

Virtual Humans – Winter 23/24

Lecture 3_1 – Surface Representations

Prof. Dr.-Ing. Gerard Pons-Moll

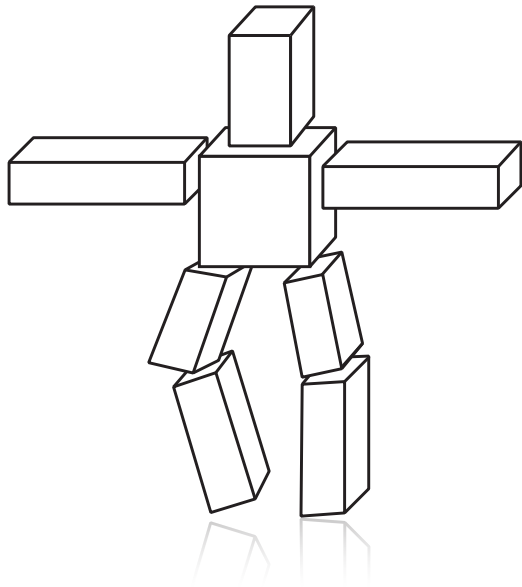
University of Tübingen / MPI-Informatics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN



Increasing complexity of our models

Transformations



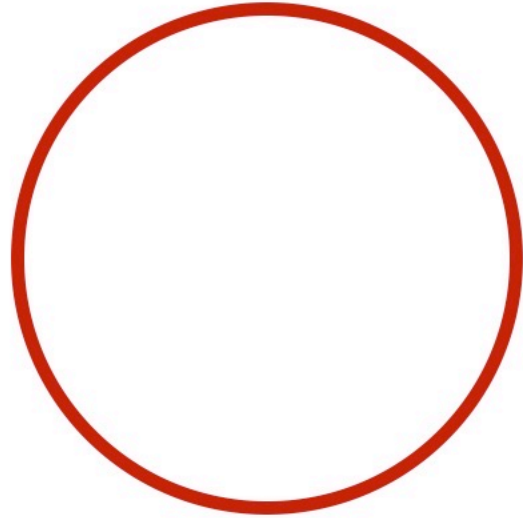
Geometry



Materials, lighting, ...



How can we describe geometry?



How can we describe geometry?

IMPLICIT

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

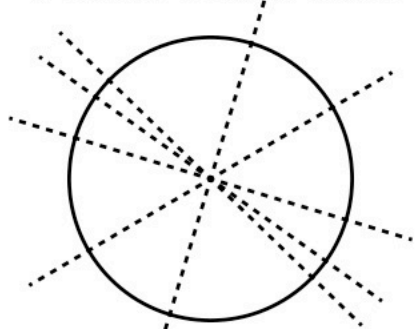
LINGUISTIC

“unit circle”

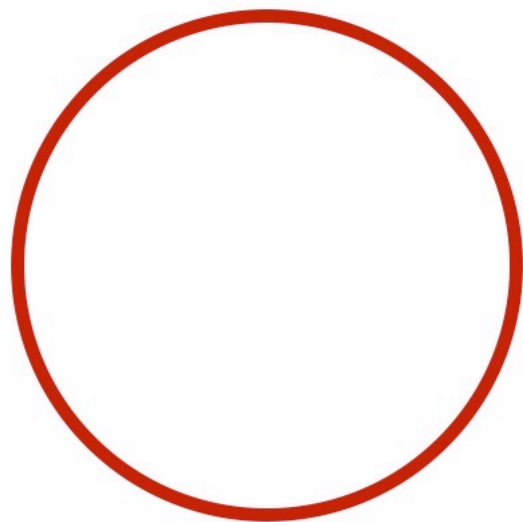
EXPLICIT

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

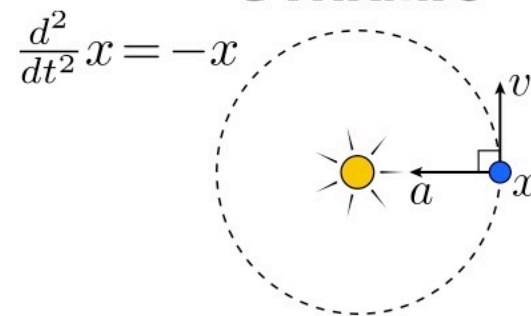
TOMOGRAPHIC



(constant density)



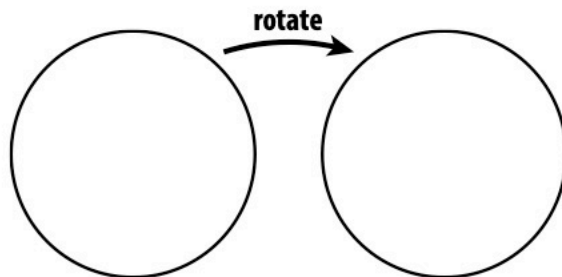
DYNAMIC



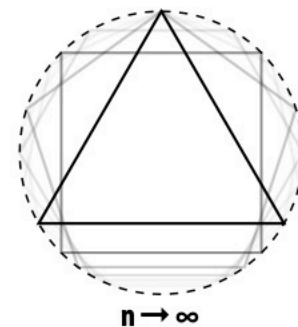
CURVATURE

$$\kappa = 1$$

SYMMETRIC



DISCRETE



$n \rightarrow \infty$

Examples of geometry



Examples of geometry



Examples of geometry



Examples of geometry

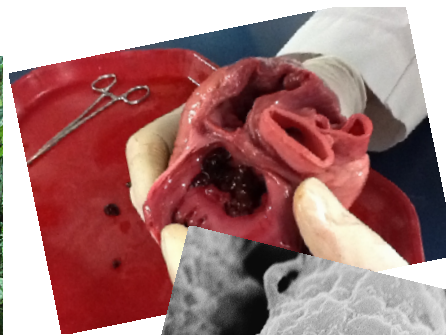


Examples of geometry



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

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

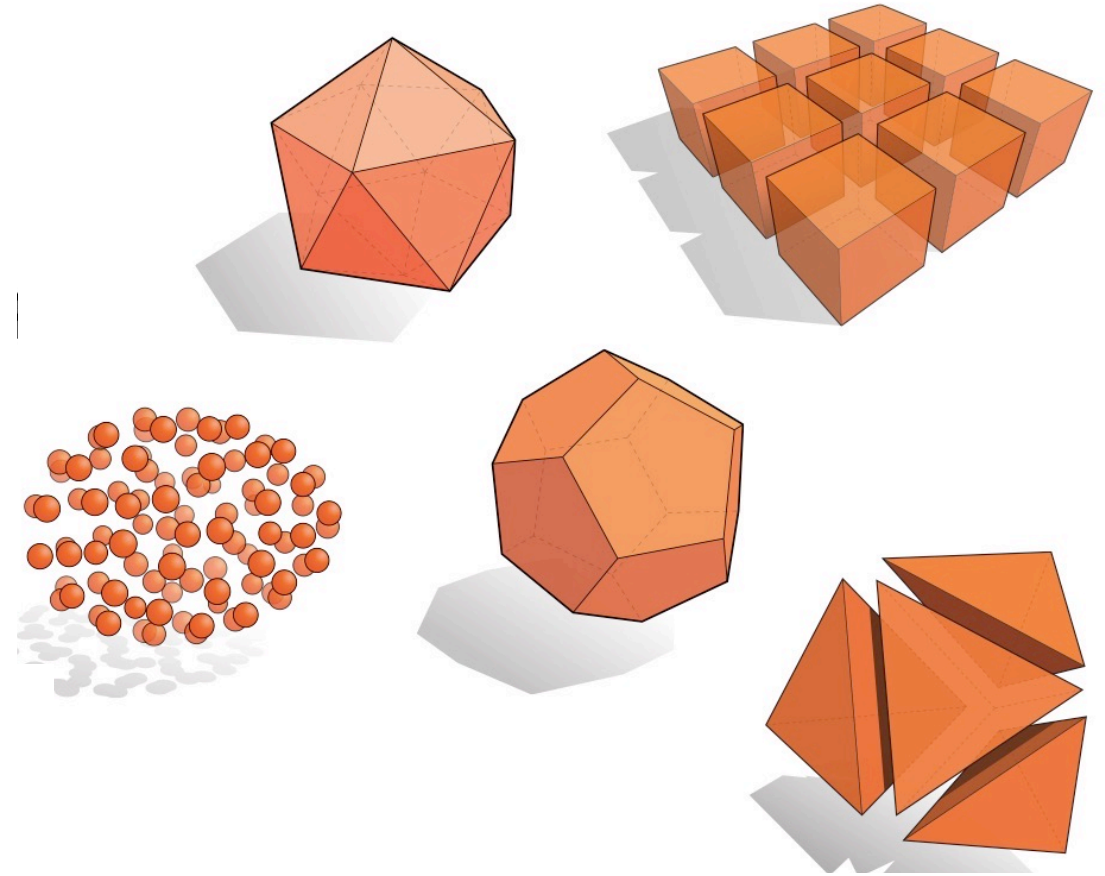
- EXPLICIT

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

- IMPLICIT

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

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



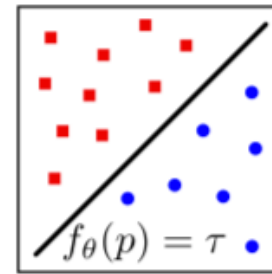
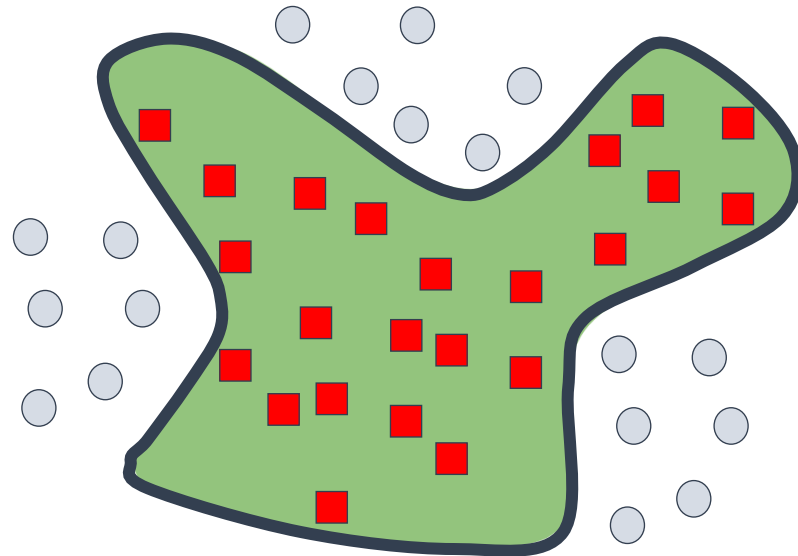
"Implicit" Representations of Geometry

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

Surfaces as an Implicit Function

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3$$

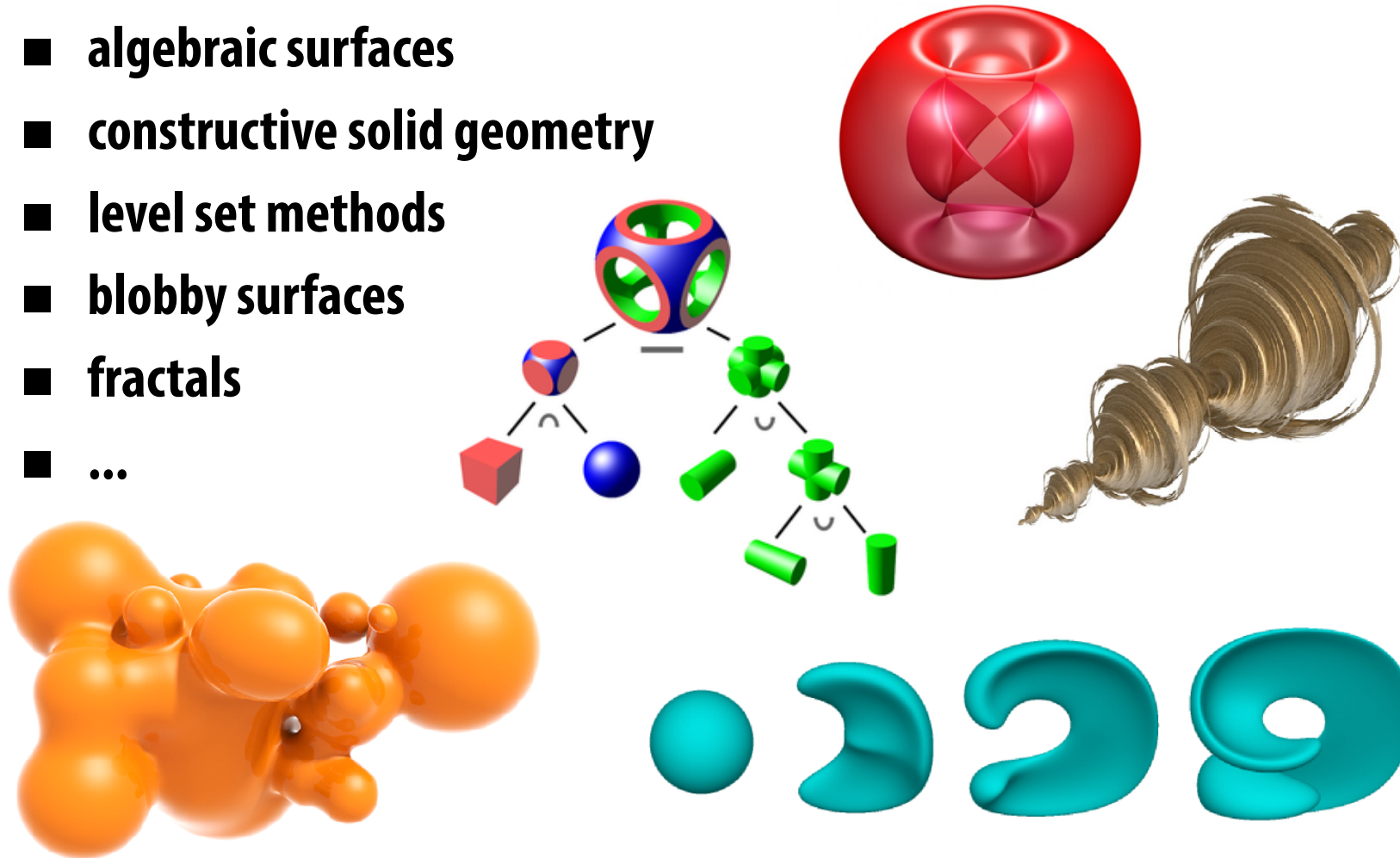
$$f(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in \text{outside } \circ \\ 1, & \text{if } \mathbf{p} \in \text{inside } \blacksquare \end{cases}$$



$$\mathcal{S} = \{\mathbf{p}, f(\mathbf{p}) = \tau\}$$

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:

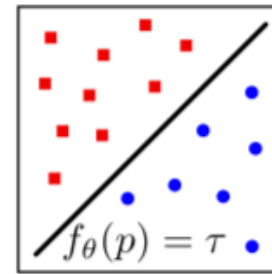
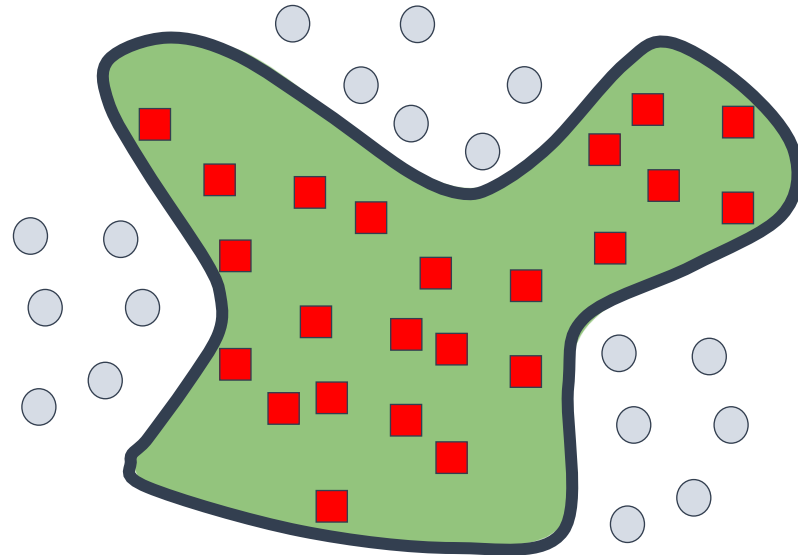
I'm thinking of an implicit surface $f(x,y,z)=0$.

Find *any* point on it.

Surfaces as an Implicit Function

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3$$

$$f(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in \text{outside } \circ \\ 1, & \text{if } \mathbf{p} \in \text{inside } \blacksquare \end{cases}$$



$$\mathcal{S} = \{\mathbf{p}, f(\mathbf{p}) = \tau\}$$

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.

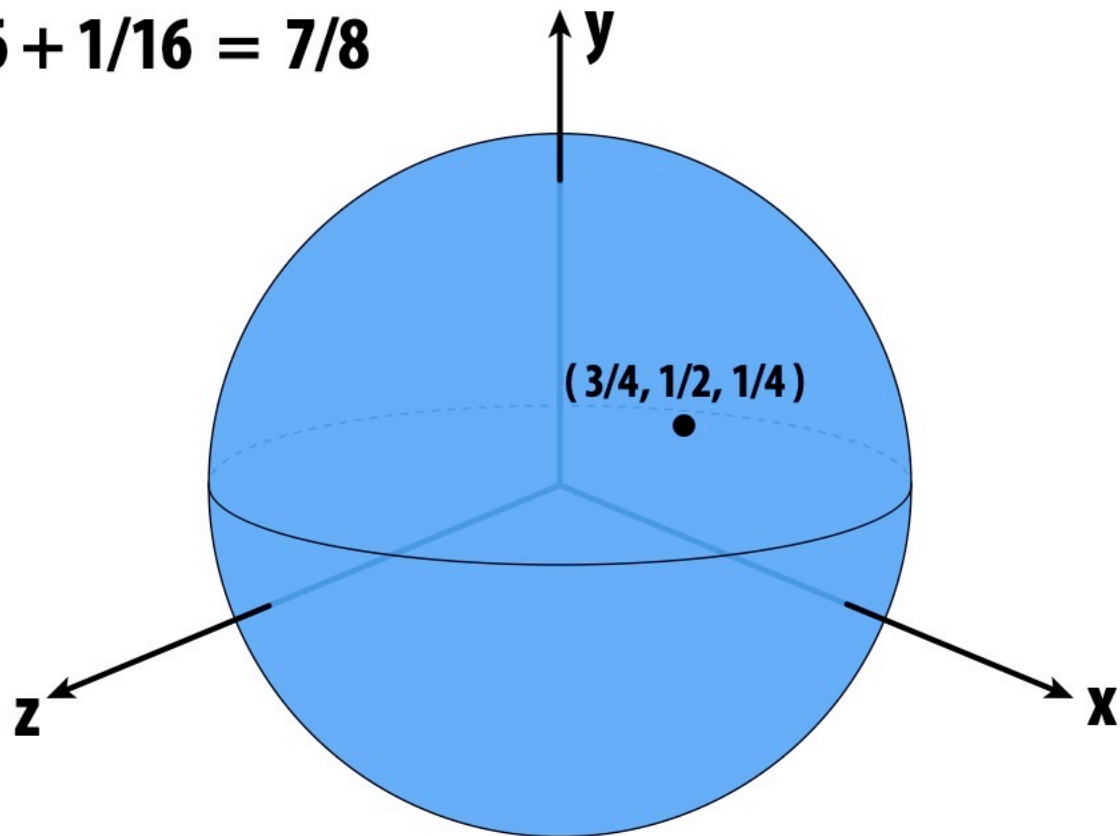
Check if this point is inside the unit sphere

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).

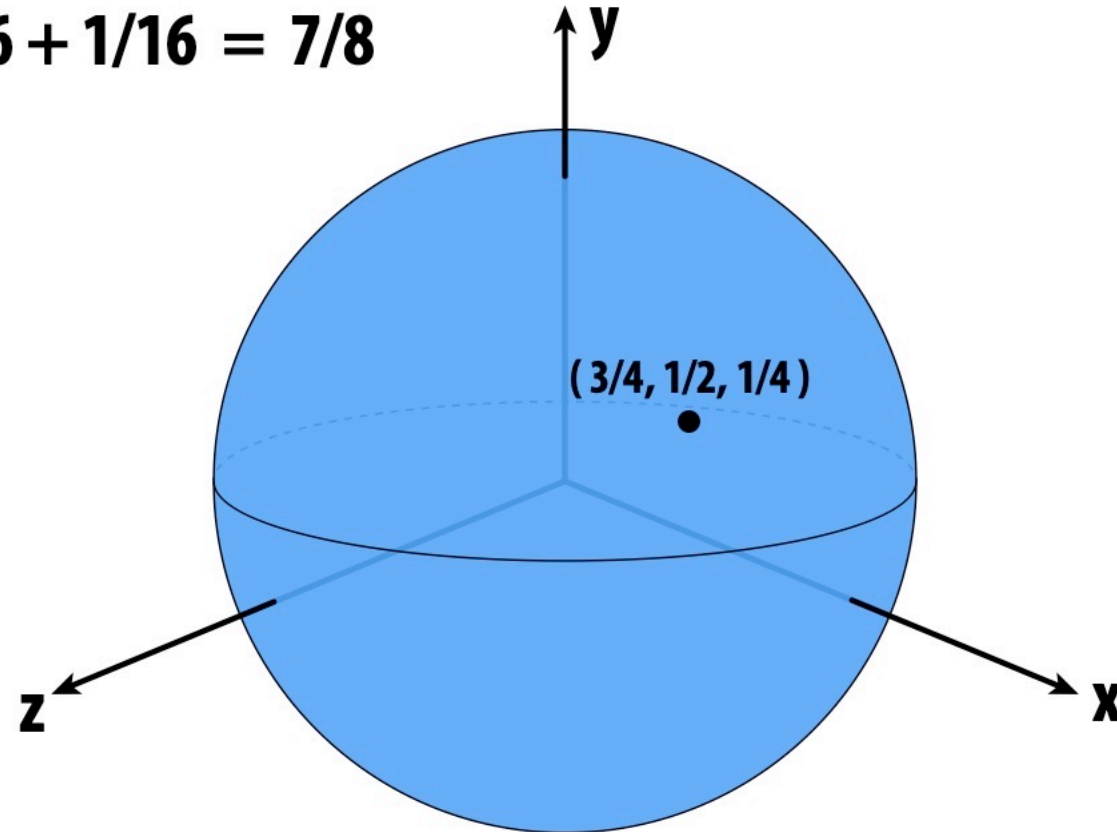
Check if this point is inside the unit sphere

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

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

$$7/8 < 1$$

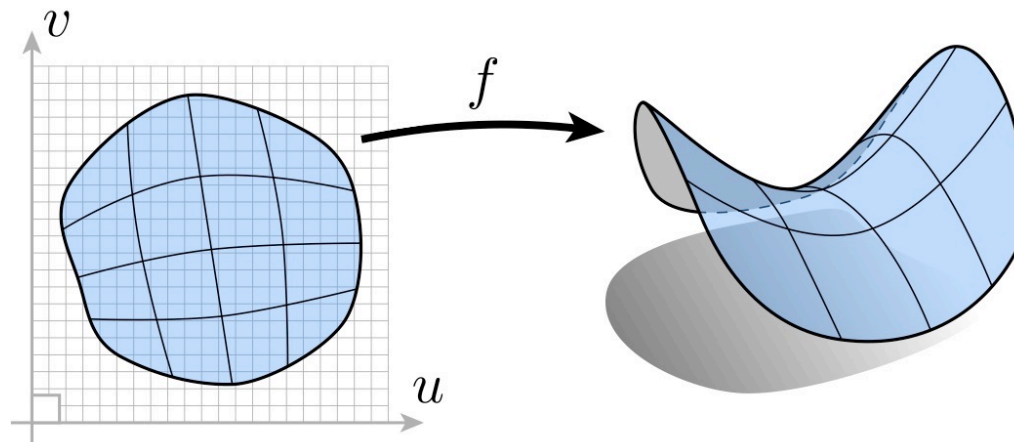
YES.



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

"Explicit" Representations of Geometry

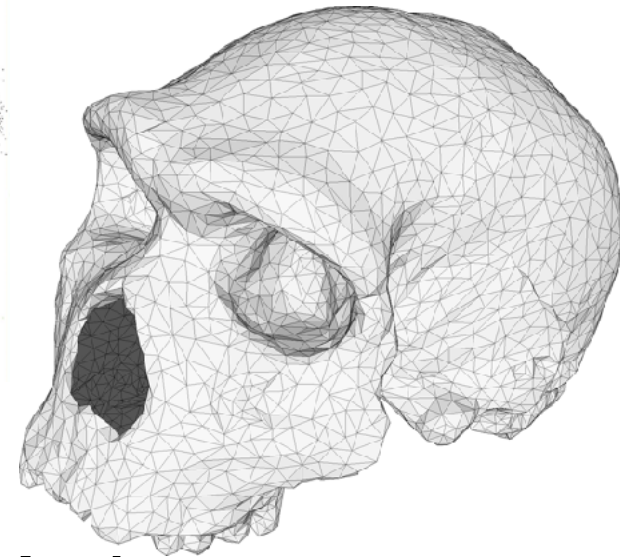
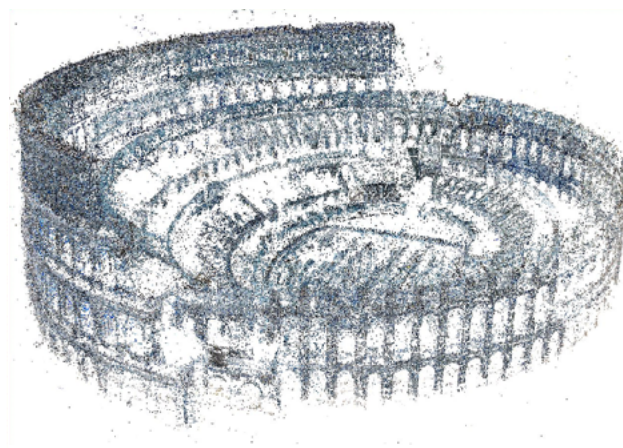
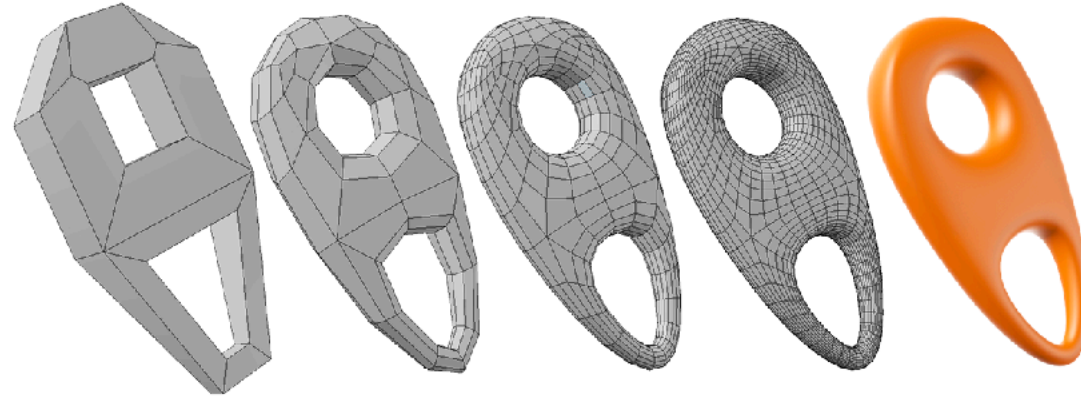
- All points are given directly
- E.g., points on sphere are $(\cos(u) \sin(v), \sin(u) \sin(v), \cos(v))$,
for $0 \leq u < 2\pi$ and $0 \leq v \leq \pi$
- More generally: $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



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

Many explicit representations in graphics

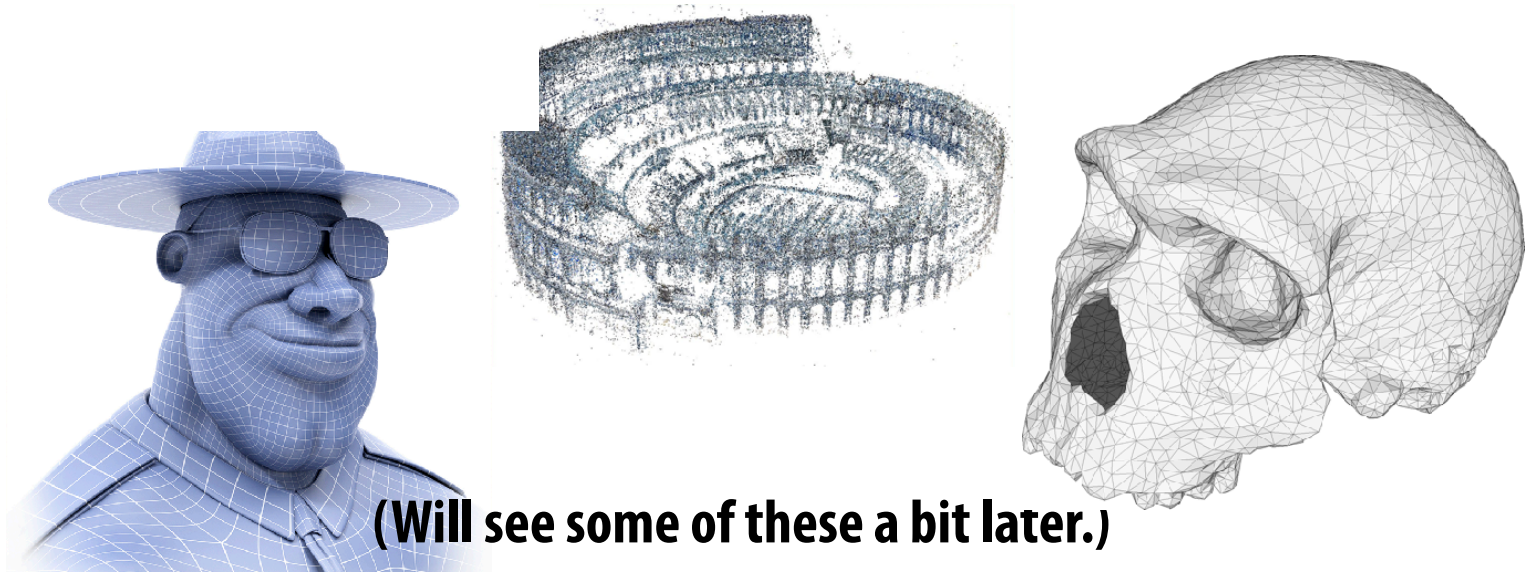
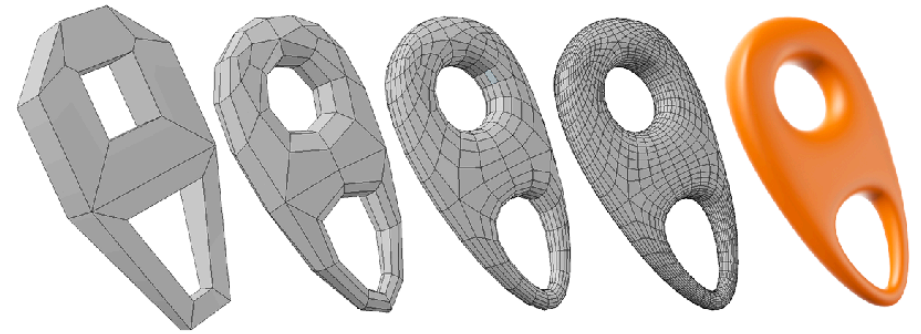
- triangle meshes
- polygon meshes
- subdivision surfaces
- NURBS
- point clouds
- ...



(Will see some of these a bit later.)

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:

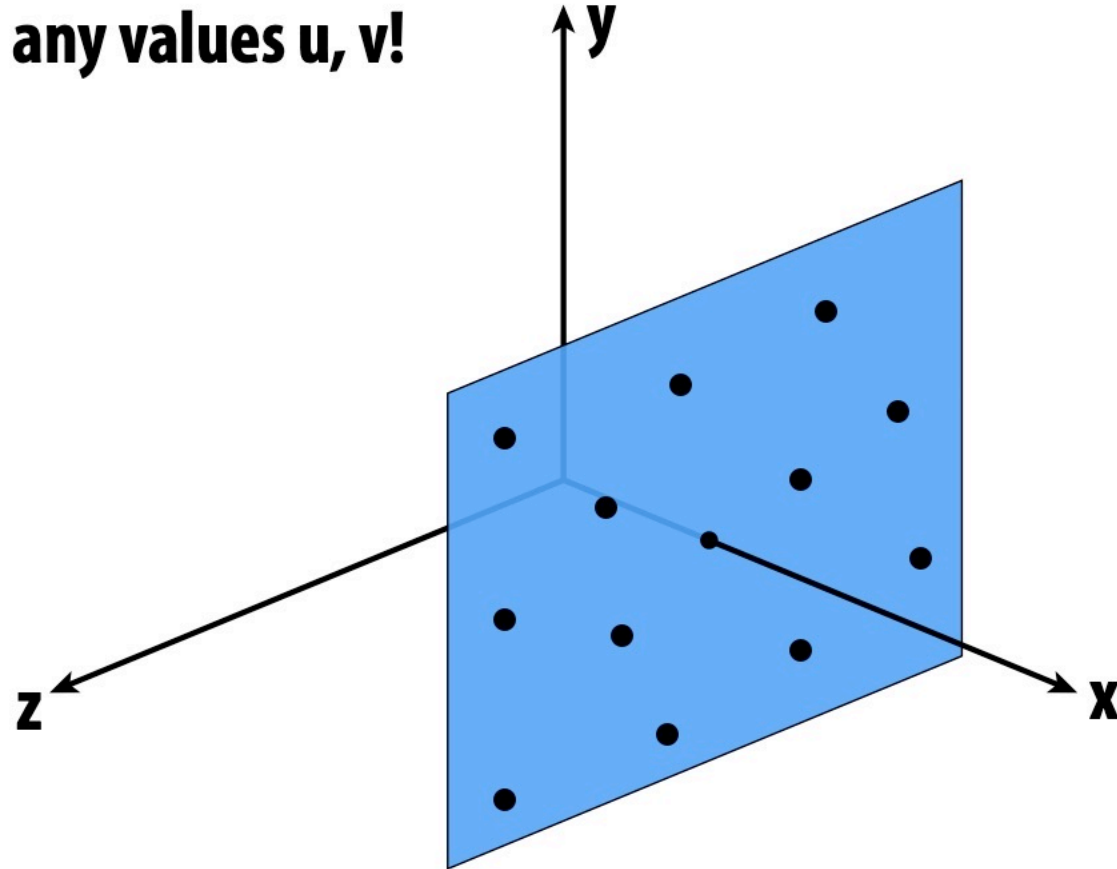
I'll give you an explicit surface.

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 $f(u,v)$.

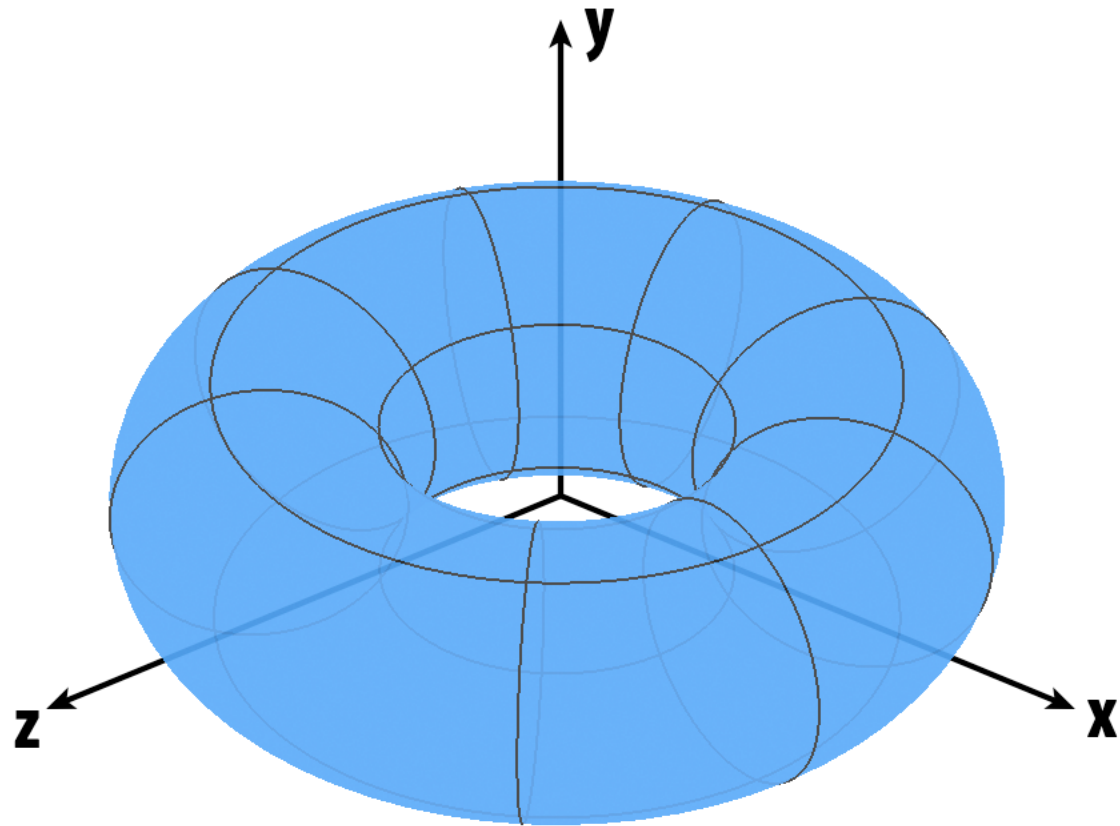
I want to see if a point is *inside* it.

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:

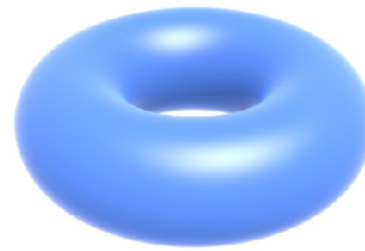
Some representations work better than others—depends on the task!

Algebraic Surfaces (Implicit)

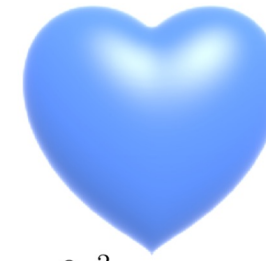
- Surface is zero set of a polynomial in x, y, z
- Examples:



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



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



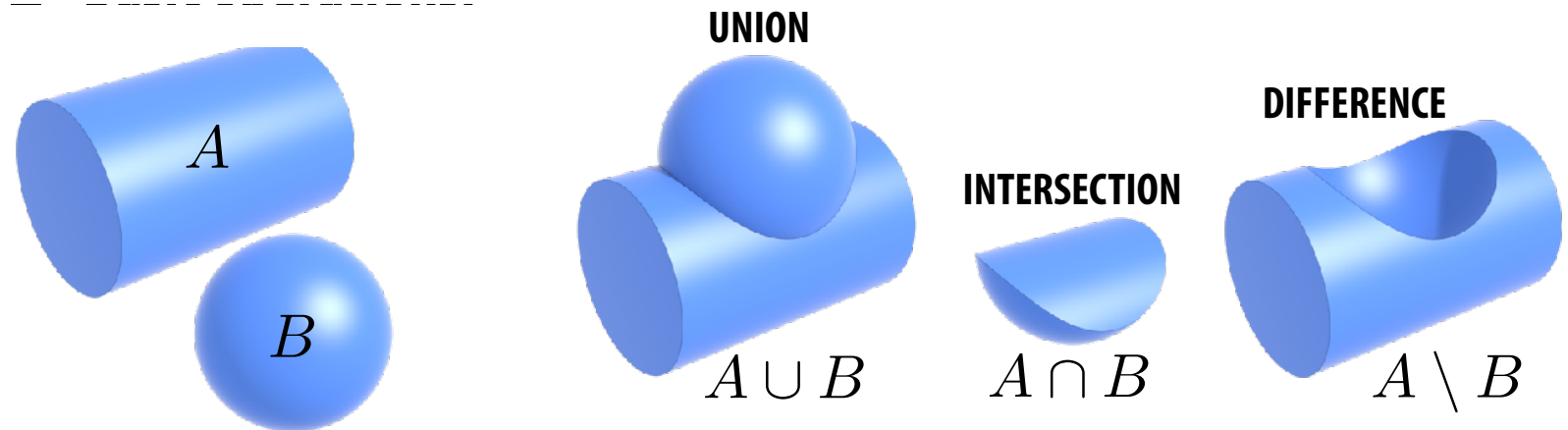
$$(x^2 + \frac{9y^2}{4} + z^2 - 1)^3 = x^2 z^3 + \frac{9y^2 z^3}{80}$$

- What about more complicated shapes?
- Very hard to come up with polynomials!

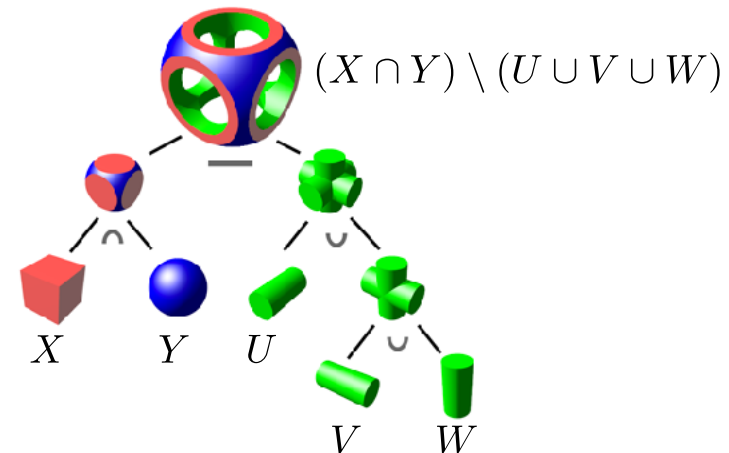


Constructive Solid Geometry (Implicit)

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

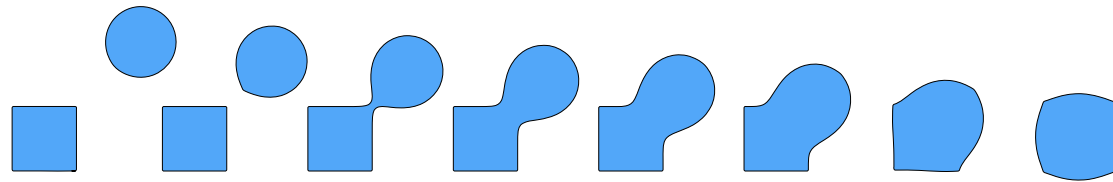


- Then chain together expressions:



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 made of pure signed distance functions

Art with math -- really hard!

Shadertoy Browse New Sign In



19.59 39.7 fps 1280 x 720 REC 🔊 ⌵

Slisesix (2008)

Views: 3904, Tags: procedural, 3d, raymarching, distancefield, sdf, demoscene Created by iq in 2021-07-22 🔗 ❤️ 84

A Shader I made in 2008. While not my first raymarched SDF, this was my first SDF based content creation exercise. Lots of the usual SDF techniques are used, although soft shadows and smooth minimum were not fully evolved yet.

Comments (15) [Sign in](#) to post a comment.

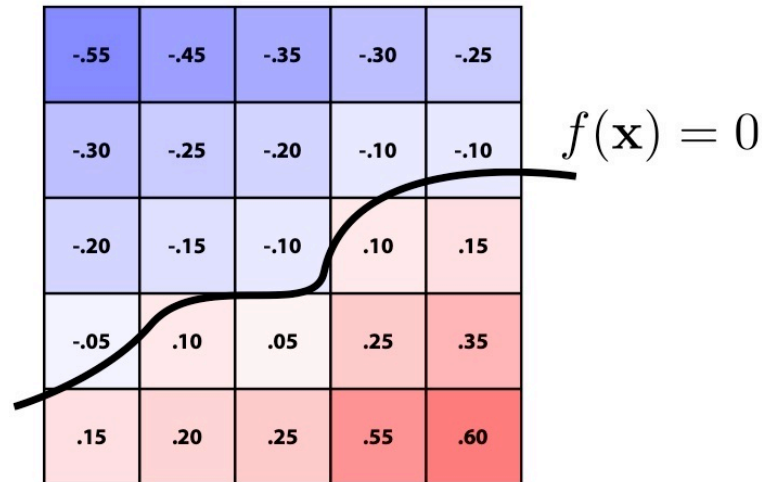
```
127
128 float columna( vec3 pos, float offx )
129 {
130     float x = pos.x;
131     float y = pos.y;
132     float z = pos.z;
133
134     float y2=y-0.40;
135     float y3=y-0.35;
136     float y4=y-1.00;
137
138     float di = udSqBox( vec3(x, y, z), vec3(0.10, 1.00, 0.10) );
139     di = min( di, udSqBox( vec3(x, y, z), vec3(0.12, 0.40, 0.12) ) );
140     di = min( di, udSqBox( vec3(x, y, z), vec3(0.05, 0.35, 0.14) ) );
141     di = min( di, udSqBox( vec3(x, y, z), vec3(0.14, 0.35, 0.05) ) );
142     di = min( di, udSqBox( vec3(x, y4, z), vec3(0.14, 0.02, 0.14) ) );
143     di = min( di, udSqBox( vec3(x-y2)*0.7071, (y2+x)*0.7071, z), vec3(0.10*0.7071, 0.10*0.7071, 0.10*0.7071) );
144     di = min( di, udSqBox( vec3(x, (y2+z)*0.7071, (z-y2)*0.7071), vec3(0.12, 0.10*0.7071, 0.10*0.7071) );
145     di = min( di, udSqBox( vec3(x-y3)*0.7071, (y3+x)*0.7071, z), vec3(0.10*0.7071, 0.10*0.7071, 0.10*0.7071) );
146     di = min( di, udSqBox( vec3(x, (y3+z)*0.7071, (z-y3)*0.7071), vec3(0.14, 0.10*0.7071, 0.10*0.7071) );
147
148     #if 1
149     float fb = fb( vec3(10.1-w+offx, 10.1-w, 10.1-w) );
```

Compiled in 0.1 secs 8868 chars ⌵ ⌵ ?

iChannel0 iChannel1 iChannel2 iChannel3

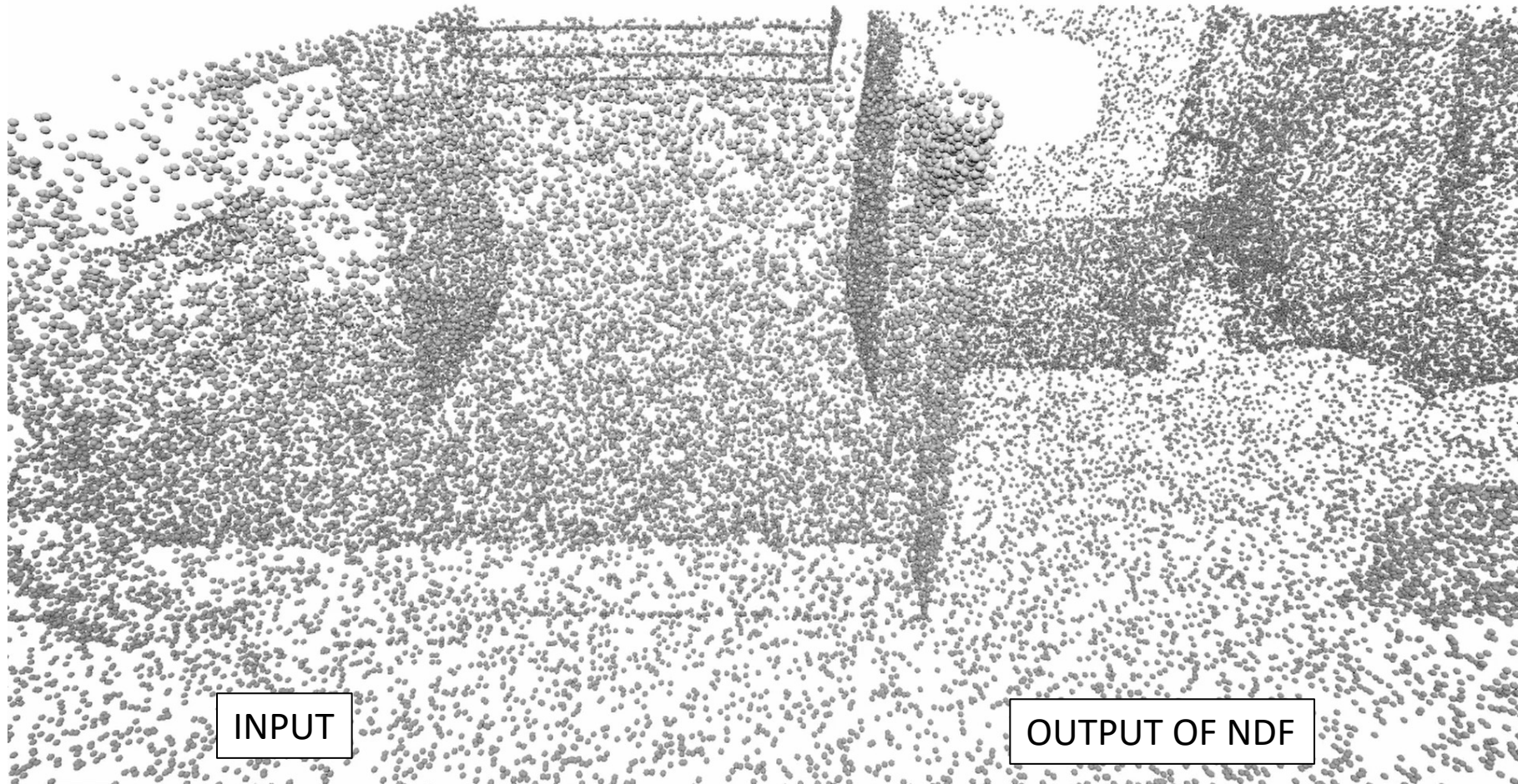
Level Set Methods (Implicit)

- 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



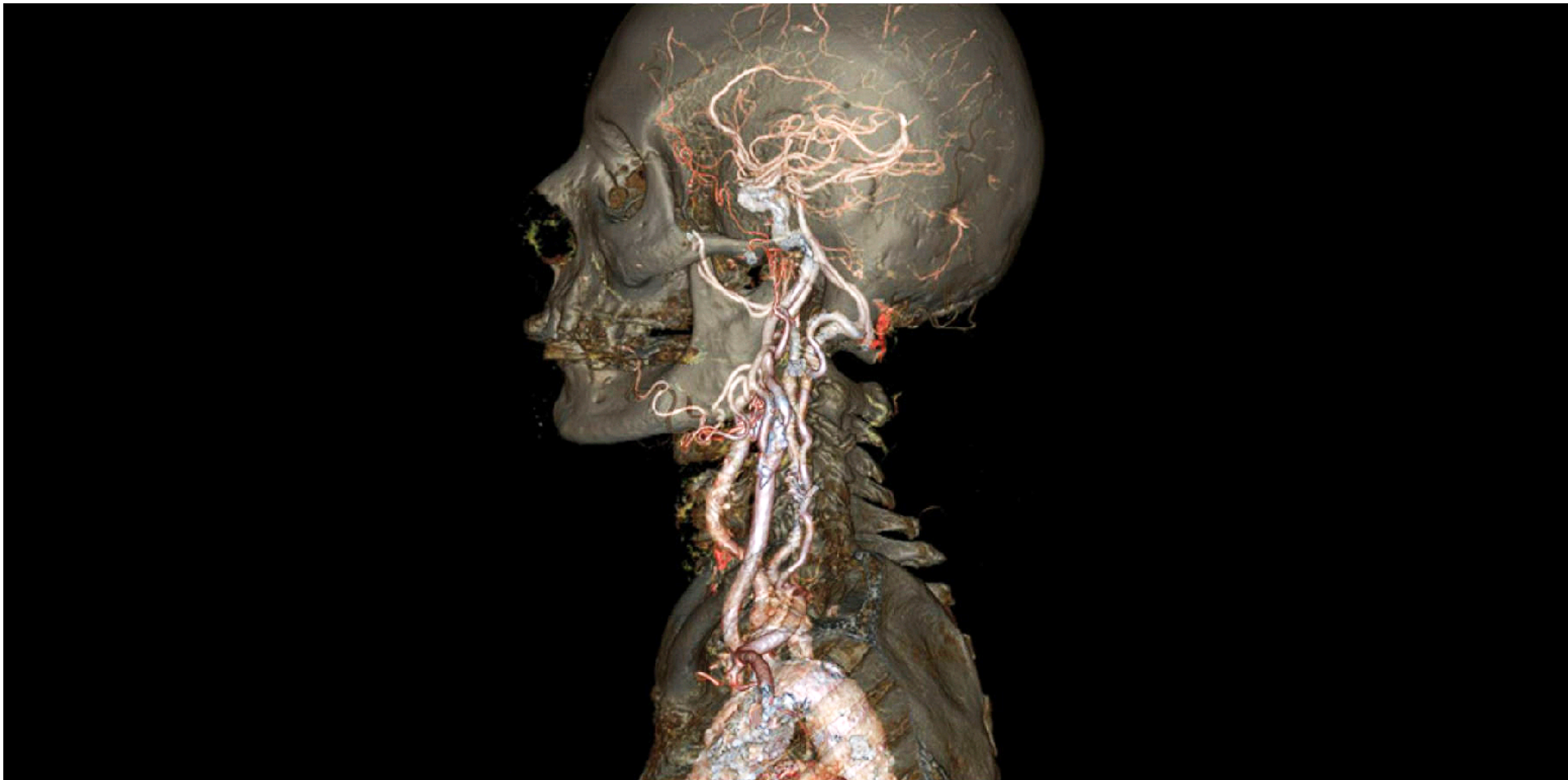
- Surface is found where interpolated values equal zero
- Provides much more explicit control over shape (like a texture)
- Unlike closed-form expressions, run into problems of aliasing!

Nowadays -- Neural Distance Fields (NDF)



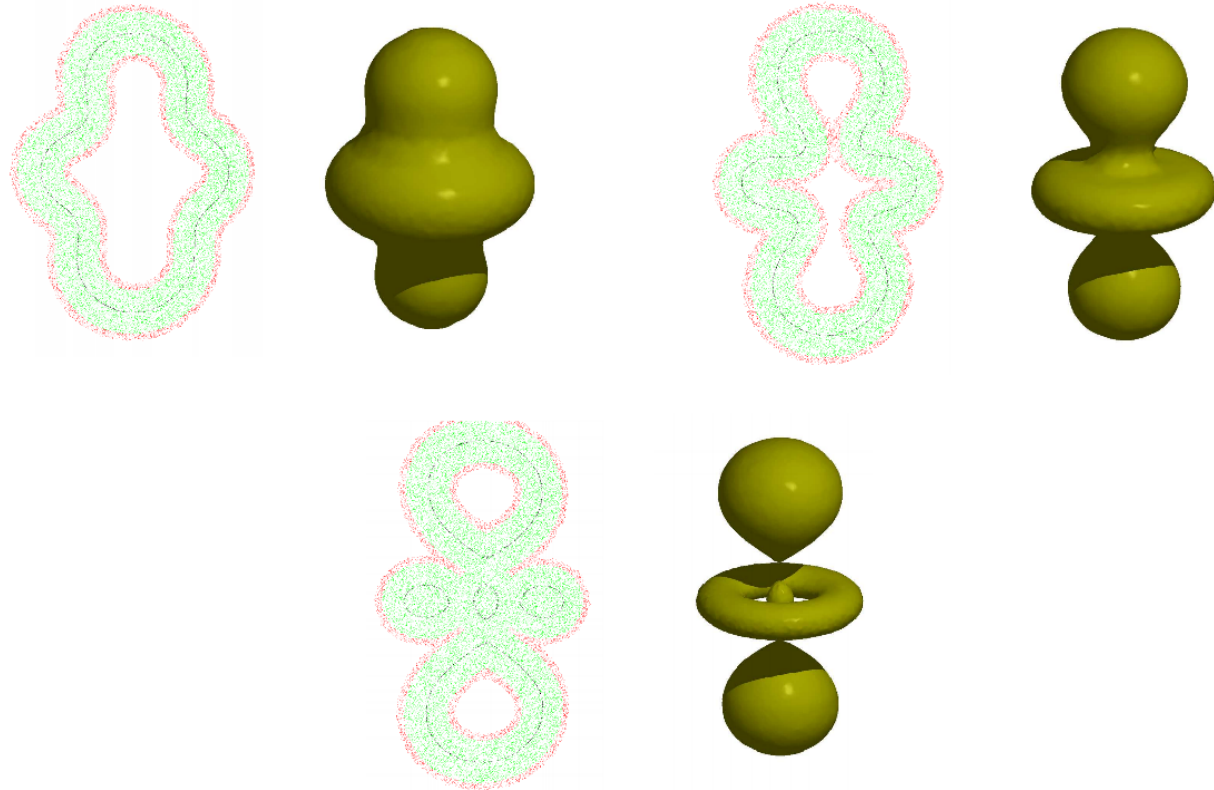
Level sets from medical data (CT, MRI etc.)

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



Level set storage

- Storage of 2D surface is now $O(n^3)$
- Can save space by only storing a narrow band around the surface.



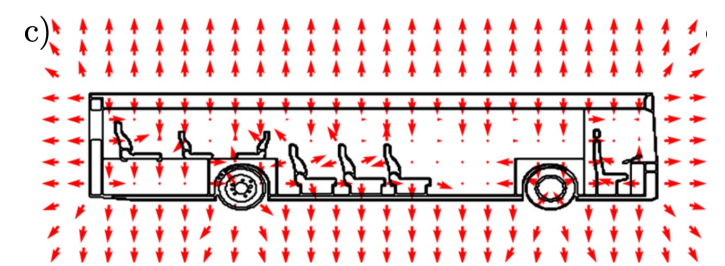
Distance Field – normals and closest points

- **Distance to surface** is given by the distance field itself $f(\mathbf{p})$

$$f(\mathbf{p}) = \min_{\mathbf{q} \in \mathcal{S}} \|\mathbf{p} - \mathbf{q}\| \quad \mathcal{S} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) = 0\}$$

- **Surface normal** are the gradients of the function

$$n(\mathbf{p}) = \nabla f(\mathbf{p})$$



- **Closest points** are found trivially as:

$$\mathbf{q} = \mathbf{p} - f(\mathbf{p}) \nabla_{\mathbf{p}} f(\mathbf{p})$$

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)

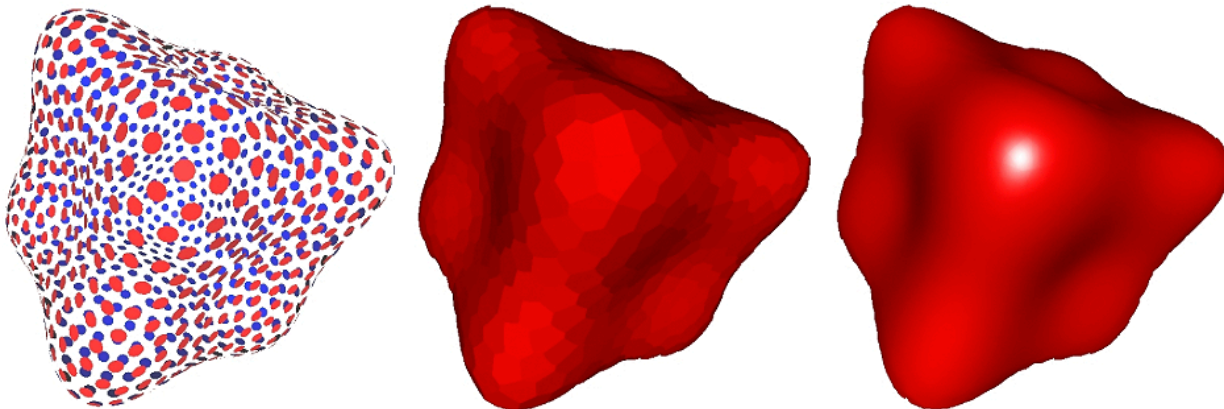
Cons:

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

What about explicit
representations?

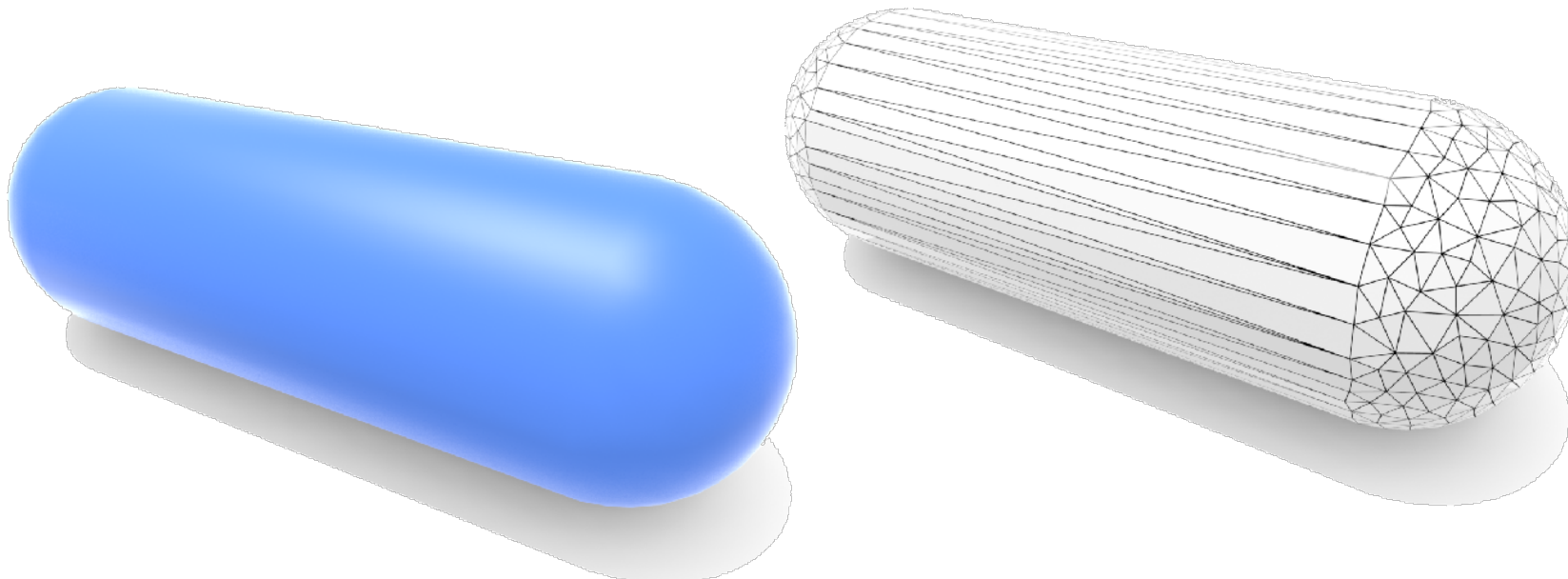
Point Cloud (Explicit)

- Easiest representation: list of points (x,y,z)
- Often augmented with normals
- Easily represent any kind of geometry
- Easy to draw dense cloud ($\gg 1$ point/pixel)
- Hard to interpolate under-sampled regions
- Hard to do processing / simulation/...



Polygon Mesh (Explicit)

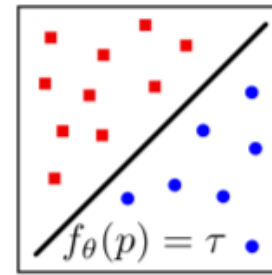
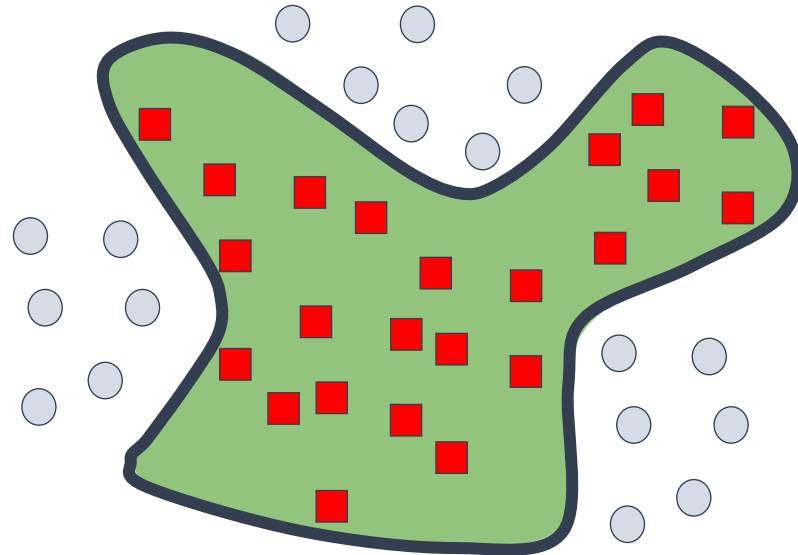
- Store vertices and polygons (most often triangles or quads)
- Easier to do processing/simulation, adaptive sampling
- More complicated data structures
- Irregular neighborhoods



Surfaces as an Implicit Function

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3$$

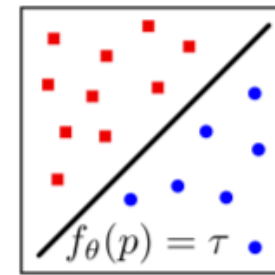
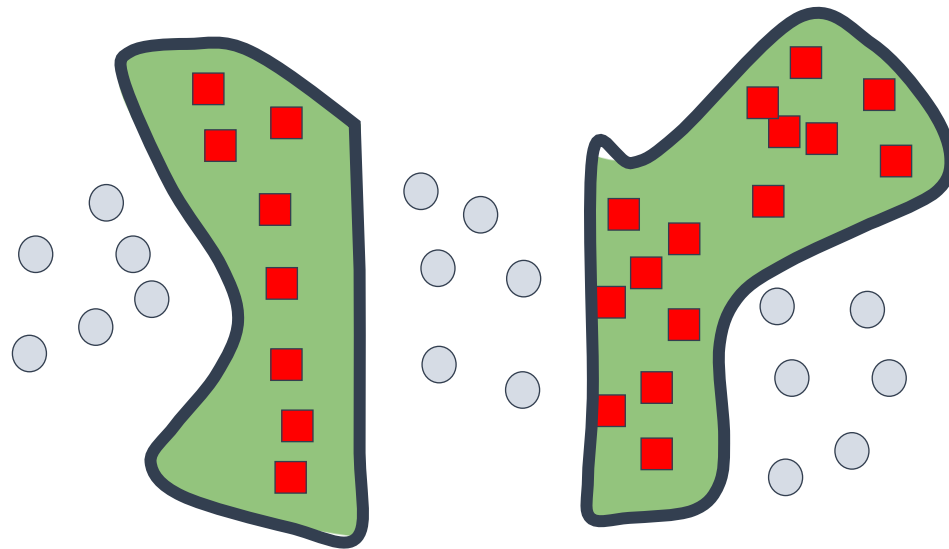
$$f(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in \text{outside } \circ \\ 1, & \text{if } \mathbf{p} \in \text{inside } \blacksquare \end{cases}$$



$$\mathcal{S} = \{\mathbf{p}, f(\mathbf{p}) = \tau\}$$

Surfaces as an Implicit Function

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3 \quad f(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in \text{outside } \circ \\ 1, & \text{if } \mathbf{p} \in \text{inside } \blacksquare \end{cases}$$



$$\mathcal{S} = \{\mathbf{p}, f(\mathbf{p}) = \tau\}$$



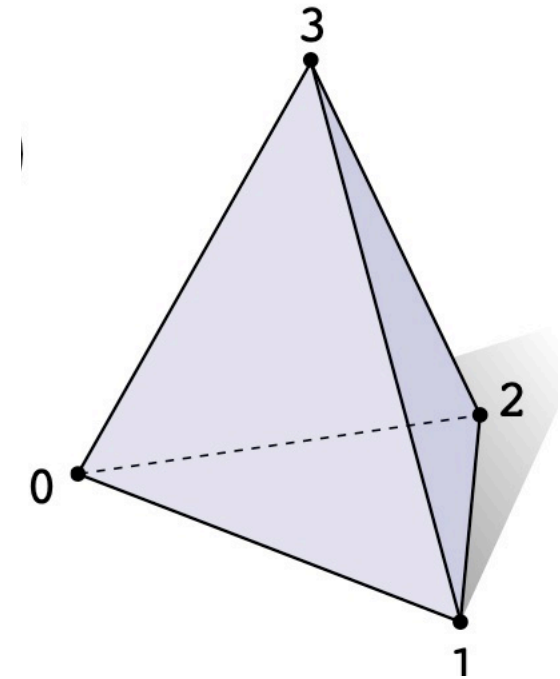
With **implicit functions** Topology changes only require changing
Mesh based representations would struggle

$f(\mathbf{p})$

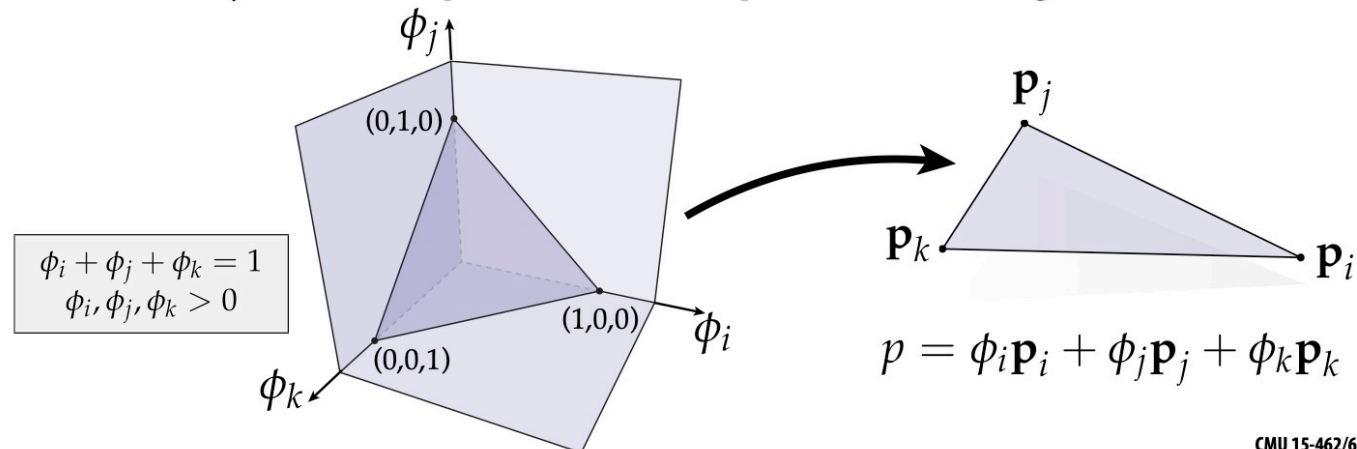
Triangle Mesh (Explicit)

- 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



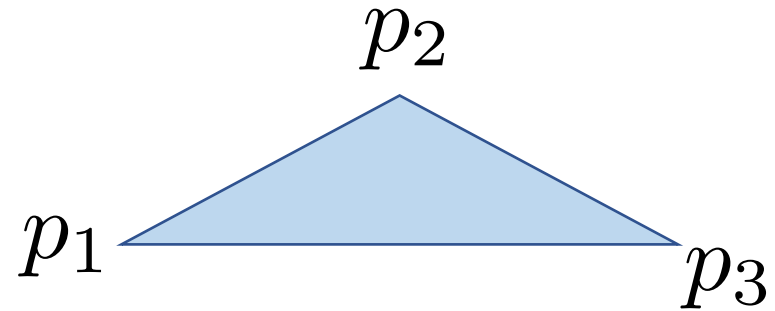
- Use barycentric interpolation to define points inside triangles:



Triangle Mesh – Normals (Explicit)

- Option1: Normals per face (Phong)

$$n = (p_2 - p_1) \times (p_3 - p_1)$$



- Option2: Interpolate incident face normal at each vertex, and interpolate with barycentric interpolation for points inside the triangle

$$n(\phi_1, \phi_2, \phi_3) = \phi_1 n_1 + \phi_2 n_2 + \phi_3 n_3$$

Distance queries and closest points

- Smooth parametric surface

- Evaluation $f(u, v)$

- Normals $\mathbf{n}(u, v) = f_u(u, v) \times f_v(u, v)$

- Distance and closest point. Find surface point $f(u, v)$ such that:

$$[\mathbf{q} - f(u, v)] \times \mathbf{n}(u, v) = \mathbf{0}$$

- Mesh

- Distance and closest point:

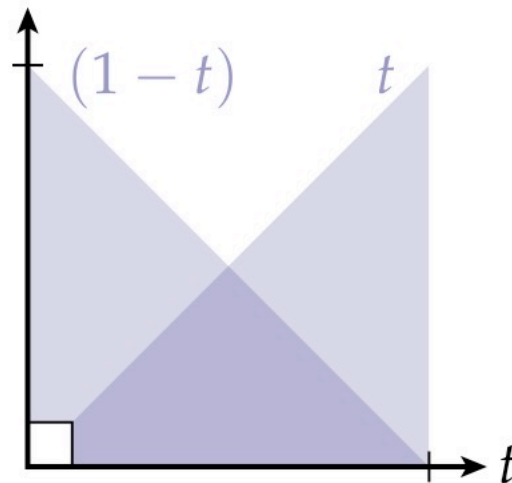
- Find closest triangle and then find closest point in triangle, and then finding point in triangle
 - Use Kd-tree, AAB Tree, to find closest triangles

Recall: Linear Interpolation (1D)

- Interpolate values using linear interpolation; in 1D:

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

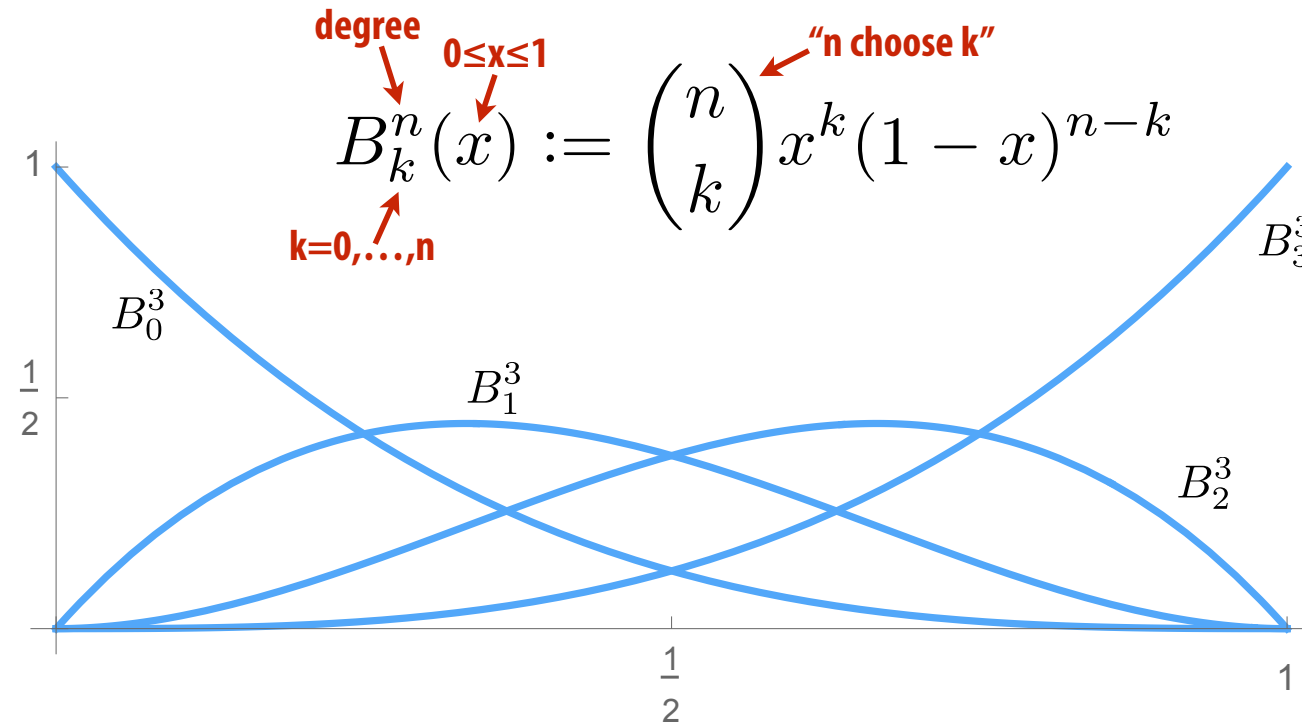
- Can think of this as a linear combination of two functions:



- Why limit ourselves to linear 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 $(1, x, x^2, x^3, \dots)$, use Bernstein basis:



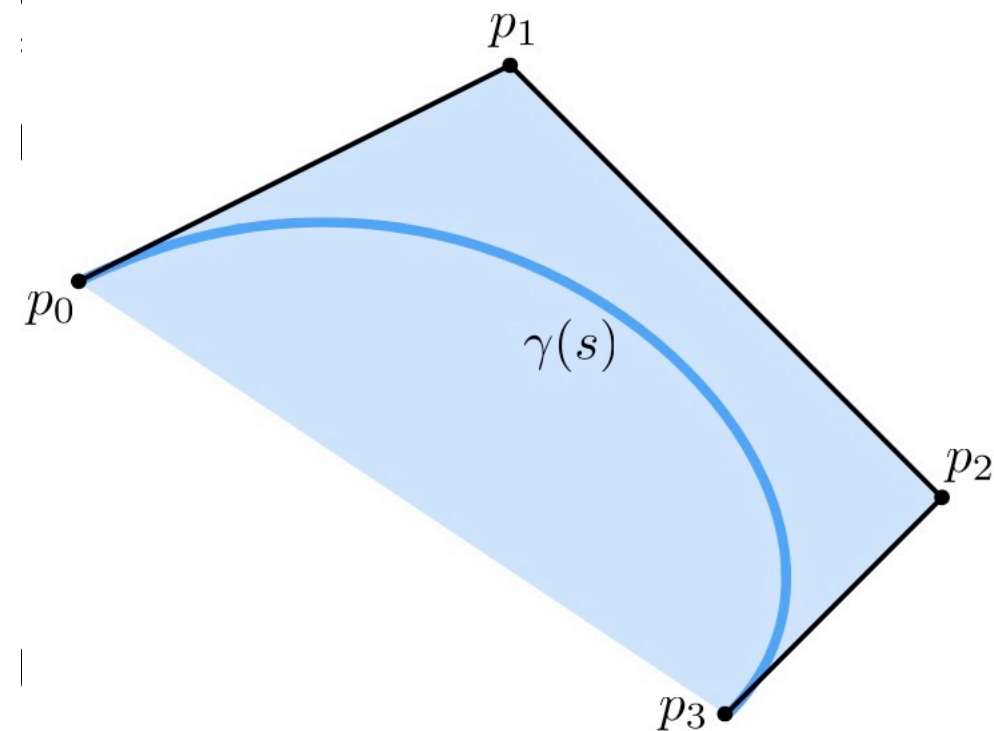
Bézier Curves (Explicit)

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

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

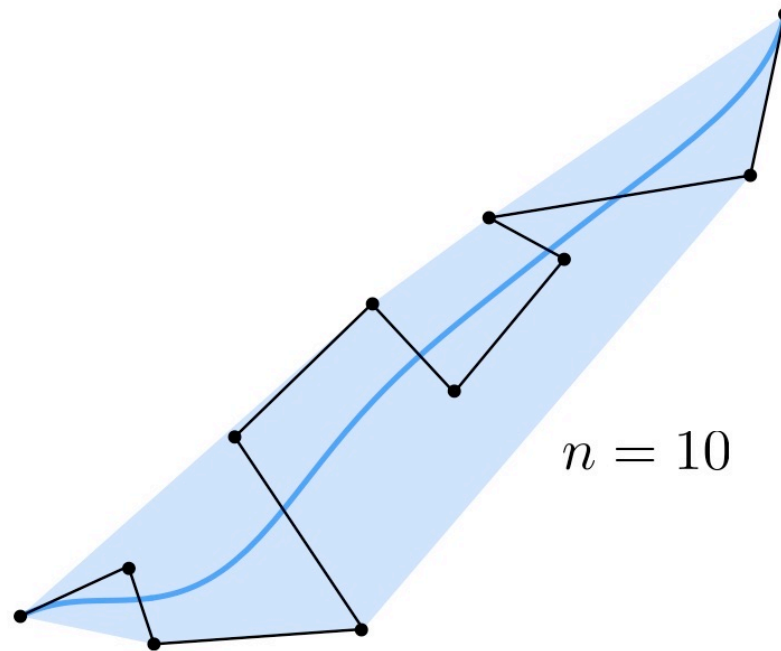
← control points

- For $n=1$, just get a line segment!
- For $n=3$, get "cubic Bézier":
- Important features:
 - interpolates endpoints
 - tangent to end segments
 - 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.)



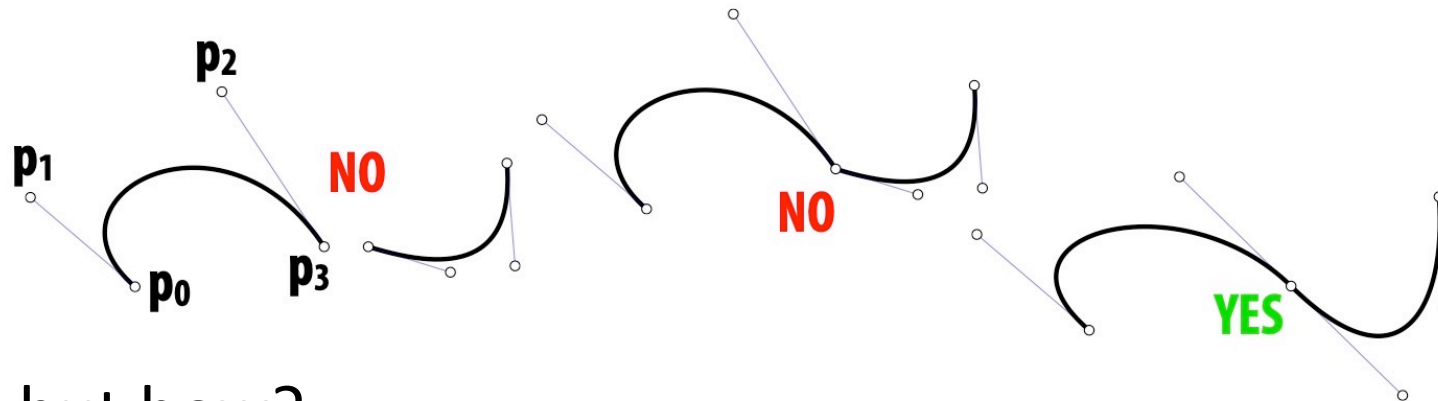
- Formerly, piecewise Bézier curve:

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

single Bézier \rightarrow γ_i

Bézier Curves - tangent continuity

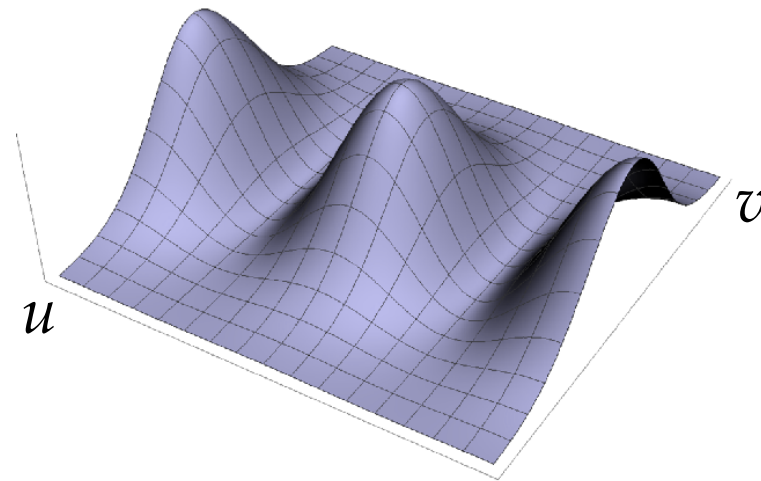
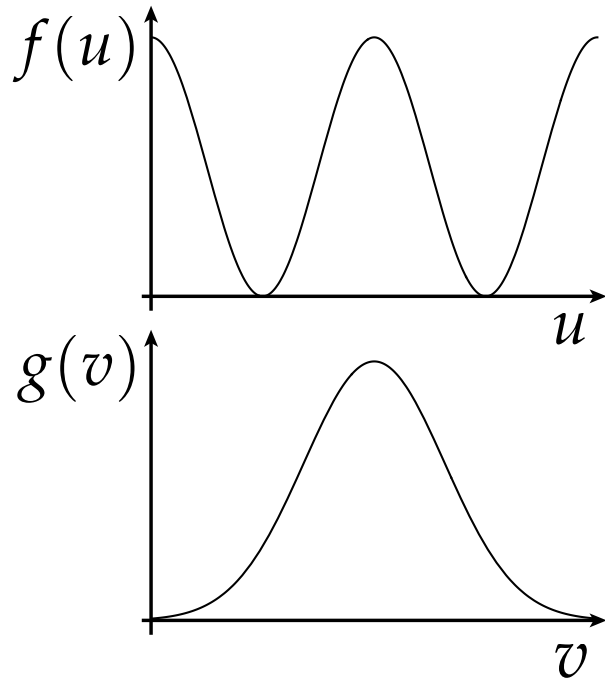
- To get "seamless" curves, need points and tangents to line up:



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

Tensor Product

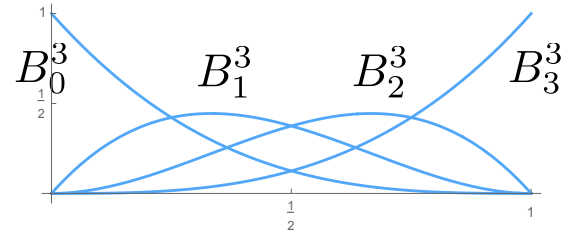
- Can use a pair of curves to get a surface.
- Value at any point (u, v) given by the product of the curve f at u and a curve g at v (tensor product).



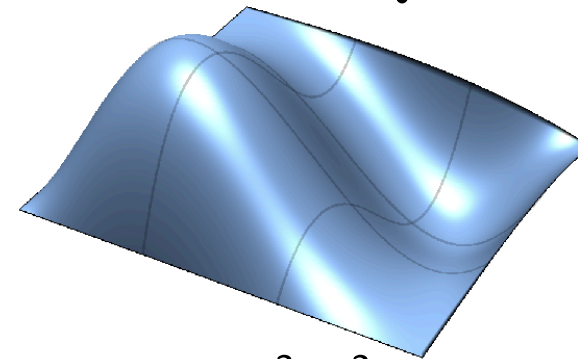
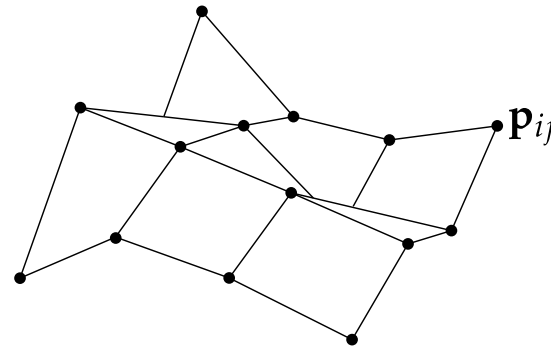
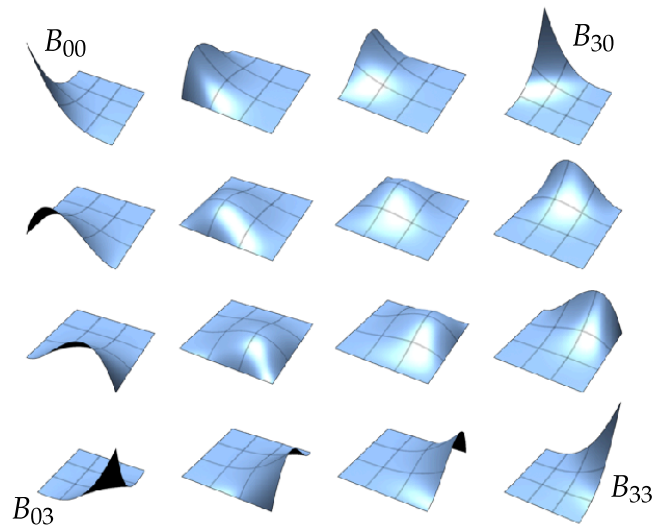
$$(f \otimes g)(u, v) := f(u)g(v)$$

Bézier Patches

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



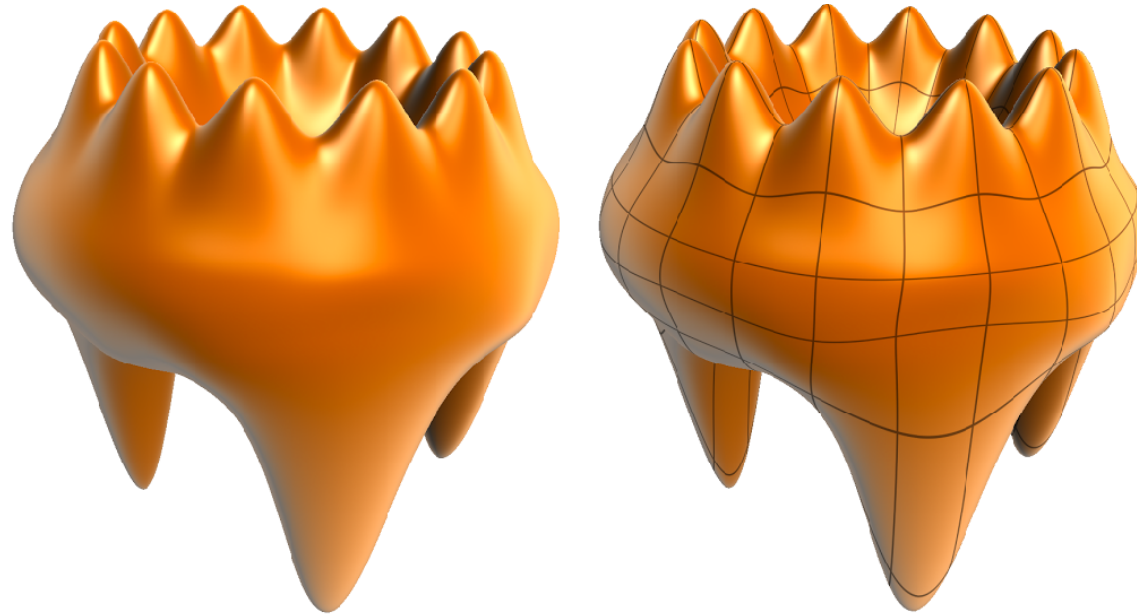
$$B_{i,j}^3(u, v) := B_i^3(u)B_j^3(v)$$



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

Bézier Surface

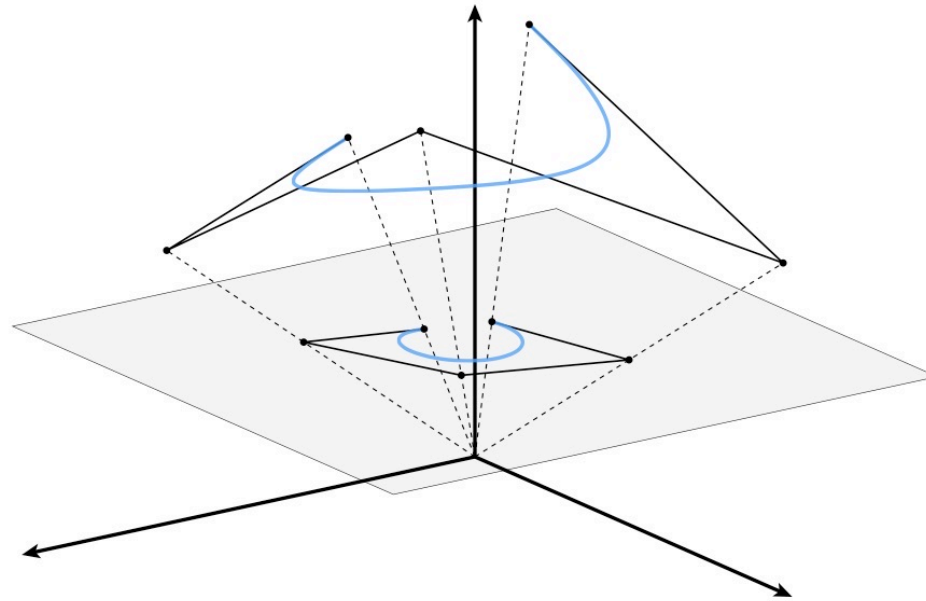
- Just as we connected Bézier curves, can connect Bézier patches to get a surface:



- 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?)

Rational B-Splines (Explicit)

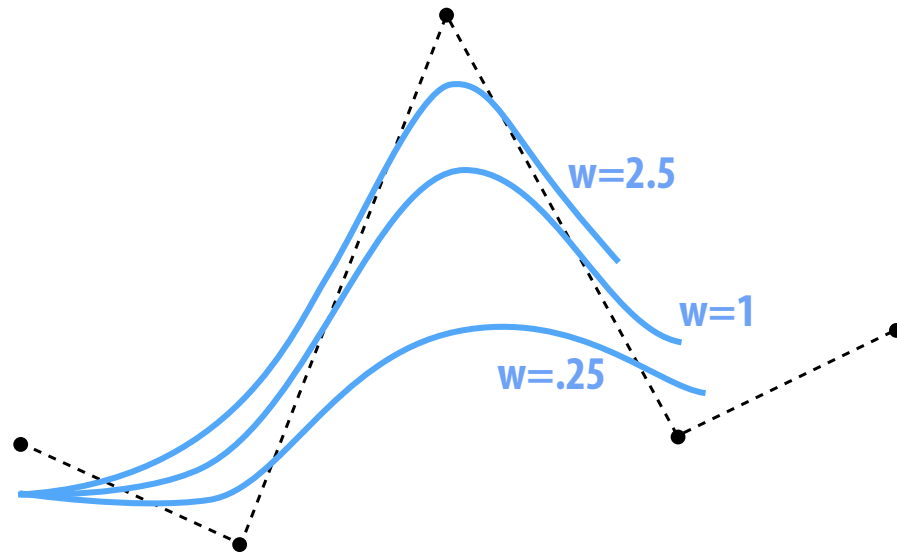
- Bézier can't exactly represent conics-not even the circle!
- Solution: interpolate in homogeneous coordinates, then project back to the plane:



Result is called a *rational* B-spline.

NURBS Surface (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:

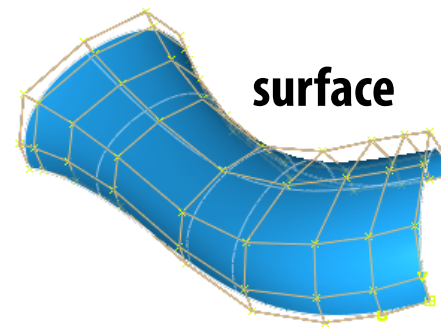
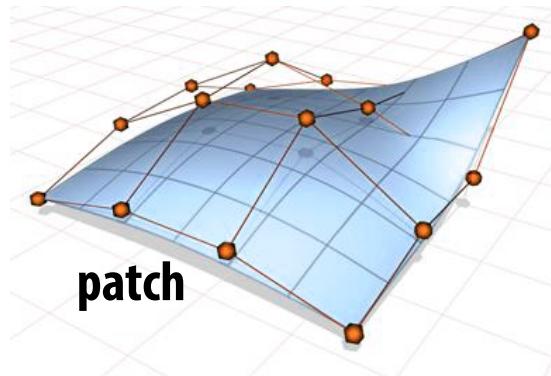


NURBS Surface (Explicit)

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

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

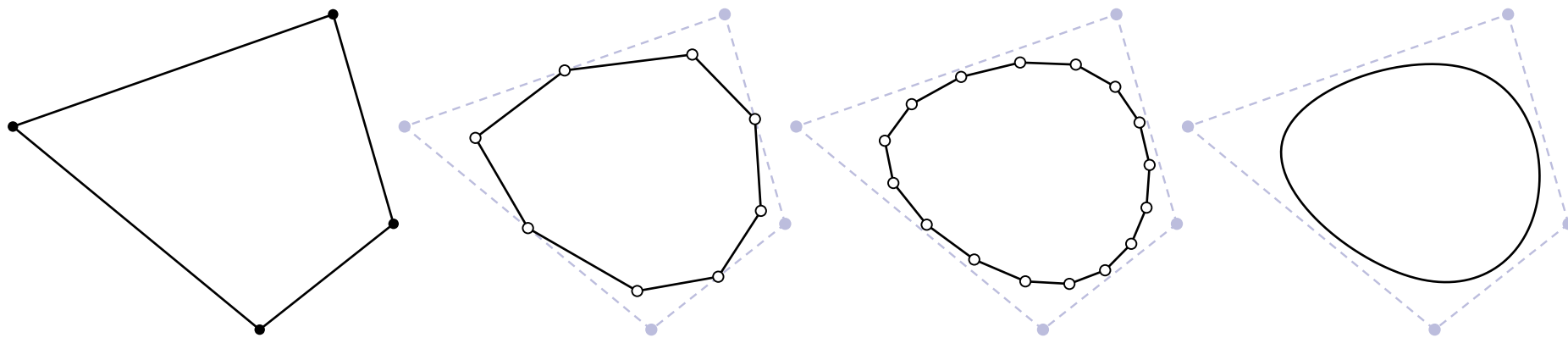
- Multiple NURBS patches form a surface



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

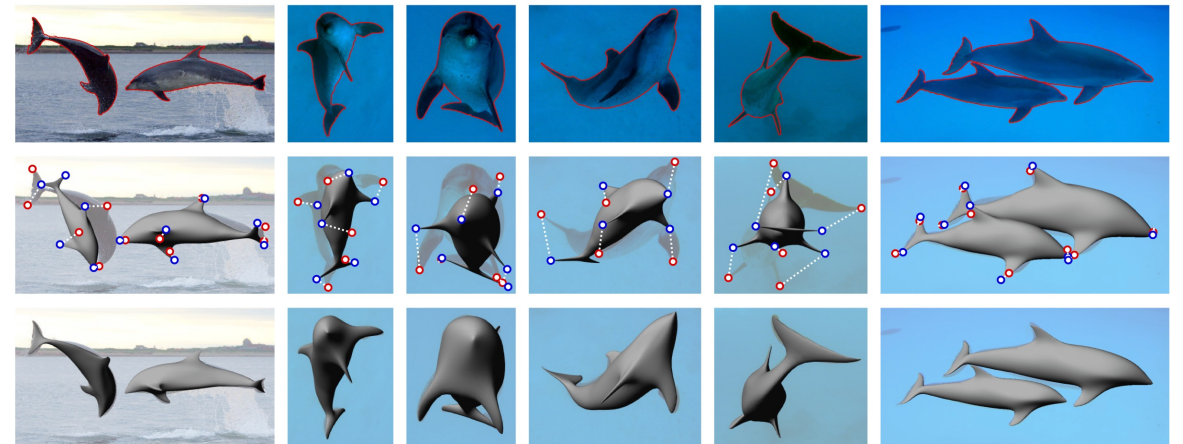
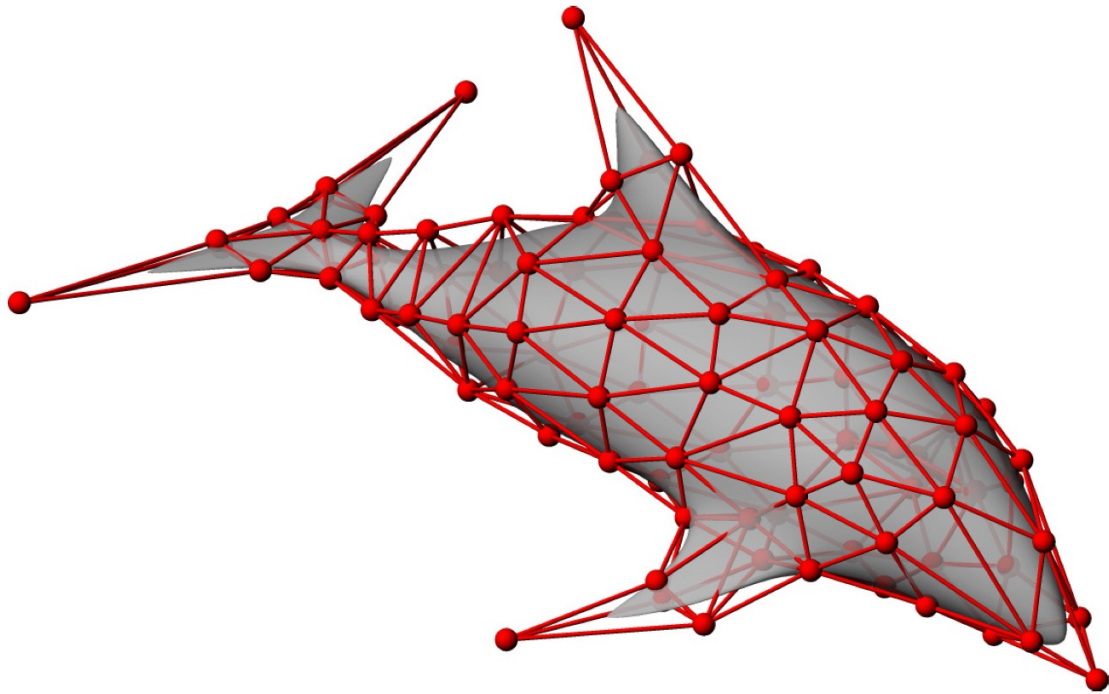
Subdivision

- 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!



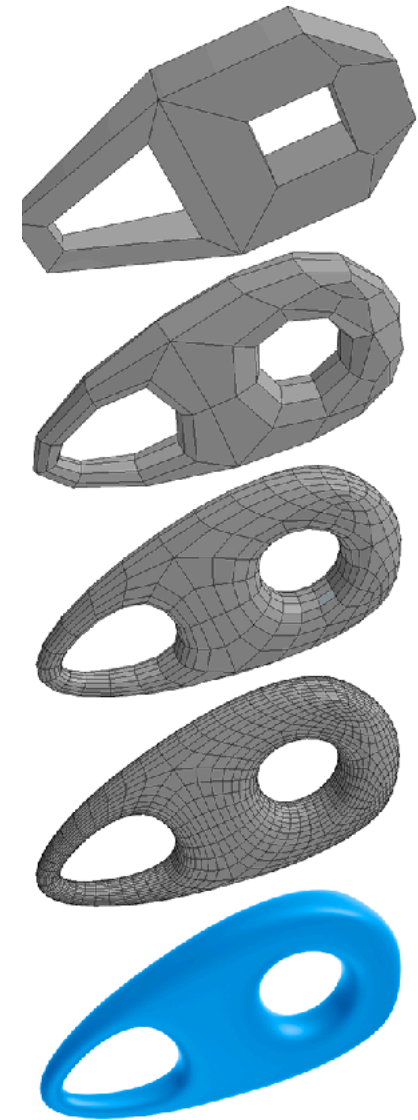
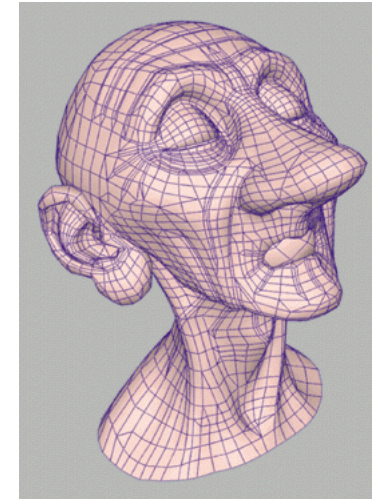
Q: Is subdivision an explicit or implicit representation?

Subdivision Surfaces in Computer Vision



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 Pixar (Pixar's "Geri's Game")



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

Slide credits and further reading

- Keenan Crane – Computer Graphics Lecture CMU 15-462/662.
Lecture 09: Introduction to Geometry.

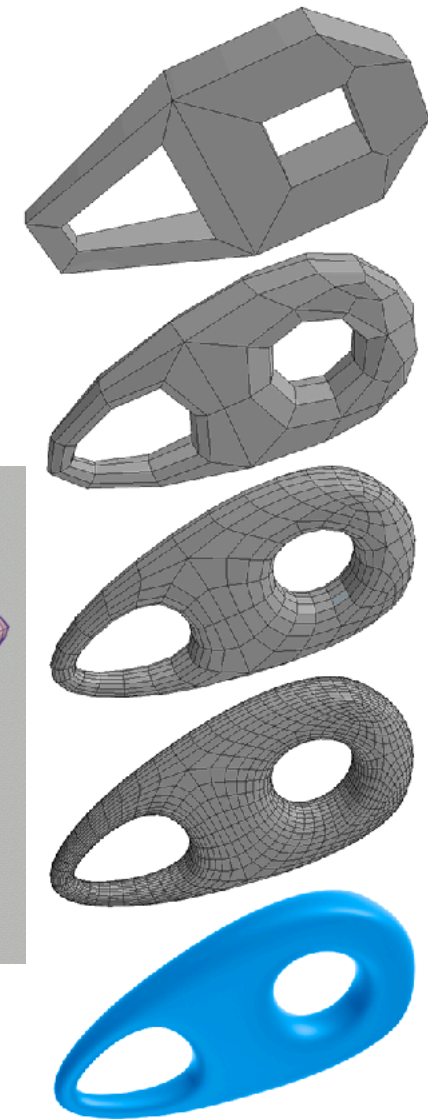
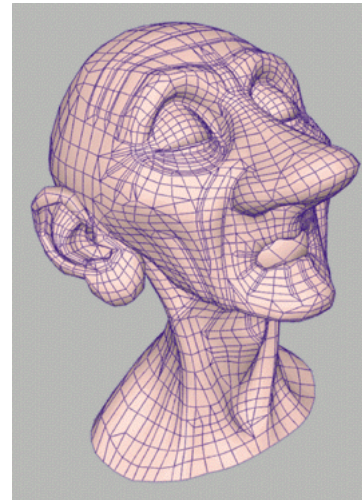
Subdivision in Action (Pixar's "Geri's Game")



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

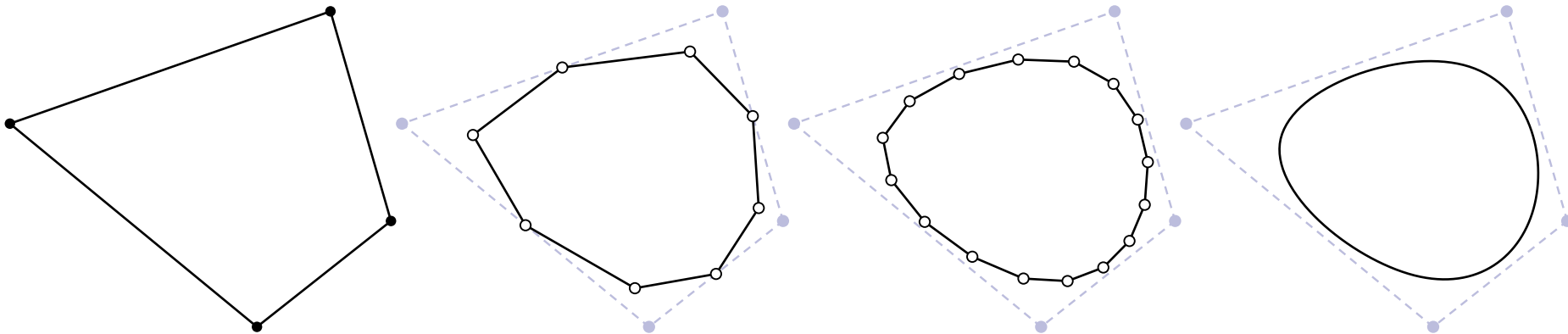
Subdivision Surfaces (Explicit)

- 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

- **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!*



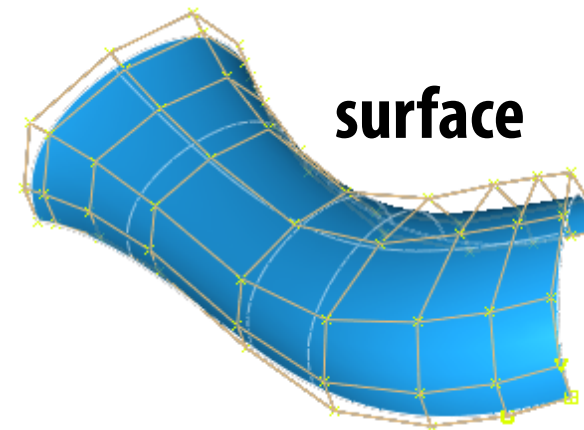
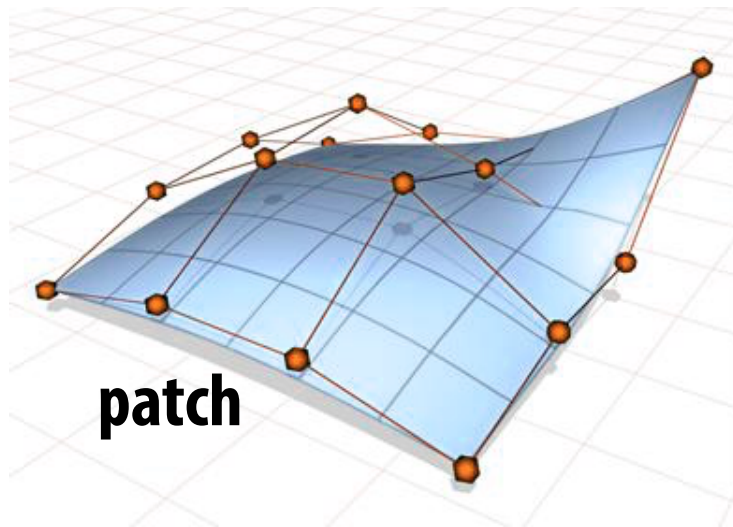
Q: Is subdivision an explicit or implicit representation?

NURBS Surface (Explicit)

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

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

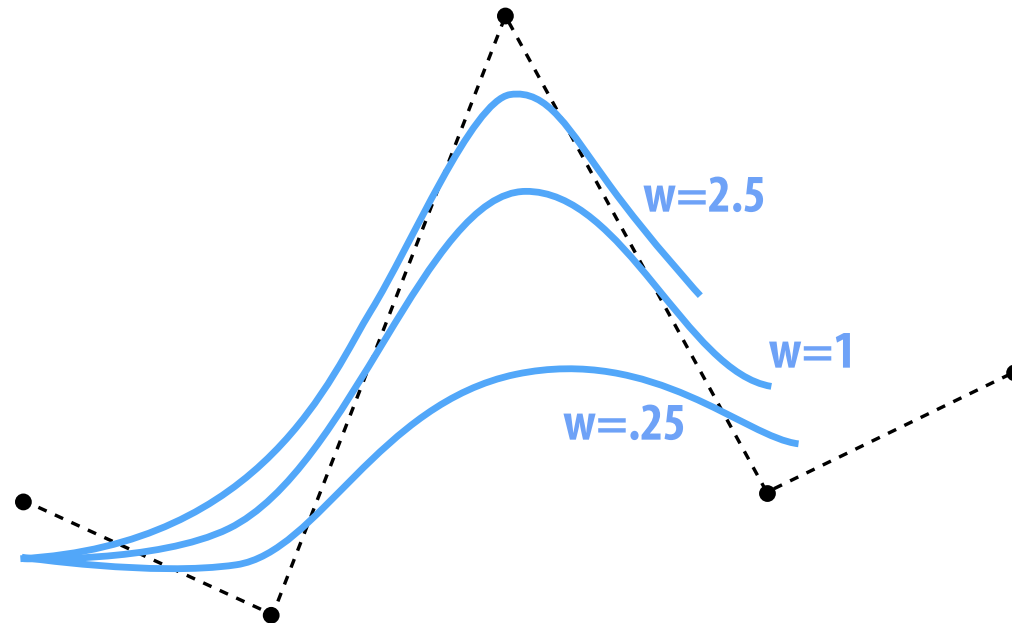
- Multiple NURBS patches form a surface



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

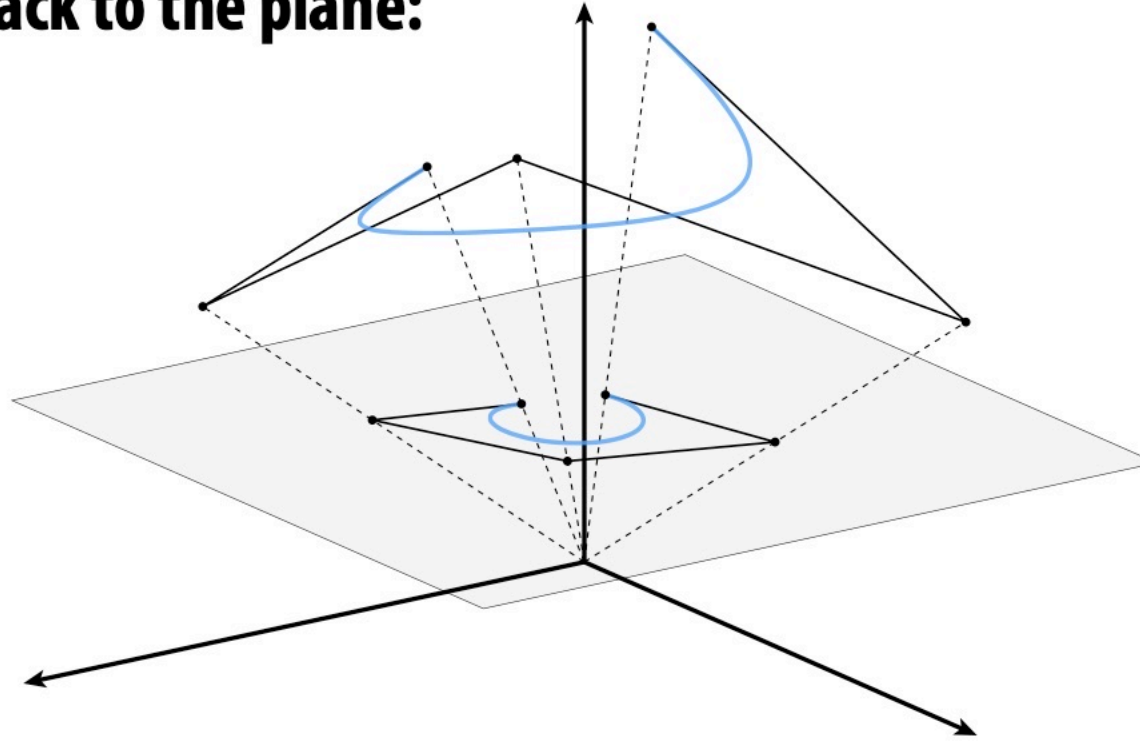
NURBS (Explicit)

- (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:



Rational B-Splines (Explicit)

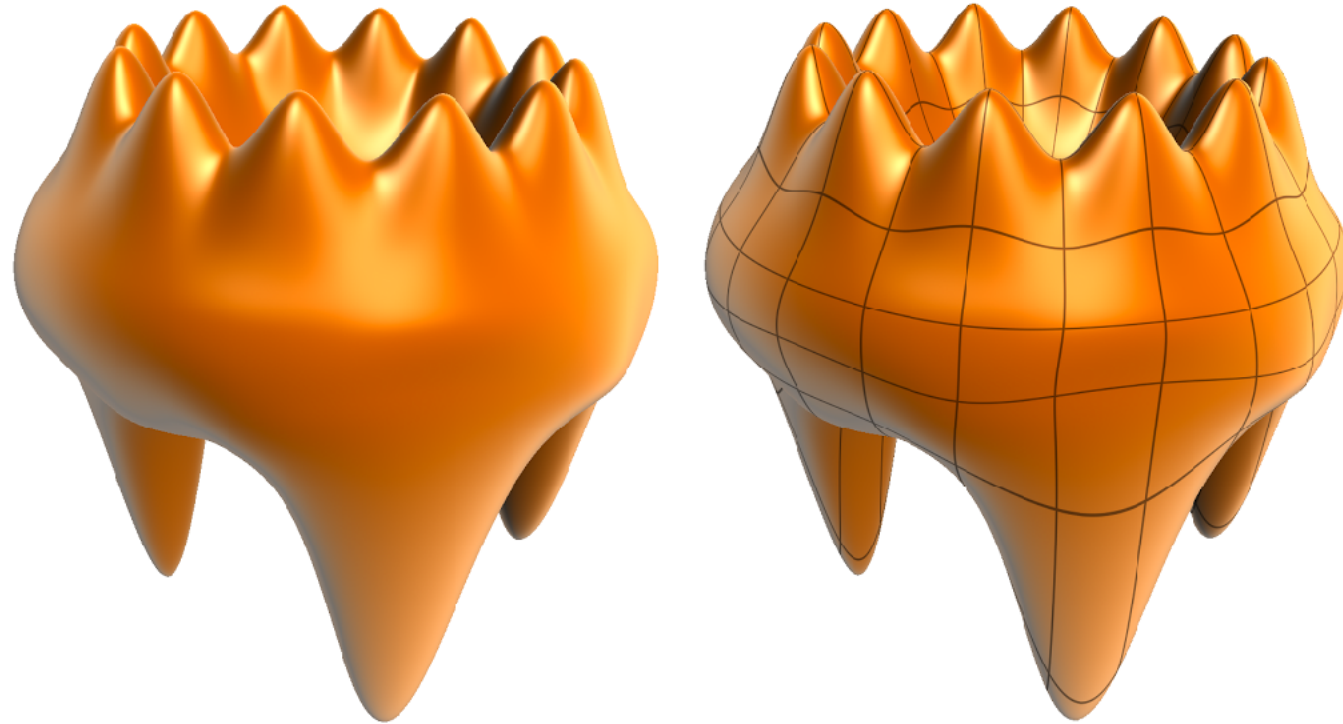
- Bézier can't exactly represent *conics*—not even the circle!
- Solution: interpolate in homogeneous coordinates, then project back to the plane:



Result is called a *rational B-spline*.

Bézier Surface

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



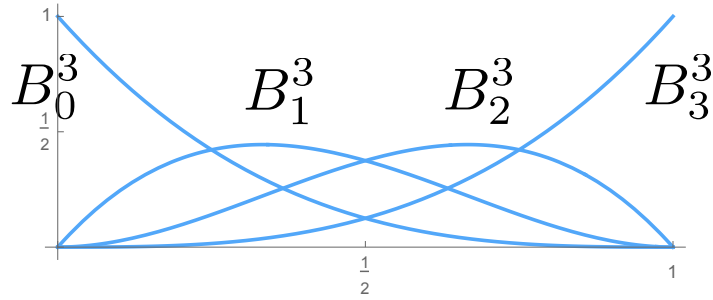
- *Very easy to draw: just dice each patch into regular (u,v) grid!*

Q: Can we always get tangent continuity?

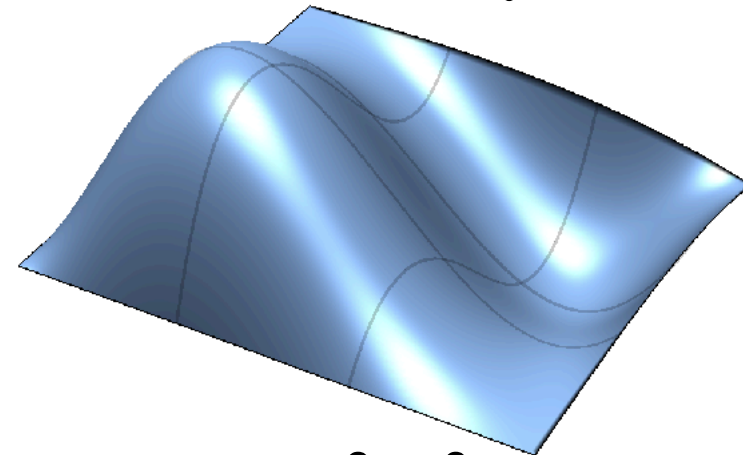
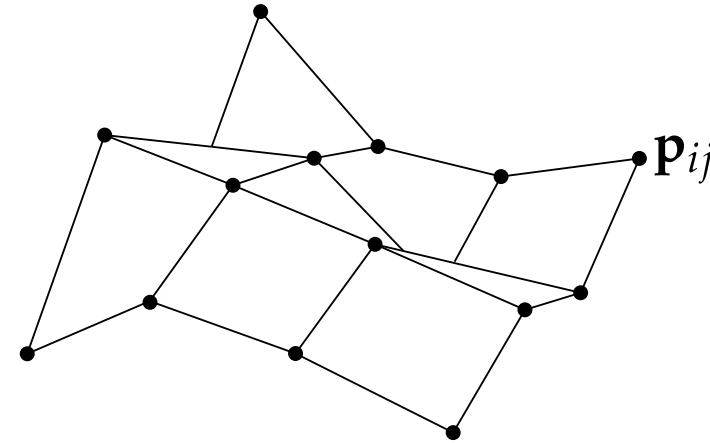
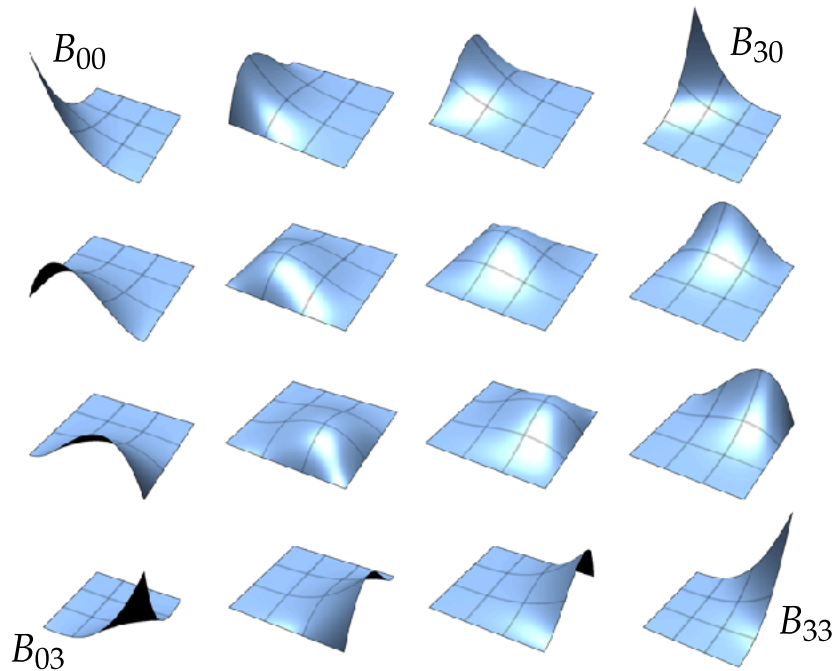
(Think: how many constraints? How many degrees of freedom?)

Bézier Patches

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



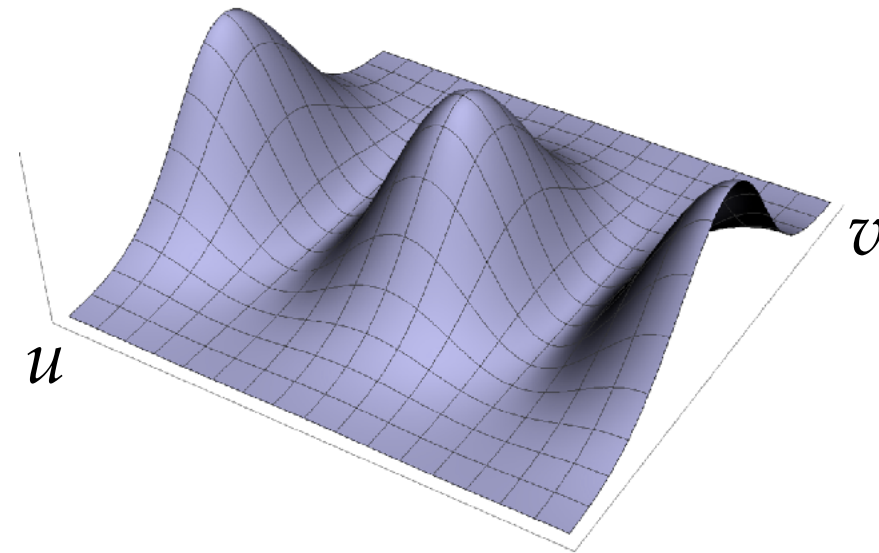
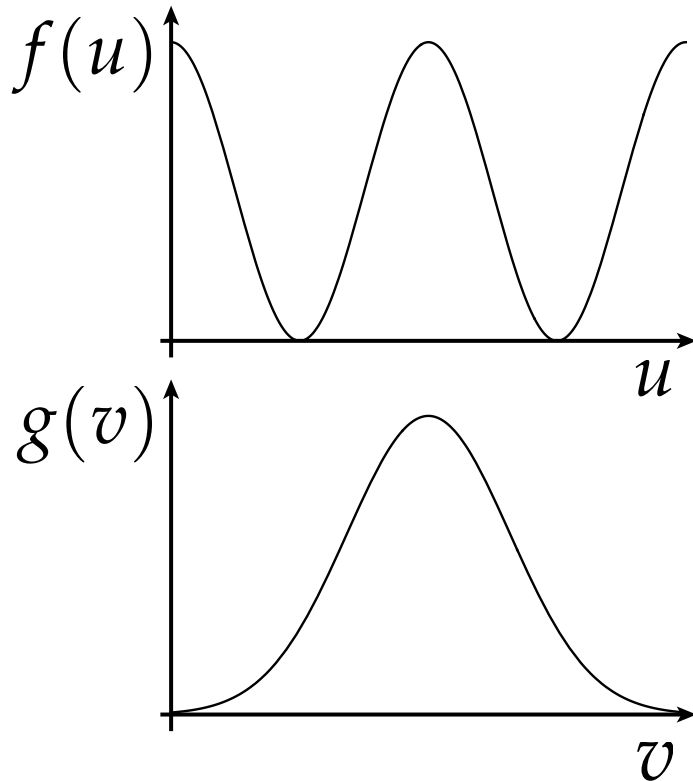
$$B_{i,j}^3(u, v) := B_i^3(u) B_j^3(v)$$



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

Tensor Product

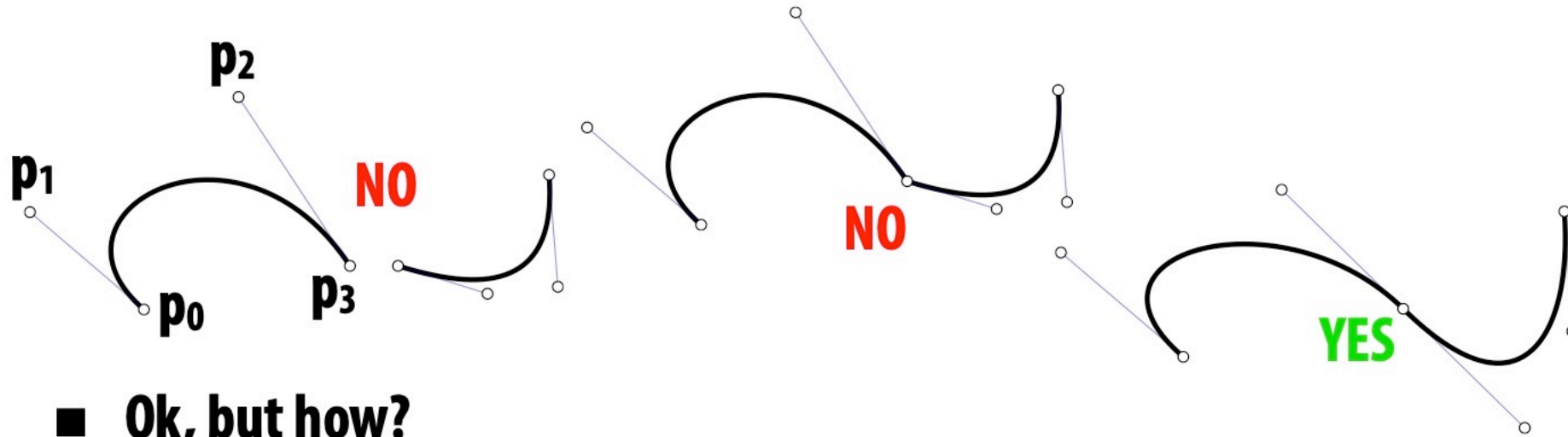
- Can use a pair of curves to get a surface
- Value at any point (u,v) given by product of a curve f at u and a curve g at v (sometimes called the “*tensor product*”):



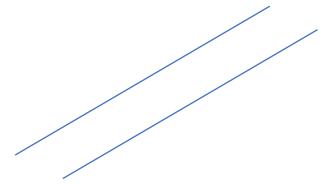
$$(f \otimes g)(u, v) := f(u)g(v)$$

Bézier Curves — tangent continuity

- To get “seamless” curves, need *points and tangents* to line up:

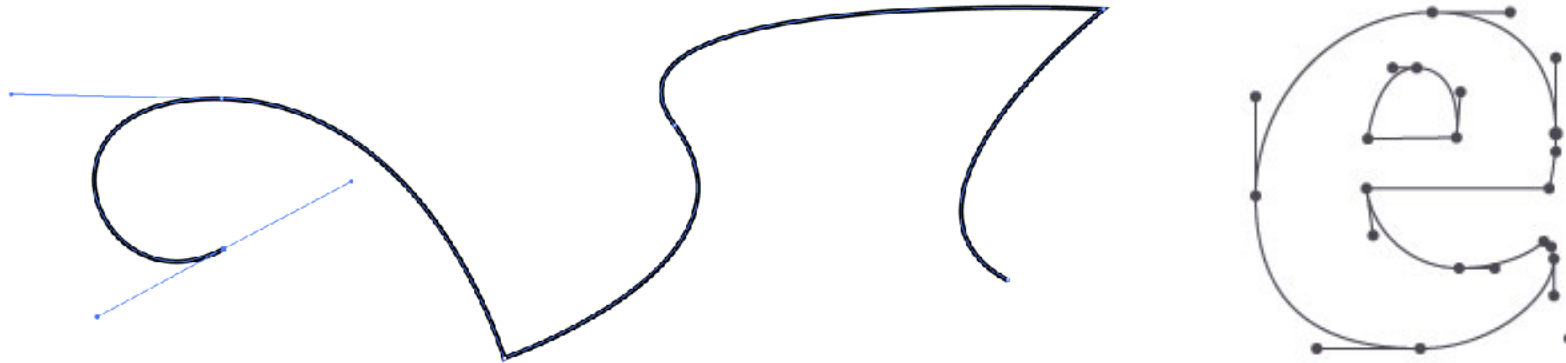


- Ok, but how?
- Each curve is cubic: $u^3p_0 + 3u^2(1-u)p_1 + 3u(1-u)^2p_2 + (1-u)^3p_3$
- Want endpoints of each segment to meet
- Want tangents at endpoints to meet
- Q: Could you do this with *quadratic* Bézier? *Linear* Bézier?



Piecewise Bézier Curves (Explicit)

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



- Formally, piecewise Bézier curve:

piecewise Bézier

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

single Bézier