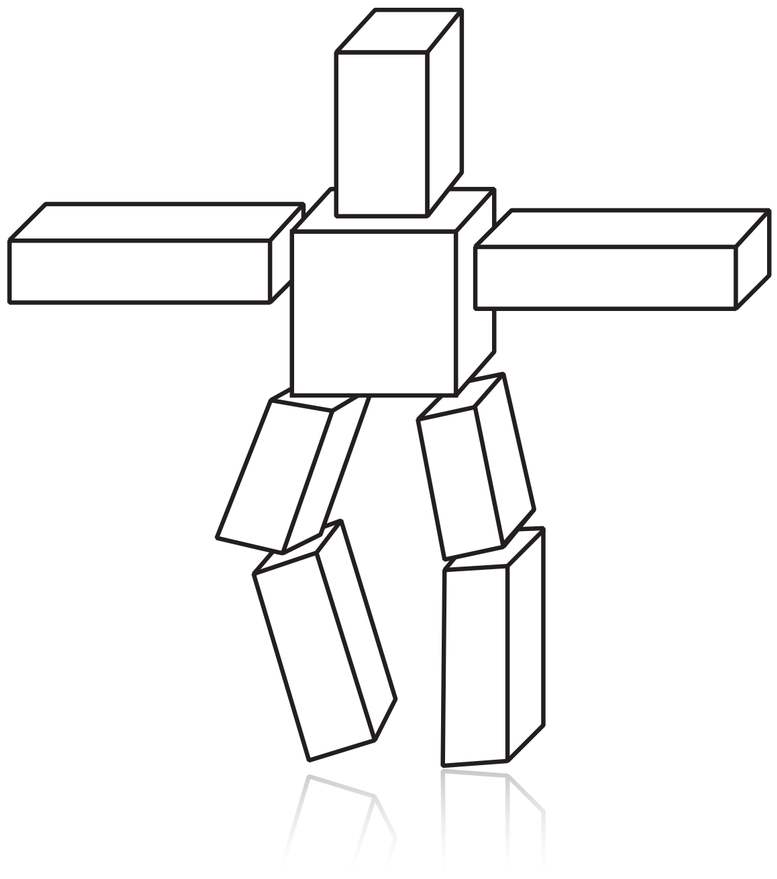
Lecture 6:

Introduction to Geometry

Interactive Computer Graphics
Stanford CS248, Winter 2019

Increasing the complexity of our models

Transformations



Geometry

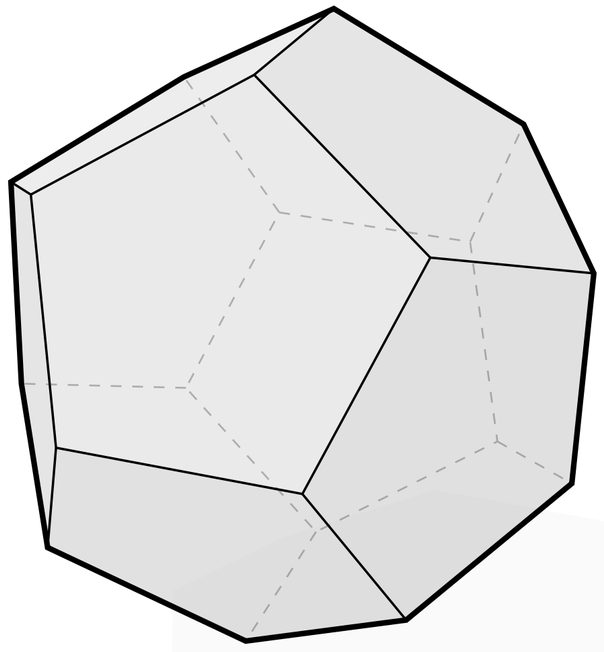
Materials, lighting, ...

What is geometry?

"Earth" "measure"

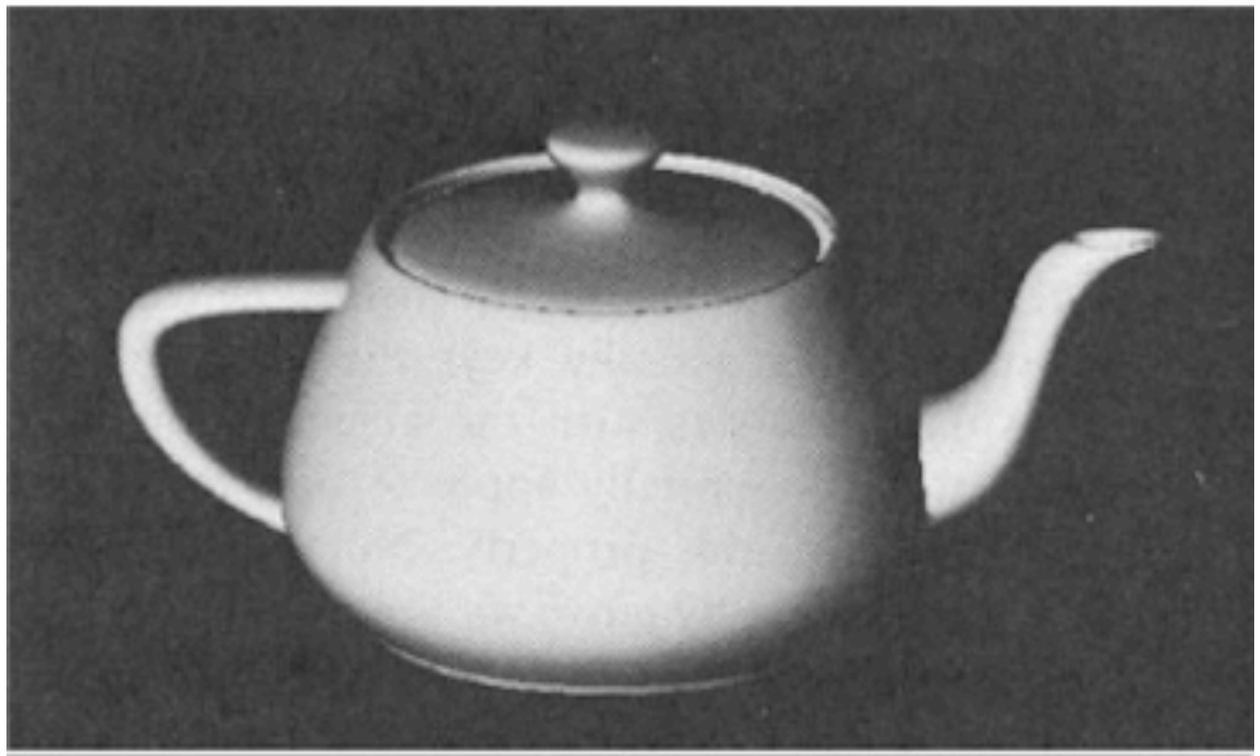
ge•om•et•ry /jēˈämətrē/ *n.*

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..."

Examples of geometry



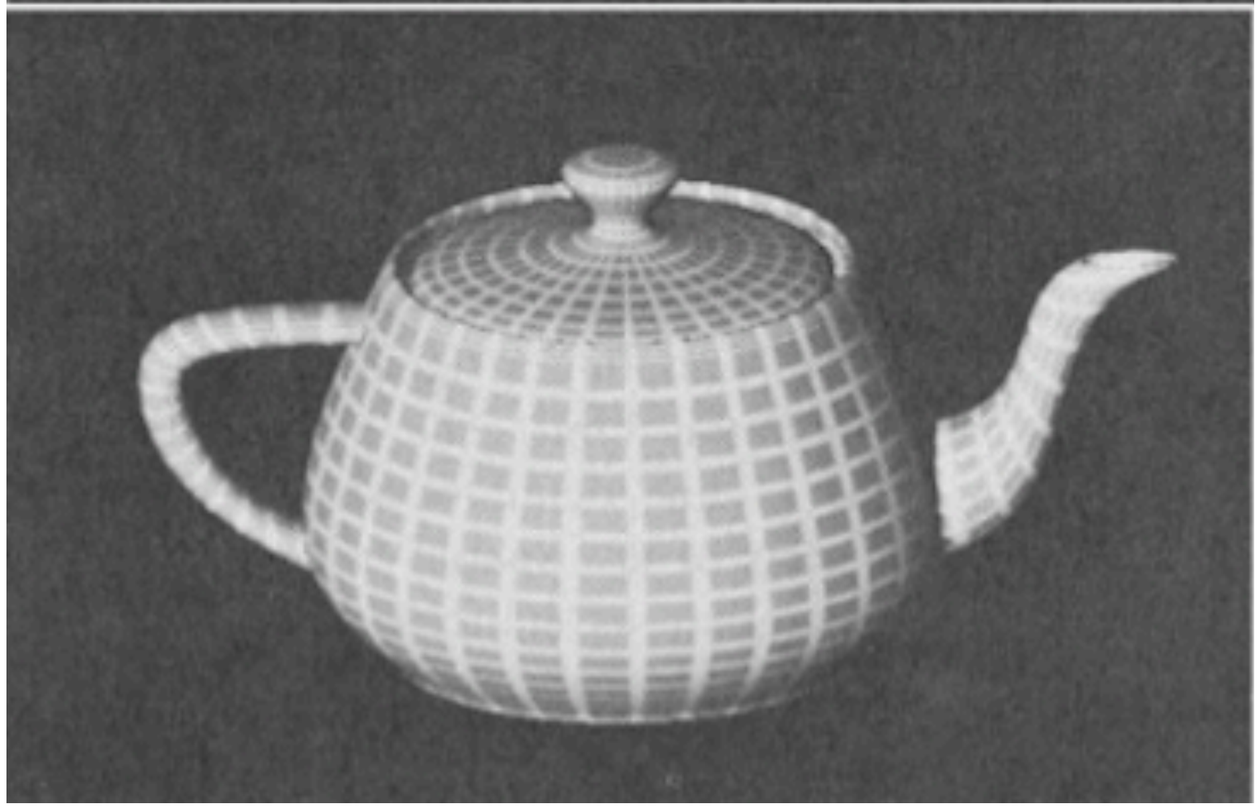
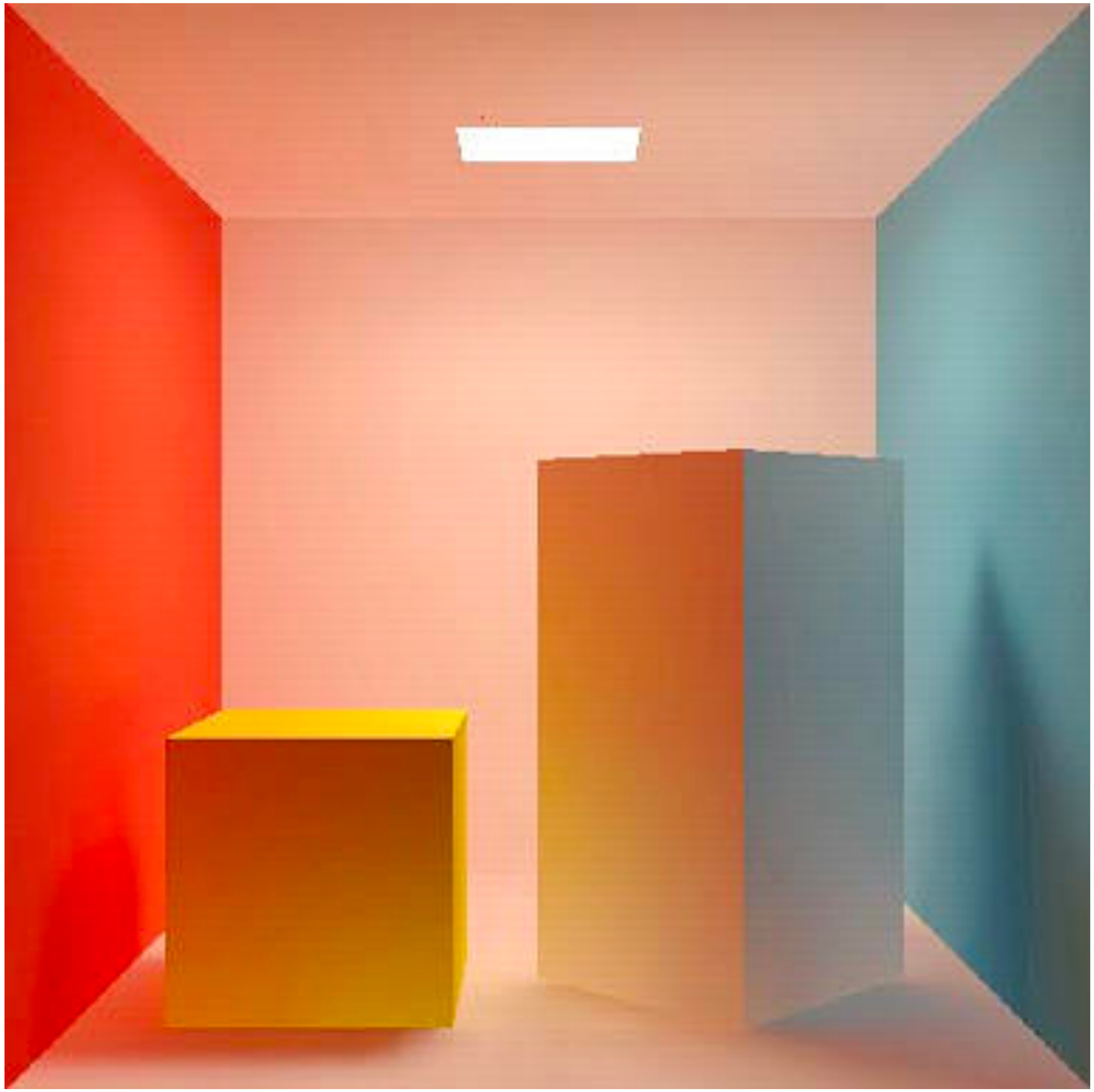


Photo of original Utah teapot
(now sitting in Computer History Museum in Mountain View)

Martin Newell's early teapot renderings
(Martin created teapot model in 1975 using Bezier curves)

Examples of geometry



Cornell Box: Originally created in 1984
(This image was rendered in 1985 by Cohen and Greenberg)

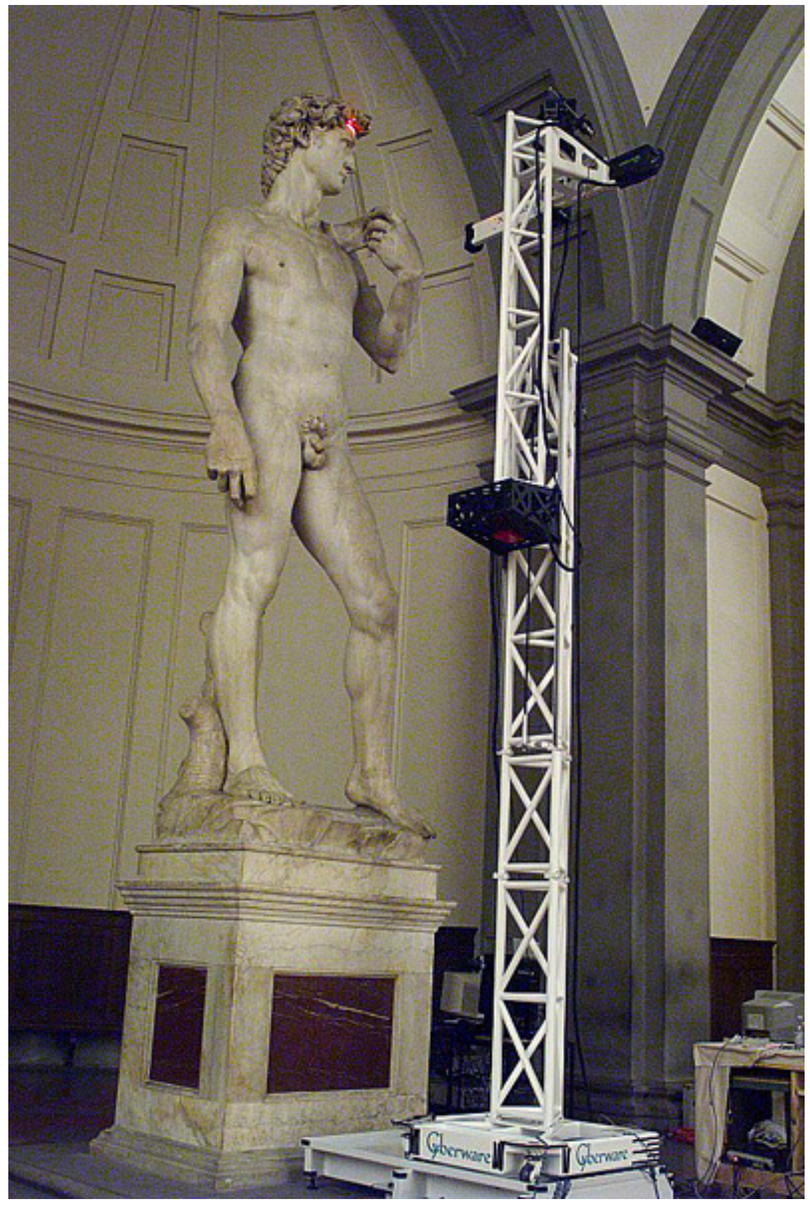
Examples of geometry

The Stanford Bunny
(Mesh created by reconstruction from laser scans)

Photograph of scanned statue
(Statue purchased by Greg Turk at a store on University Ave in 1994)

Examples of geometry

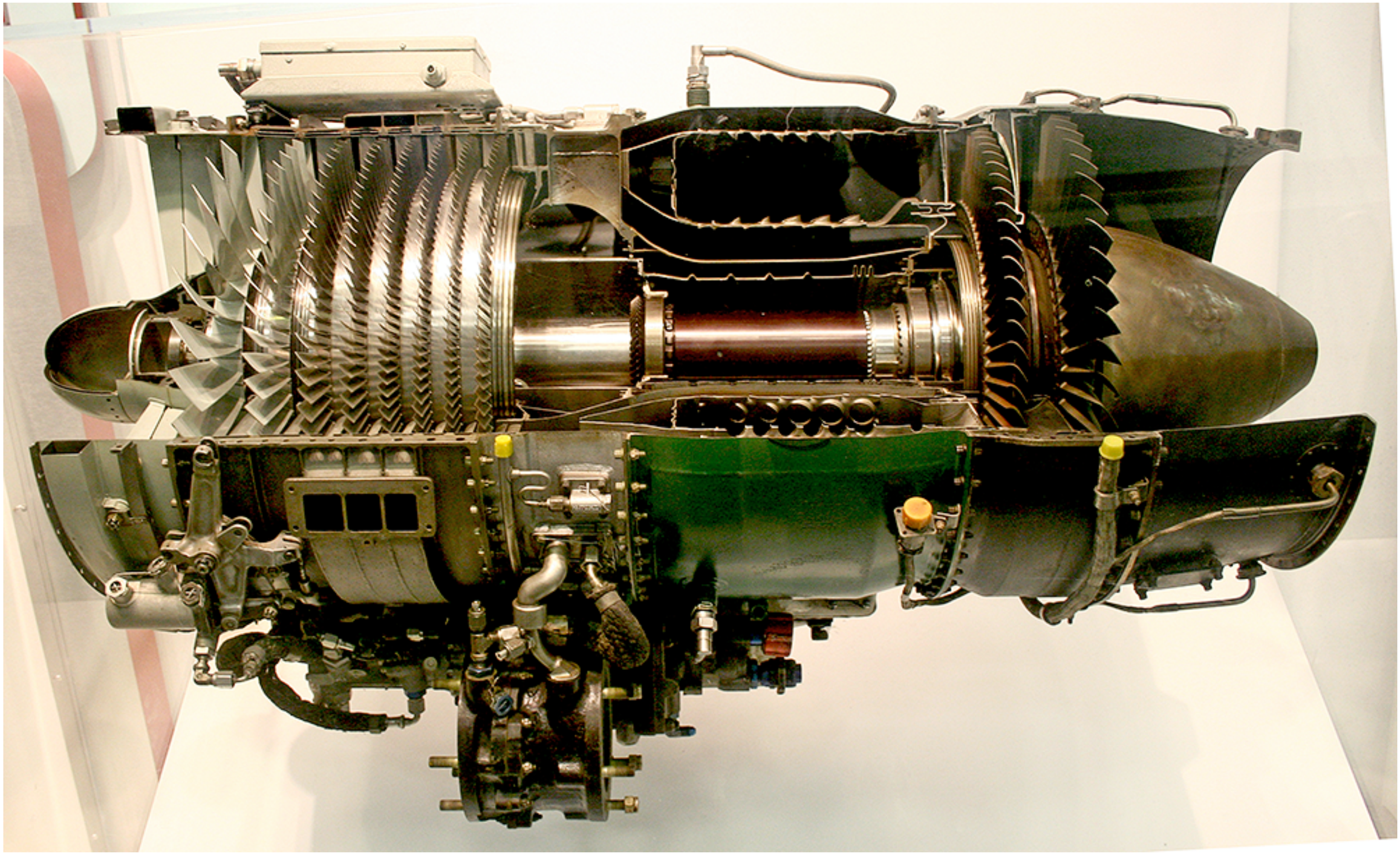
Laser scan of Michelangelo's David (created as part of Stanford's Digital Michelangelo project in 1999)



Examples of geometry

Curly hair in Pixar's "Brave" (2012)

Examples of geometry



Examples of geometry

Examples of geometry

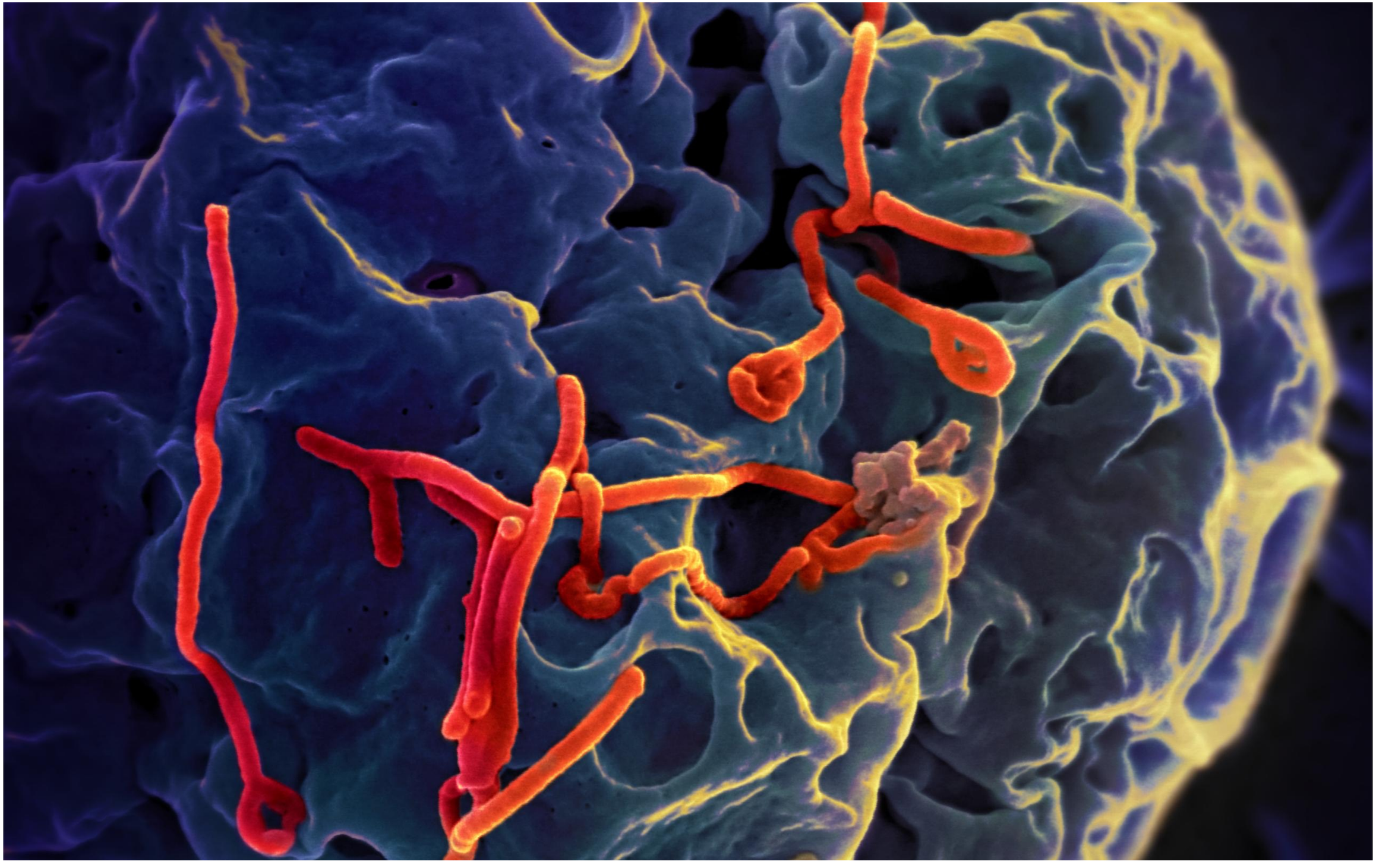
Examples of geometry

Examples of geometry

Examples of geometry

Examples of geometry

Examples of geometry



What's the best way to encode geometry on a computer?

No one "best" choice—geometry is hard!

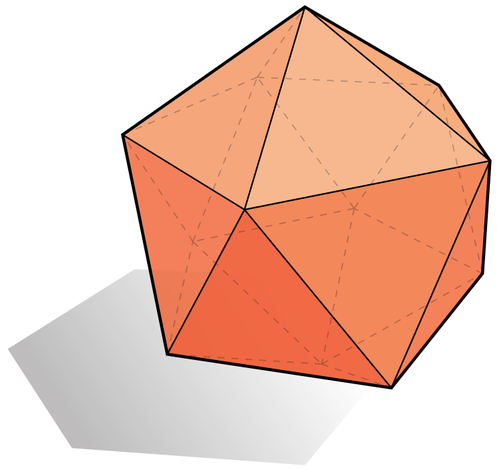
"I hate meshes.
I cannot believe how hard this is.
Geometry is hard."

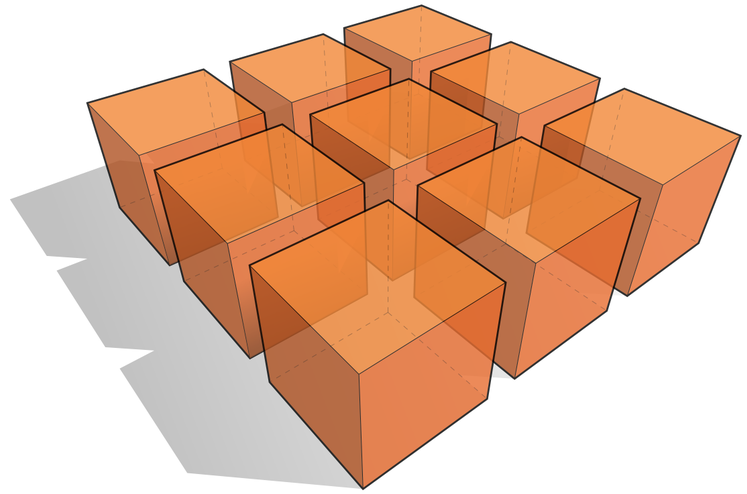
—David Baraff

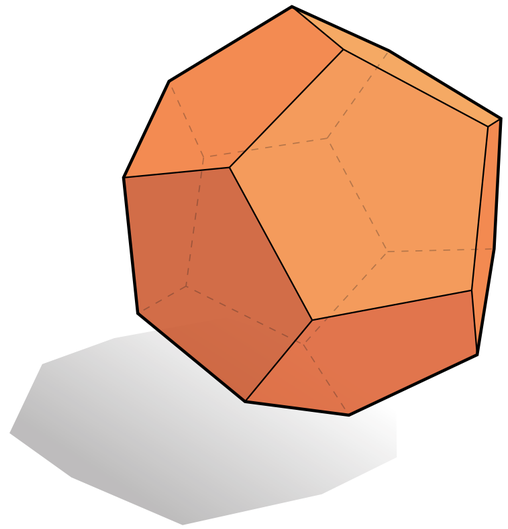
Senior Research Scientist
Pixar Animation Studios

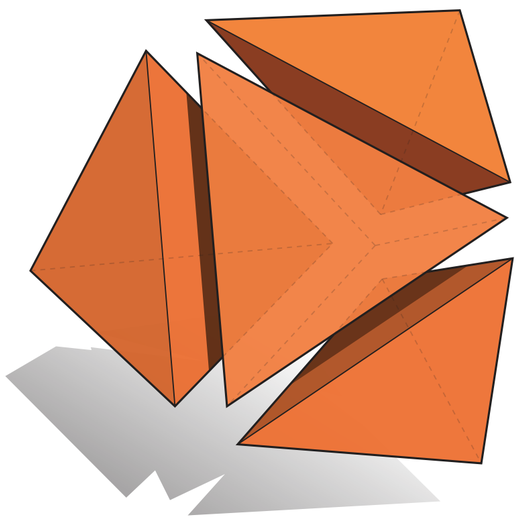
Many ways to digitally encode geometry

- **EXPLICIT**
  - point cloud
  - polygon mesh
  - subdivision, NURBS
  - …
- **IMPLICIT**
  - level set
  - algebraic surface
  - L-systems
  - …





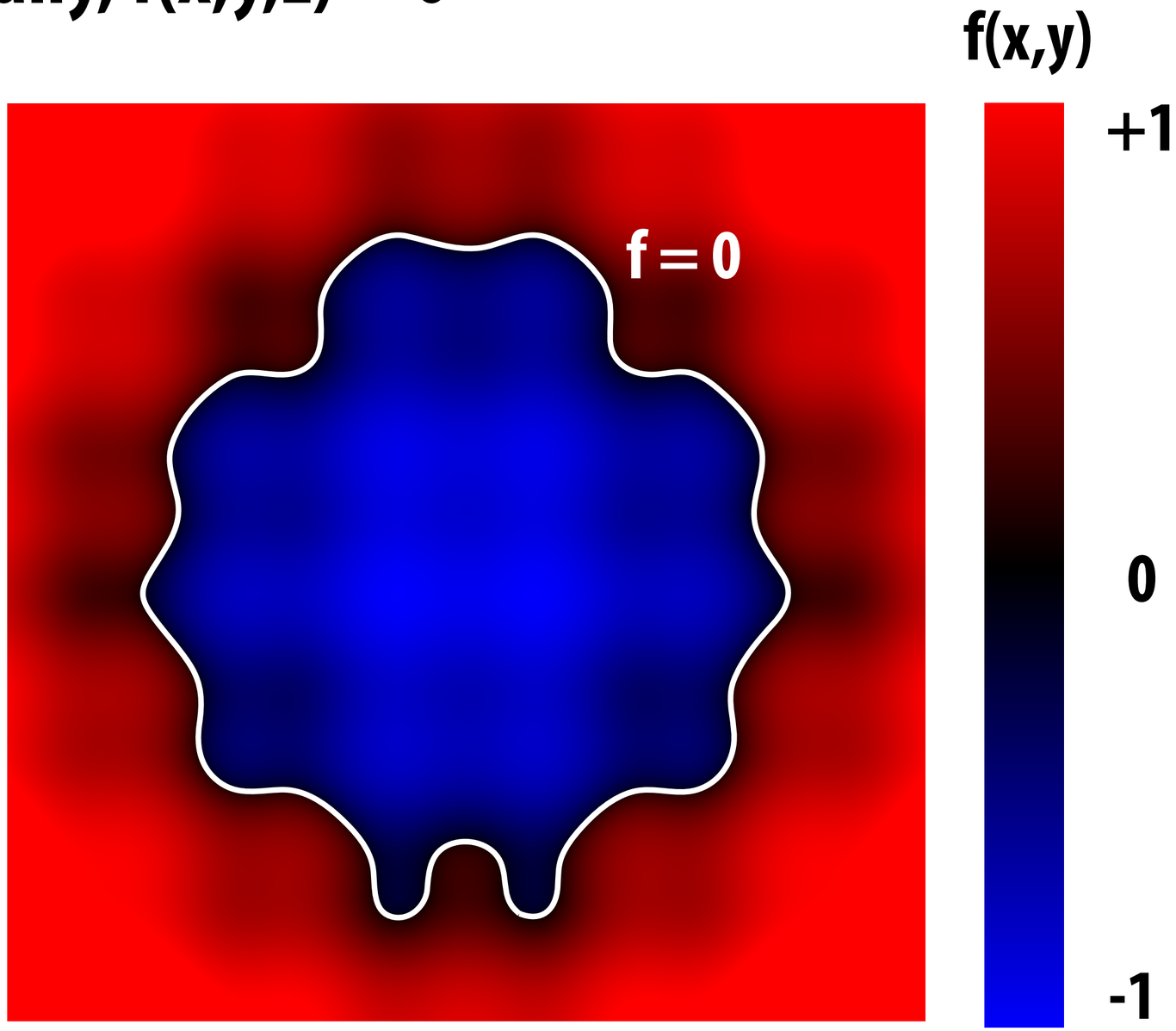




- **Each choice best suited to a different task/type of geometry**

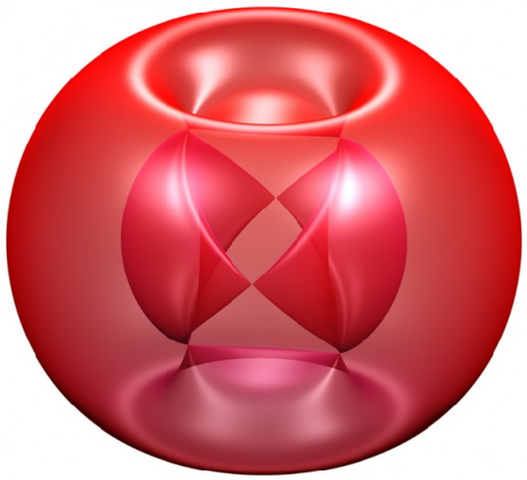
“Implicit” representations of geometry

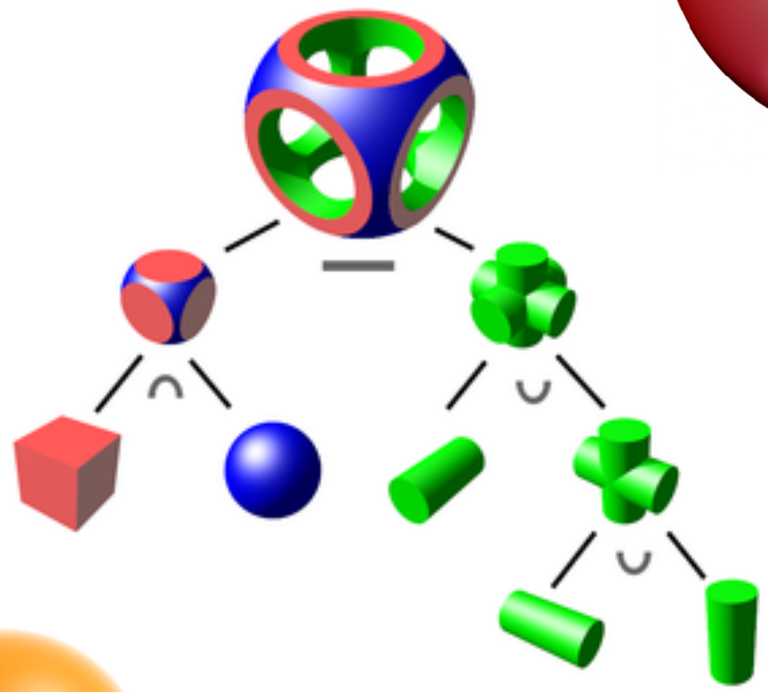
- Points aren’t known directly, but satisfy some relationship
- E.g., unit sphere is all points such that $x^2+y^2+z^2=1$
- More generally, $f(x,y,z) = 0$

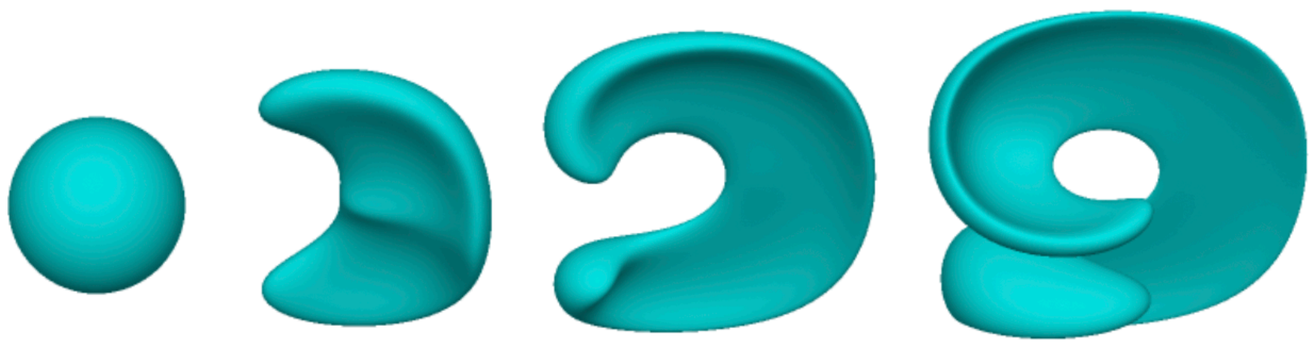


Many implicit representations in graphics

- 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:

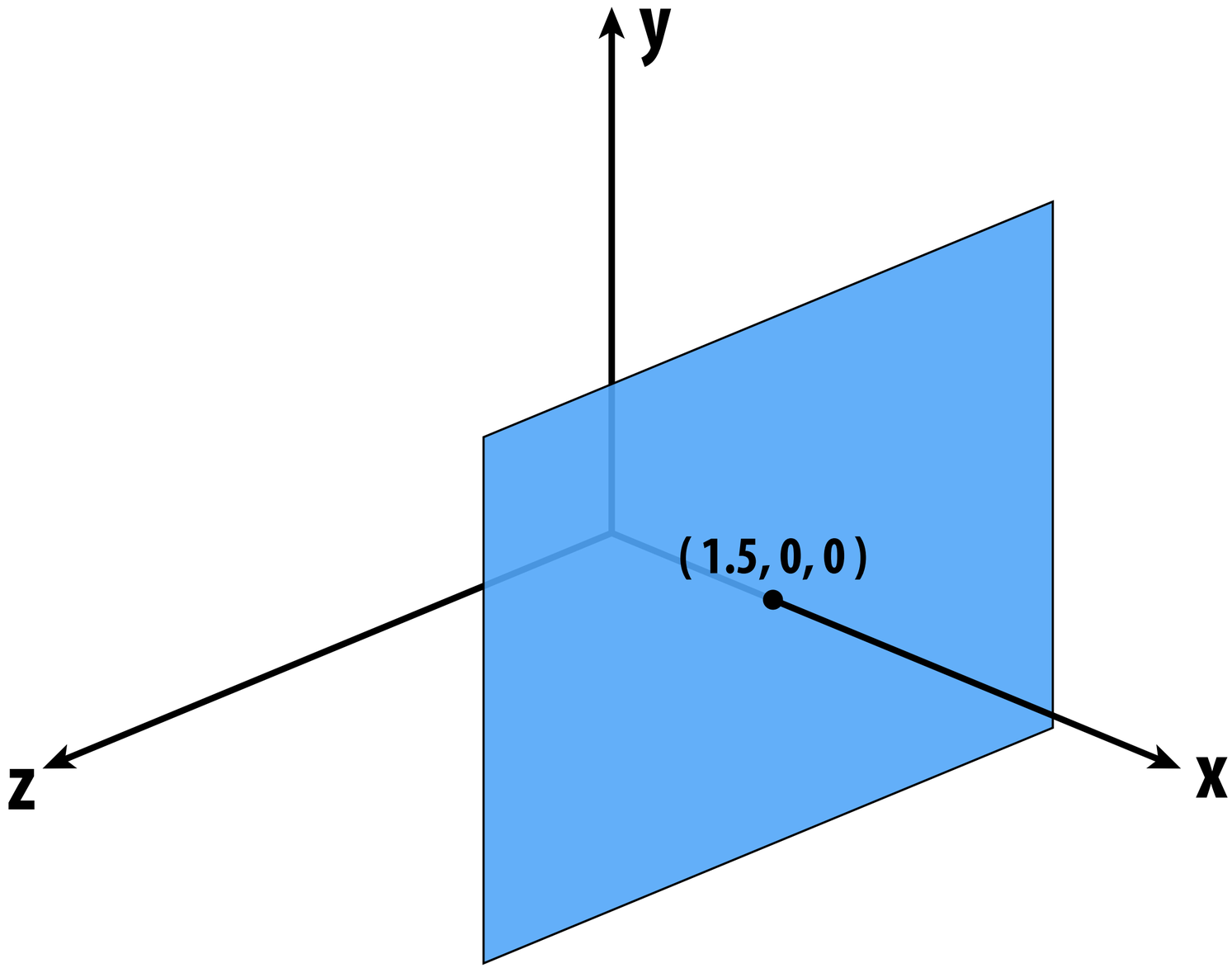
I'm thinking of an implicit surface

$f(x,y,z)=0$

Find *any* point on it.

Give up?

My function was f(x,y,z) = x - 1.5 (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.

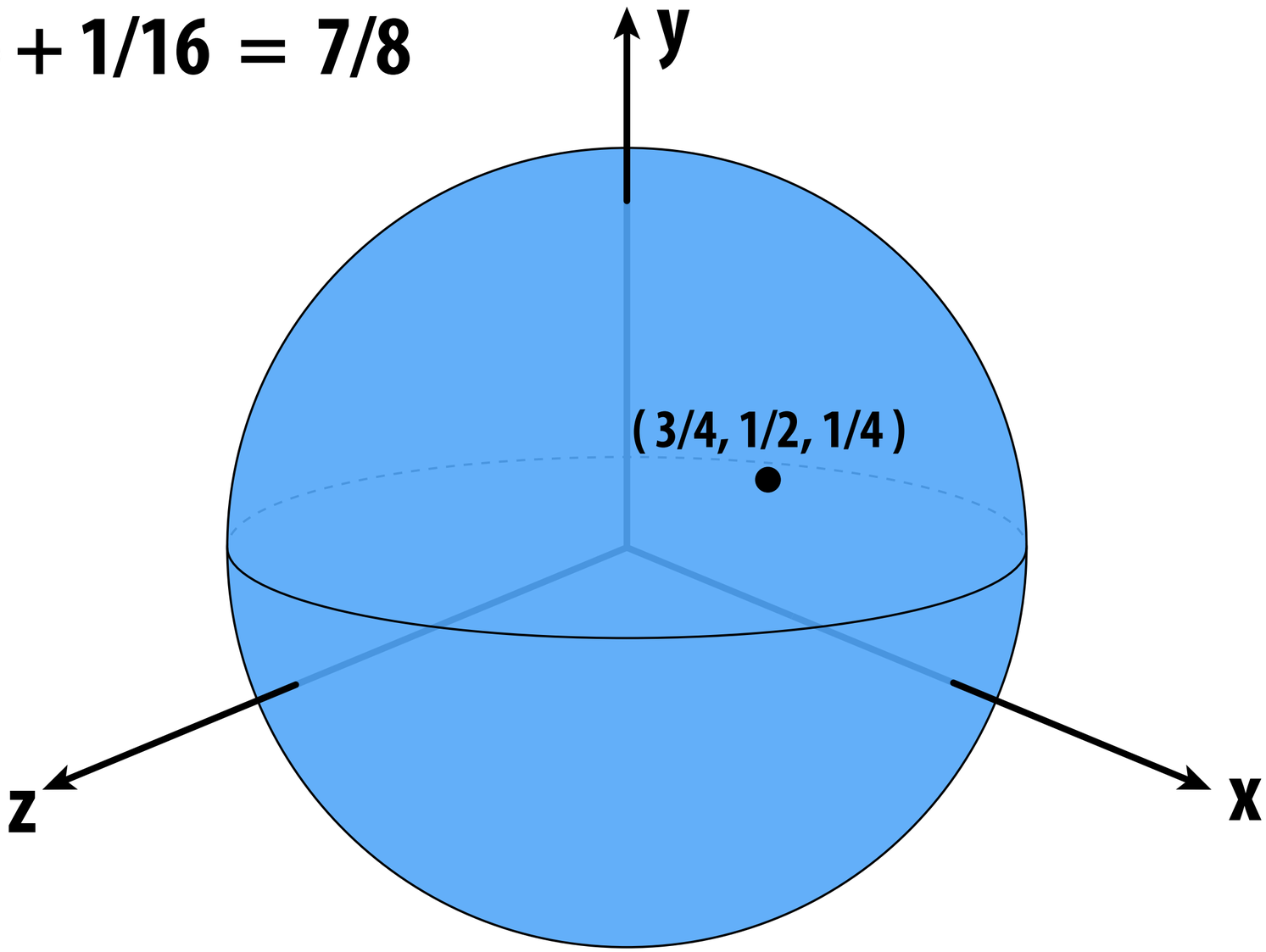
Check if this point is inside the unit sphere

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

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

$$7/8 < 1$$

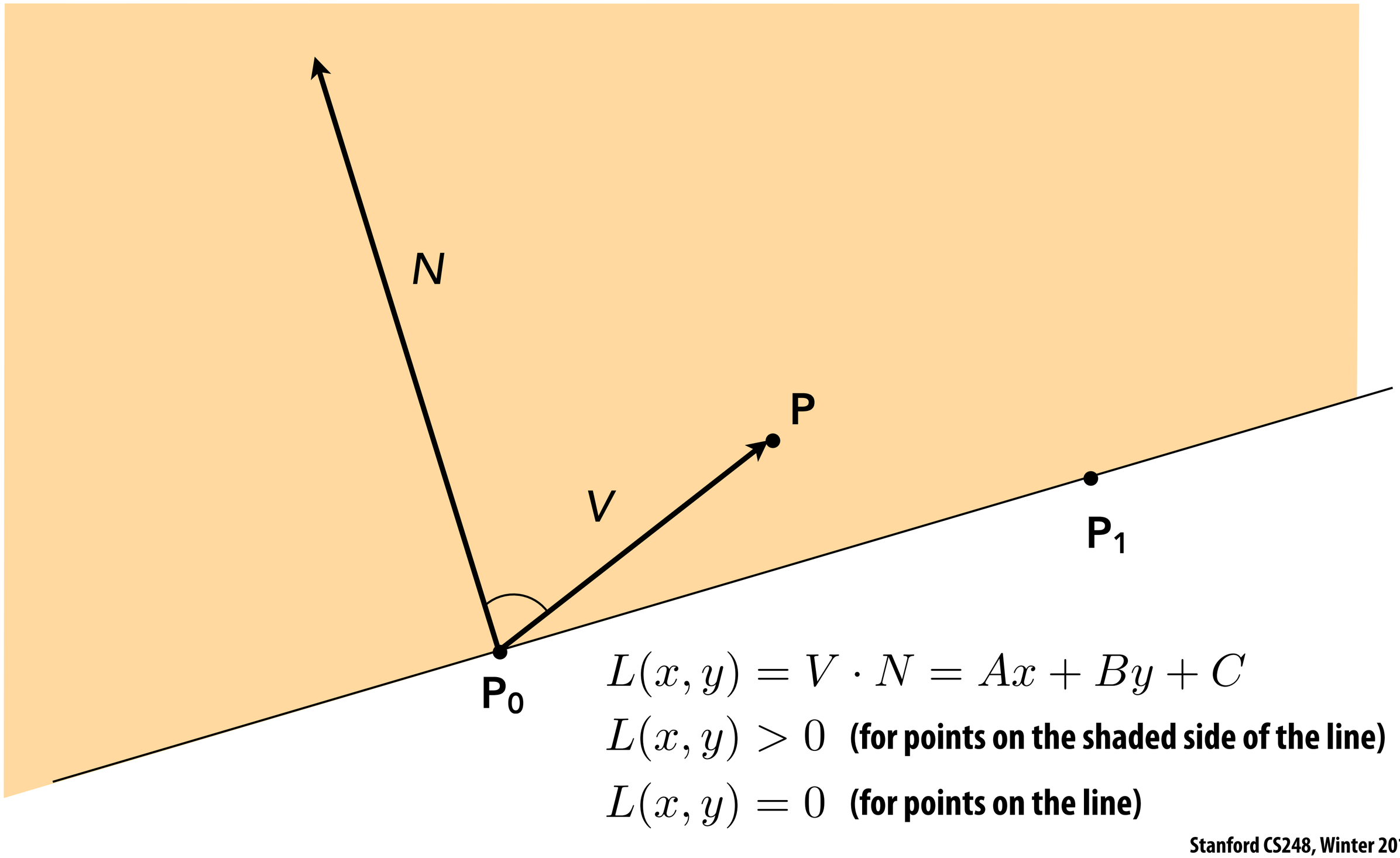
YES.



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

Recall: implicit form of a line

- Easy to test if a point is on the "positive" or negative side of the line



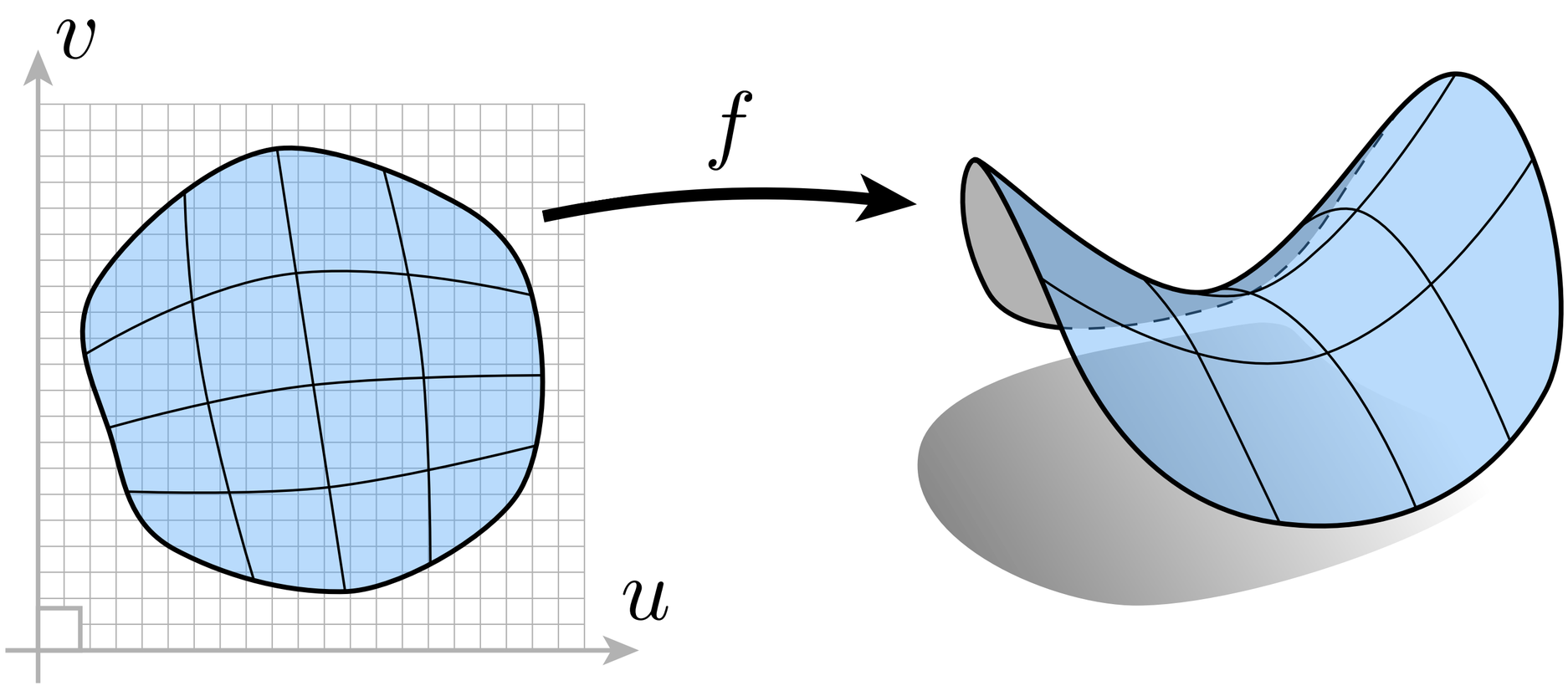
$$L(x,y) = V \cdot N = Ax + By + C$$

$L(x,y) > 0$ **(for points on the shaded side of the line)**

$L(x,y) = 0$ **(for points on the line)**

"Explicit" representations of geometry

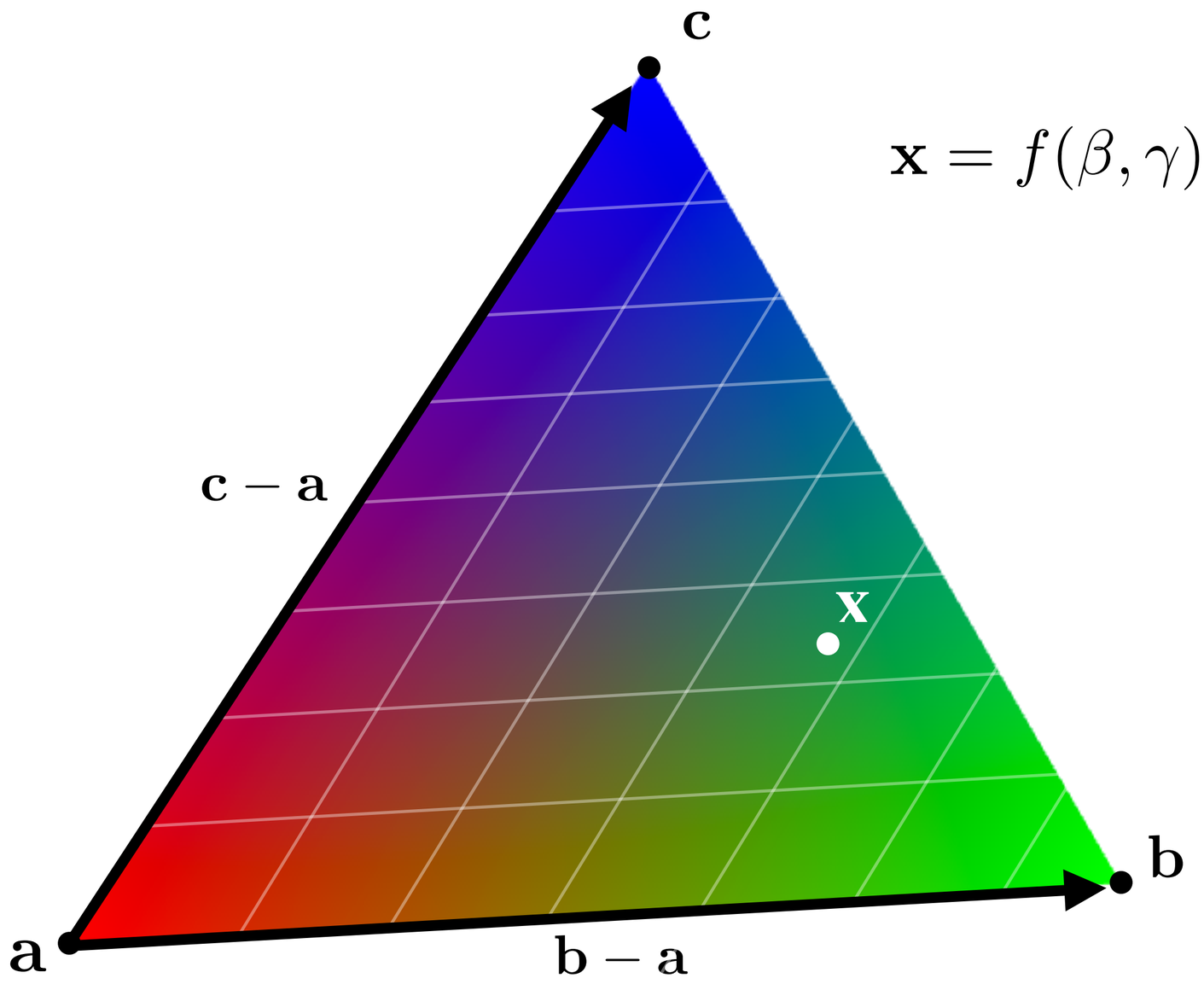
- **All points are given directly**
- **E.g., points on sphere are** $(\cos(u)\sin(v), \sin(u)\sin(v), \cos(v))$,
  $\text{for } 0 \leq u < 2\pi \text{ and } 0 \leq v \leq \pi$
- **More generally:** $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



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

“Explicit” representations of geometry

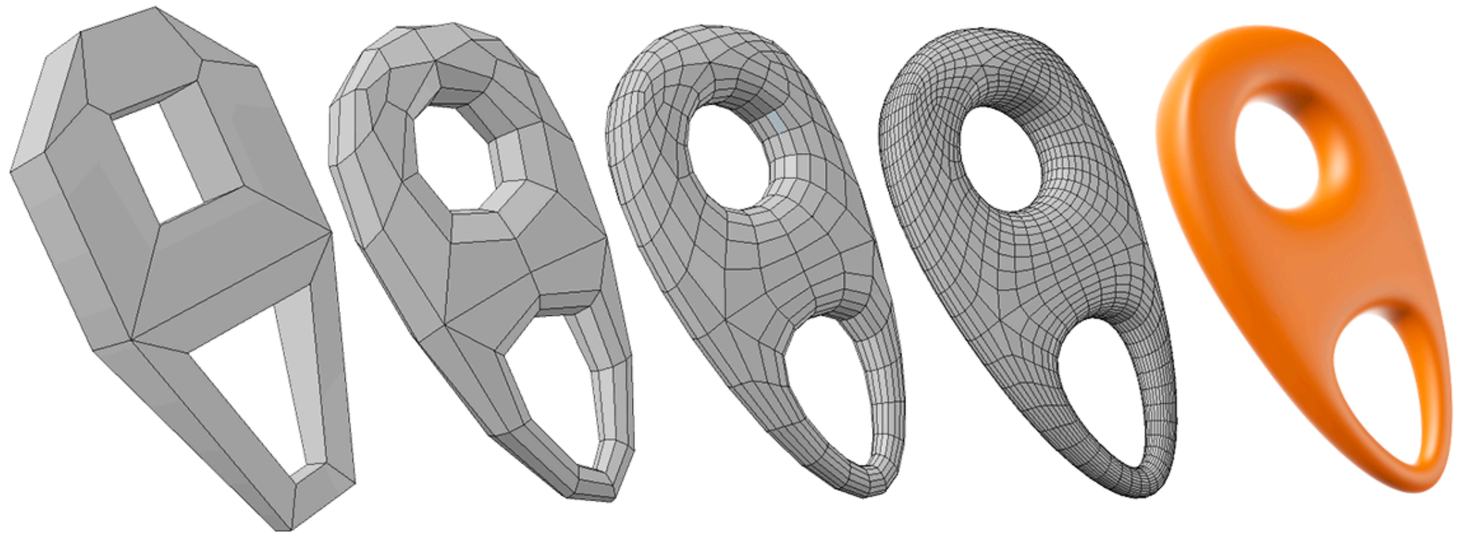
- **More generally:** $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$
- **Example: a triangle**

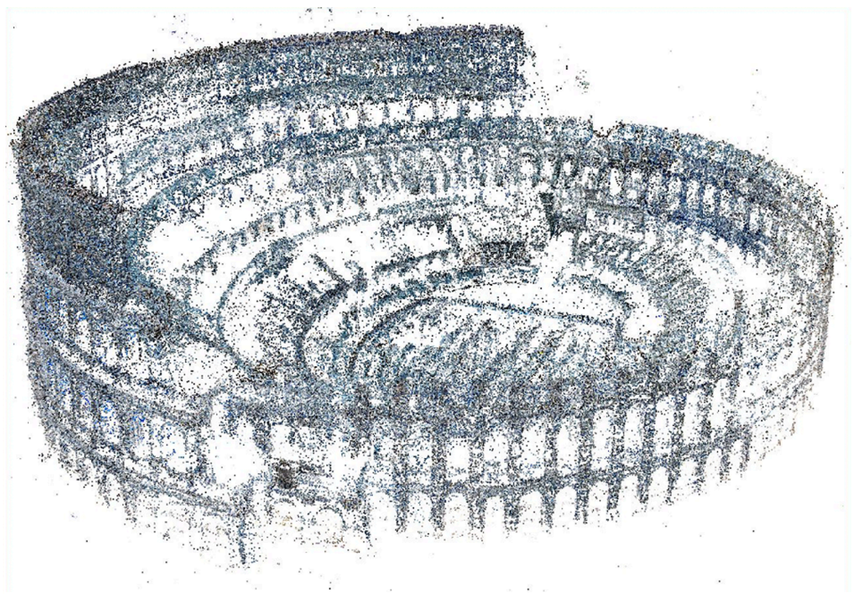


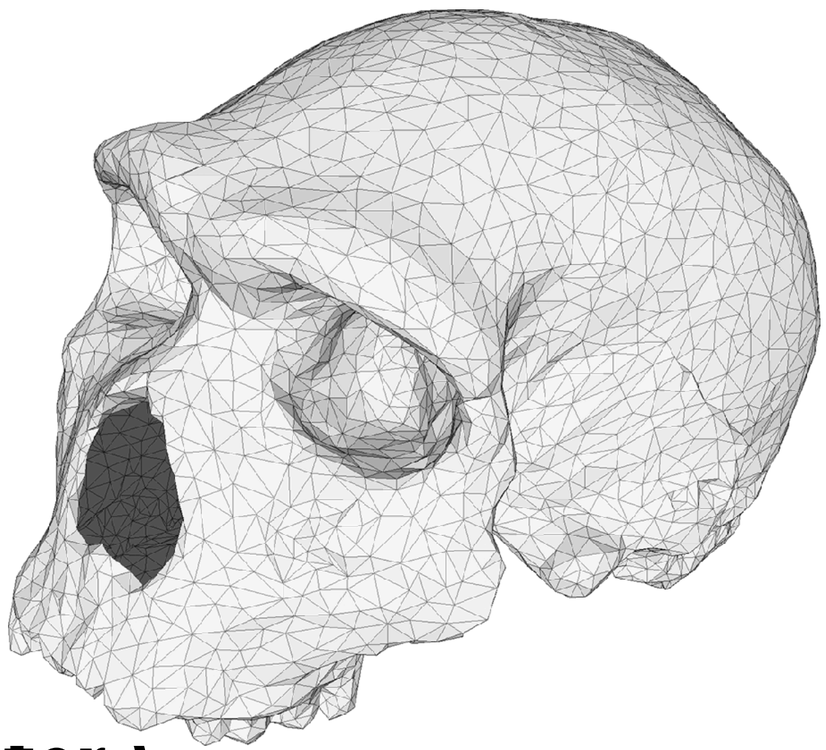
$$\mathbf{x} = f(\beta, \gamma) = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

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:

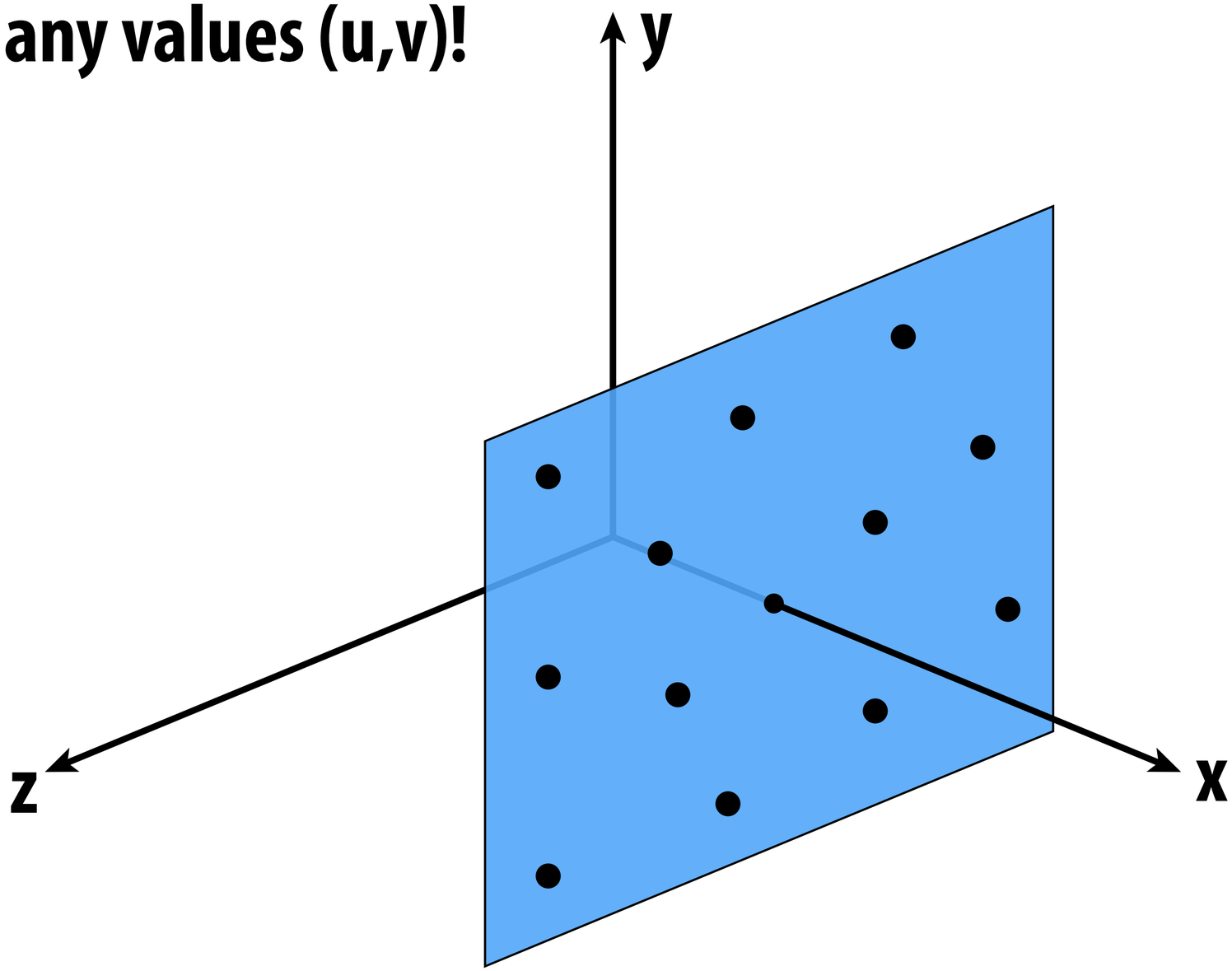
I'll give you an explicit surface.

You give me some points on it.

Sampling an explicit surface

My surface is f(u, v) = (1.5, 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).

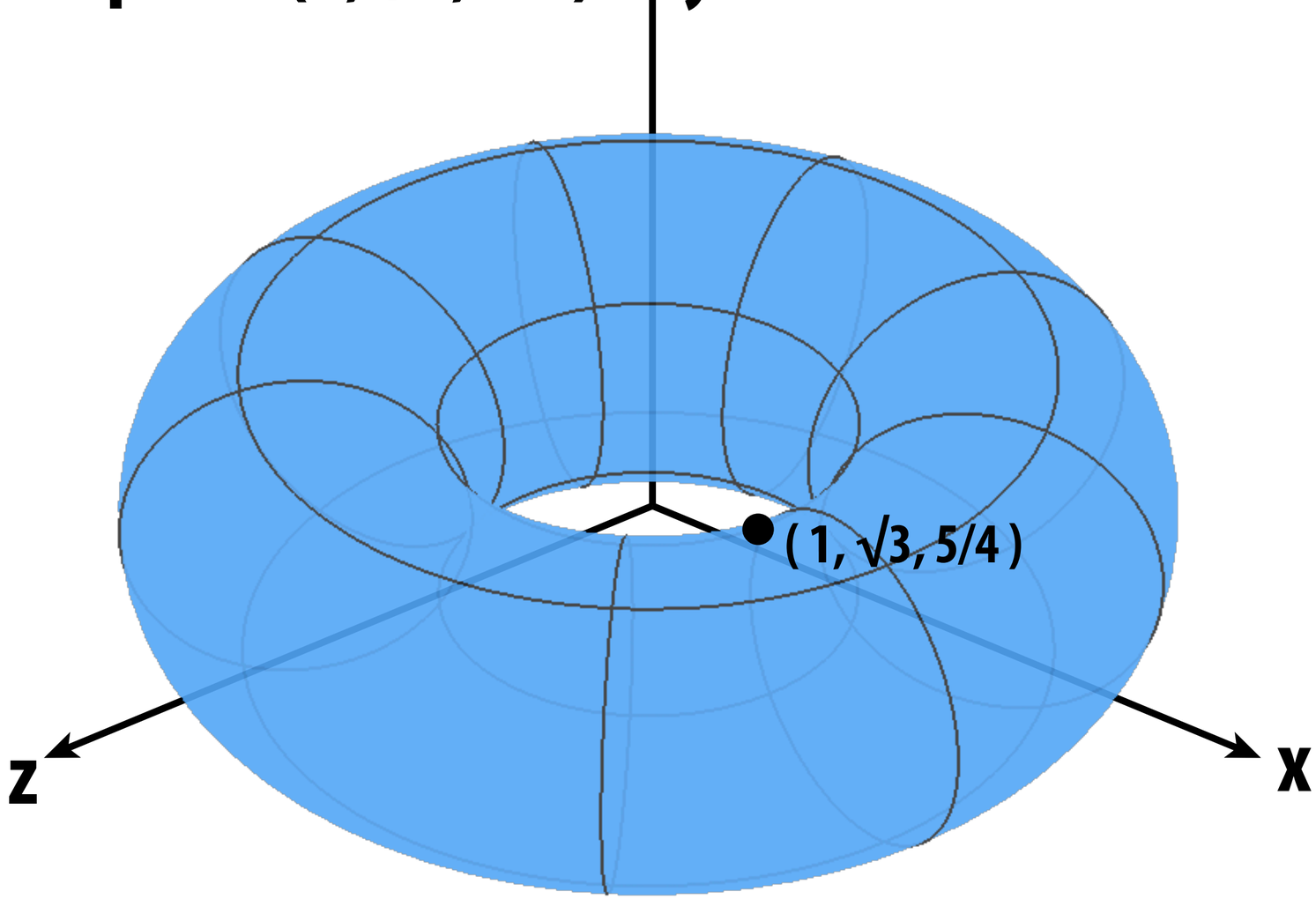
I want to see if a point is *inside* it.

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,√3,5/4)?

...NO!



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

CONCLUSION:

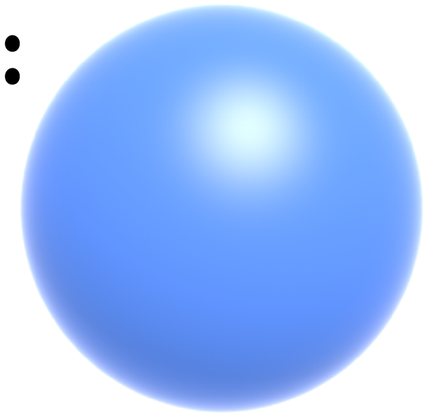
Some representations work better than others—depends on the task!

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

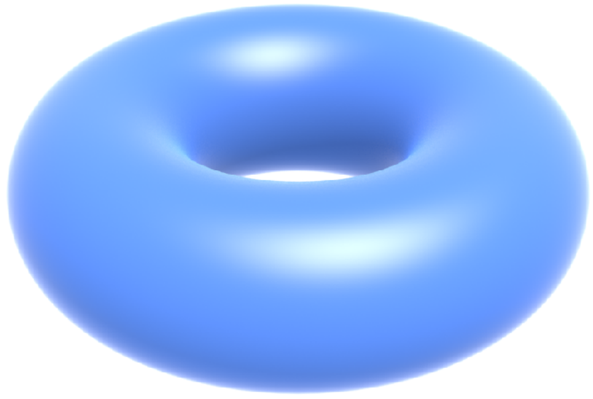
Let's take a look at some common representations used in computer graphics.

Algebraic surfaces (implicit)

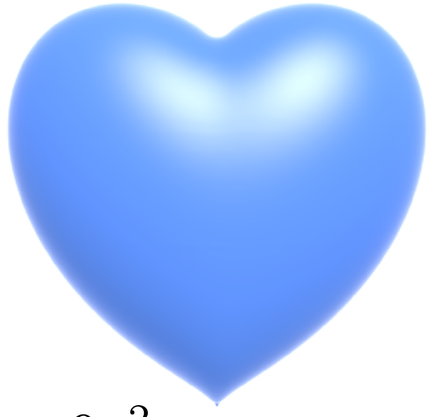
- Surface is zero set of a polynomial in x, y, z ("algebraic variety")
- Examples:



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



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



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

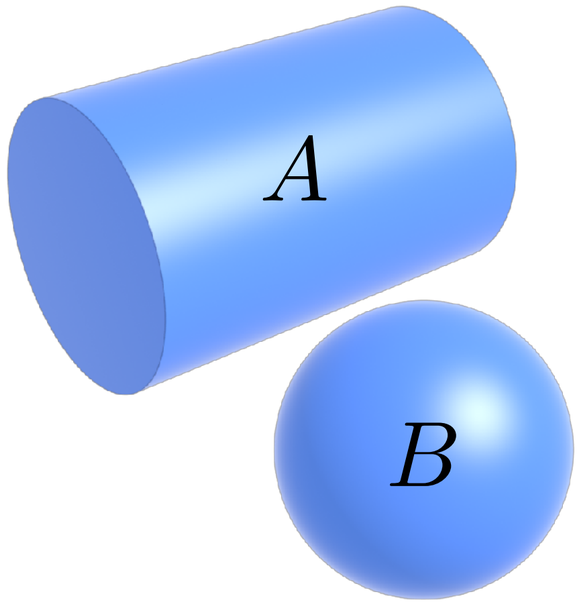
- What about more complicated shapes?

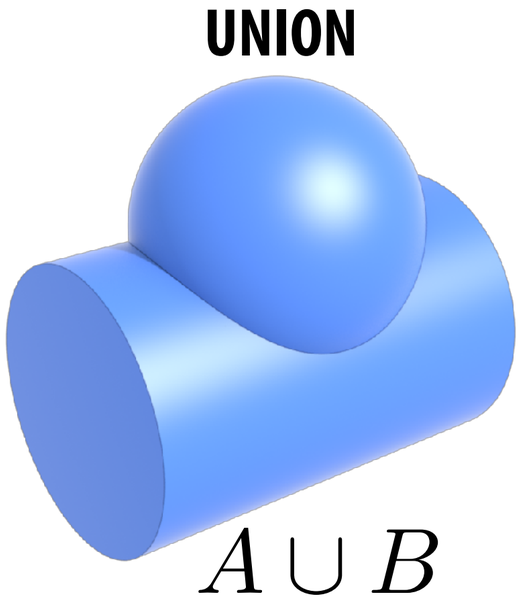
$$x^2z^3+\frac{9y^2z^3}{80}$$

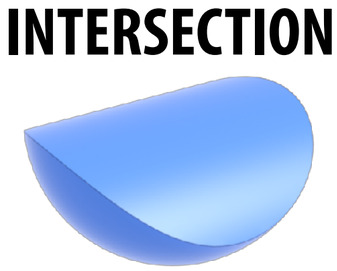
- Very hard to come up with polynomials for complex shapes!

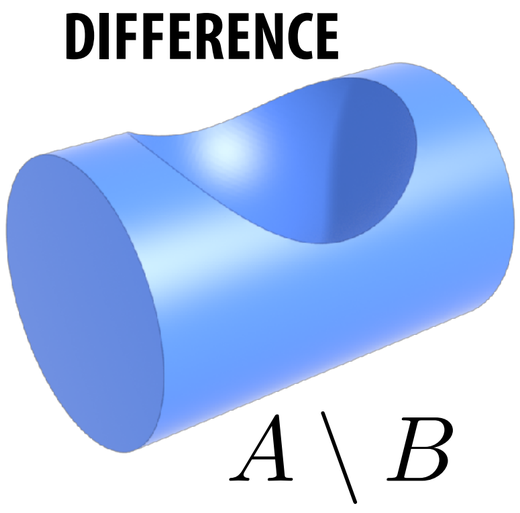
Constructive solid geometry (implicit)

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

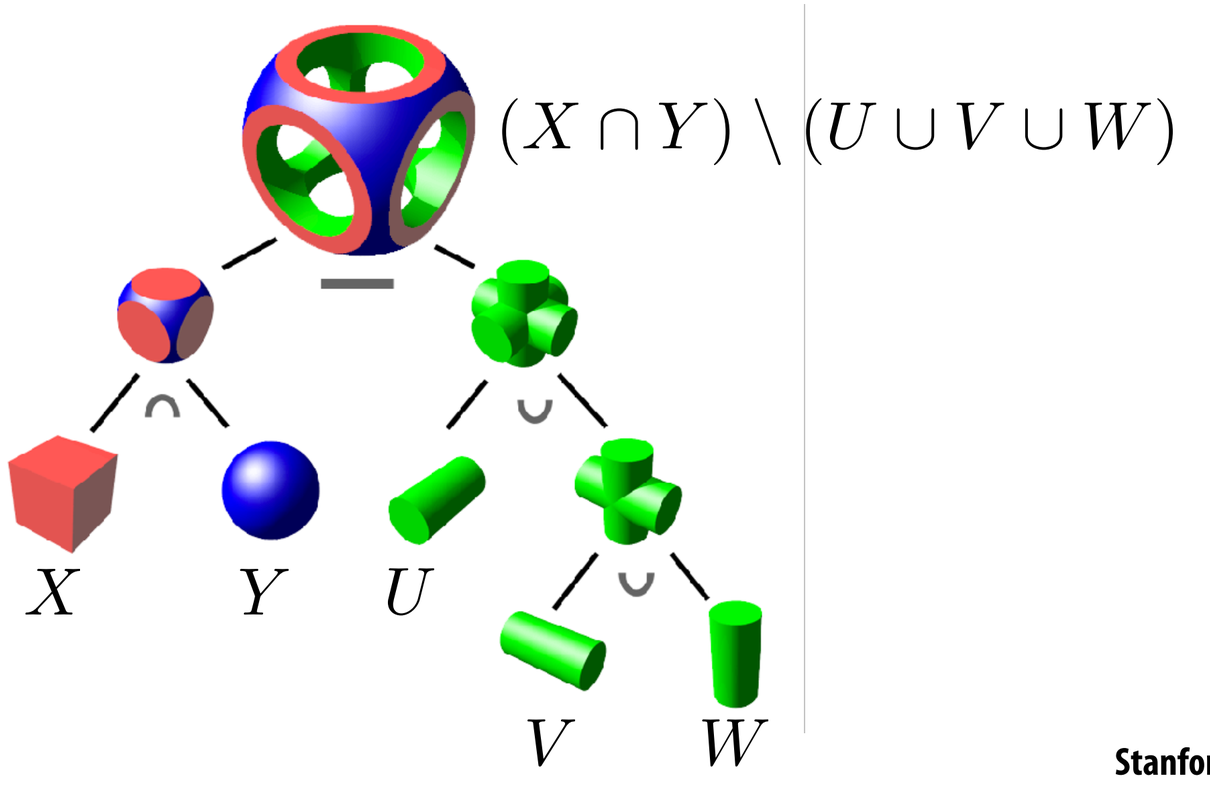






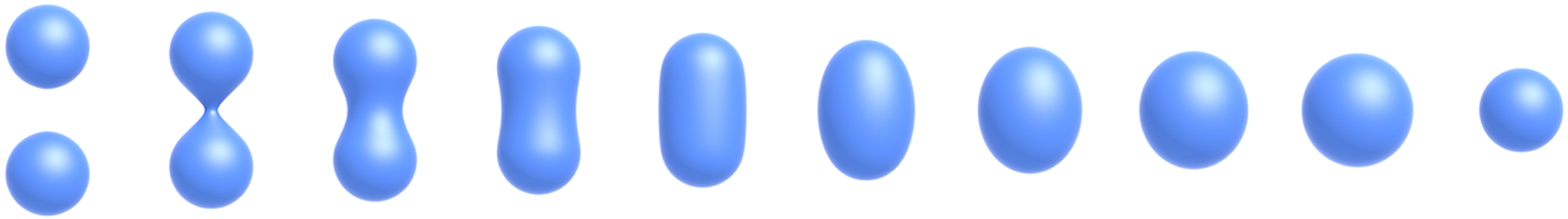


- Then build more complex expressions:



Blobby surfaces (implicit)

- Instead of booleans, gradually blend surfaces together:



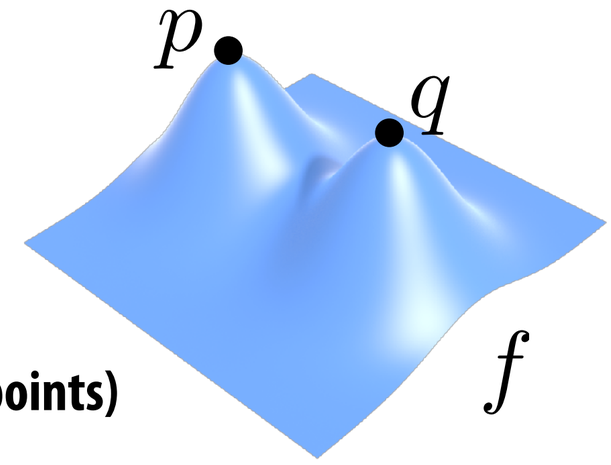
- Easier to understand in 2D:

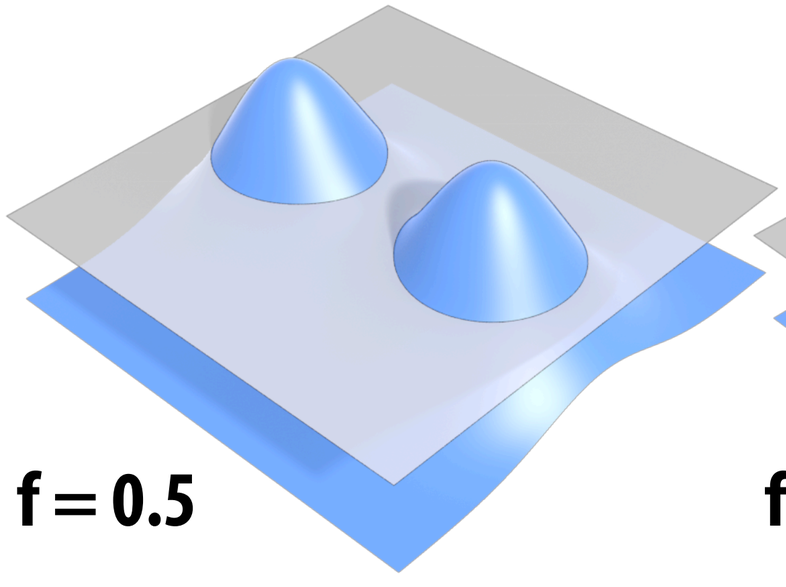
$$\phi_p(x) := e^{-|x-p|^2}$$

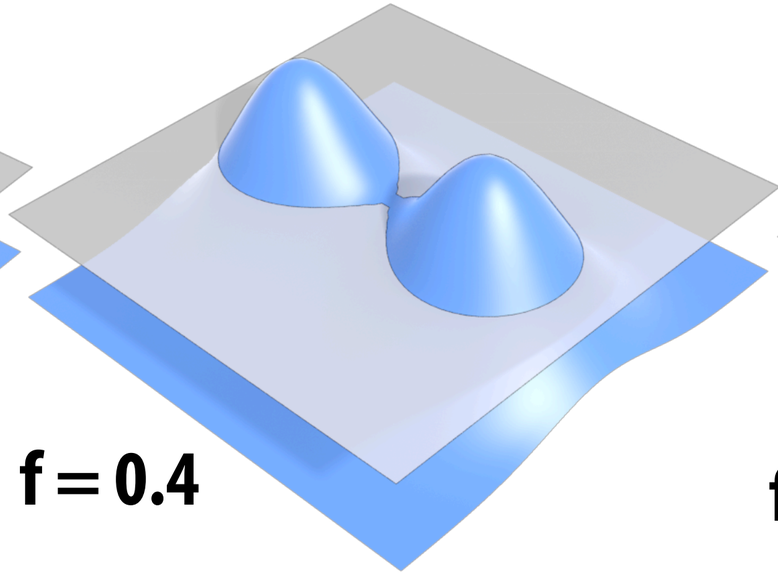
(Gaussian centered at p)

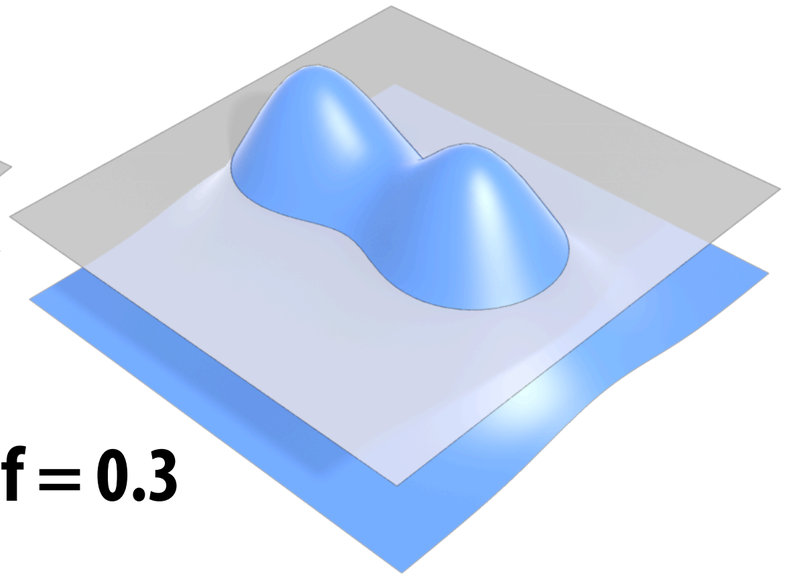
$$f := \phi_p + \phi_q$$

(Sum of Gaussians centered at different points)



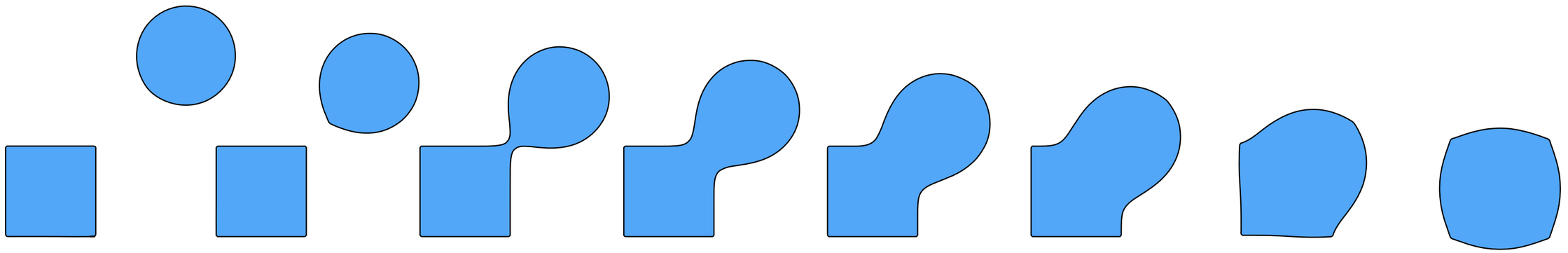






Blending distance functions (implicit)

- 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.,

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

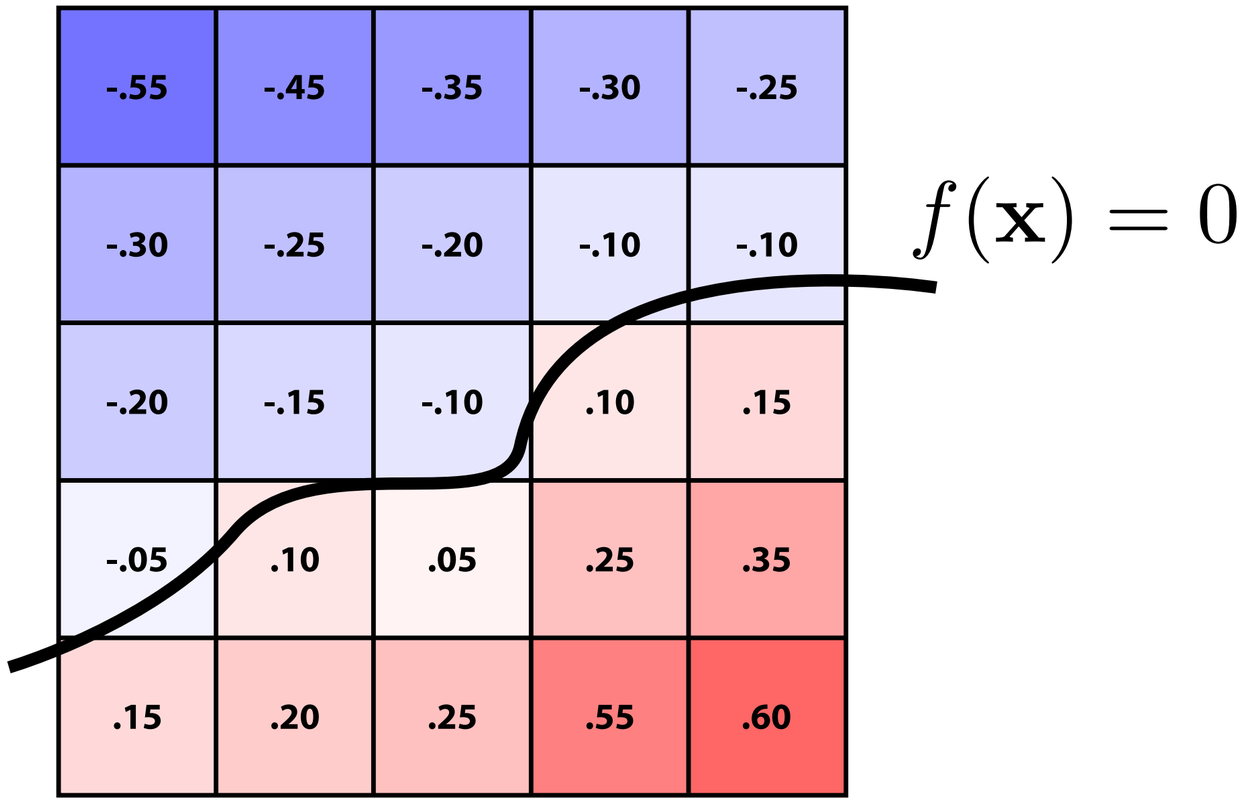
- Appearance depends on exactly how we combine functions
- Q: How do we implement a simple Boolean union?
- A: Just take the minimum: $f(x) := \min(d_1(x)) + \min(d_2(x))$

Scene of pure distance functions (not easy!)

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

Level set methods (implicit)

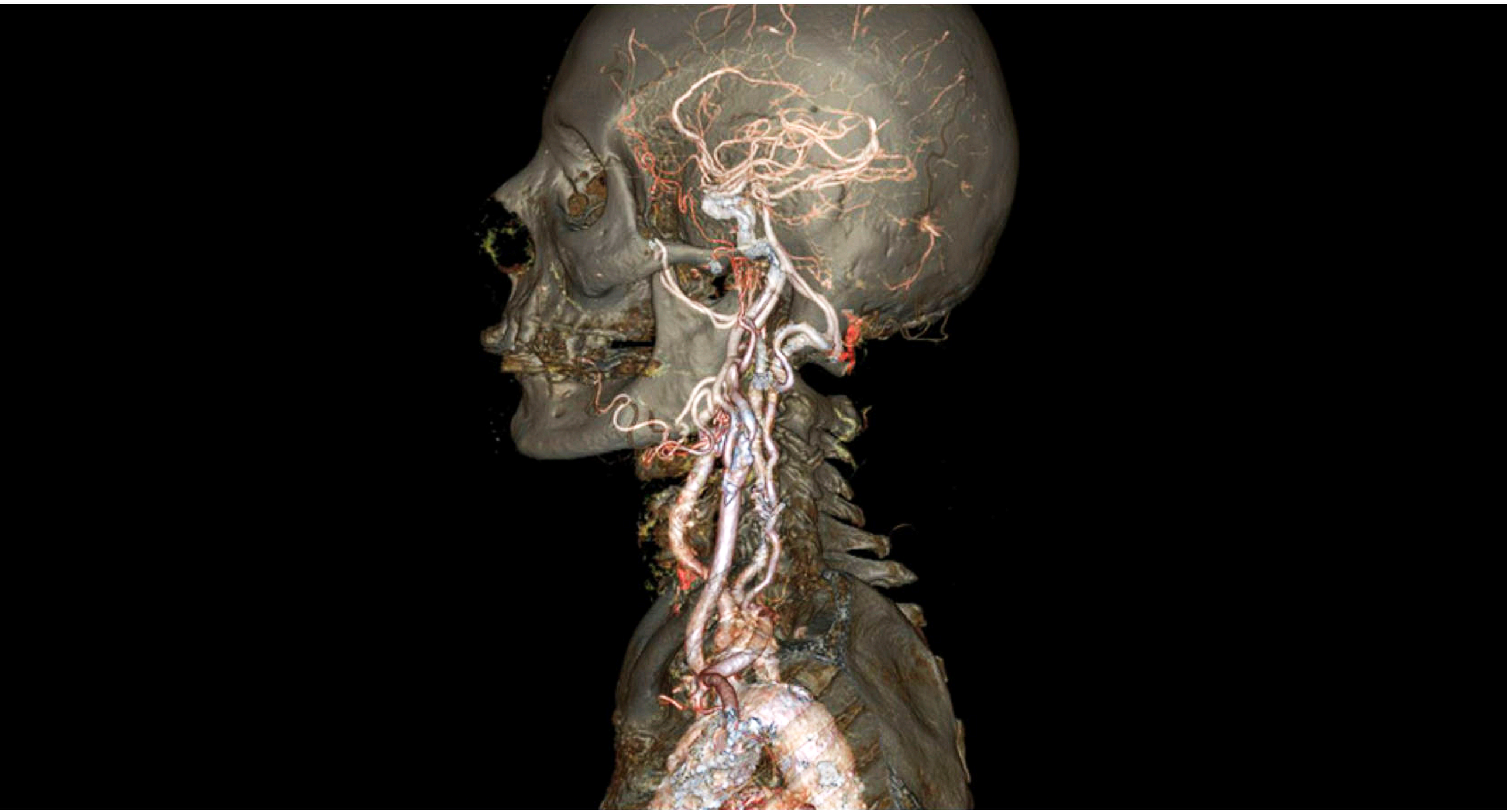
- Implicit surfaces have some nice features (e.g., merging/splitting)
- But, hard to describe complex shapes in closed form
- Alternative: store a grid of values approximating function



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

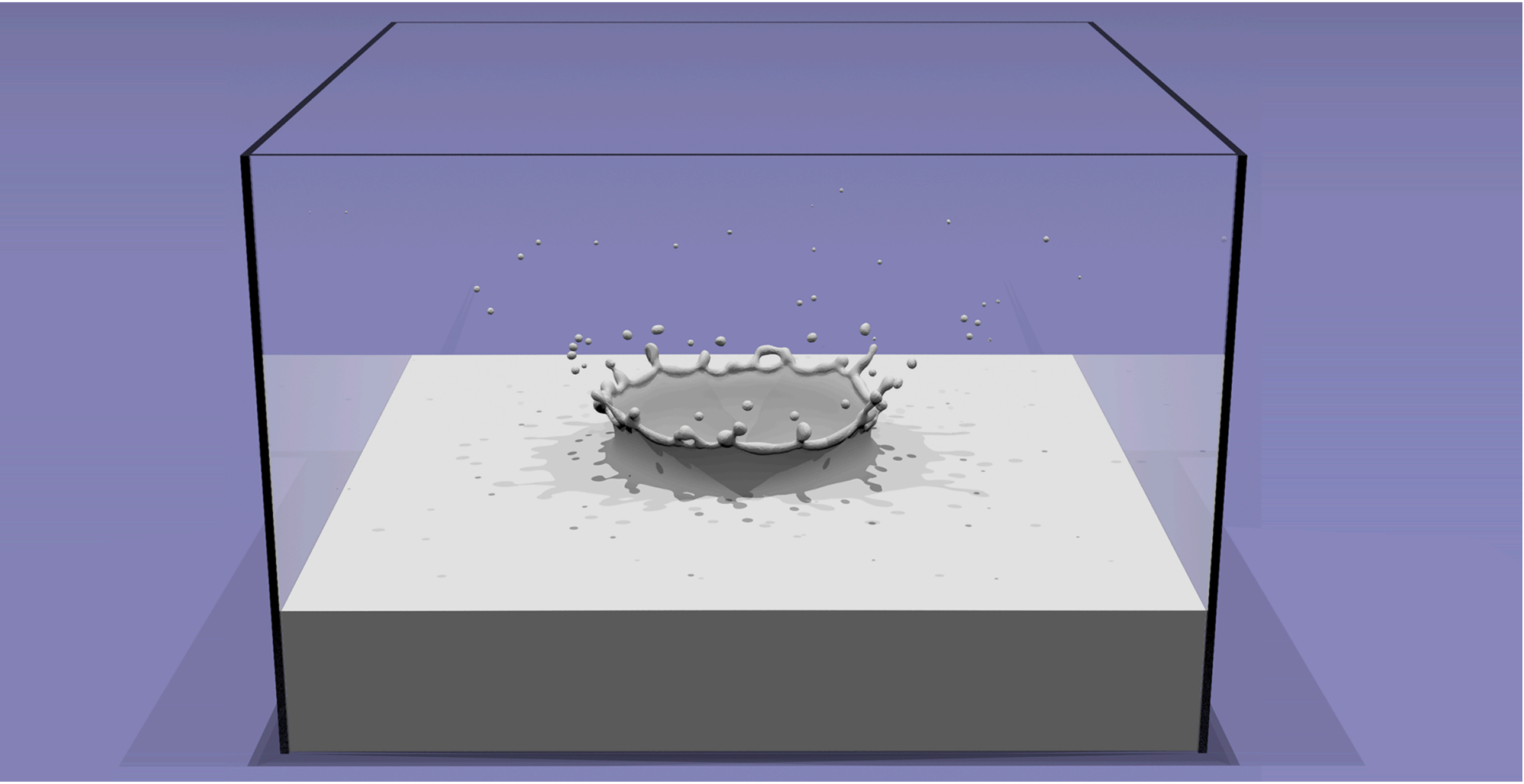
Level sets from medical data (CT, MRI, etc.)

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



Level sets in physical simulation

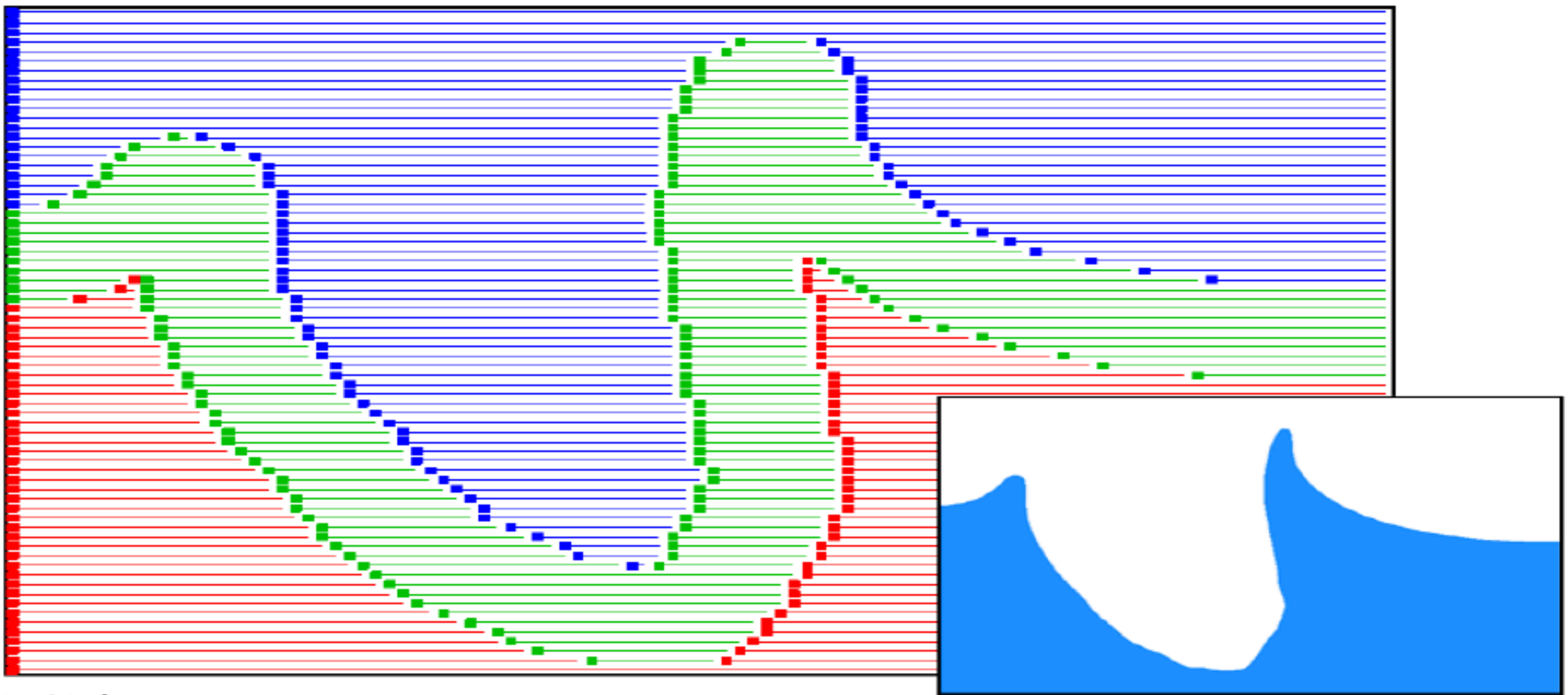
- Level set encodes distance to air-liquid boundary



See http://physbam.stanford.edu

Level set storage

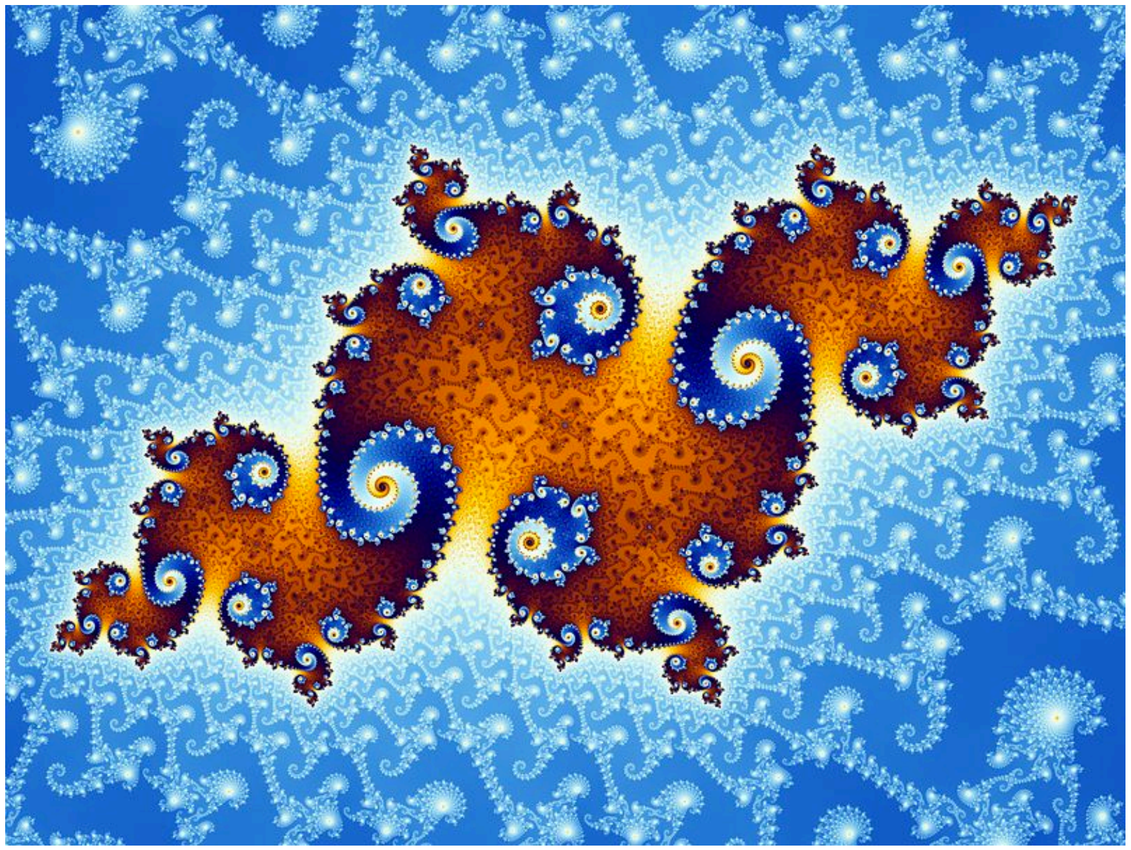
- Drawback: storage for 2D surface is now $O(n^3)$
- Can reduce cost by storing only a narrow band of distances around surface:



In this figure:
red = clearly within water
blue = clearly outside water
green = regions where we store level set values to encode surface

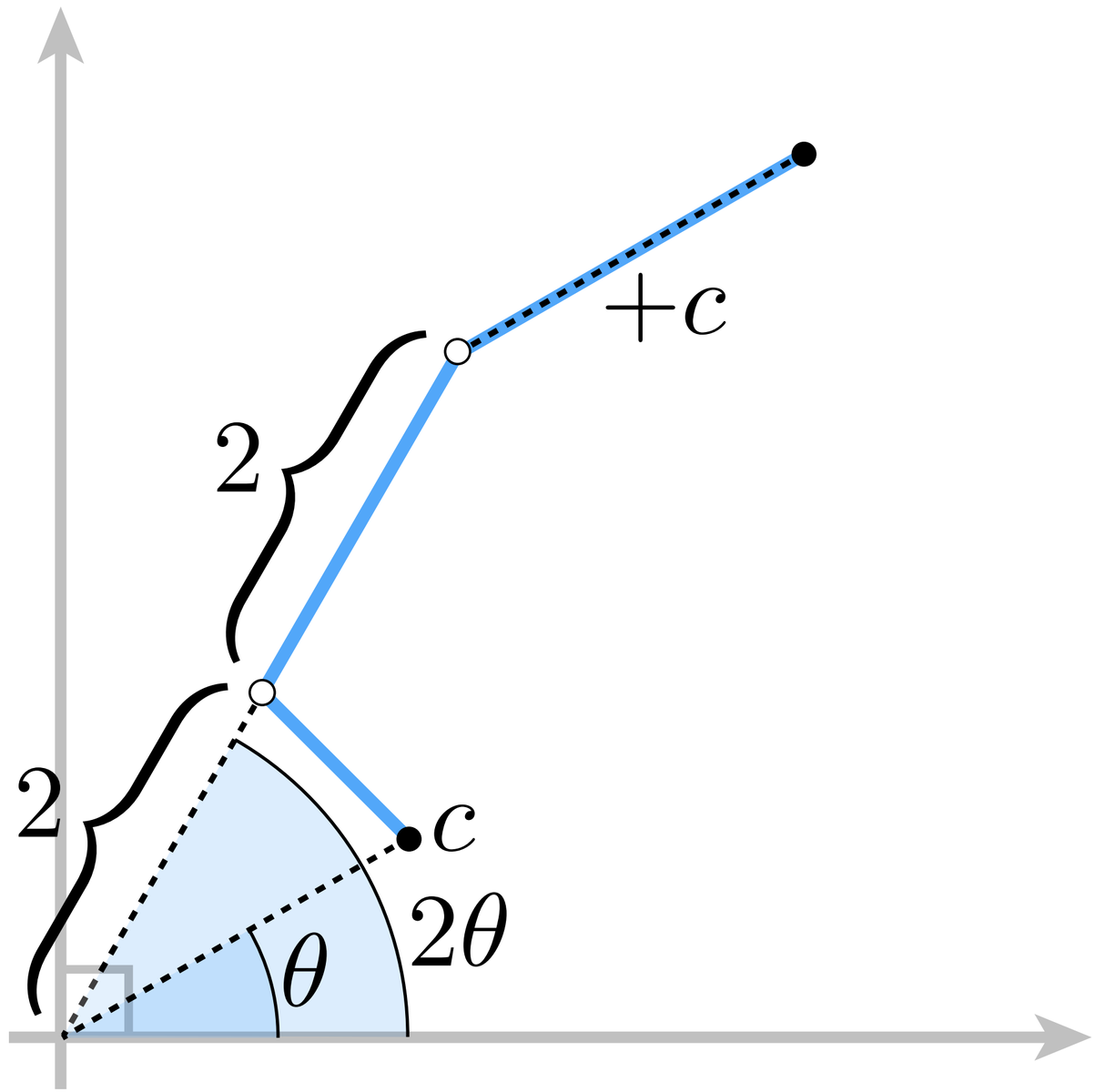
Fractals (implicit)

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



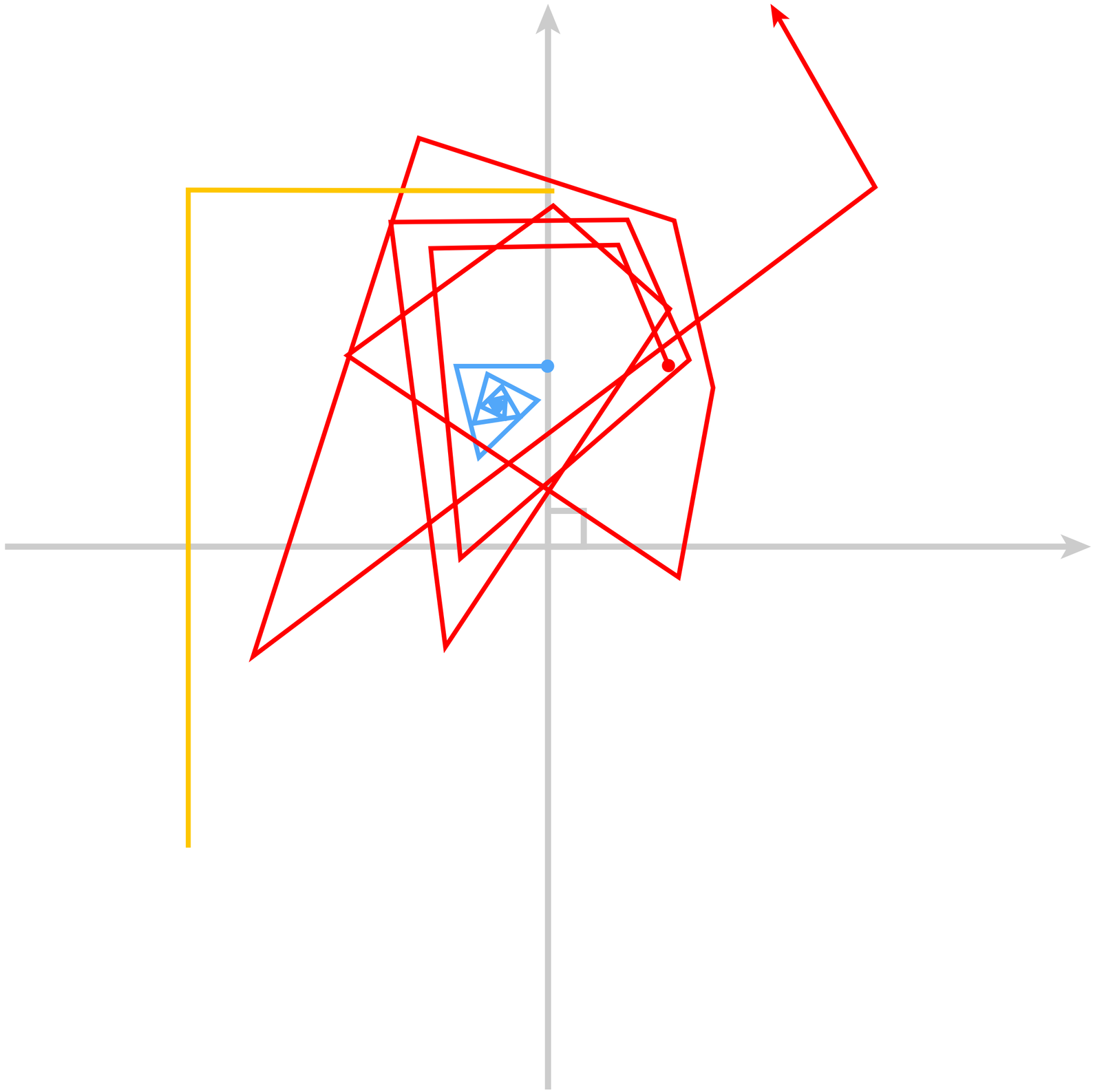
Mandelbrot set - definition

- For each point c in the plane:
  - double the angle
  - square the magnitude
  - add the original point c
  - repeat



If the point remains bounded (never goes to ∞), it's in the set.

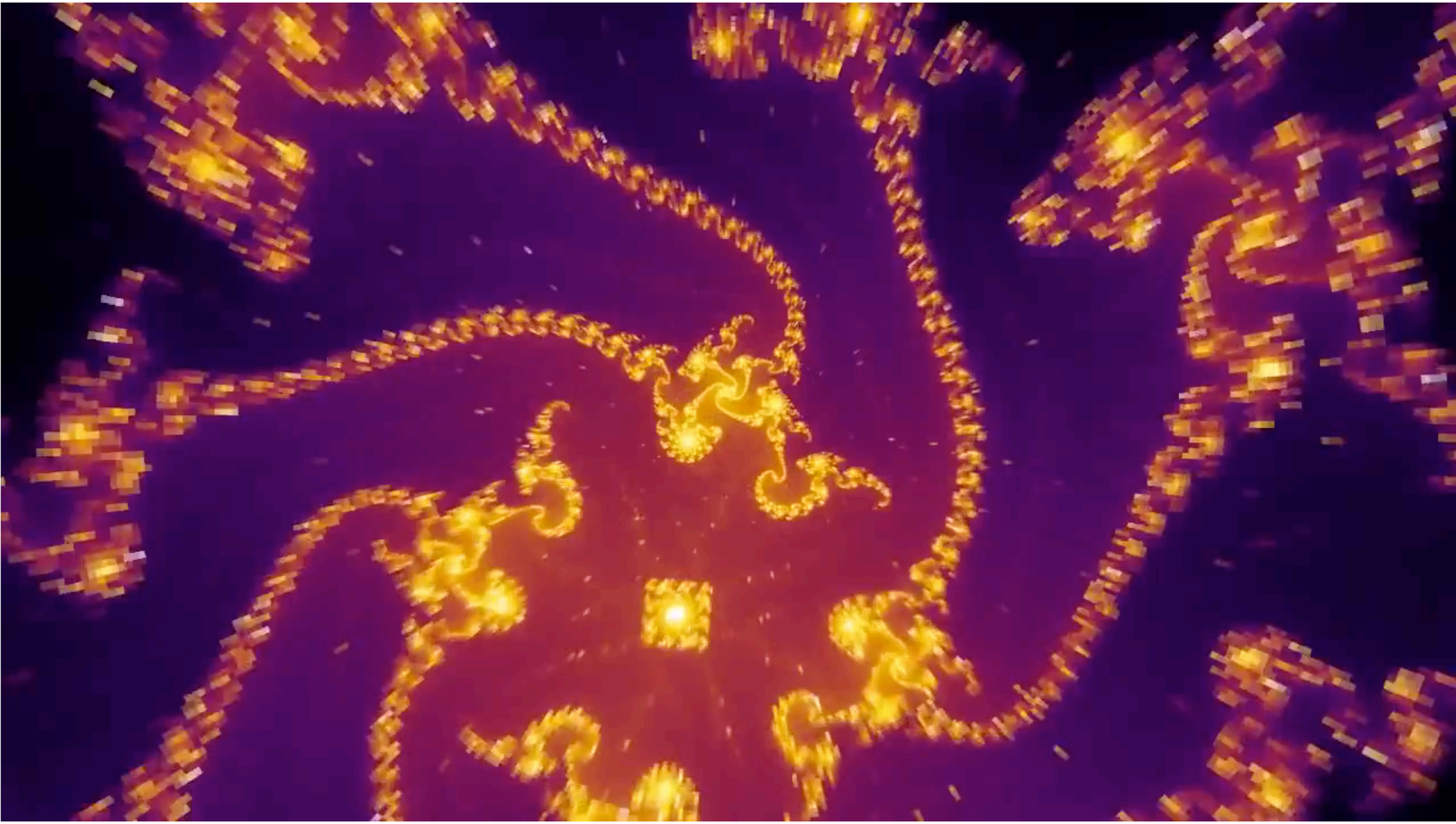
Mandelbrot set - examples



starting point

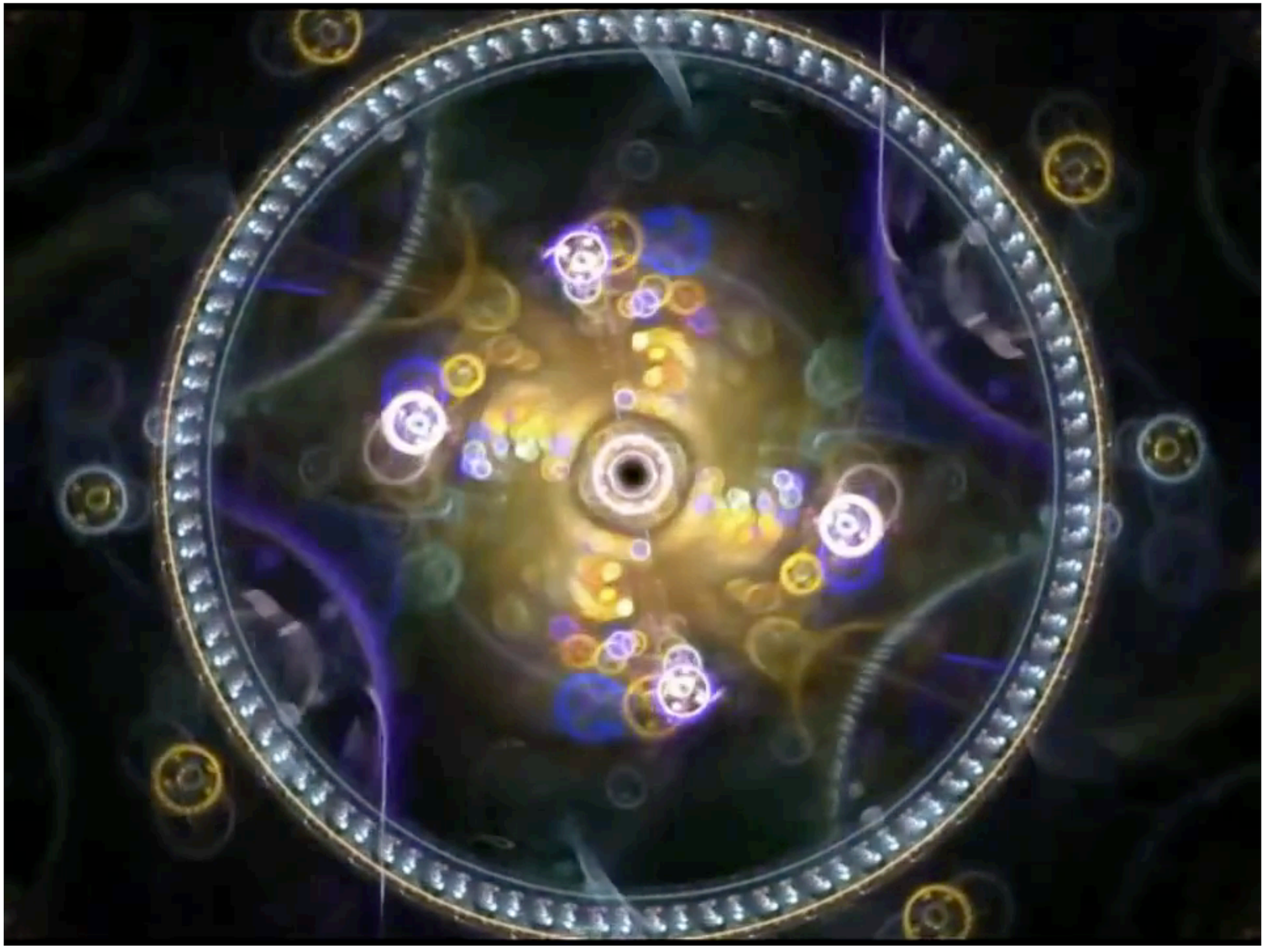
- $(0, 1/2)$ **(converges)**
- $(0, 1)$ **(periodic)**
- $(1/3, 1/2)$ **(diverges)**

Mandelbrot set - zooming in



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

Iterated function systems



Scott Draves (CMU Alumnus) - see http://electricsheep.org

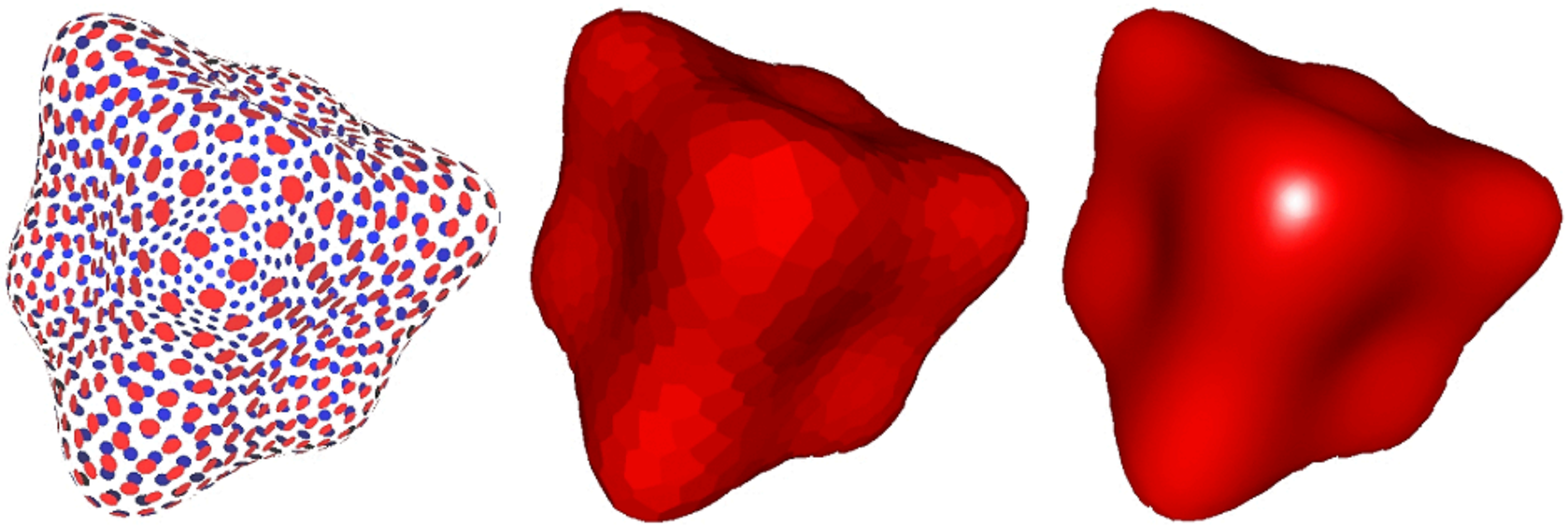
Implicit representations - pros and cons

- **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)

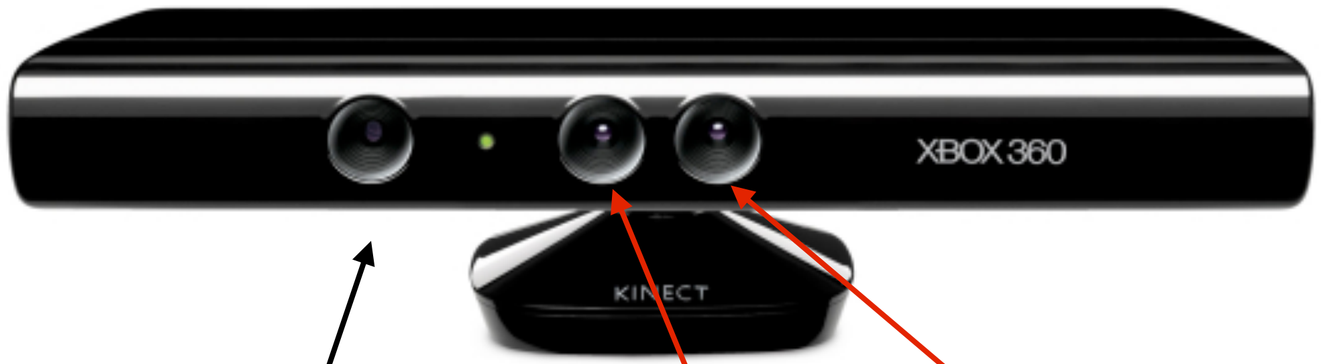
- Easiest representation: list of points (x,y,z)
- Often augmented with *normals*
- Easily represent any kind of geometry
- Useful for LARGE datasets (>>1 point/pixel)
- Hard to interpolate undersampled regions
- Hard to do processing / simulation / …



Point cloud via laser scanning

Image Credit: 3Dling

Another example: Microsoft XBox 360 Kinect



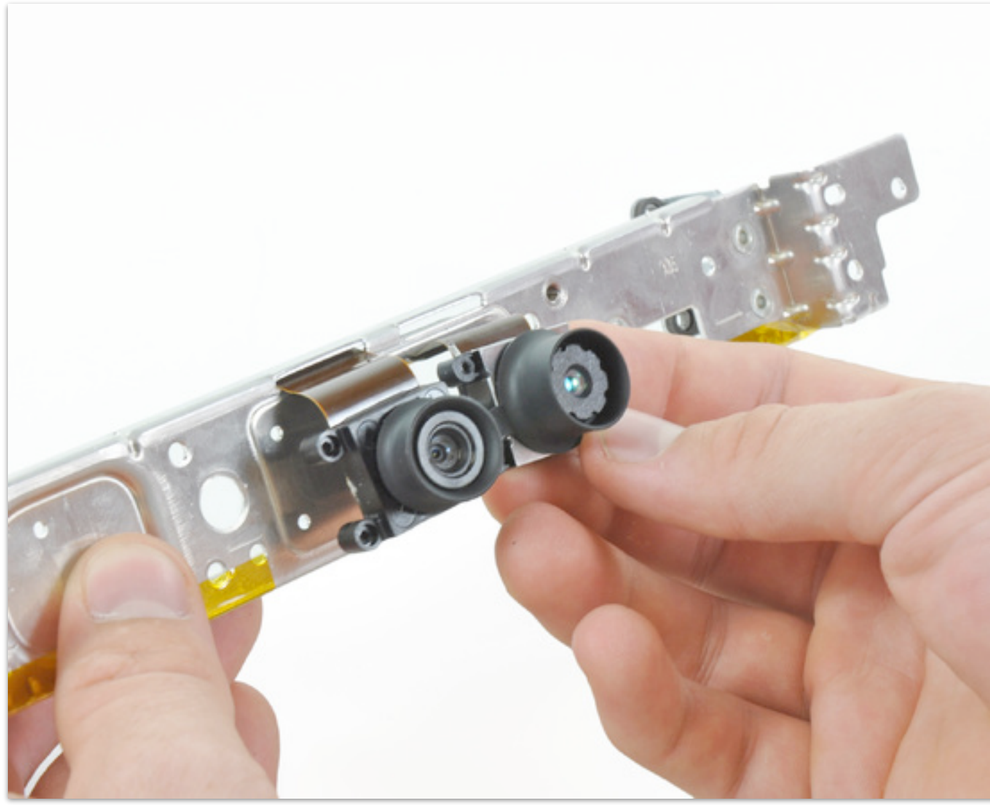


Image credit: iFixIt

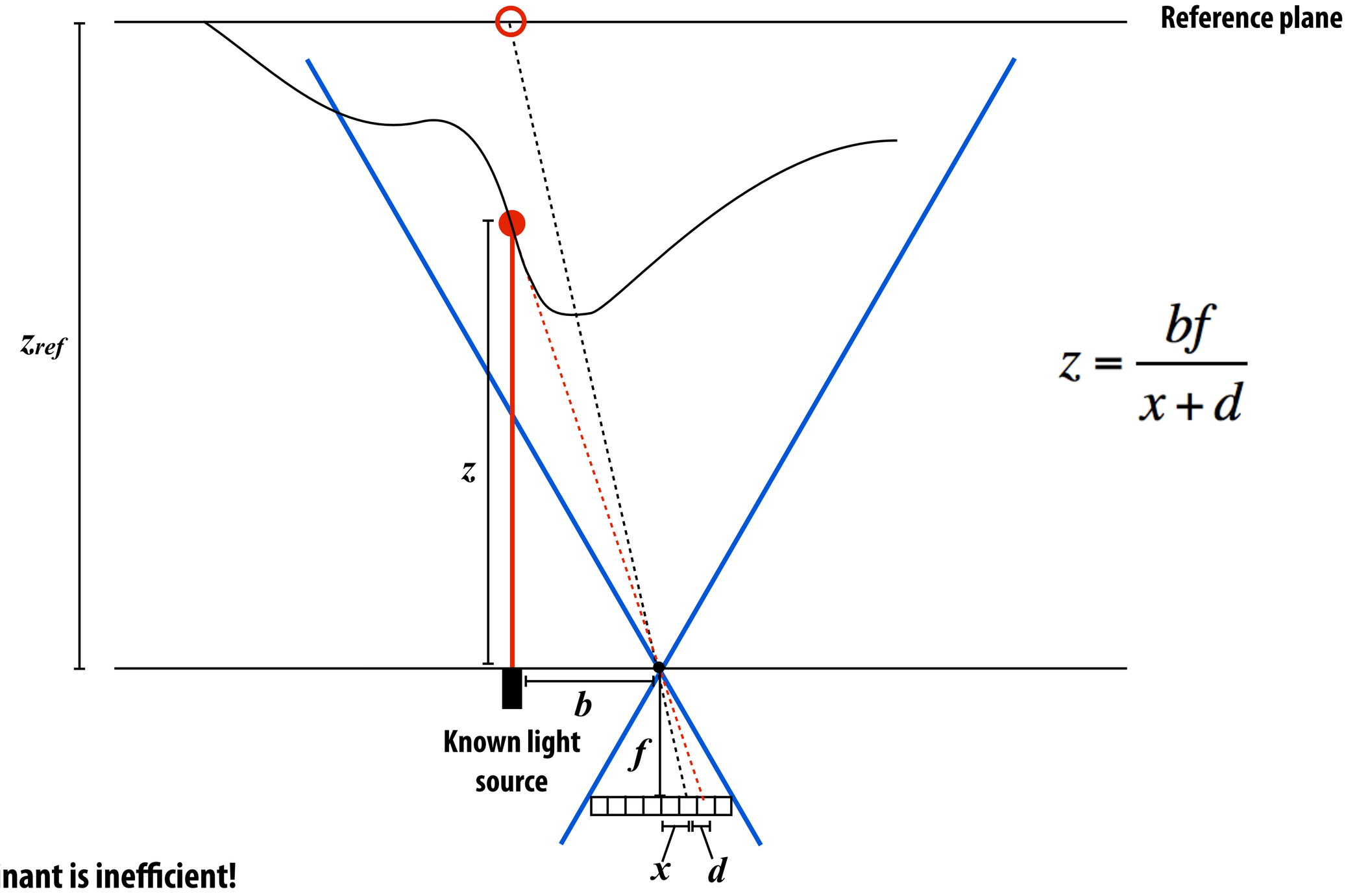
Illuminant
(Infrared Laser + diffuser)

RGB Sensor
640x480

Monochrome Infrared
Sensor

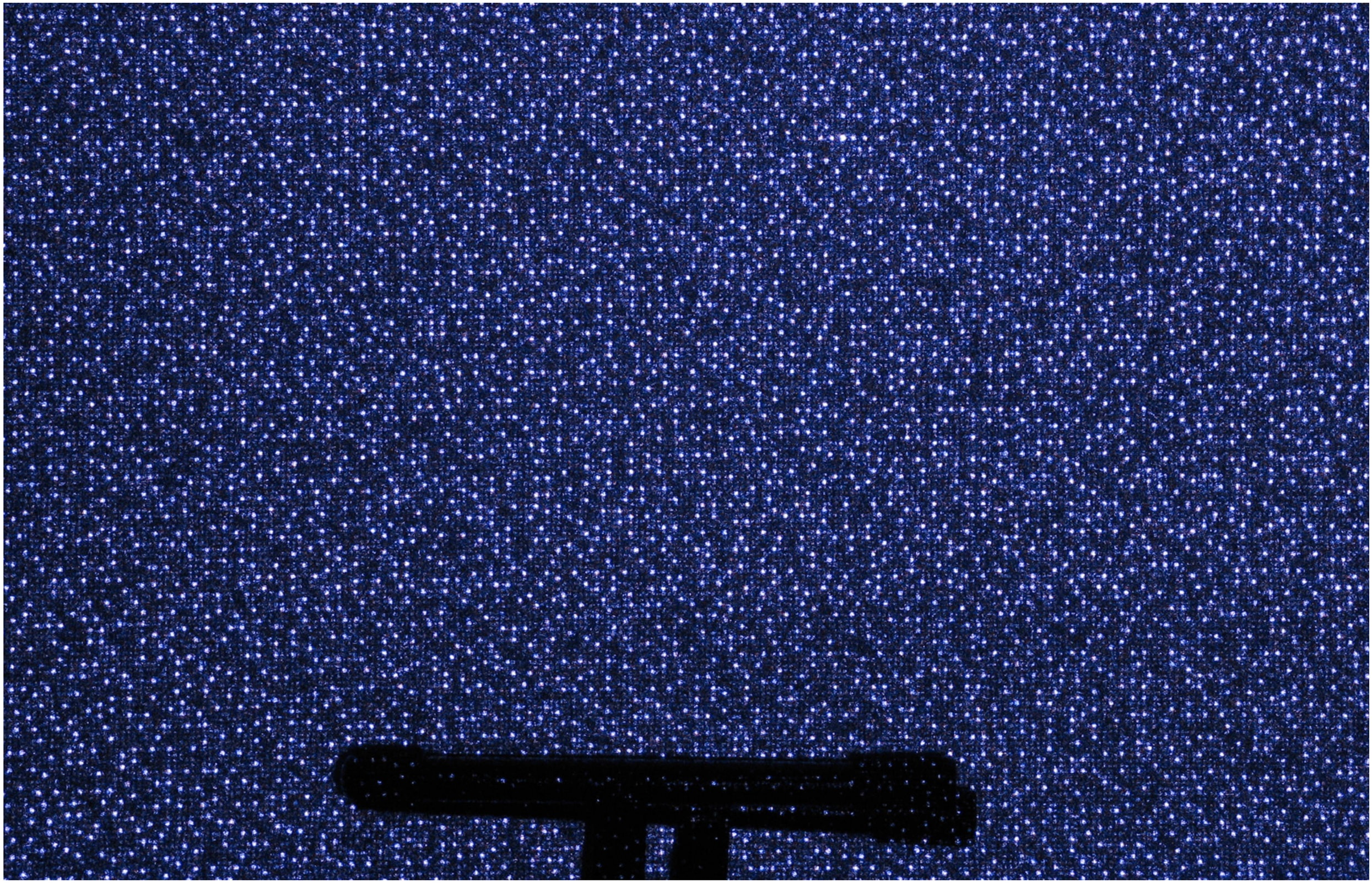
Structured light

System: one light source emitting known beam + one camera measuring scene appearance
If the scene is at reference plane, image that will be recorded by camera is known
(correspondence between pixel in recorded image and scene point is known)



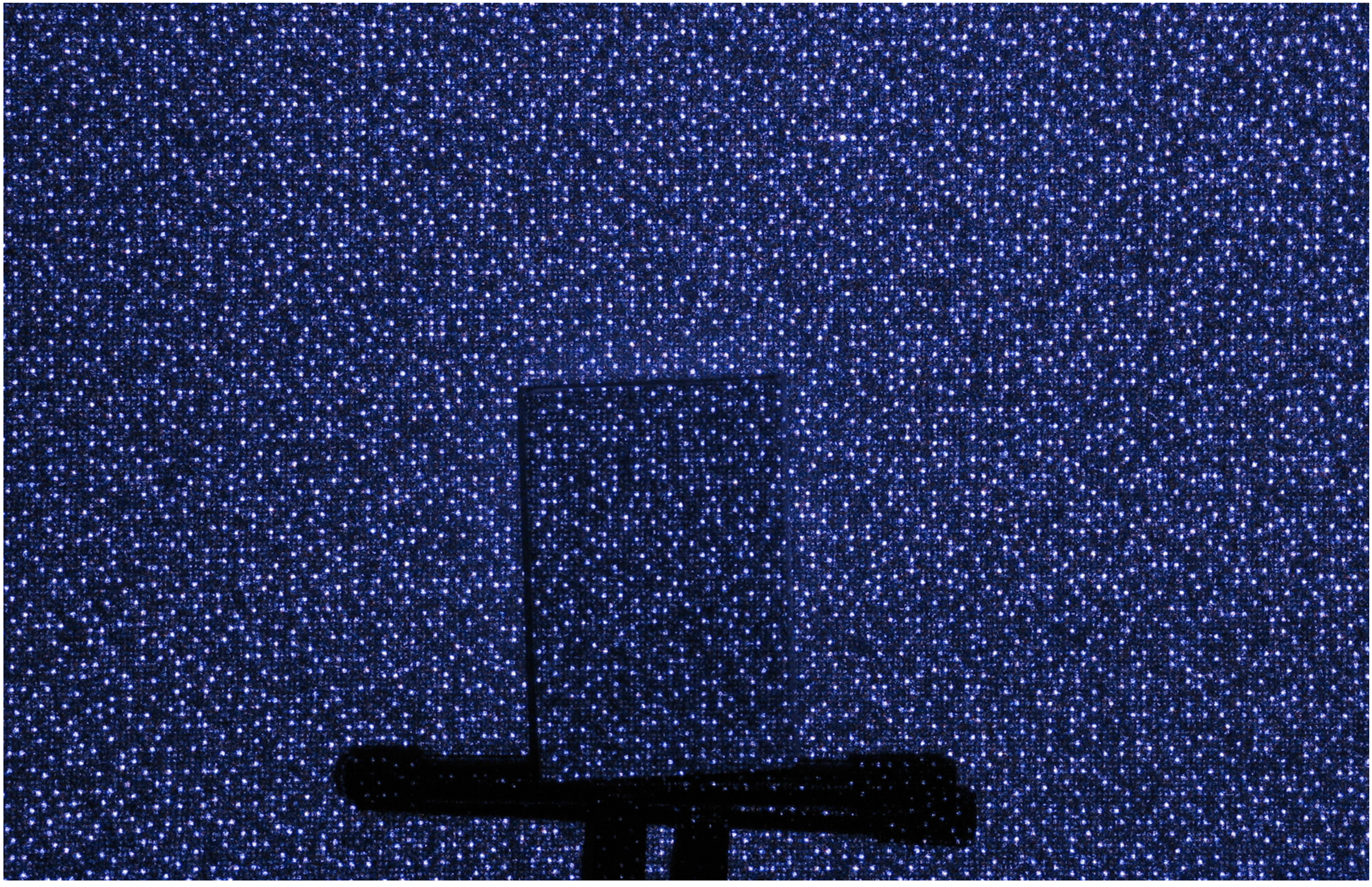
Single spot illuminant is inefficient!
(must "scan" scene with spot to get depth, so high latency to retrieve a single depth image)

Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

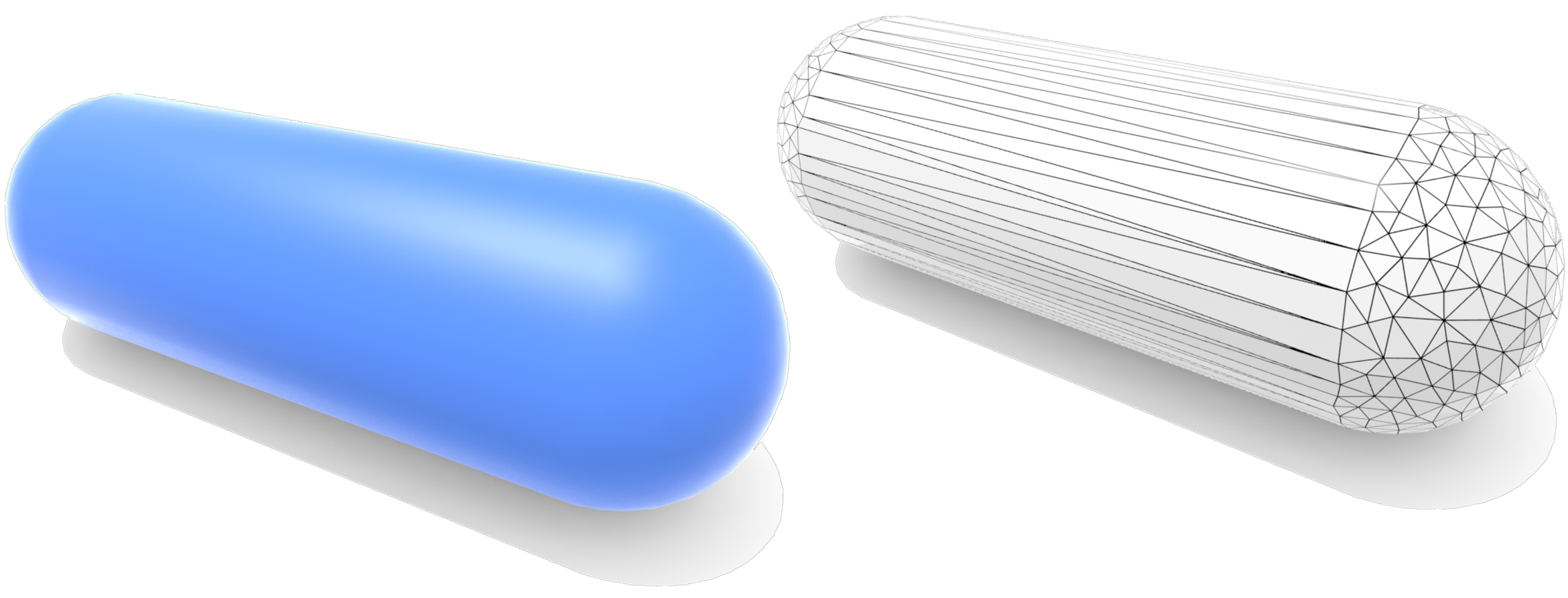
Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

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)

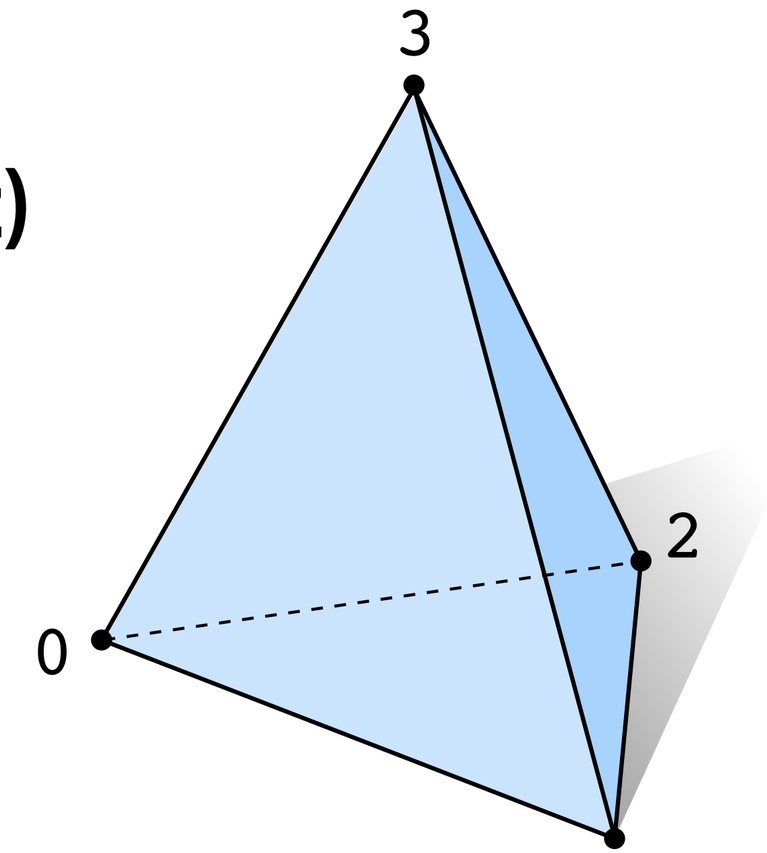
- Store vertices as triples of coordinates (x,y,z)
- Store triangles as triples of indices (i,j,k)
- E.g., tetrahedron:

VERTICES

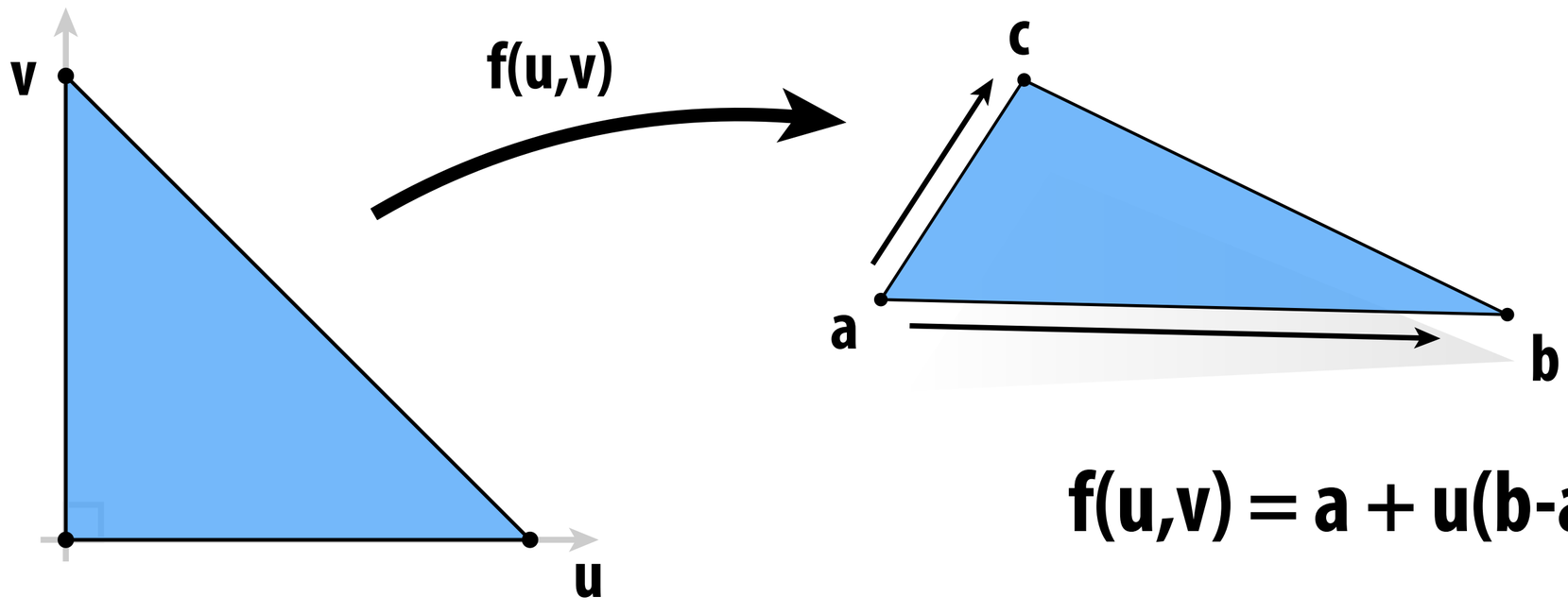
	x	y	z
0:	-1	-1	-1
1:	1	-1	1
2:	1	1	-1
3:	-1	1	1

TRIANGLES

i	j	k
0	2	1
0	3	2
3	0	1
3	1	2

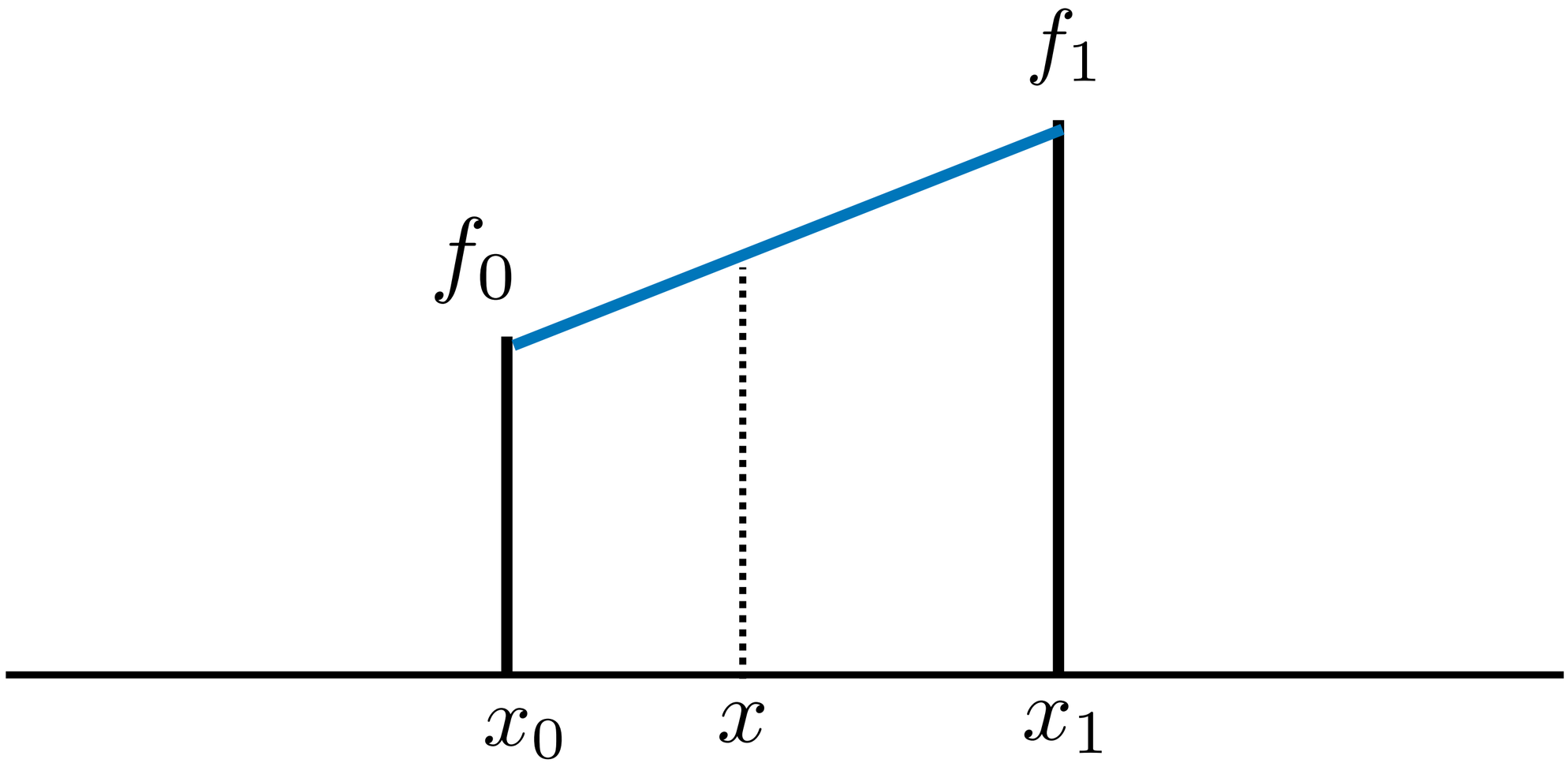


- Use linear interpolation to define points inside triangles:



$$f(u,v) = a + u(b-a) + v(c-a)$$

Linear interpolation of samples (in 1D)

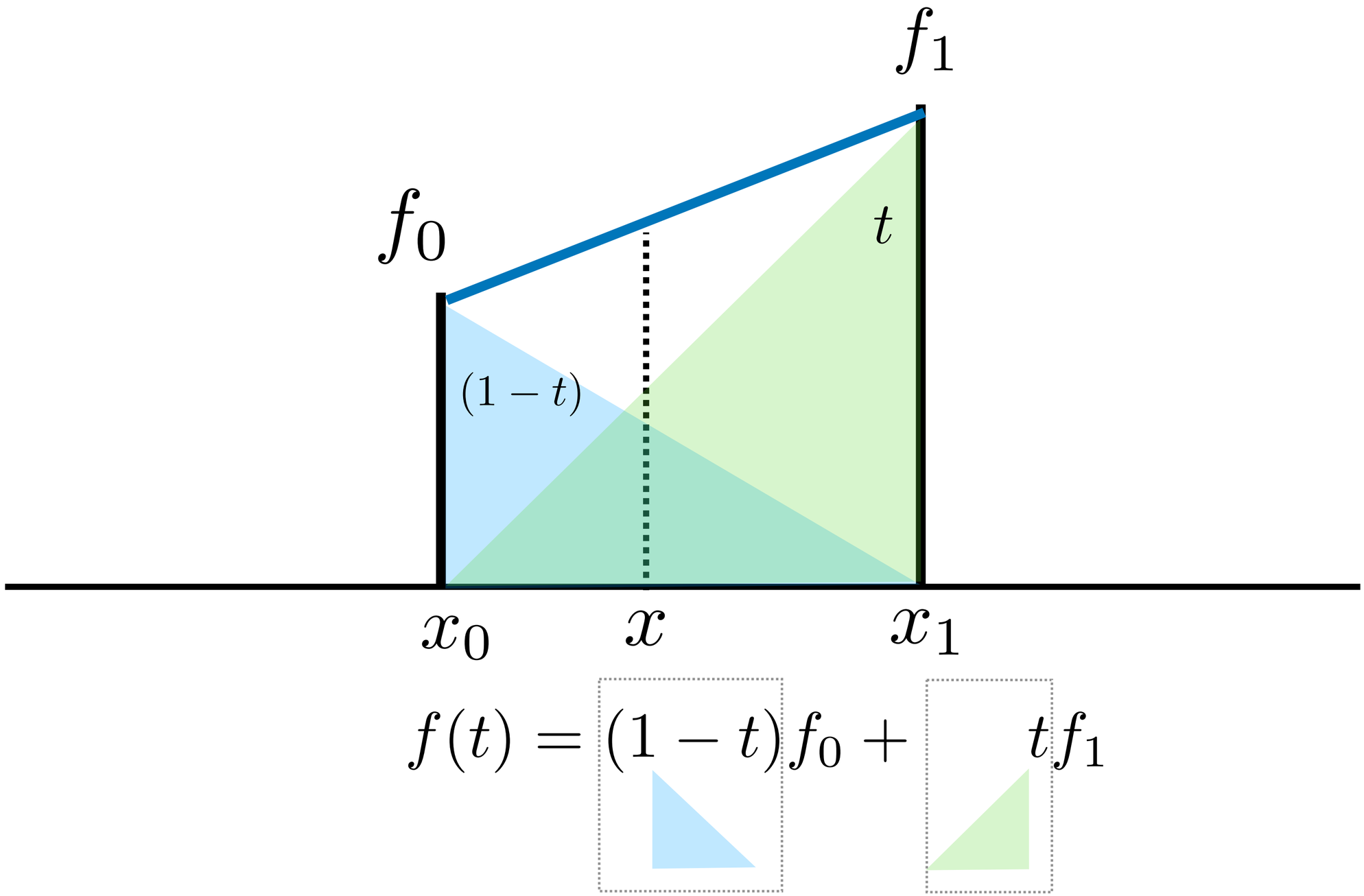


$$f(t) = (1-t)f_0 + tf_1$$

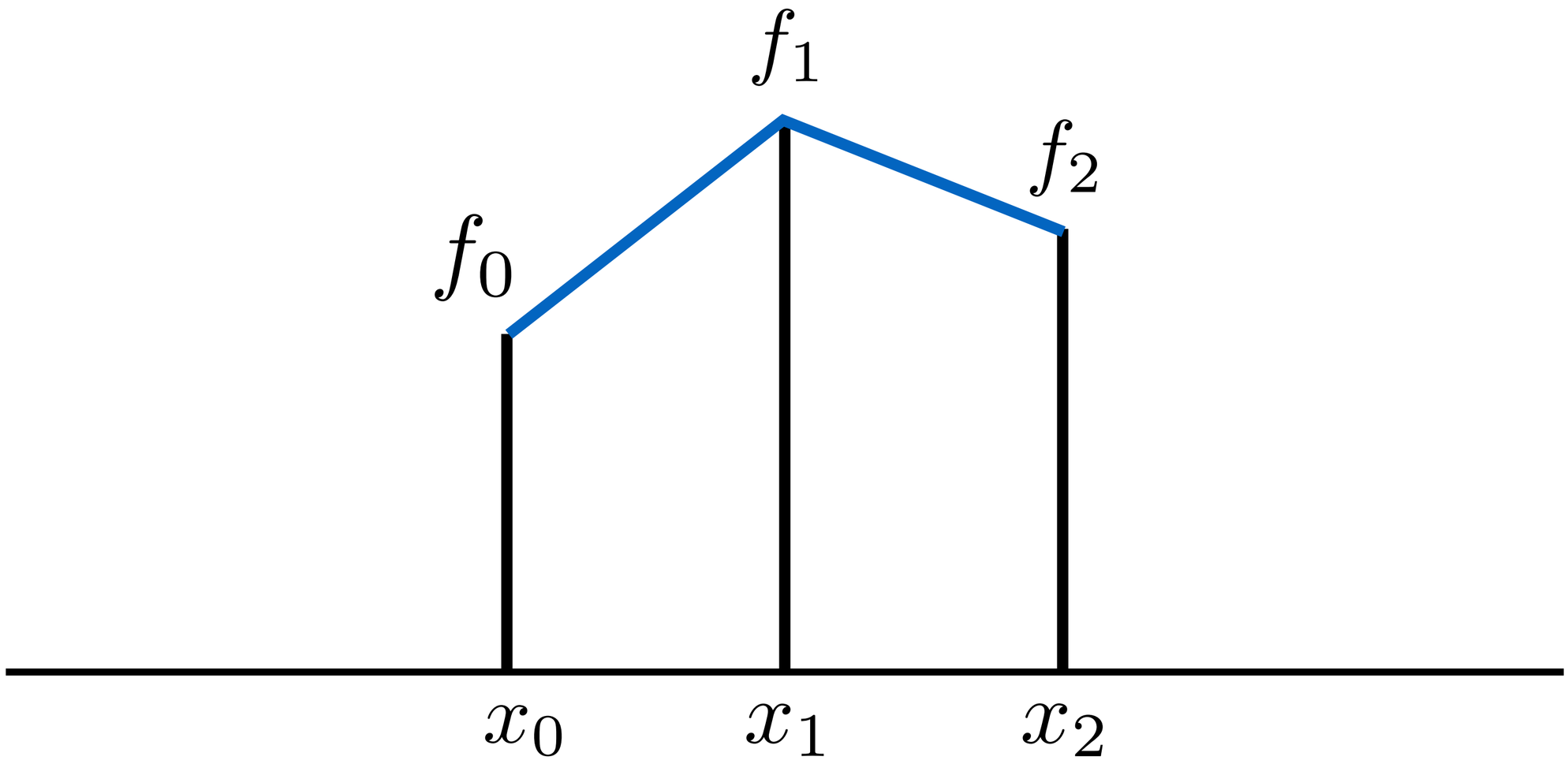
$$t = \frac{x - x_0}{x_1 - x_0}$$

Can think of linear interpolation as linear combination of two functions

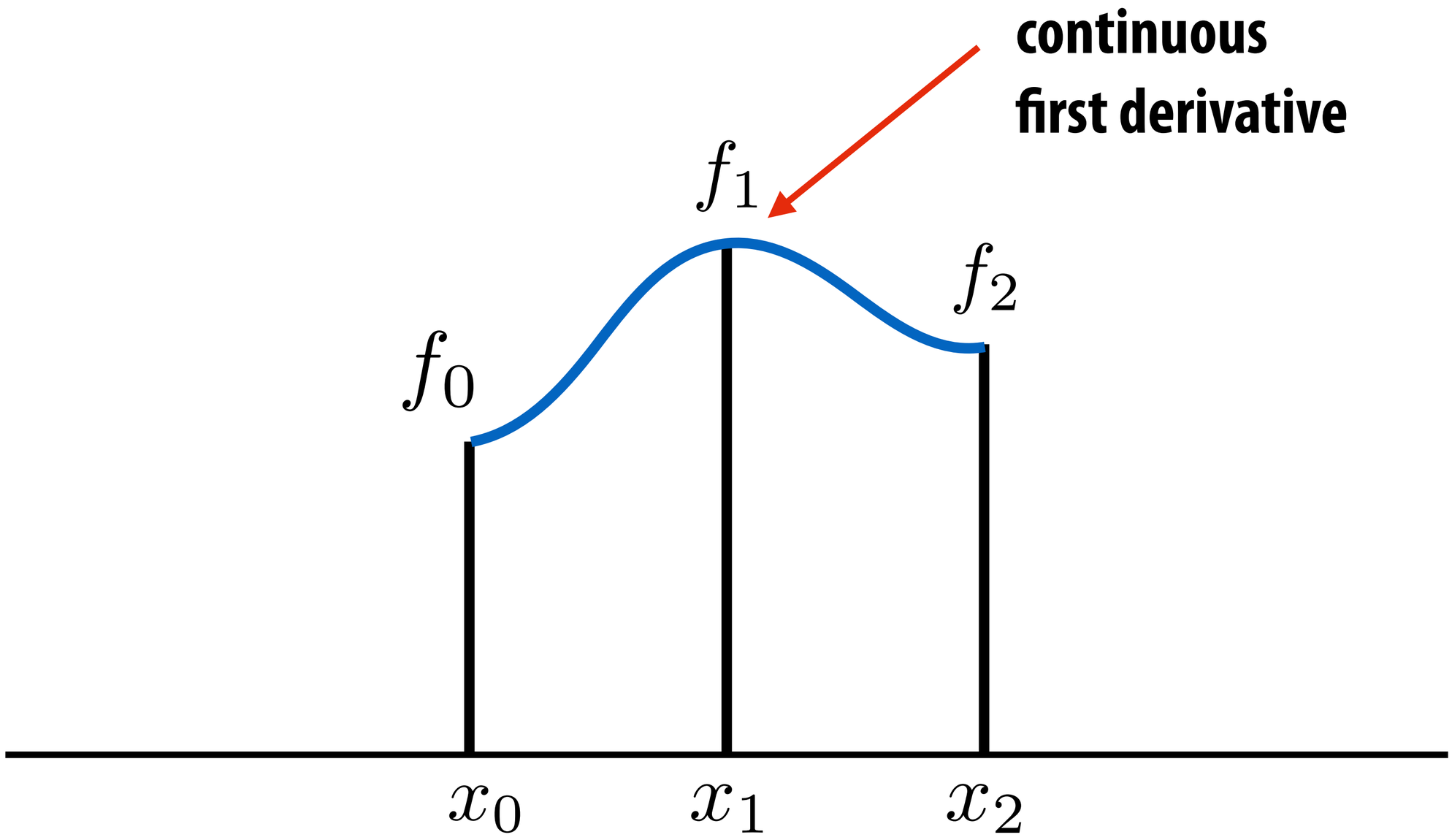
Weights are given by the two values (f_0 and f_1) being interpolated



Problem with piecewise linear interpolation: derivates not continuous



Smooth interpolation?



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, ...), use Bernstein basis:

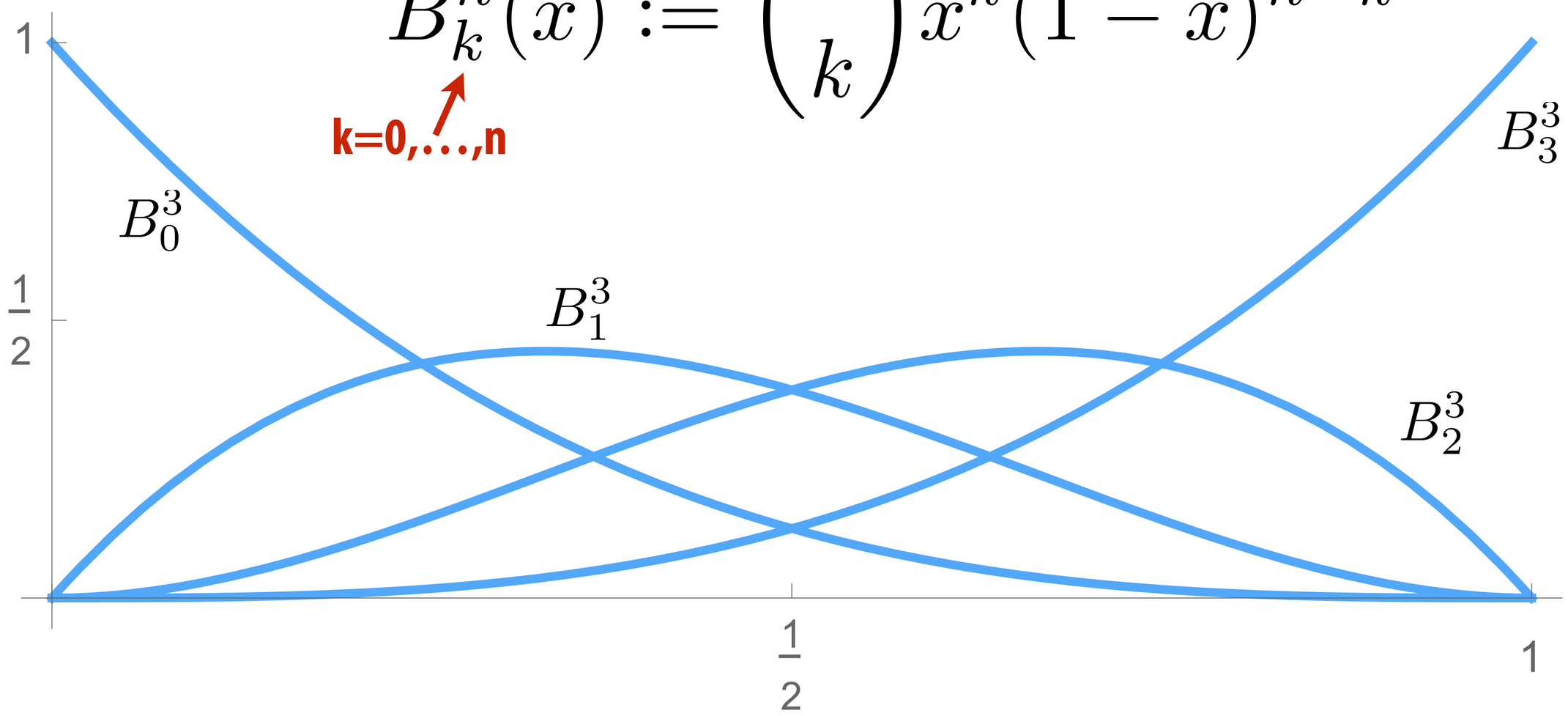
degree

0≤x≤1

"n choose k"

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

k=0,...,n



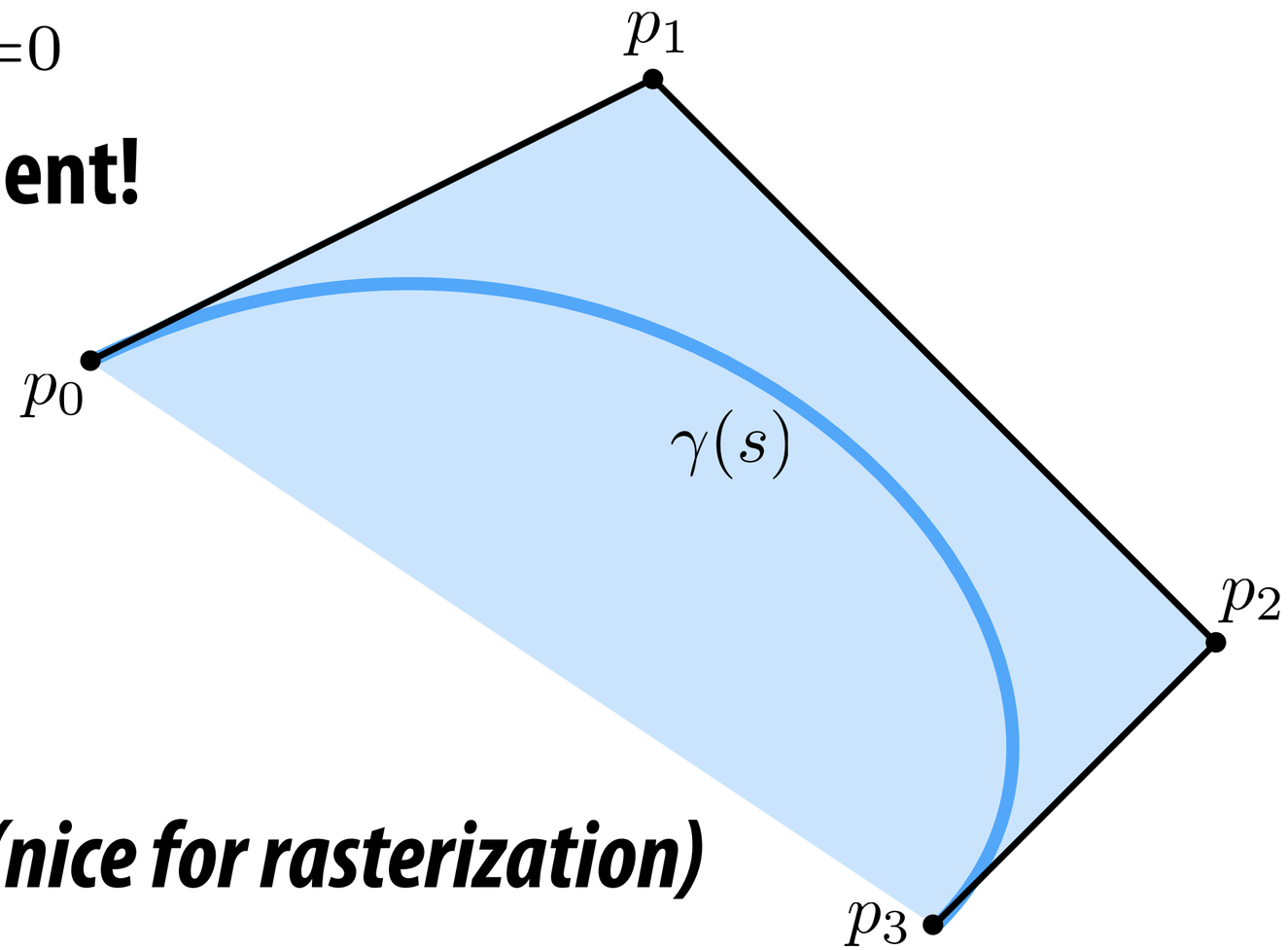
Bézier curves (explicit)

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

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

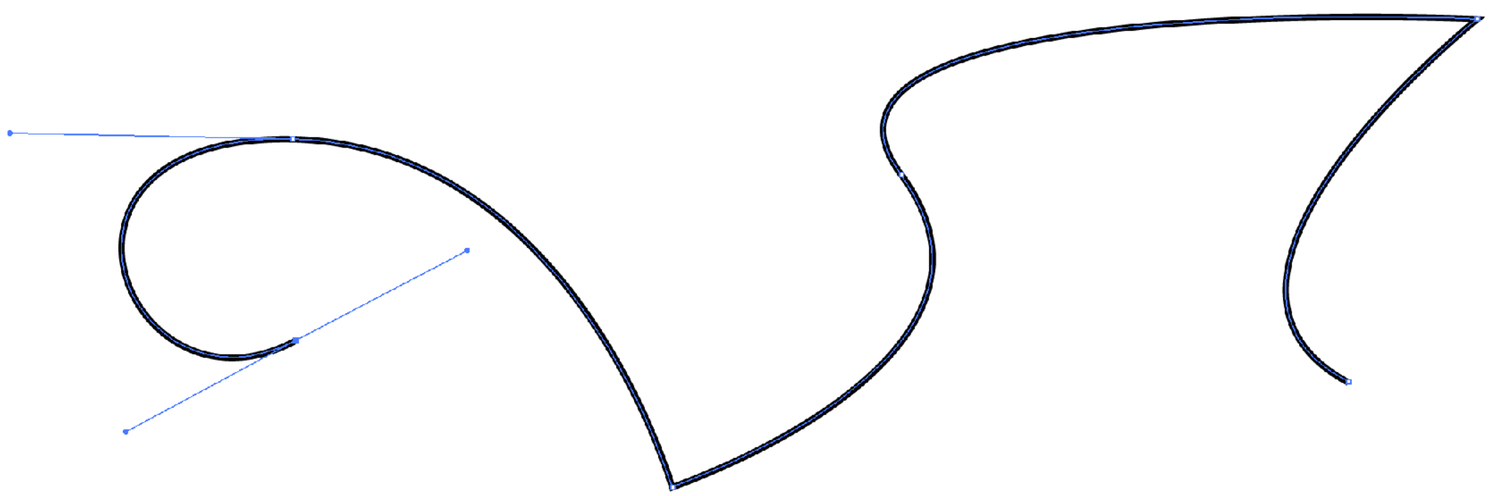
control points

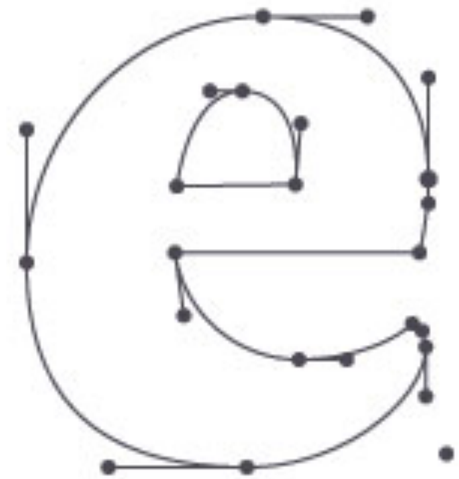
- For n=1, just get a line segment!
- For n=3, get "cubic Bézier":
- Important features:
  1. interpolates endpoints
  2. tangent to end segments
  3. contained in convex hull *(nice for rasterization)*



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

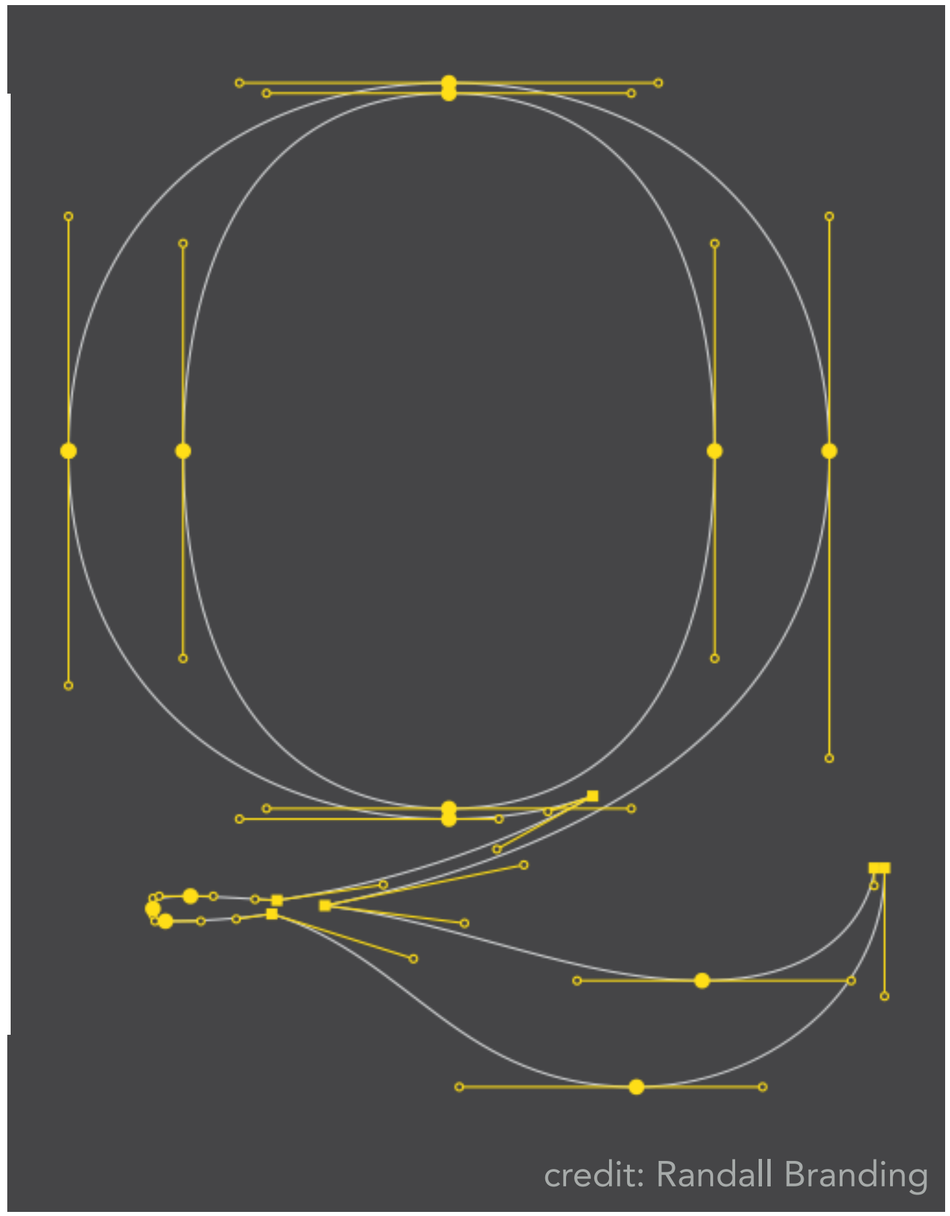
$$\gamma(u) := \gamma_i \left(\frac{u - u_i}{u_{i+1} - u_i} \right), \qquad u_i \leq u < u_{i+1}$$

single Bézier

Vector fonts

The Quick Brown Fox Jumps Over The Lazy Dog

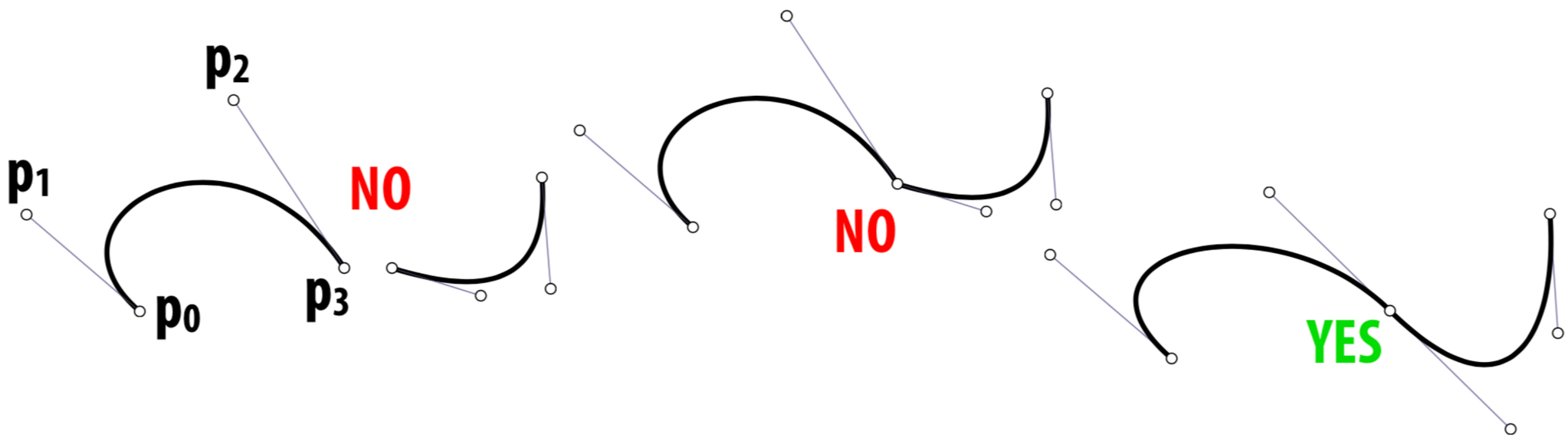
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789



Baskerville font - represented as cubic Bézier splines

Bézier curves — tangent continuity

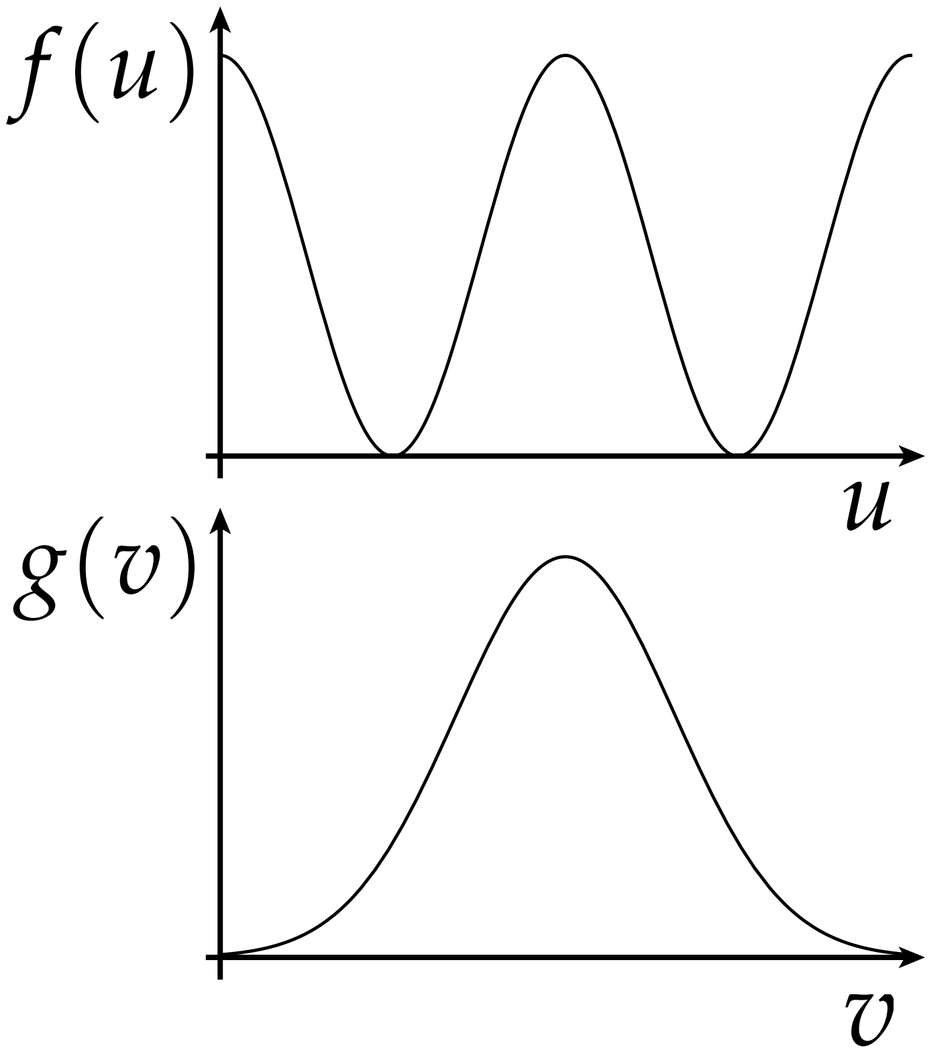
- To get "seamless" curves, want *points* and *tangents* to line up:

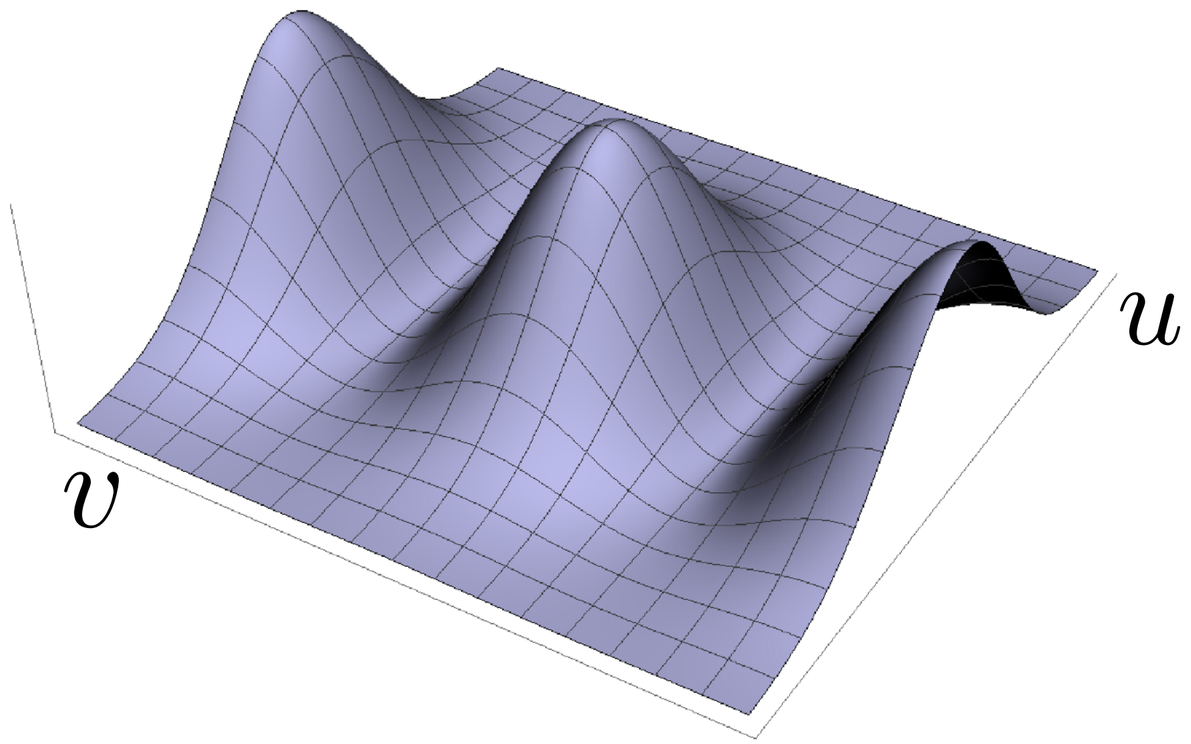


- Ok, but how?
- Each curve is cubic: $u^3p_0 + 3u^2(1-u)p_1 + 3u(1-u)^2p_2 + (1-u)^3p_3$
- Q: How many constraints vs. degrees of freedom?
- Q: Could you do this with *quadratic* Bézier? *Linear* Bézier?

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"*):

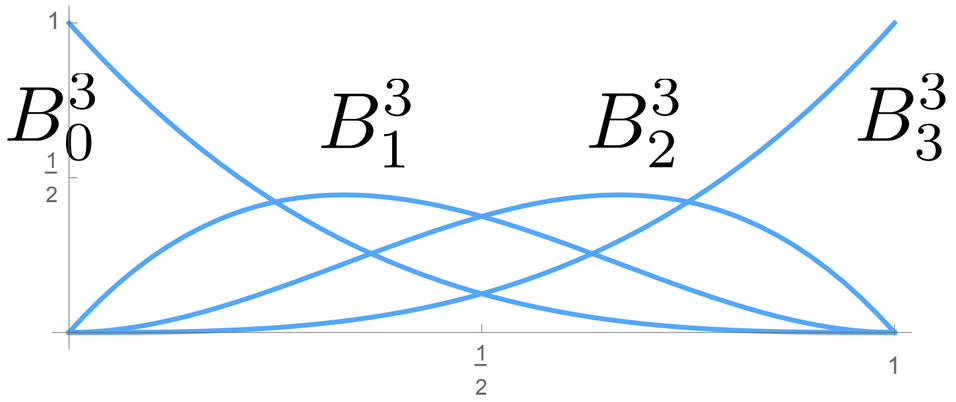




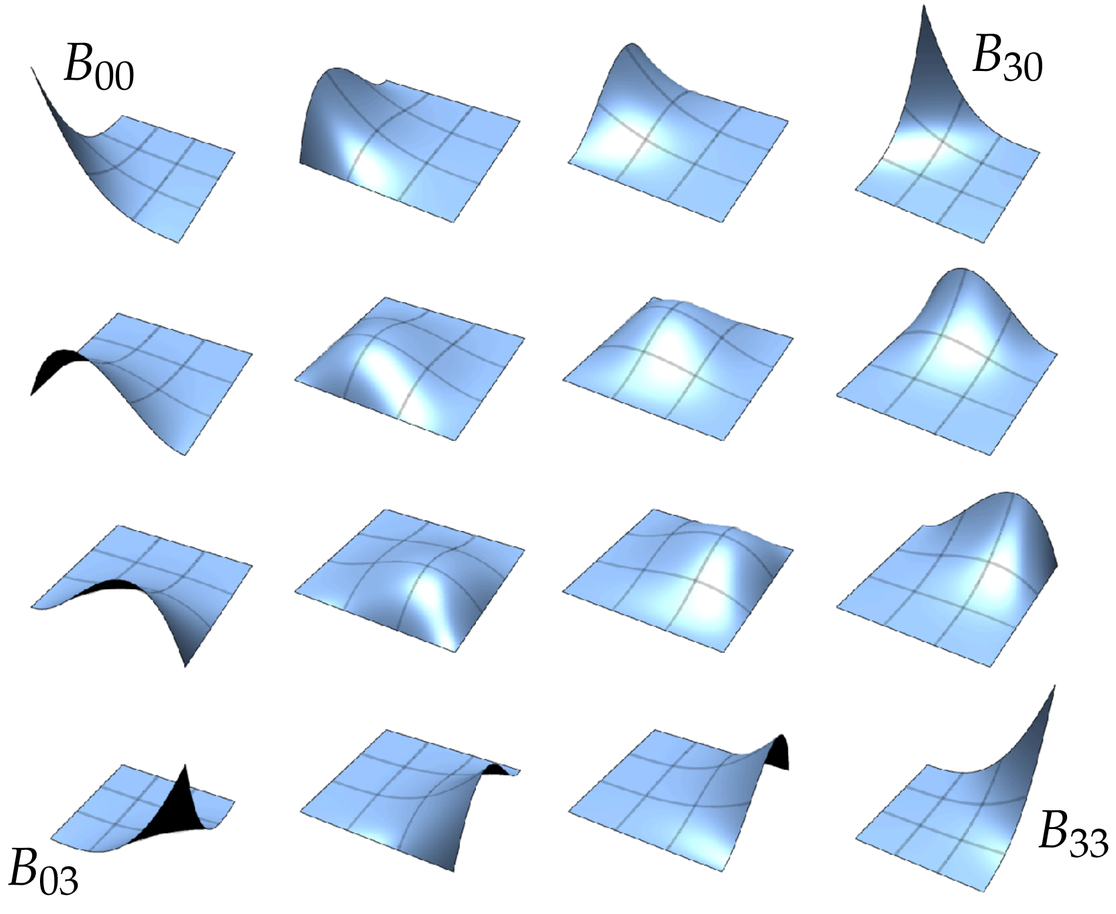
$$(f \otimes g)(u,v) := f(u)g(v)$$

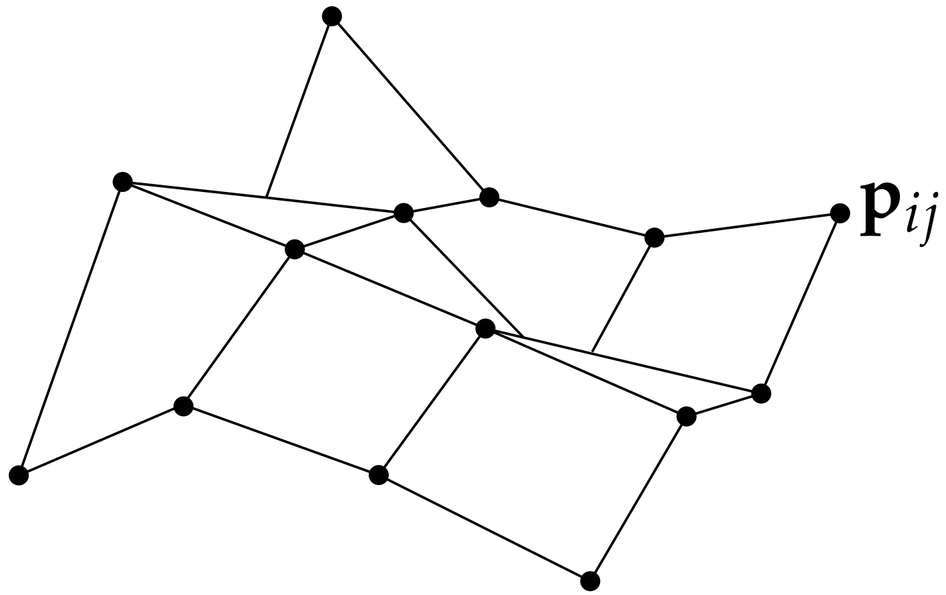
Bézier patches

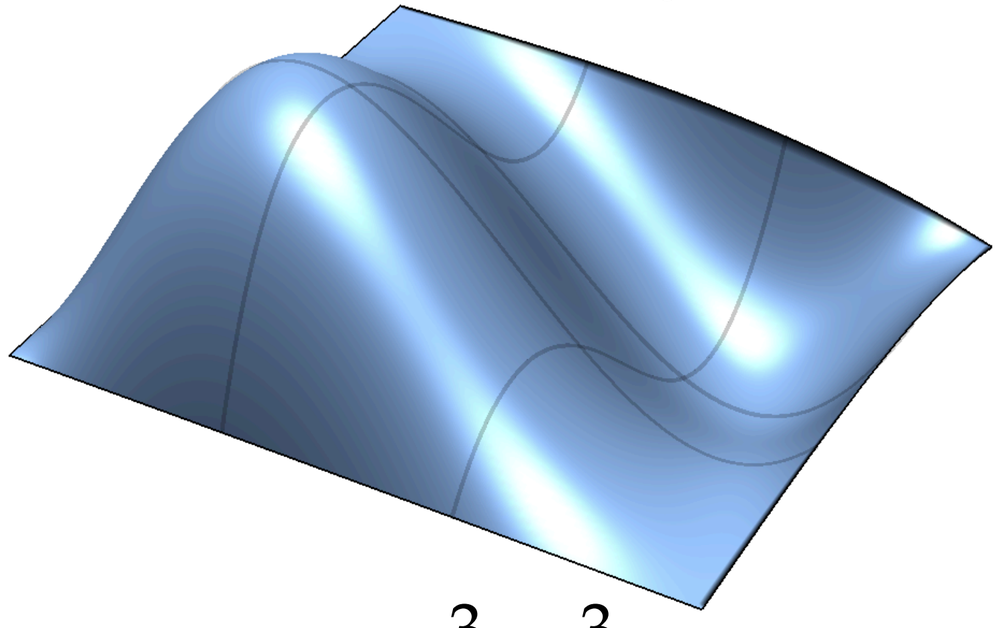
- ***Bézier patch*** **is sum of (tensor) products of Bernstein bases**



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



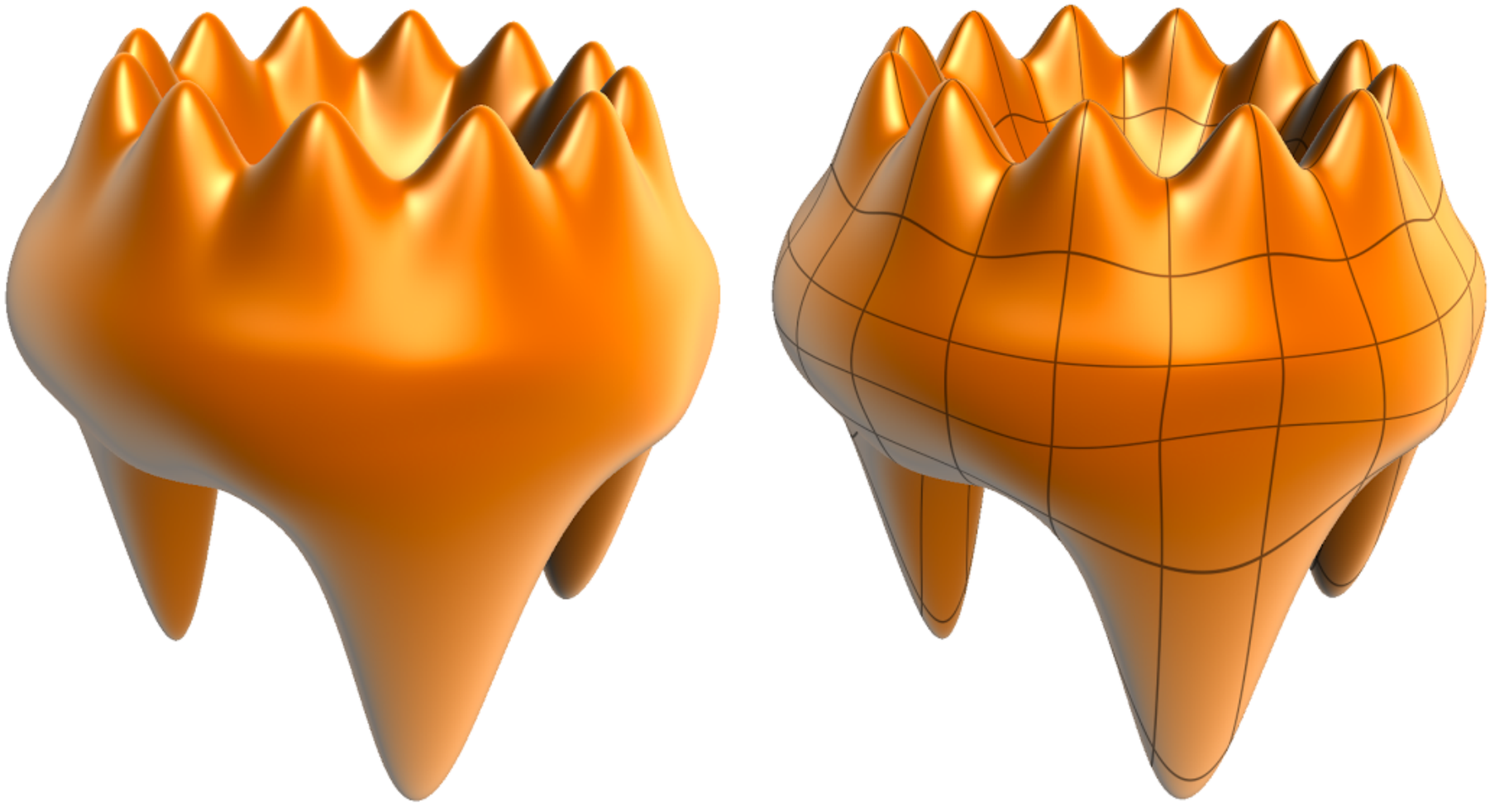




$$S(u,v) := \sum_{i=0}^{3}\sum_{j=0}^{3} B_{i,j}^3(u,v)\mathbf{p}_{ij}$$

Bézier surface

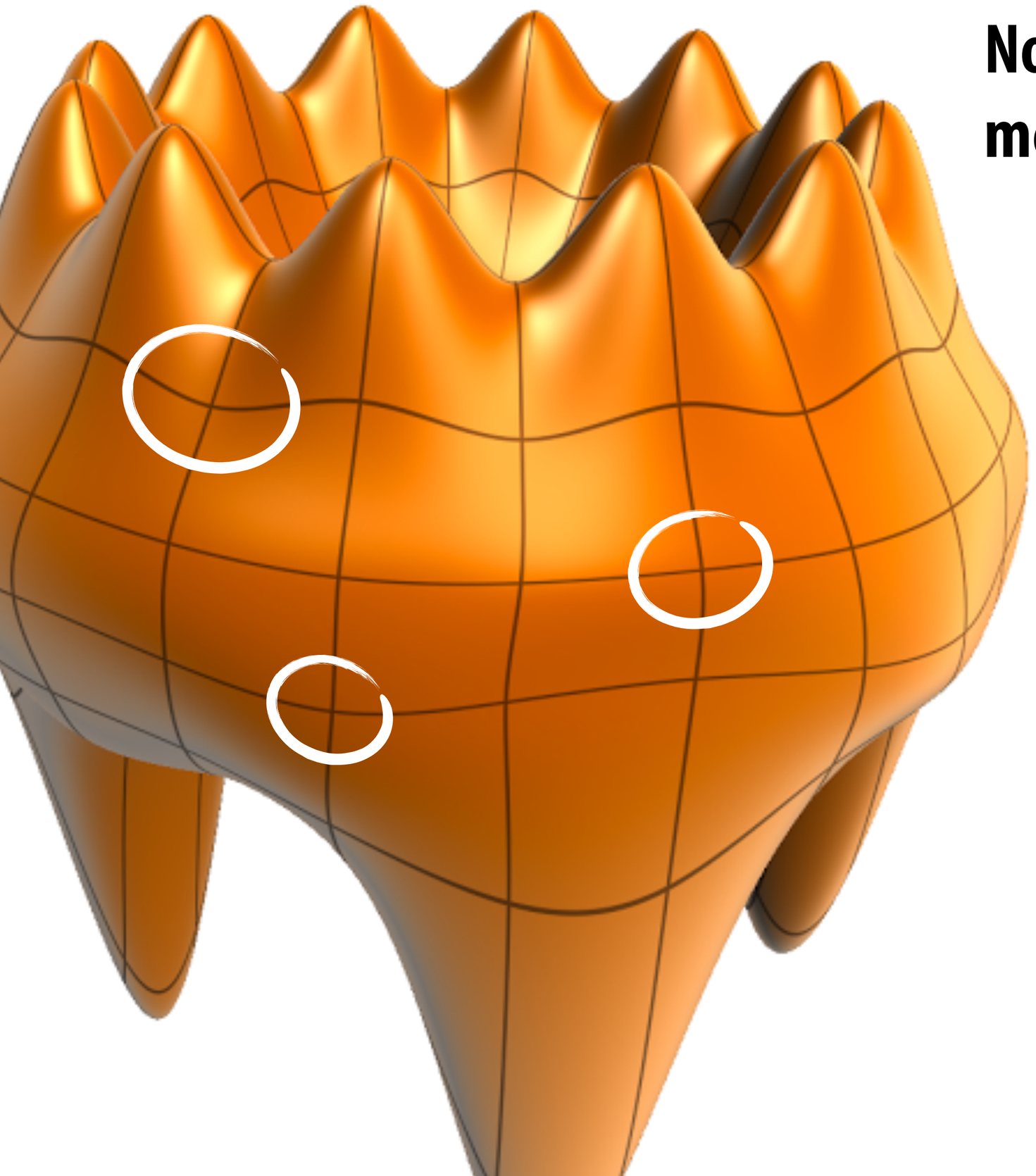
- Just as we connected Bézier *curves*, can connect Bézier *patches* to get a surface:



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

Notice anything fishy about the last picture?

Bézier patches are too simple



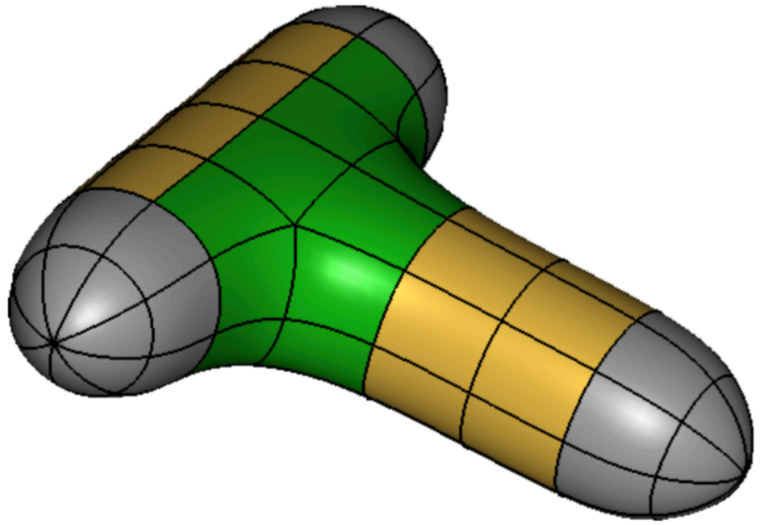
Notice that exactly four patches meet around *every* vertex!

In practice, this is far too constrained.

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

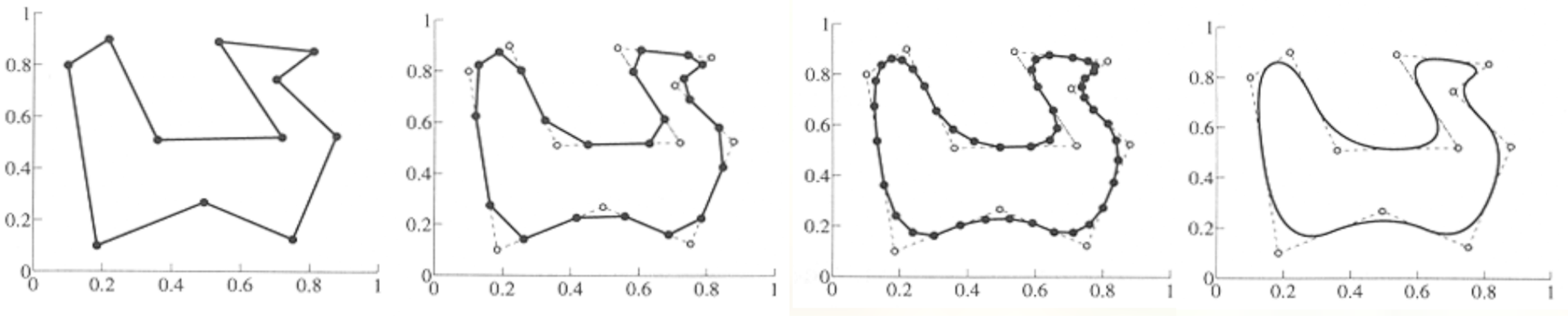
Spline patch schemes

- 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?)

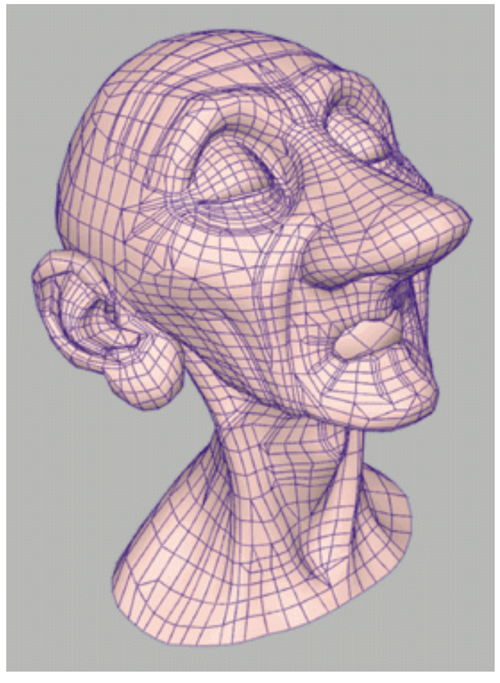
- Alternative starting point for curves/surfaces: *subdivision*
- Start with control curve
- Insert new vertex at each edge midpoint
- Update vertex positions according to fixed rule
- For careful choice of averaging rule, yields smooth curve
  - *Some subdivision schemes correspond to well-known spline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

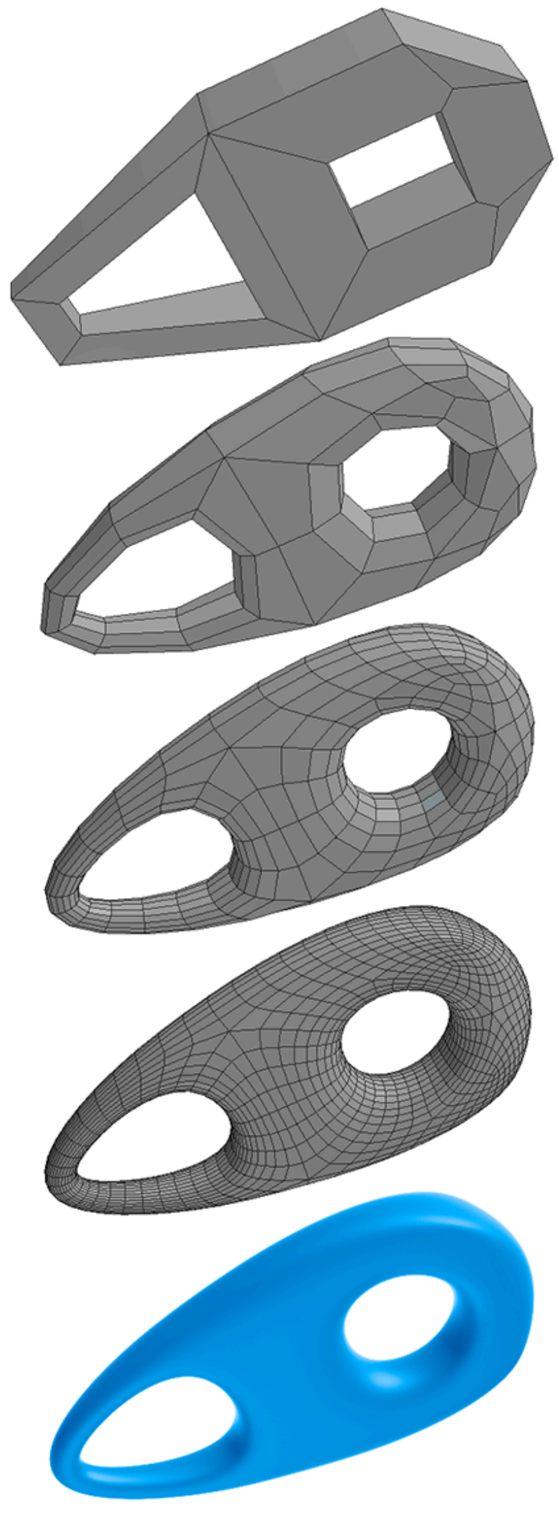


Slide cribbed from Don Fussell.

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



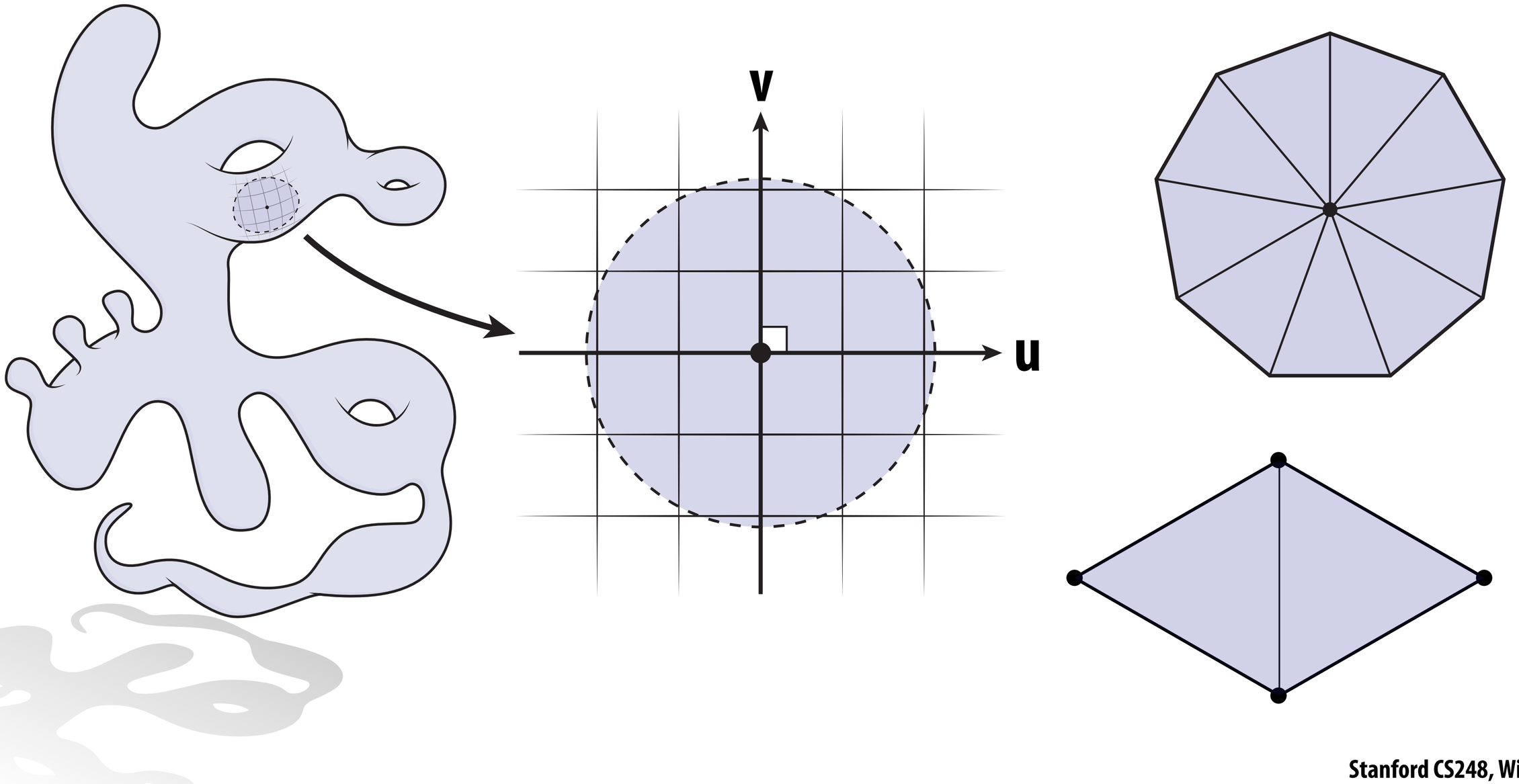


Subdivision in action (Pixar's "Geri's Game")

Surfaces and manifolds

Manifold assumption

- I'll now introduce the idea of *manifold* geometry
- Can be hard to understand motivation at first!
  - Will become more clear next class



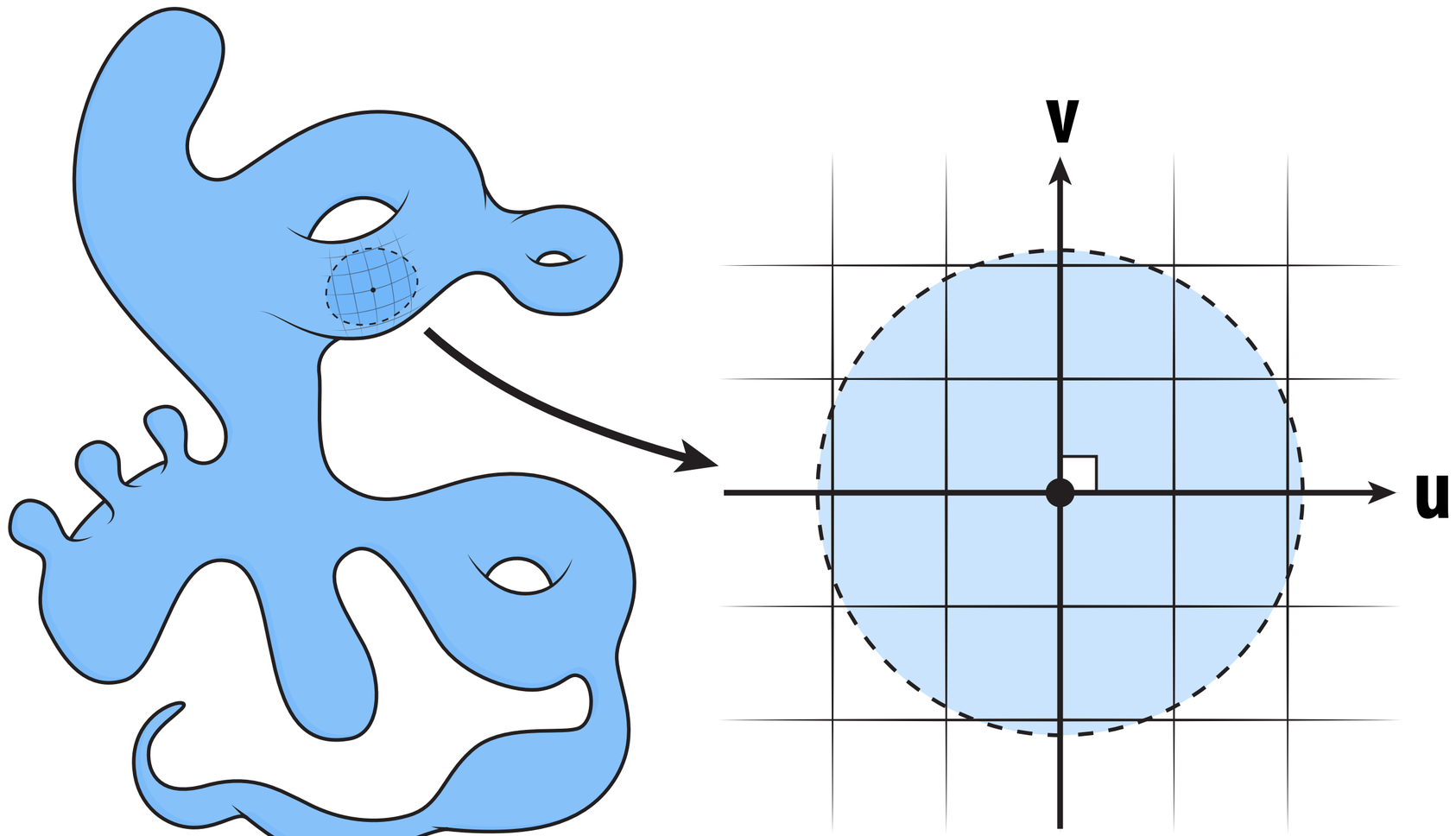
Smooth surfaces

- Intuitively, a *surface* 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 (at any point) looks like a plane*
  - E.g., the Earth from space vs. from the ground

*…or can easily be flattened into the plane, without cutting or ripping.

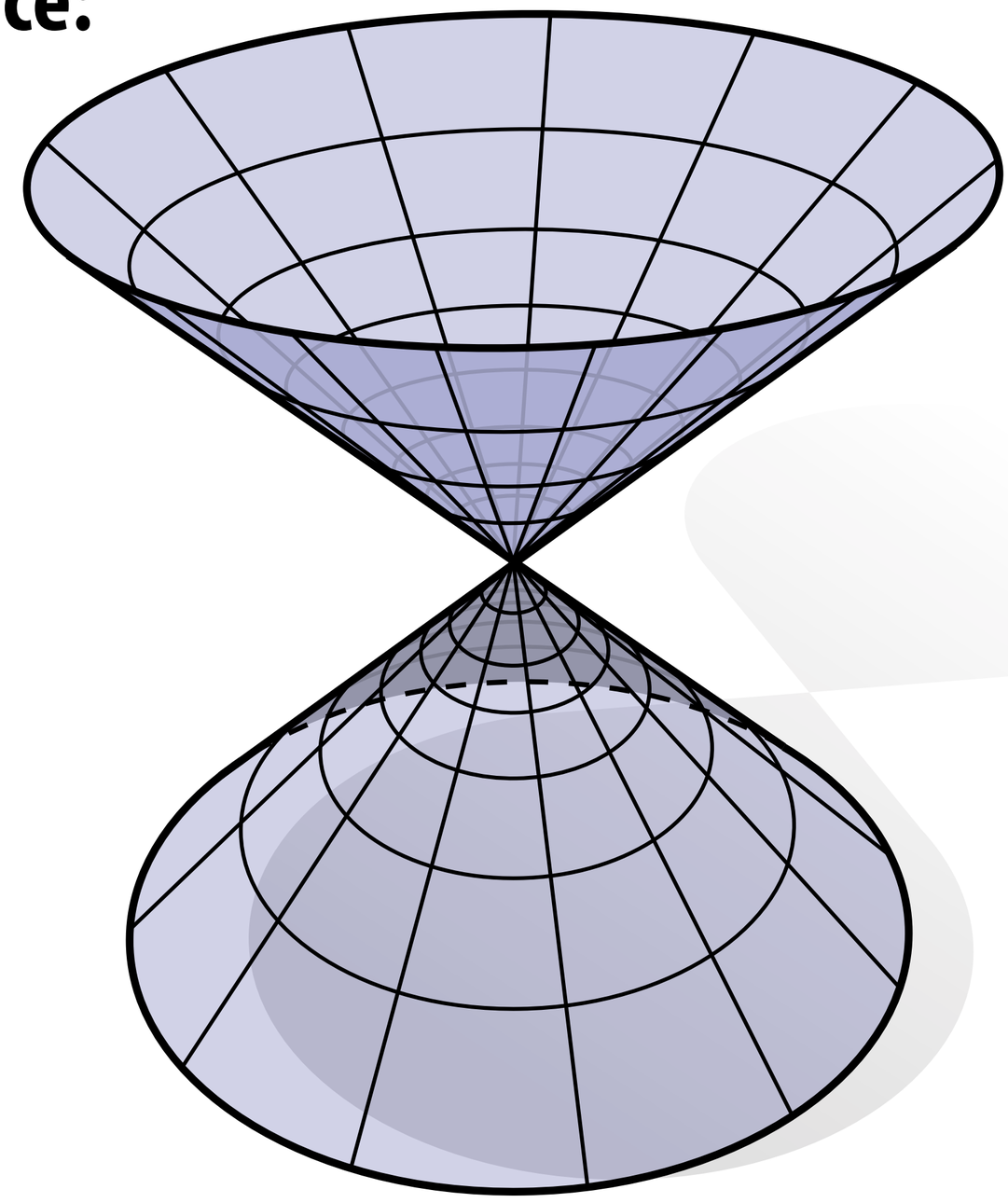
Why is the manifold property valuable?

- Makes life simple: all surfaces look the same (at least locally)
- Gives us coordinates! (at least locally)



Isn’t every shape manifold?

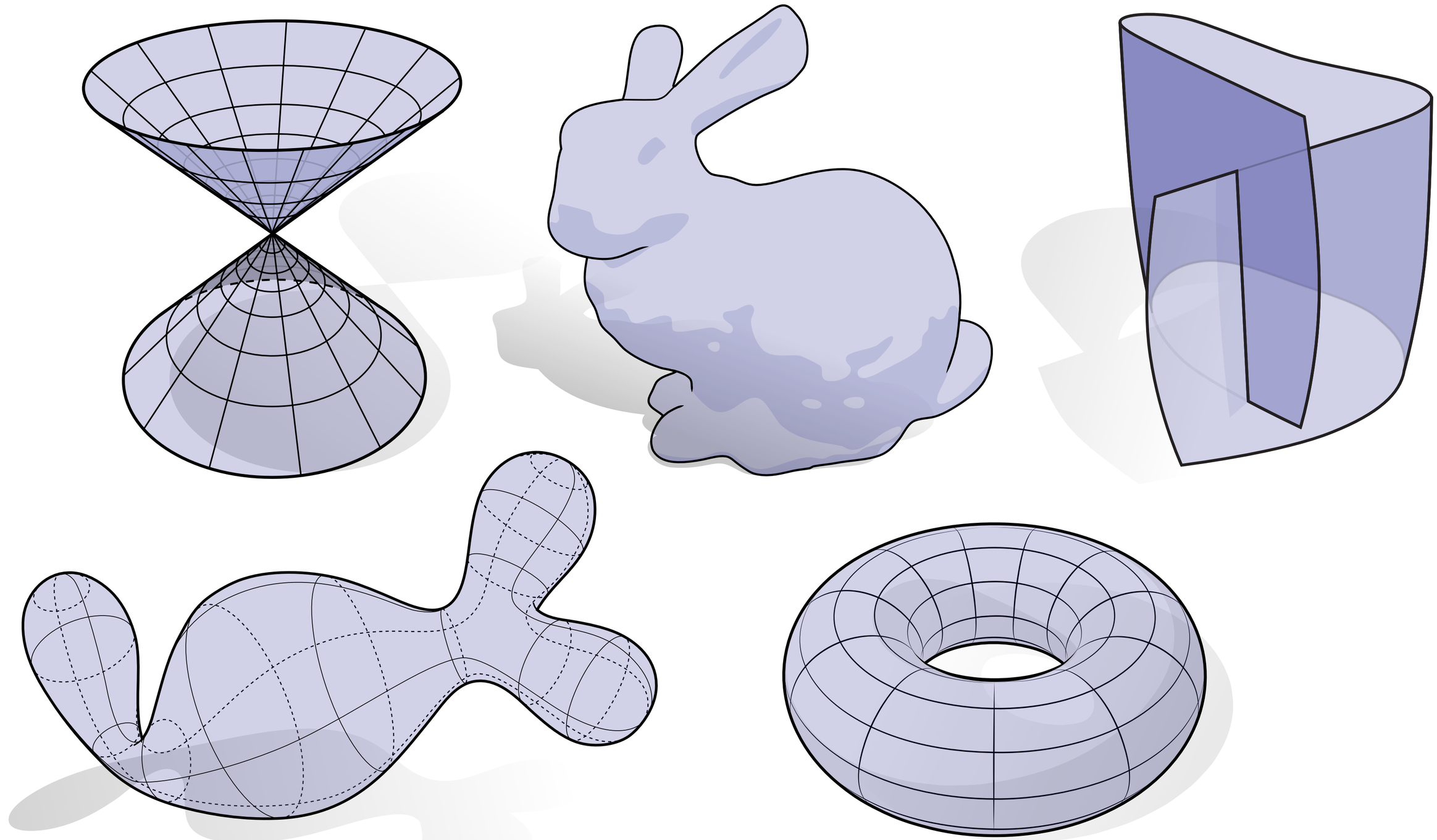
- No, for instance:



Center point never looks like the plane, no matter how close we get.

More examples of smooth surfaces

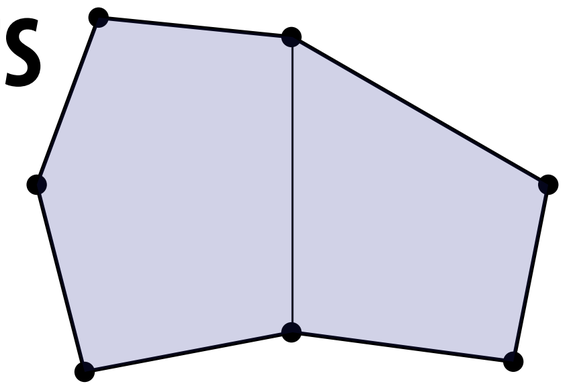
- Which of these shapes are manifold?



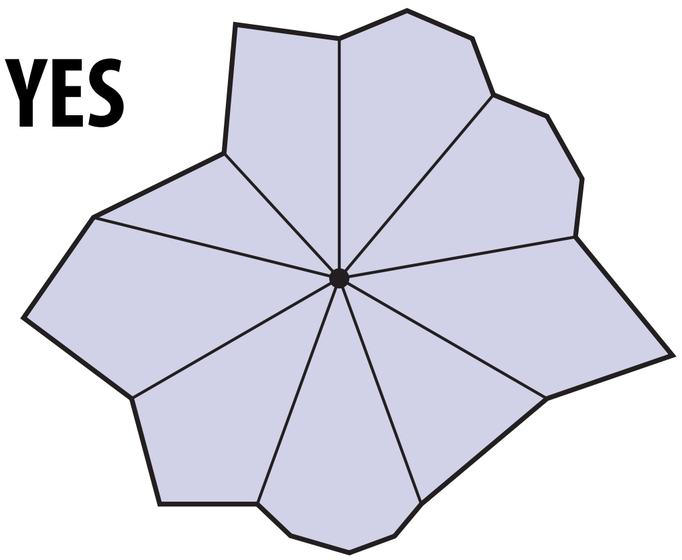
A manifold polygon mesh has fans, not fins

- 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"

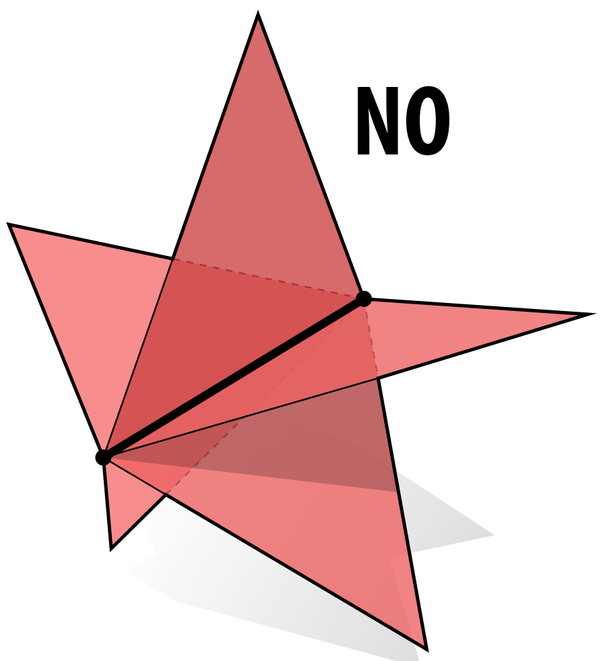
YES



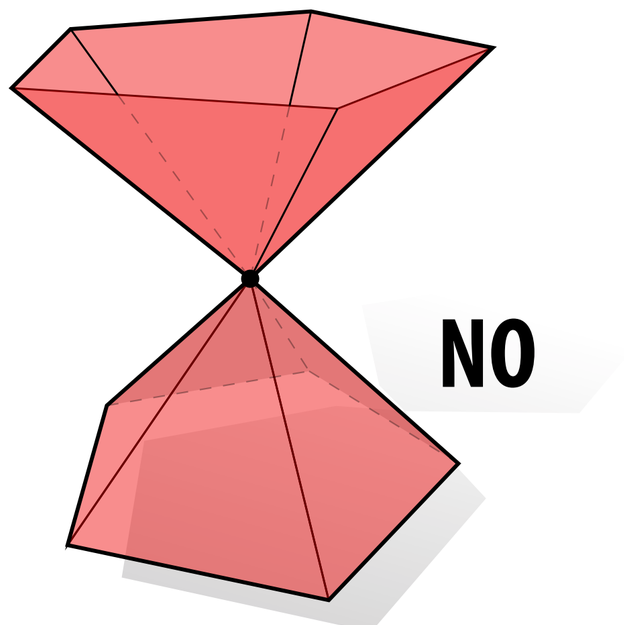
YES



NO

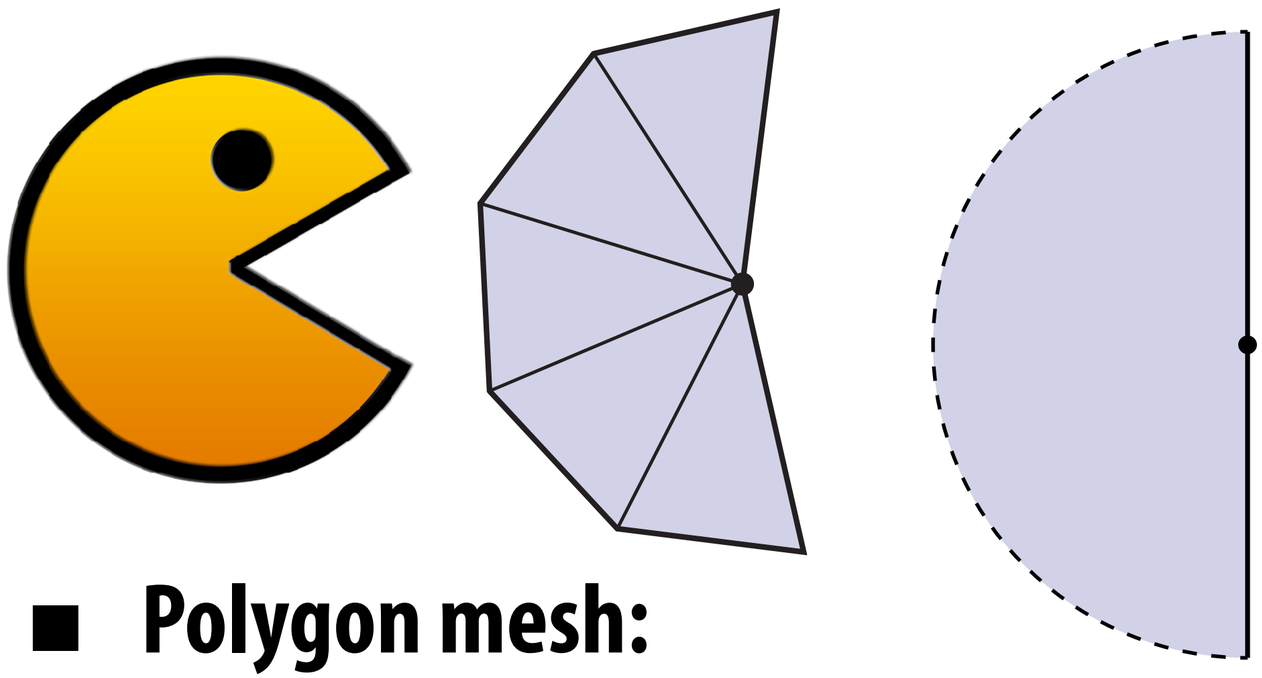


NO



What about boundary?

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



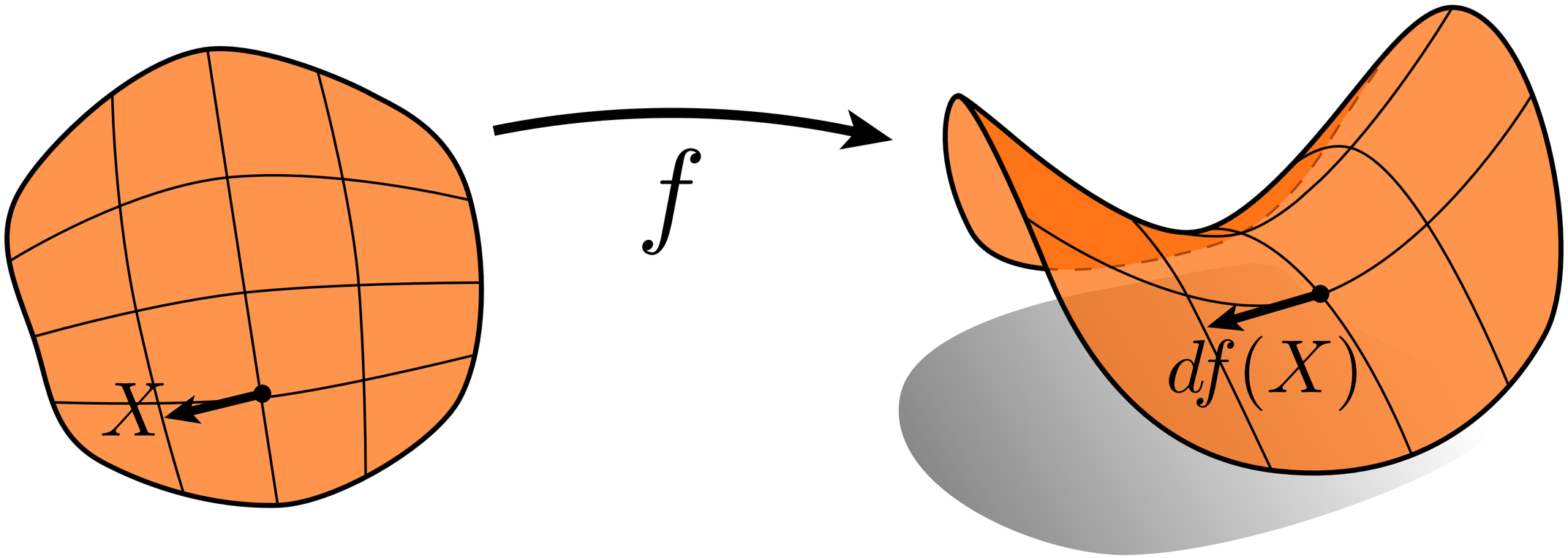
NO

YES

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

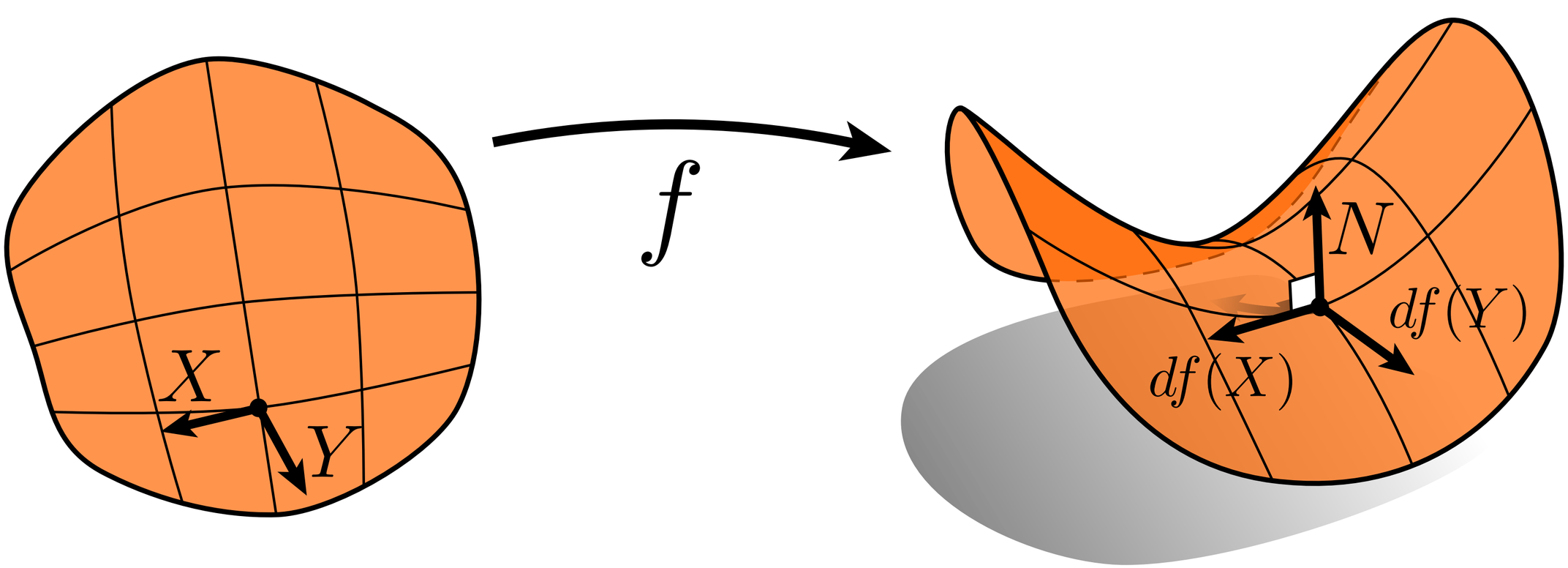
Measurements of surfaces

Surface tangent



Surface normal (N) is orthogonal to all tangents

$$N \cdot df(X) = 0 \quad \forall X$$



A common visualization of normals

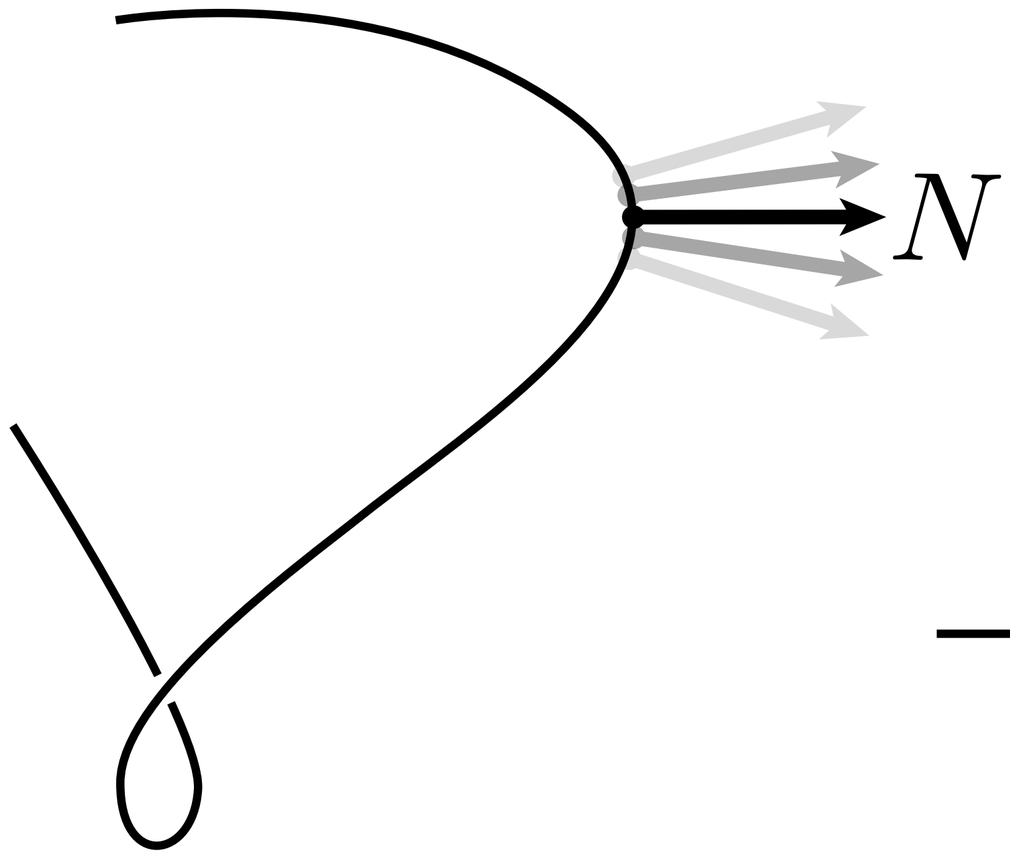
Encode normal direction as RGB color as difference from gray

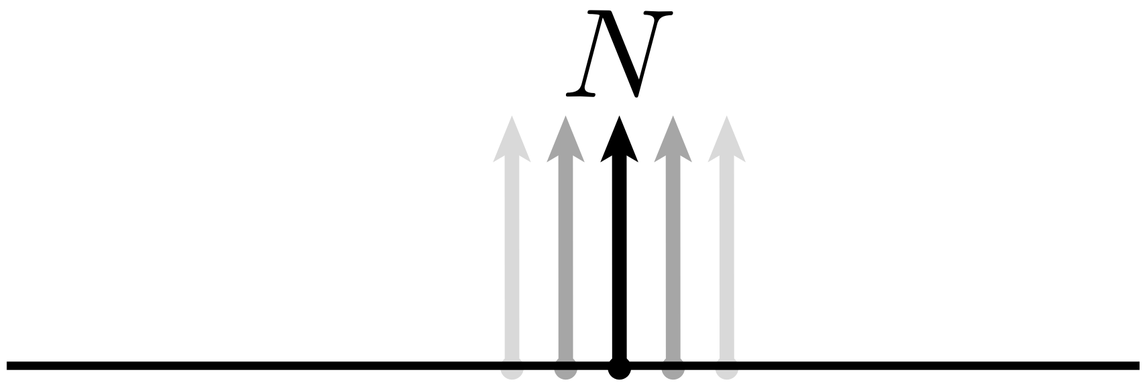
$R = 0.5 + 0.5\ N.x$
$G = 0.5 + 0.5\ N.y$
$B = 0.5 + 0.5\ N.z$

Notice: scale and bias normal values so we can represent negative components of normal as valid colors

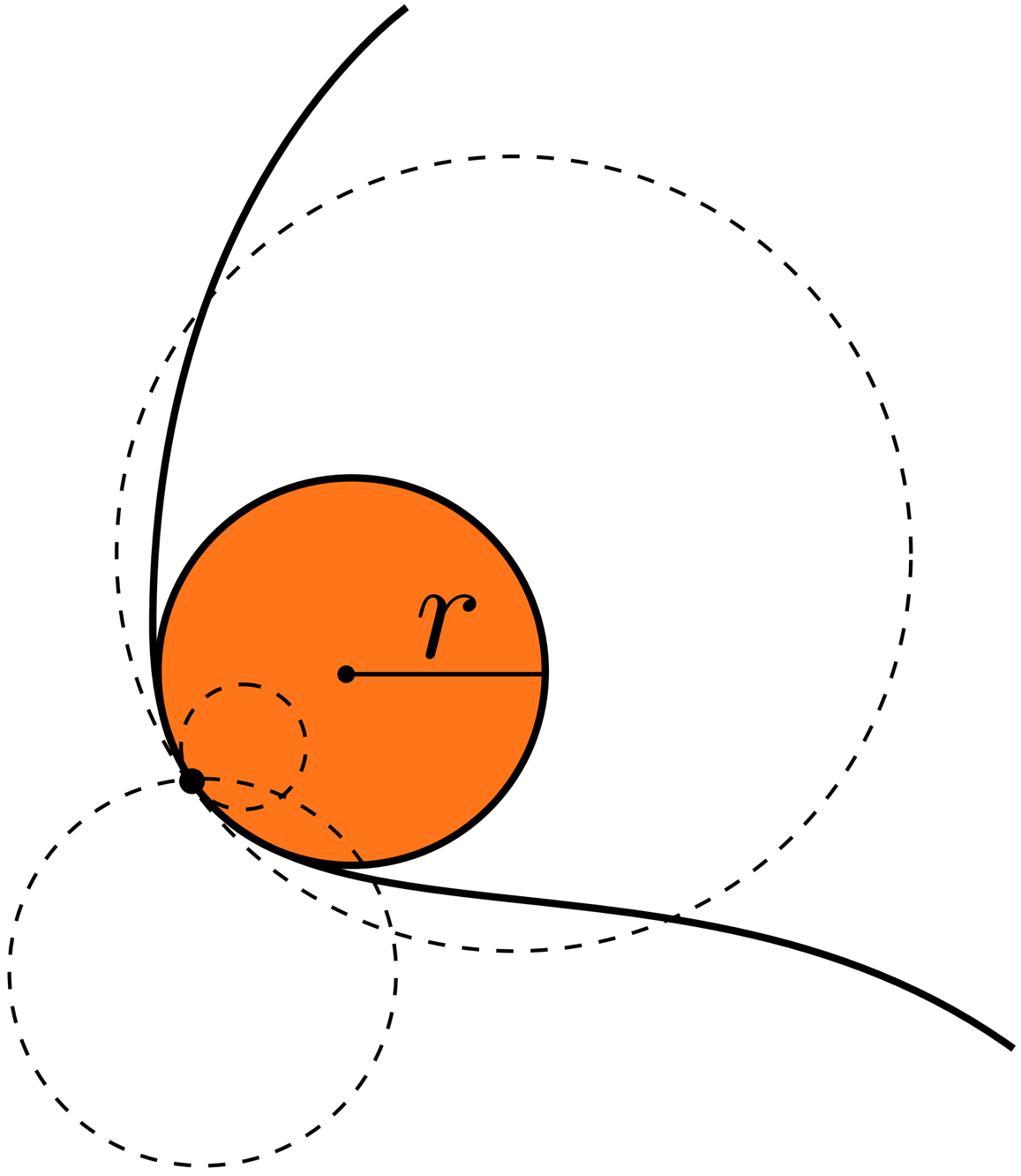
Image credit: https://www.3dgep.com/forward-plus/

Curvature is *change* in normal





Radius of curvature



$$\kappa = \frac{1}{r}$$

curvature

Acknowledgements

- Thanks to Keenan Crane and Ren Ng for slide materials