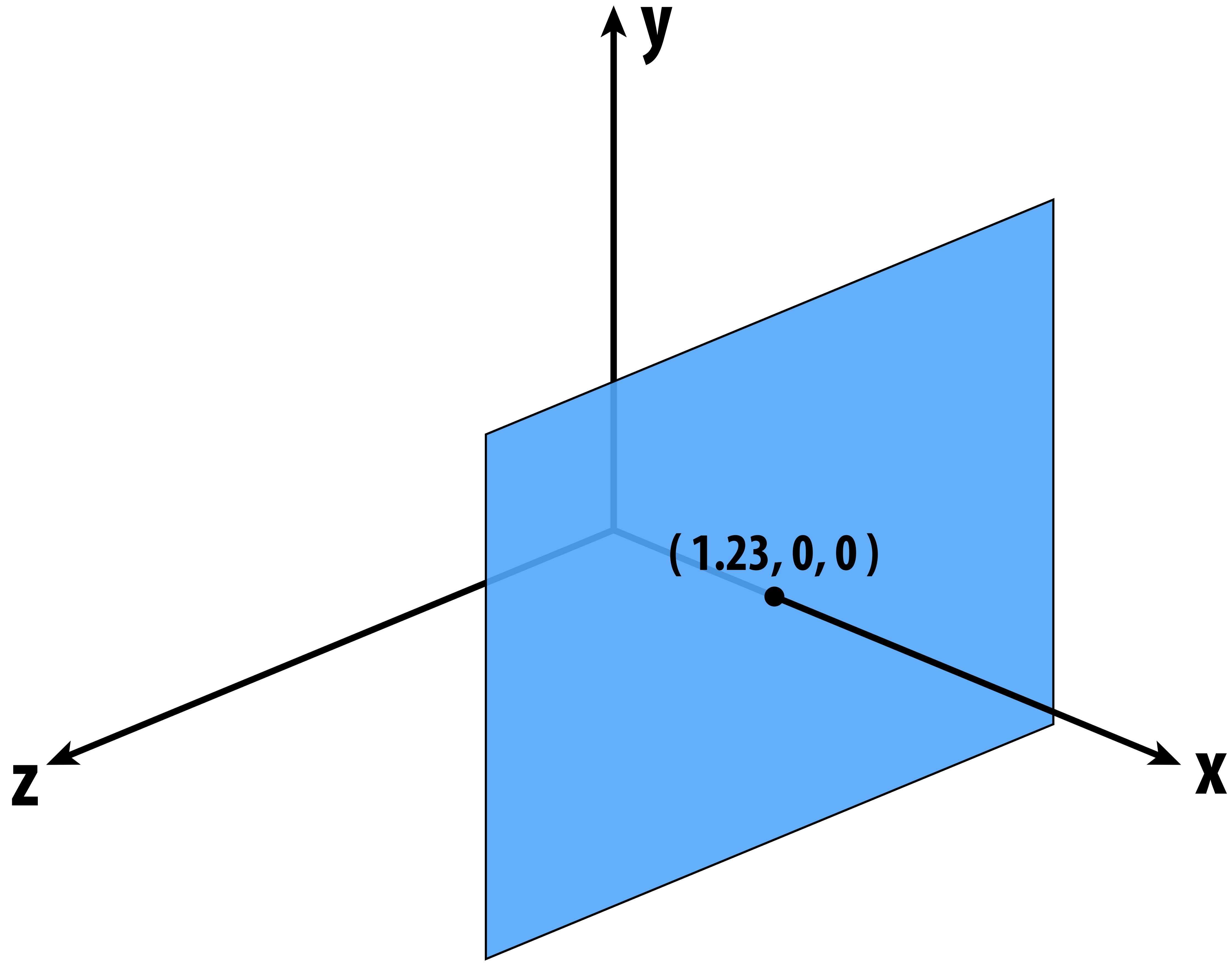
But first, let's play a game:

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):



Observation: 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.

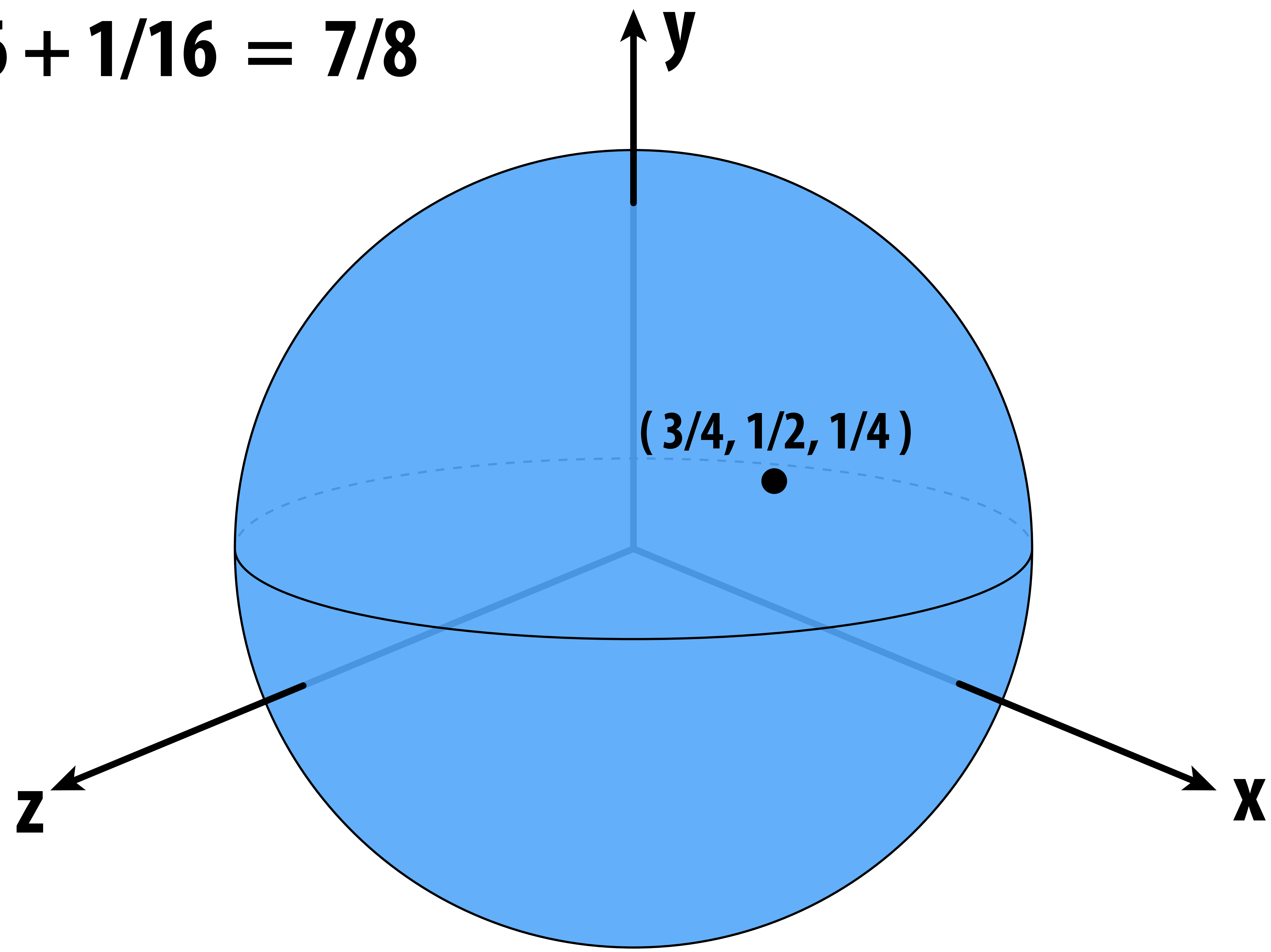
Check if this point is inside the unit sphere

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

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

$$7/8 < 1$$

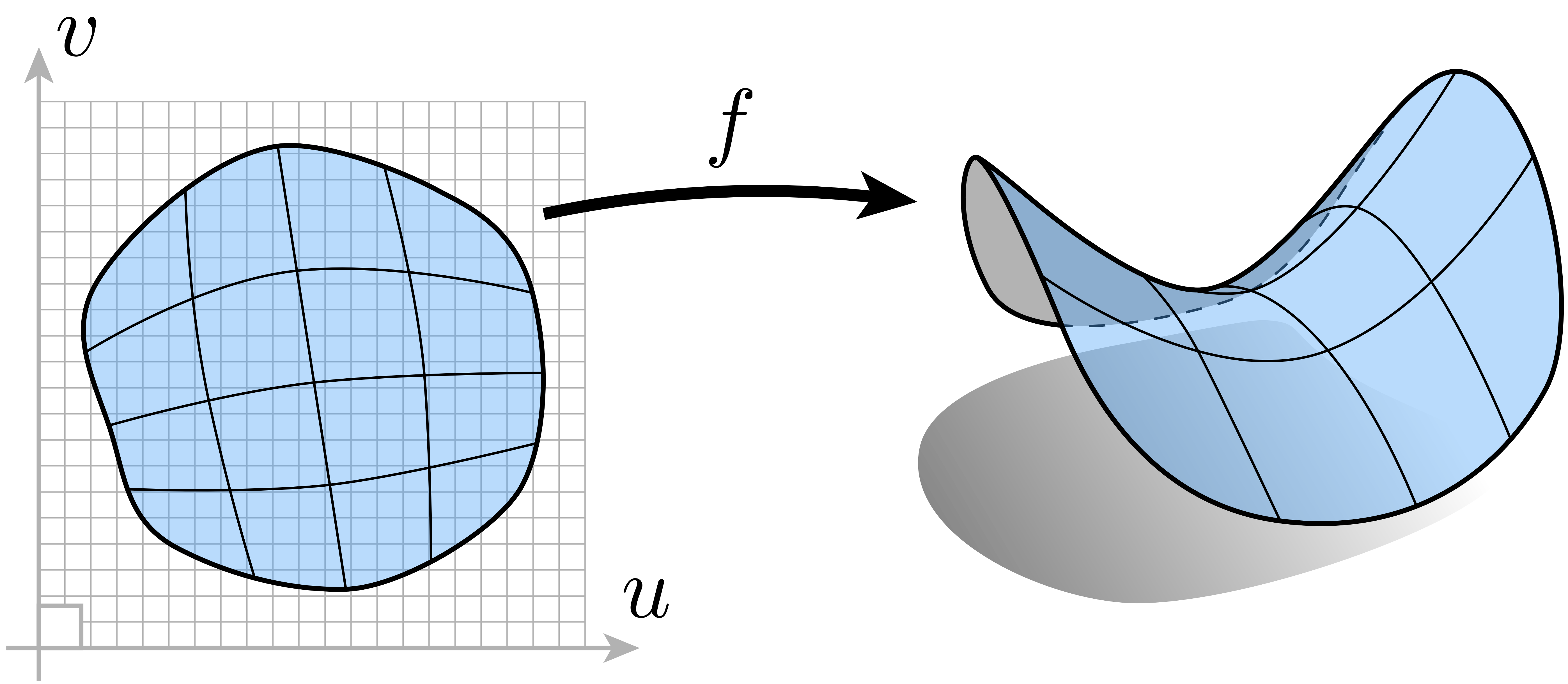
YES.



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

"Explicit" Representations of Geometry

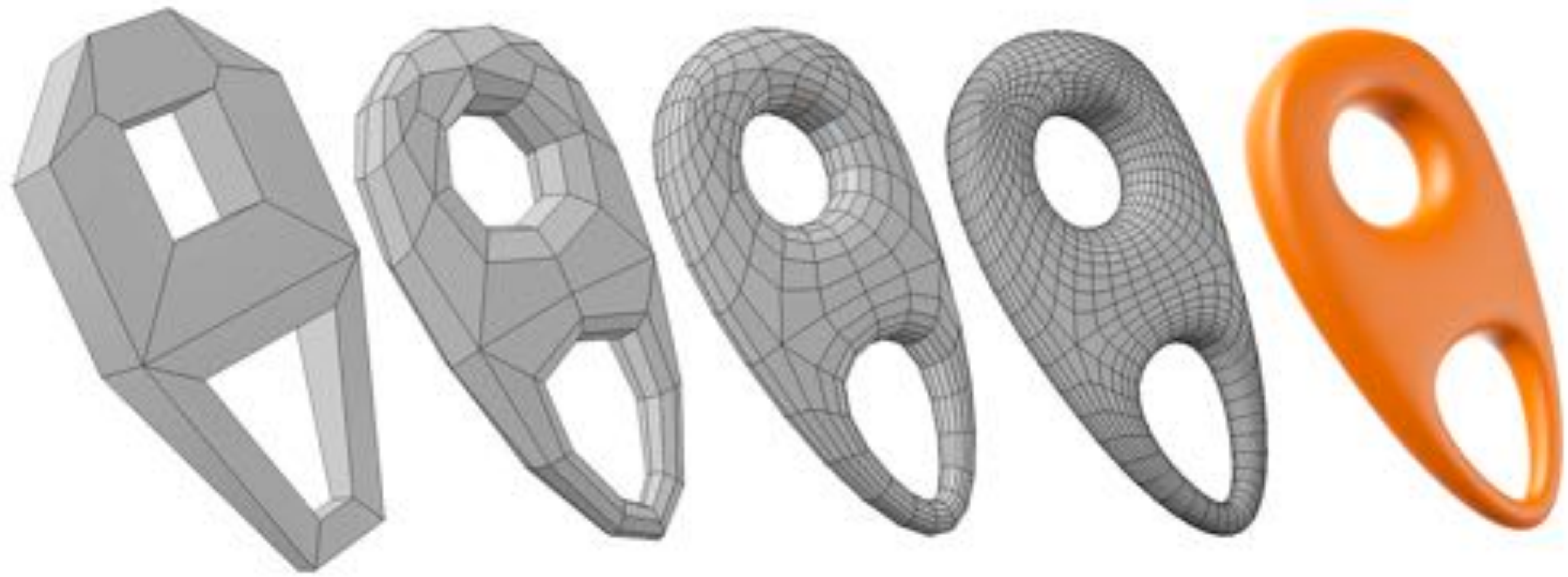
- **All points are given directly**
- **E.g., points on sphere are** $(\cos(u)\sin(v), \sin(u)\sin(v), \cos(v)),$
  $\text{for } 0 \leq u < 2\pi \text{ and } 0 \leq v \leq \pi$
- **More generally:** $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$

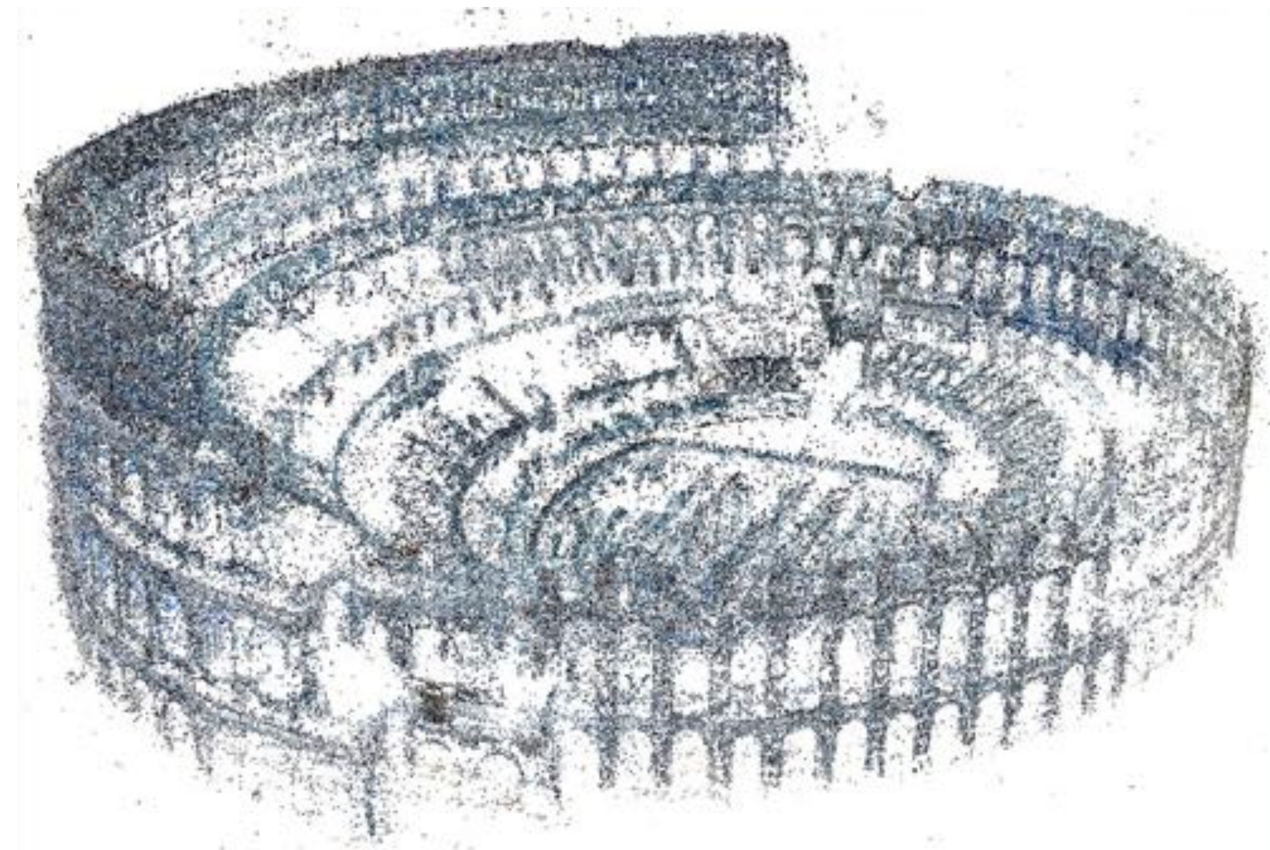


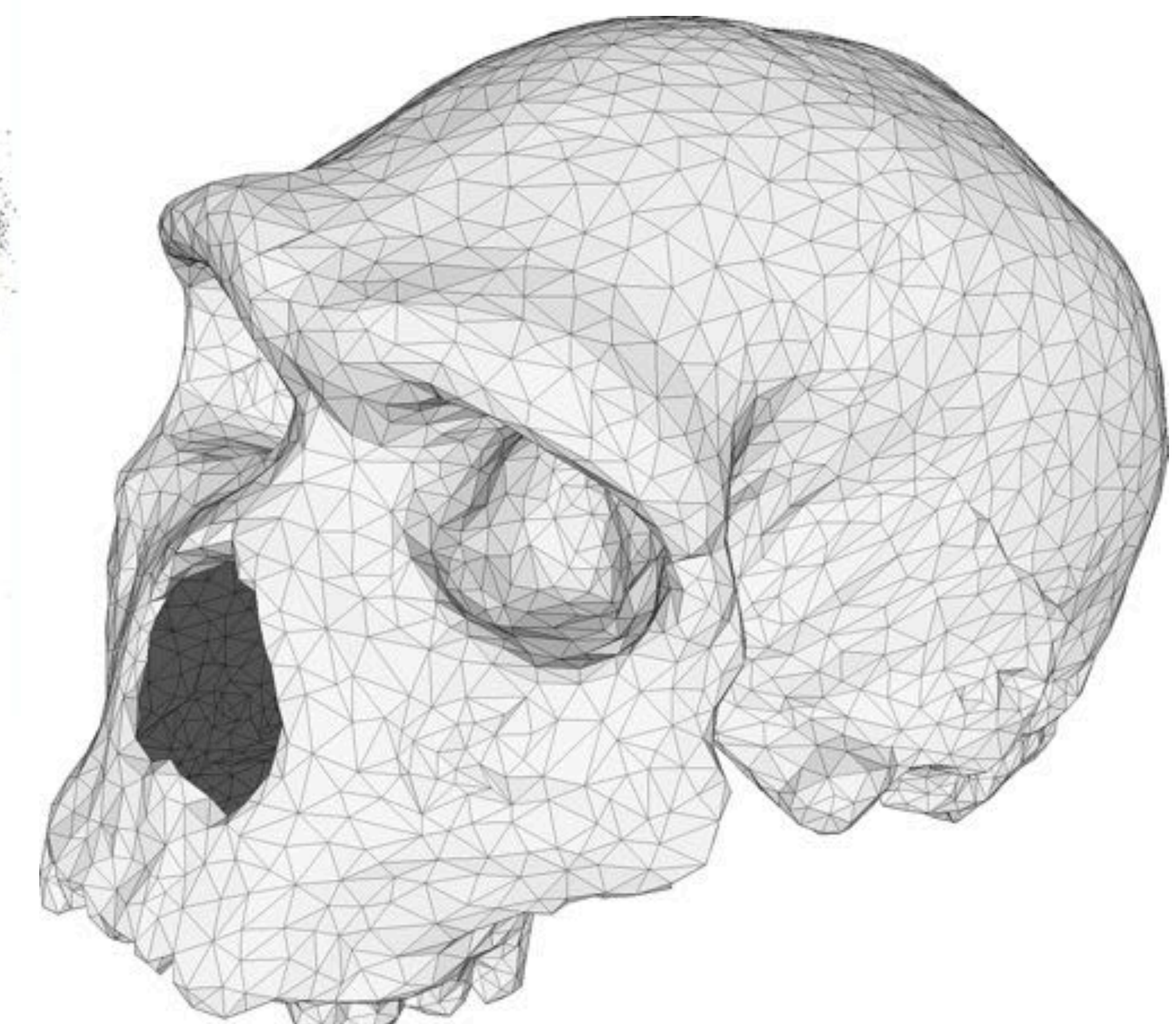
- **(Might have a bunch of these maps, e.g., one per triangle!)**

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:

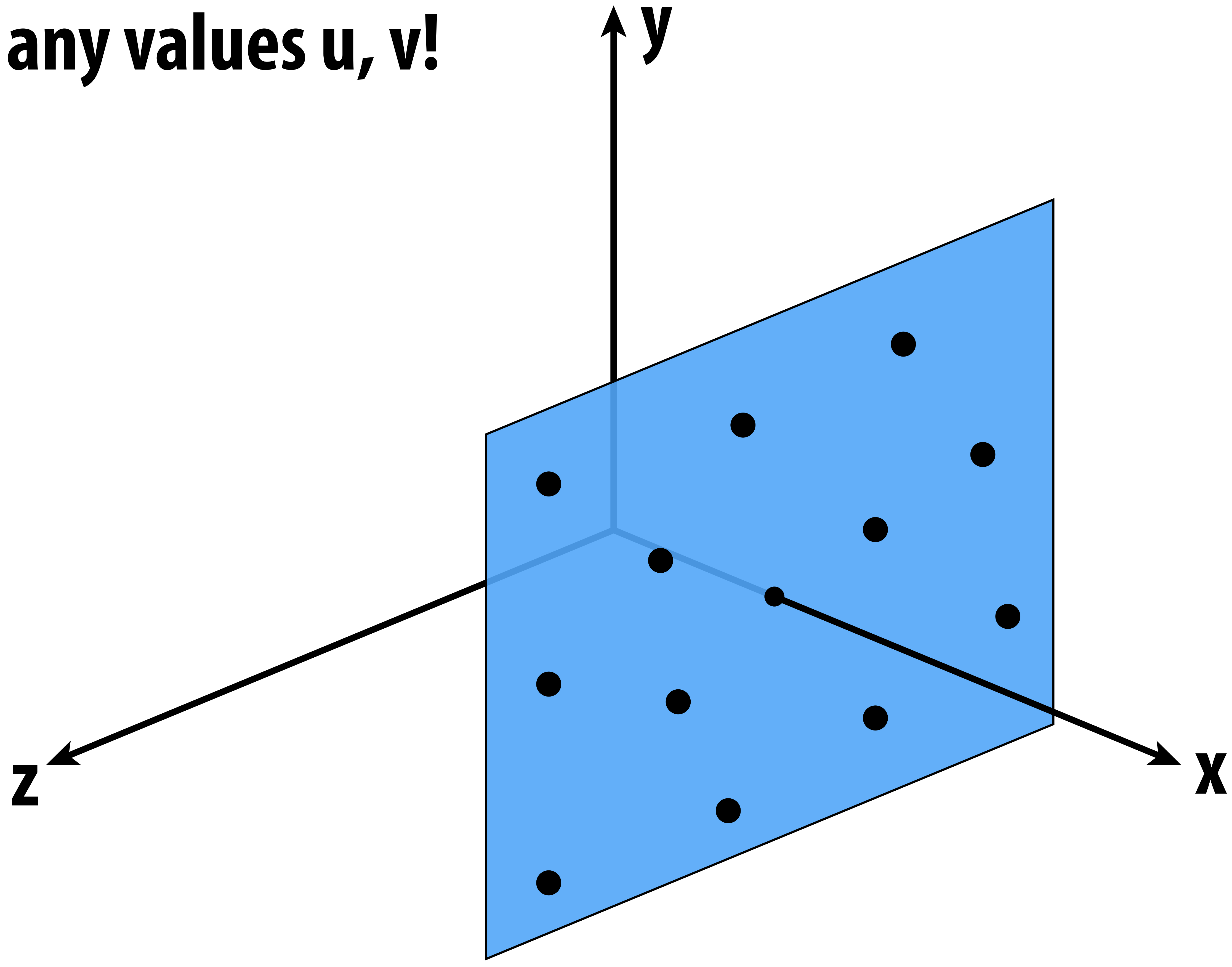
I'll give you an explicit surface.

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).

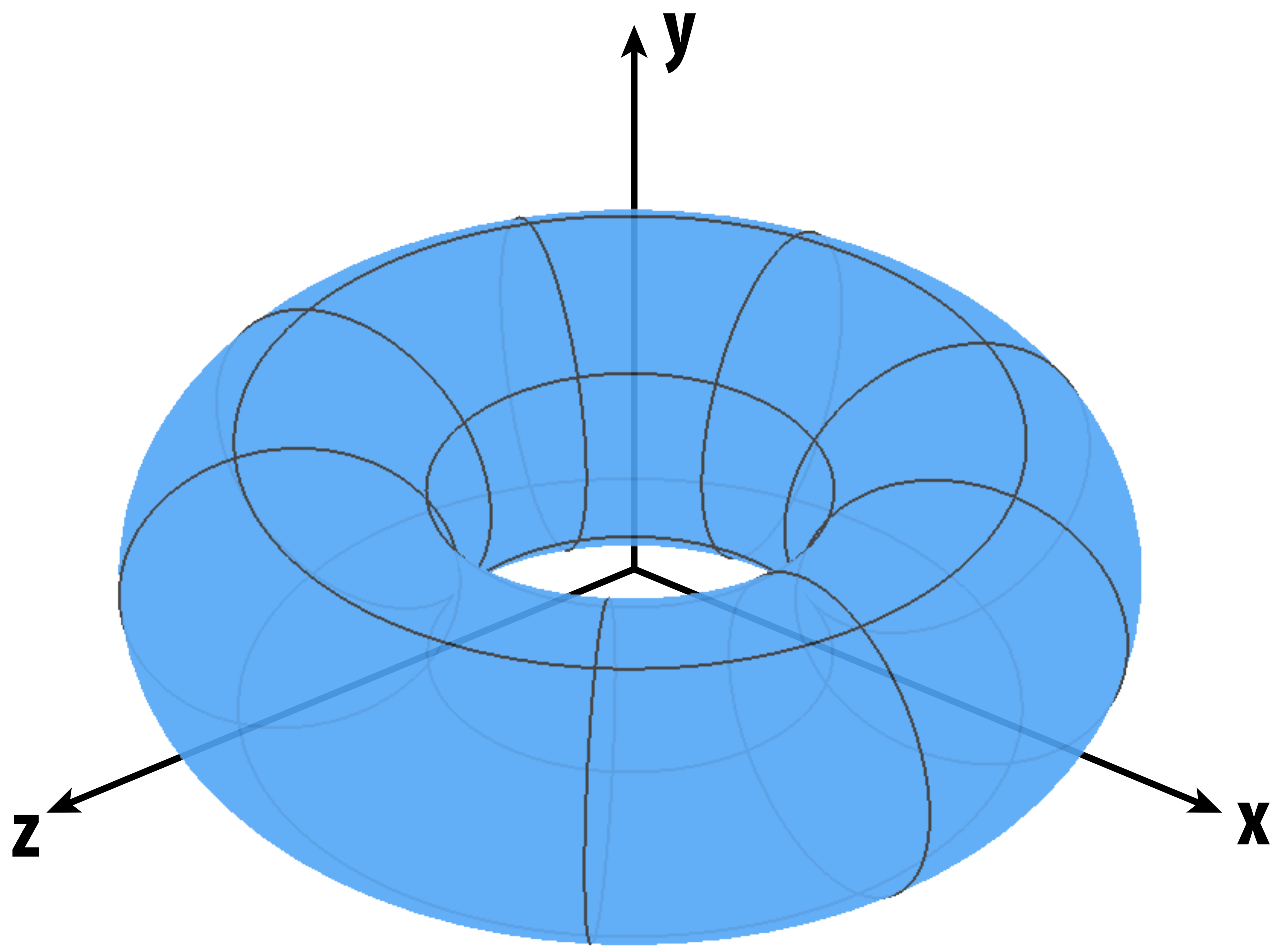
I want to see if a point is inside it.

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 (1.96, -0.39, 0.9)?

...NO!



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

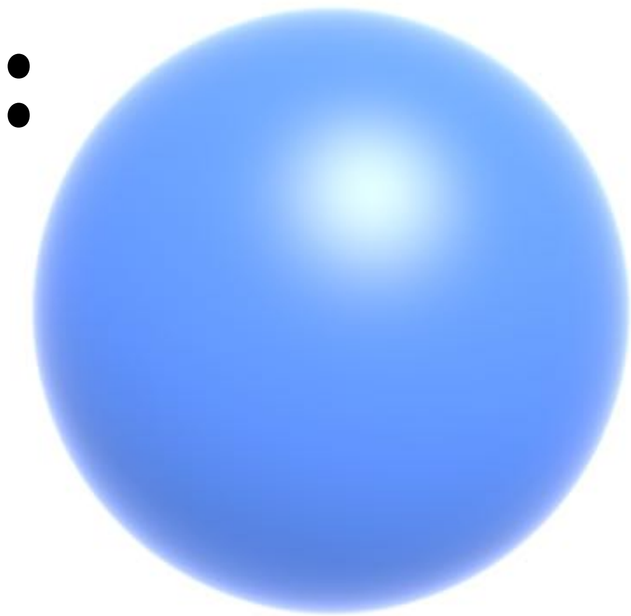
CONCLUSION:
Some representations work better than others—depends on the task!

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.

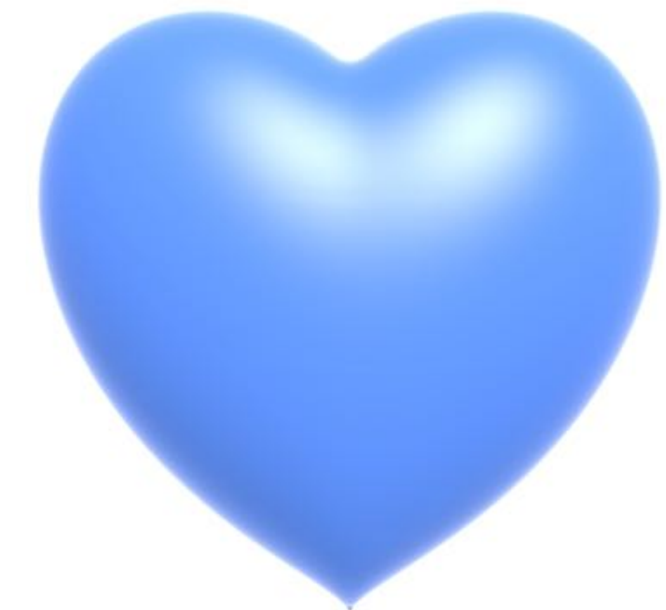
Algebraic Surfaces (Implicit)

- **Surface is zero set of a polynomial in x, y, z**
- **Examples:**



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

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



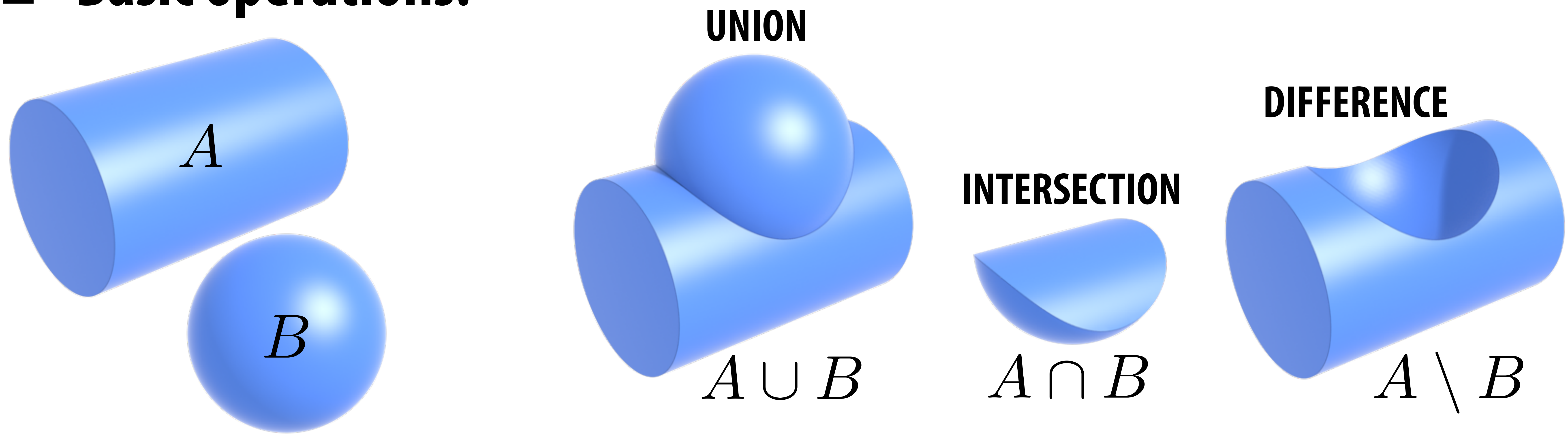
$$(x^2+\frac{9y^2}{4}+z^2-1)^3 = x^2z^3+\frac{9y^2z^3}{80}$$

- **What about more complicated shapes?**

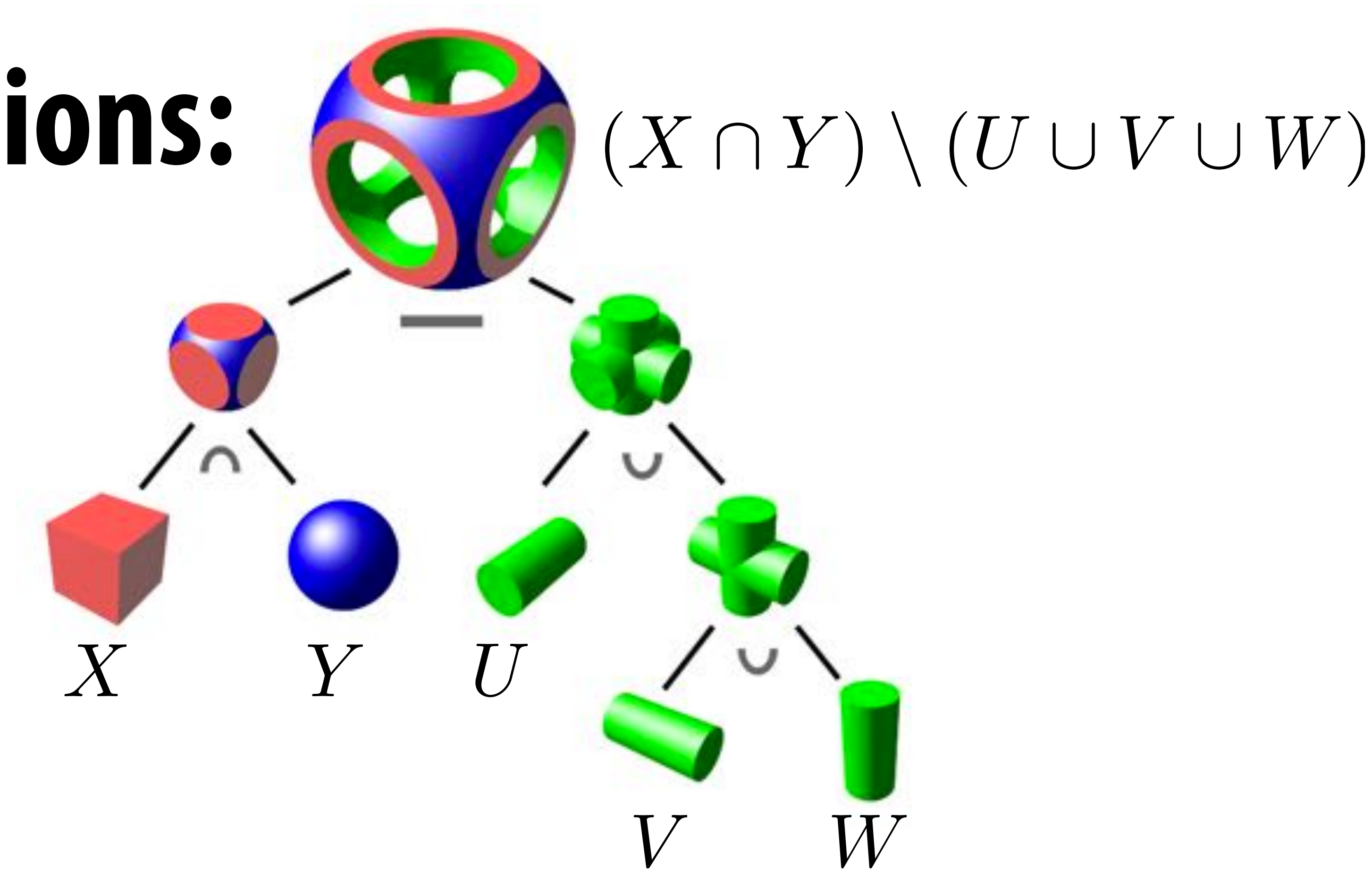
- **Very hard to come up with polynomials!**

Constructive Solid Geometry (Implicit)

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

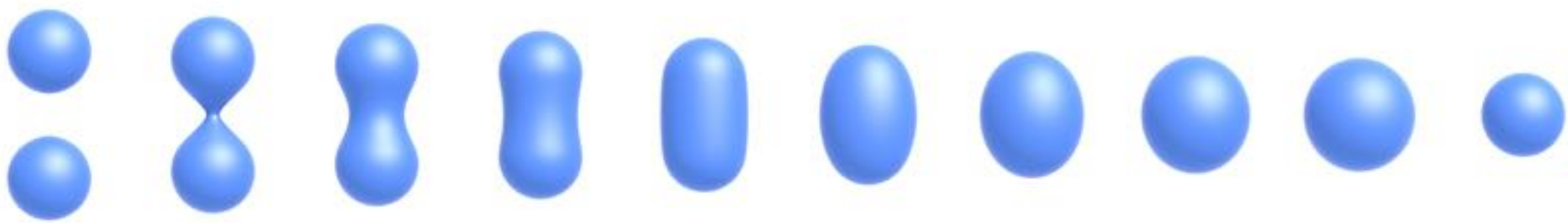


- Then chain together expressions:

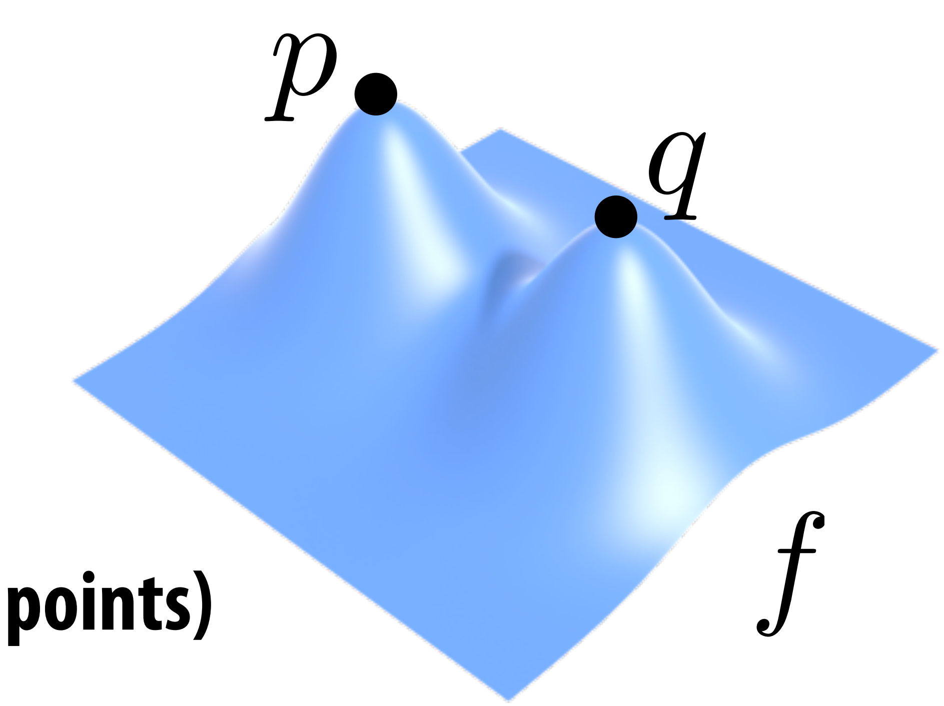


Blobby Surfaces (Implicit)

- Instead of Booleans, gradually blend surfaces together:

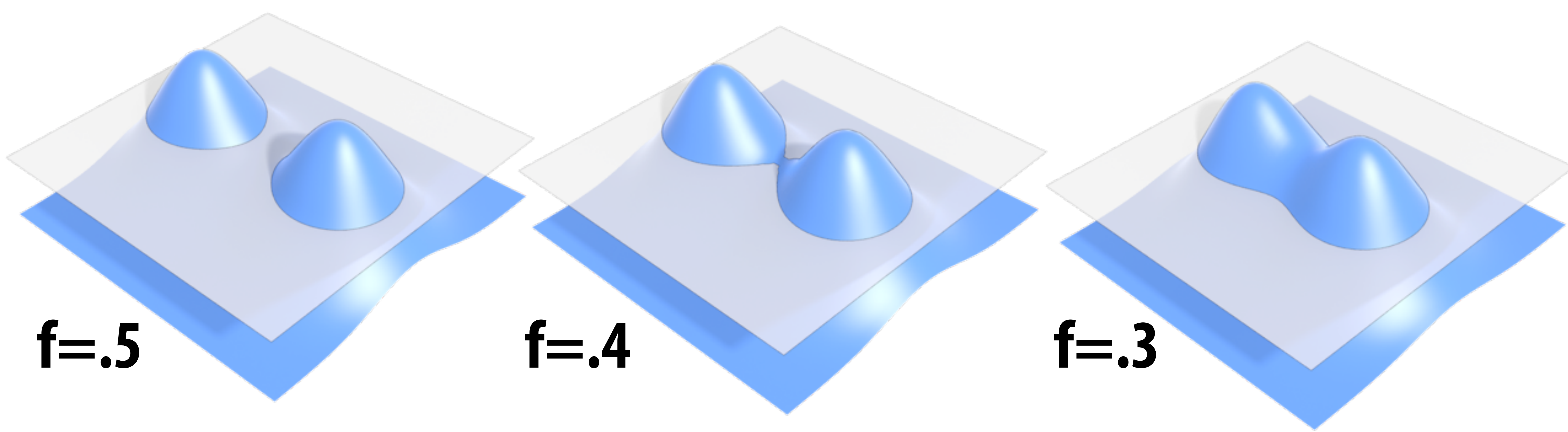


- Easier to understand in 2D:



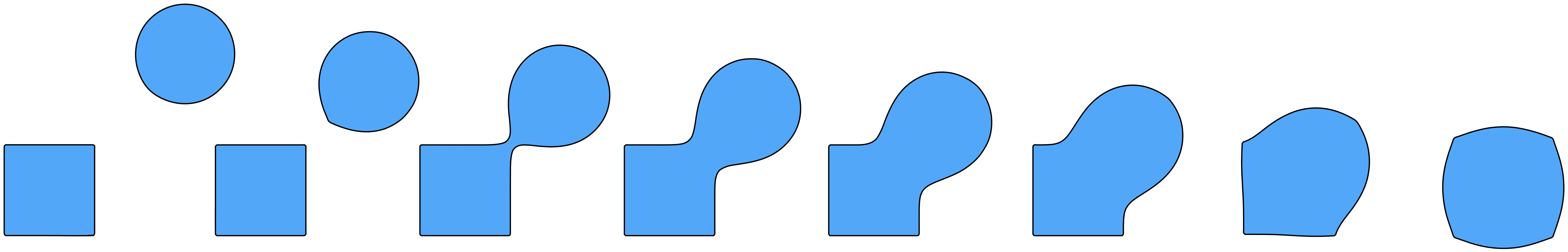
$\phi_p(x) := e^{-|x-p|^2}$ (Gaussian centered at p)

$f := \phi_p + \phi_q$ (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:**

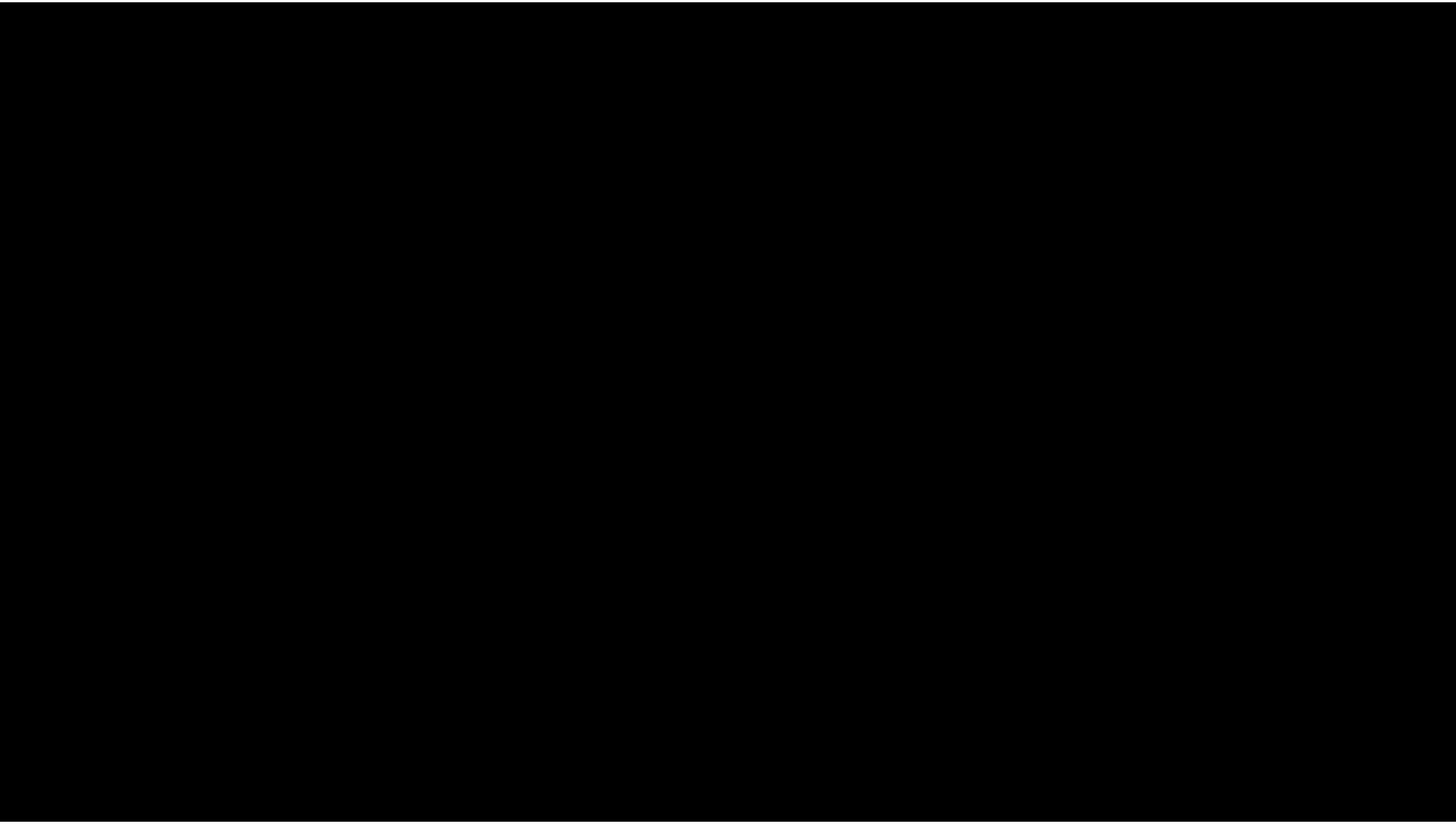


- **Similar strategy to points, though many possibilities. E.g.,**

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

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

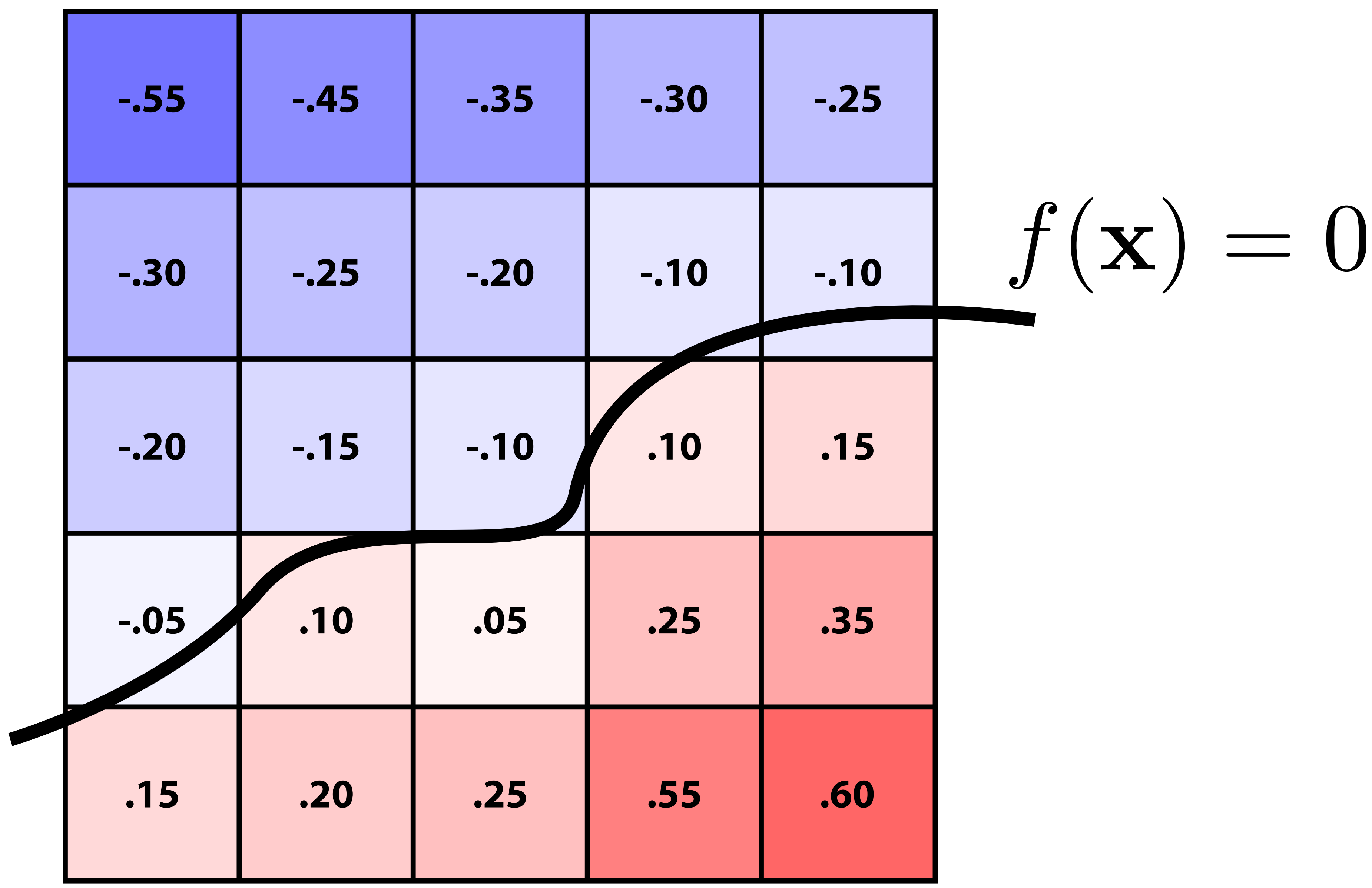
Scene of pure distance functions (not easy!)



see http://iquilezles.org/

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**



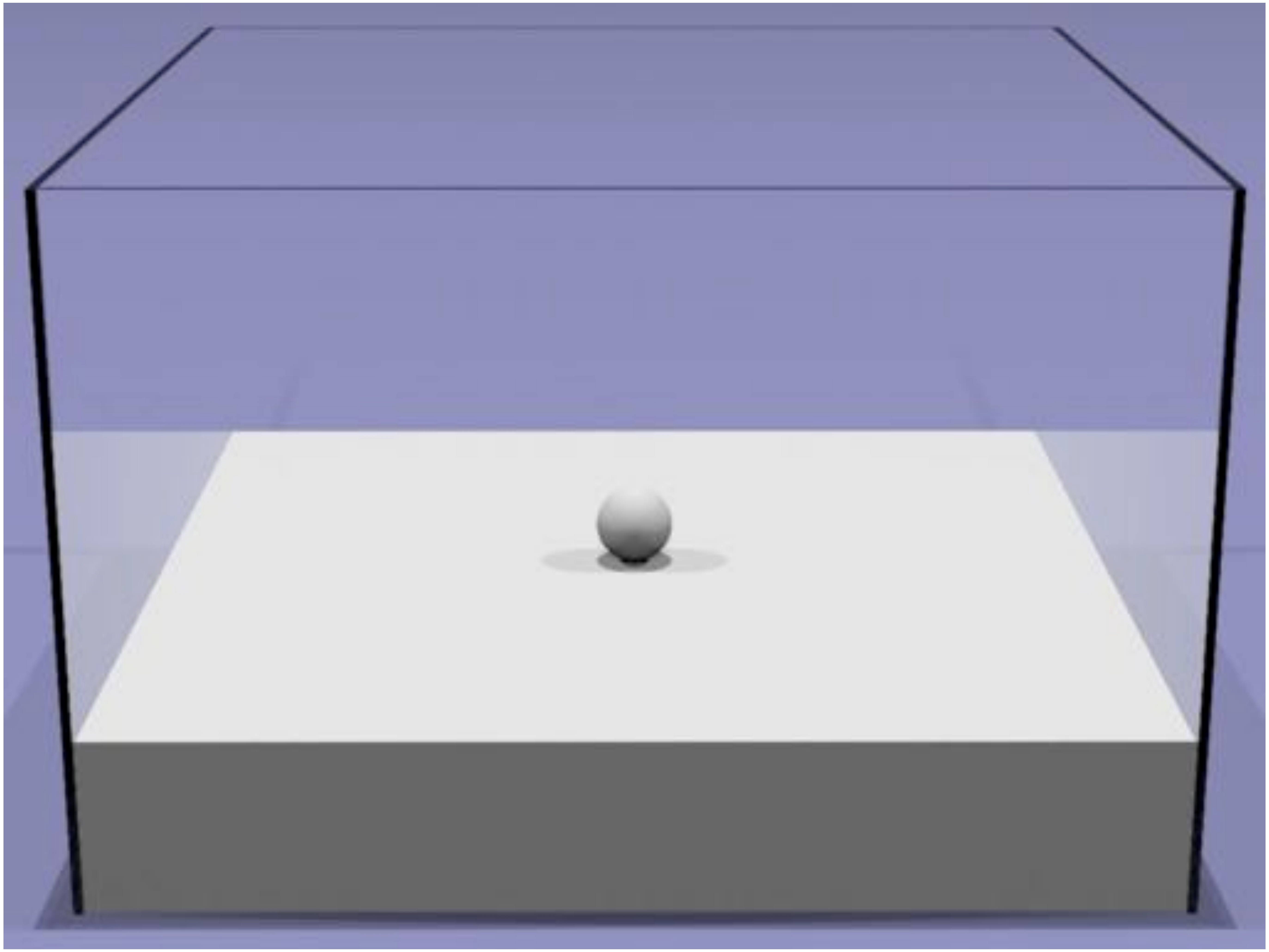
- **Surface is found where interpolated values equal zero**
- **Provides much more explicit control over shape (like a texture)**
- **Unlike closed-form expressions, run into problems of <u>aliasing</u>!**

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

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

Level Sets in Physical Simulation

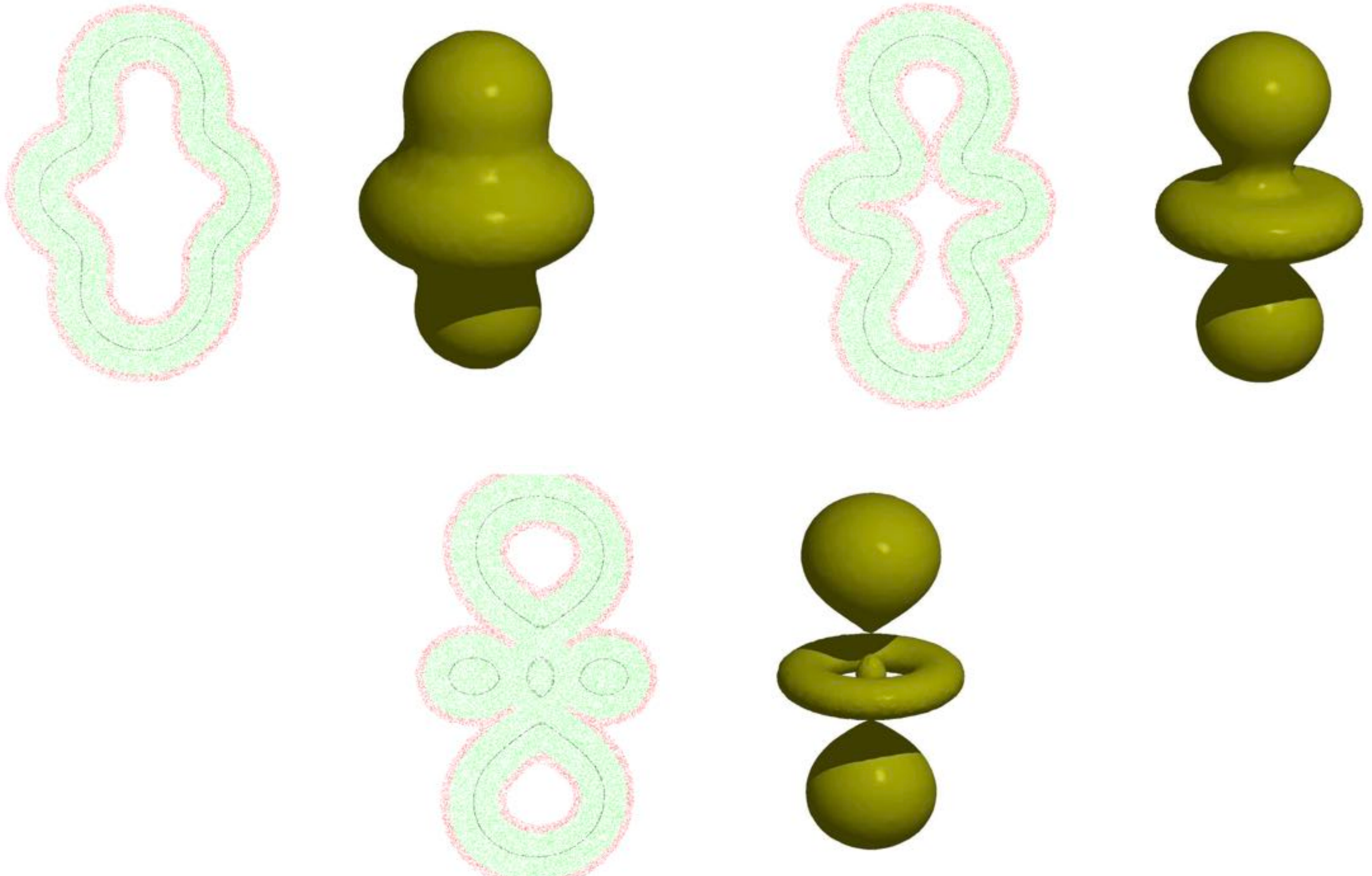
Level set encodes distance to air-liquid boundary:



see http://physbam.stanford.edu

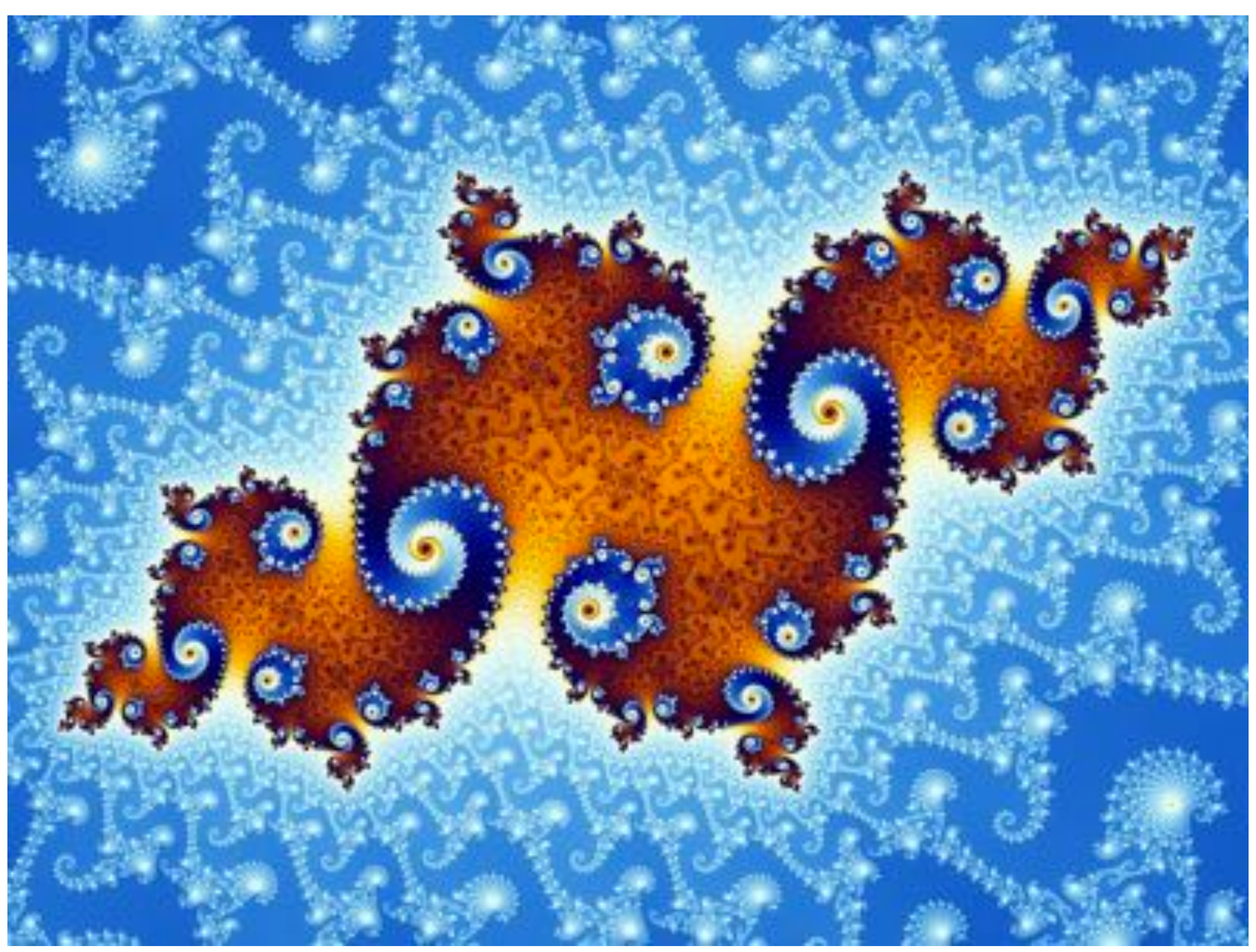
Level Set Storage

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



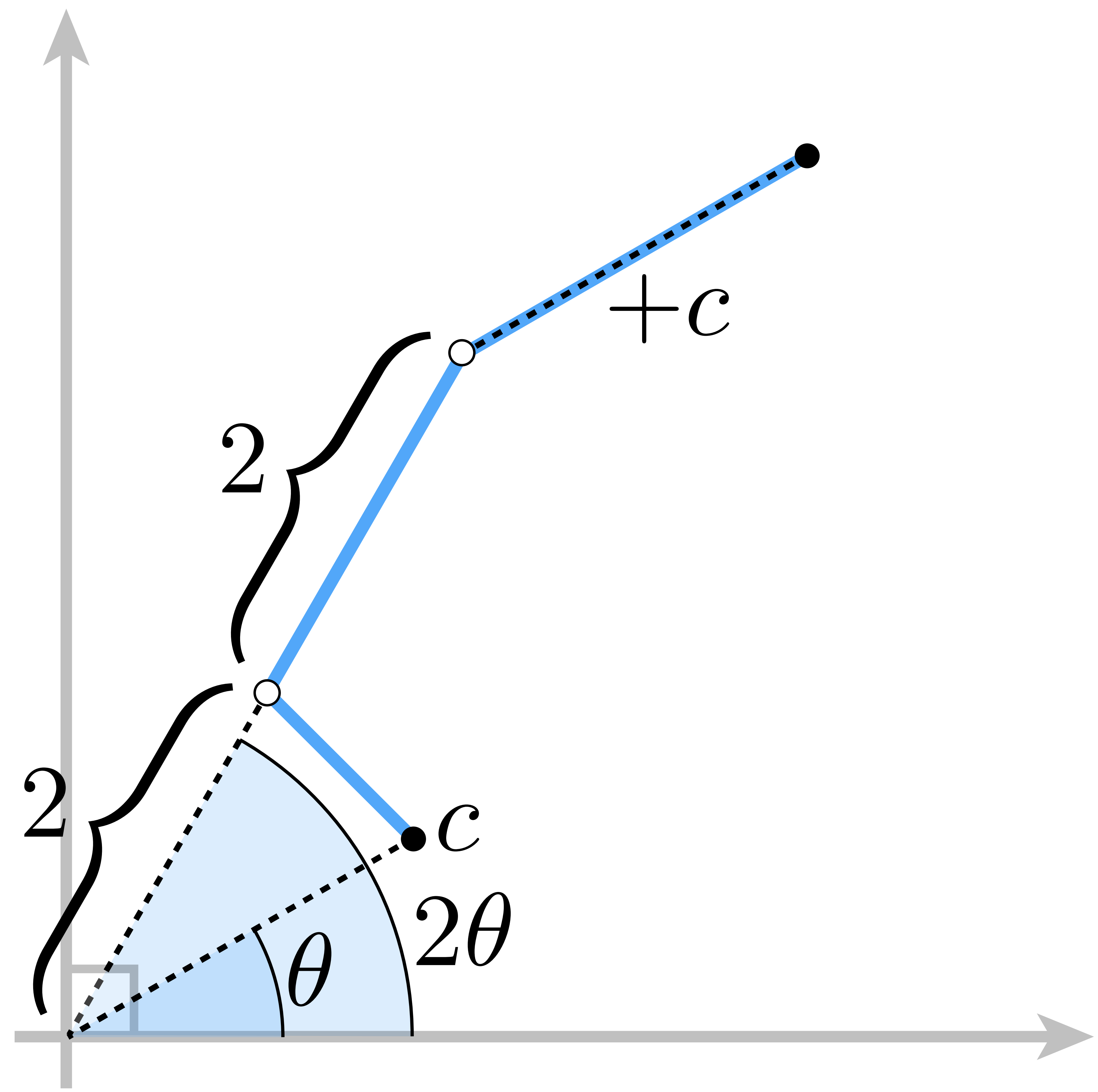
Fractals (Implicit)

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



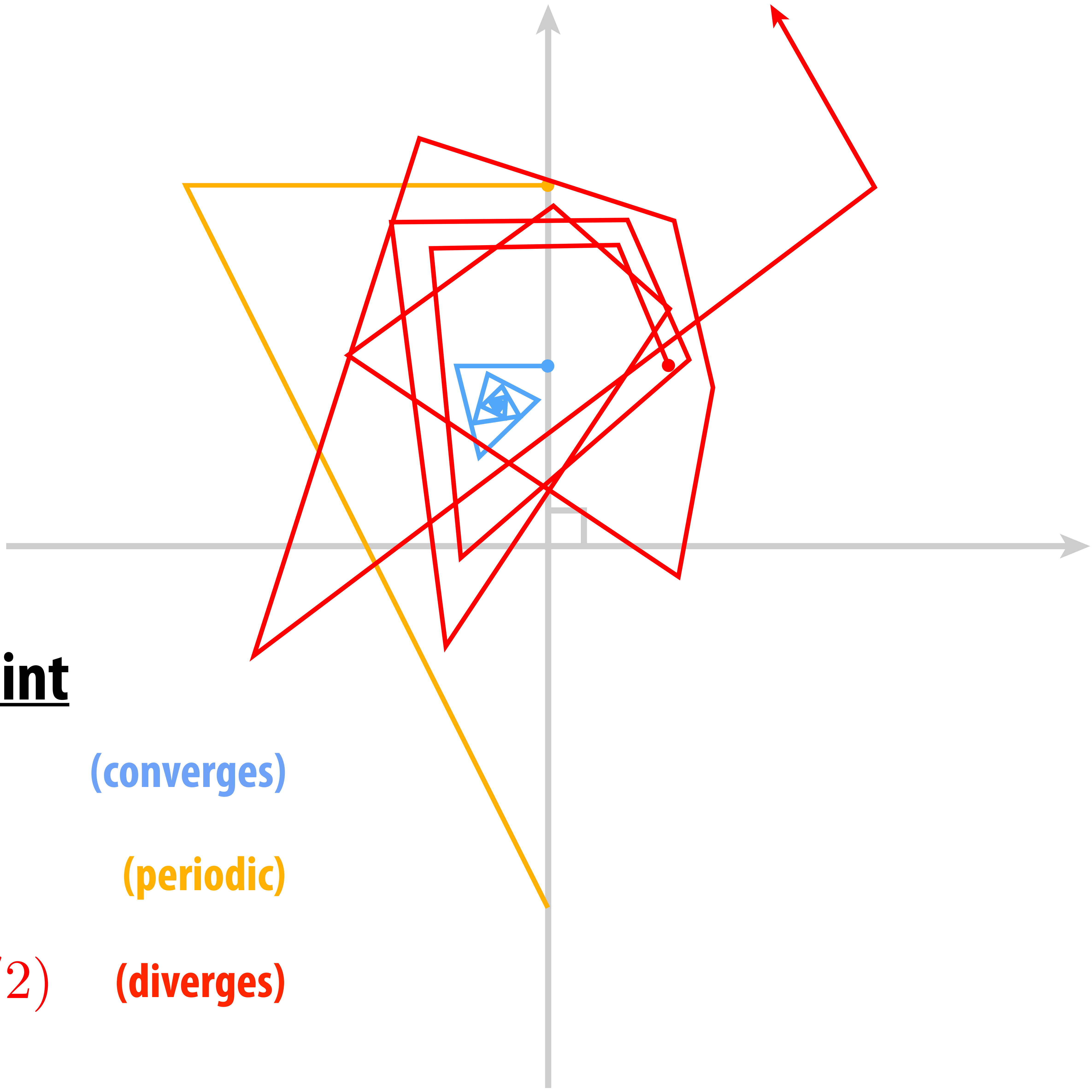
Mandelbrot Set - Definition

- **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 magnitude remains bounded (never goes to ∞), it's in the Mandelbrot set.

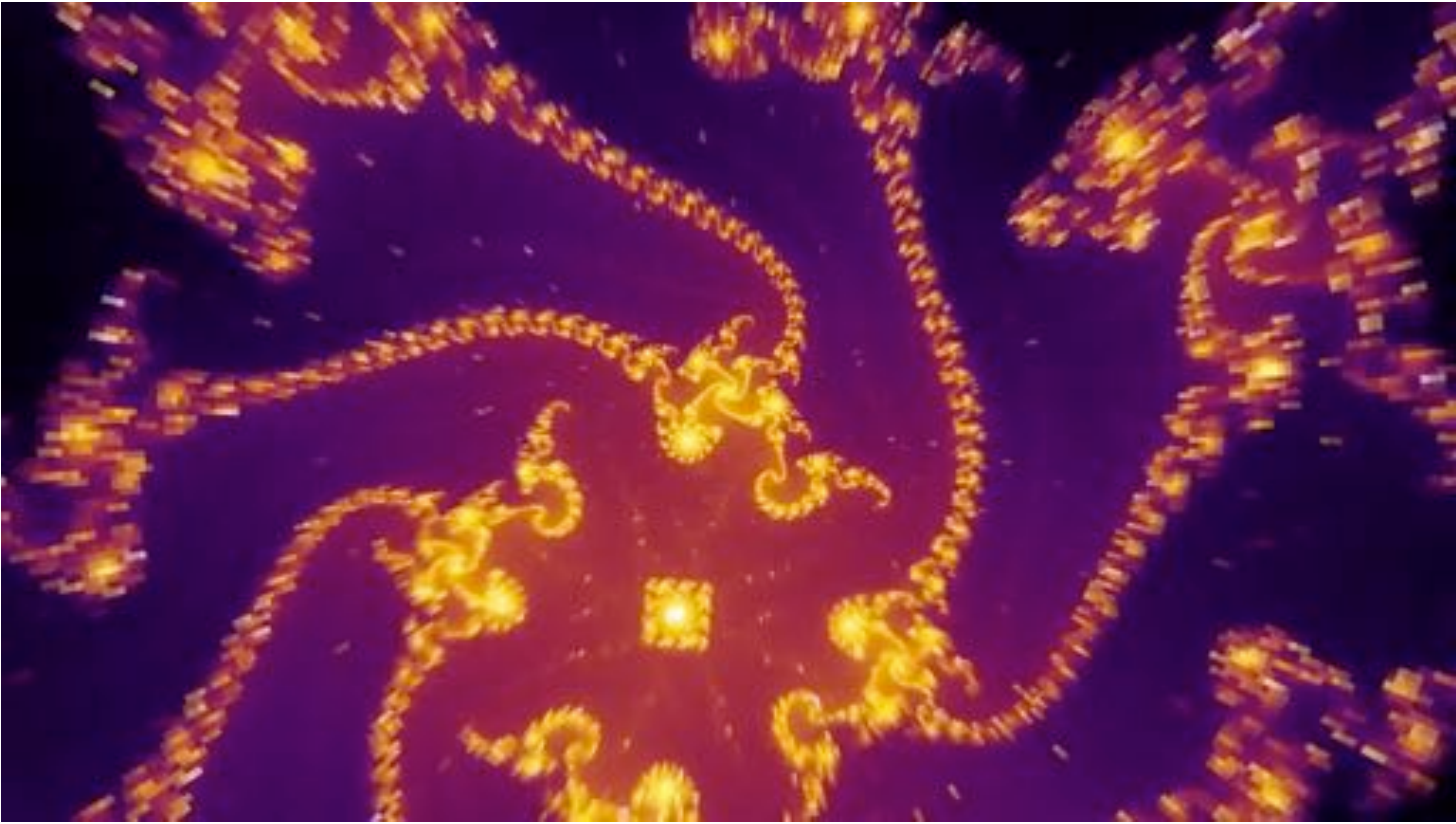
Mandelbrot Set - Examples



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 alumn) - see http://electricsheep.org

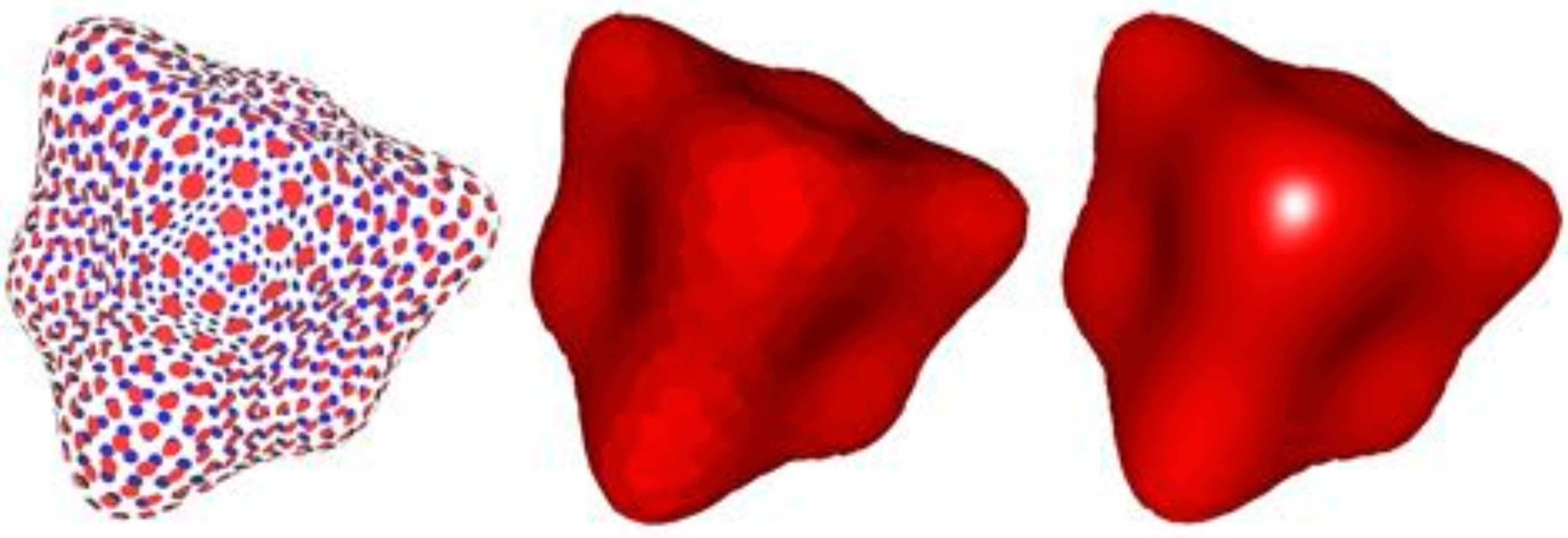
Implicit Representations - Pros & Cons

- **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?

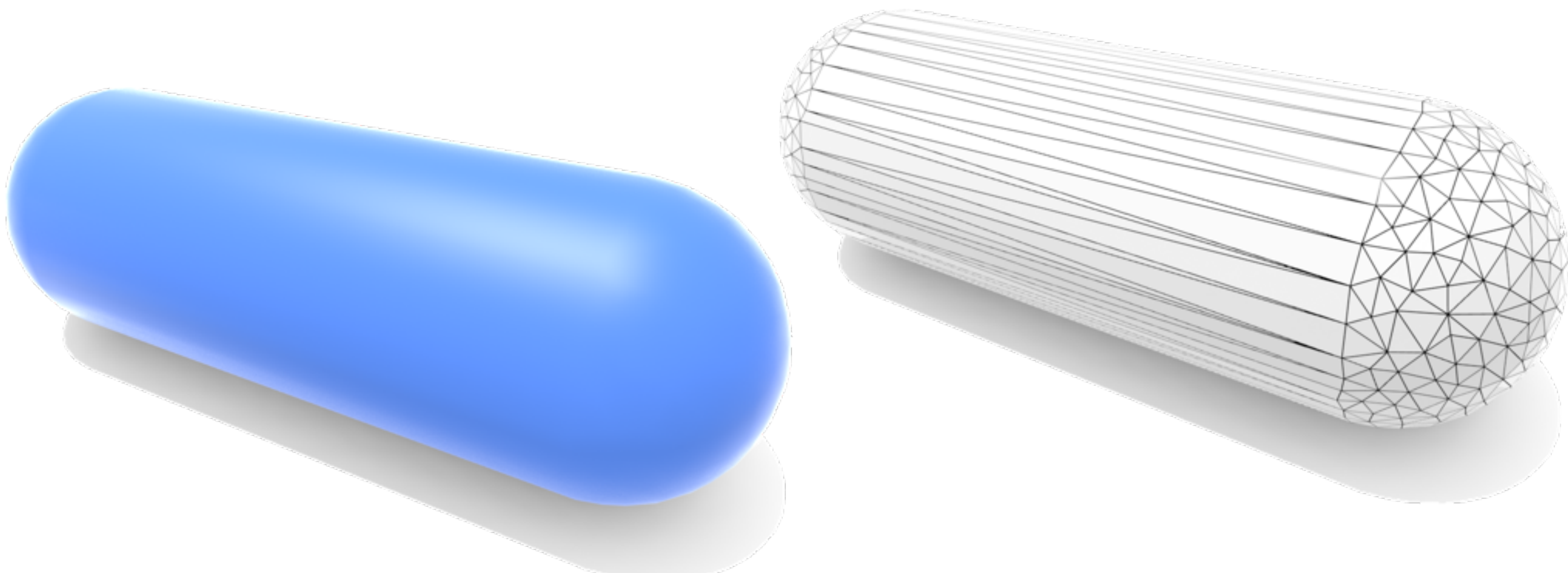
Point Cloud (Explicit)

- Easiest representation: list of points (x,y,z)
- Often augmented with normals
- Easily represent any kind of geometry
- Easy to draw dense cloud (>>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
- Irregular neighborhoods

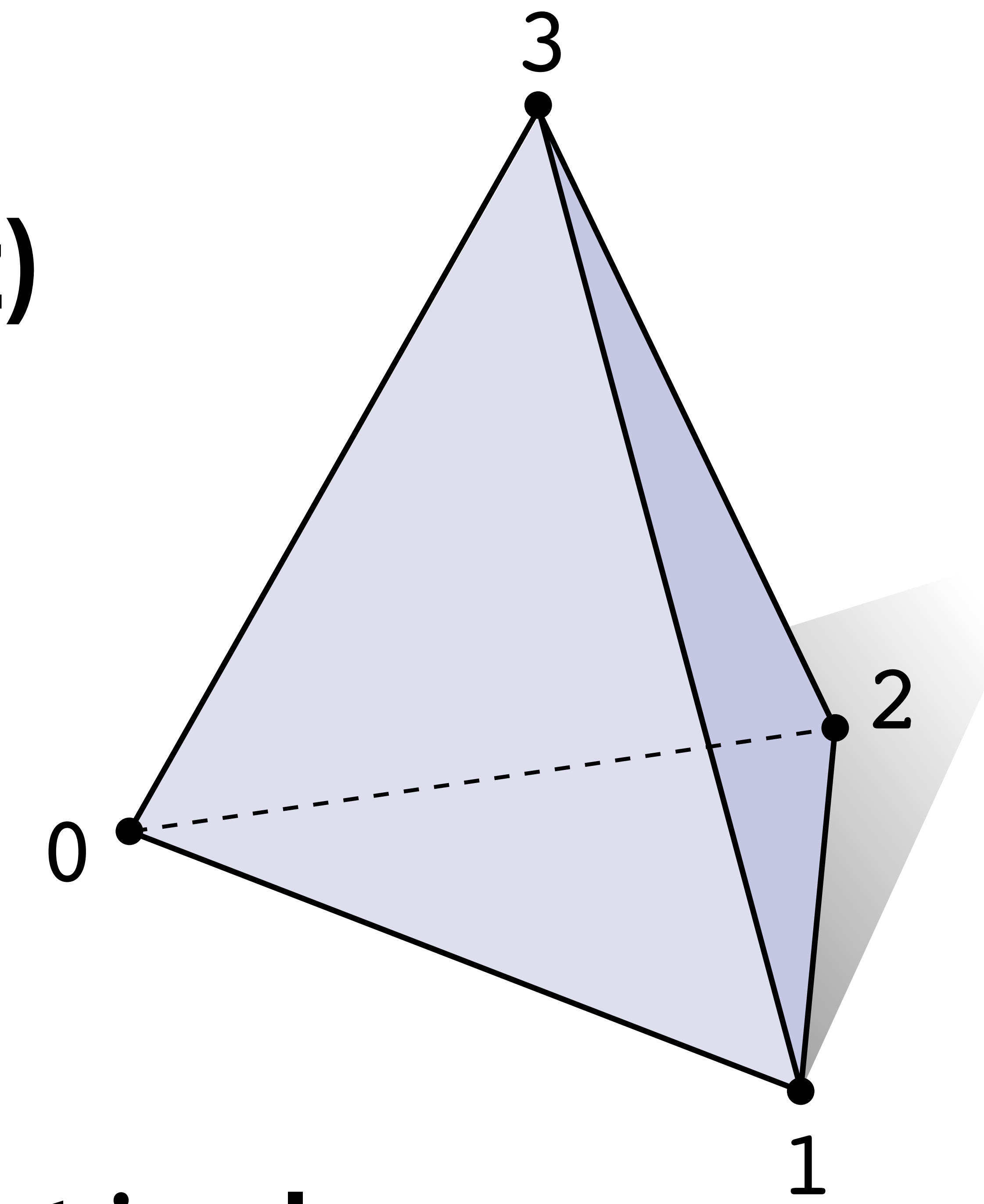


(Much more about polygon meshes in upcoming lectures!)

Triangle Mesh (Explicit)

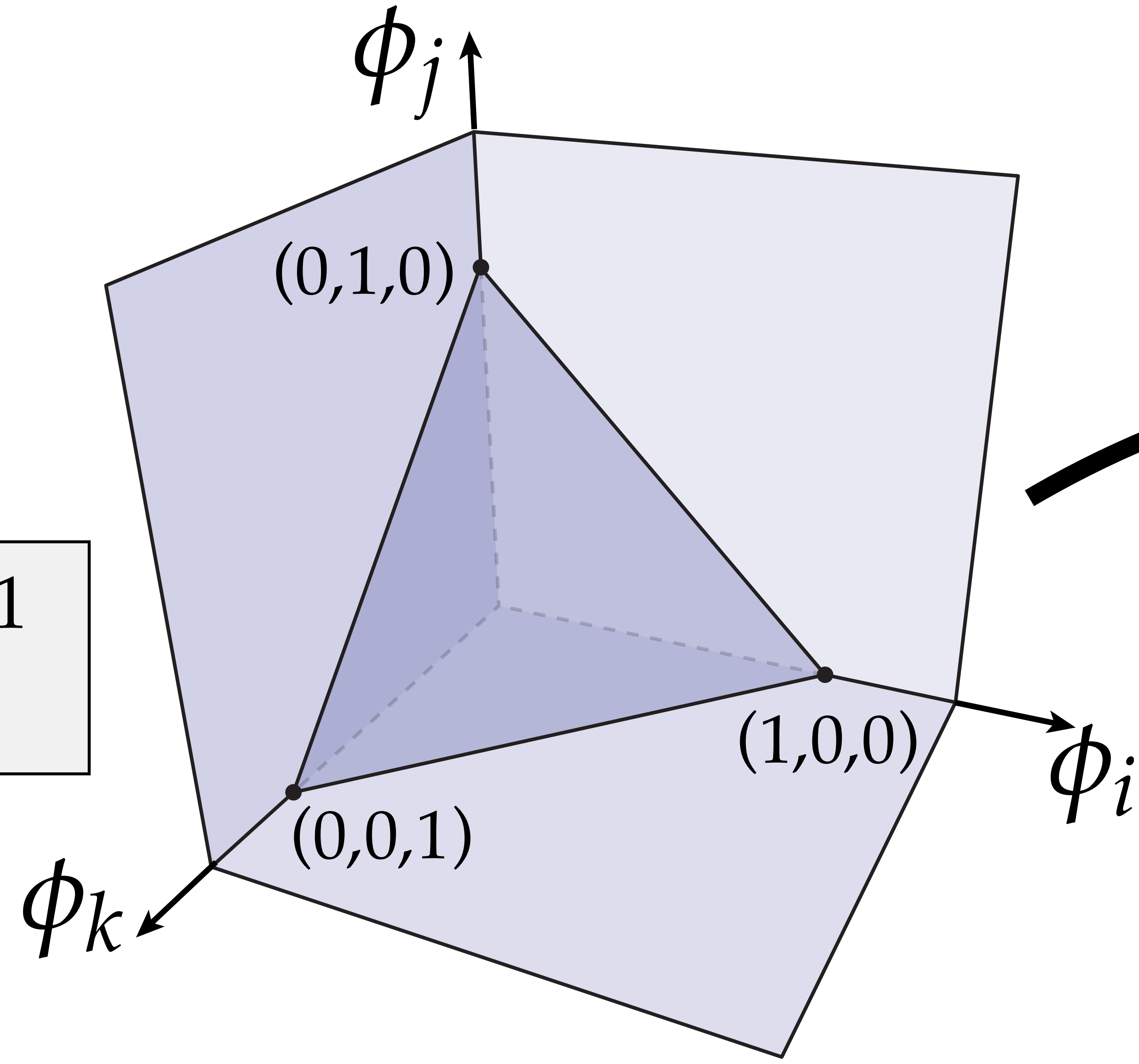
- Store vertices as triples of coordinates (x,y,z)
- Store triangles as triples of indices (i,j,k)
- E.g., tetrahedron:

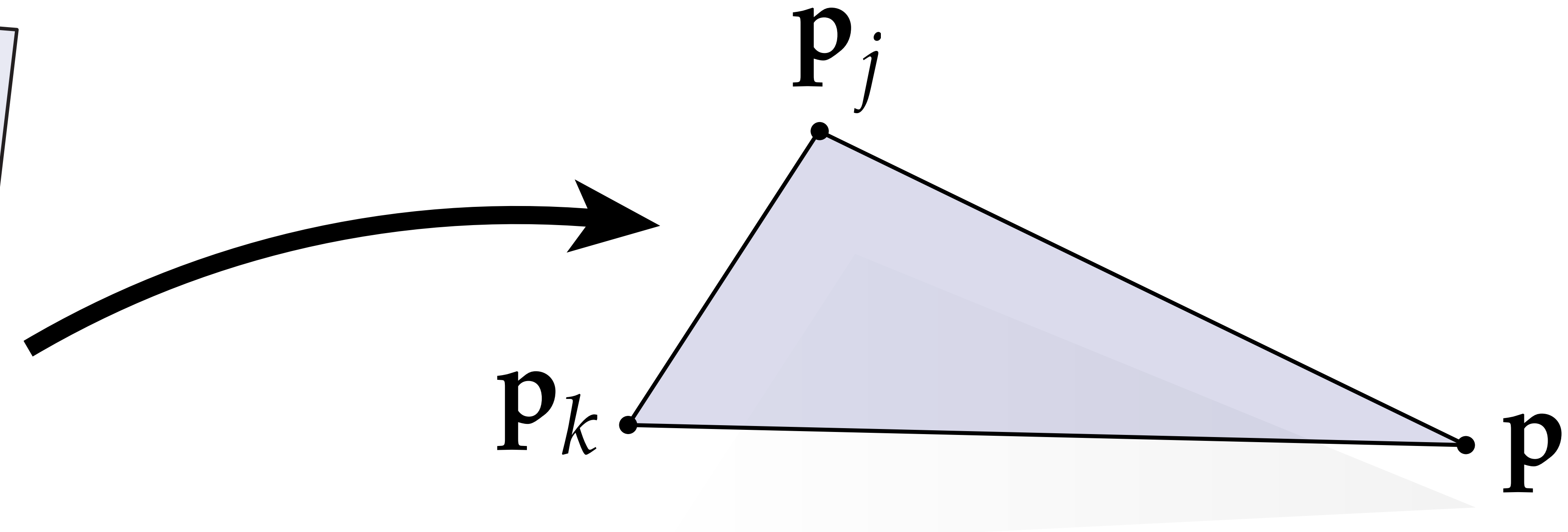
VERTICES	x	y	z	TRIANGLES	i	j	k
0:	−1	−1	−1		0	2	1
1:	1	−1	1		0	3	2
2:	1	1	−1		3	0	1
3:	−1	1	1		3	1	2



- Use barycentric interpolation to define points inside triangles:

$$\phi_i + \phi_j + \phi_k = 1$$
$$\phi_i, \phi_j, \phi_k > 0$$





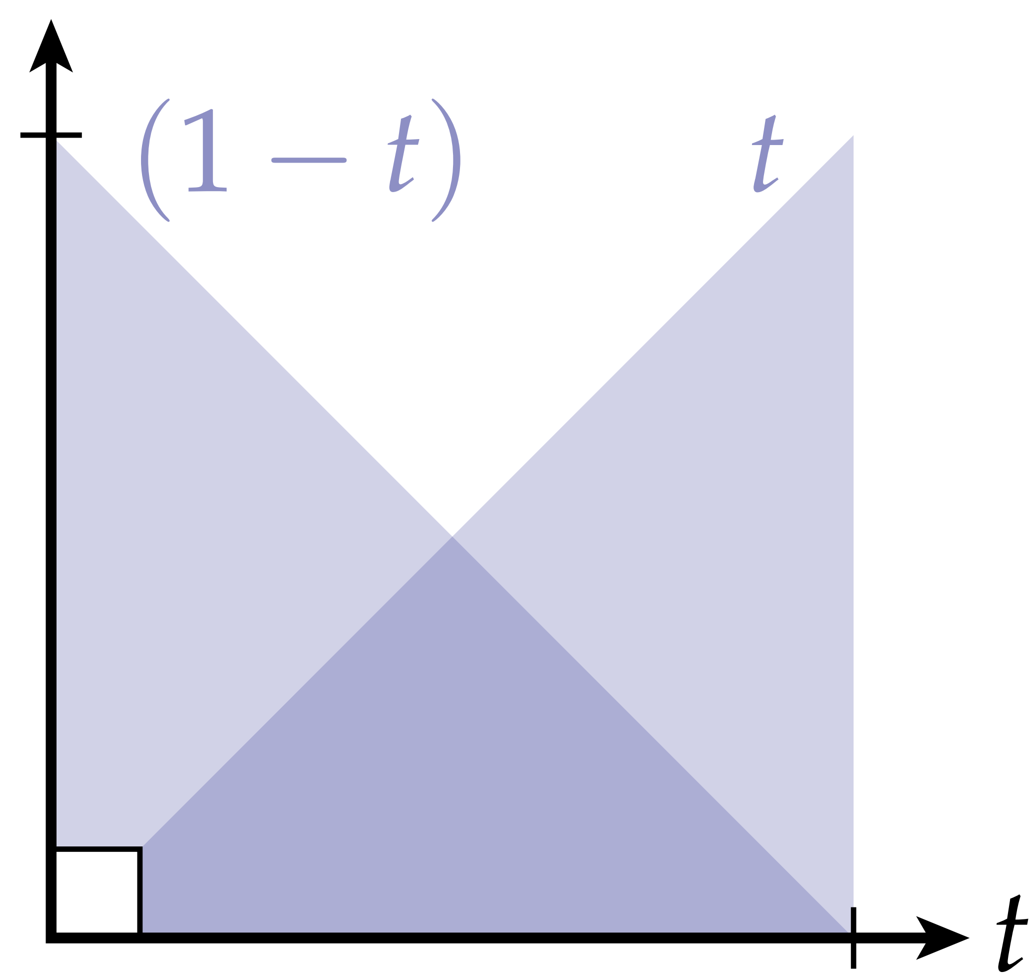
$$p = \phi_i \mathbf{p}_i + \phi_j \mathbf{p}_j + \phi_k \mathbf{p}_k$$

Recall: Linear Interpolation (1D)

- **Interpolate values using linear interpolation; in 1D:**

$$\hat{f}(t) = (1-t)f_i + tf_j$$

- **Can think of this as a linear combination of two functions:**



- **Why limit ourselves to <u>linear</u> basis functions?**
- **Can we get more interesting geometry with other bases?**

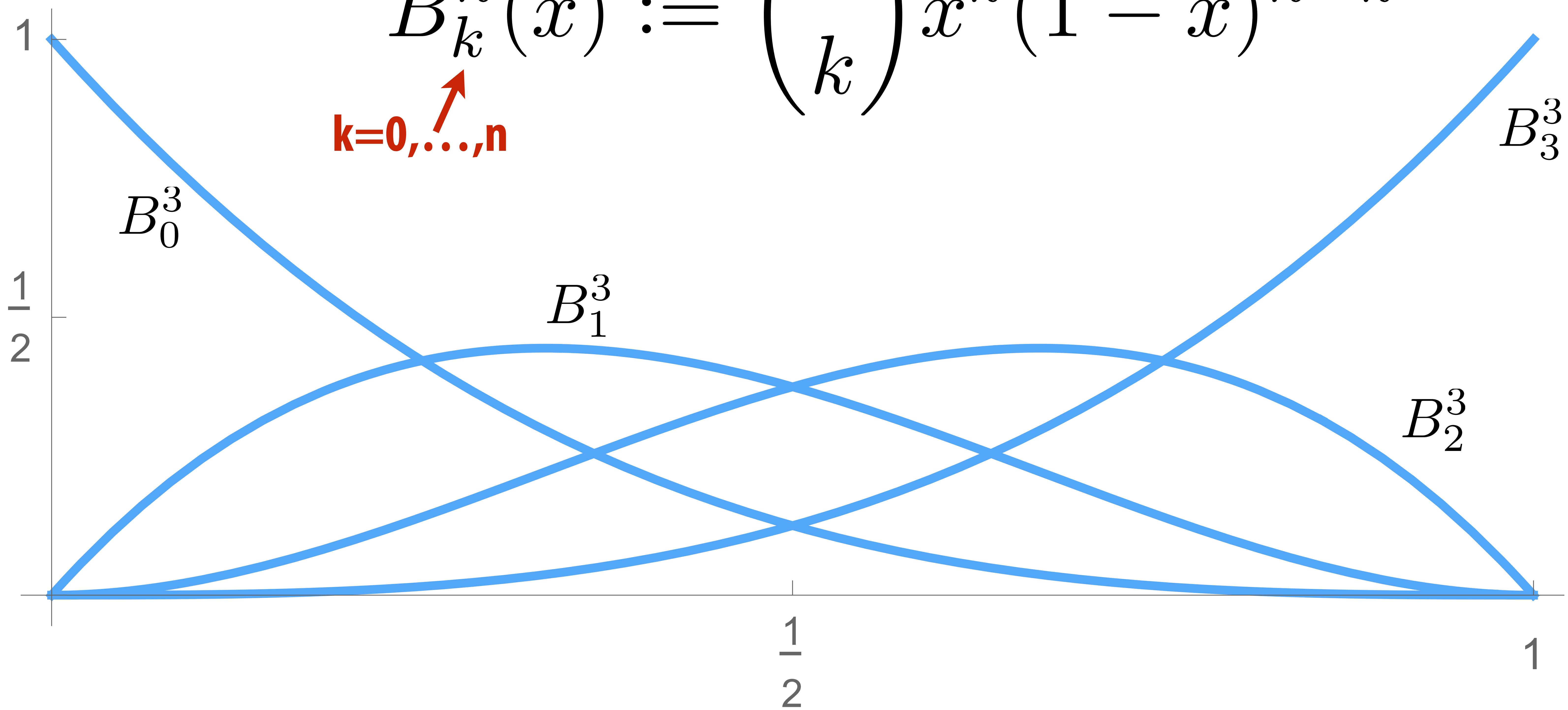
Bernstein Basis

- Linear interpolation essentially uses 1st-order polynomials
- Provide more flexibility by using higher-order polynomials
- Instead of usual basis (1, x, x², x³, ...), use Bernstein basis:

degree
0≤x≤1
"n choose k"

$$B_k^n(x) := \binom{n}{k} x^k (1-x)^{n-k}$$

k=0,…,n



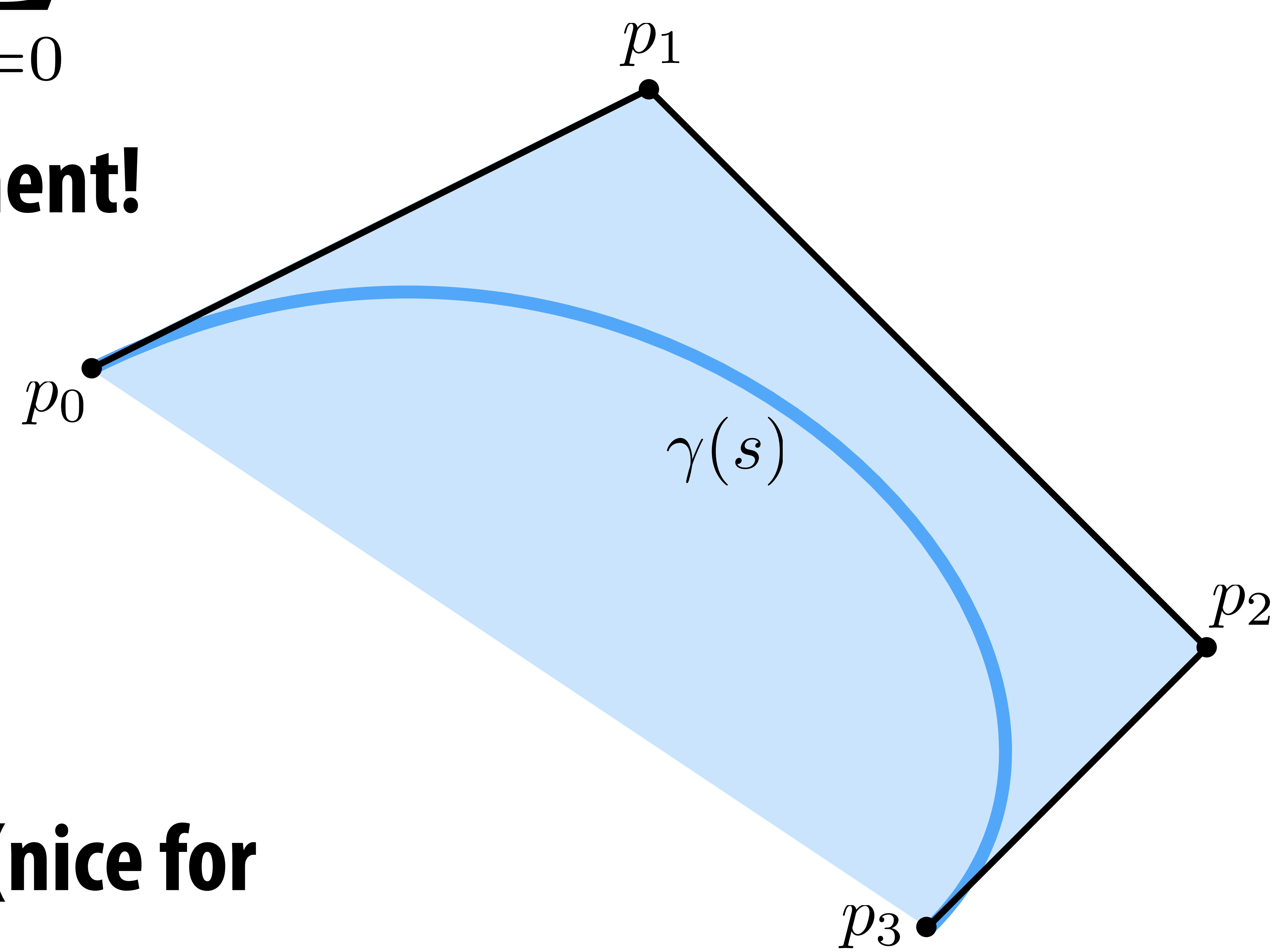
Bézier Curves (Explicit)

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

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

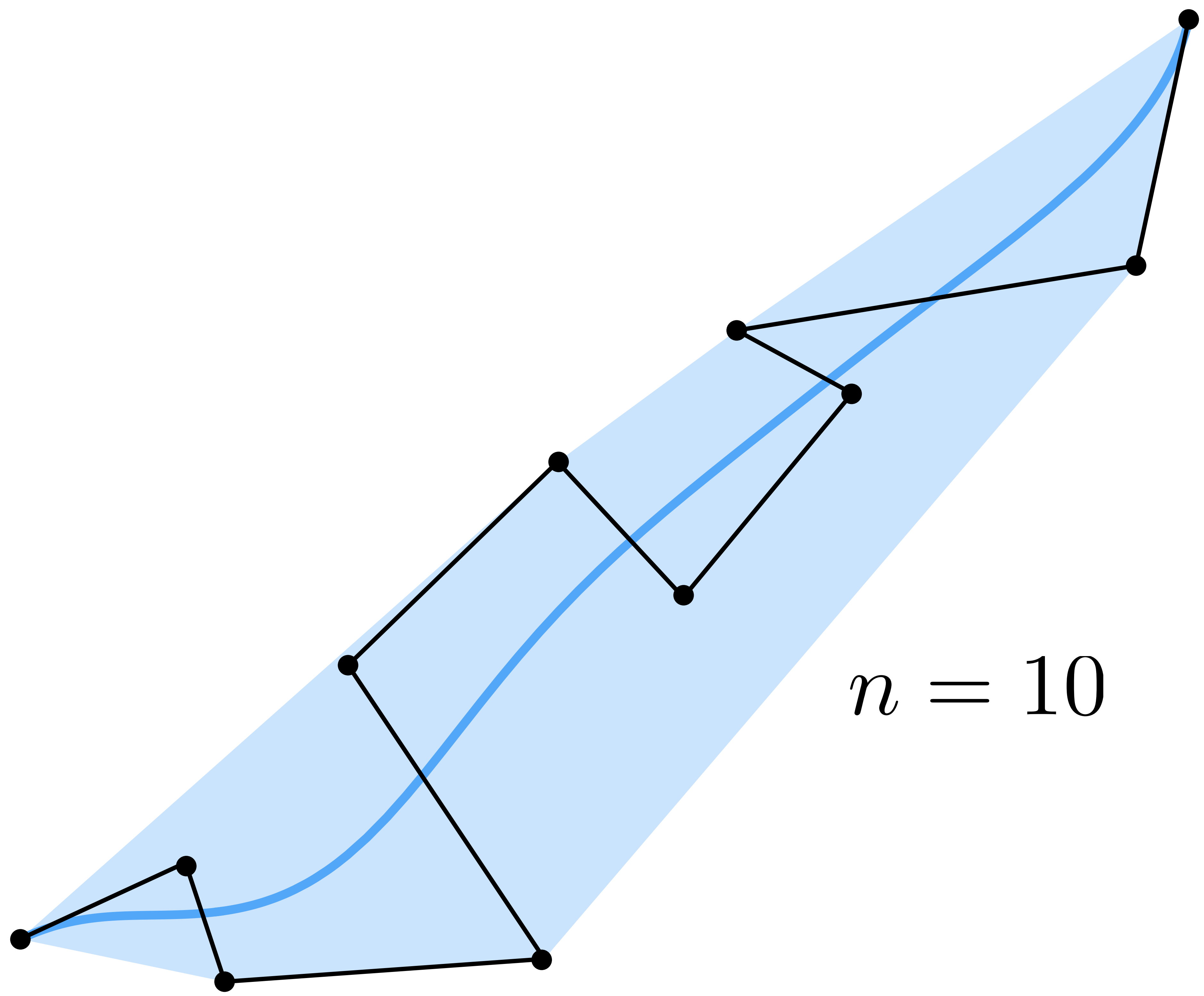
control points

- 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)



Just keep going…?

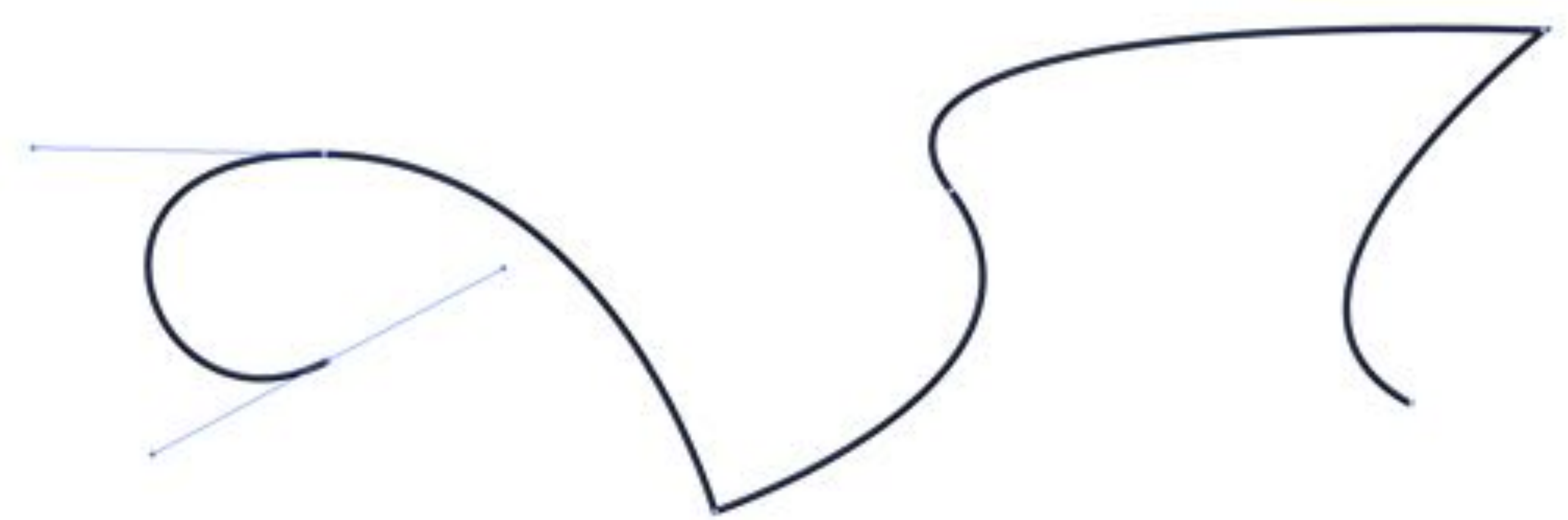
- **What if we want an even more interesting curve?**
- **High-degree Bernstein polynomials don't interpolate well:**



Very hard to control!

Piecewise Bézier Curves (Explicit)

- Alternative idea: piece together many Bézier curves
- Widely-used technique (Illustrator, fonts, SVG, etc.)



- Formally, piecewise Bézier curve:

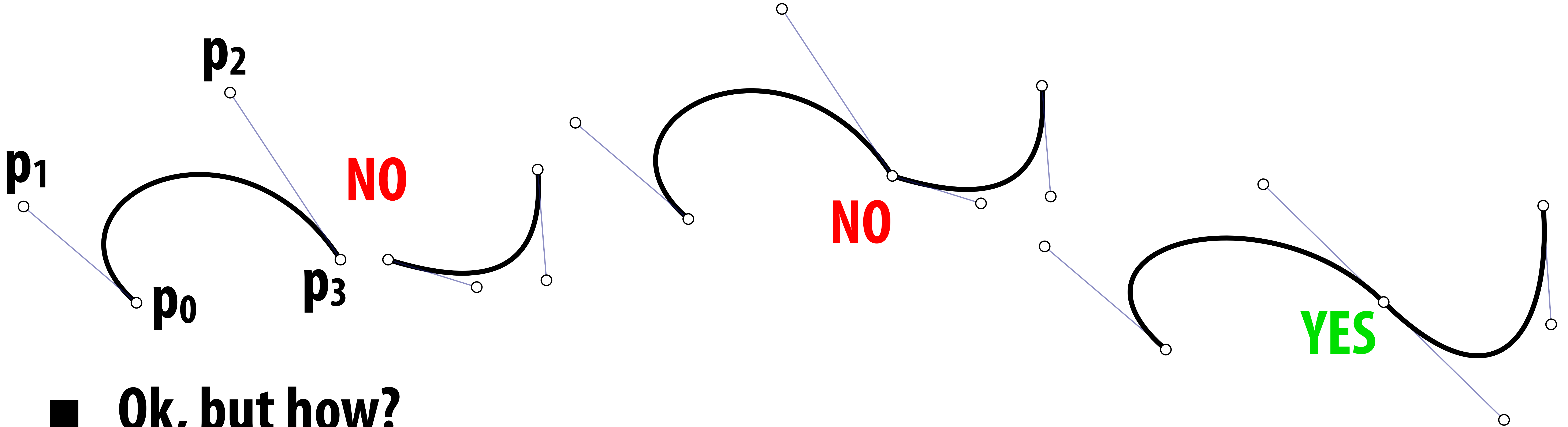
piecewise Bézier

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

single Bézier

Bézier Curves — tangent continuity

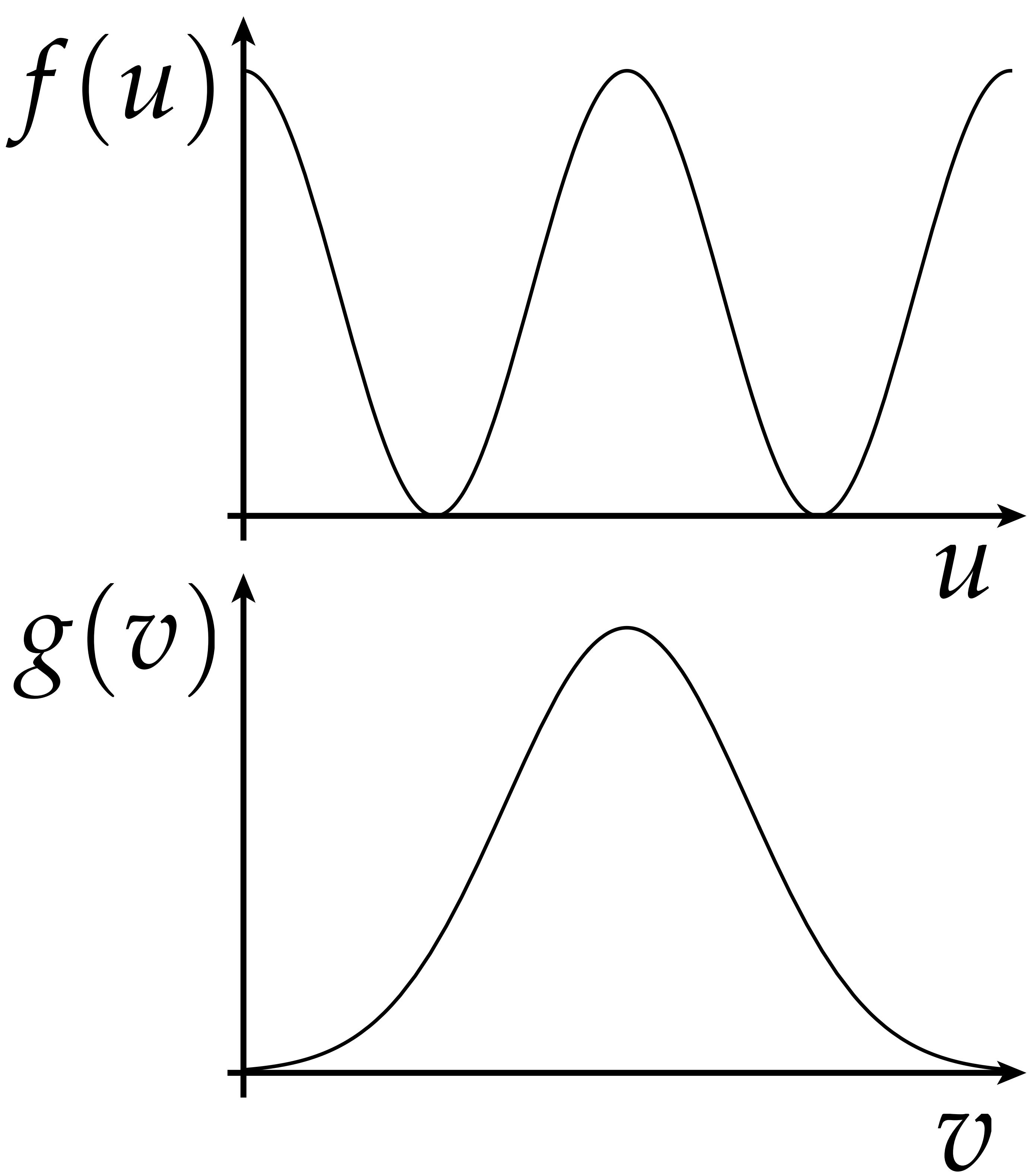
- To get "seamless" curves, need points <u>and</u> tangents to line up:

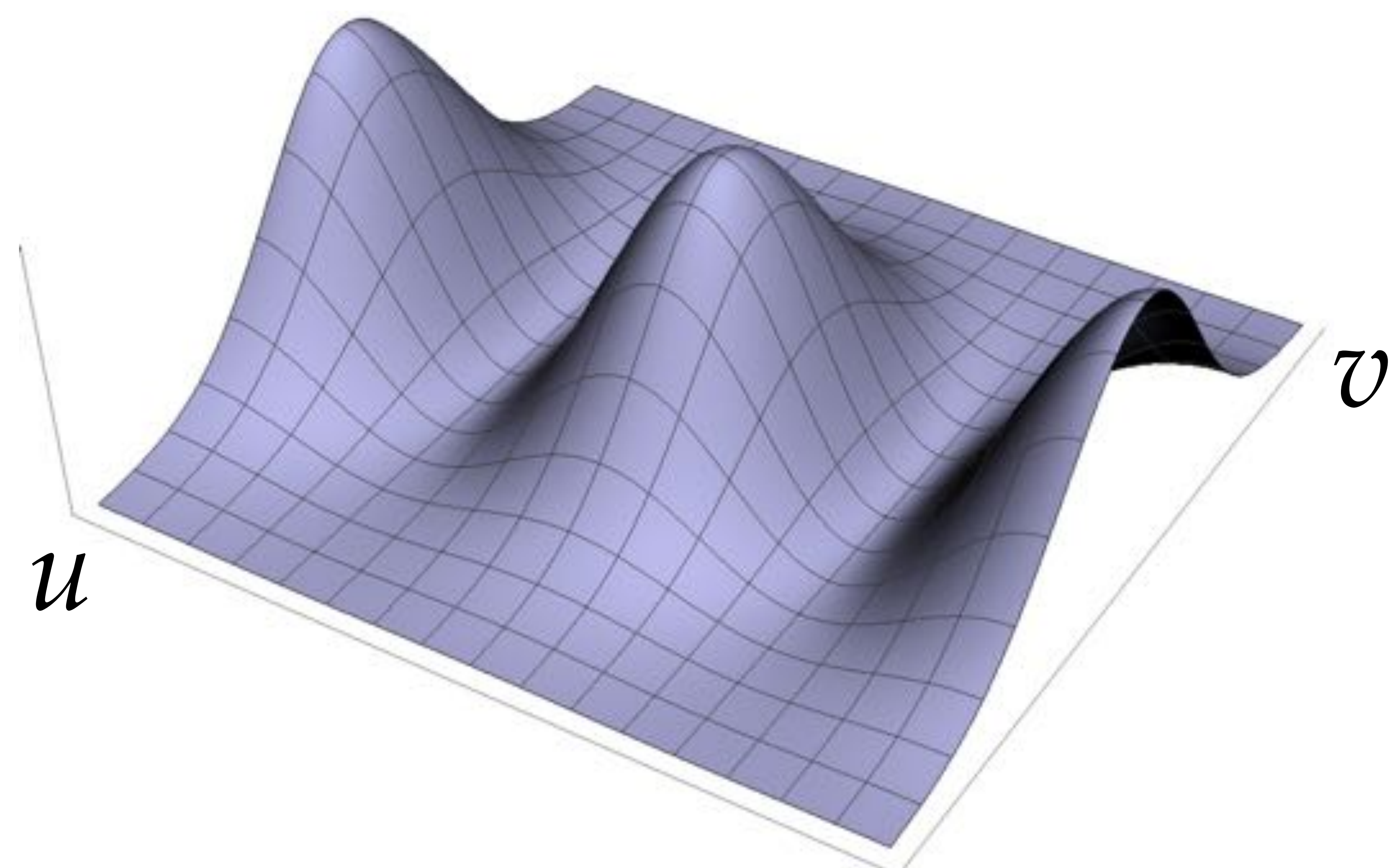


- Ok, but how?
- Each curve is cubic: $u^3 p_0 + 3u^2(1-u)p_1 + 3u(1-u)^2 p_2 + (1-u)^3 p_3$
- Want endpoints of each segment to meet
- Want tangents at endpoints to meet
- Q: How many constraints vs. degrees of freedom?
- Q: Could you do this with quadratic Bézier? Linear Bézier?

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"):**

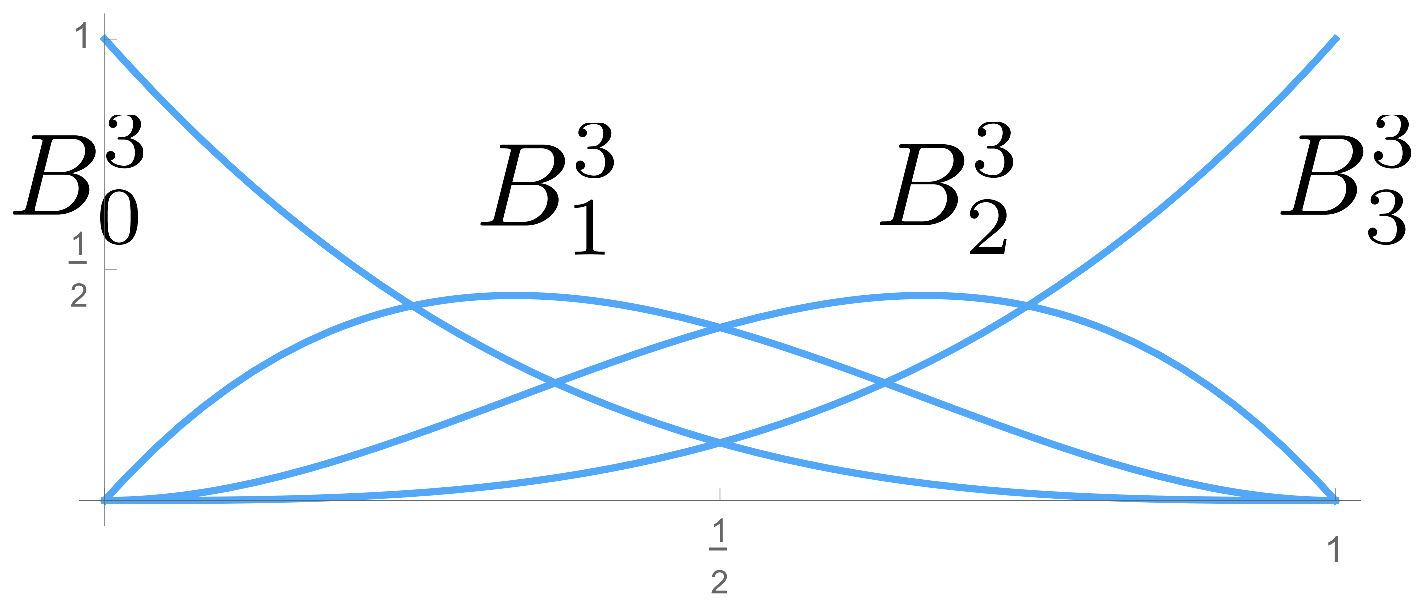




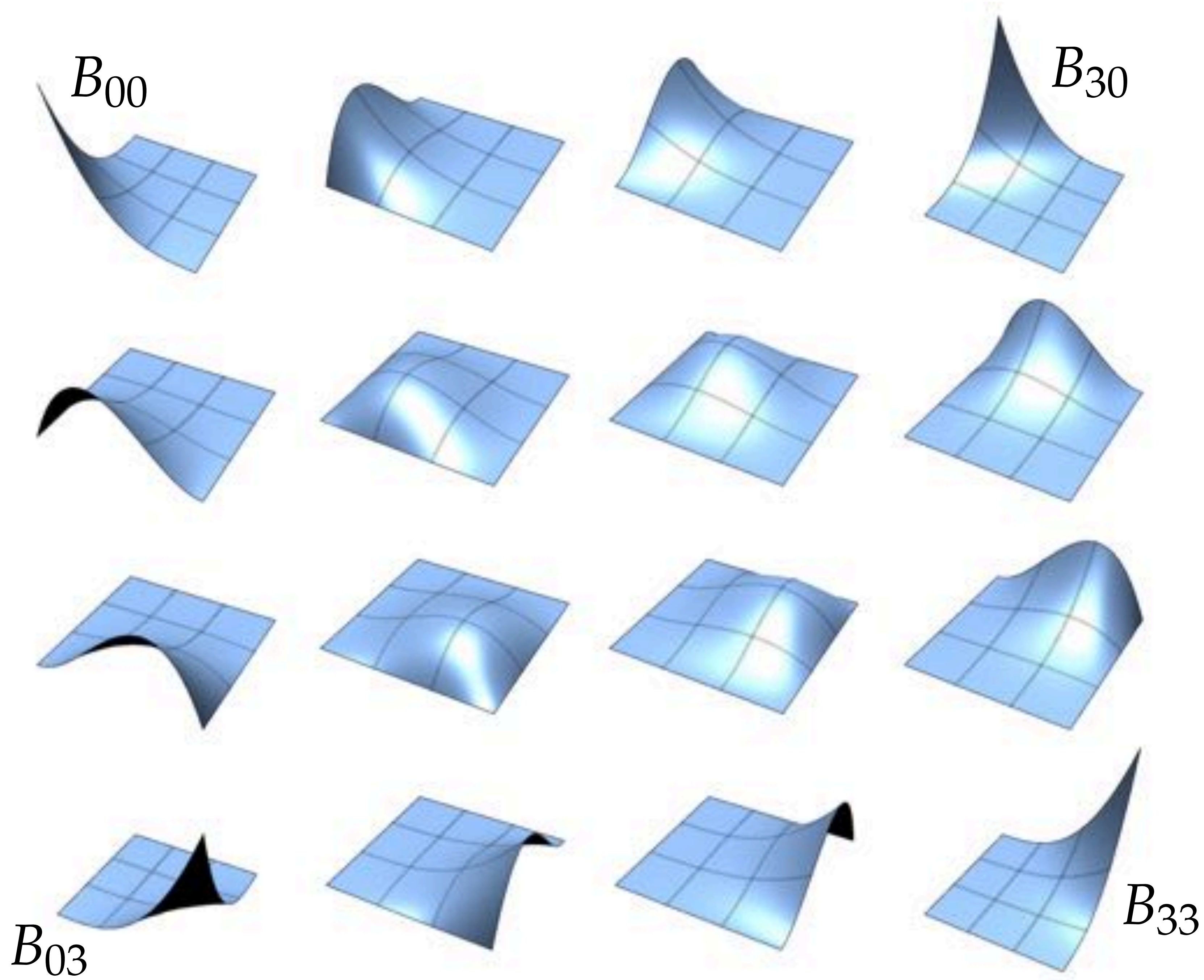
$$(f \otimes g)(u,v) := f(u)g(v)$$

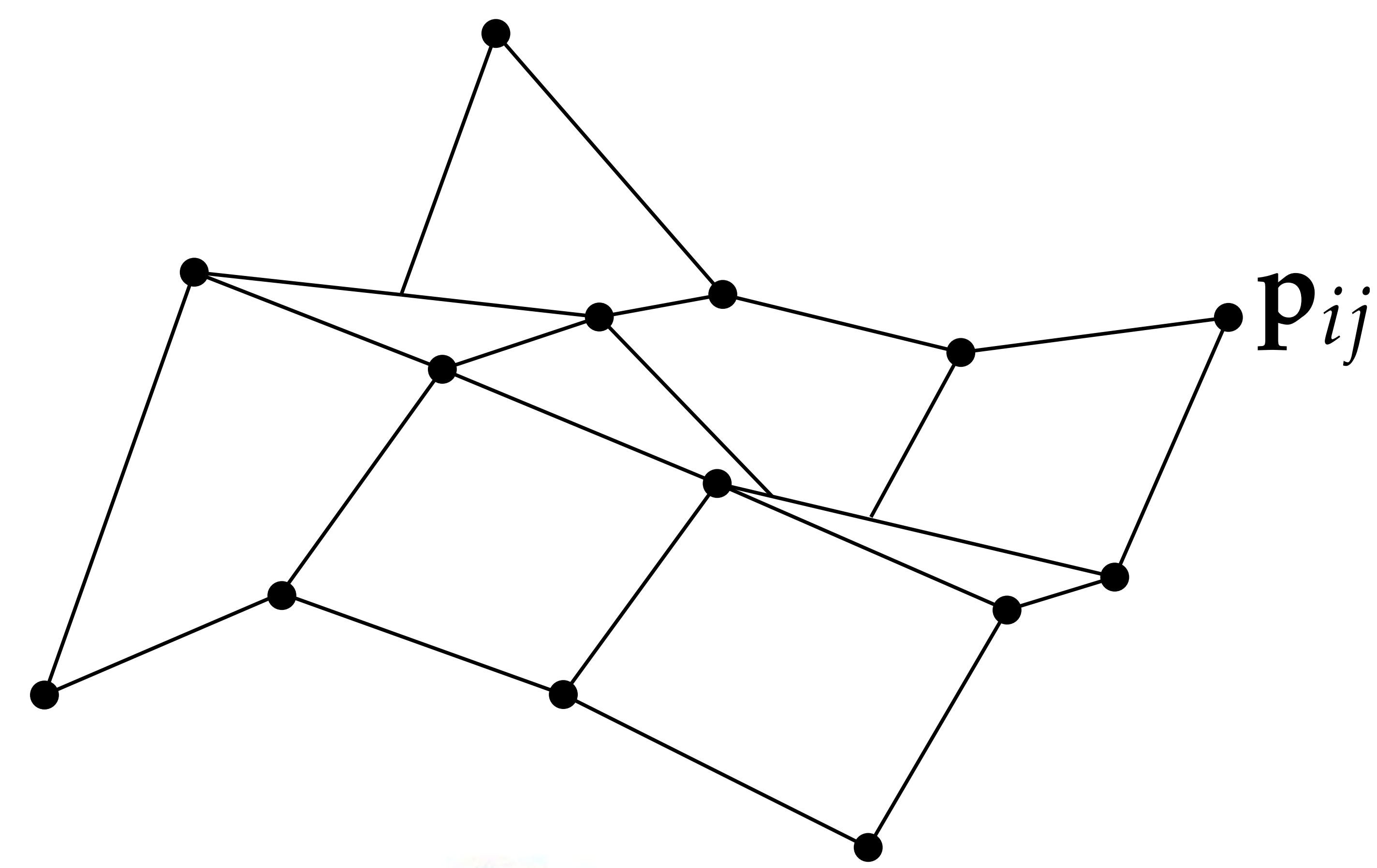
Bézier Patches

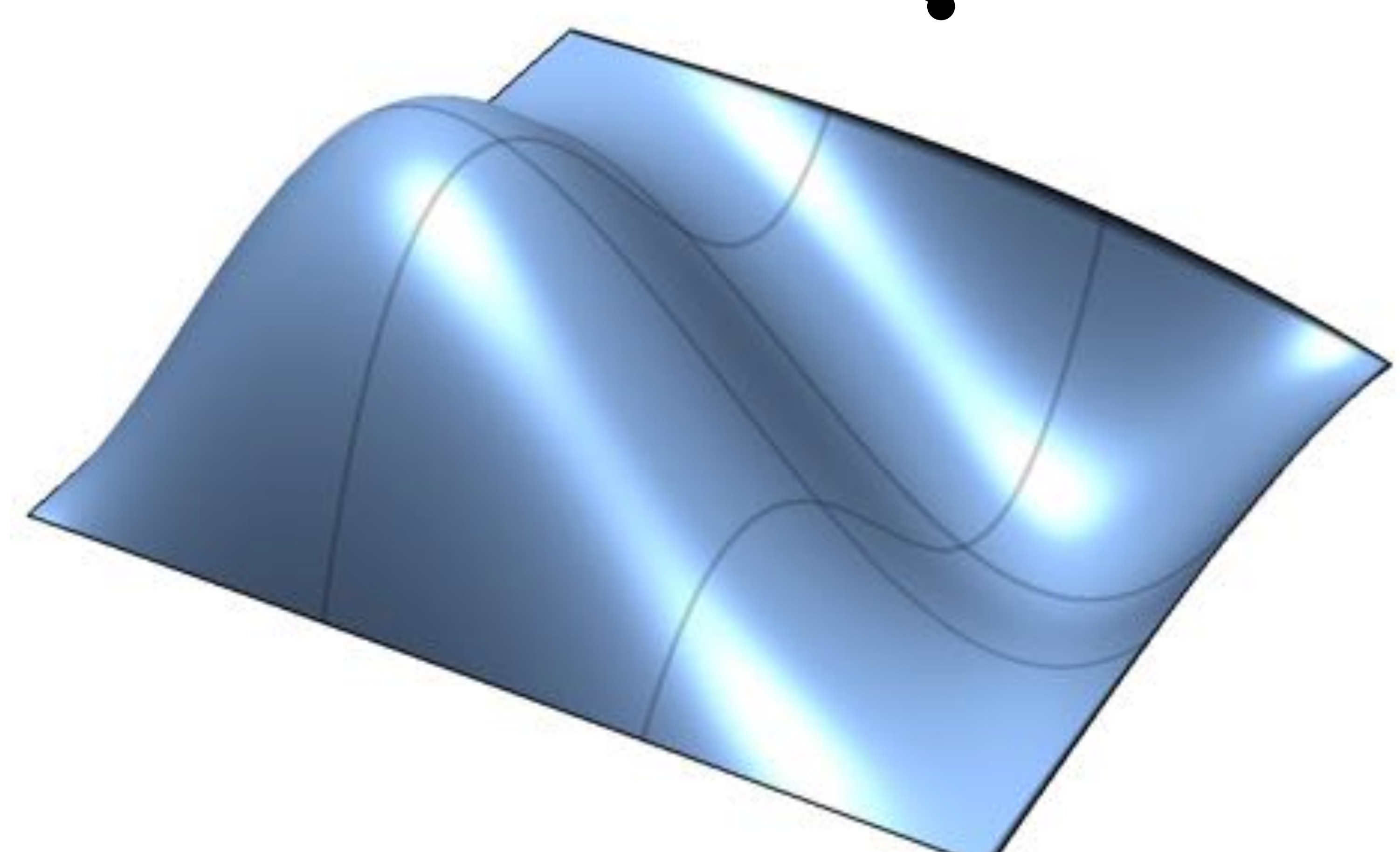
- **Bézier patch is sum of (tensor) products of Bernstein bases**



$$B^3_{i,j}(u,v) := B^3_i(u)B^3_j(v)$$



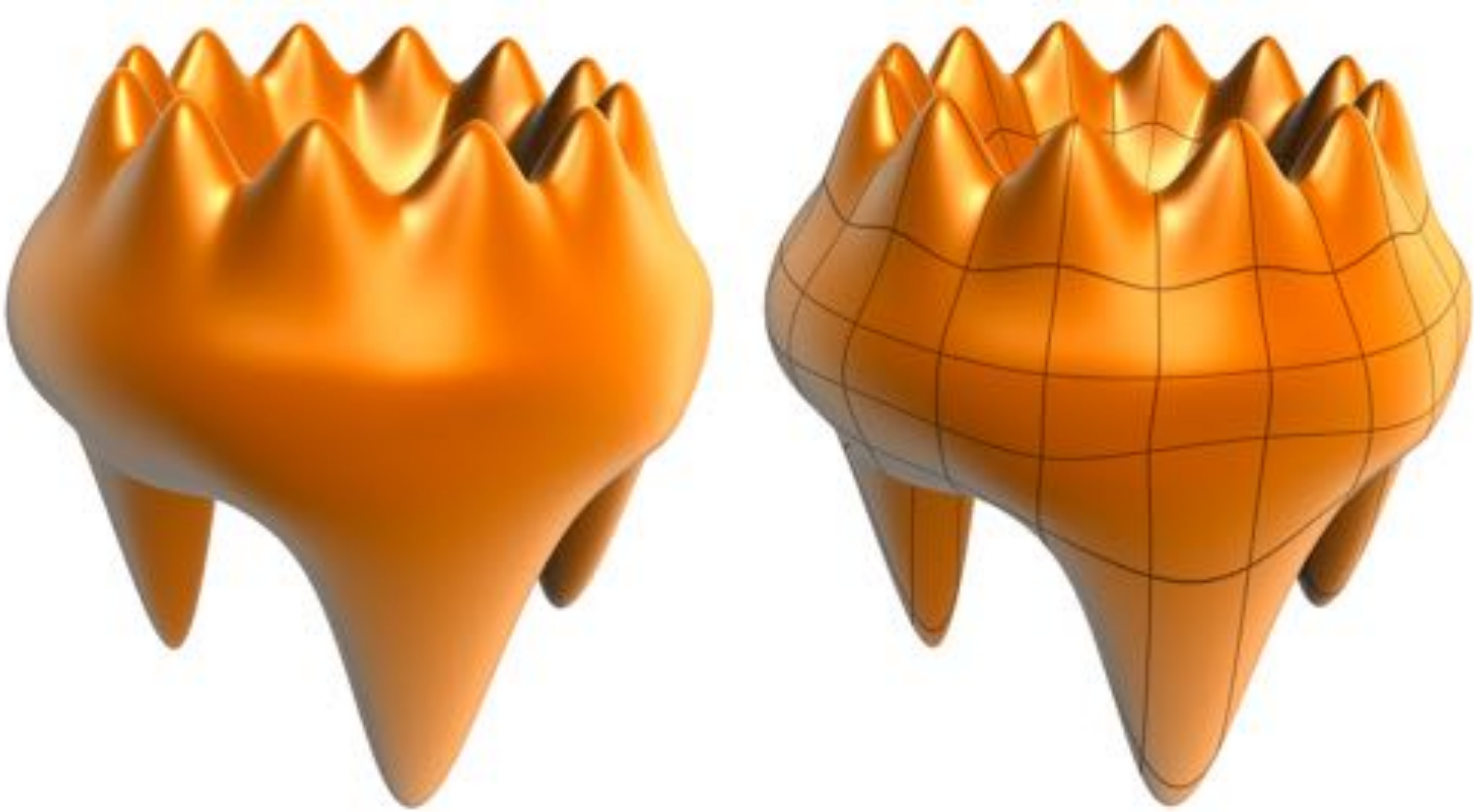




$$S(u,v) := \sum_{i=0}^{3}\sum_{j=0}^{3} B^3_{i,j}(u,v)\mathbf{p}_{ij}$$

Bézier Surface

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



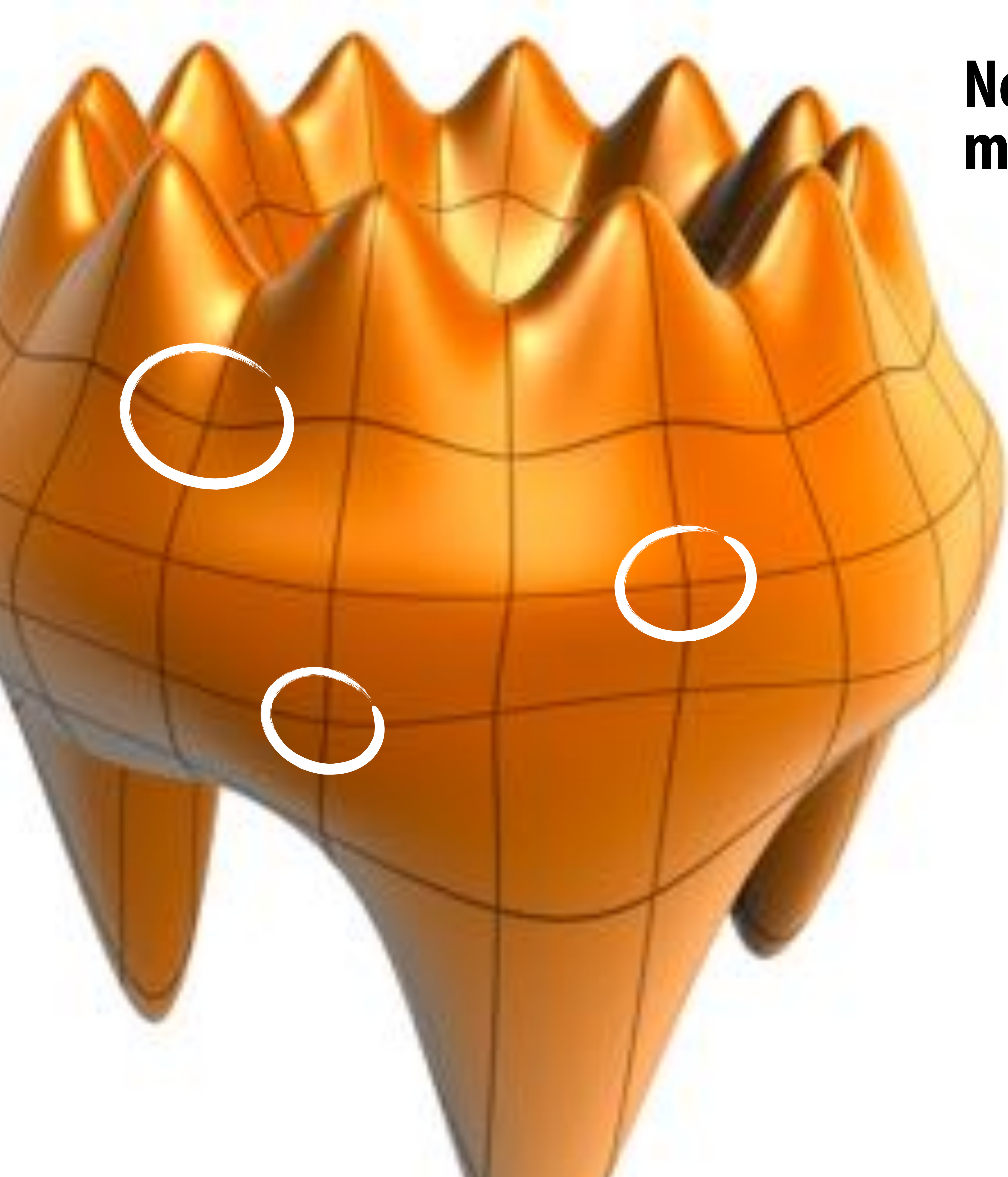
- Very easy to draw: just dice each patch into regular (u,v) grid!

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



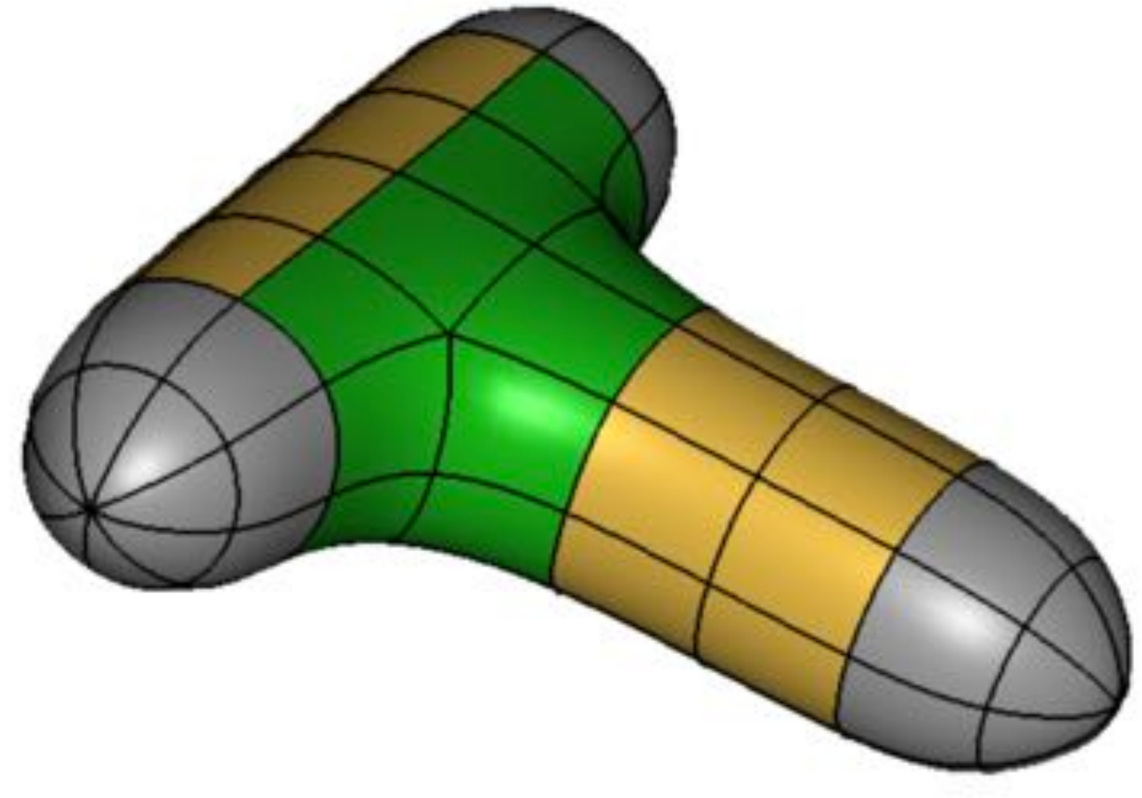
Notice that exactly four patches meet around every vertex!

In practice, far too constrained.

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

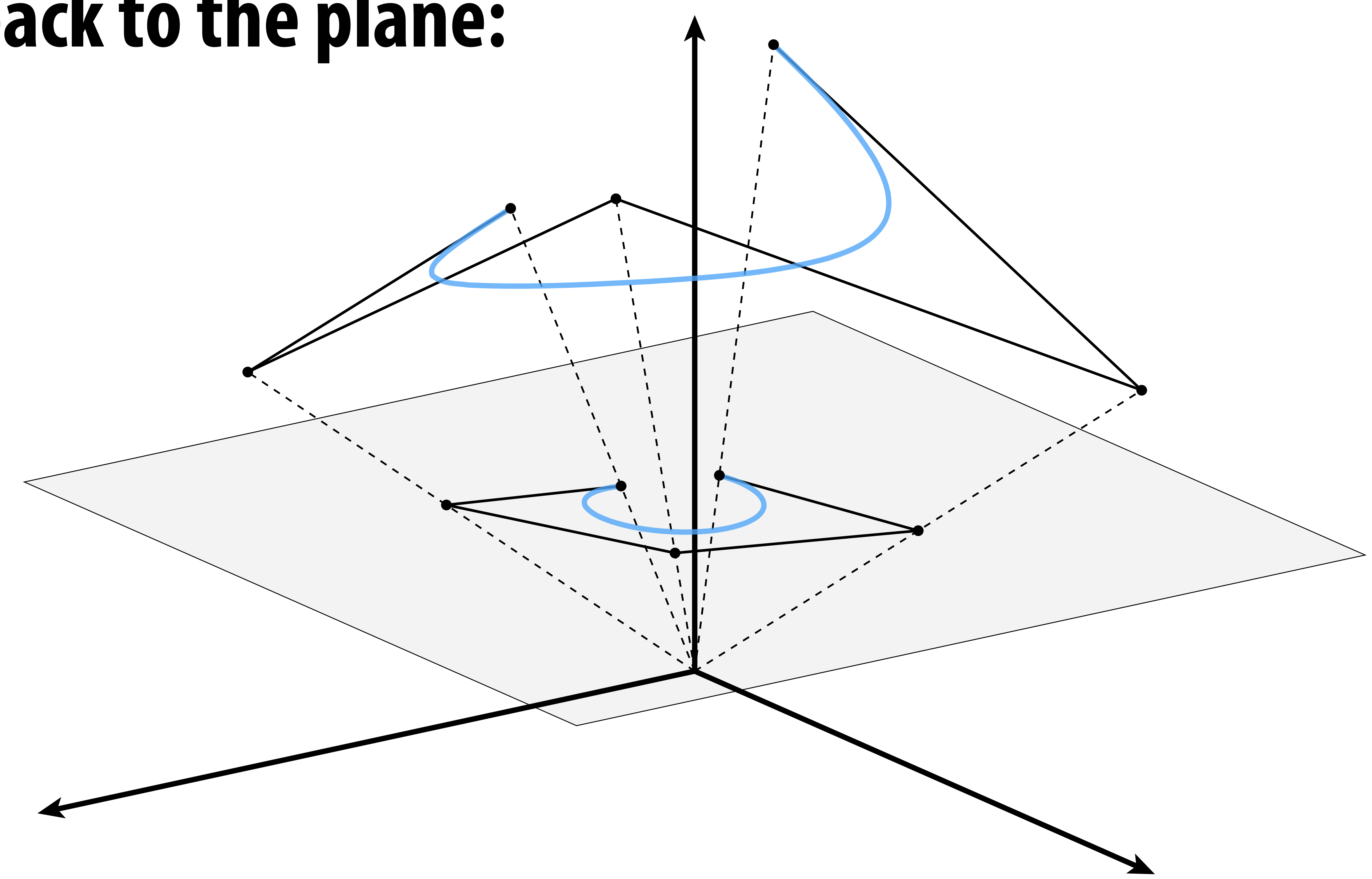
Spline patch schemes

- 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!



Rational B-Splines (Explicit)

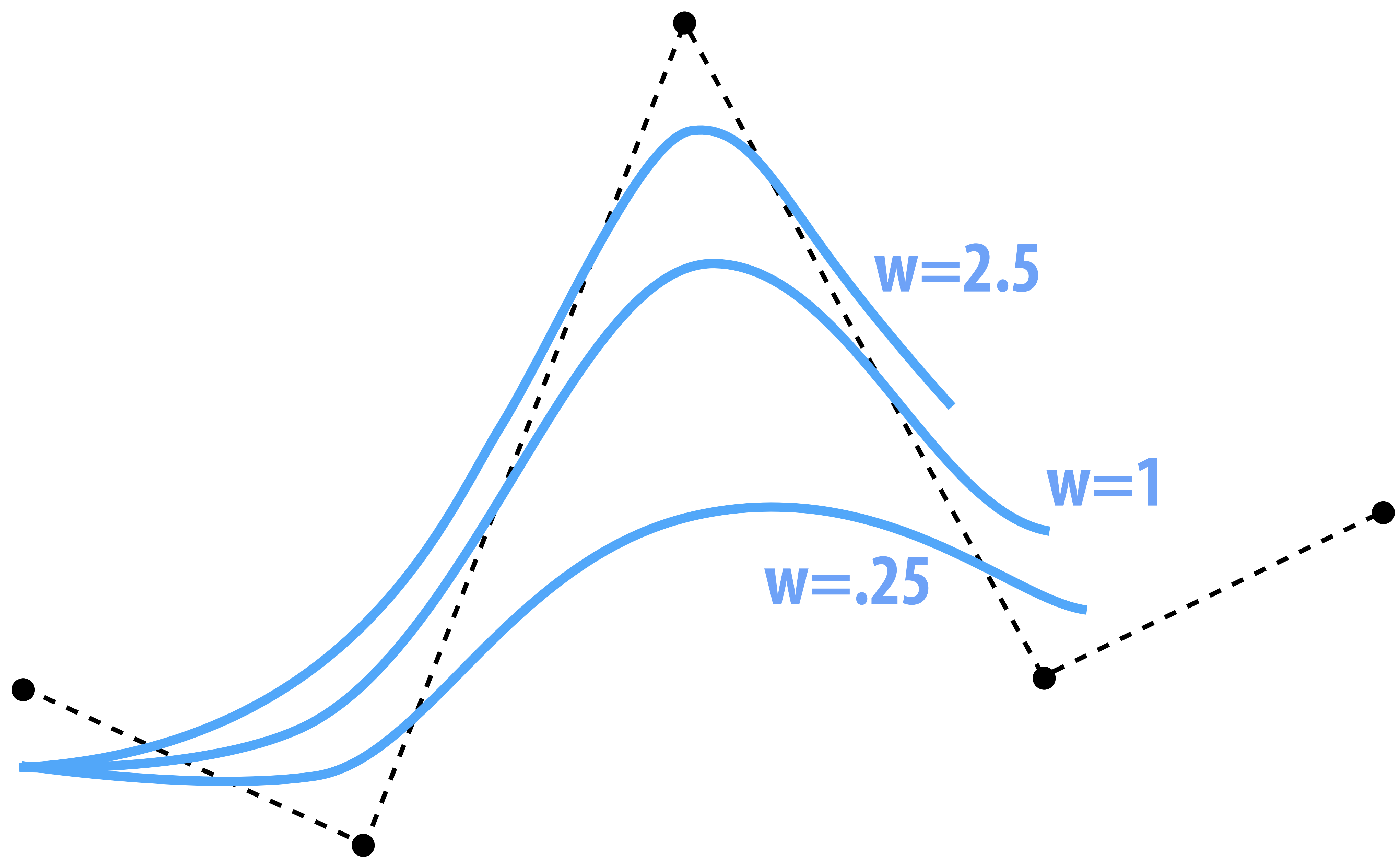
- Bézier can't exactly represent conics—not even the circle!
- Solution: interpolate in homogeneous coordinates, then project back to the plane:



Result is called a rational B-spline.

NURBS (Explicit)

- (N)on-(U)niform (R)ational (B)-(S)pline
  - knots at arbitrary locations (non-uniform)
  - expressed in homogeneous coordinates (rational)
  - piecewise polynomial curve (B-Spline)
- Homogeneous coordinate w controls "strength" of a vertex:

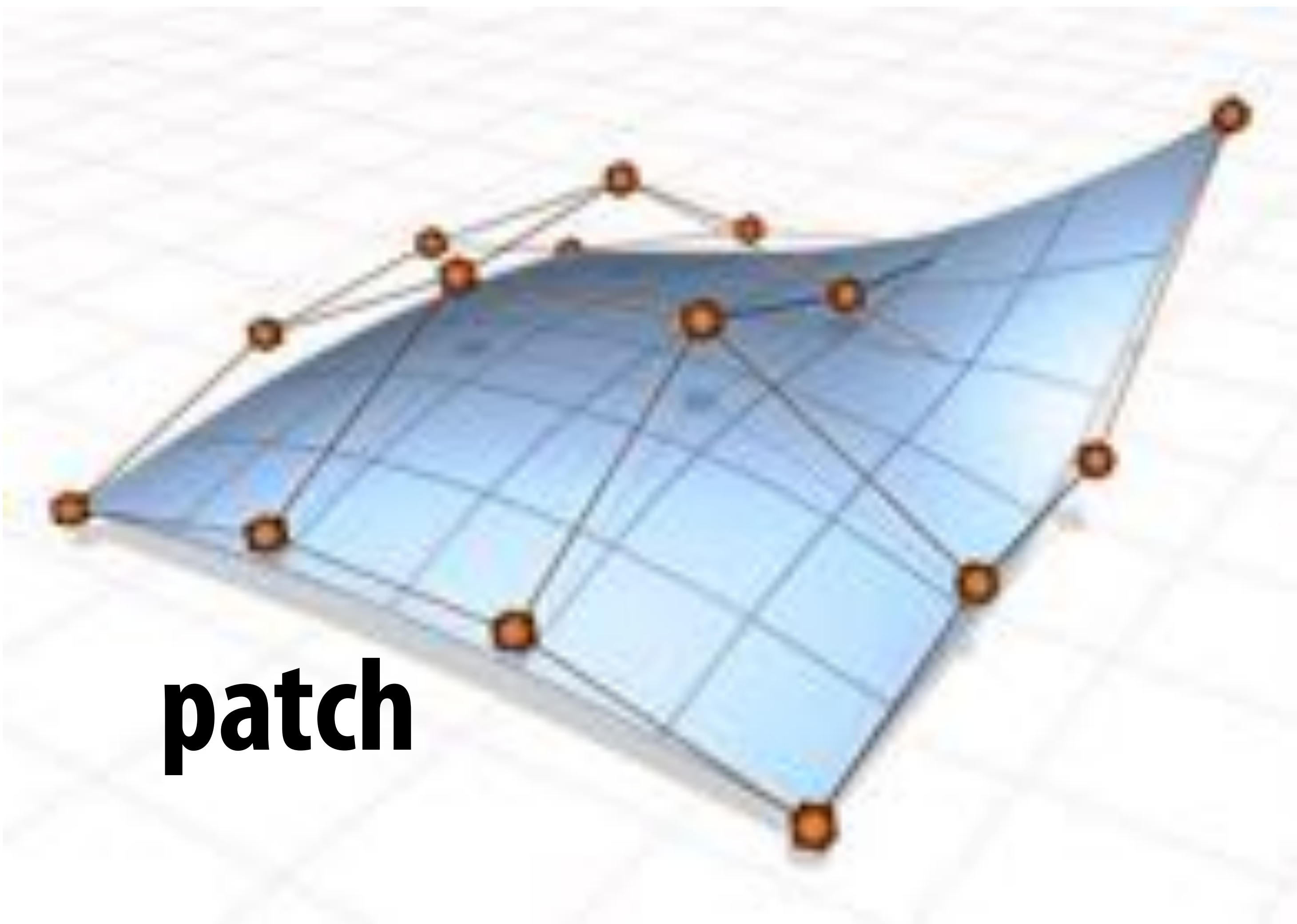


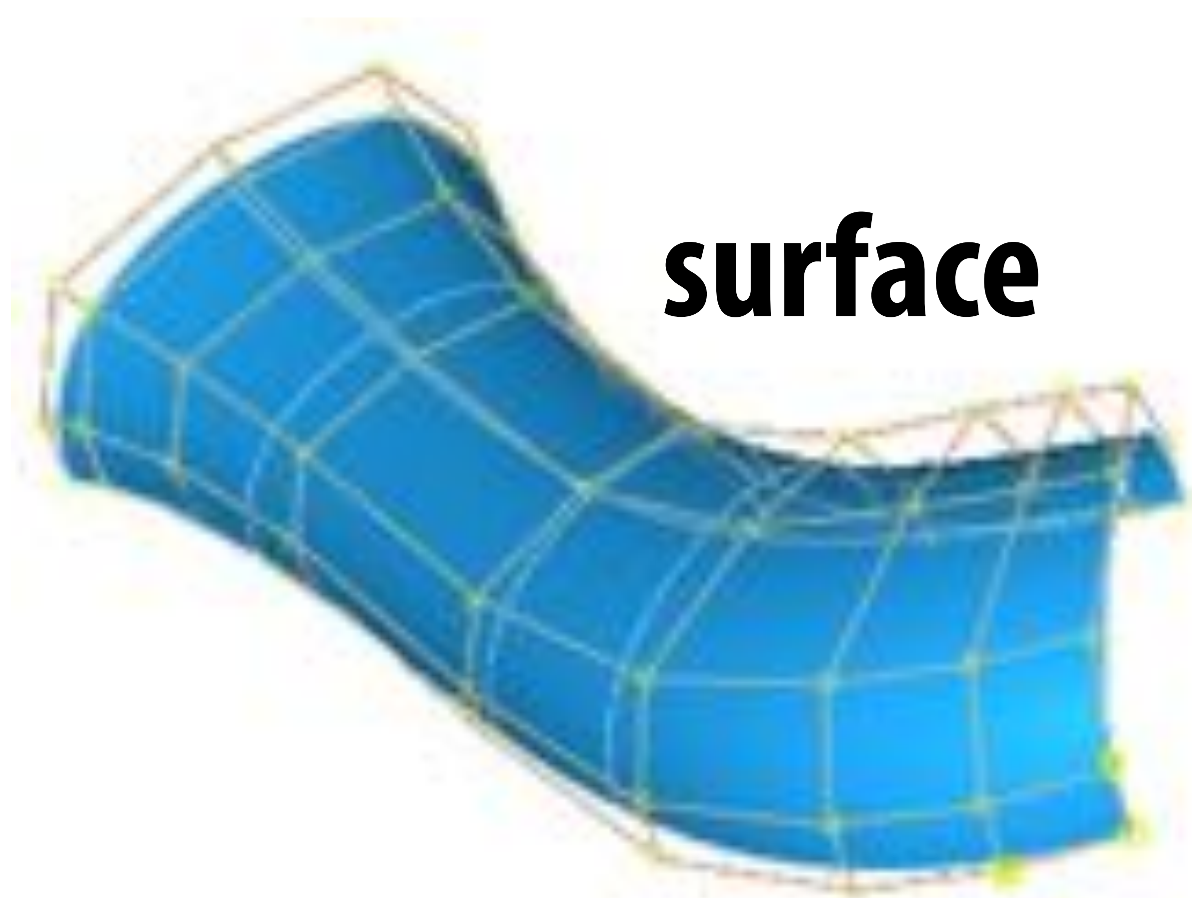
NURBS Surface (Explicit)

- How do we go from curves to surfaces?
- Use tensor product of NURBS curves to get a patch:

$$S(u,v) := N_i(u)N_j(v)p_{ij}$$

- Multiple NURBS patches form a surface

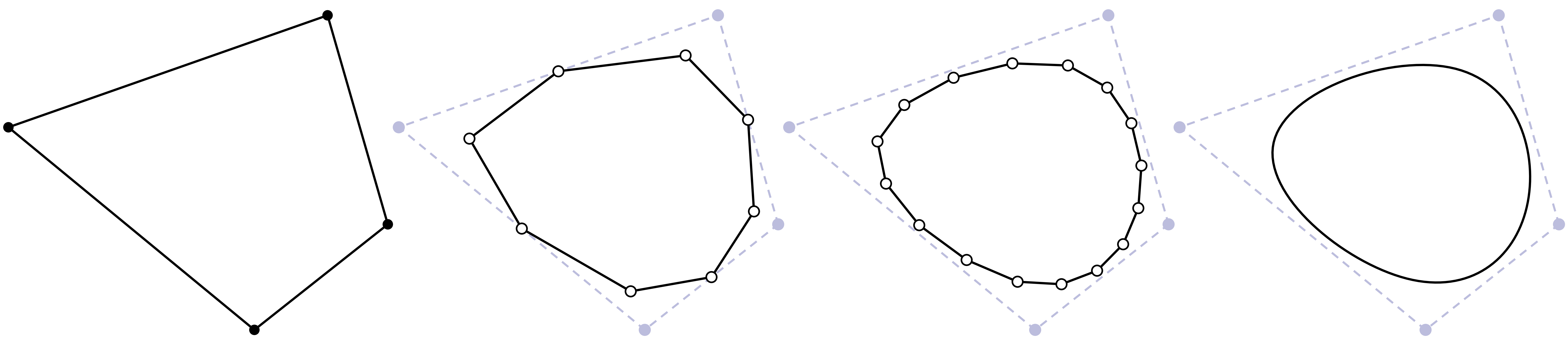




- Pros: easy to evaluate, exact conics, high degree of continuity
- Cons: Hard to piece together patches / hard to edit (many DOFs)

Subdivision

- Alternative starting point for curves/surfaces: subdivision
- Start with "control curve"
- Repeatedly split, take weighted average to get new positions
- For careful choice of averaging rule, approaches nice limit curve
  - Often exact same curve as well-known spline schemes!



Q: Is subdivision an explicit or implicit representation?

Subdivision—Example

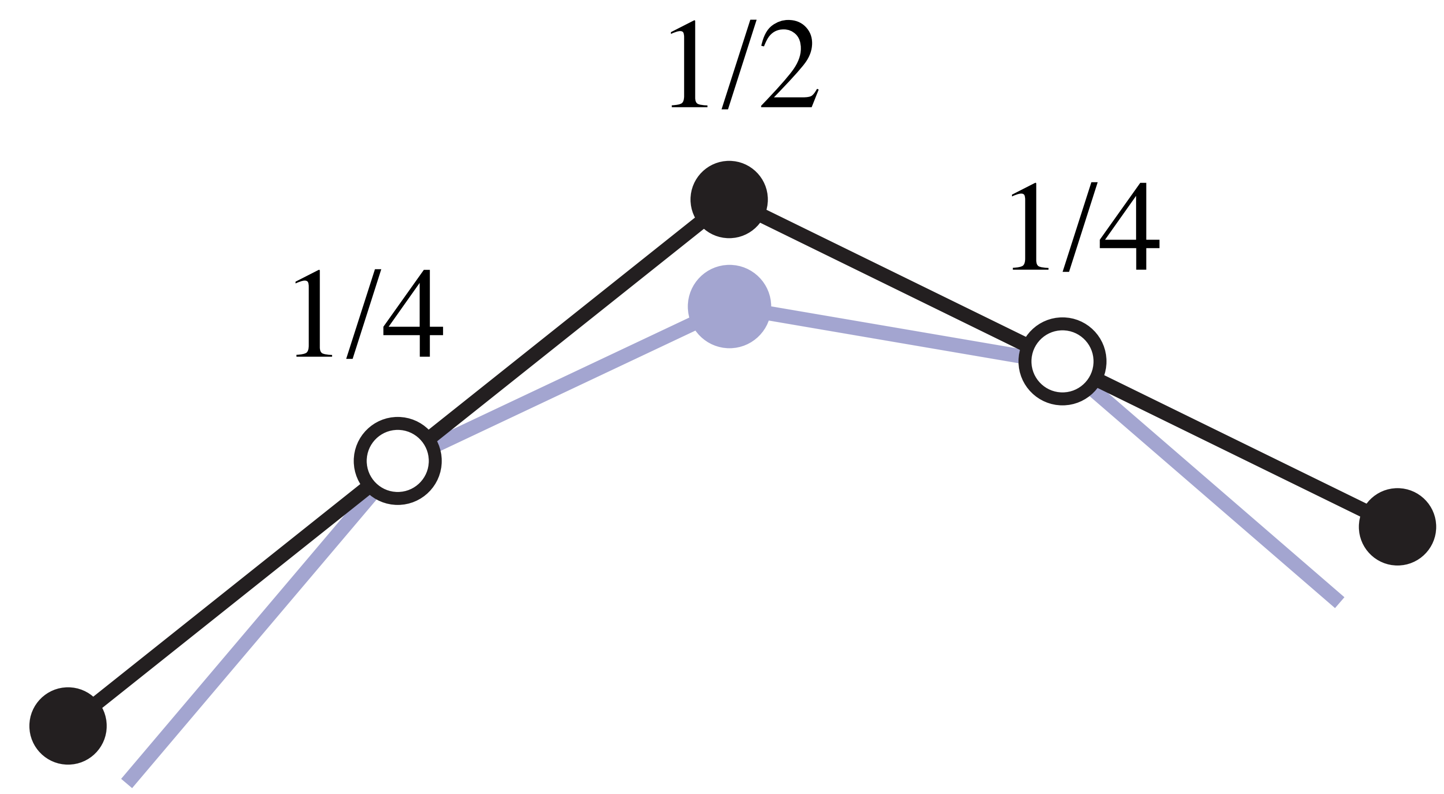
- **One possible scheme: Lane-Riesenfeld**
  - **insert midpoint of each edge**
  - **use row k of Pascal's triangle (normalized to 1) as weights for neighbors**
  - **e.g., $k = 2$, get weights** (1/4,1/2,1/4)
  - **limit is B-spline of degree $k + 1$**

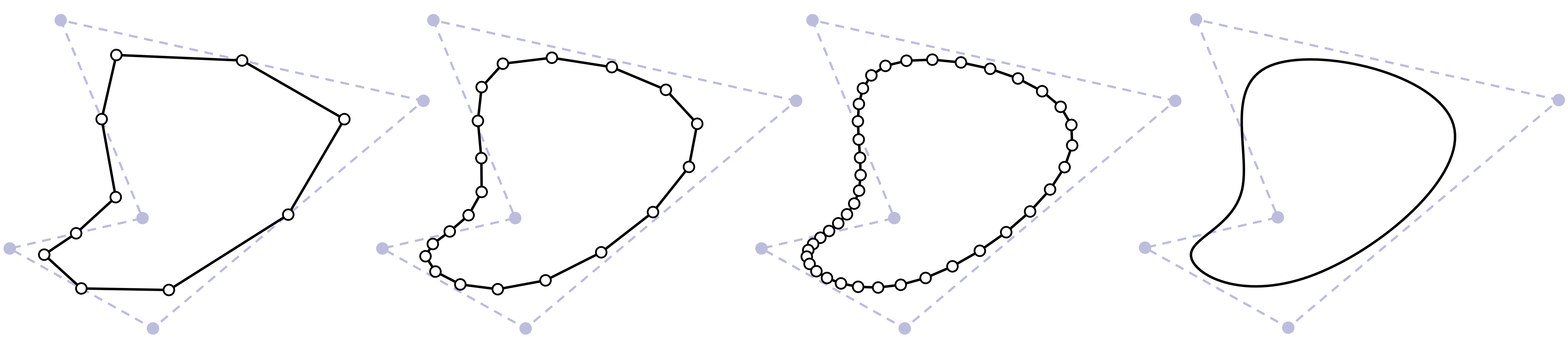
$k = 0:$ 1

$k = 1:$ 1 1

$k = 2:$ 1 2 1

$k = 3:$ 1 3 3 1

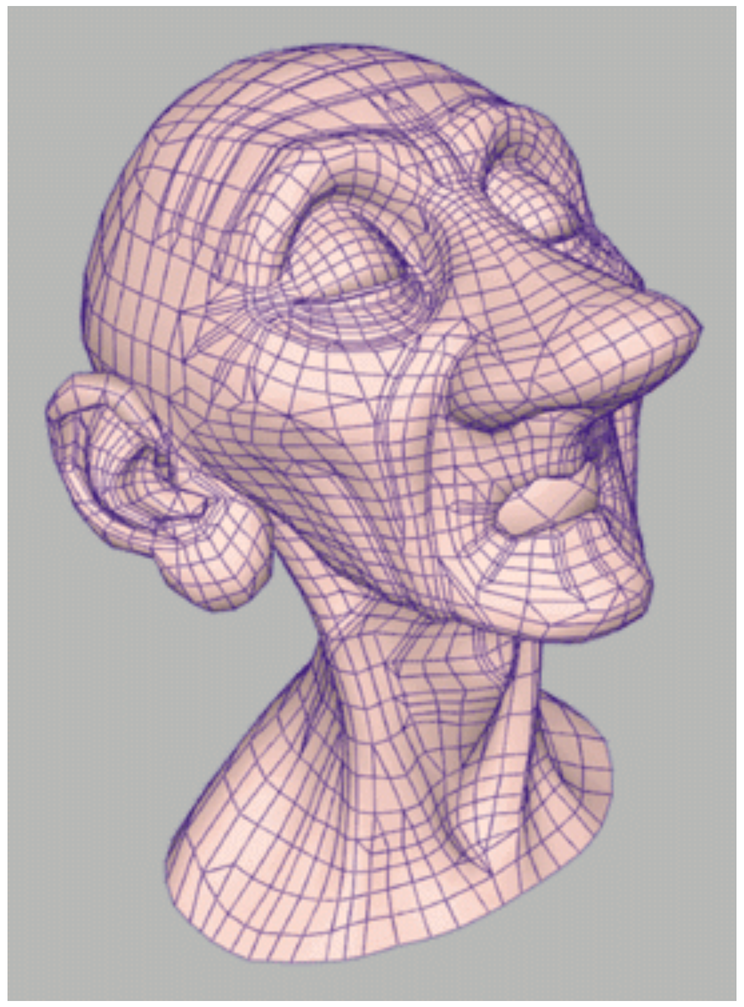


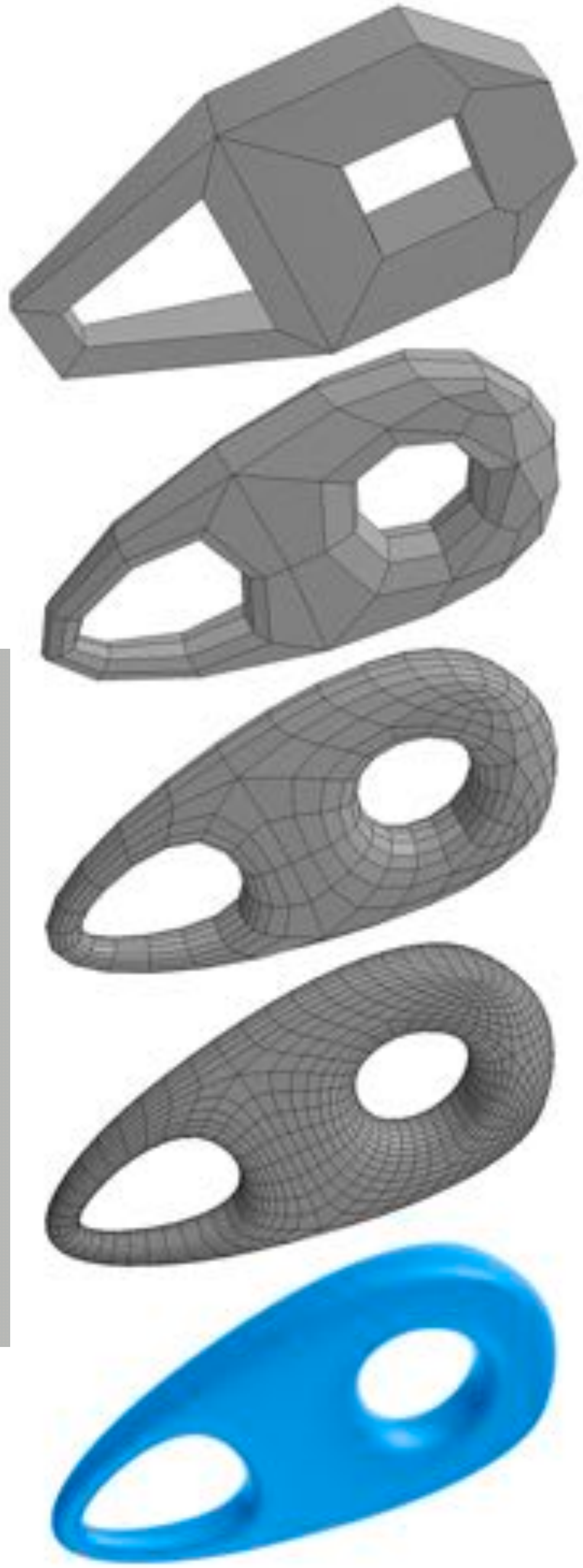


iteration 1 **iteration 2** **iteration 3** **limit curve**

Subdivision Surfaces (Explicit)

- Start with coarse polygon mesh ("control cage")
- Subdivide each element
- Update vertices via local averaging
- Many possible rules:
  - Catmull-Clark (quads)
  - Loop (triangles)
  - ...
- Common issues:
  - interpolating or approximating?
  - continuity at vertices?
- Easier than splines for modeling; harder to evaluate pointwise
- Widely used in practice (2019 Academy Awards!)





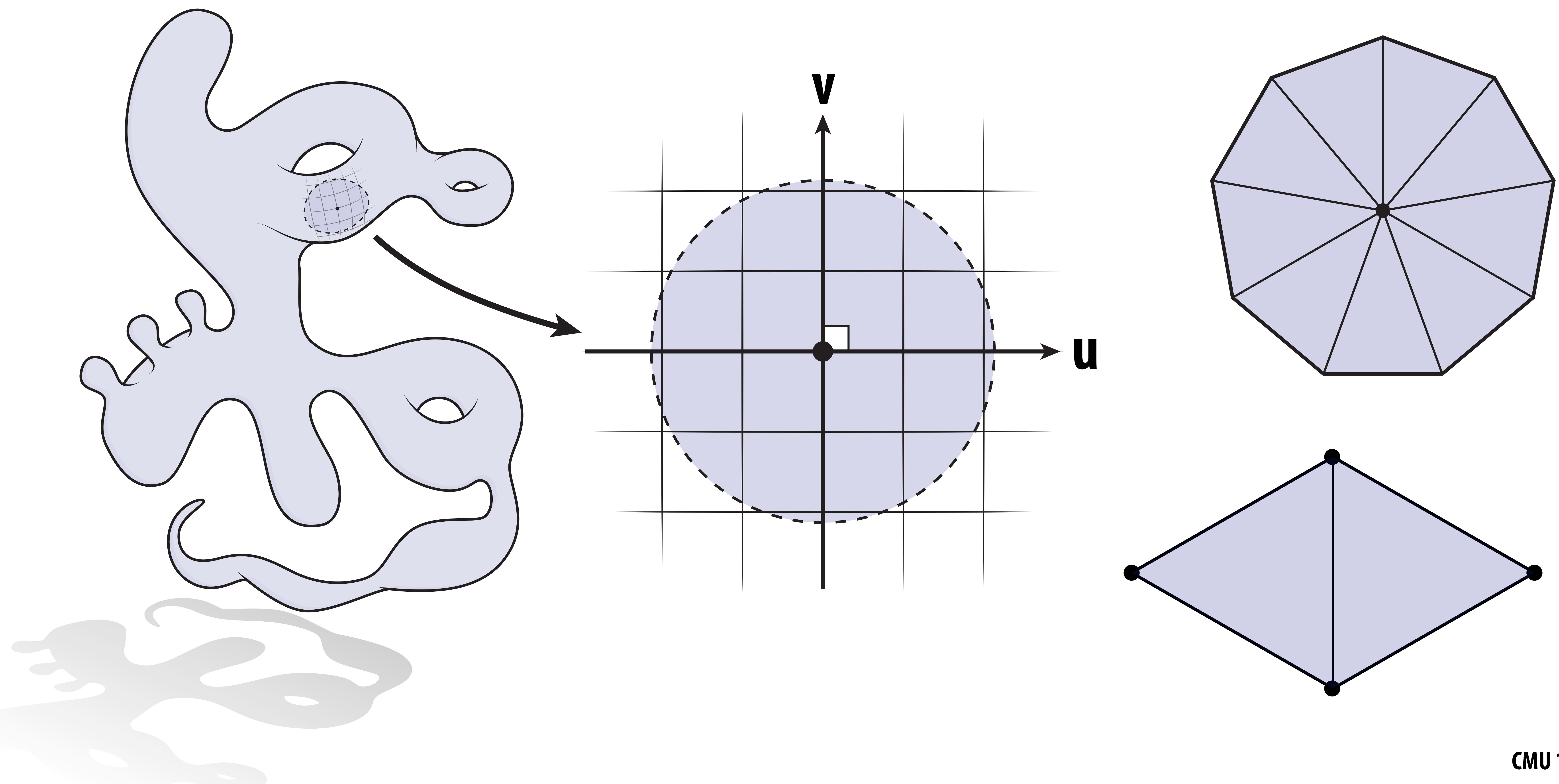
Subdivision in Action (Pixar's "Geri's Game")

see: de Rose et al, "Subdivision Surfaces in Character Animation"

What is a surface anyway?

Manifold Assumption

- First, let's define manifold geometry
- Can be hard to understand motivation at first!
- Let's revisit a more familiar example...

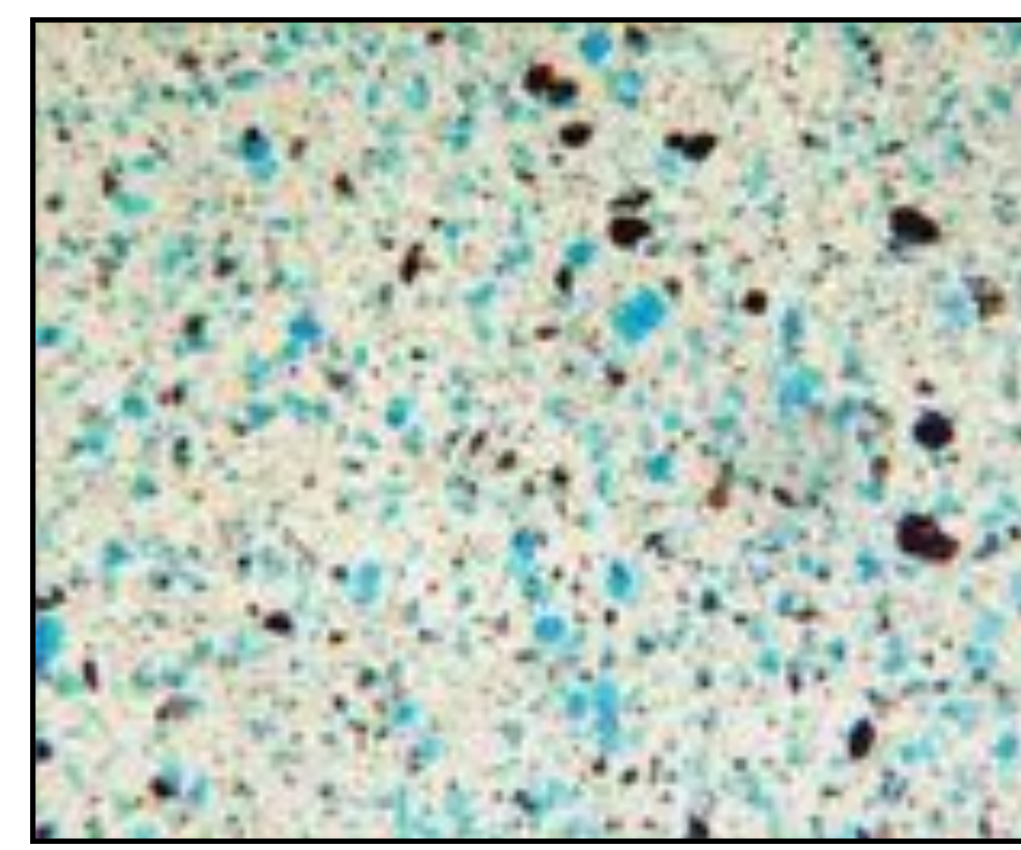


Bitmap Images, Revisited

To encode images, we used a regular grid of pixels:

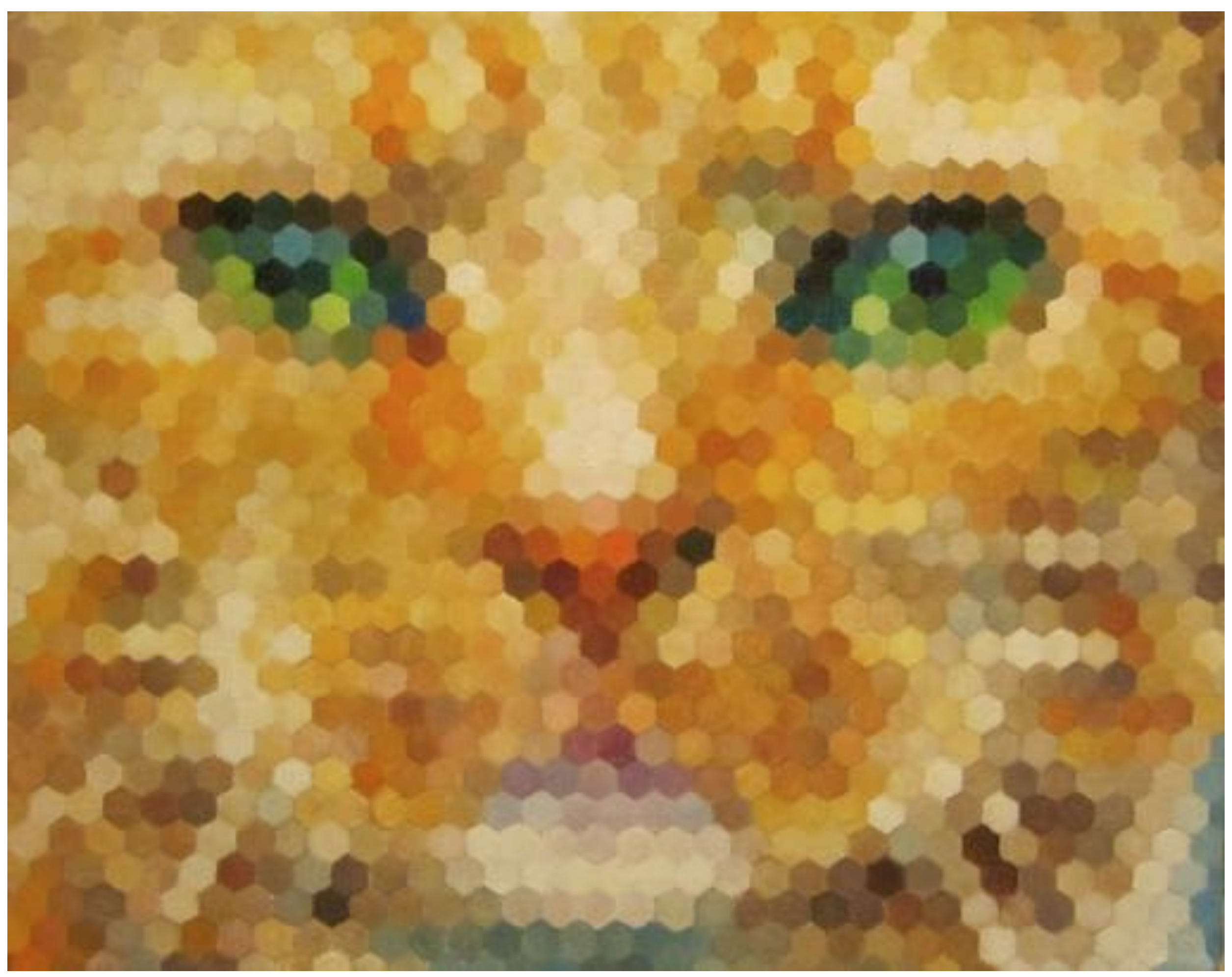
But images are not fundamentally made of little squares:

Goyō Hashiguchi, *Kamisuki* (ca 1920)



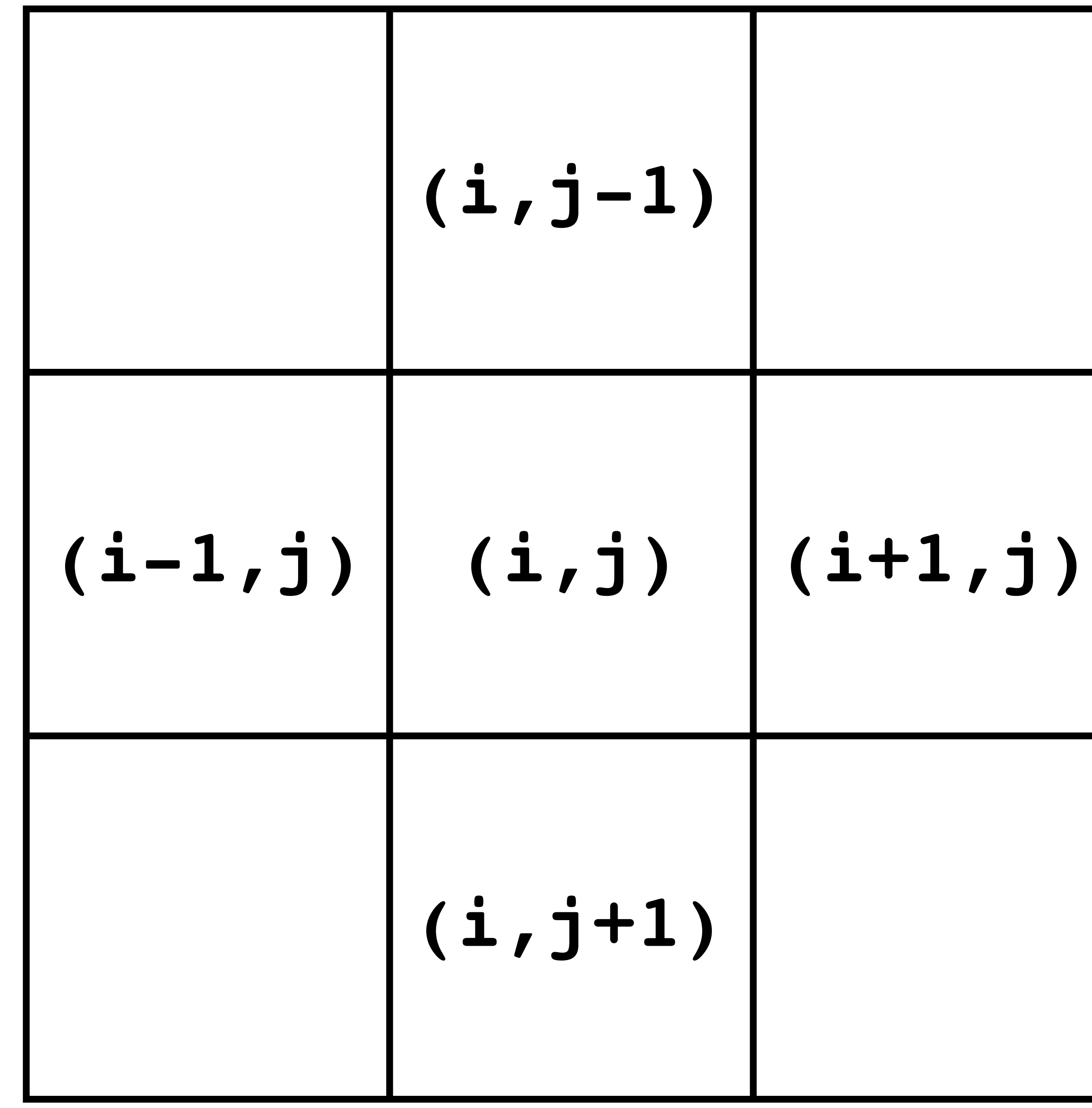
photomicrograph of paint

So why did we choose a square grid?



...rather than dozens of possible alternatives?

Regular grids make life easy



- One reason: SIMPLICITY / EFFICIENCY
  - E.g., always have four neighbors
  - Easy to index, easy to filter…
  - Storage is just a list of numbers
- Another reason: GENERALITY
  - Can encode basically any image
- Are regular grids always the best choice for bitmap images?
  - No! E.g., suffer from anisotropy, don't capture edges, ...
  - But more often than not are a pretty good choice
- Will see a similar story with geometry...

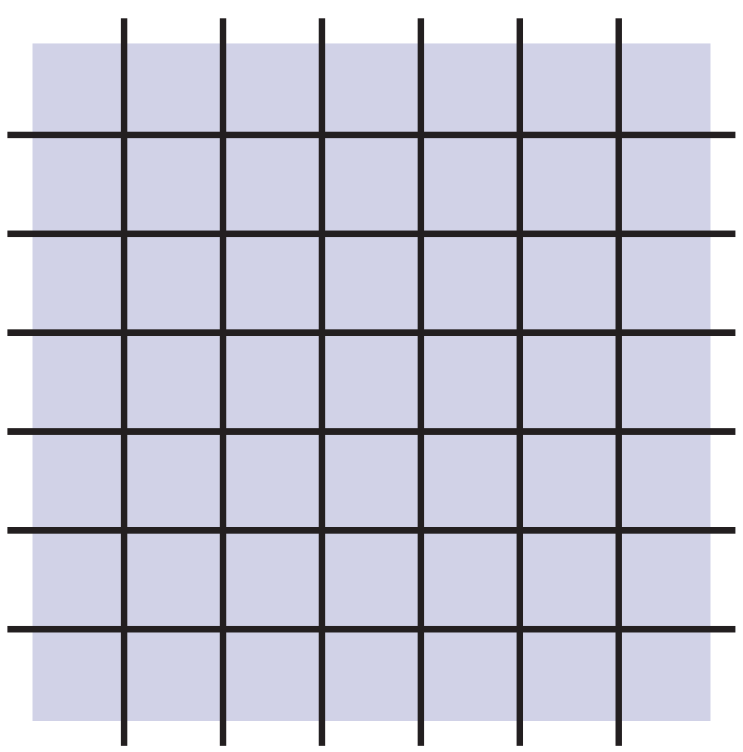
So, how should we encode surfaces?

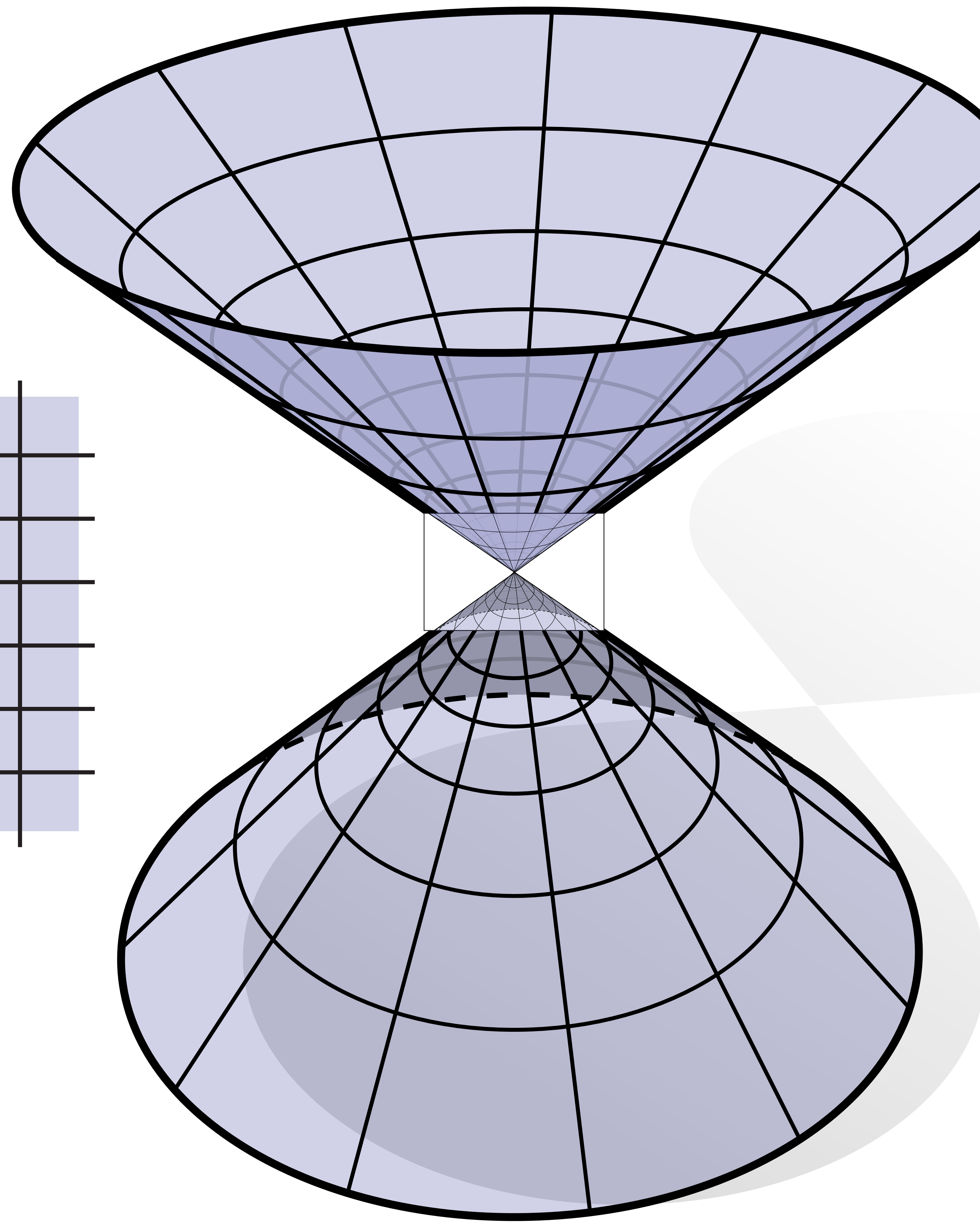
Smooth Surfaces

- Intuitively, a <u>surface</u> is the boundary or "shell" of an object
- (Think about the candy shell, not the chocolate.)
- Surfaces are manifold:
  - If you zoom in far enough, can draw a regular coordinate grid
  - E.g., the Earth from space vs. from the ground

Isn't every shape manifold?

- No, for instance:

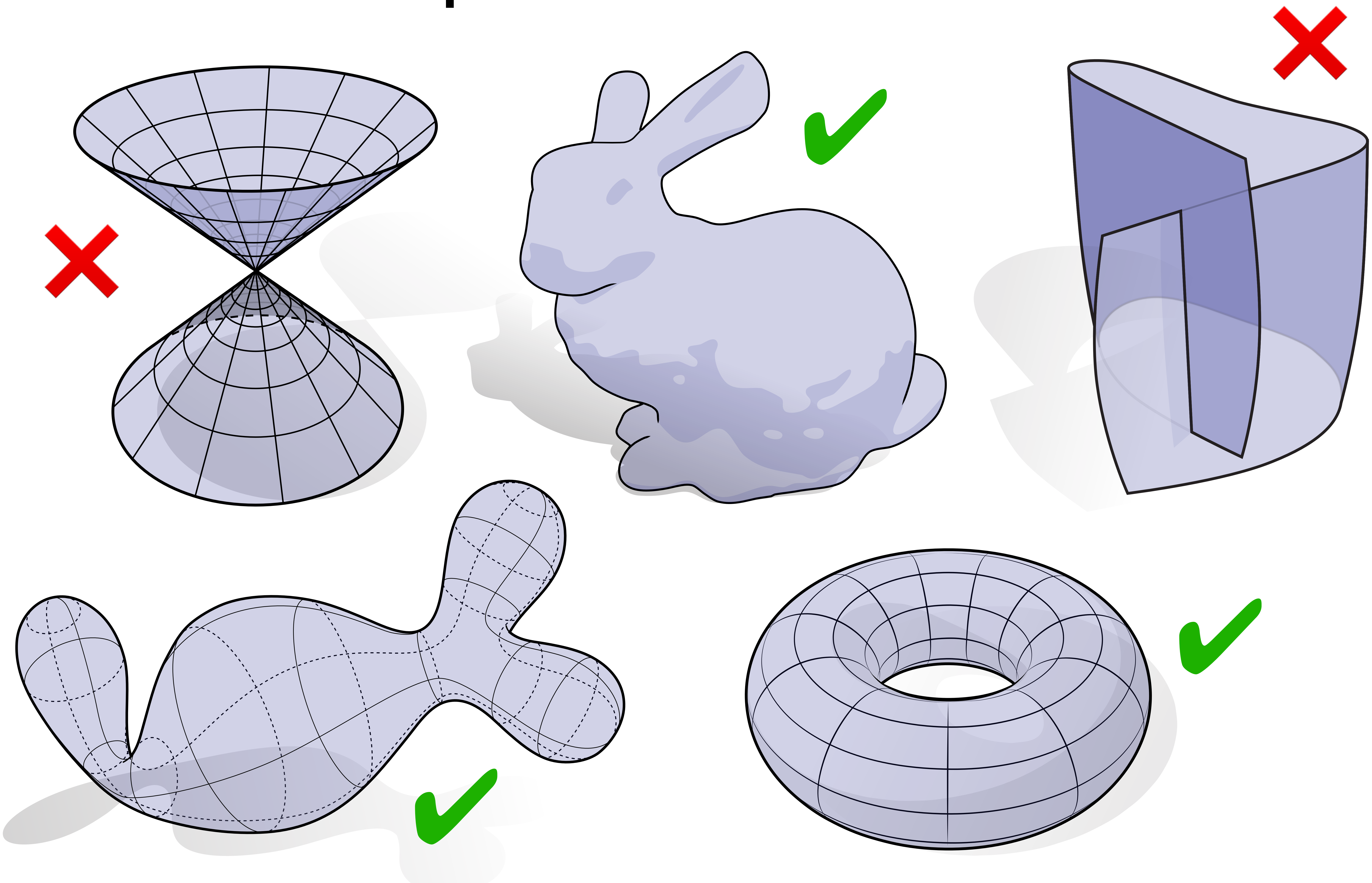




Can't draw ordinary 2D grid at center, no matter how close we get.

Examples—Manifold vs. Nonmanifold

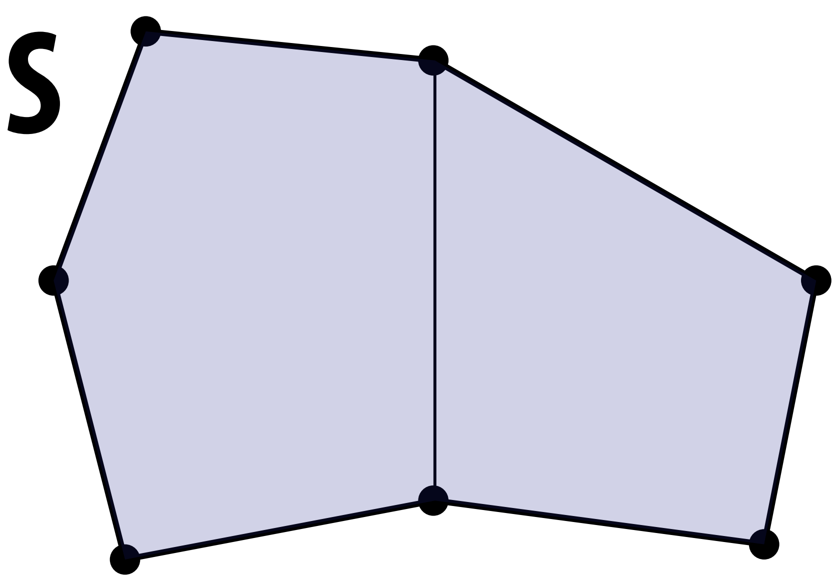
- Which of these shapes are manifold?



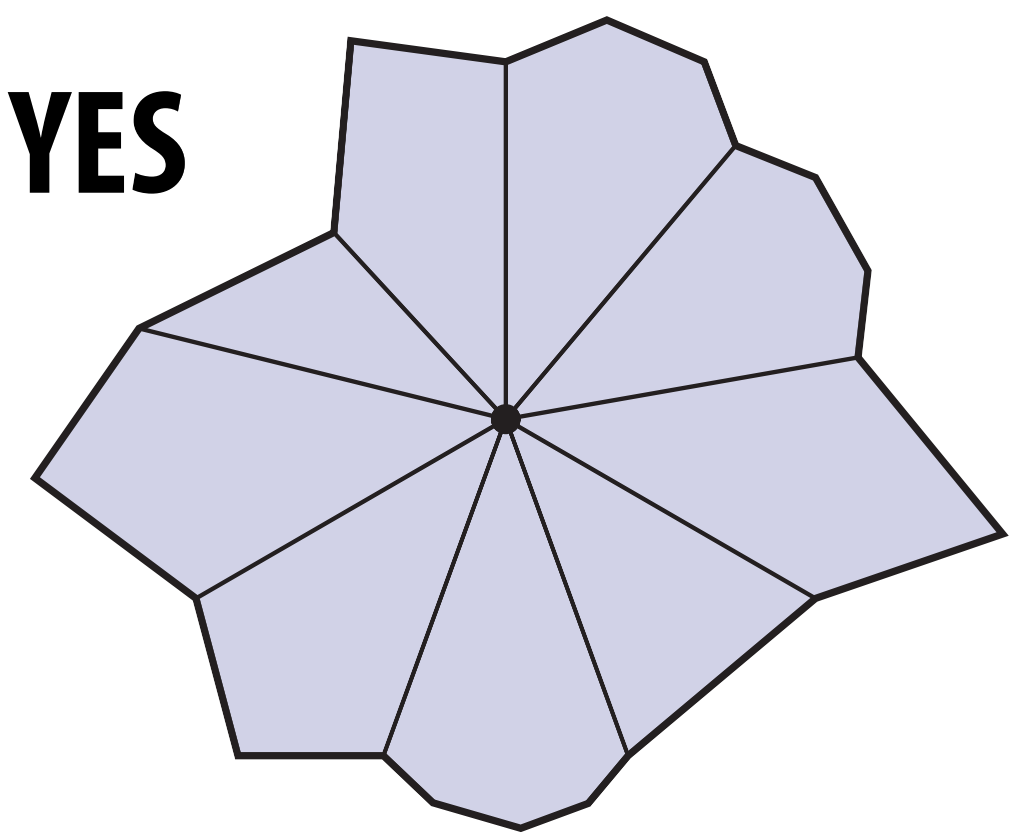
A manifold polygon mesh has fans, not fins

- For polygonal surfaces just two easy conditions to check:
  1. Every edge is contained in only two polygons (no “fins”)
  2. The polygons containing each vertex make a single “fan”

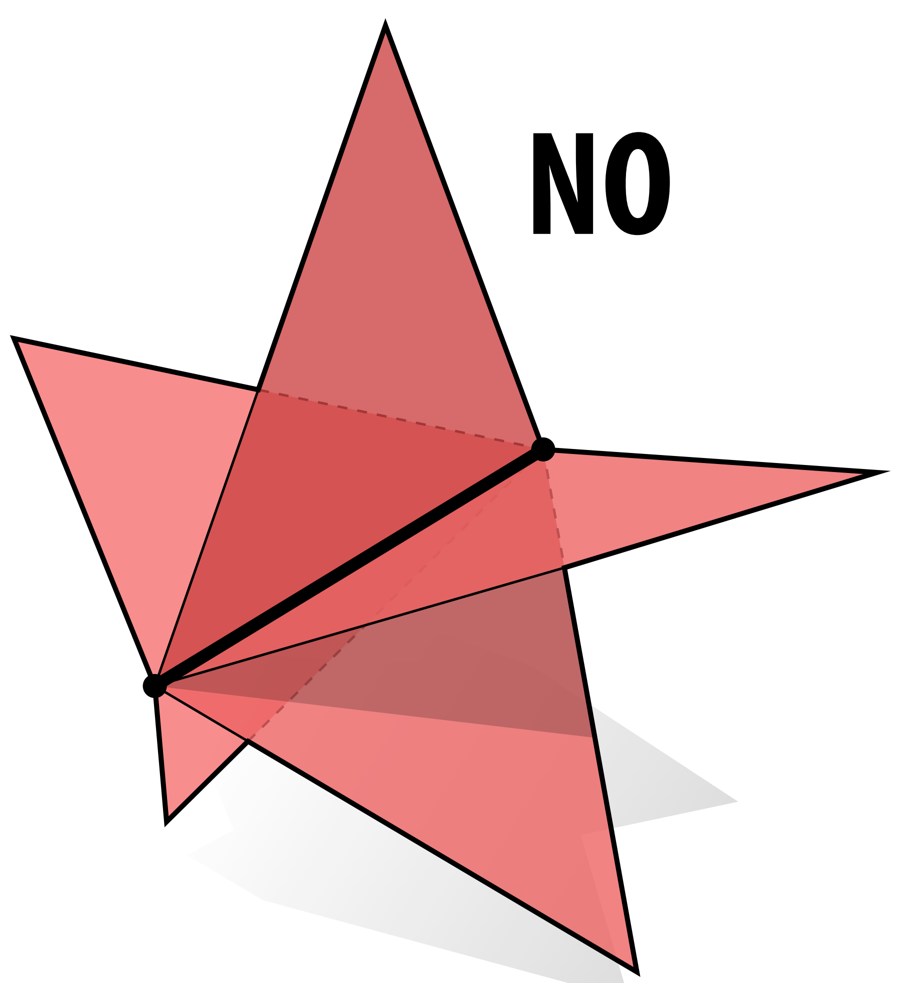
YES



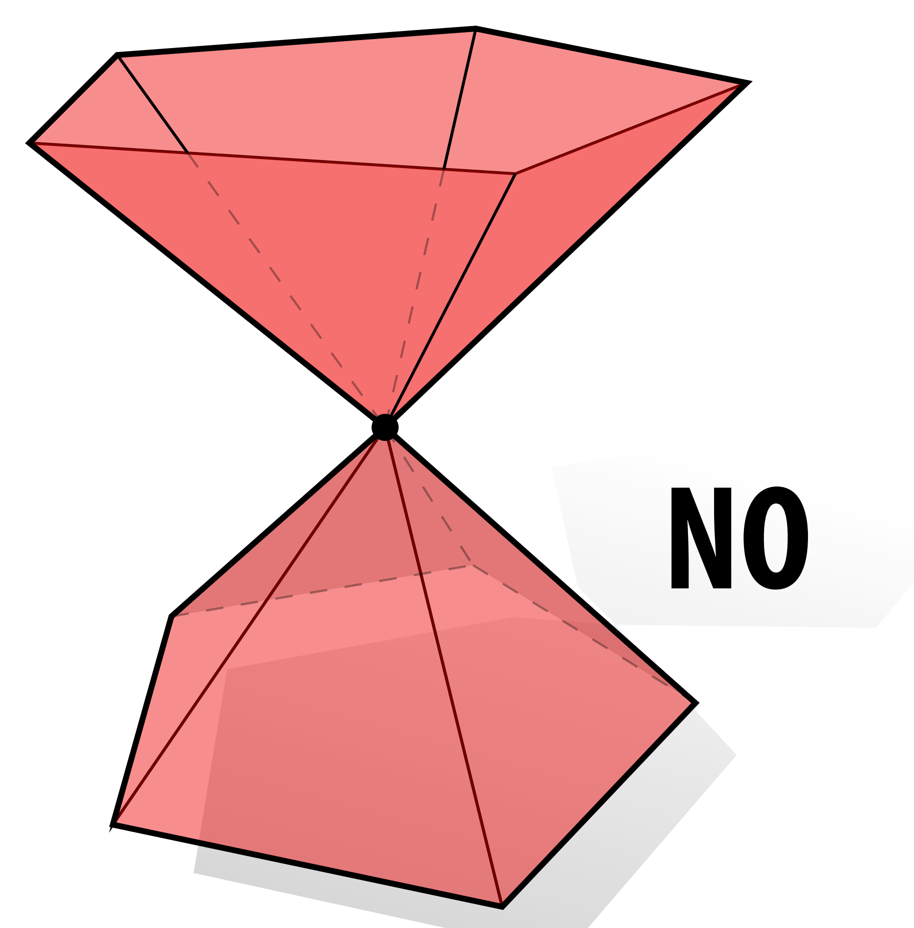
YES



NO

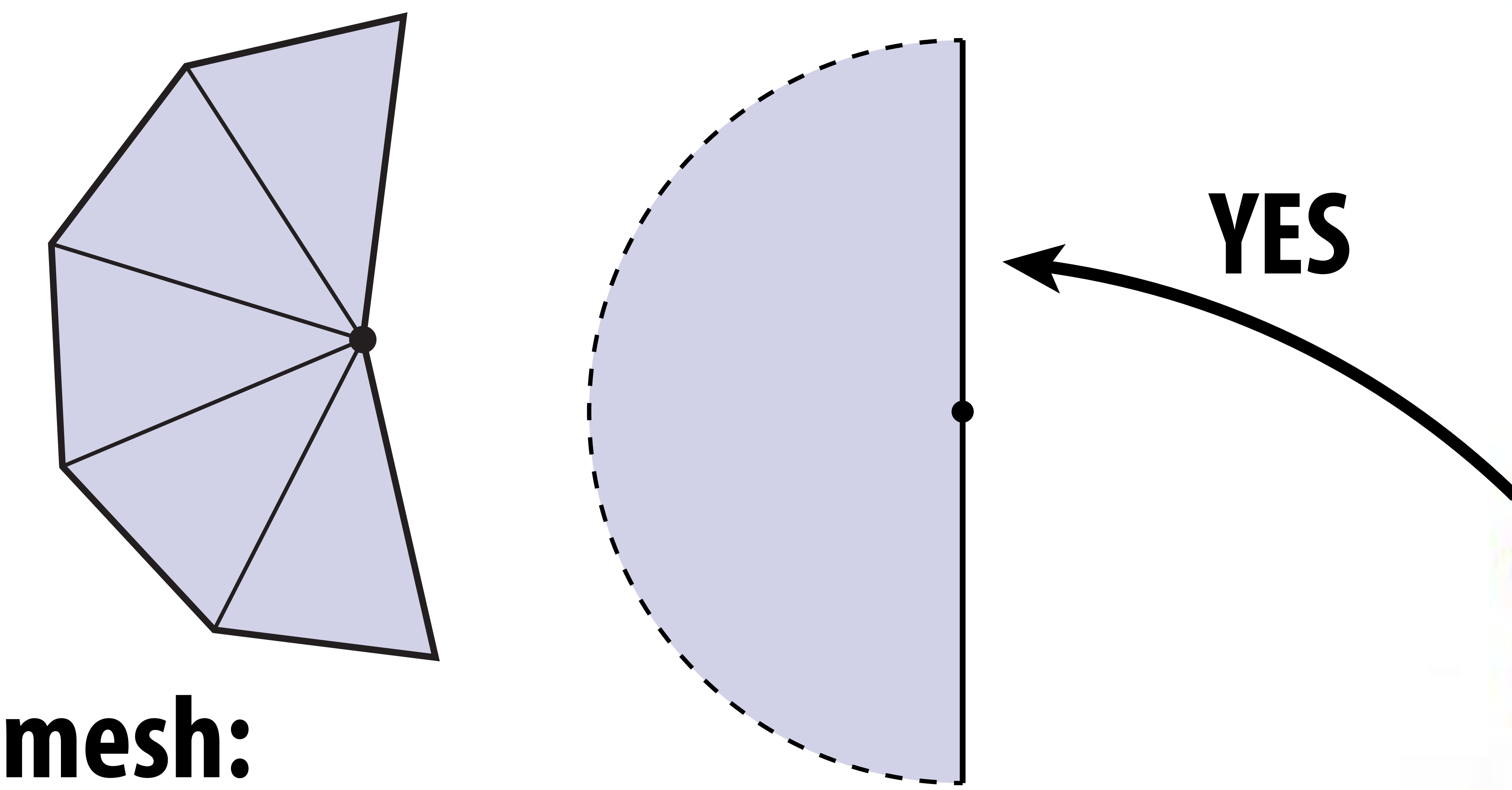


NO



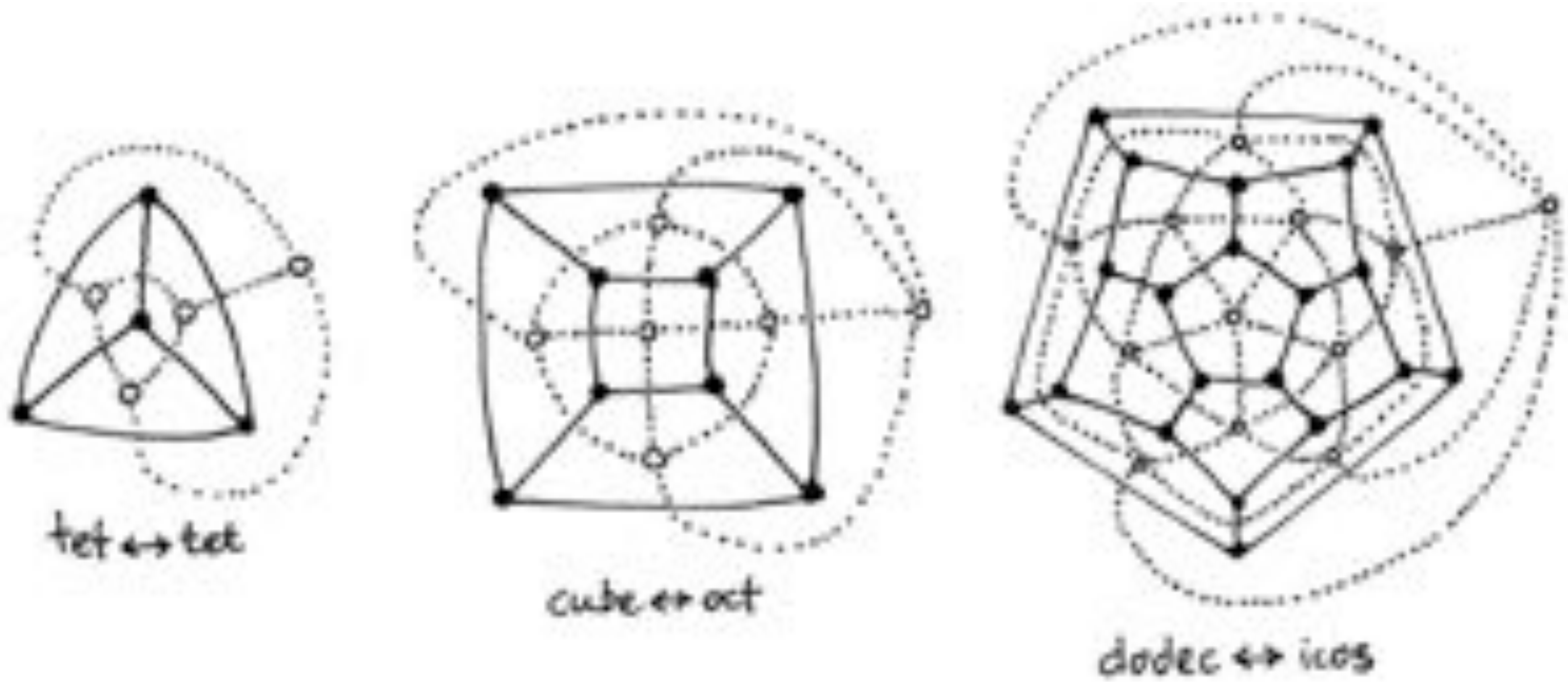
What about boundary?

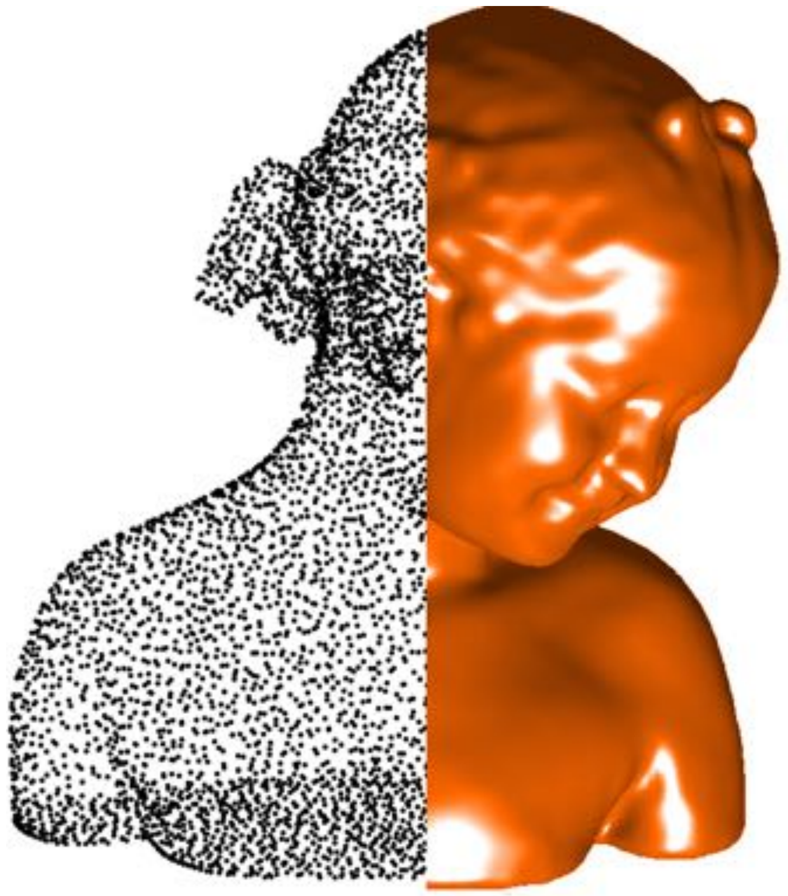
- The boundary is where the surface "ends."
- E.g., waist & ankles on a pair of pants.
- Locally, looks like a half disk
- Globally, each boundary forms a loop



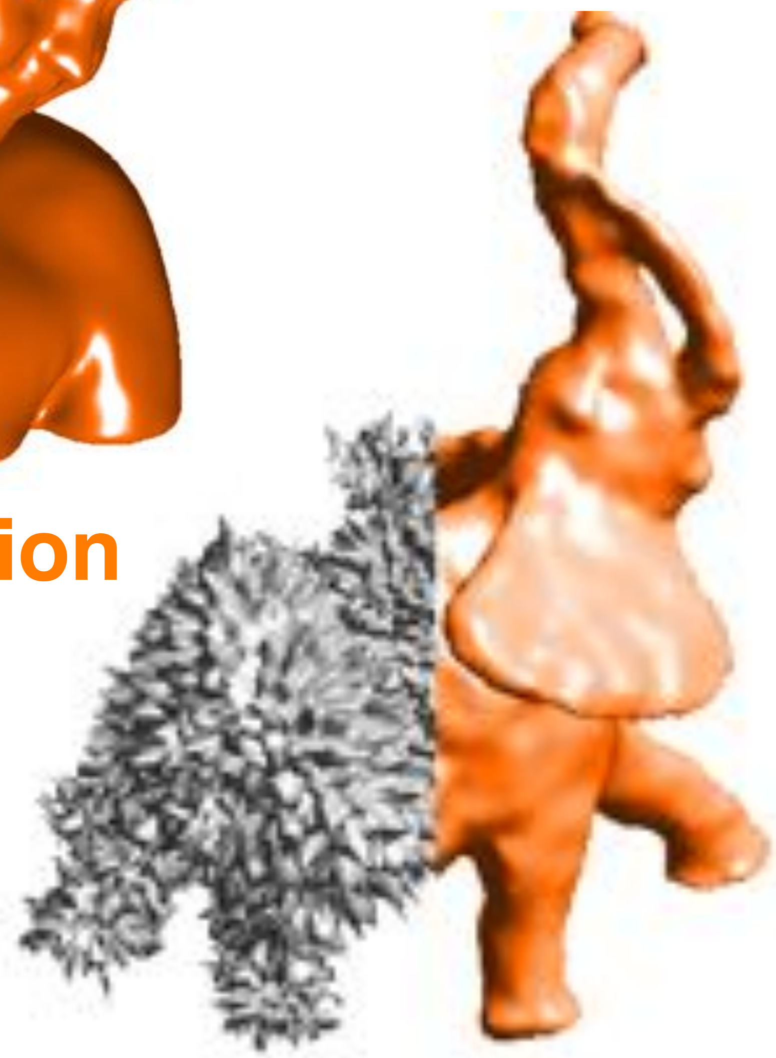
- Polygon mesh:
  - one polygon per boundary edge
  - boundary vertex looks like "pacman"

Next time: Representing and Processing Geometry

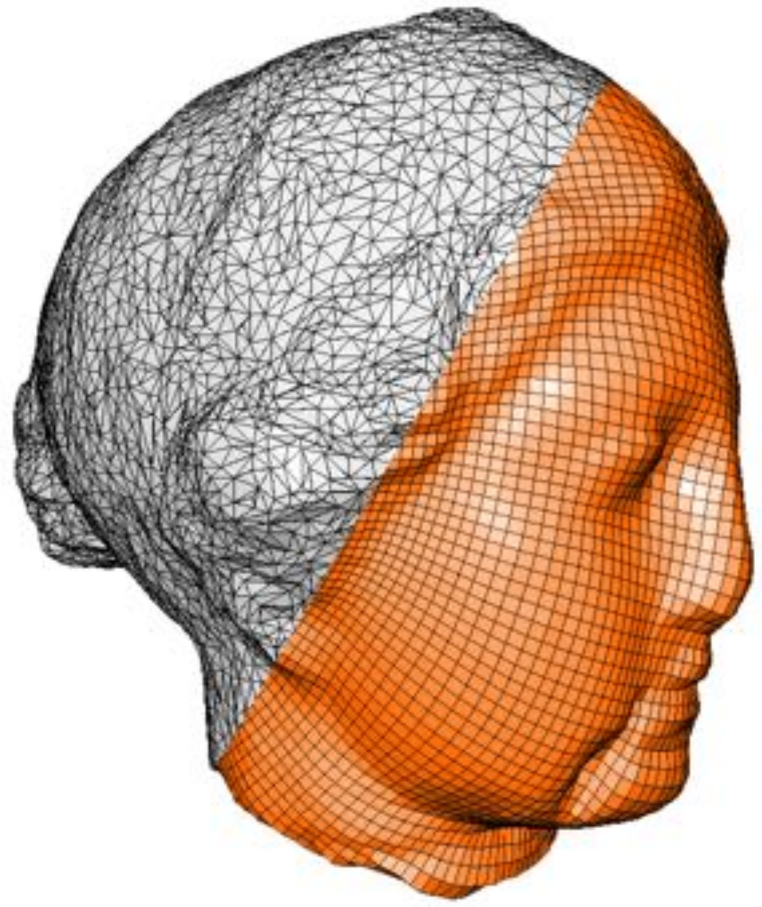




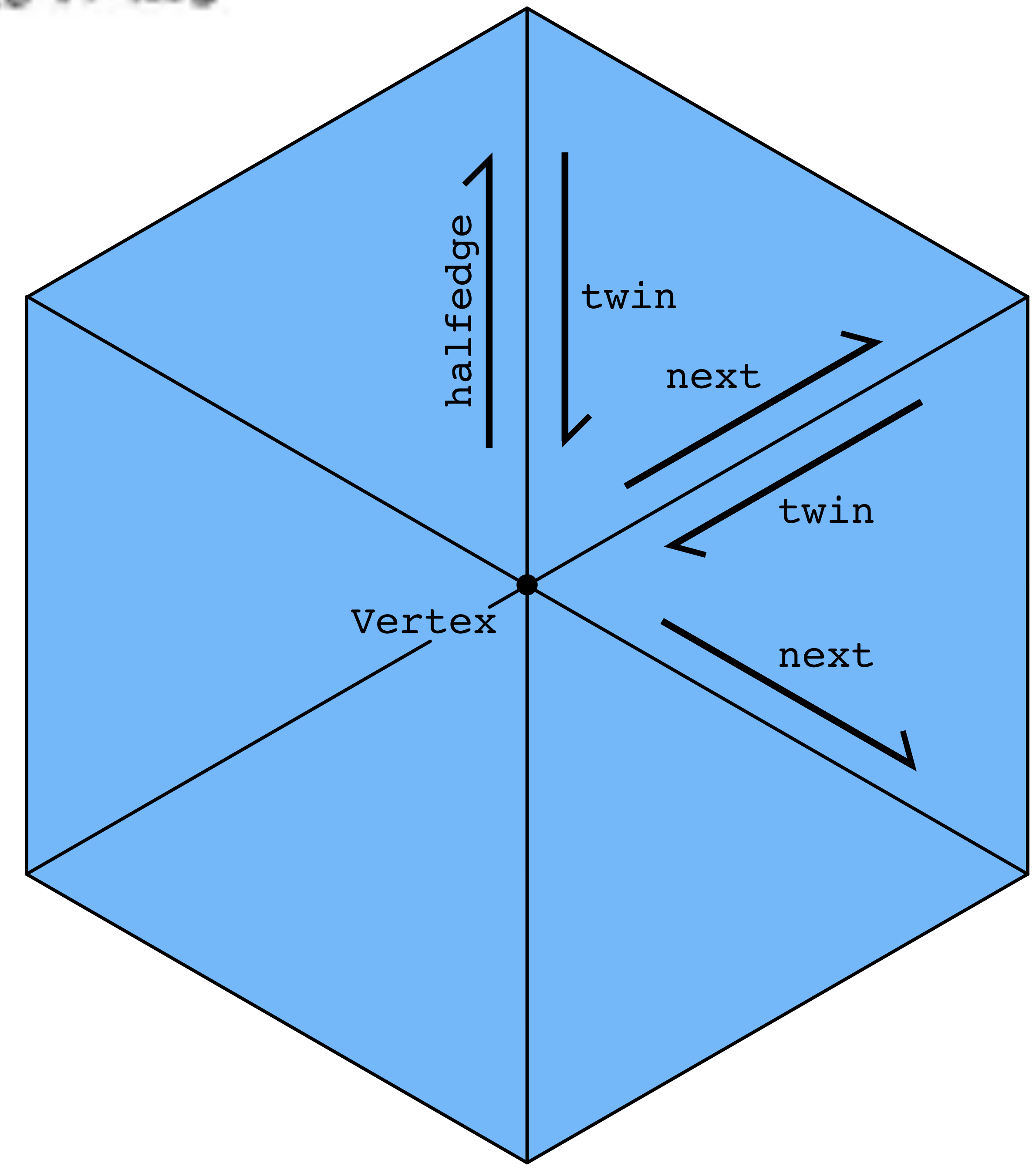
reconstruction



filtering



remeshing



After the midterm