

Computer Graphics

-Introduction to Geometry & its Representations

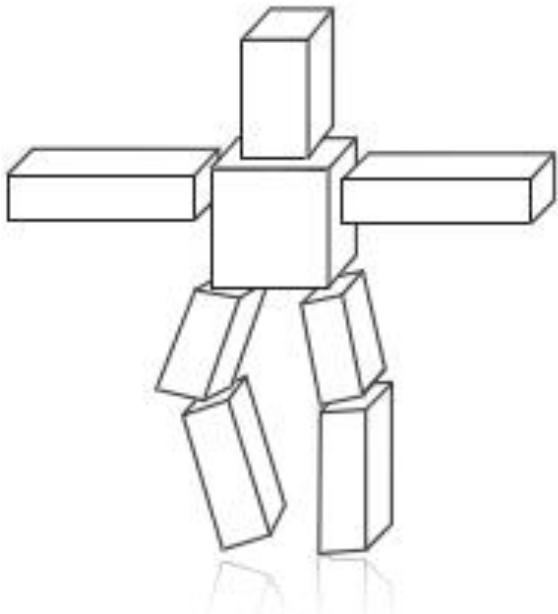
Junjie Cao @ DLUT

Spring 2016

<http://jjcao.github.io/ComputerGraphics/>

Increasing the complexity of our models

Transformations



Geometry



Materials, lighting, ...



What is geometry?

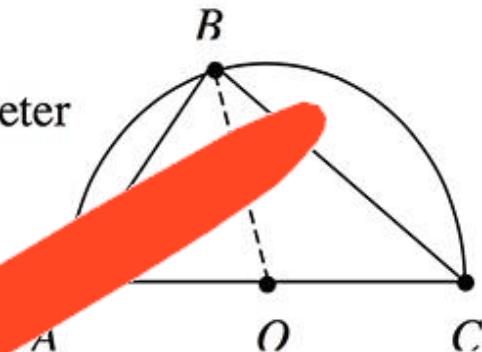
THEOREM 9.5. Let $\triangle ABC$ be inscribed in a semicircle with diameter \overline{AC} .

Then $\angle ABC$ is a right angle.

Proof:

Statement

1. Draw radius OB . Then $OB = OC = OA$ Given
2. $m\angle OBC = m\angle BCA$ Isosceles Triangle Theorem
 $m\angle OBA = m\angle BAC$
3. $m\angle ABC = m\angle OBA + m\angle BAC$ Angle Addition Postulate
4. $m\angle ABC + m\angle BCA + m\angle BAC = 180$ The sum of the angles of a triangle is 180
5. $m\angle ABC + m\angle BCA + m\angle OBA = 180$ Substitution (line 2)
6. $2m\angle ABC = 180$ Substitution (line 3)
7. $m\angle ABC = 90$ Division Property of Equality
8. $\angle ABC$ is a right angle Definition of Right Angle



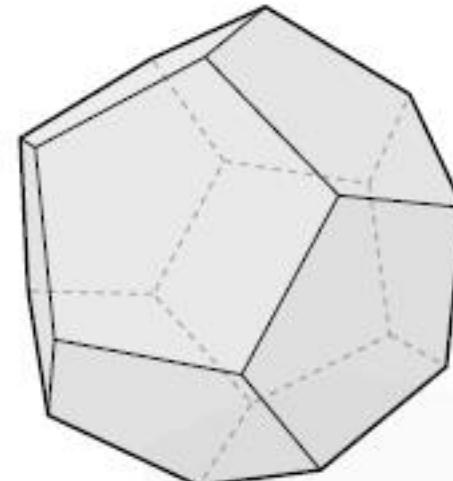
What is geometry?

"Earth" "measure"

ge•o•n•et•ry

/jē'ämətrē/ *n.*

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



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

How can we describe geometry?

IMPLICIT

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

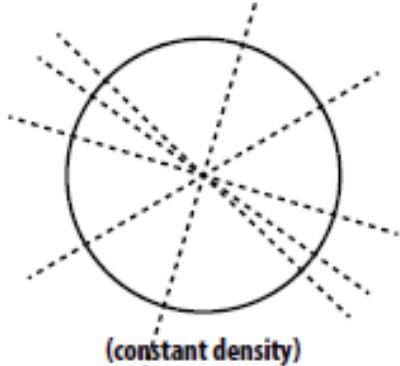
LINGUISTIC

“unit circle”

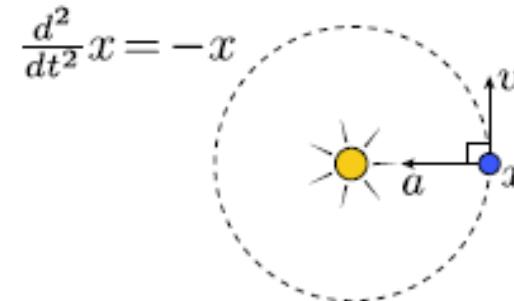
EXPLICIT

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

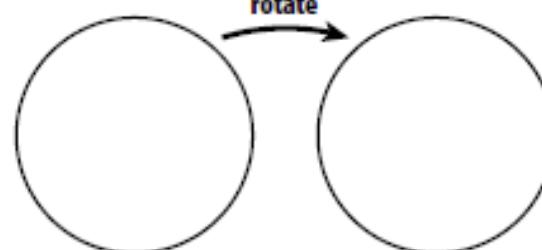
TOMOGRAPHIC



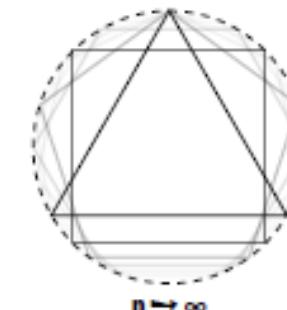
DYNAMIC



SYMMETRIC



DISCRETE



CURVATURE

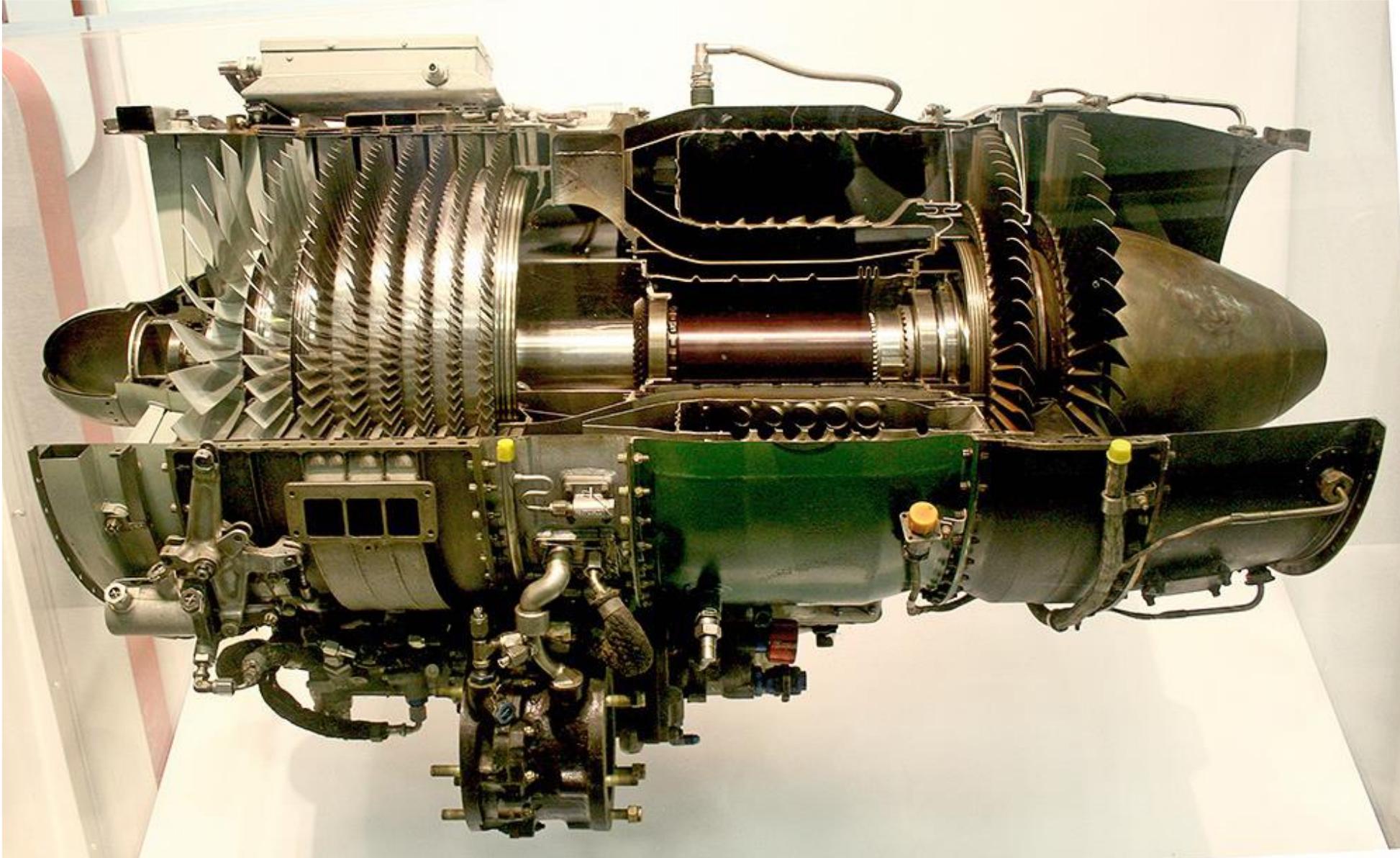
$$\kappa = 1$$

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



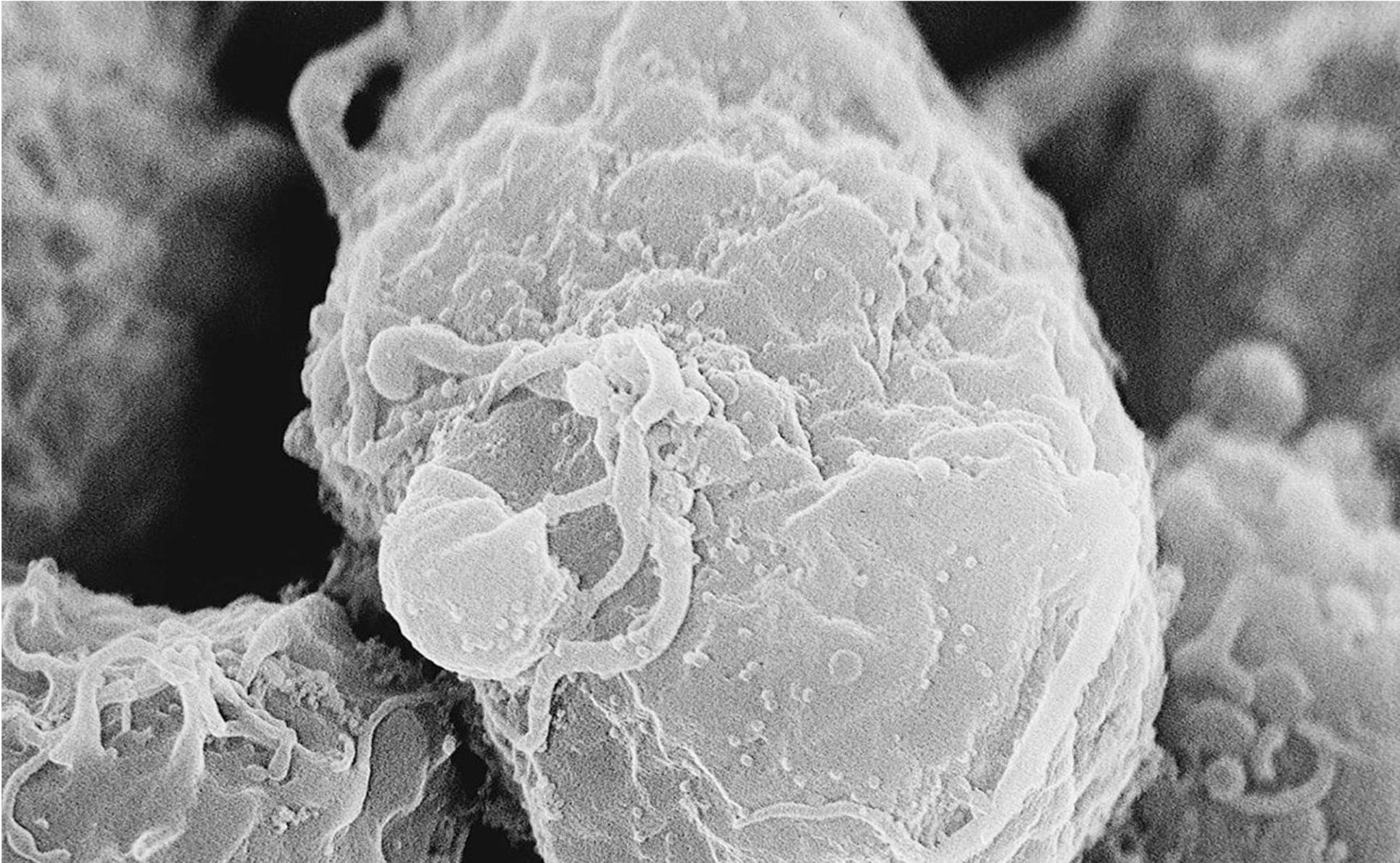
Examples of geometry



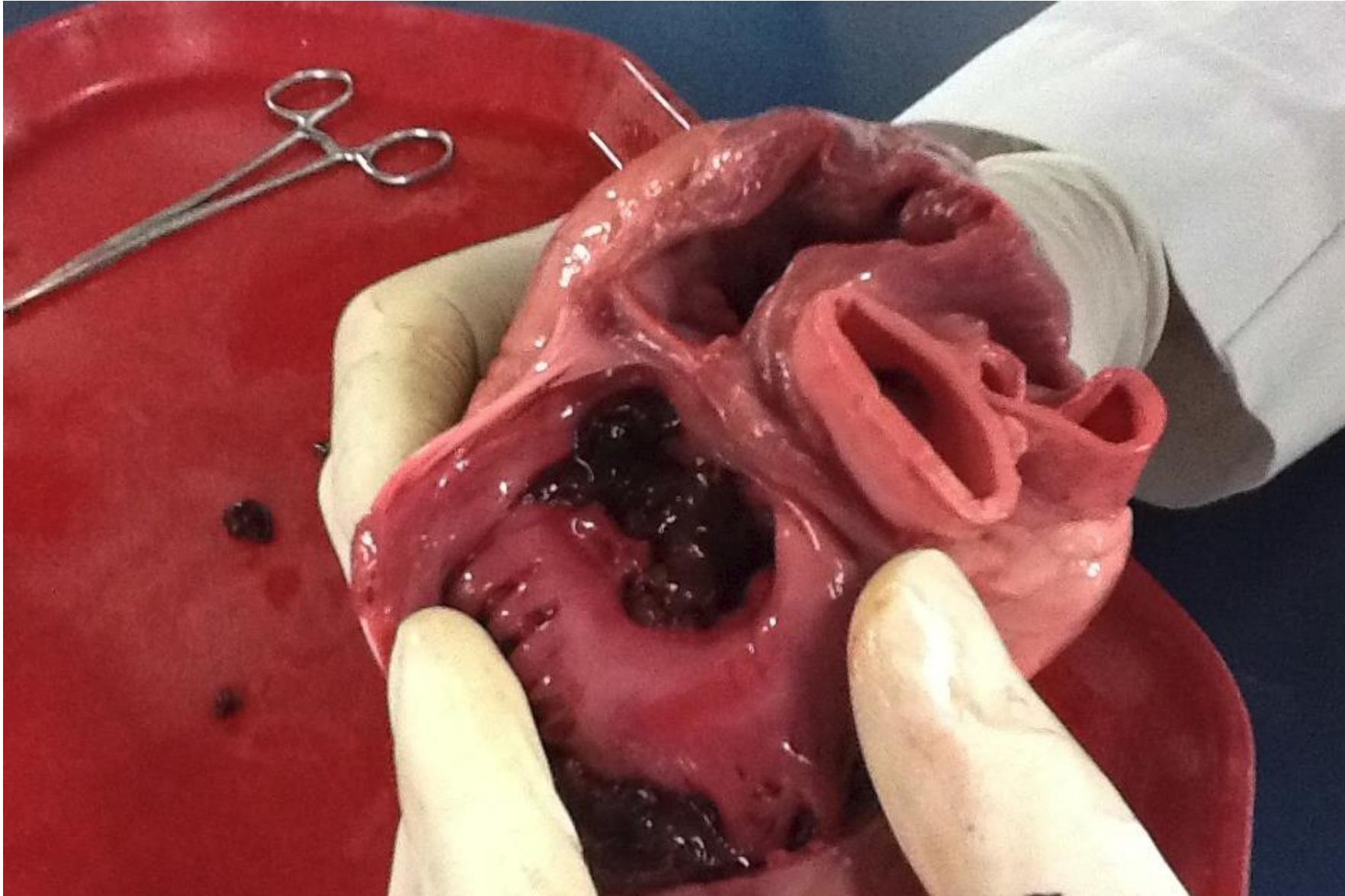
Examples of geometry



Examples of geometry



Examples of geometry



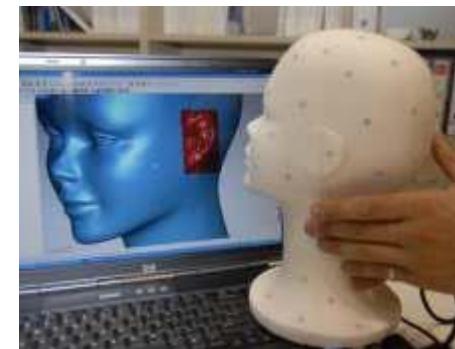
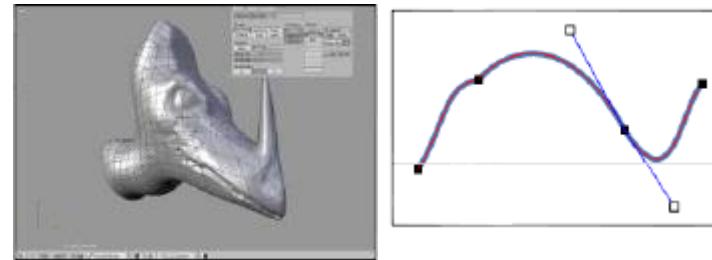
Examples of geometry



Shape representation

Where does the shape come from?

- Modeling “by hand”
 - Higher-level representations,
 - Amenable to modification, control
- Acquired real-world objects
 - Discrete sampling
 - Points, meshes



No one “best” choice—geometry is hard!

“I hate meshes.

I cannot believe how hard this is.

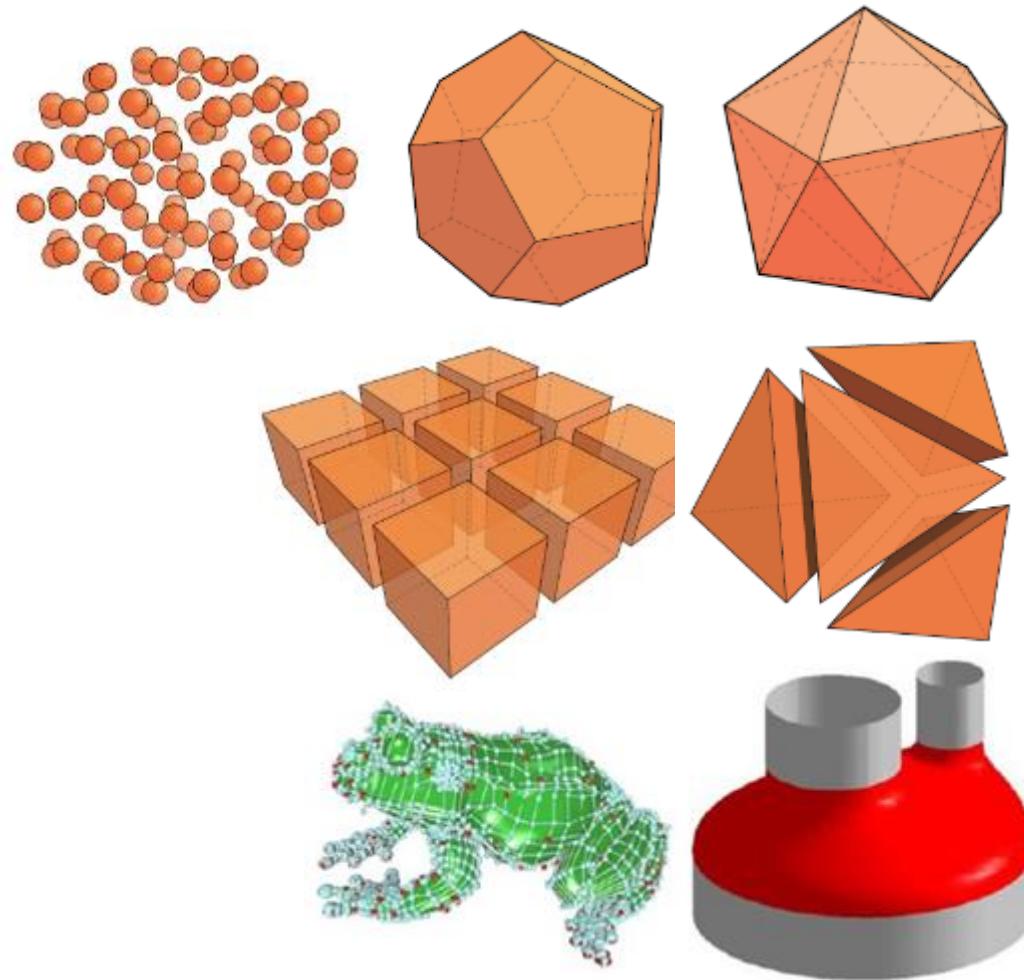
Geometry is hard.”

—David Baraff

Senior Research Scientist
Pixar Animation Studios

Many ways to digitally encode geometry

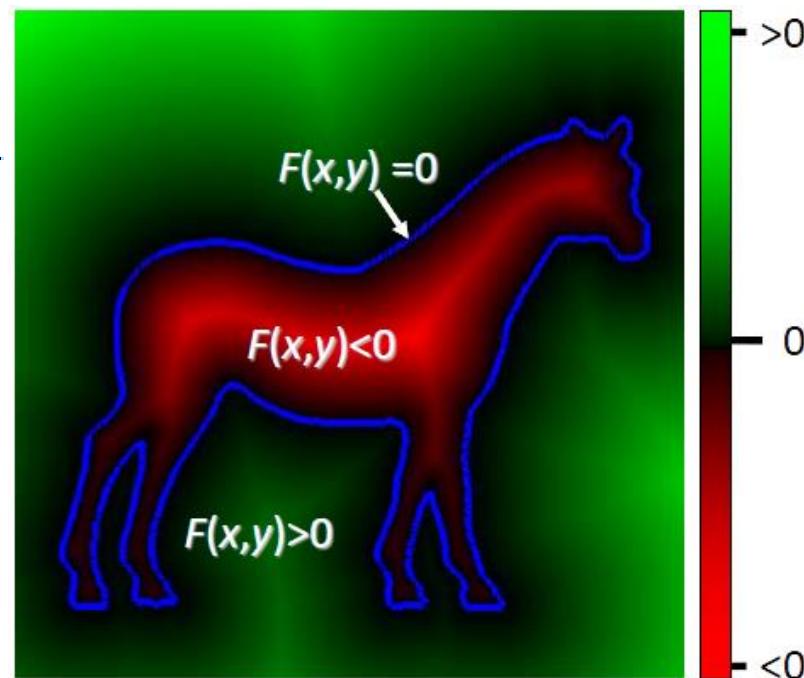
- **EXPLICIT**
- - point cloud
- - polygon mesh
- - subdivision, NURBS
- - L-systems
- - ...
- **IMPLICIT**
- - level set
- - algebraic surface
- - ...
- **Each choice best suited to a different task/type of geometry**



“Implicit” Representations of Geometry

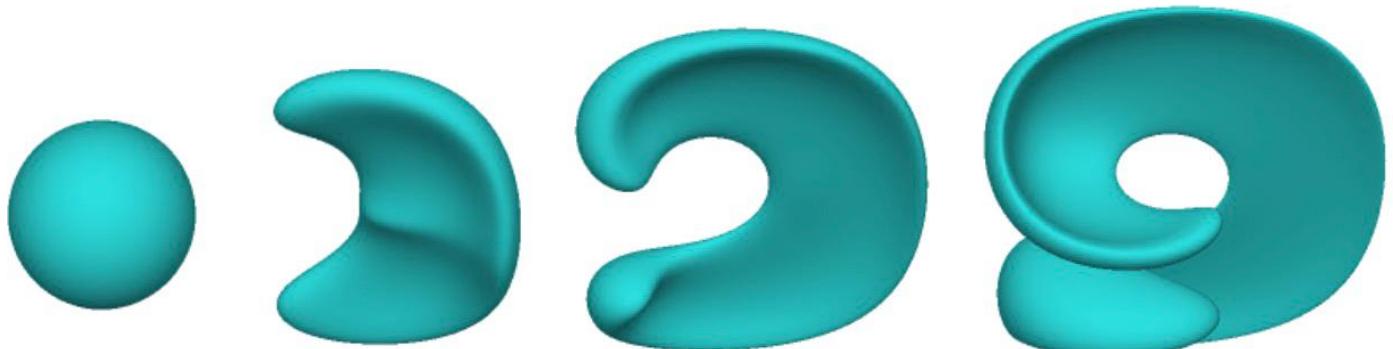
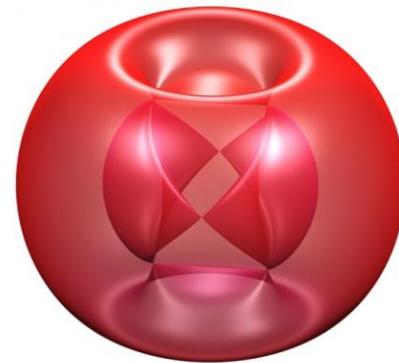
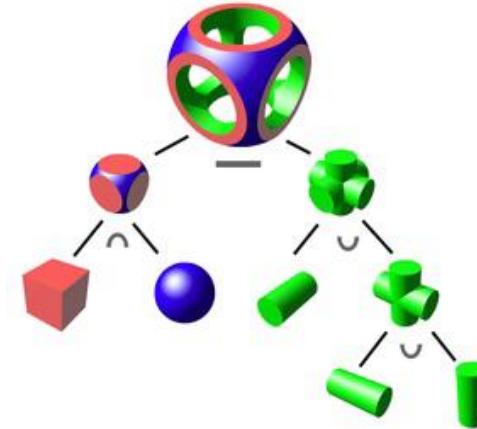
- Points aren't known directly, but satisfy some relationship
 - E.g., unit sphere is all points x such that $x^2+y^2+z^2=1$
 - More generally, $f(x,y,z) = 0$
- Represent a surface as the zero set of a (regular) function defined in R^3 .

$$K = g^{-1}(0) = \{\mathbf{p} \in \mathbb{R}^3 : g(\mathbf{p}) = 0\}$$



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:

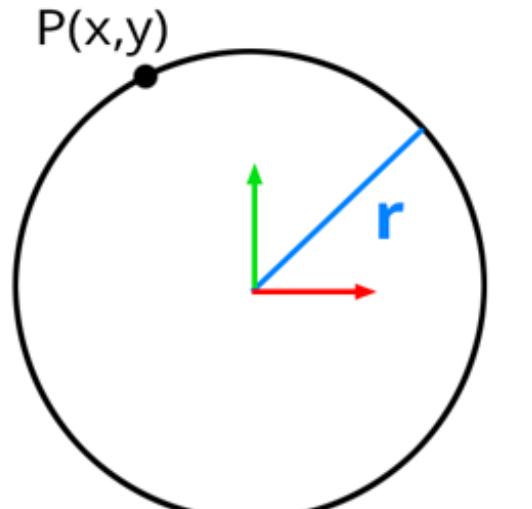
I'm thinking of an implicit surface $f(x,y,z)=0$.

Find *any* point on it.

Give up?

- My function was

$$g(x,y,z) = x^2 + y^2 + z^2 - r^2$$



© www.scratchapixel.com

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

Let's play another game.

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

I want to see if a point is inside it.

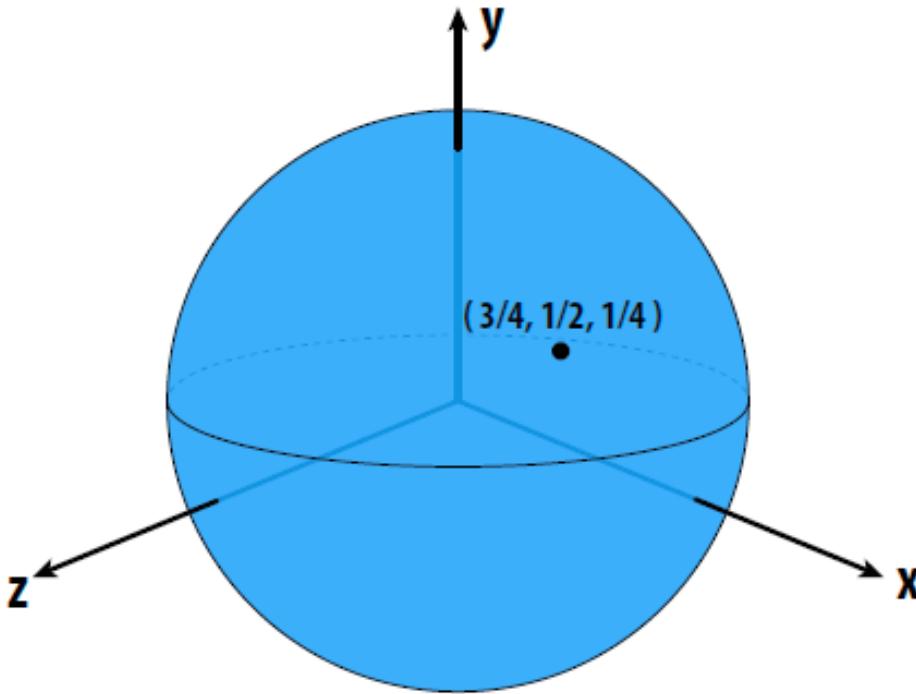
Check if this point is inside the unit sphere

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

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

$$7/8 < 1$$

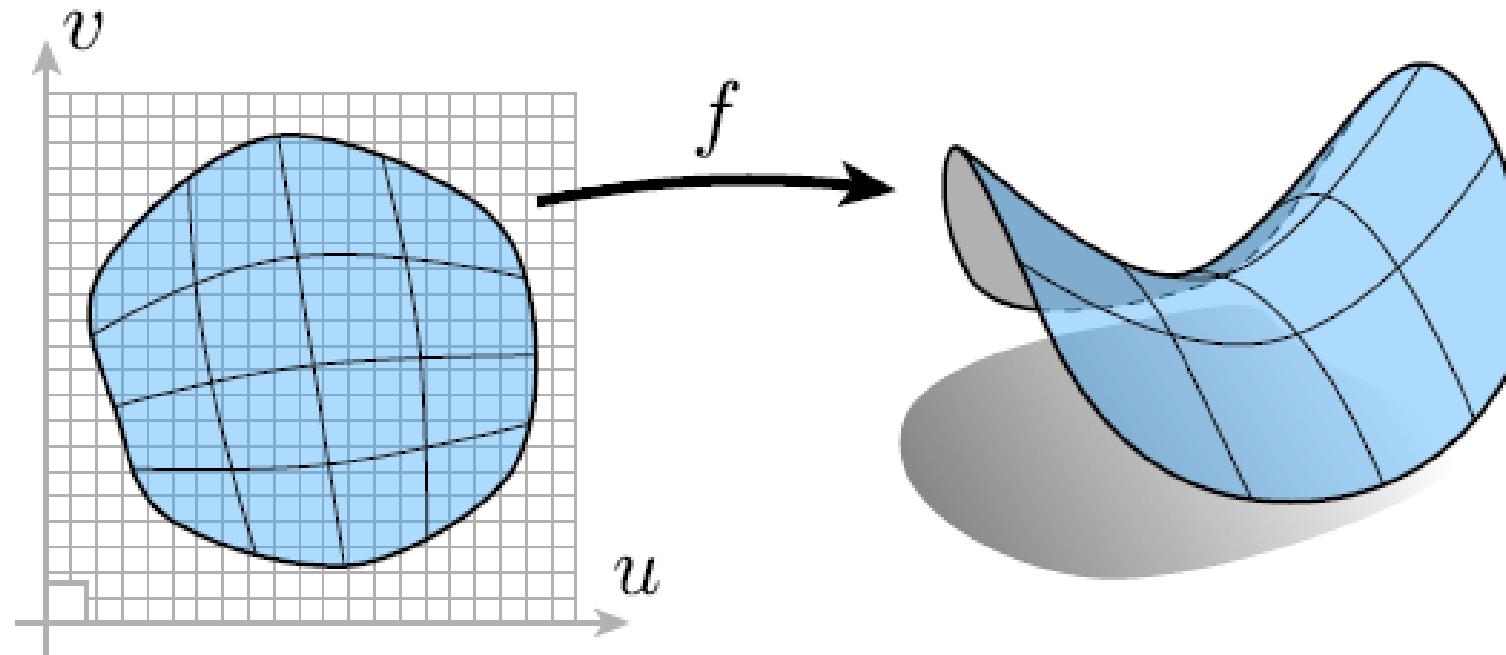
YES.



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

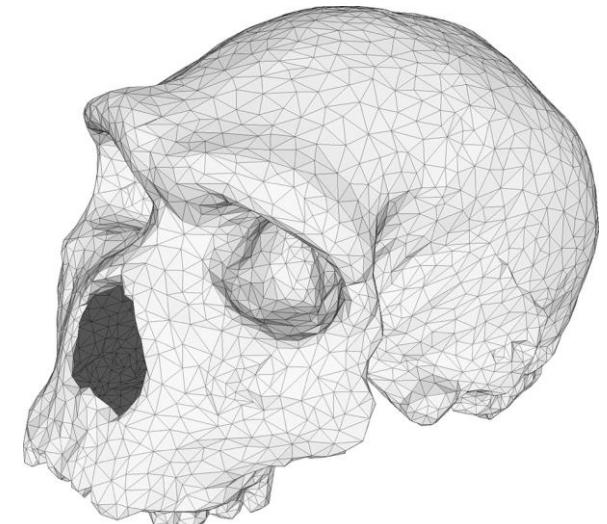
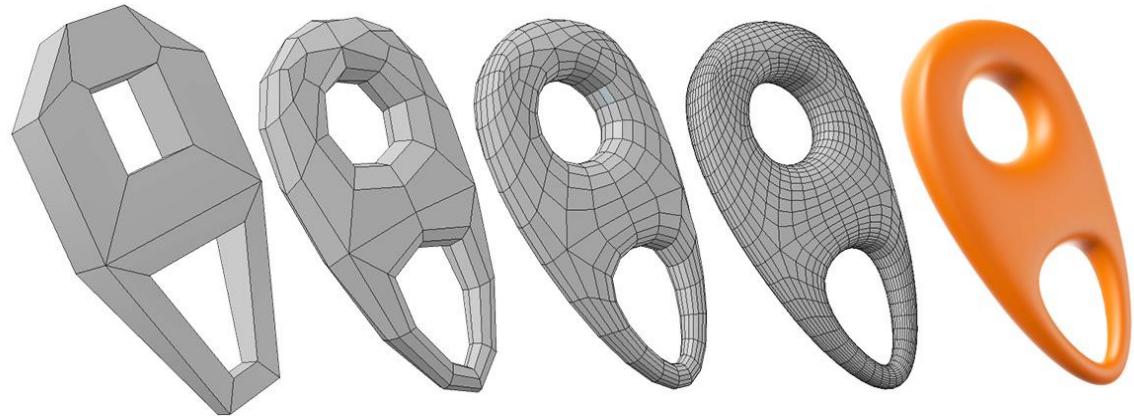
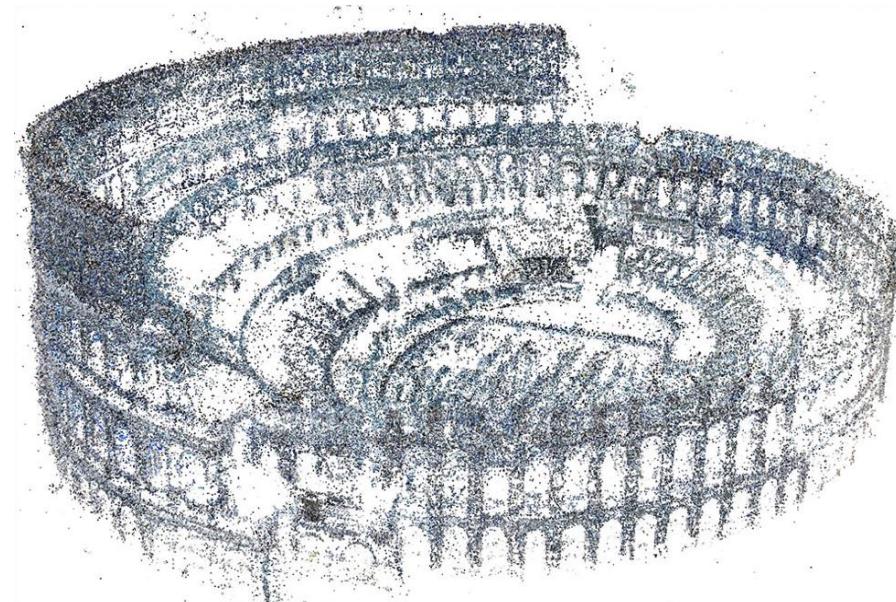
“Explicit” Representations of Geometry

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



Many explicit representations in graphics

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



(Will see some of these a bit later.)

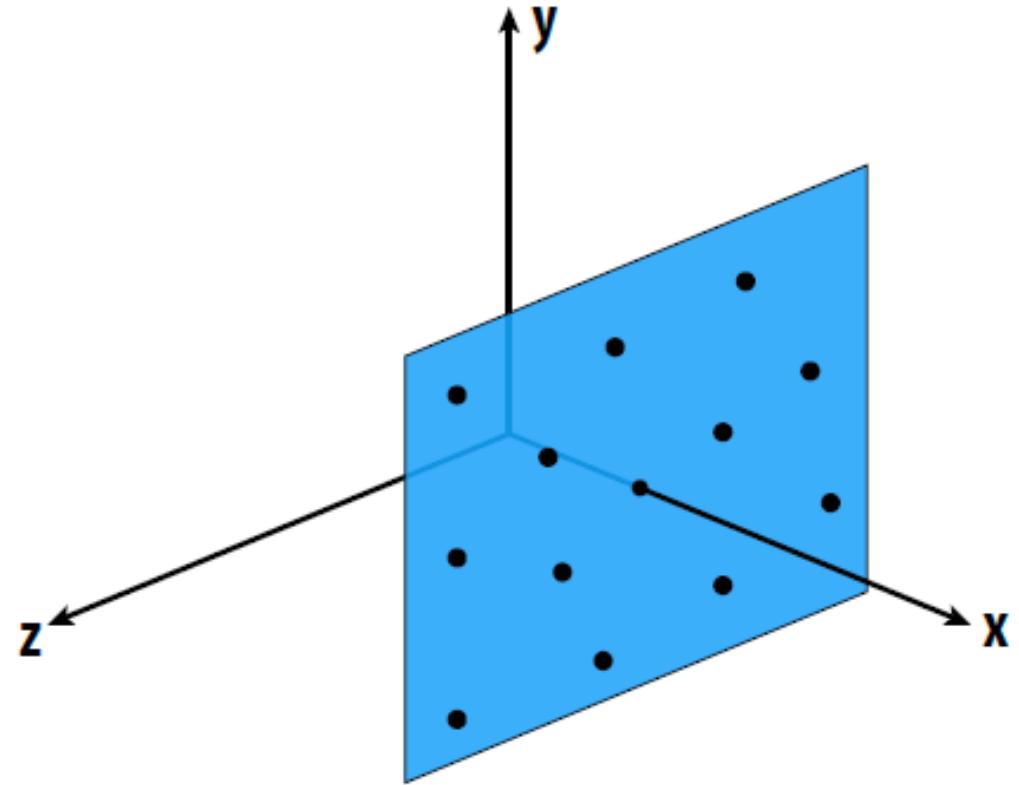
But first, let's play a game:

I'll give you an explicit surface.

You give me some points on it.

Sampling an explicit surface

- My surface is $f(u, v) = (1.23, u, v)$.
- Just plug in any values $(u,v)!$



- Explicit surfaces make some tasks easy (like sampling).

Let's play another game.

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

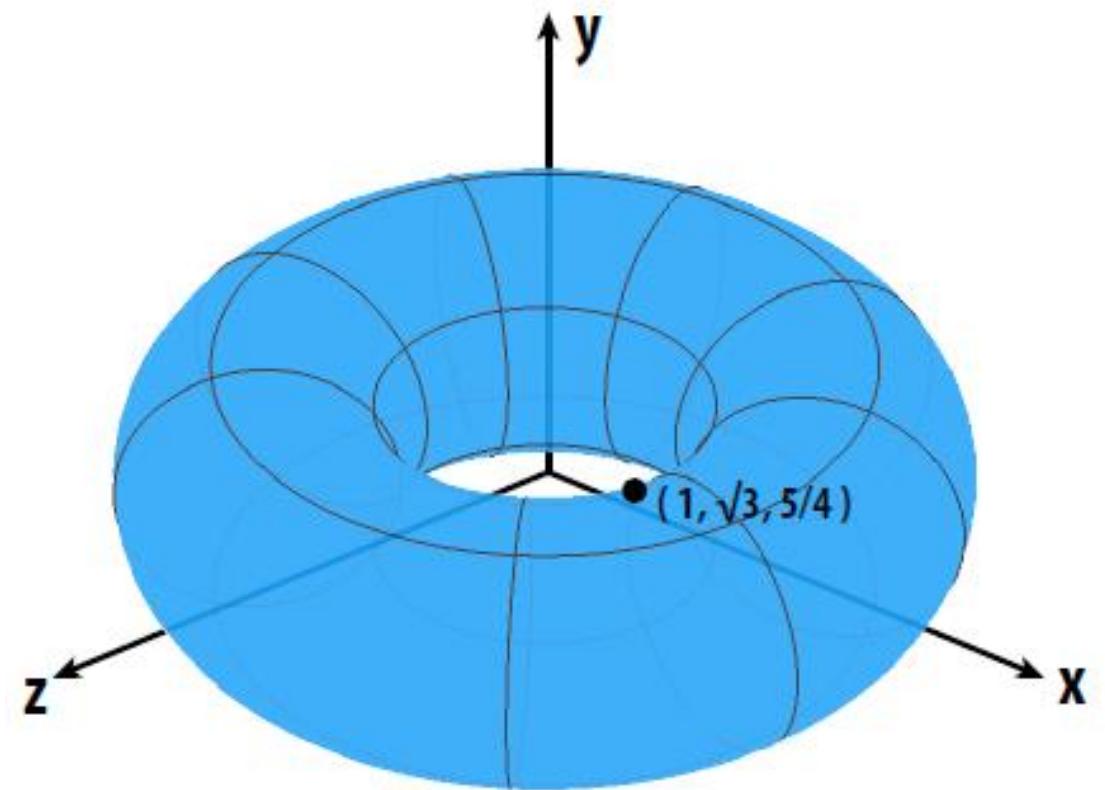
I want to see if a point is inside it.

Check if this point is inside the torus

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

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

...NO!



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

CONCLUSION:

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

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

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

Algebraic Surfaces (Implicit)

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



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

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

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

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

- What about more complicated shapes?



- Very hard to come up with polynomials!

Gradient of Implicit Surfaces

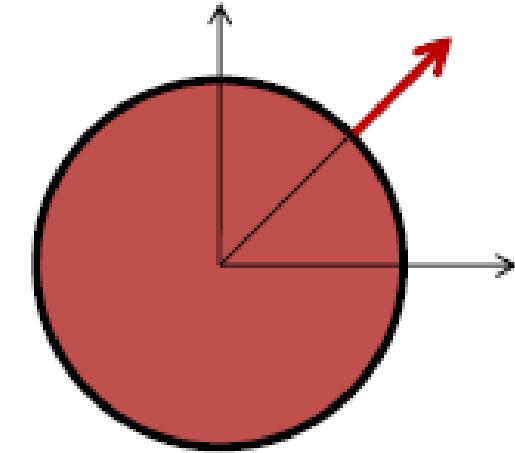
- The normal vector to the surface is given by the gradient of the (scalar) implicit function

$$\nabla g(x, y, z) = \left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}, \frac{\partial g}{\partial z} \right)^T$$

- Example

$$g(x, y, z) = x^2 + y^2 + z^2 - r^2$$

$$\nabla g(x, y, z) = (2x, 2y, 2z)^T$$

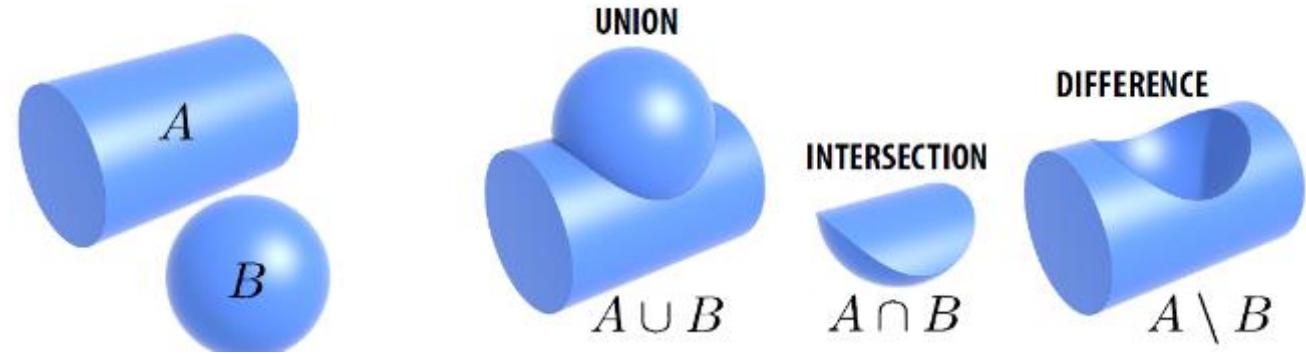


$$\nabla g(x, y, z) = (2, 2, 0)^T$$

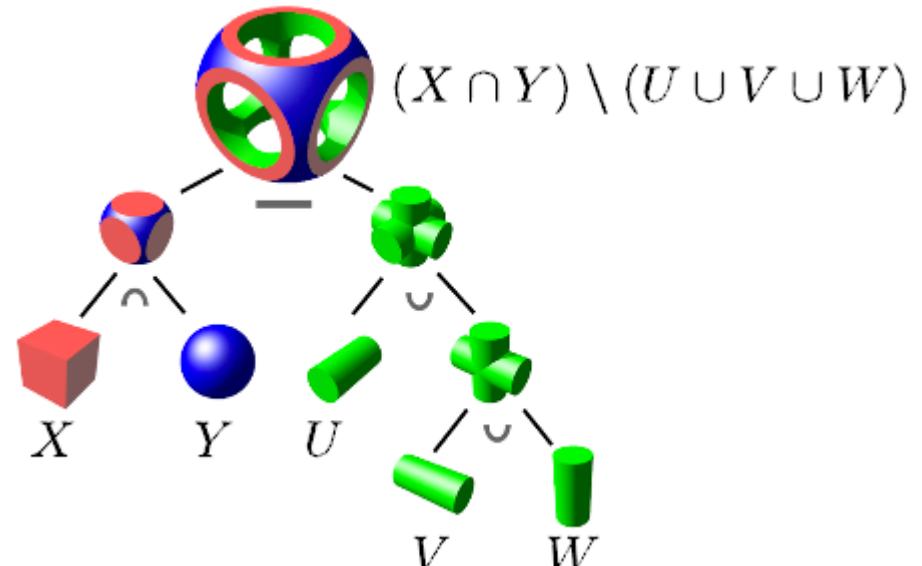
- Why is the condition that the function be regular (i.e. have non-vanishing derivative) necessary?
- How smooth is the surface?

Constructive Solid Geometry (Implicit)

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

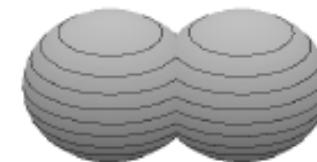


- Then chain together expressions:

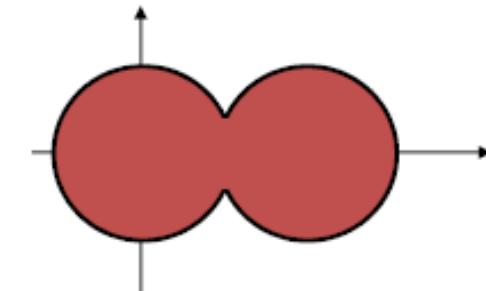


Implicit Surfaces -- Smooth set operation

- Standard operations: union and intersection



$$\bigcup_i g_i(\mathbf{p}) = \min_i g_i(\mathbf{p})$$
$$\bigcap_i g_i(\mathbf{p}) = \max_i g_i(\mathbf{p})$$



- In many cases, smooth blending is desired
 - Pasko and Savchenko [1994]

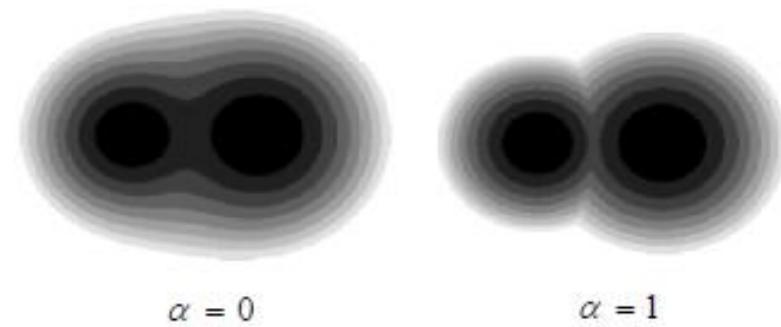
$$g \cup f = \frac{1}{1+\alpha} \left(g + f - \sqrt{g^2 + f^2 - 2\alpha gf} \right)$$

$$g \cap f = \frac{1}{1+\alpha} \left(g + f + \sqrt{g^2 + f^2 - 2\alpha gf} \right)$$

- alpha

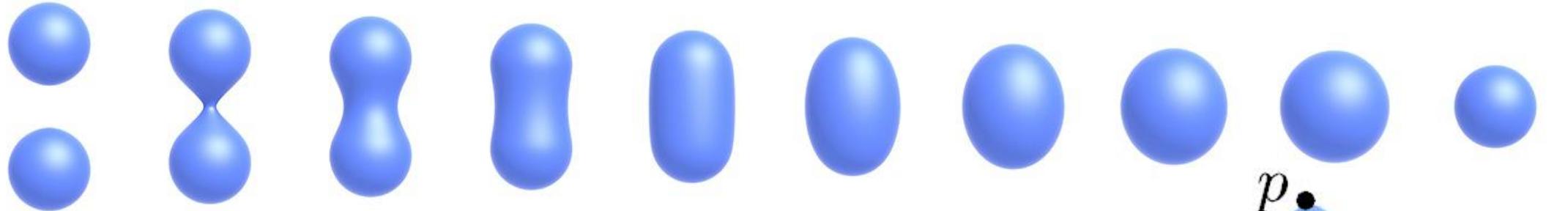
$$\lim_{\alpha \rightarrow 1} g \cup f = \frac{1}{2} \left(g + f - \sqrt{(g-f)^2} \right) = \frac{g+f}{2} - \frac{|g-f|}{2} = \min(g, f)$$

$$\lim_{\alpha \rightarrow 1} g \cap f = \frac{1}{2} \left(g + f + \sqrt{(g-f)^2} \right) = \frac{g+f}{2} + \frac{|g-f|}{2} = \max(g, f)$$



Blobby Surfaces (Implicit)

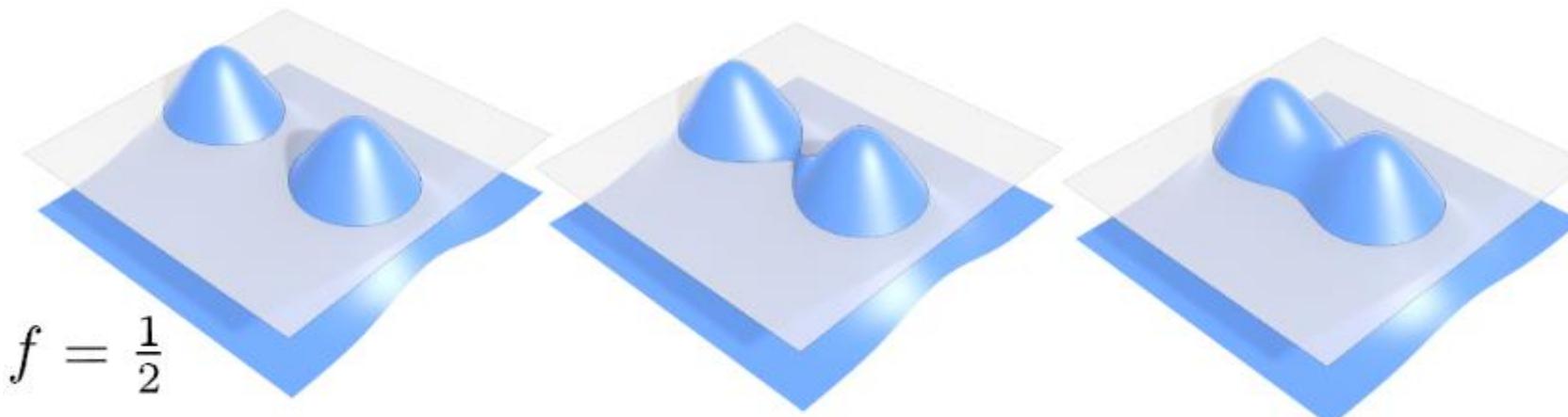
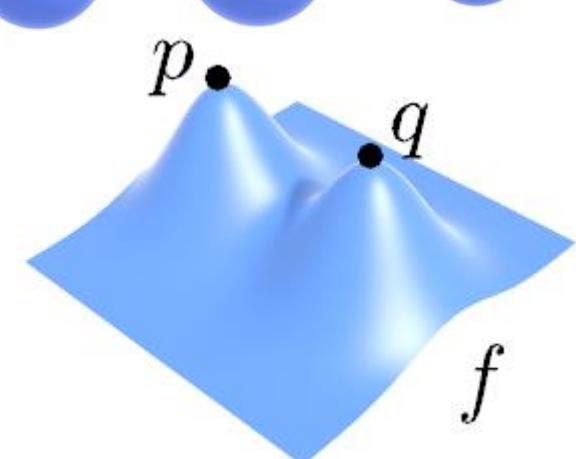
- Instead of Booleans, gradually blend surfaces together:



- Easier to understand in 2D:

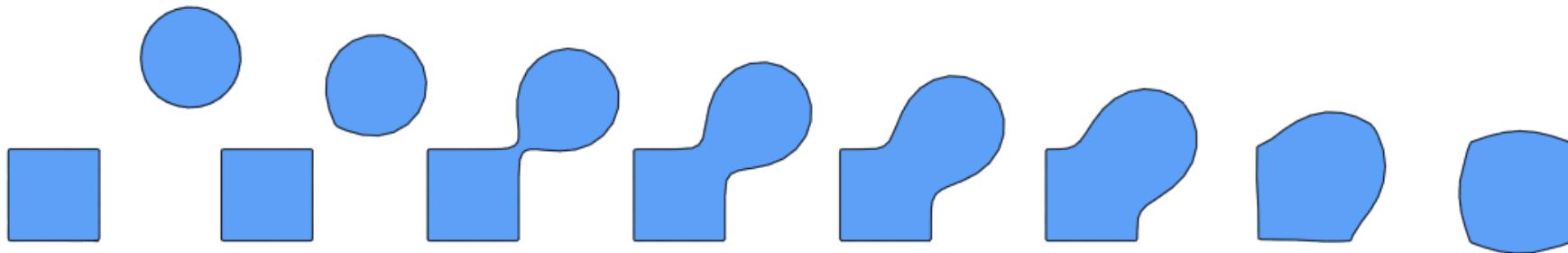
$$\phi_p(x) := e^{-|x-p|^2} \quad (\text{Gaussian centered at } p)$$

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



Blending Distance Functions (Implicit)

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



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

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

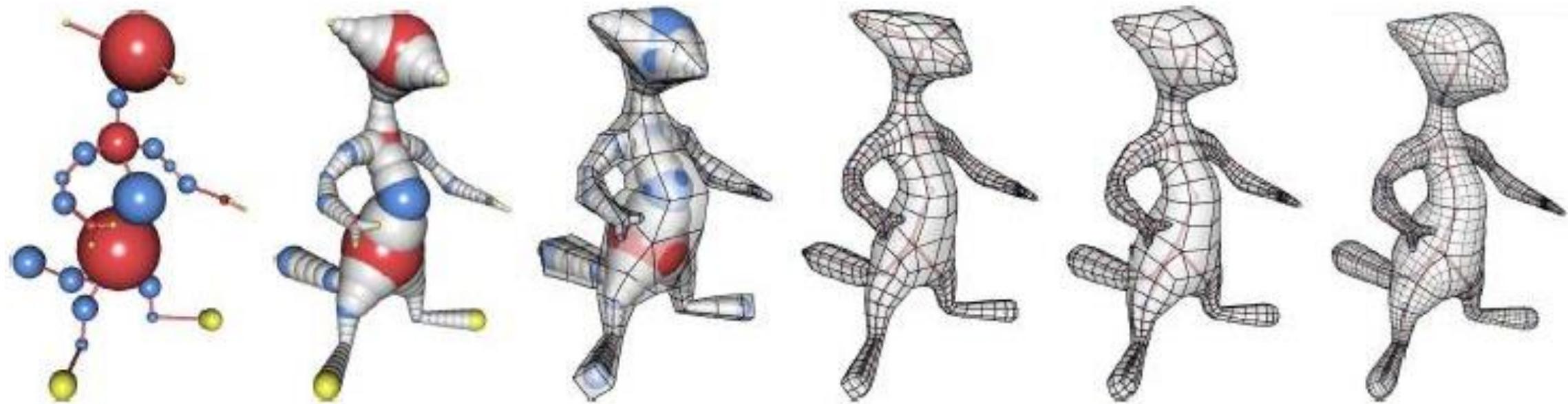
- Appearance depends on exactly how we combine functions
- Q: How do we implement a simple Boolean union?
- A: Just take the product: $f(x) := d_1(x)d_2(x)$

Scene of pure distance functions (not easy!)



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

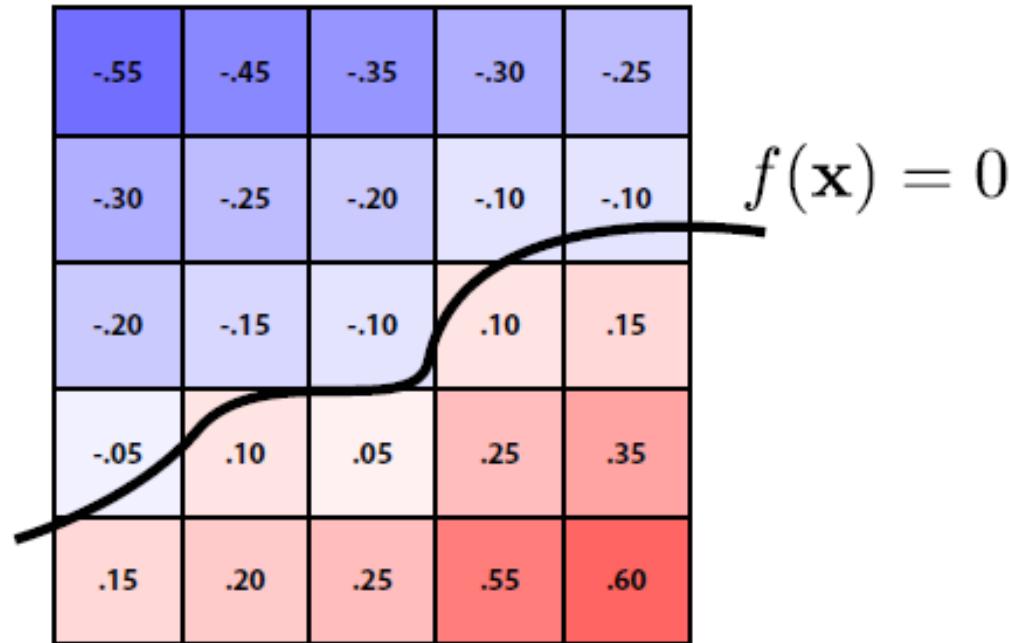
pg10_B-Mesh: A Fast Modeling System for Base Meshes of 3D Articulated Shapes



Overview of our B-Mesh modeling approach. (a) Specifying the skeleton and key-balls at the nodes by users; (b) creating inbetween-balls (in gray) by interpolating the key-balls; (c) generating an initial mesh; (d) subdividing the mesh (c); (e) evolving the mesh (d); (f) obtaining the final mesh with more subdivision and evolution.

Level Set Methods (Implicit)

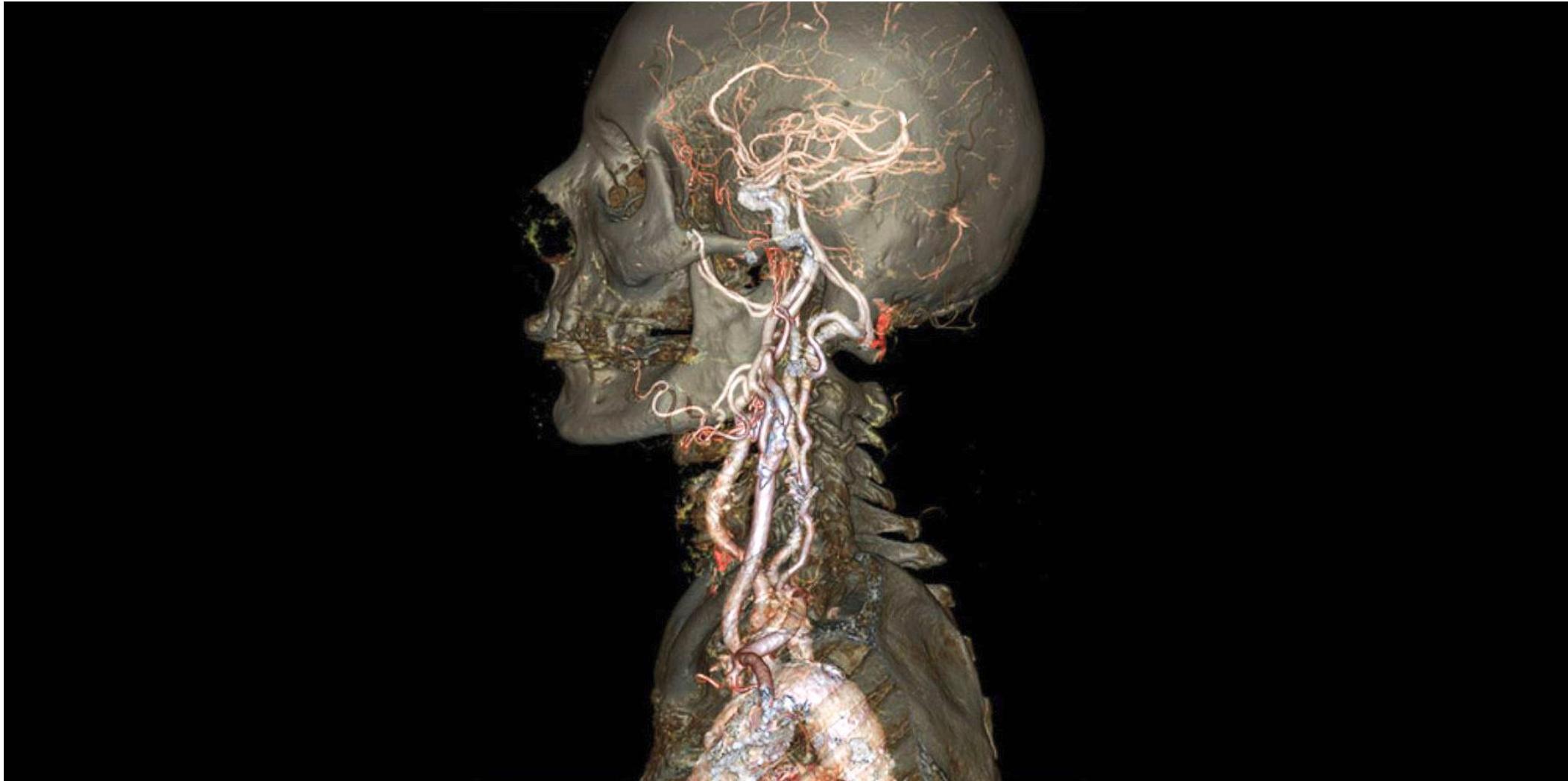
- Implicit surfaces have some nice features (e.g., merging/splitting)
- But, hard to describe complex shapes in closed form
- Alternative: store a grid of values approximating function



- Surface is found where *interpolated* values equal zero
- Provides much more explicit control over shape (like a texture)

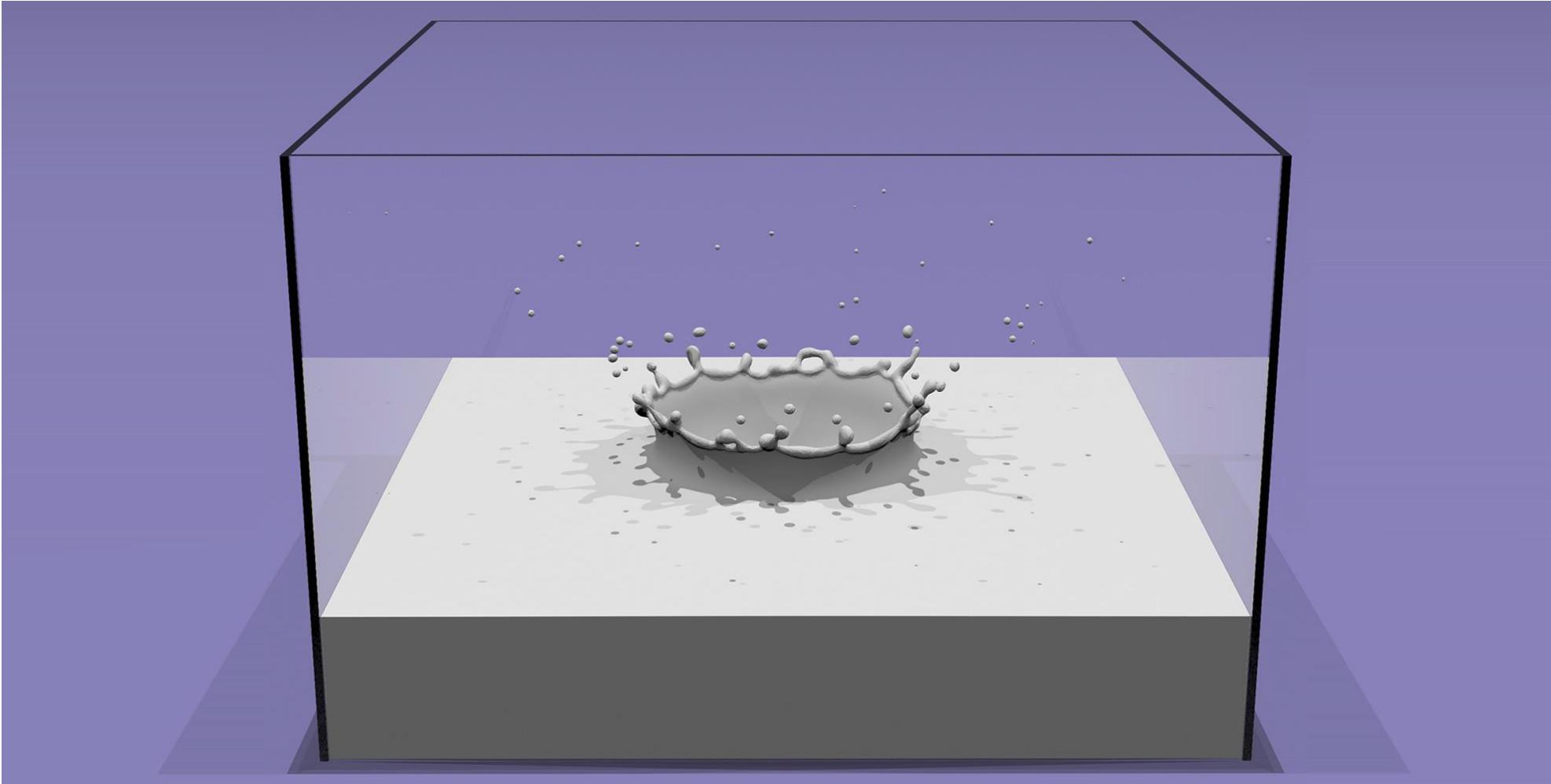
Level Sets from Medical Data (CT, MRI, etc.)

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



Level Sets in Physical Simulation

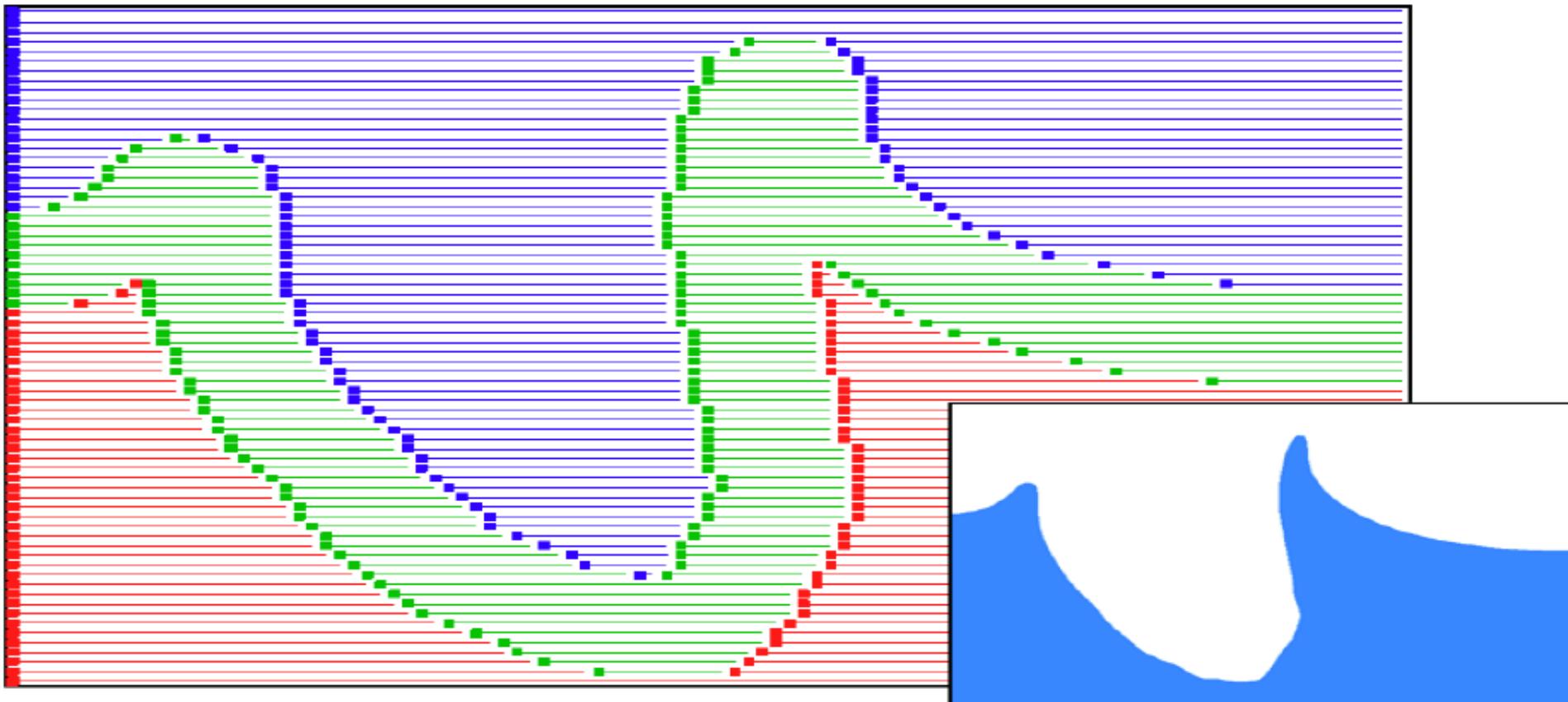
- Level set encodes distance to air-liquid boundary



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

Level Set Storage

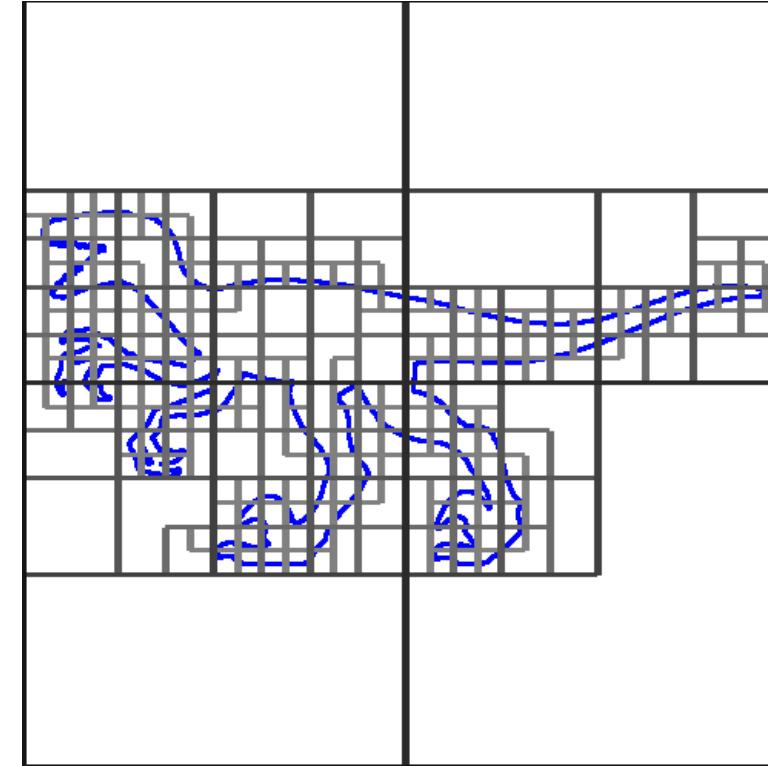
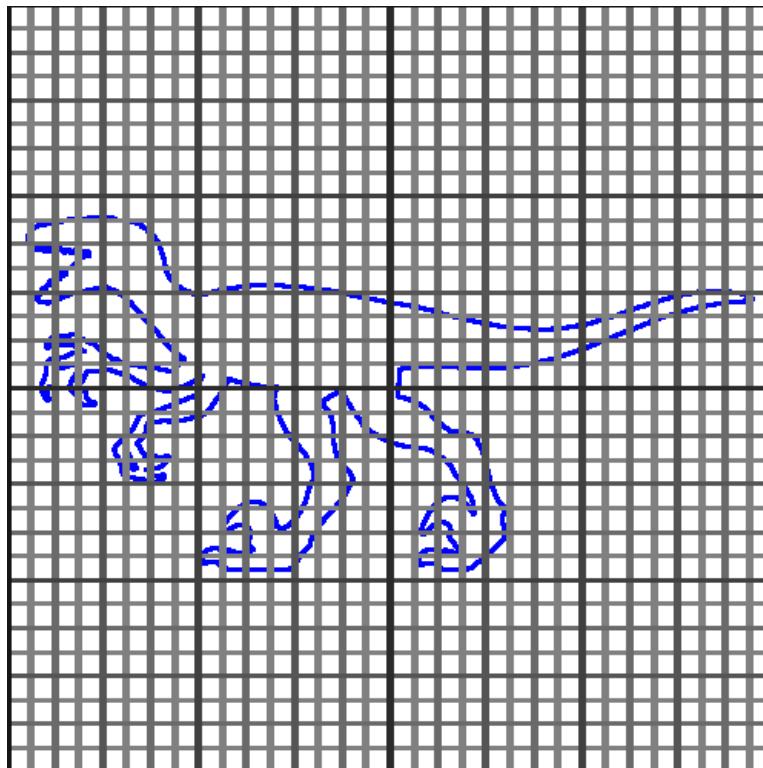
- Drawback: storage for 2D surface is now $O(n^3)$
- Can reduce cost by storing only a narrow band around surface:



Implicit Representations

Adaptive Grids: Quadtree

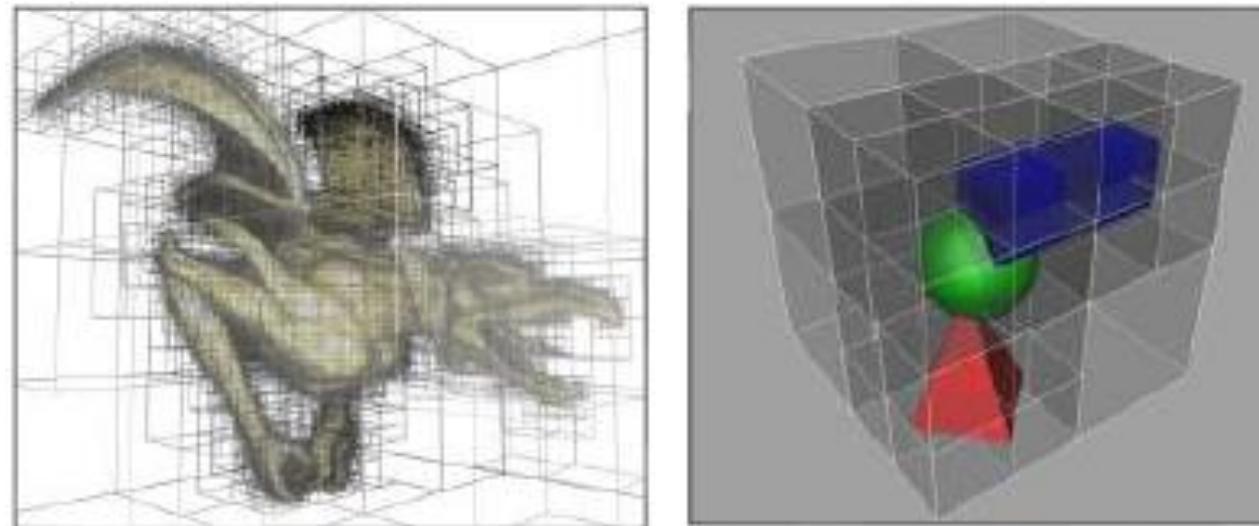
In practice, we may only need a high-precision representation of the implicit function near the surface, so represent the function over an adaptive grid.



Octree

- A hierarchical tree build by sequential subdivision (8) of a occupied cells.
- Adaptive, i.e. only splits when too many points in cell
- Proximity search by (recursive) tree traversal and distance to neighboring cells
- Tree might not be balanced
- Widely used for complicated scenes that need faster processing and lower accuracy

E.g. Collision detection in real-time simulation or animation



Fractals (Implicit)

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



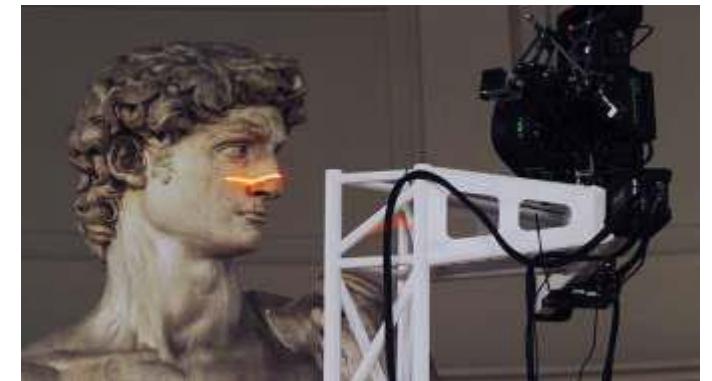
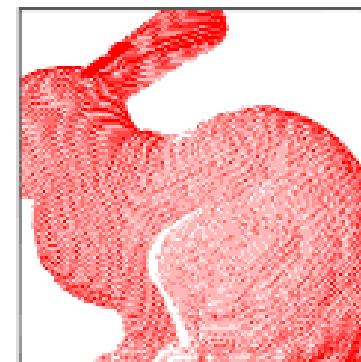
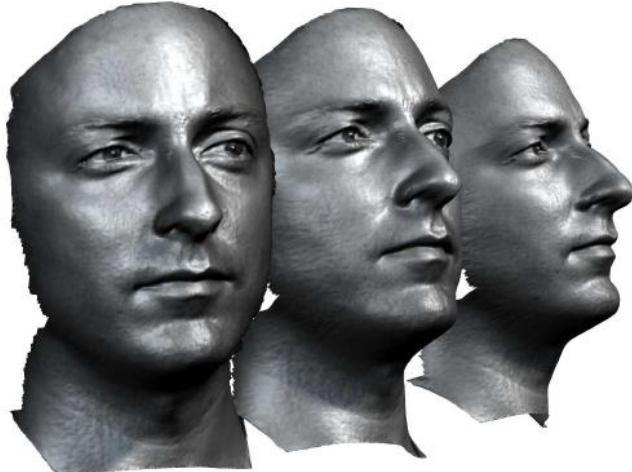
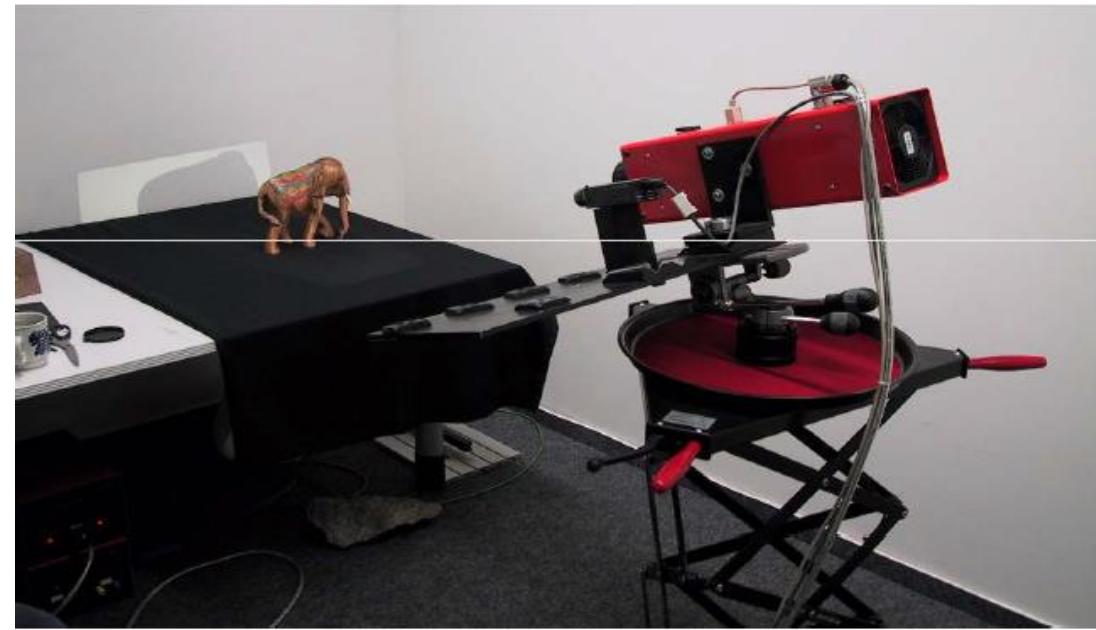
Implicit Representations - Pros & Cons

- **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*

Shape Acquisition

Sampling of real world objects

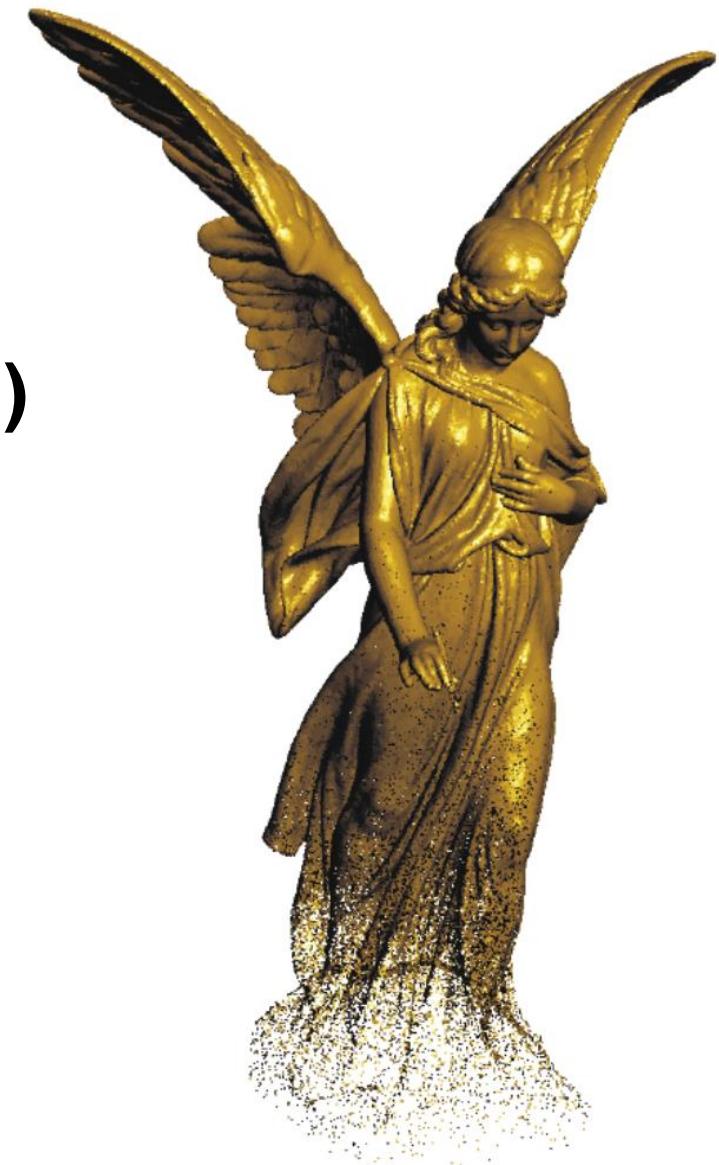
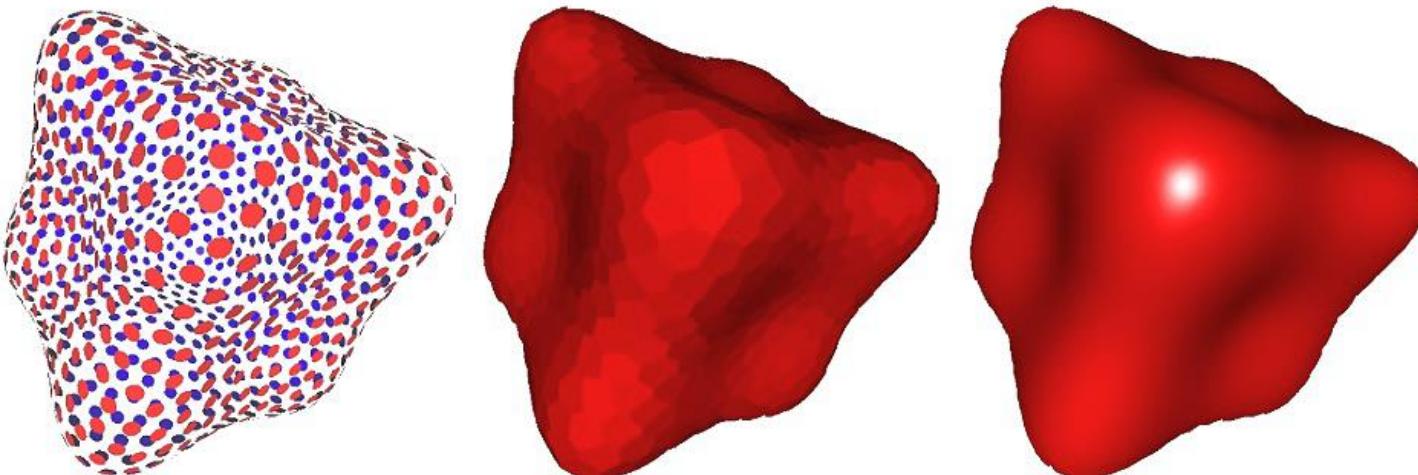
- Scanners
 - Laser
 - Depth imaging
- Properties & Operations
 - Potentially noisy, with outliers
 - Registration of multiple images
 - Non-uniform sampling, sparse, holes



180 frames per second (From David Gu)

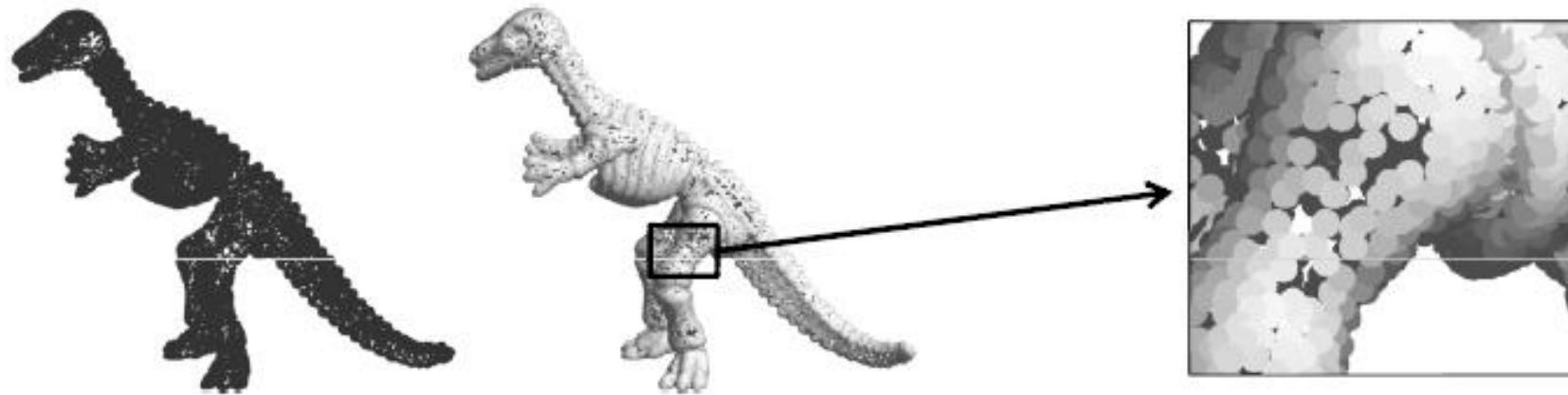
What about explicit representations?

- 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)
- Difficult to draw in undersampled regions
- Hard to do processing / simulation



Points: Neighborhood information

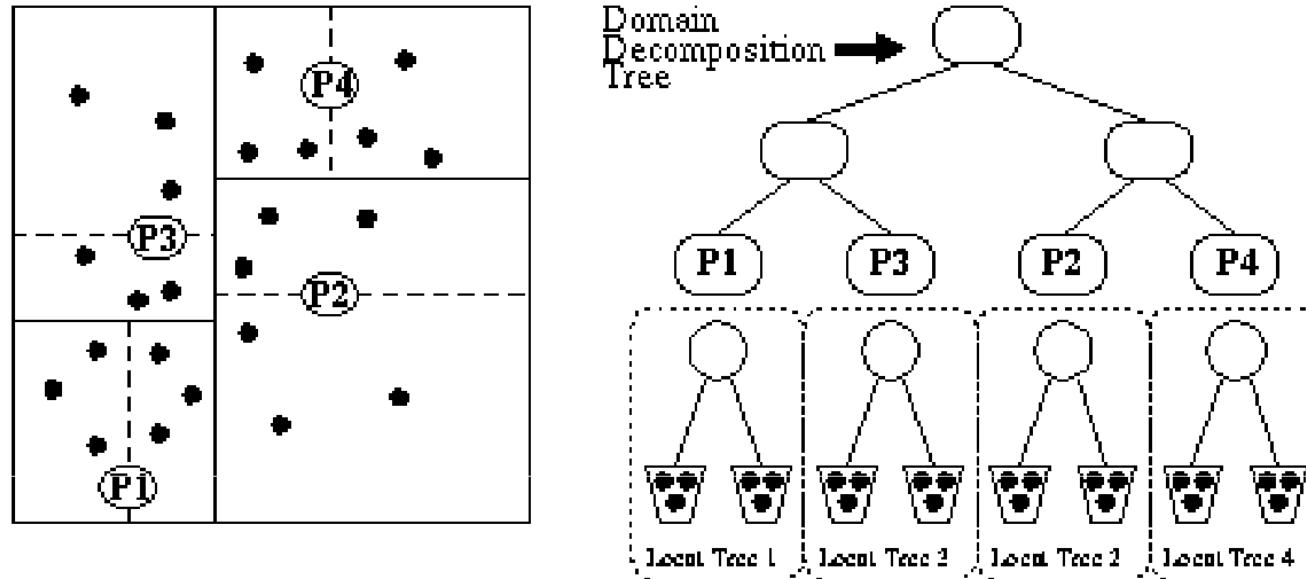
- Efficient point processing & modeling requires a spatial partitioning data structure
 - To figure out **neighborhoods**
 - Why do we need neighbors?



- Need sub-linear implementations of
 - K-nearest neighbors to point x (knn)
 - In radius search

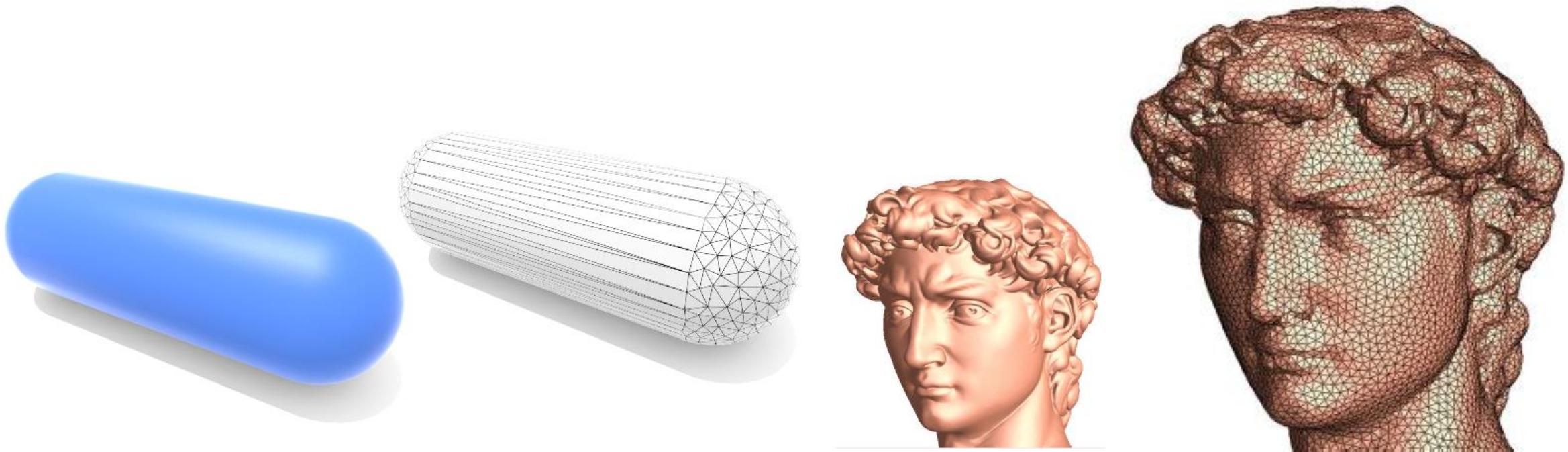
Kd-Tree

- Each cell is individually split along the median into two cells
- Same amount of points in cells
- Perfectly balanced tree
- Proximity search similar to the recursive search in an Octree.
- More data storage required for inhomogeneous cell dimensions



Polygon Mesh (Explicit)

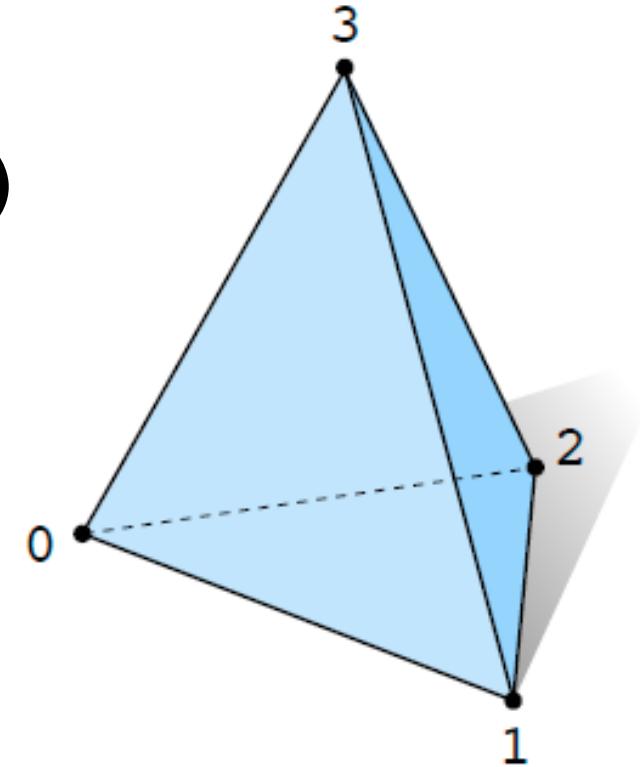
- Store vertices *and* polygons (most often triangles or quads)
- Easier to do processing/simulation, adaptive sampling
- More complicated data structures
- Perhaps most common representation in graphics



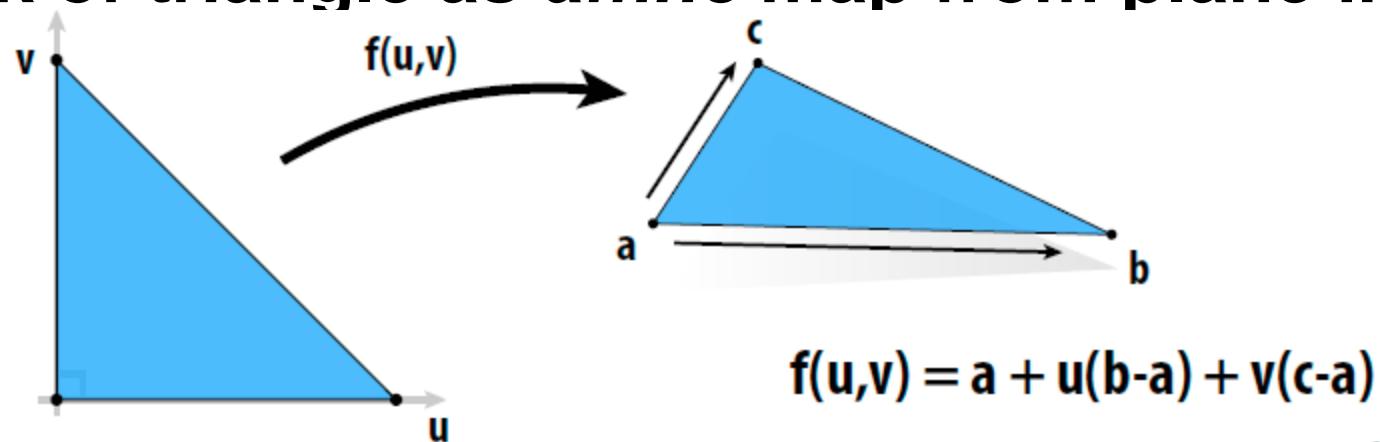
Triangle Mesh (Explicit)

- 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

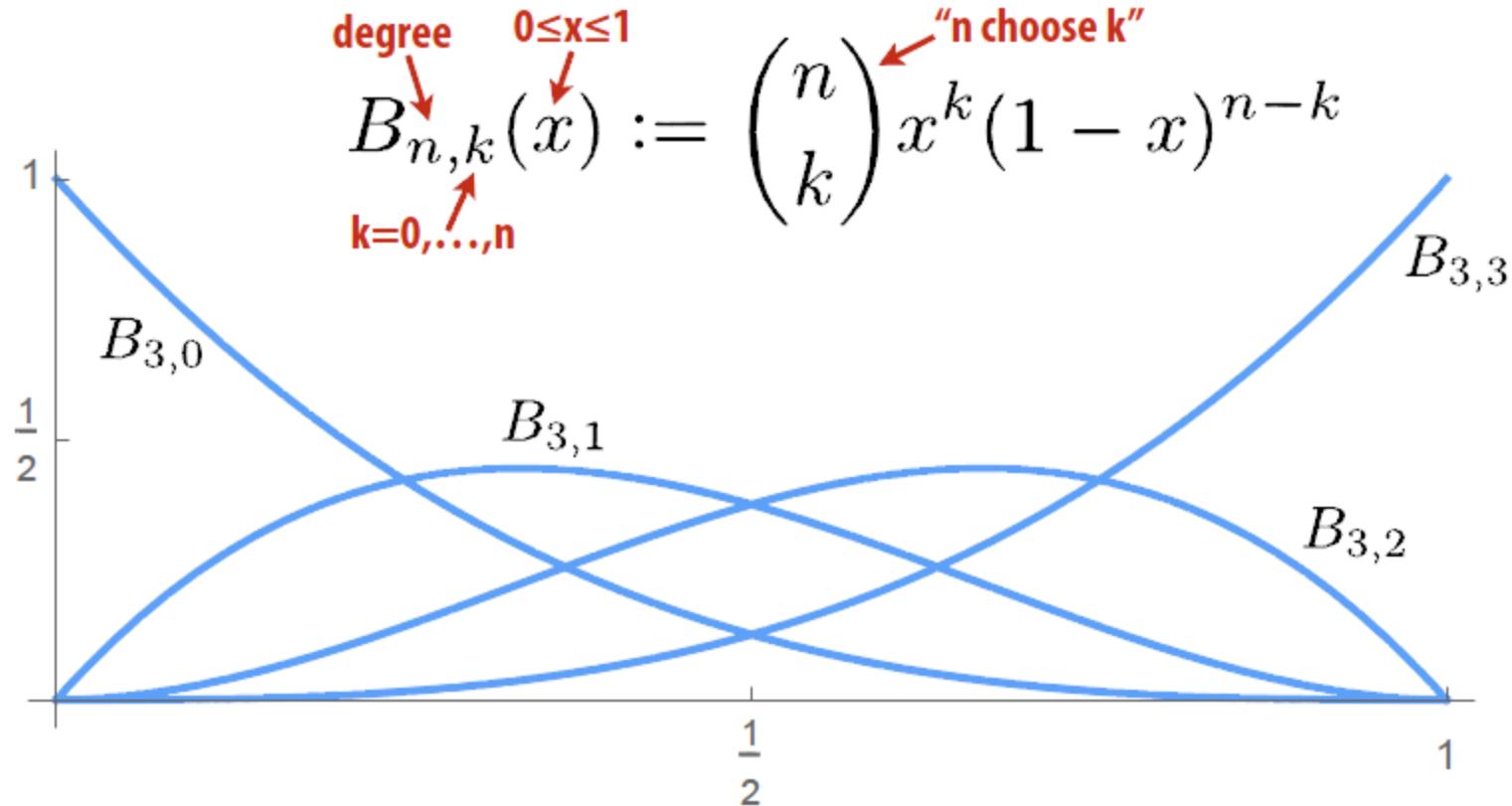


- Can think of triangle as affine map from plane into space:



Bernstein Basis

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



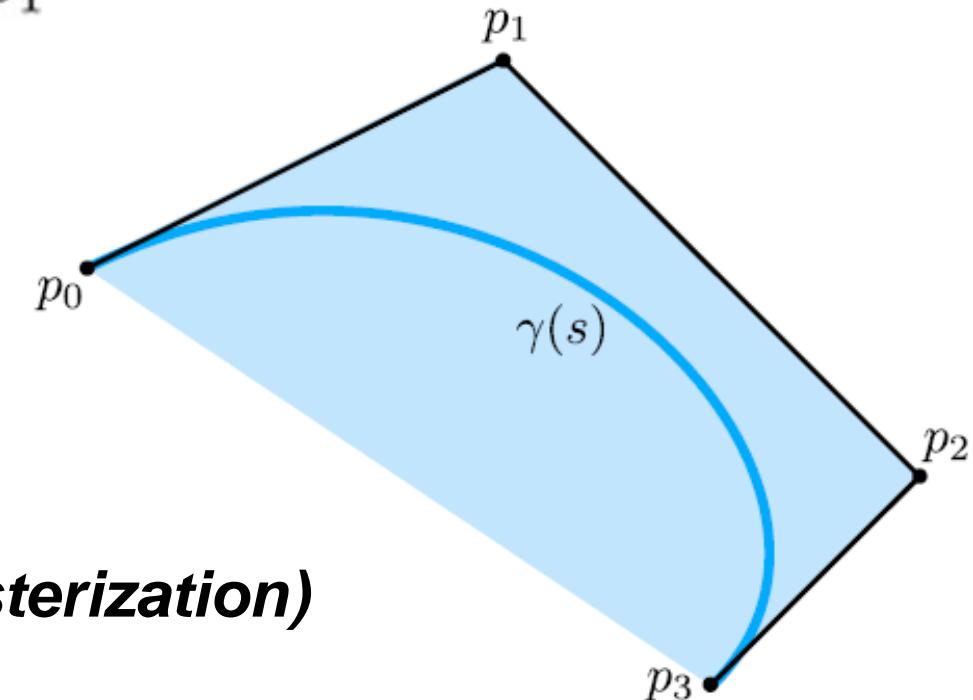
Bézier Curves (Explicit)

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

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

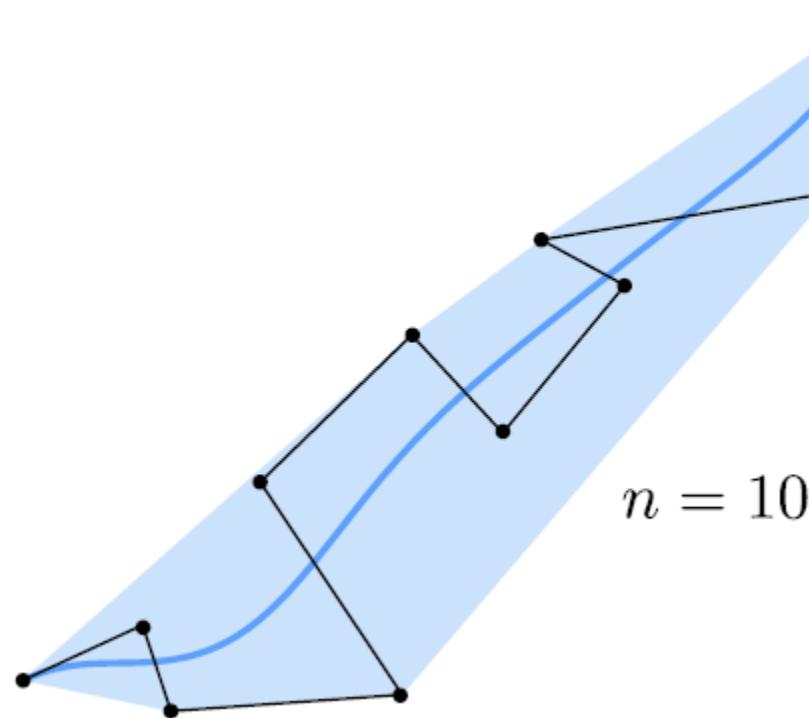
control points

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



Higher-order Bézier Curves?

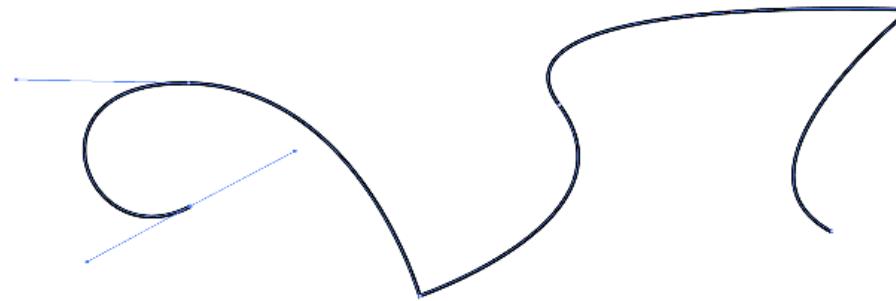
- What if we want a more interesting curve?
- High-degree Bernstein polynomials don't interpolate well:



Very hard to control!

B-Spline Curves (Explicit)

- Instead, use many low-order Bézier curve (B-spline)
- Widely-used technique in software (Illustrator, Inkscape, etc.)



- Formally, piecewise Bézier curve:

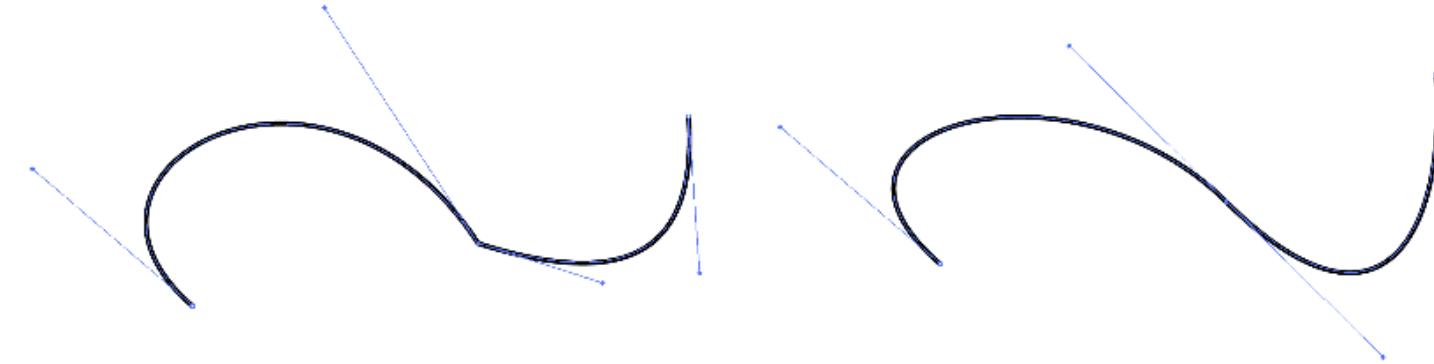
$$\gamma(u) := \gamma_i \left(\frac{u - u_i}{u_{i+1} - u_i} \right), \quad u_i \leq u < u_{i+1}$$

B-spline
→
Bézier
→

- Location of u_i parameters are called “knots”

B-Splines — tangent continuity

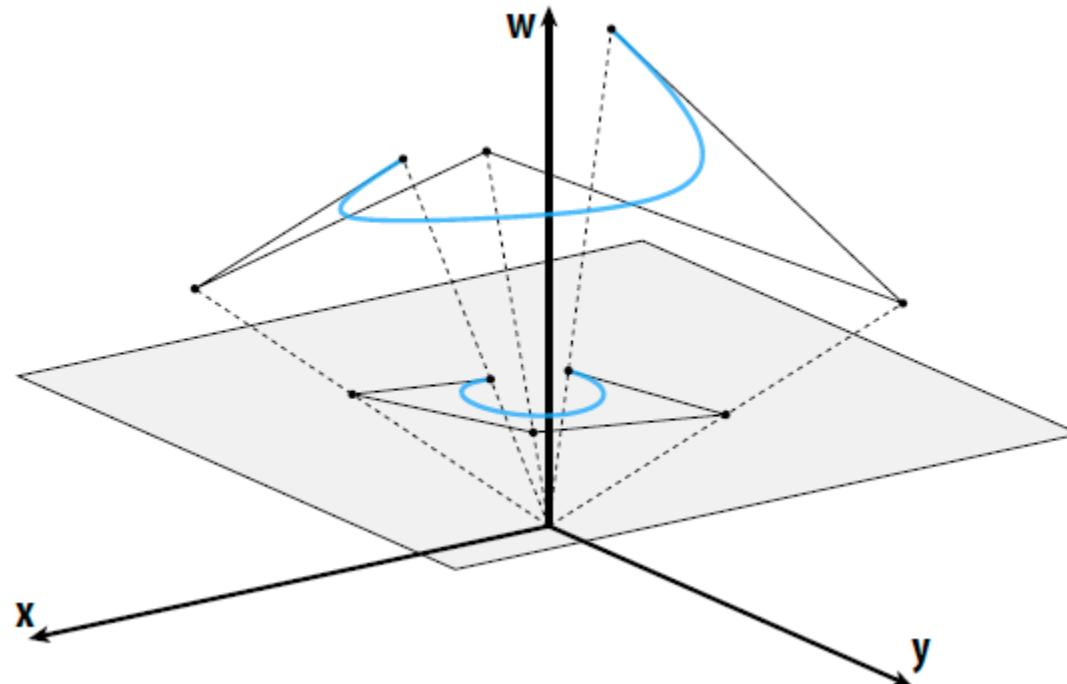
- To get “seamless” curves, want *tangents* to line up:



- Ok, but how?
- Each curve is cubic: $u^3 p_0 + 3u^2(1-u)p_1 + 3u(1-u)^2 p_2 + (1-u)^3 p_3$
- Q: How many degrees of freedom in a single cubic Bézier
- Tangents are difference between first two & last two points
- Q: How many degrees of freedom per B-spline segment?
- Q: Could you do this with quadratic Bézier? Linear Bézier?

Rational B-Splines (Explicit)

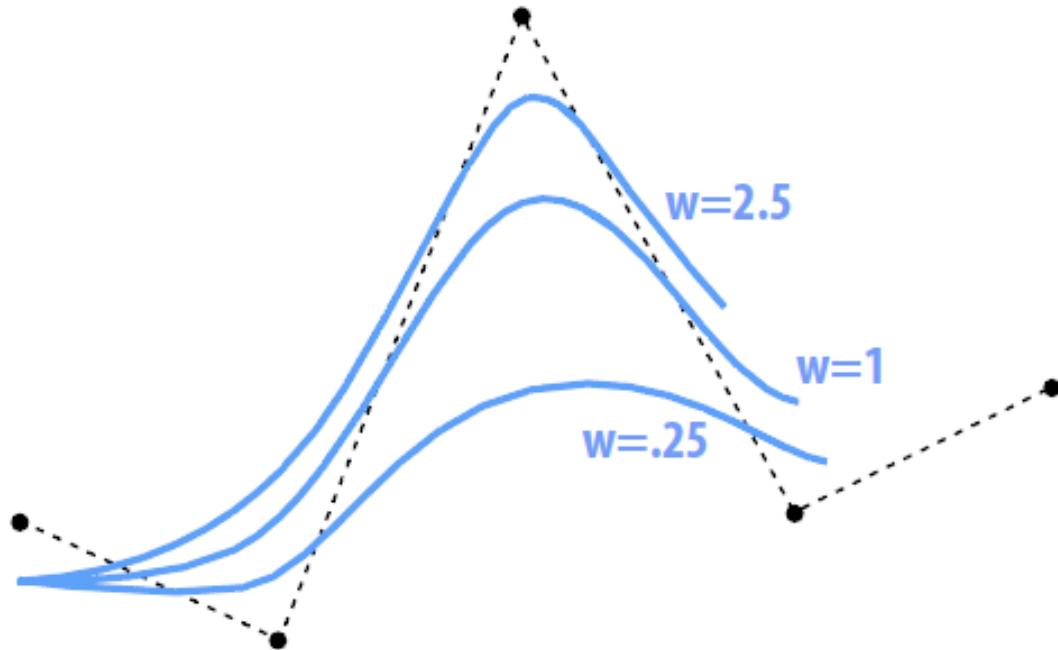
- B-Splines can't exactly represent **conics**—not even the circle!
- Solution: interpolate in homogeneous coordinates, then
- project back to the plane:



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:

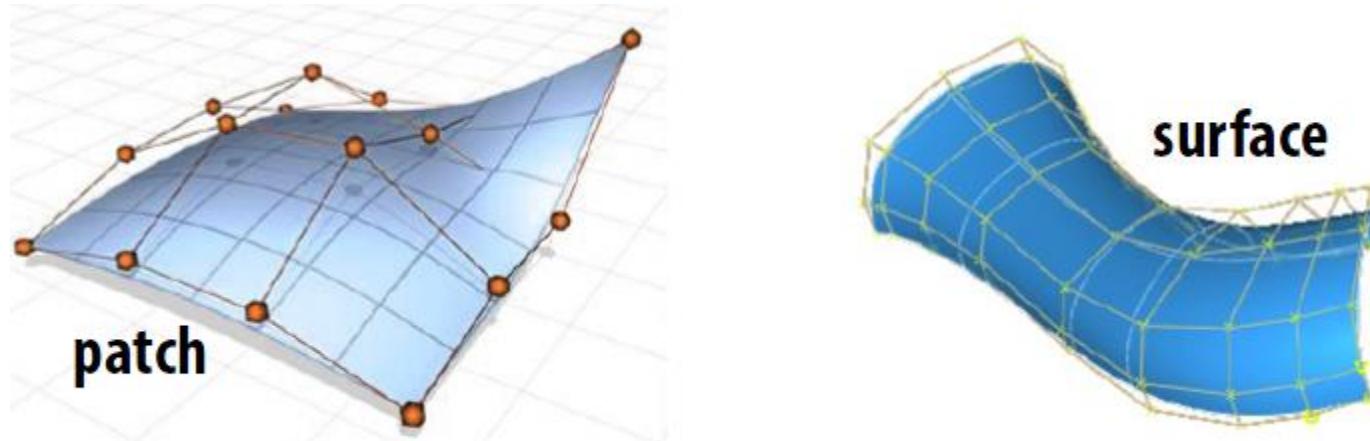


NURBS Surface (Explicit)

- Much more common than using NURBS for curves
- Use *tensor product* of scalar NURBS curves to get a patch:

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

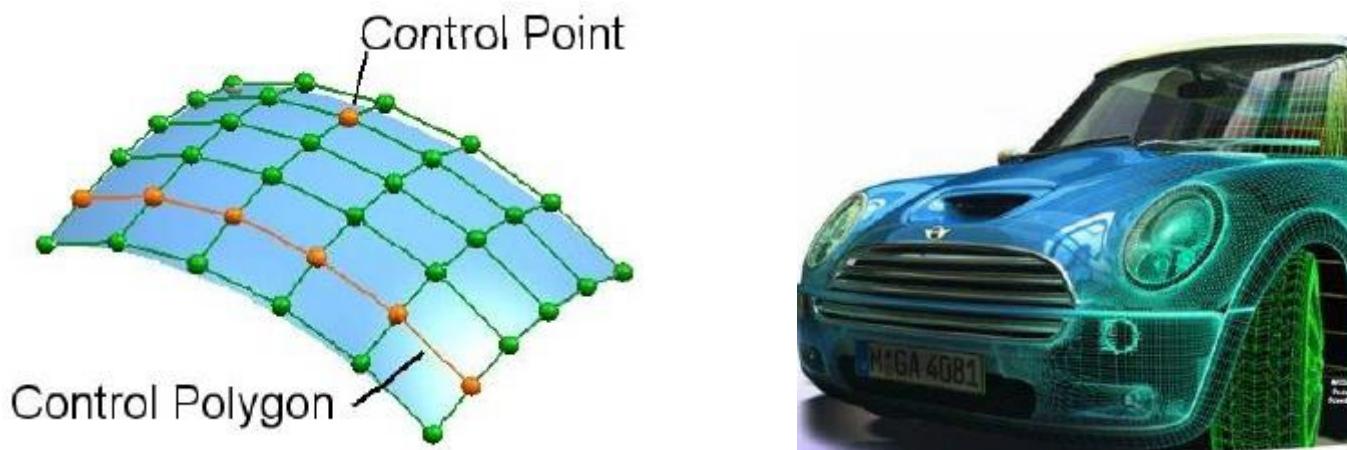
- Multiple NURBS patches form a surface



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

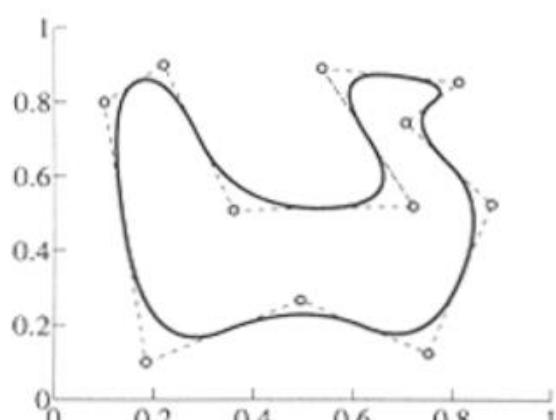
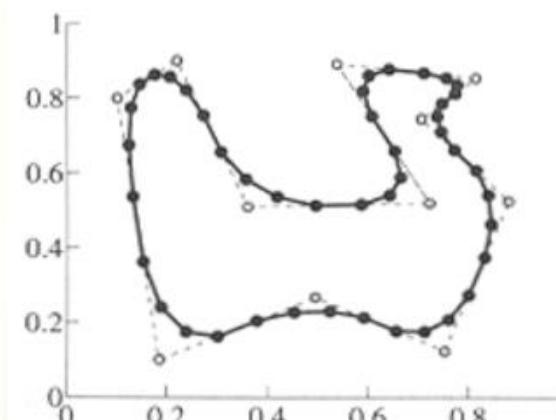
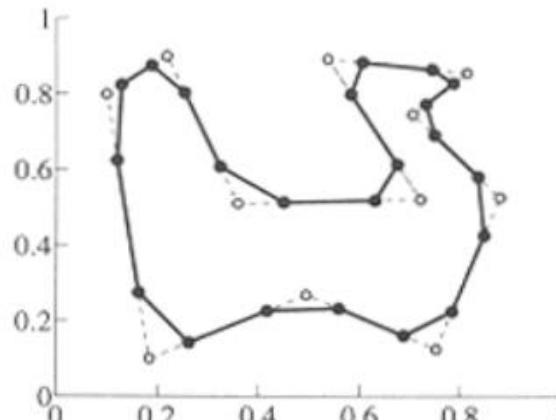
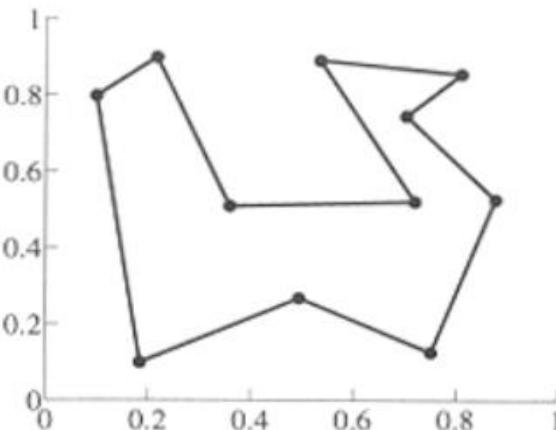
Spline & NURBS

- Extract analytical rep.
- Support interactive shape editing
- Compact rep.
- Major modeling techniques in CAD



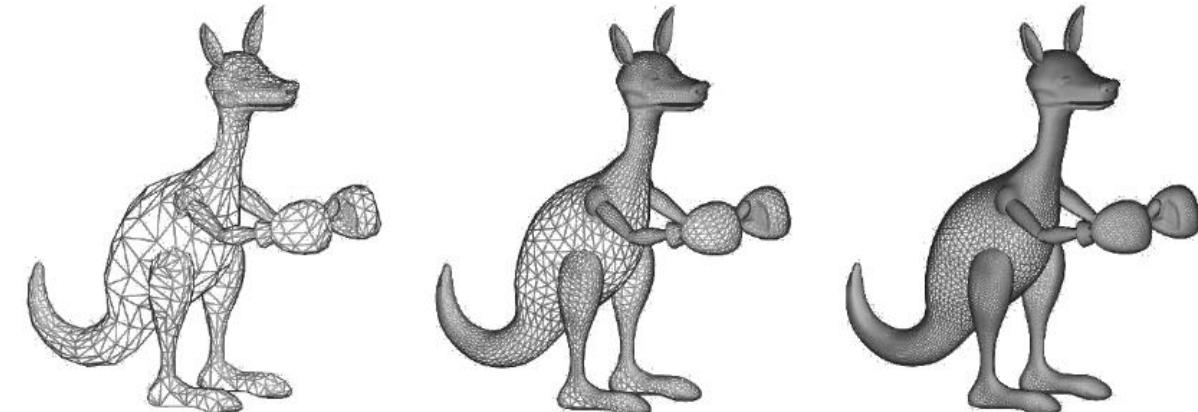
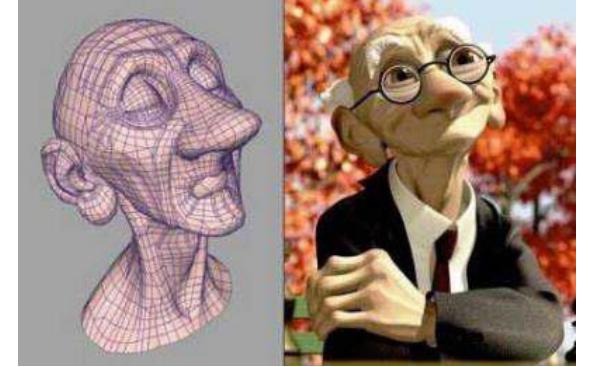
Subdivision (Explicit)

- Alternative starting point for B-spline curves: *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
 - Average with “next” neighbor (Chaikin): quadratic B-spline
 - Lane-Riesenfeld: B-spline curve of any degree



Subdivision Surfaces (Explicit)

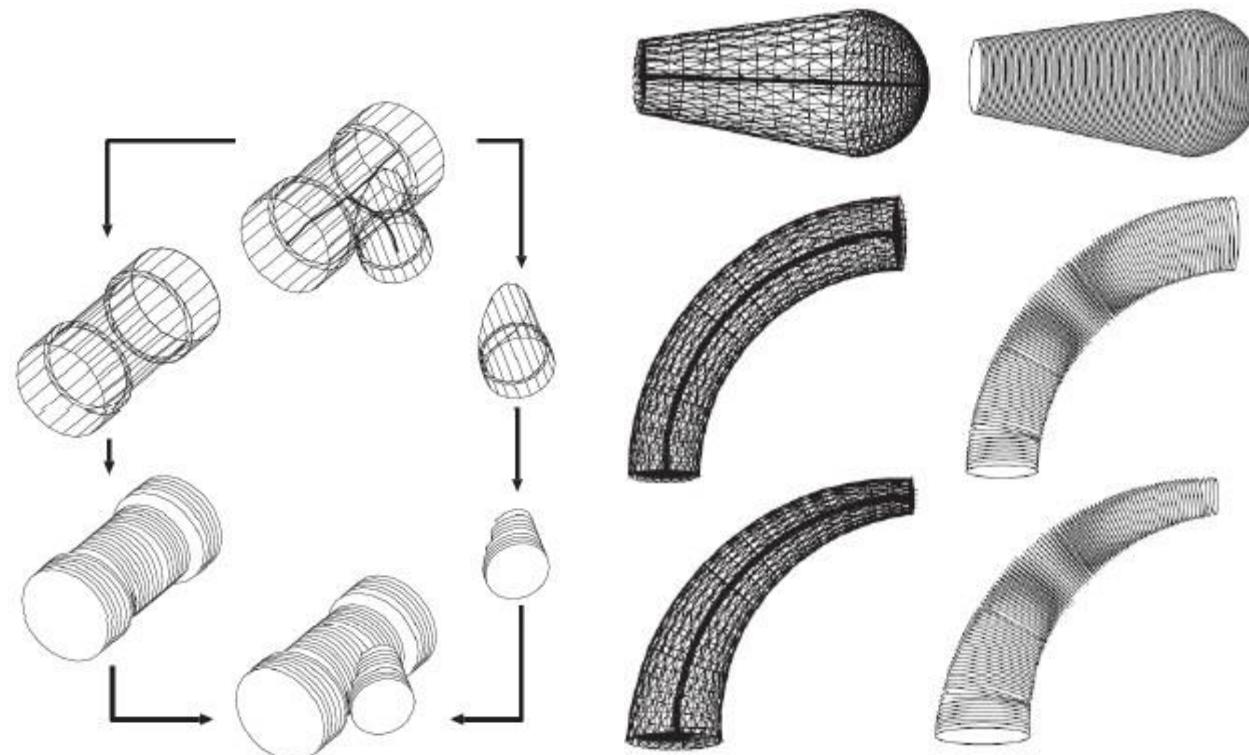
- Start with coarse polygon mesh (“control cage”)
- Subdivide each element
- Update vertices via local averaging
- Many possible rule:
 - Catmull-Clark (quads)
 - Loop (triangles)
 - ...
- Common issues:
 - interpolating or approximating?
 - continuity at vertices?
- Easier than NURBS for modeling; harder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



Other representations

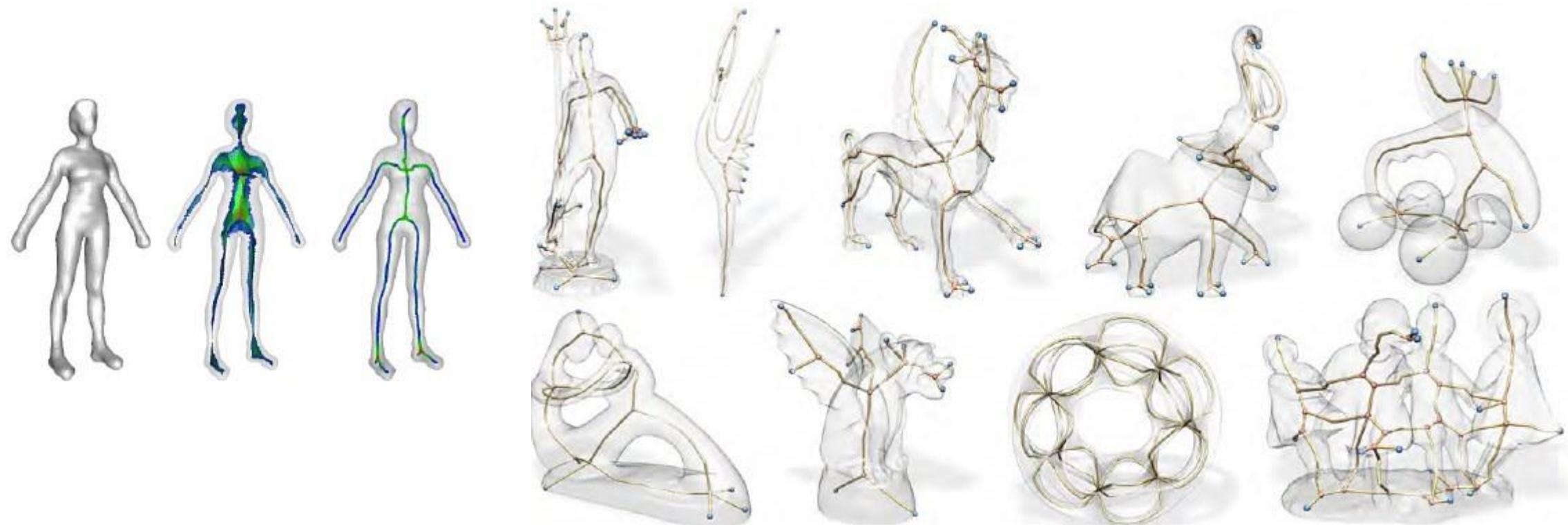
Generalized cylinder rep.

- A shape = {axis, a cross-section curve, a scaling function}
- Good for symmetric shapes with few local details and with clear skeletal structure
- Widely used in vision community for shape recognition, and shape recover

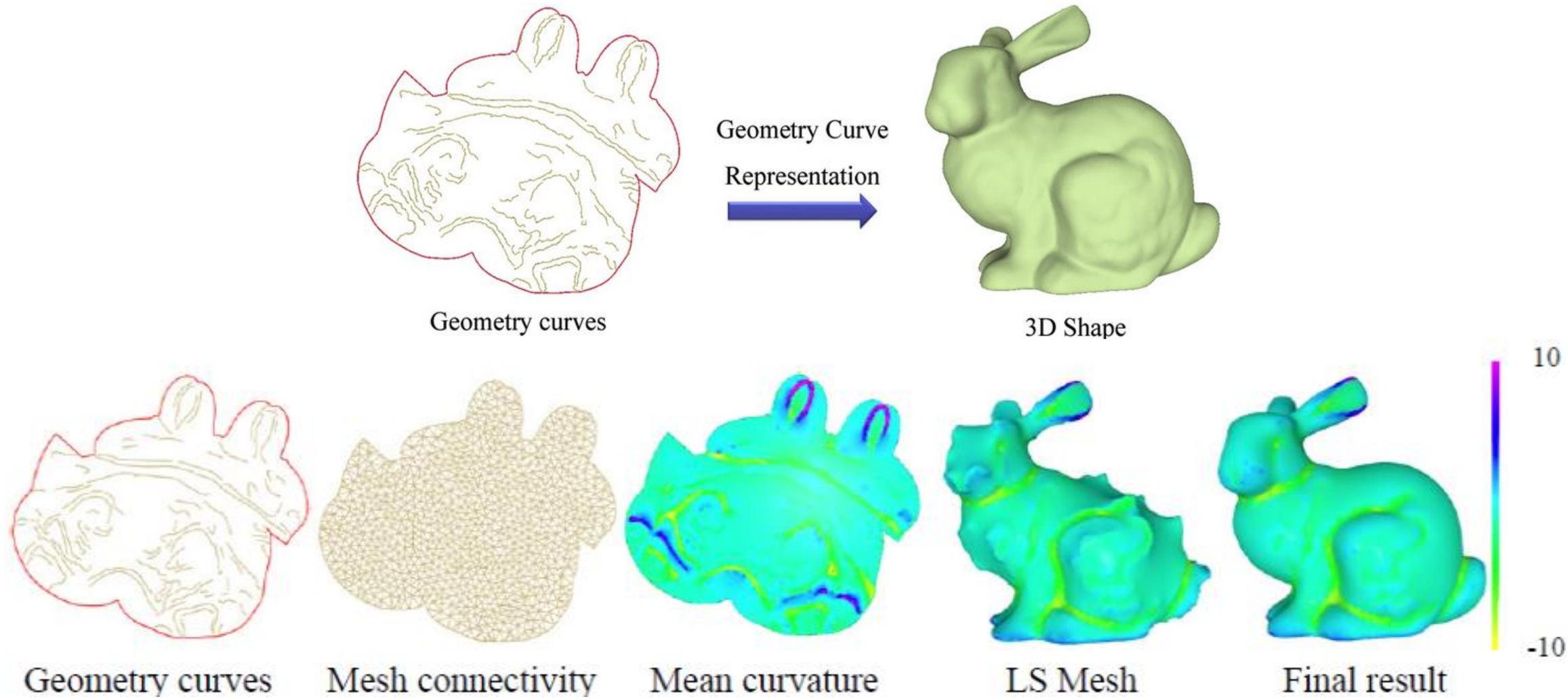


Skeleton Rep.

- A thin 1D or 2D representation of 2D/3D objects
- A (hierarchical) set of bones + attached skins
- Widely used in animation, matching, object recognition



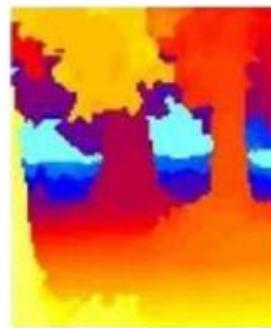
Gm13_Geometry Curves: A Compact Representation for 3D Shapes



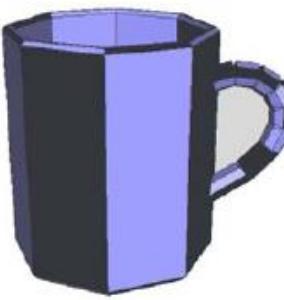
Shape Representation



(a)点云



(b)深度图像



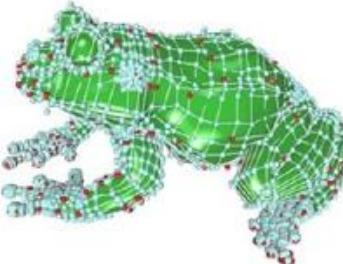
(c) Polygon soup



(d)网格



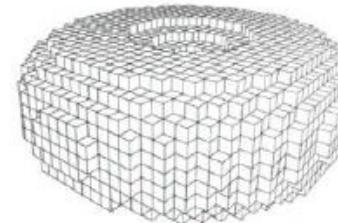
(e)细分



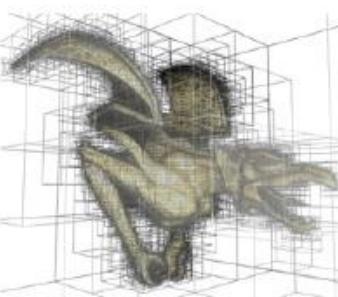
(f)参数曲面



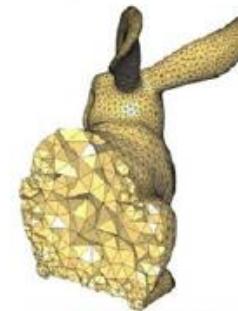
(g)隐式曲面



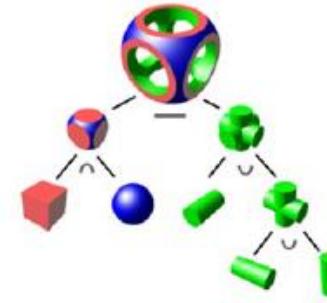
(h)体素表示



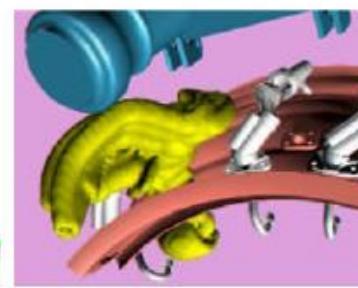
(i)空间分解表示



(j)四面体网格



(k)构造实体几何



(l)扫描表示

Resources

- read & display a mesh: jjcao_plot/eg_trisurf.m
- Read & display a huge point set (100k to 1 million points)
 - [PC_processing_1.0](#)
 - jjcao_code/tools/pcd_viewer