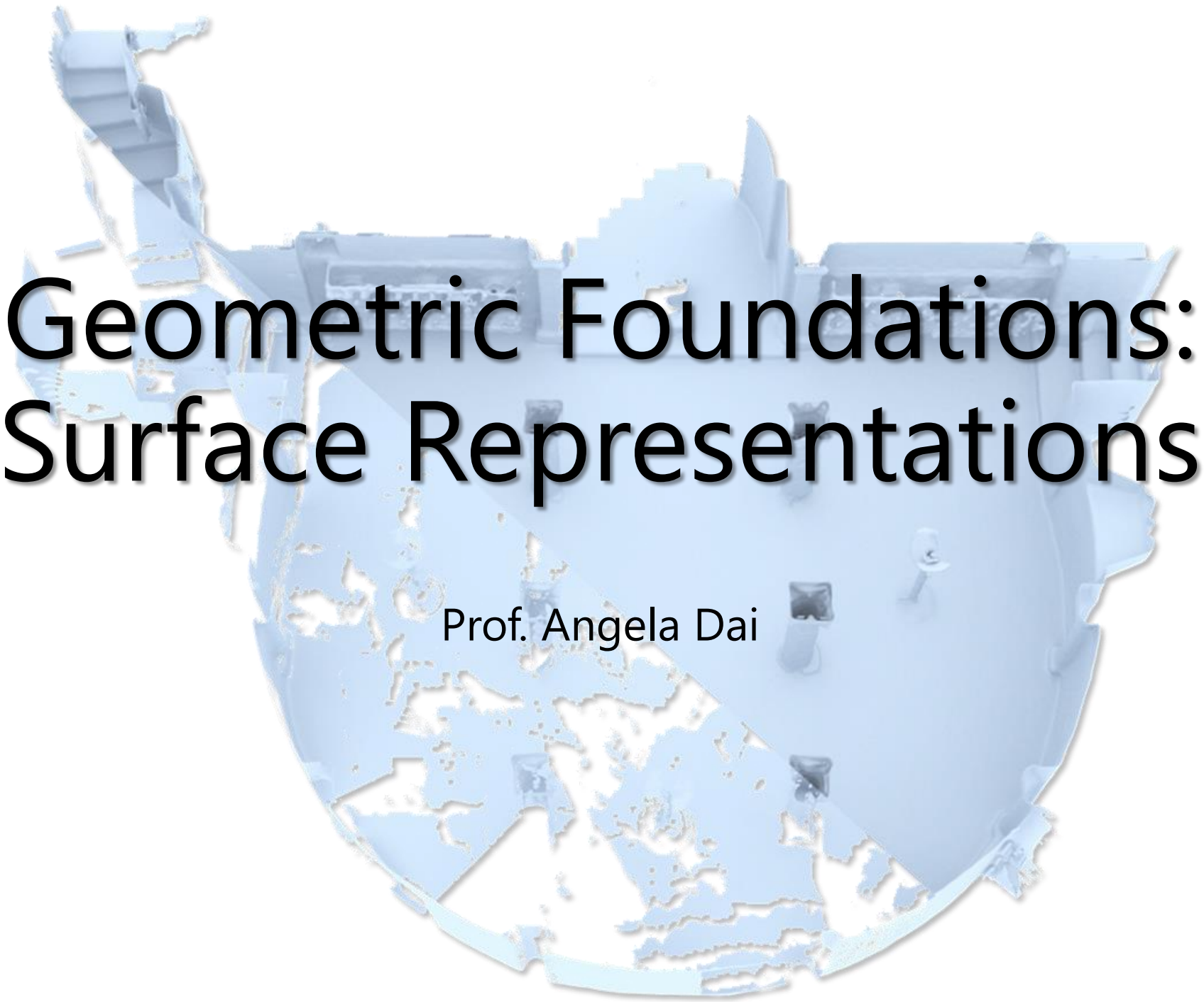
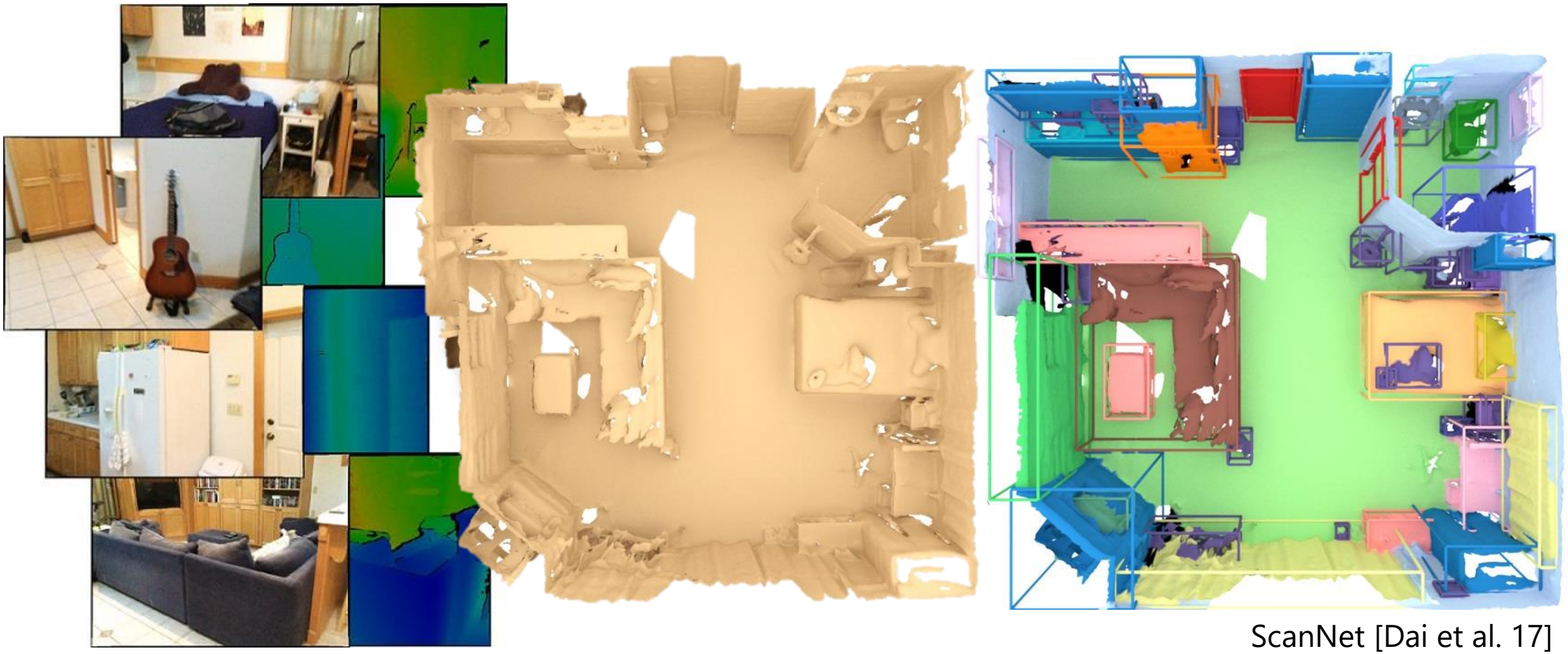


Geometric Foundations: Surface Representations

Prof. Angela Dai

Brief Recap

Machine Perception of Real-World Environments



ScanNet [Dai et al. 17]

We *perceive* and *interact* with a 3D world

ASIMO, Honda

Star Trek TNG (Phantasms)

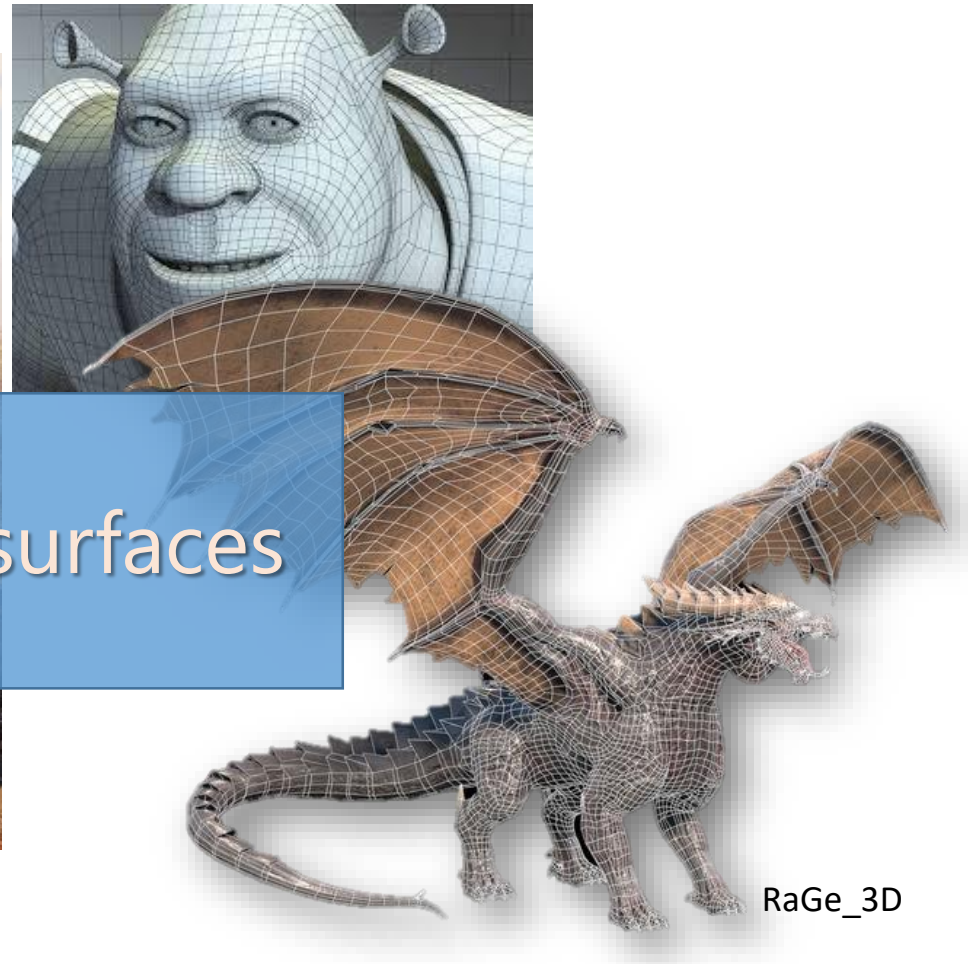
How to represent 3D?

Looking at source and target applications

Looking at source and target applications

- Movies and games

Assassin's Creed

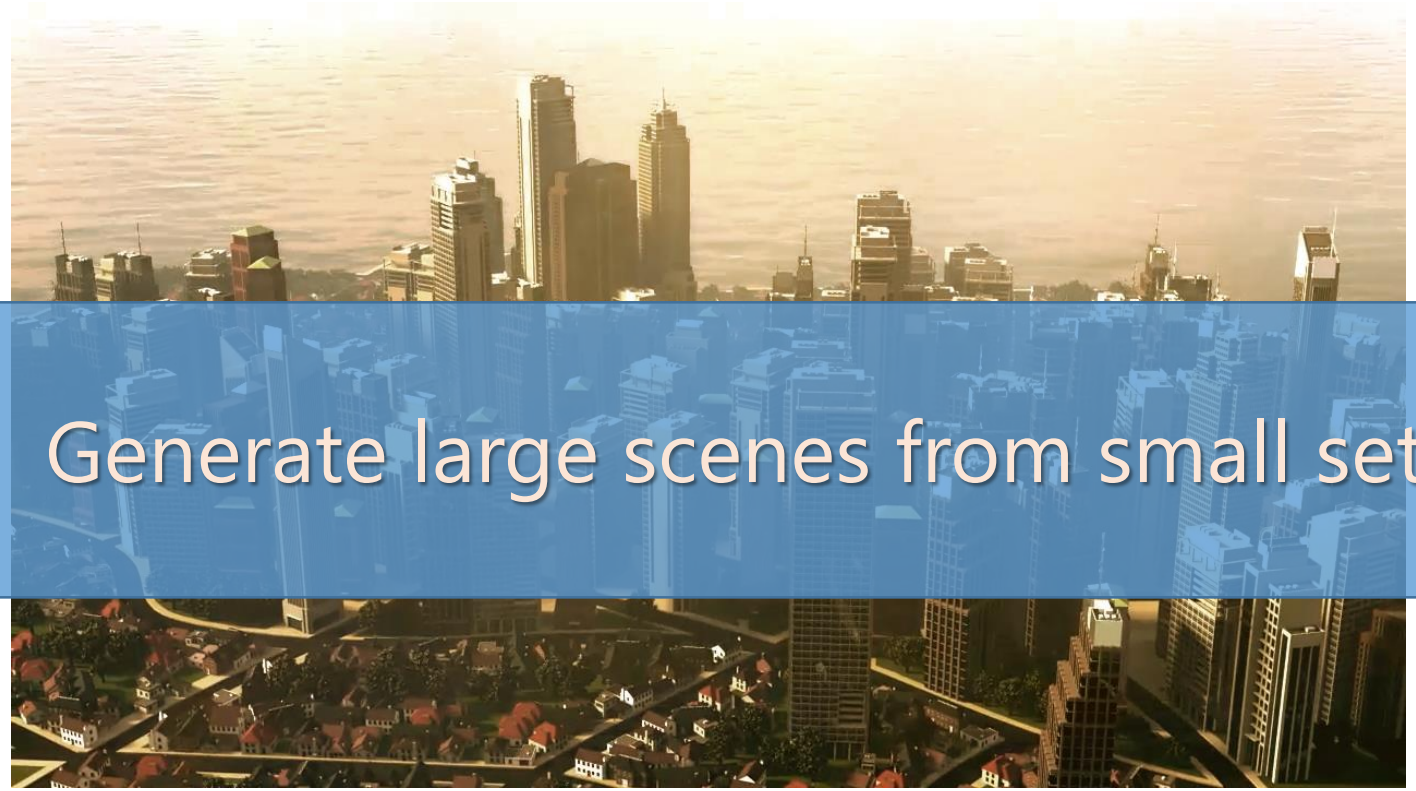


RaGe_3D

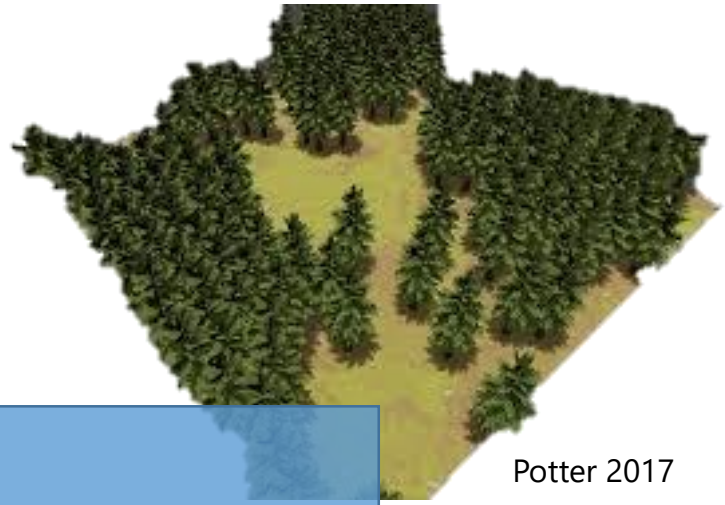
Need to be able to control and modify surfaces

Looking at source and target applications

- Movies and games: backgrounds

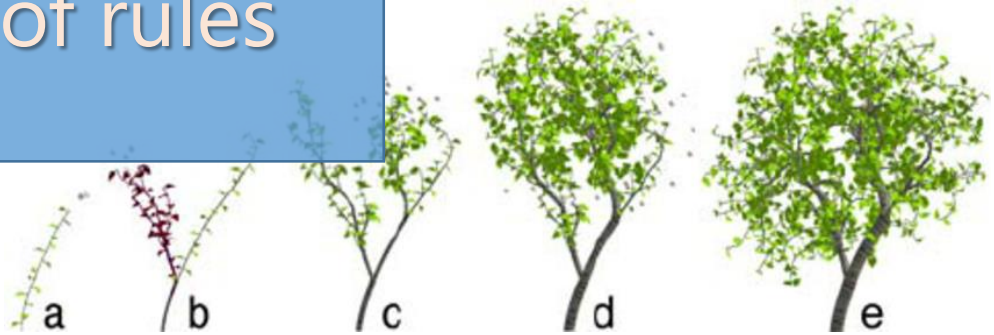


https://www.pluralsight.com/courses/houdini-python-procedural-cities



Potter 2017

Generate large scenes from small set of rules



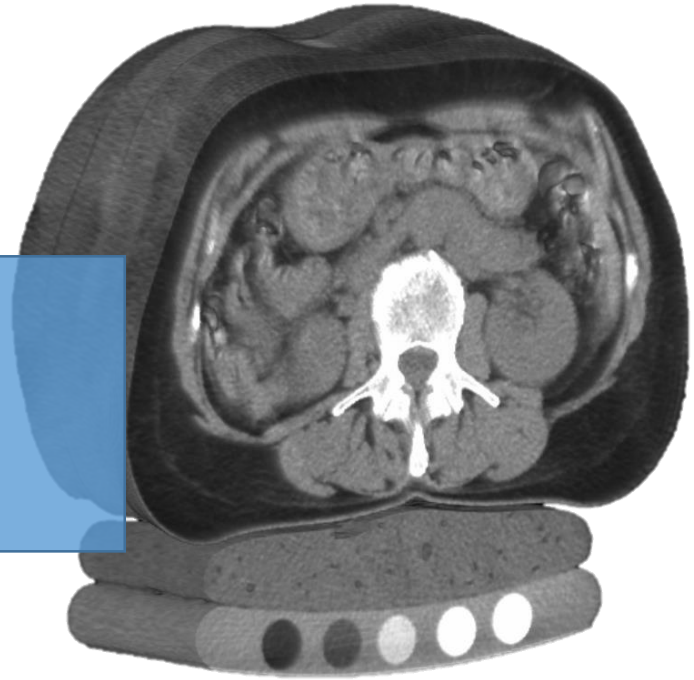
[Palubicki et al. '09]

Looking at source and target applications

- Real-world data capture

Digital Michelangelo

Waymo Open Dataset



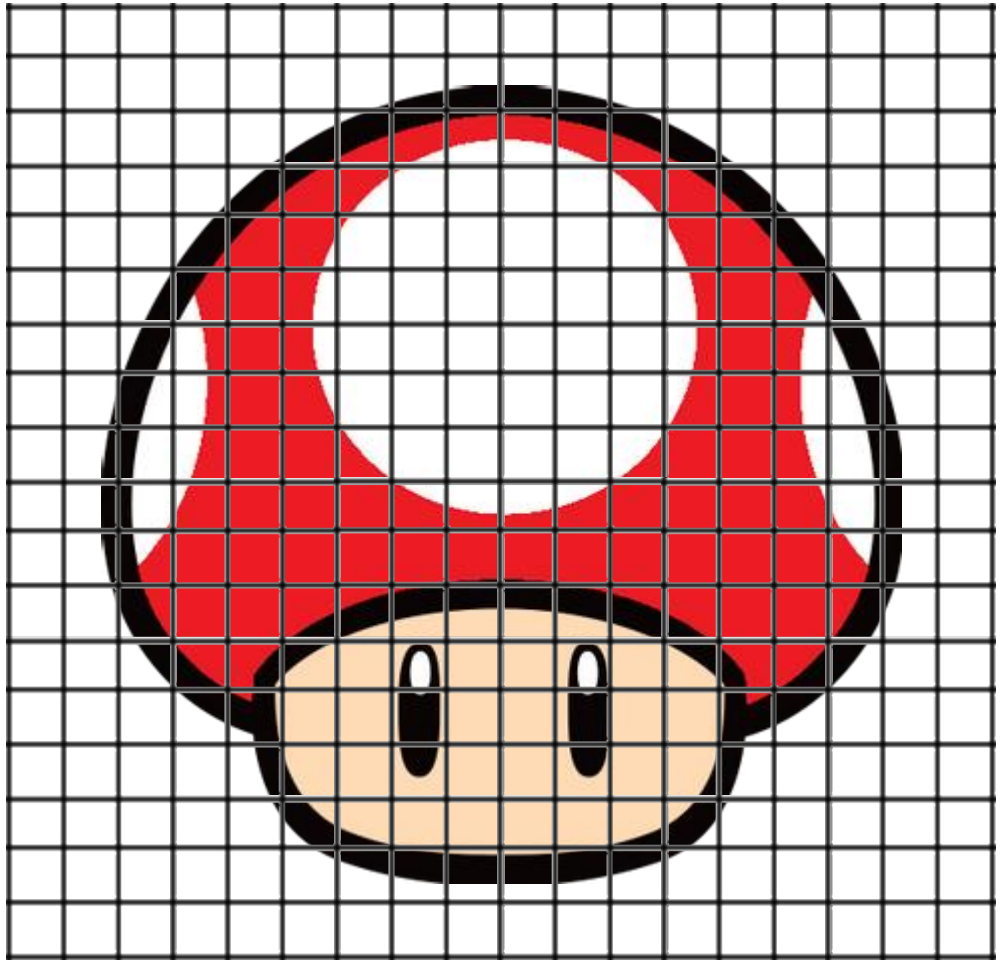
MindwaysCT Software

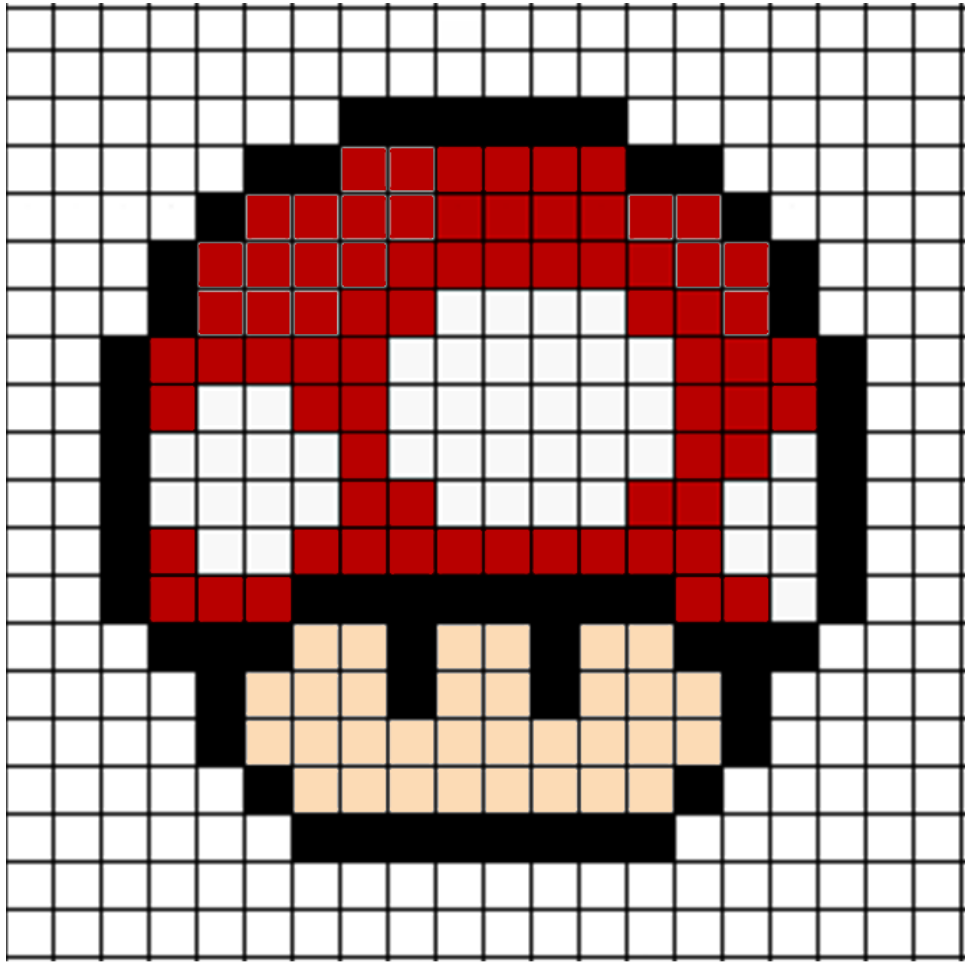
Discrete samples

Various 3D Representations

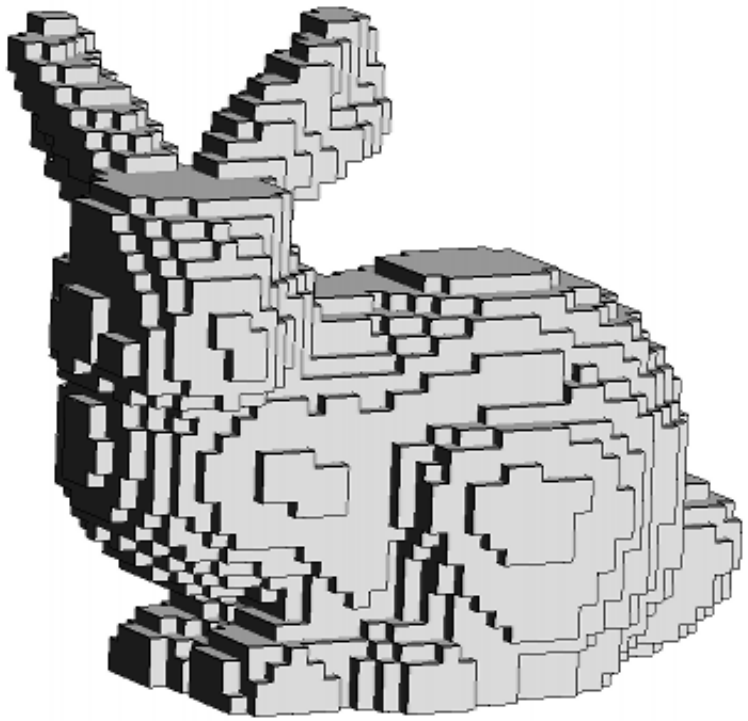
- Things to consider:
- How well does it fit with the constraints of the source data or target application?
- How much memory is necessary for storage?
- Efficiency of operations (editing, transforms, rendering, etc.)

2D Image Representation

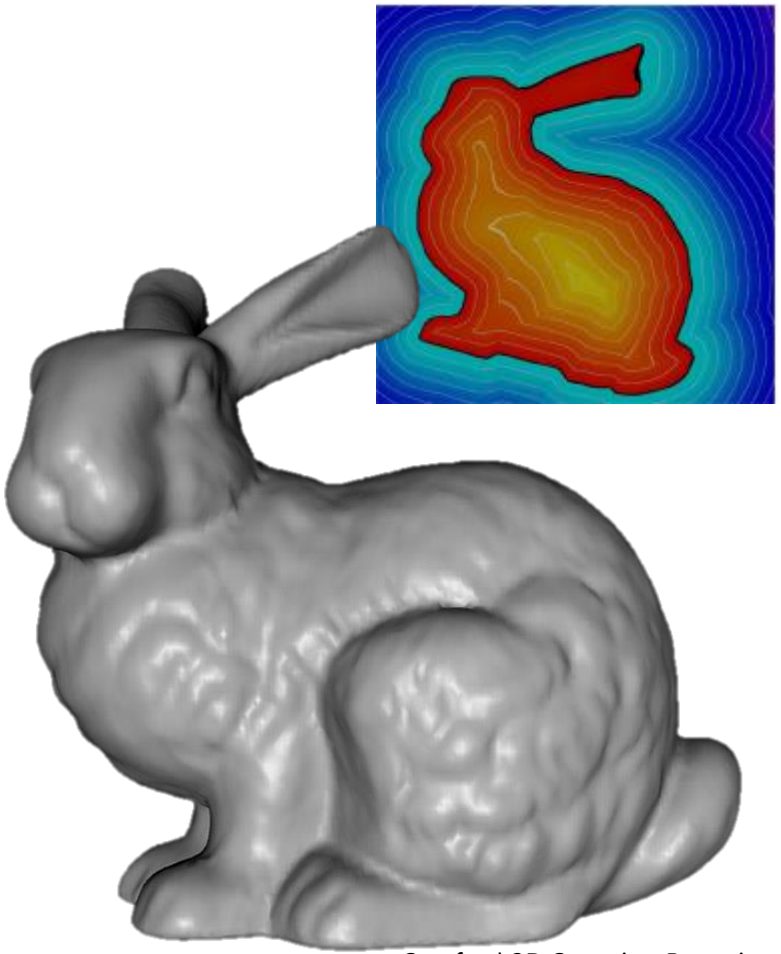




3D Voxels



sketchfab - elbriga



Stanford 3D Scanning Repository

3D Voxels

sketchfab - elbriga

- Regular grid structure (convolutions)
- Store attributes in each grid voxel
  - occupancy, distance, color, normals, etc.
- Represent arbitrary topologies
- Easy to query, operate on neighbors
- Cubic growth for high resolutions

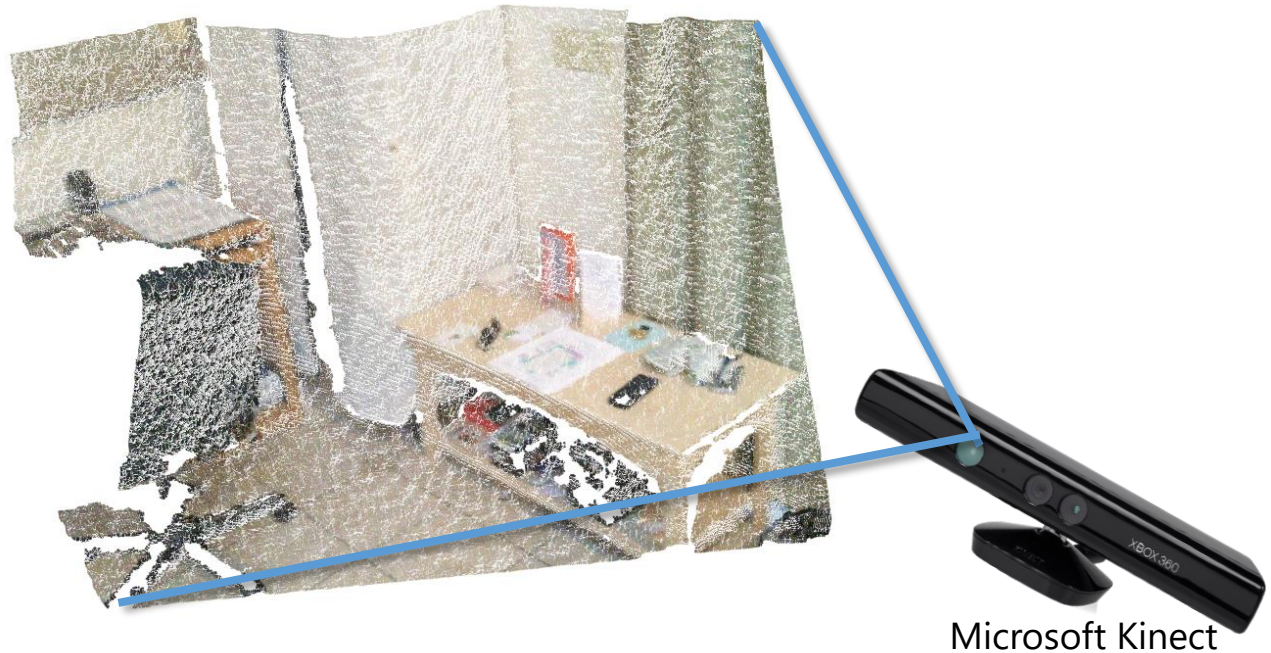
16^3: 13.1% 32^3: 7.5% 64^3: 4.2% 128^3: 2.3%

Point Clouds

- Set of points as (x,y,z) locations
- (optionally other attributes, e.g., color)

Point Clouds

- Unordered set of points as (x,y,z) locations
- (optionally other attributes, e.g., color)
- Can represent raw data capture



Microsoft Kinect

Digital Michelangelo Project

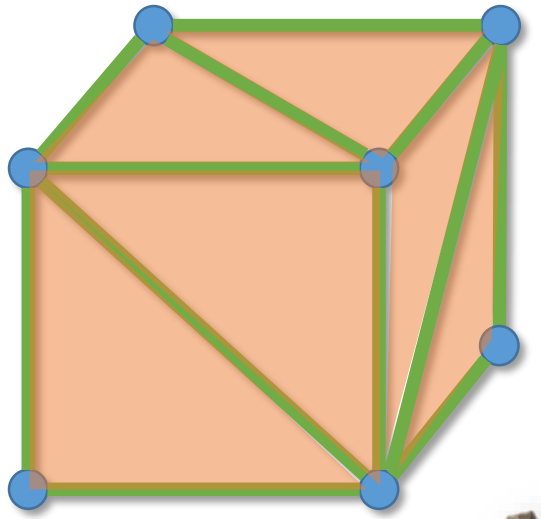
Point Clouds

- Unordered set of points as (x,y,z) locations
- (optionally other attributes, e.g., color)
- Can represent raw data capture

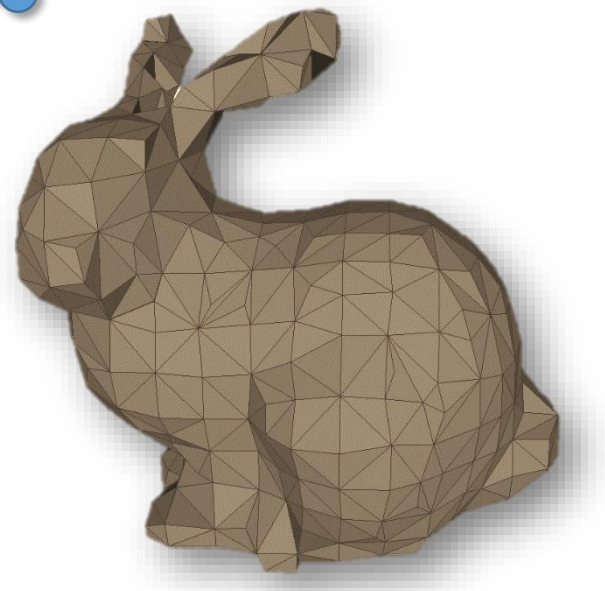
- Can more efficiently represent surface than dense grids
- No spatial structure
- Not so efficient neighbor queries

Polygon Meshes

- Collection of vertices, edges, faces
  - Often triangles

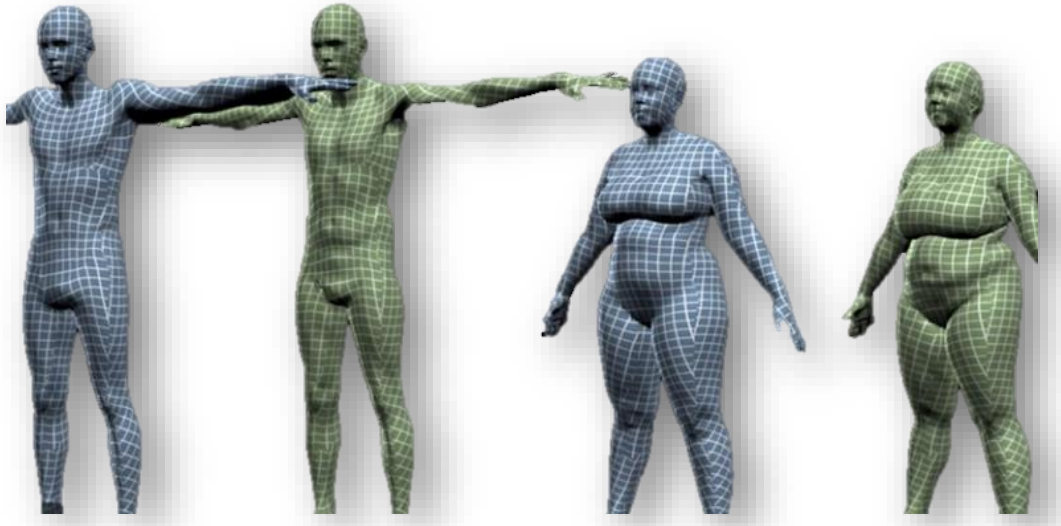


- Vertex: 3D position (x, y, z)
- Edge: connection between two vertices
- Face: closed set of edges



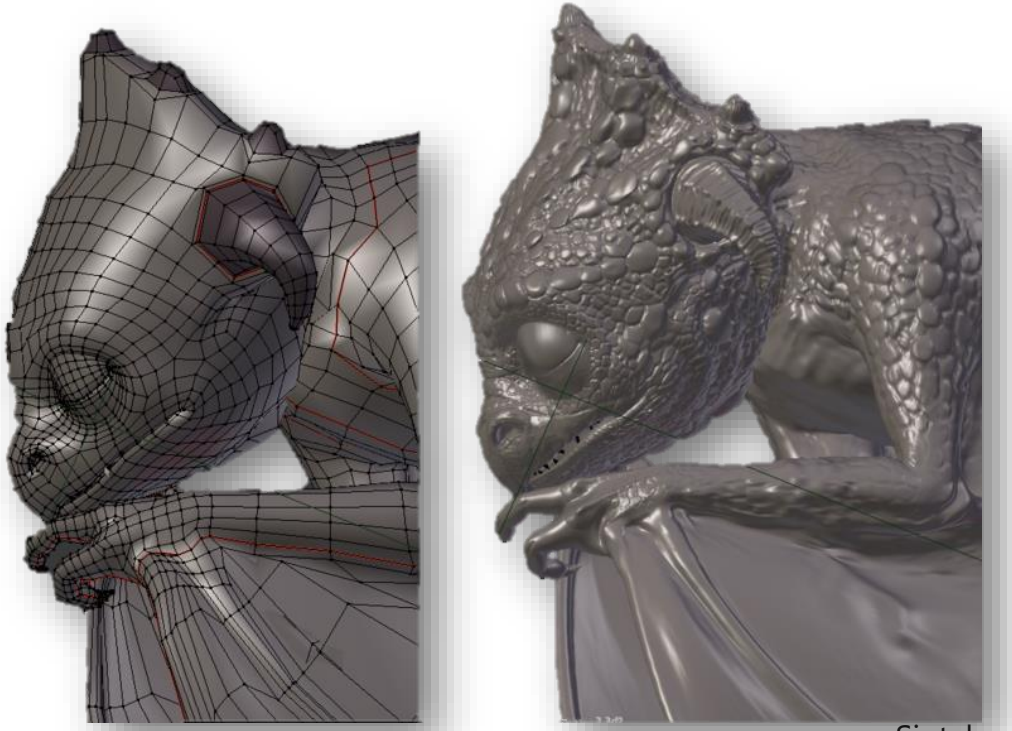
Polygon Meshes

- Many applications



Template Models

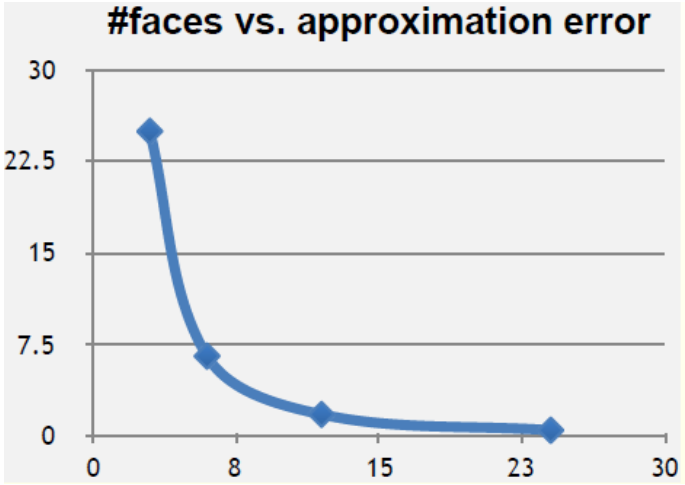
Content Creation: Movies, Games, Mixed Reality

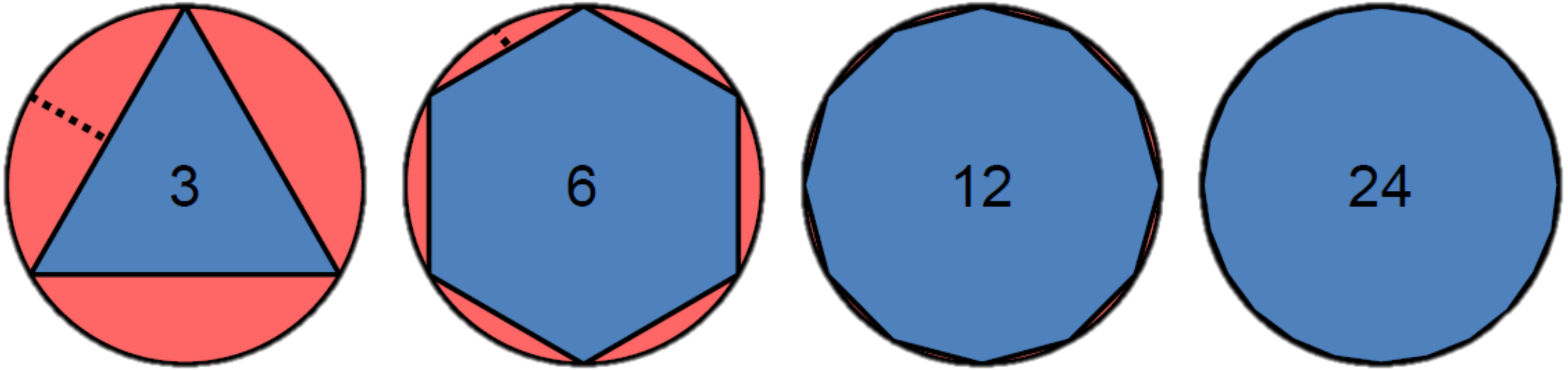


Sintel

Polygon Meshes

- Piecewise linear approximation of a surface
  - $O(h^2)$ error

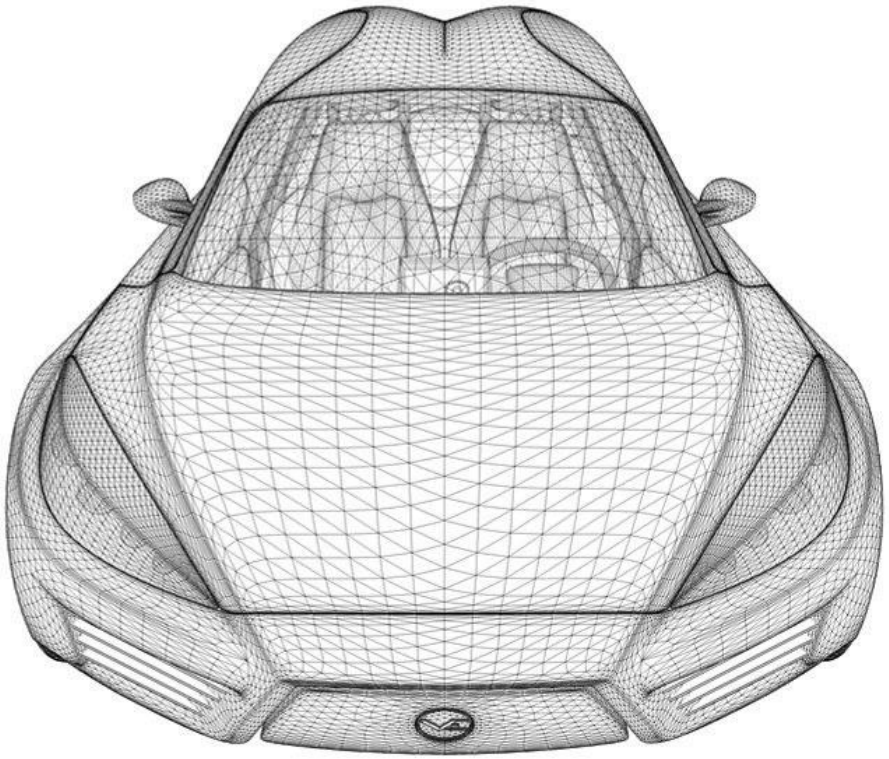




25% 6.5% 1.7% 0.4%

Polygon Meshes

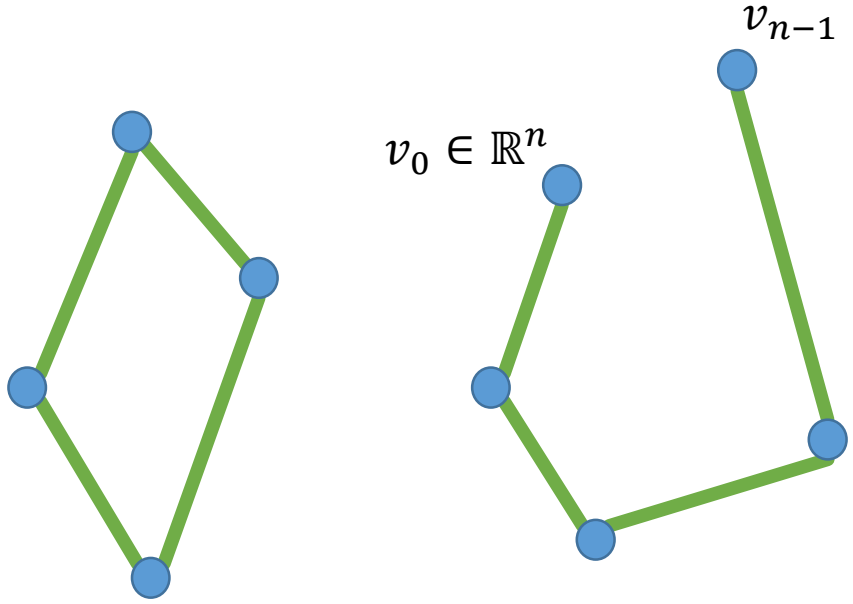
- Popular representation
- Can represent arbitrary topologies
- Can be edited/manipulated
- Efficient rendering



Polygon

- Vertices: $v_0, v_1, \dots, v_{n-1}$
- Edges: $\{(v_0, v_1), \dots, (v_{n-2}, v_{n-1})\}$

- Closed: $v_0 = v_{n-1}$
- Planar: all vertices on a plane
- Simple: not self-intersecting

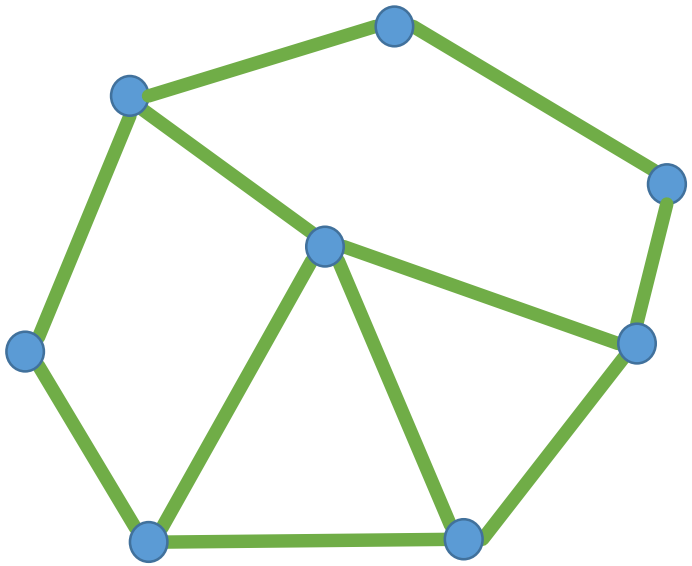


Polygon Mesh

- Polygon Mesh: finite set M of closed, simple polygons P_i

$$M = \langle V, E, F \rangle$$

- Intersection of two polygons in M is either empty, a vertex, or an edge



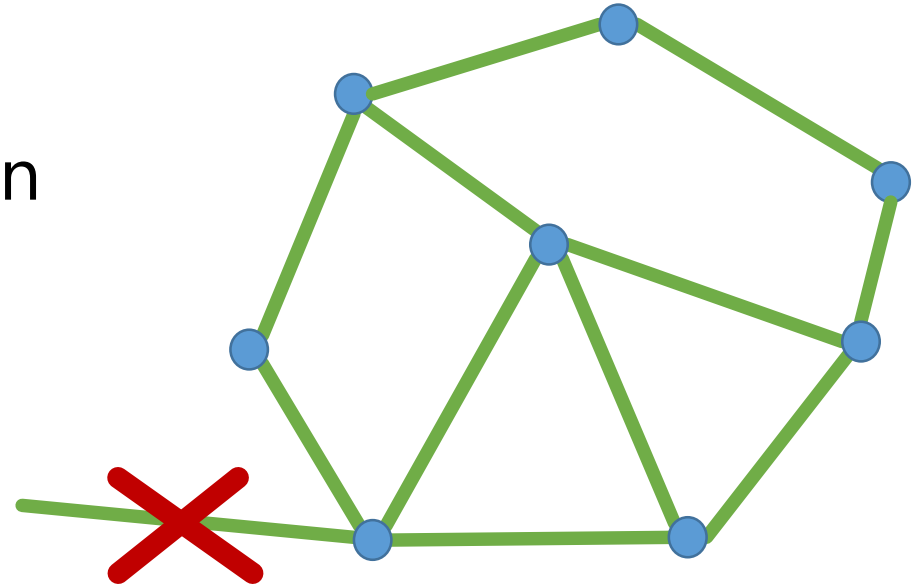
Polygon Mesh

- Polygon Mesh: finite set M of closed, simple polygons P_i

$$M = \langle V, E, F \rangle$$

- Intersection of two polygons in M is either empty, a vertex, or an edge

- Every edge belongs to at least one polygon

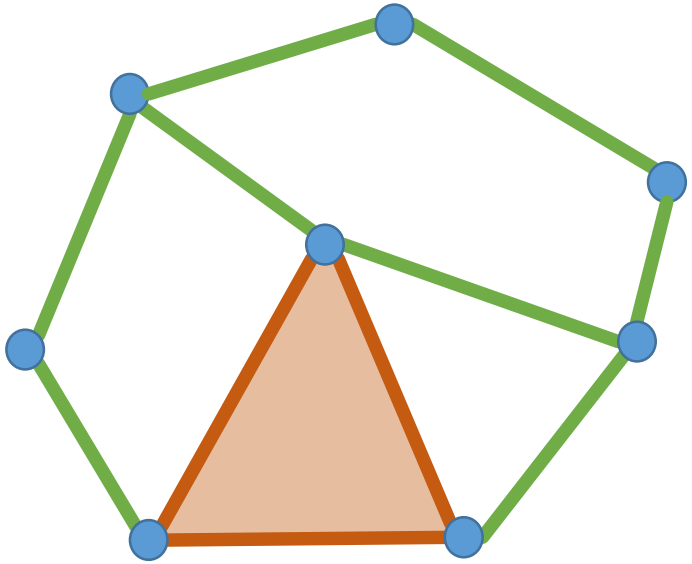


Polygon Mesh

- Polygon Mesh: finite set M of closed, simple polygons P_i

$$M = \langle V, E, F \rangle$$

- Intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each P_i defines a *face* of the polygonal mesh

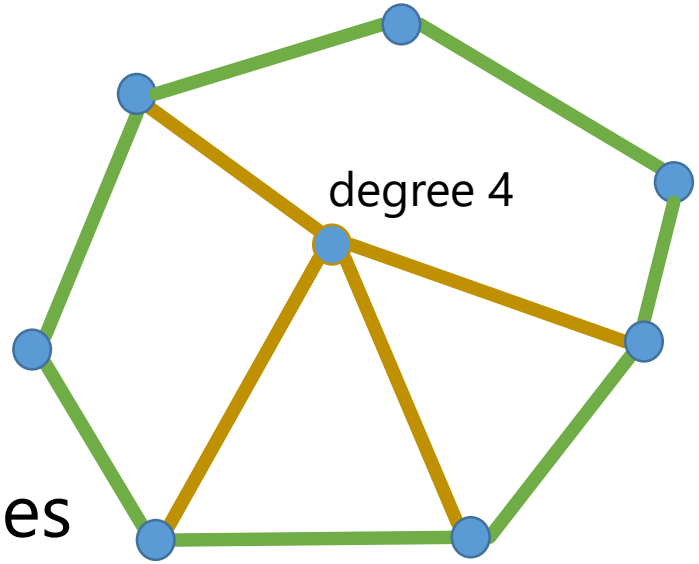


Polygon Mesh

- Polygon Mesh: finite set M of closed, simple polygons P_i

$$M = \langle V, E, F \rangle$$

- Intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each P_i defines a *face* of the polygonal mesh
- Vertex *degree* or *valence*: number of incident edges

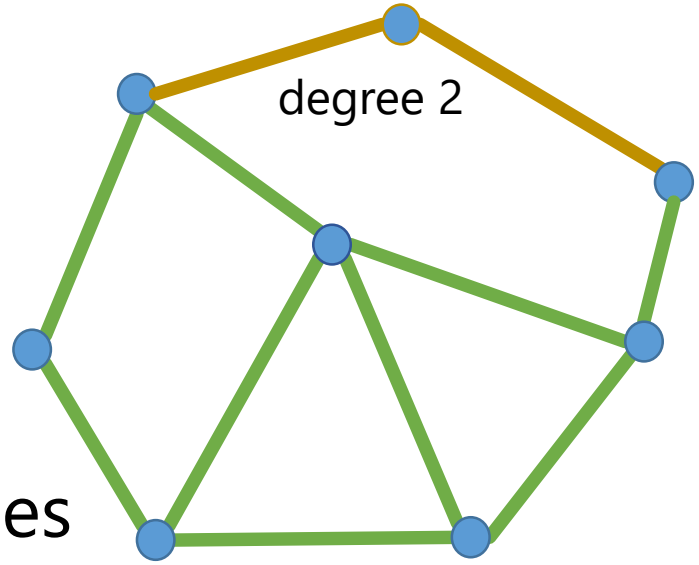


Polygon Mesh

- Polygon Mesh: finite set M of closed, simple polygons P_i

$$M = \langle V, E, F \rangle$$

- Intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each P_i defines a *face* of the polygonal mesh
- Vertex *degree* or *valence*: number of incident edges

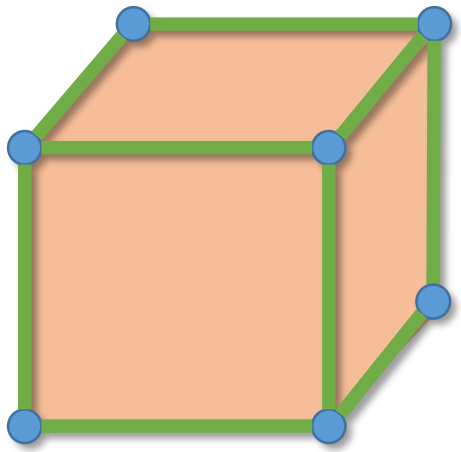


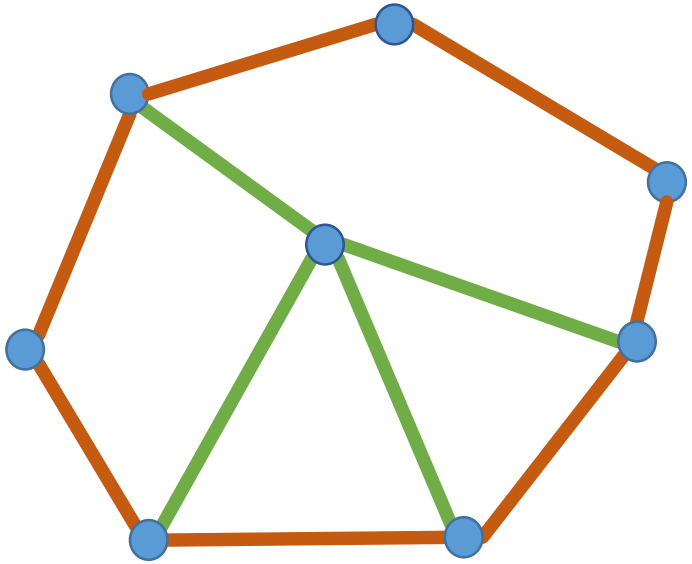
Polygon Mesh

- Polygon Mesh: finite set M of closed, simple polygons P_i

$$M = \langle V, E, F \rangle$$

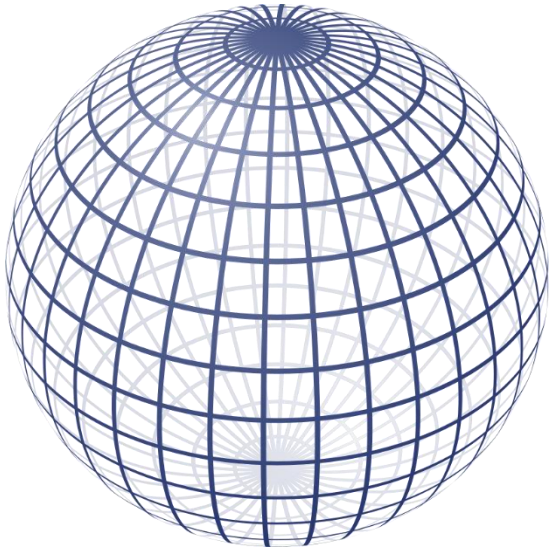
- Boundary: set of all edges that belong only to one polygon
  - Either empty or forms closed loops
  - If empty, then the polygon mesh is closed

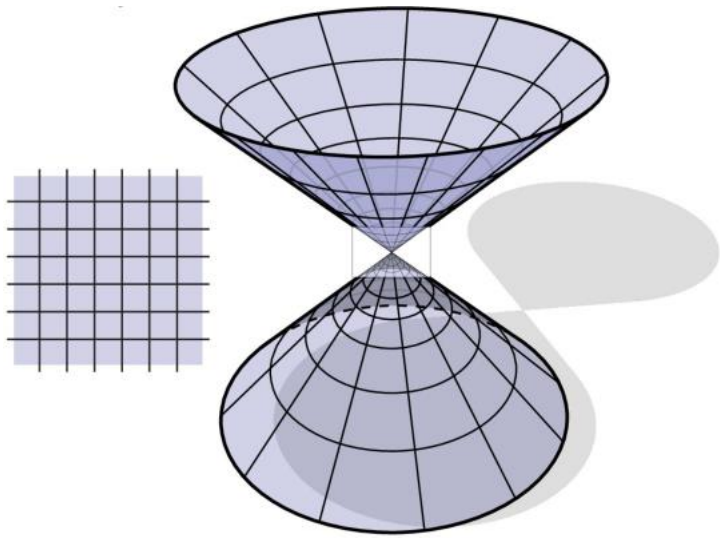




Manifold Geometry

- Recall: we aim to represent surfaces, e.g., “boundary” of an object
- Surfaces are manifold: locally Euclidean
  - i.e., around every point, there is a neighborhood topologically the same as the open unit ball in $\mathbb{R}^n$





Manifold vs Non-Manifold

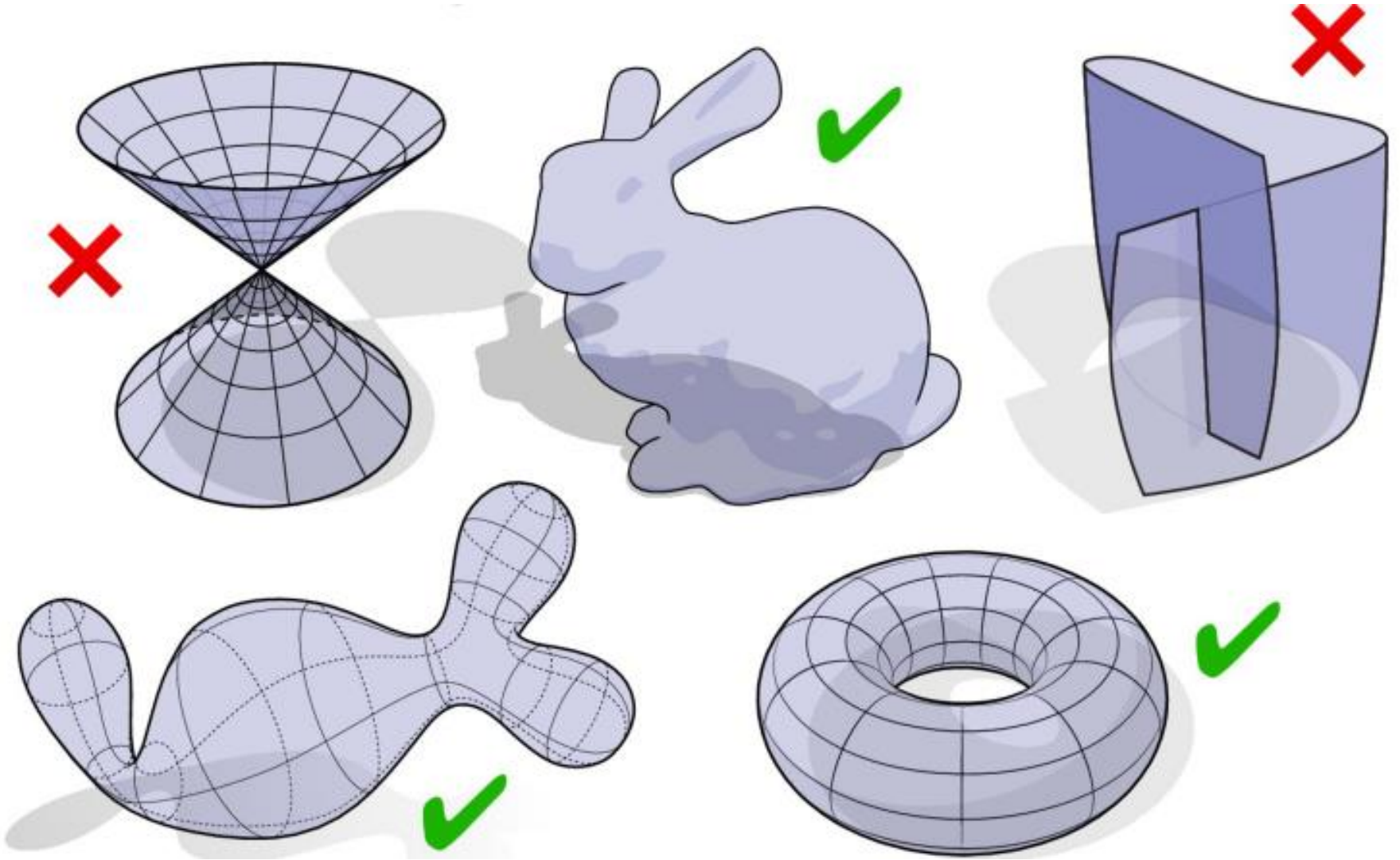


Image Credit: Keenan Crane, CMU 15-462/662

Manifold Polygon Mesh

- Each edge is contained in only two polygons
- The polygons containing each vertex make a single “fan”

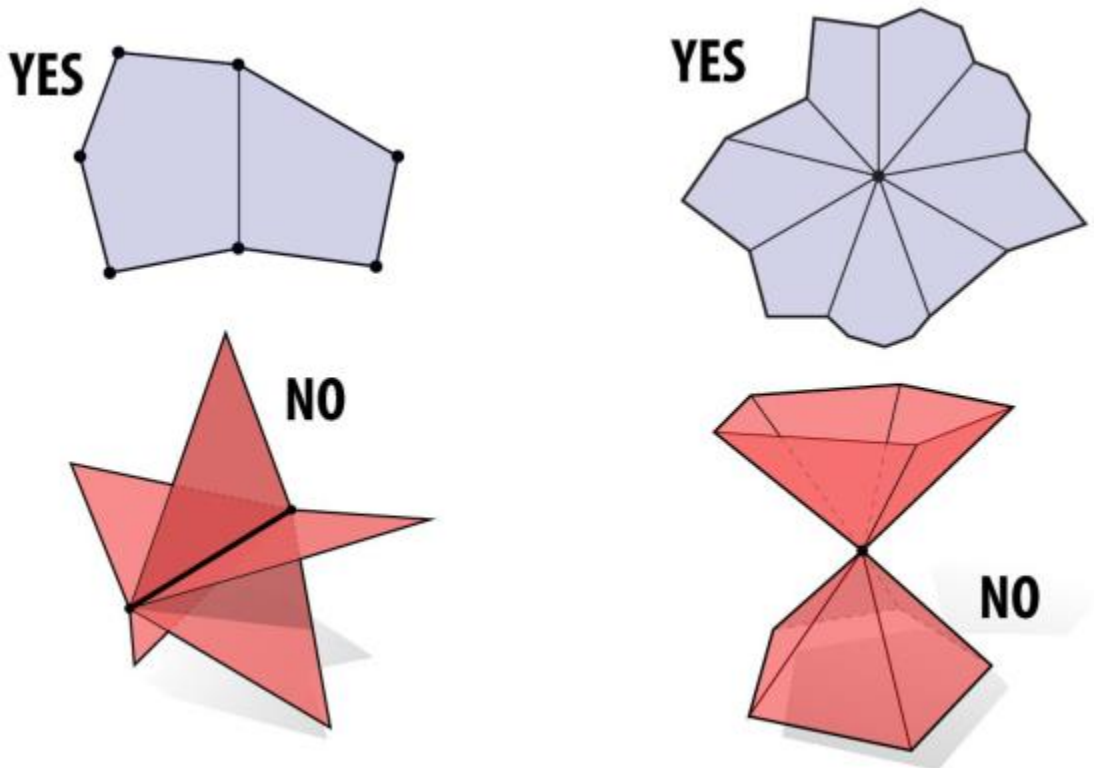
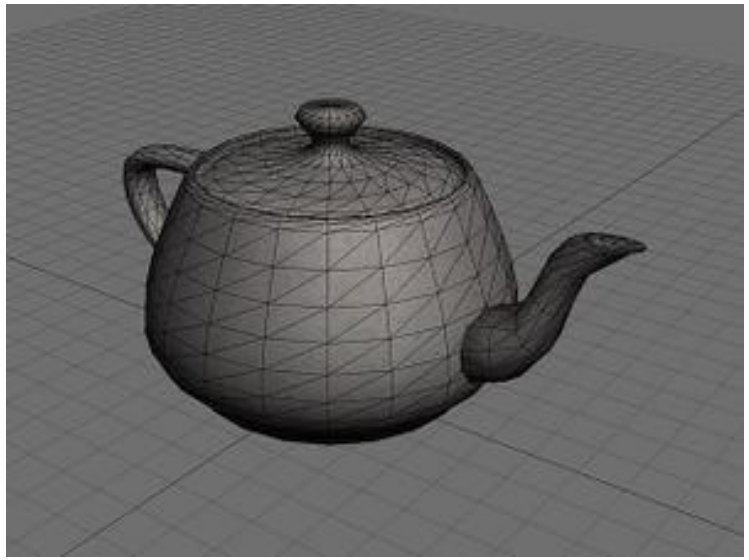
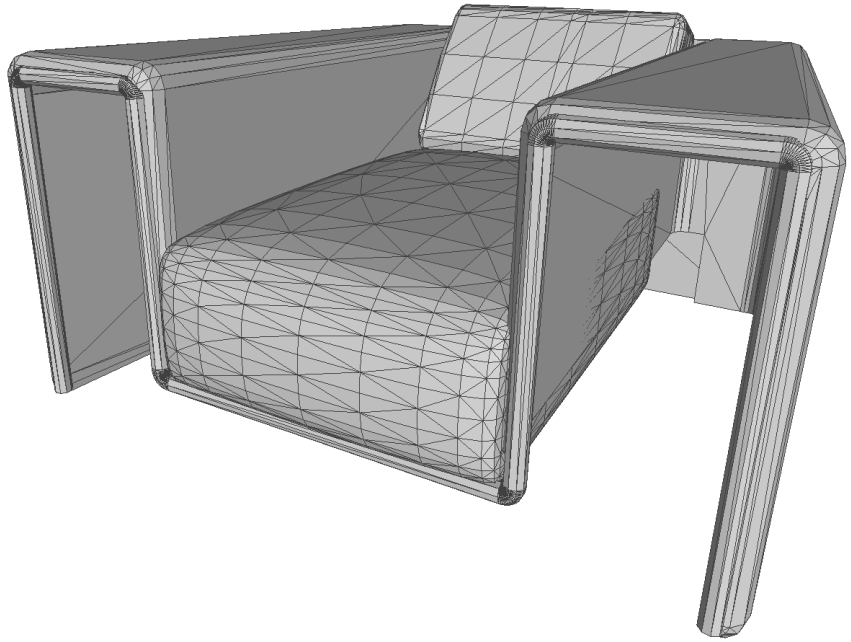


Image Credit: Keenan Crane, CMU 15-462/662

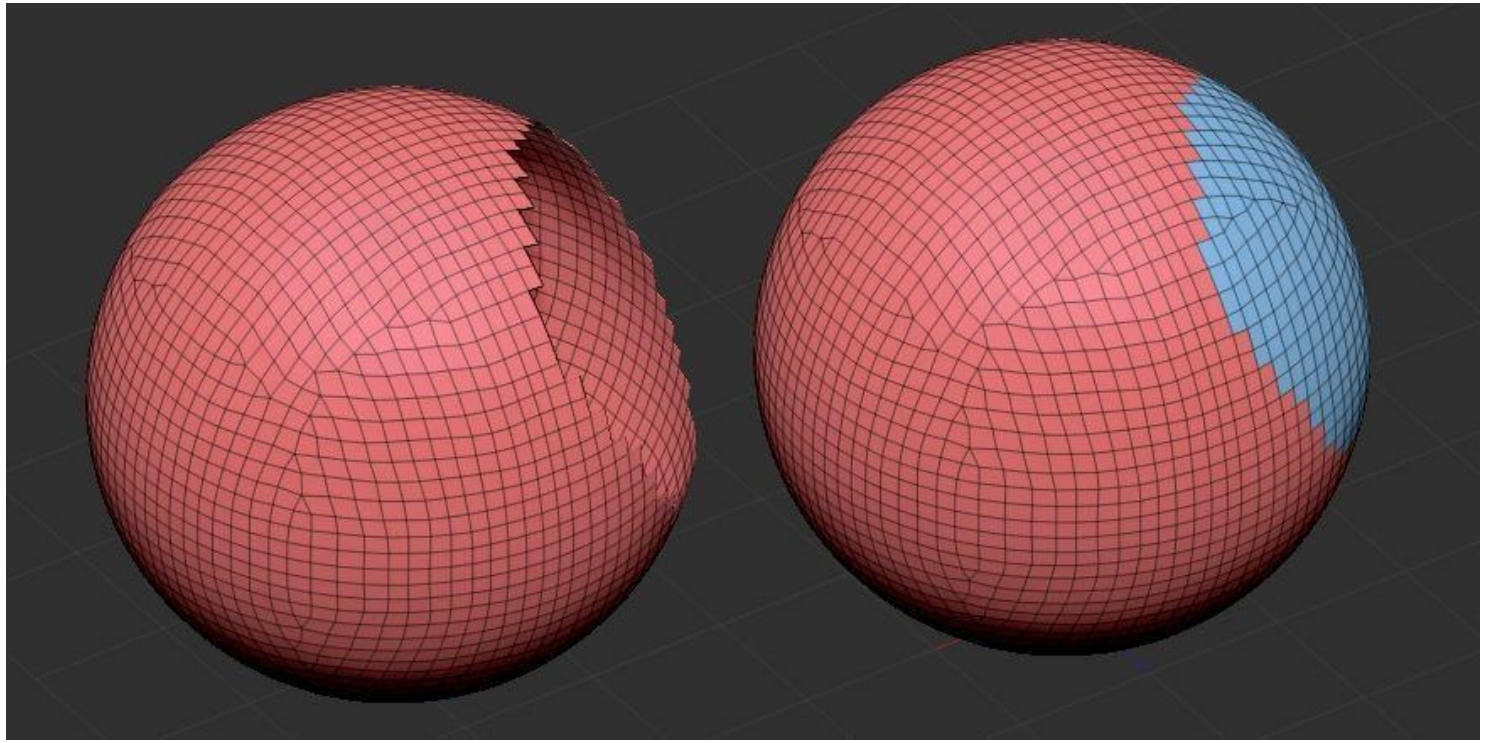
Manifold Assumption





Watertight Mesh

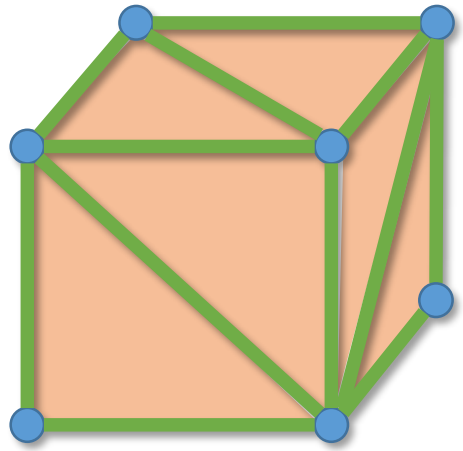
- Does not have holes
- Has a designated inside and outside



Triangle Mesh

- Polygon mesh where every face is a triangle

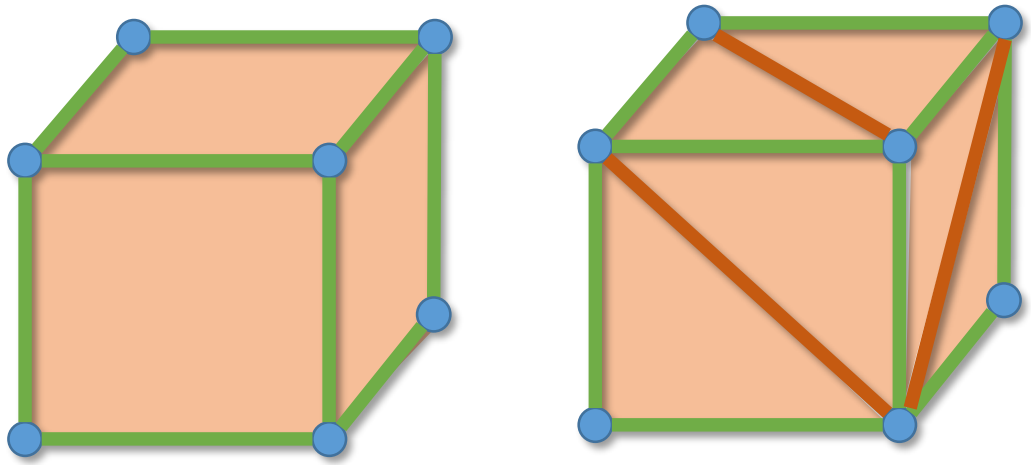
- Popular Representation
- Simplifies data structures
- Simplifies rendering
- Simplifies algorithms
- Each face is planar and convex
- Any polygon can be triangulated



Triangle Mesh

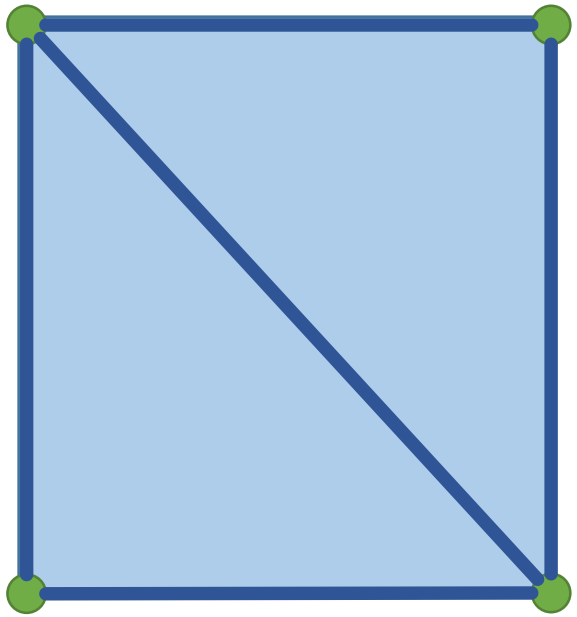
- Polygon mesh where every face is a triangle

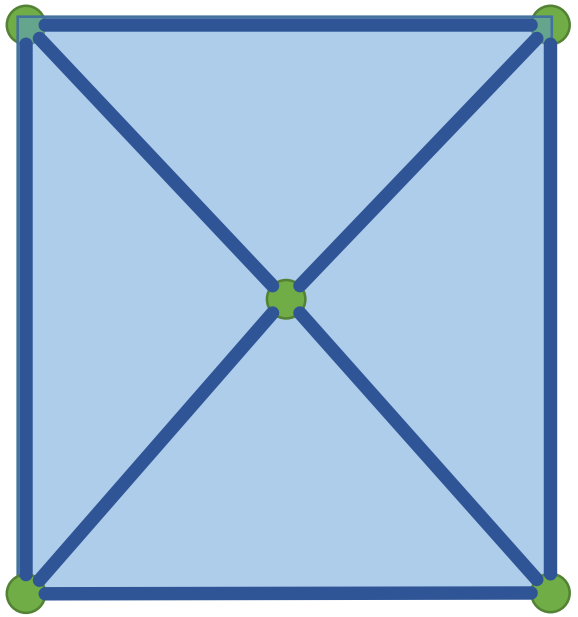
- Popular Representation
- Simplifies data structures
- Simplifies rendering
- Simplifies algorithms
- Each face is planar and convex
- Any polygon can be triangulated



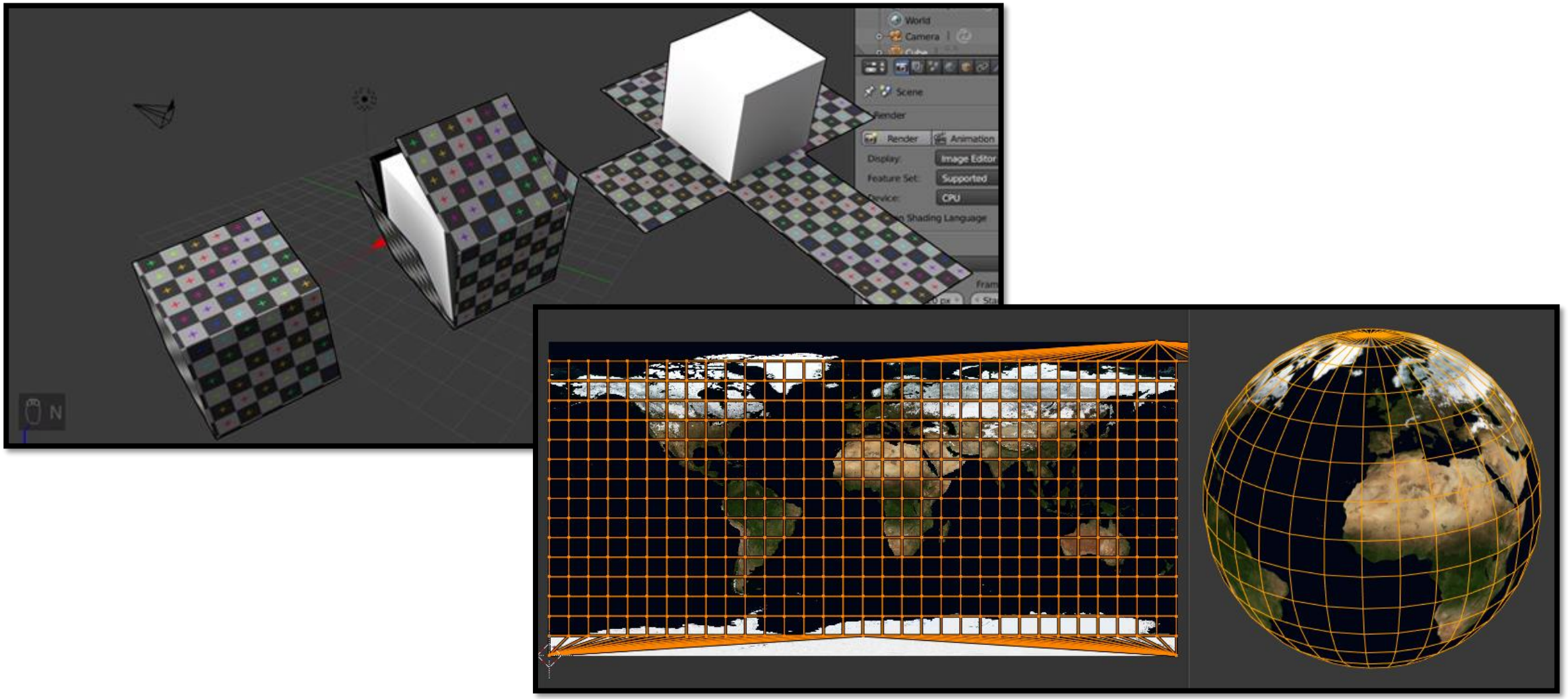
Polygon Meshes

- Many different ways to describe the same surface





Textured Polygon Meshes



Polygon Meshes – Data Structure

- What should be stored?

- Geometry: vertices, edges, faces
- Connectivity
  - adjacency relationships
- Attributes
  - normals, color, texture coordinates
  - per-vertex, per-face, per-edge

Polygon Meshes – Simple Data Structures

- Triangle List
  - Face: 3 positions
  - 4 bytes per coordinate
  - 36 bytes per face
- No connectivity information
- STL format

Triangles			
0	x0	y0	z0
1	x1	y1	z1
2	x2	y2	z2
3	x3	y3	z3
4	x4	y4	z4
5	x5	y5	z5
…	…	…	…

Polygon Meshes – Simple Data Structures

- Indexed Face Set
  - Vertex position
  - Face: vertex indices
  - 12 bytes per vertex
  - 12 bytes per face

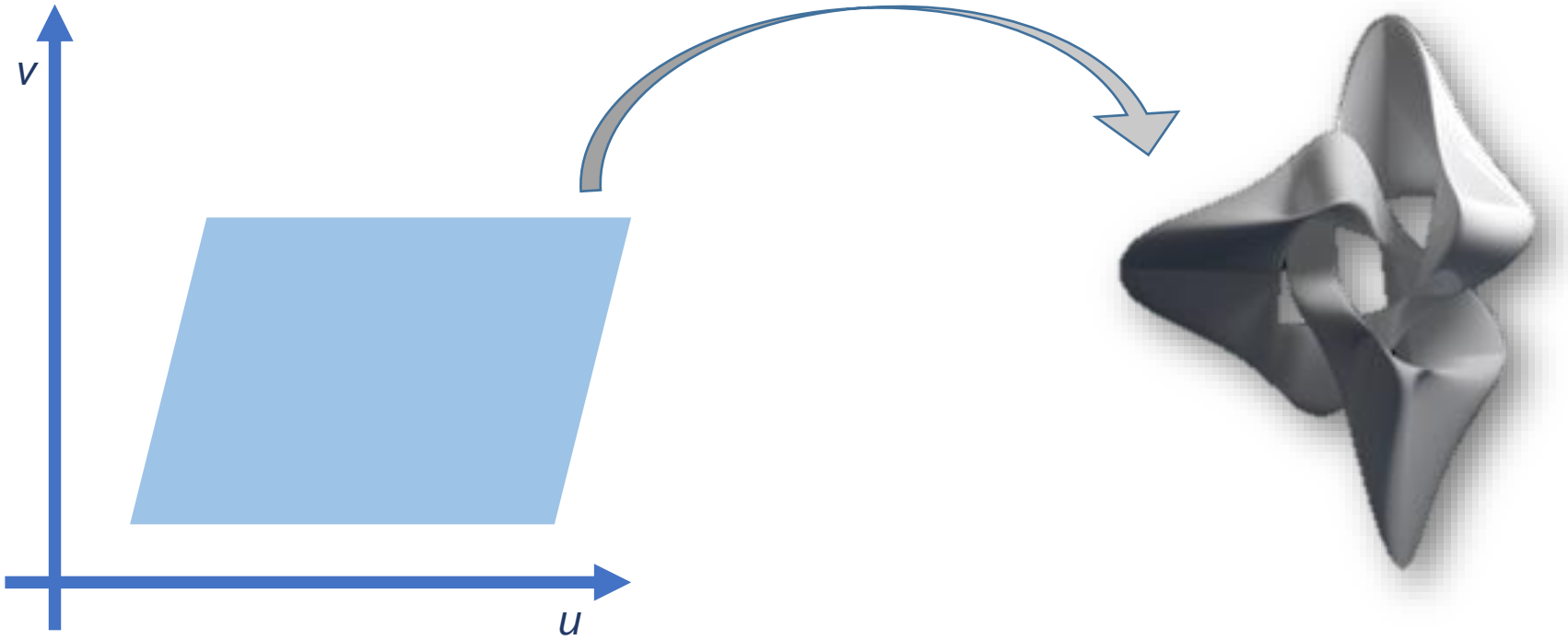
Triangles			
0	v0	v1	v2
1	v0	v1	v3
2	v2	v4	v3
3	v5	v2	v6
…	…	…	…

Vertices			
v0	x0	y0	z0
v1	x1	y1	z1
v2	x2	y2	z2
v3	x3	y3	z3
v4	x4	y4	z4
v5	x5	y5	z5
…	…	…	…

- No explicit neighborhood information

- OBJ, OFF, WRL formats

Parametric Surfaces



$$\boldsymbol{p}\colon \mathbb{R}^2 \to \mathbb{R}^3$$

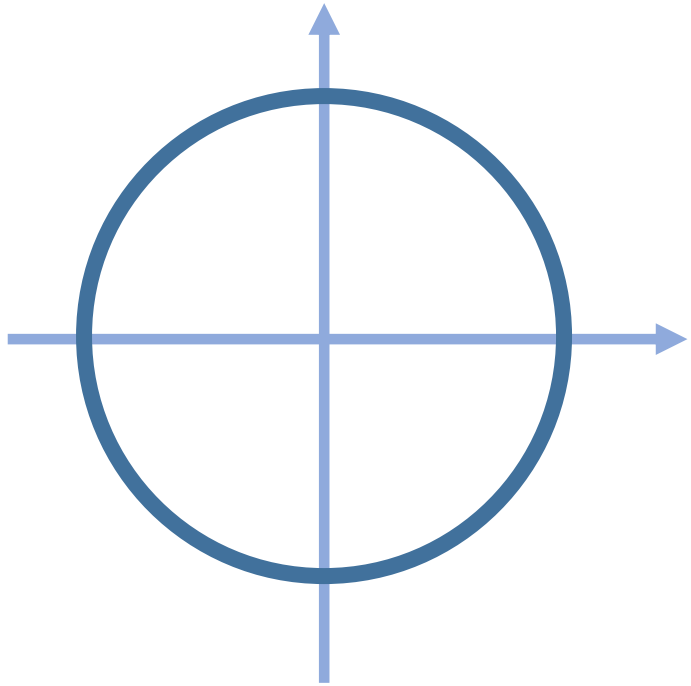
$$\boldsymbol{p}(u,v) = (x(u,v), y(u,v), z(u,v))$$

Parametric Curves

$$\boldsymbol{p}: \mathbb{R} \to \mathbb{R}^2$$
$$t \mapsto \boldsymbol{p}(t) = (x(t), y(t))$$

Circle:

$$\boldsymbol{p}(t) = r\big(cos(t), sin(t)\big)$$
$$t \in [0, 2\pi)$$

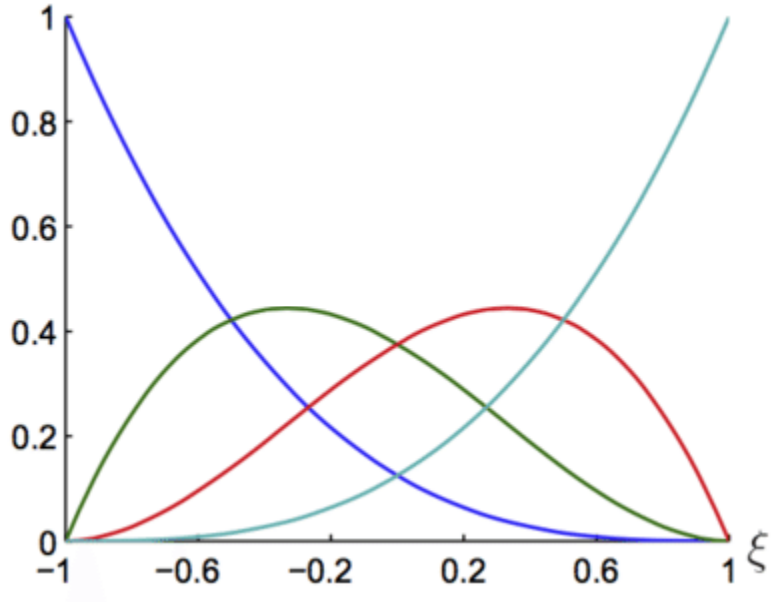


Parametric Curves

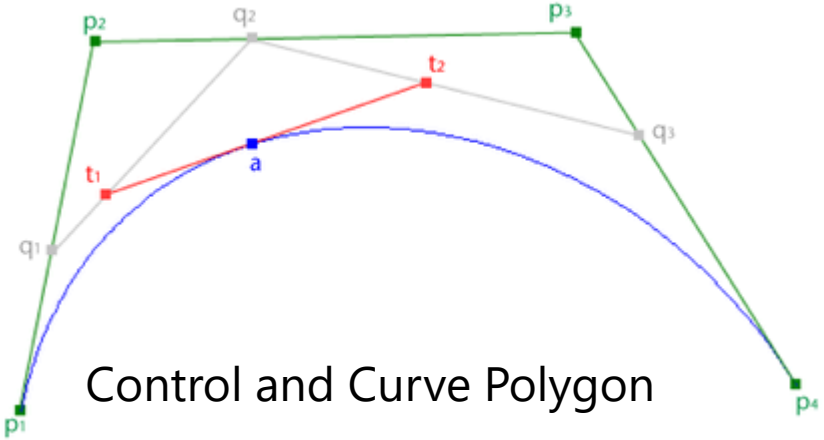
- Bezier curves, splines

$$s(t) = \sum_{i=0}^{n} \boldsymbol{p}_i B_i^n(t) \qquad \sum_{i=0}^{n} B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

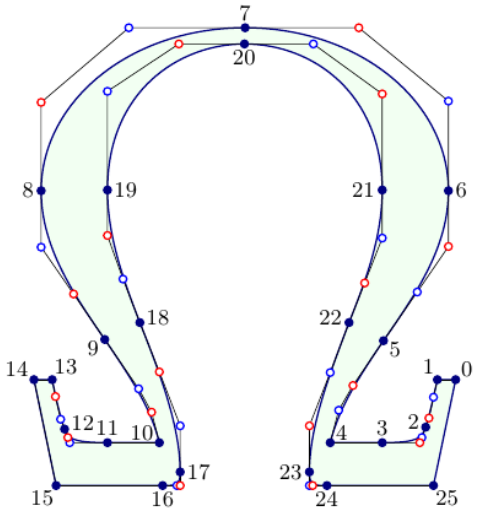
Control Points



Cubic Bezier Basis Functions



Control and Curve Polygon



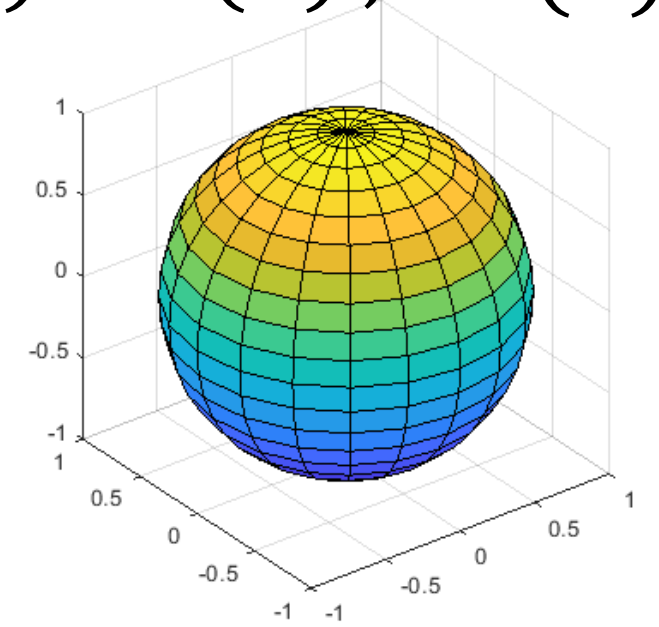
Parametric Surfaces

- Example: Sphere

$$\boldsymbol{p}: \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\boldsymbol{p}(u,v) = r(cos(u)\cos(v), sin(u)\cos(v), \sin(v))$$

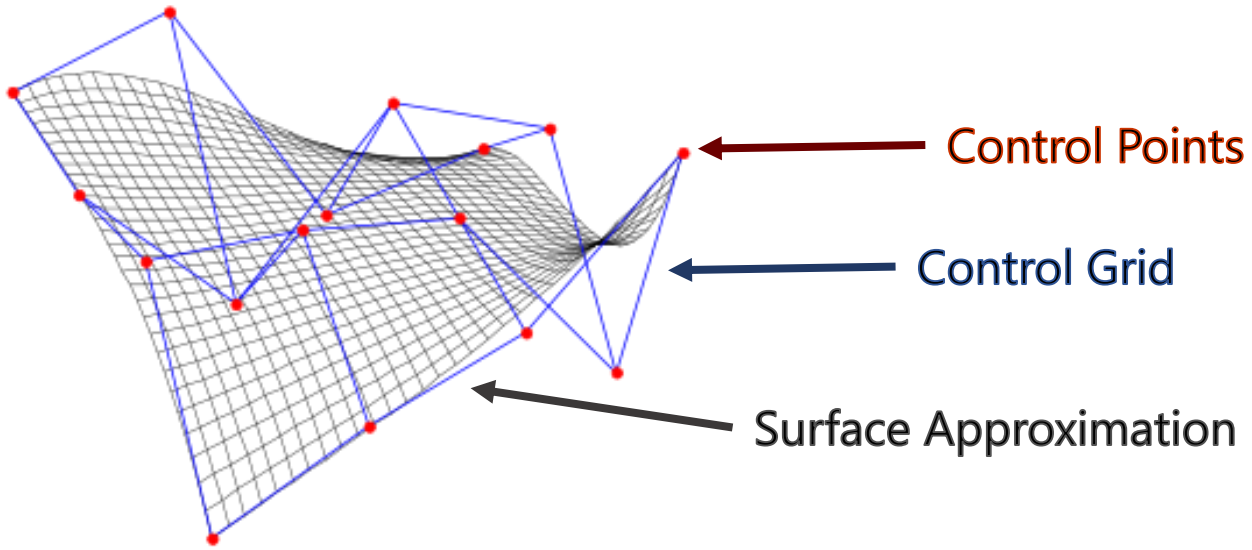
$$(u,v) \in [0,2\pi) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$



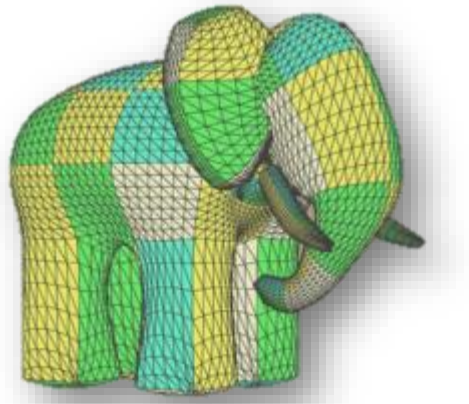
Parametric Surfaces

- Bezier Surface
  - $(n+1)(m+1)$ control points $\boldsymbol{p}_{i,j}$

$$s(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \boldsymbol{p}_{i,j}\, B_i^n(u) B_j^m(v)$$



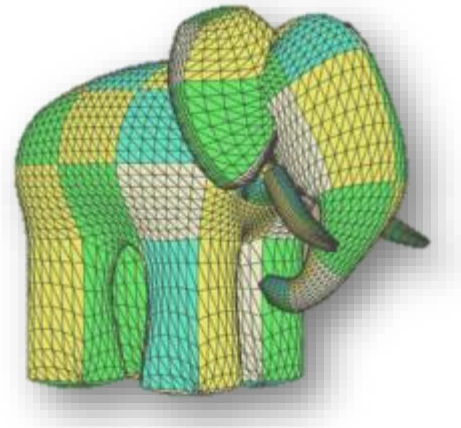
Bicubic Patches (n=3, m=3)



Gumbo, Ed Catmull

Parametric Surfaces

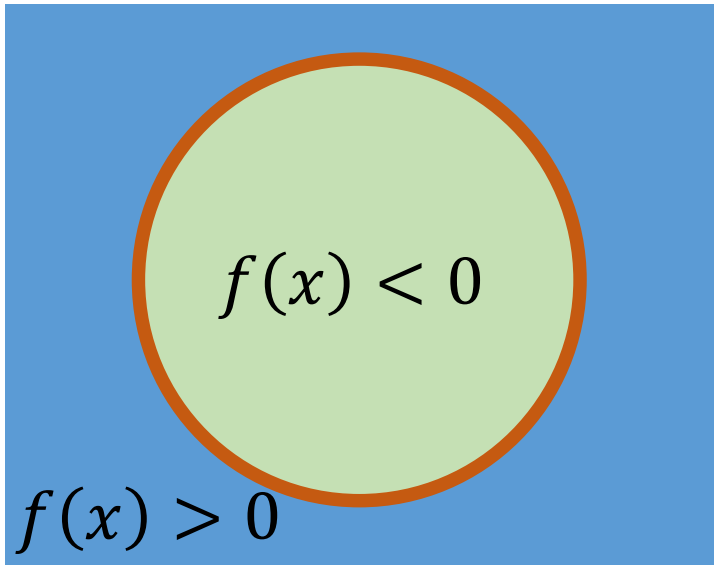
- Bezier Patch Meshes
  - Requires fewer points to represent a surface than a triangle mesh
  - Easy to manipulate
    - E.g., transforms the same way as control points under linear transformations
  - Easy to generate points on the surface
  - Easier to ensure continuity
  - Difficult to determine inside/outside
  - Difficult to determine if point is on the surface
  - Rendering more complex –
    must calculate intersections with lines



Implicit Surfaces

$$f: \mathbb{R}^m \to \mathbb{R}$$

Surface in 3D: $S = \{x \in \mathbb{R}^3 \mid f(x) = 0\}$



$\{x \in \mathbb{R}^m \mid f(x) > 0\}$ Outside

$\{x \in \mathbb{R}^m \mid f(x) = 0\}$ Surface

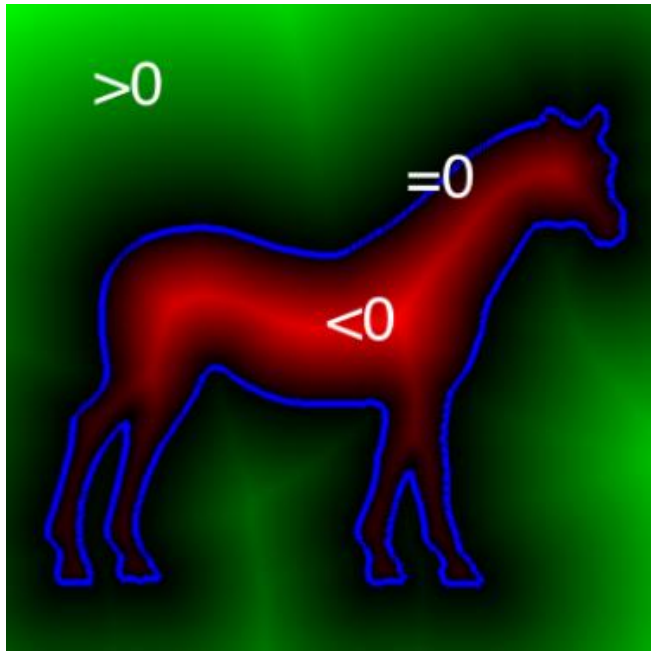
$\{x \in \mathbb{R}^m \mid f(x) < 0\}$ Inside

Implicit Surfaces

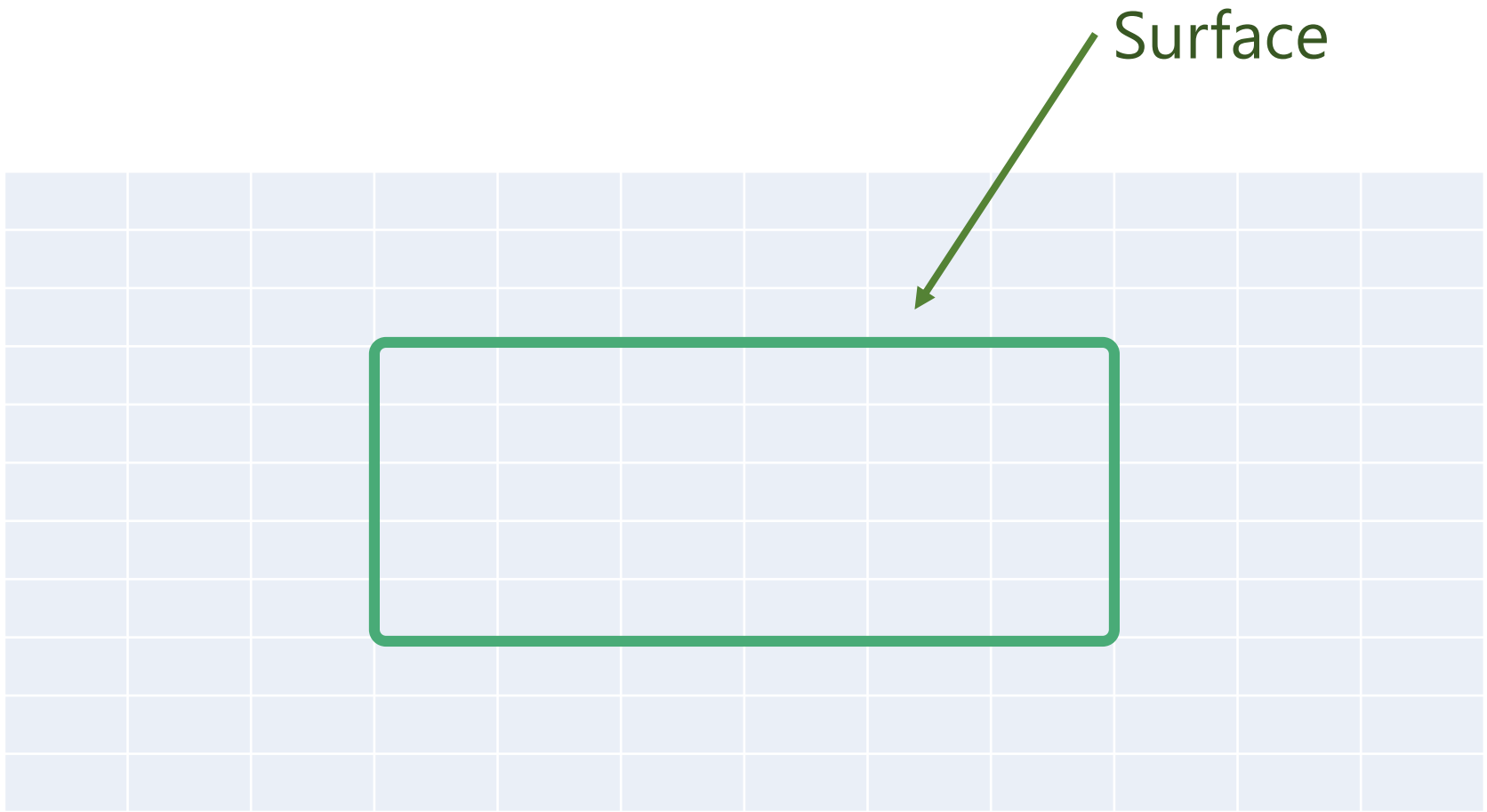
$$f: \mathbb{R}^m \rightarrow \mathbb{R}$$

Surface in 3D: $S = \{x \in \mathbb{R}^3 \mid f(x) = 0\}$

- Signed Distance Function
  - distance to the surface



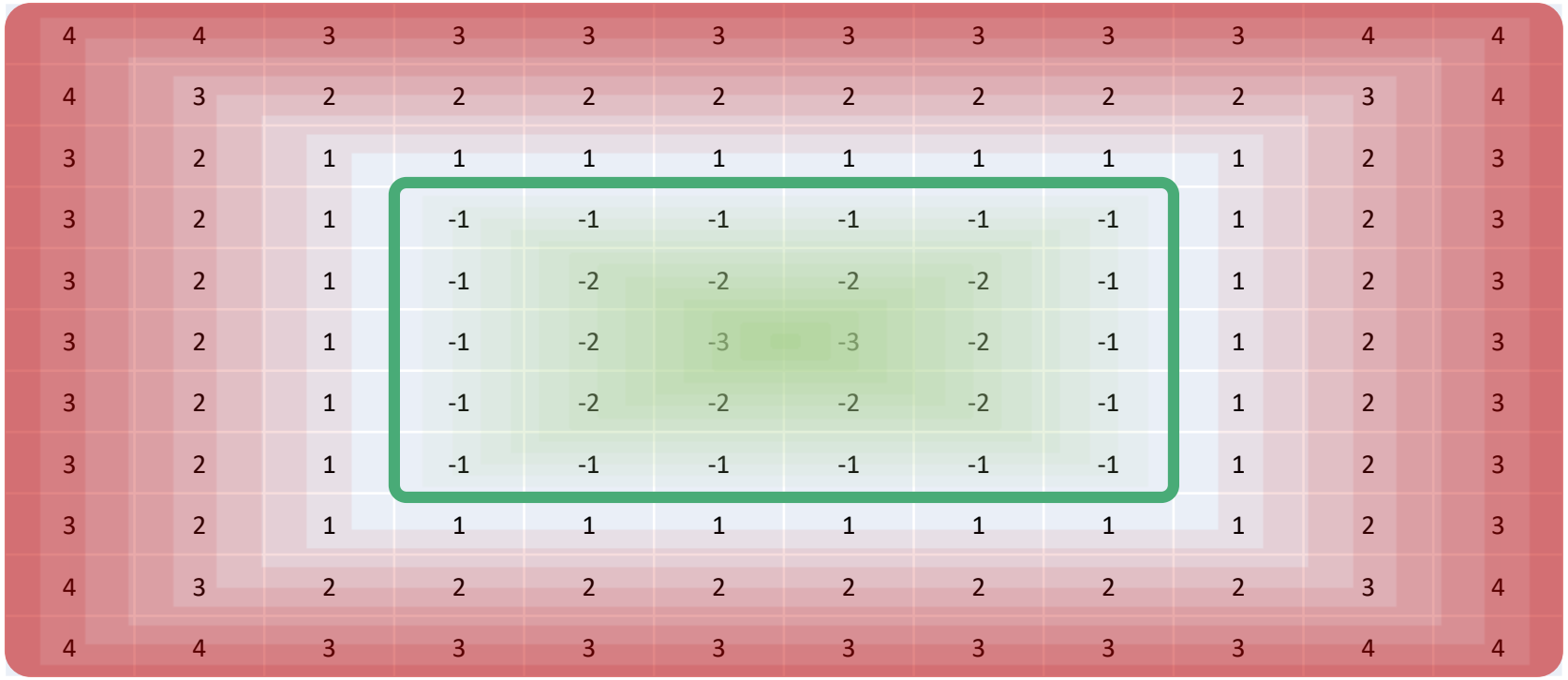
Signed Distance Field



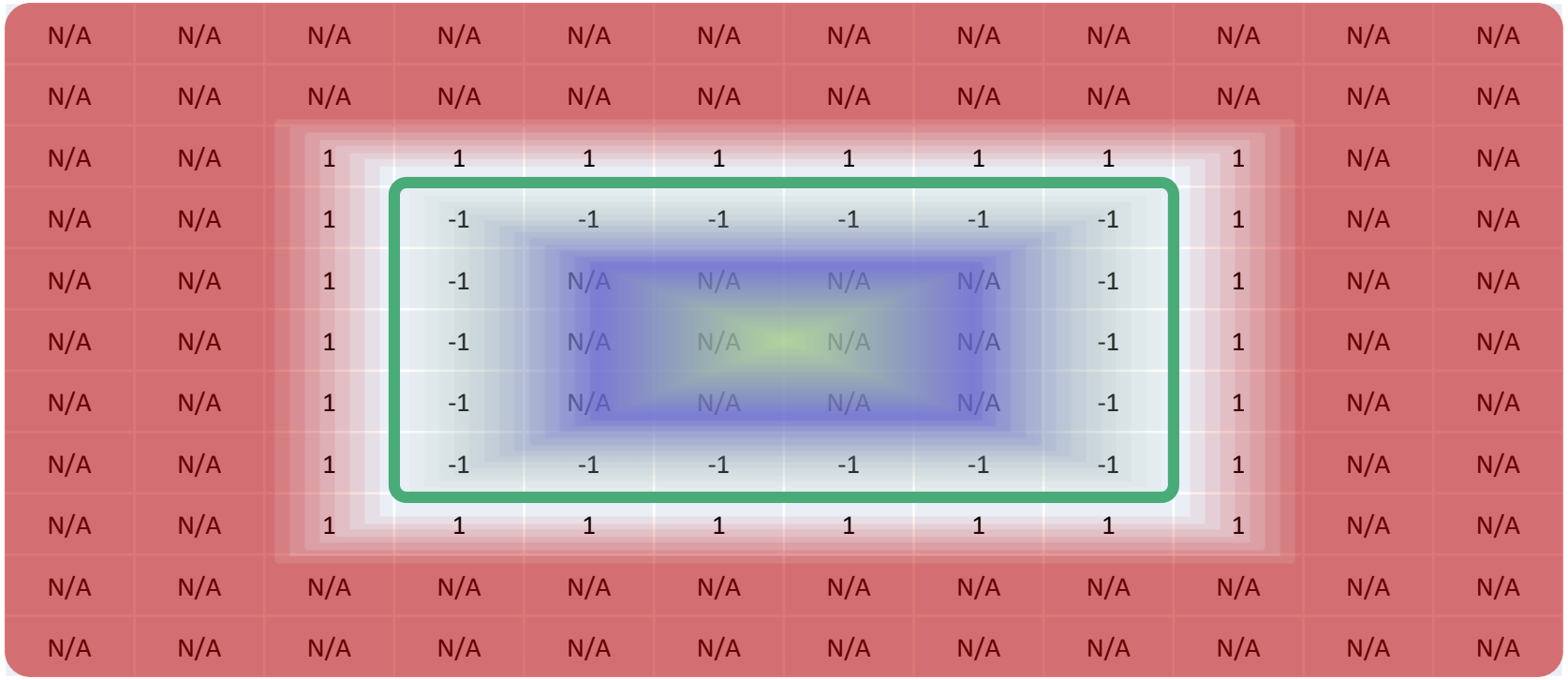
Occupancy Grid

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Signed Distance Field



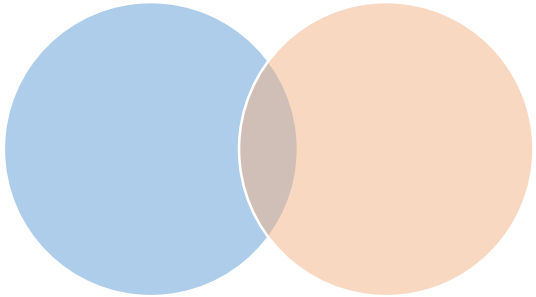
Truncated Signed Distance Field

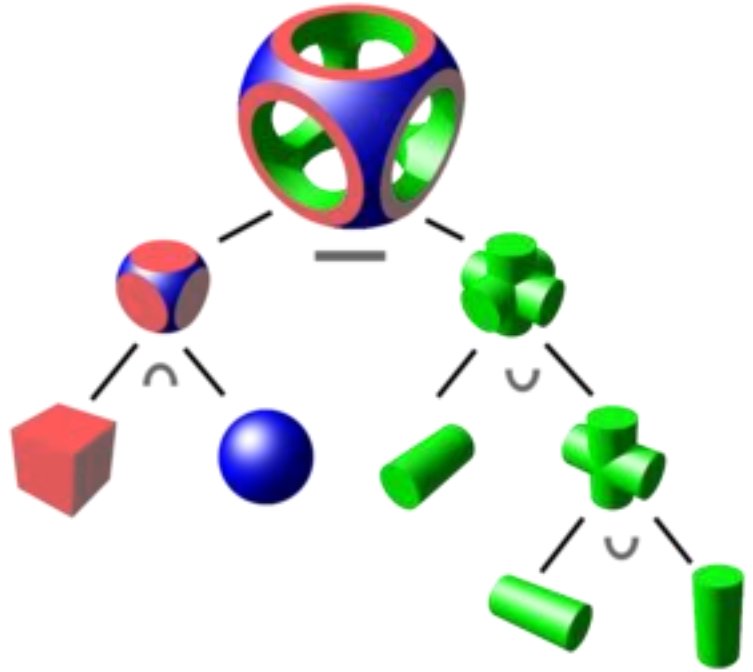


Operations on Implicit Surfaces

- Create a complex object by Boolean operators on simpler objects
- Surface is the boundary of the constructed object

- Boolean Operators (1: object, 0: not object)
  - Union
  - Intersection



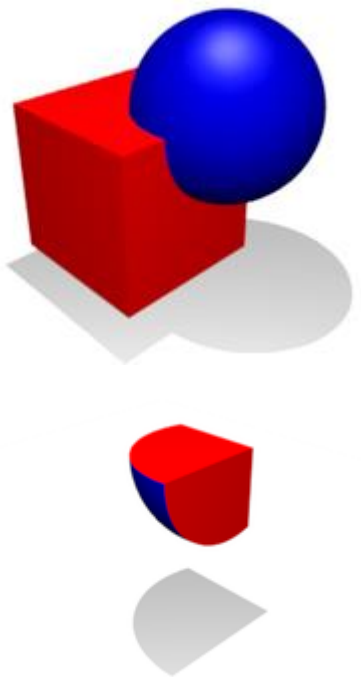


Boolean Operators on Implicit Surfaces

- Positive: outside
- Negative: inside

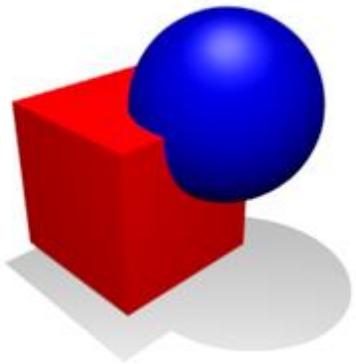
- Union: $\mathrm{U}_i\, f_i(x) = \min f_i(x)$

- Intersection: $\bigcap_i f_i(x) = \max f_i(x)$



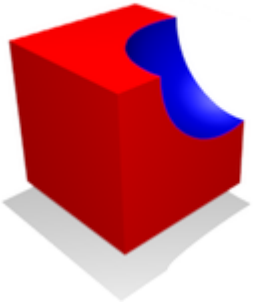
Boolean Operators on Implicit Surfaces

- Positive: outside
- Negative: inside



Union

Intersection



Subtraction

- Boolean Subtraction

	$f > 0$	$f < 0$
$g > 0$	$h > 0$	$h < 0$
$g < 0$	$h > 0$	$h > 0$

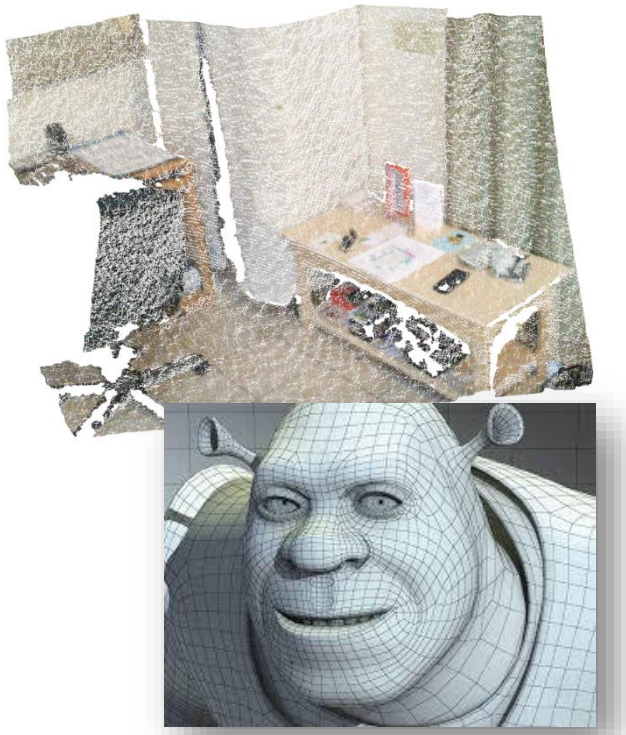
$$h = \max(f, -g)$$

- Easier than for parametric surfaces

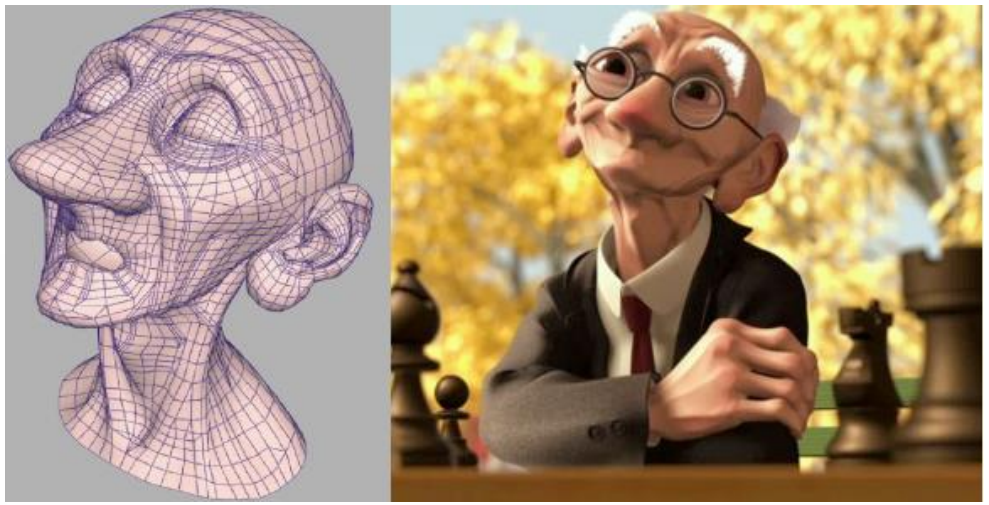
Implicit Surfaces

- Easy to determine if a point is inside/outside
- Easy to determine if a point is on the surface
- Difficult to generate points on the surface

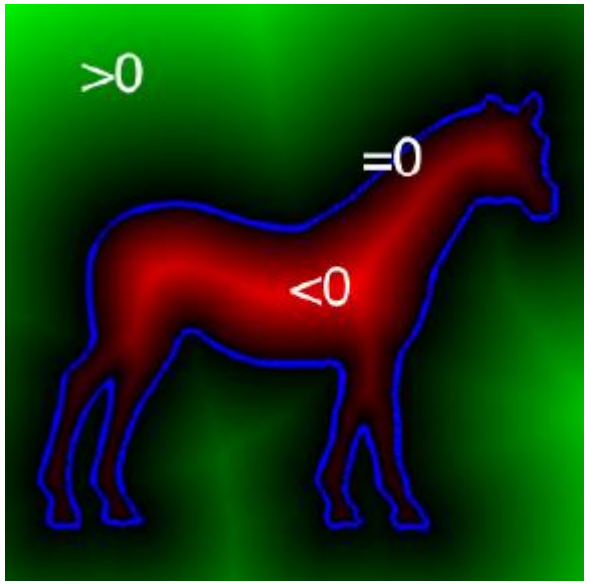
Summary



Discrete:
Meshes,
Point Samples



Parametric



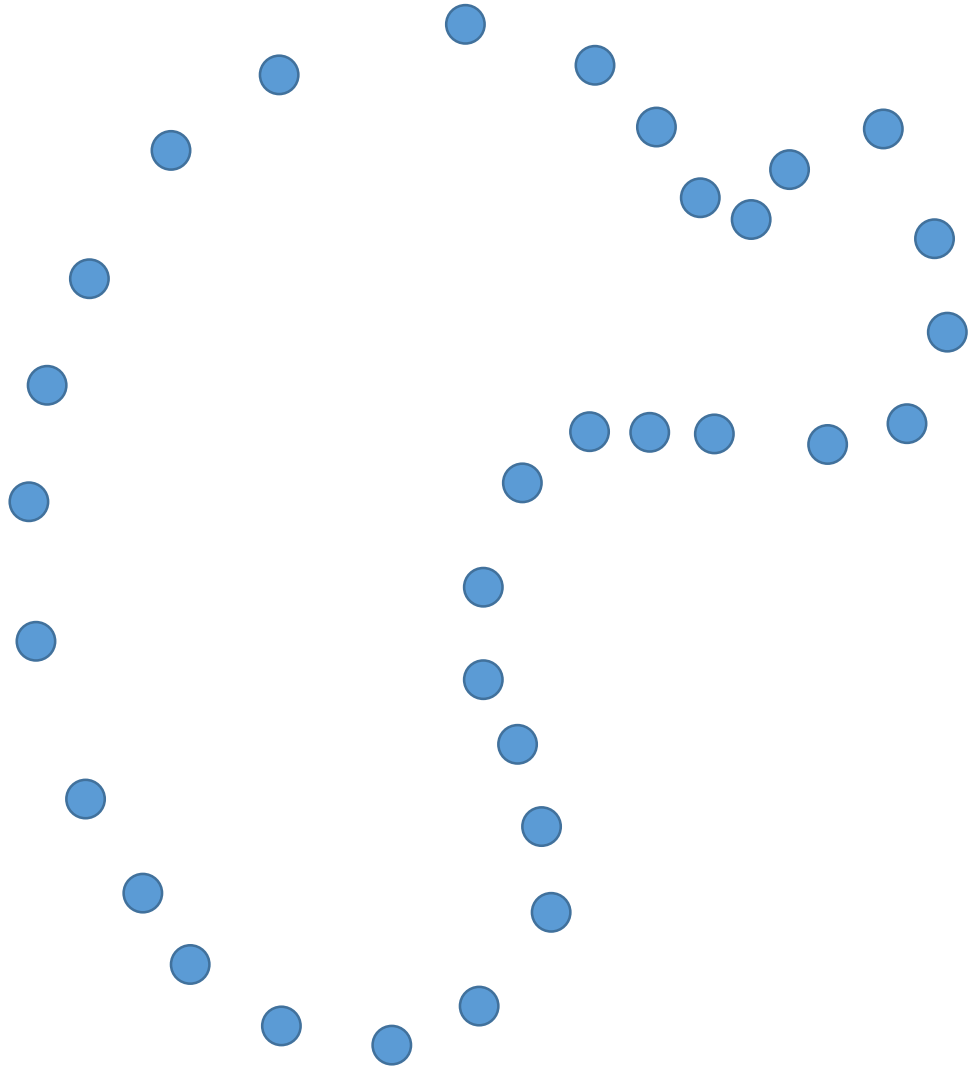
Implicit:
Distance Fields

Conversion between representations

Points to Implicit

- Fit a function
  $f: \mathbb{R}^3 \rightarrow \mathbb{R}$

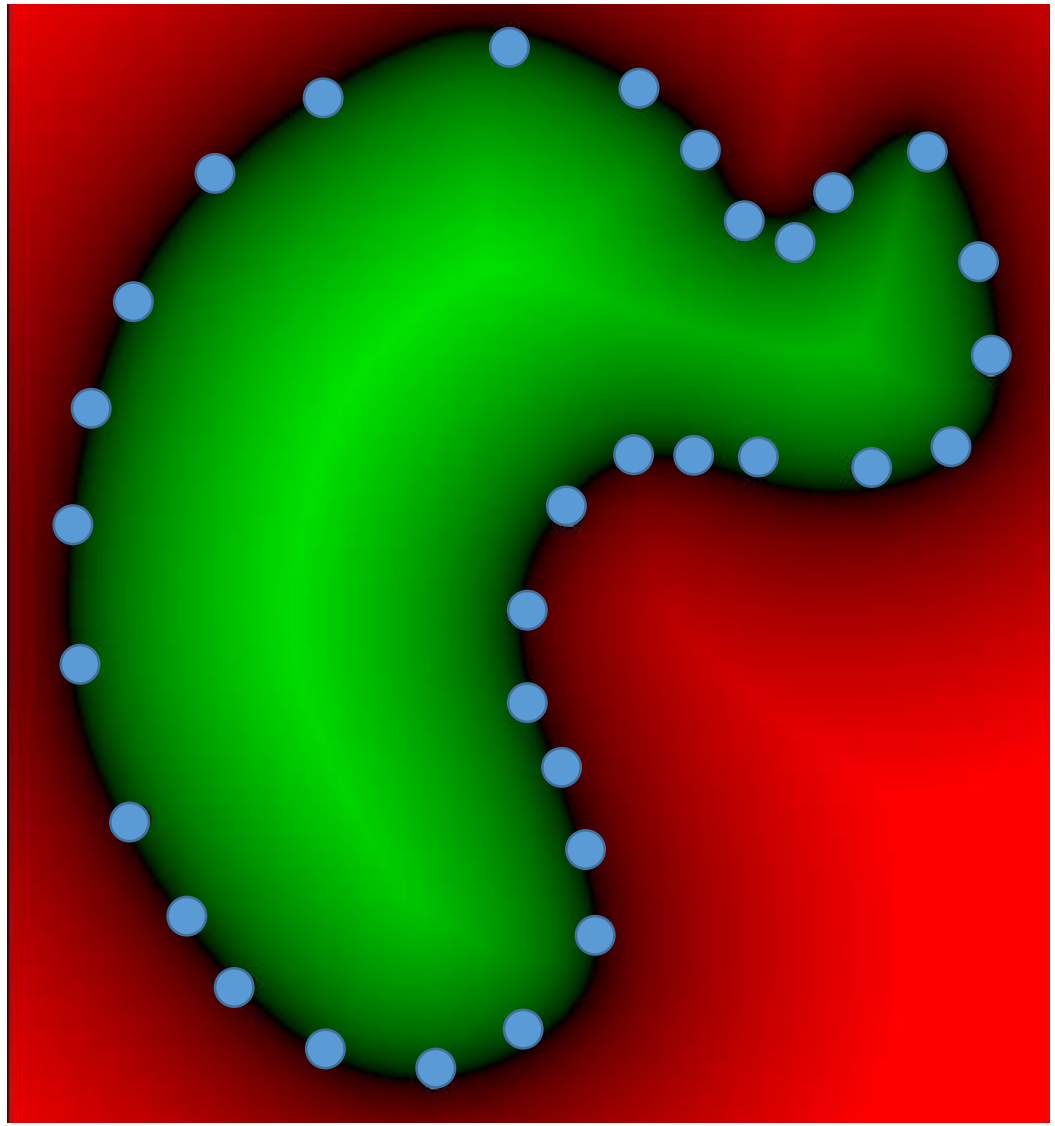
with $f < 0$ outside and $f > 0$ inside



Points to Implicit

- Fit a function
  $f: \mathbb{R}^3 \rightarrow \mathbb{R}$

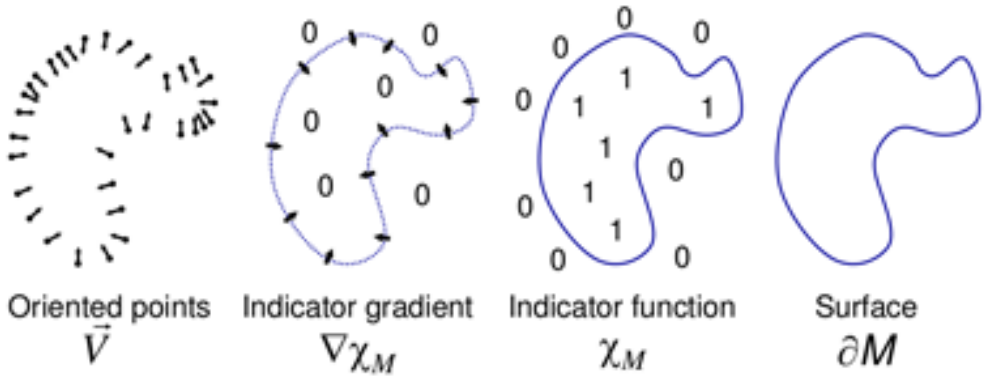
with $f < 0$ outside and $f > 0$ inside



Poisson Surface Reconstruction

- Kazhdan et al. 2006

- Given oriented points (points with normals), compute implicit indicator function

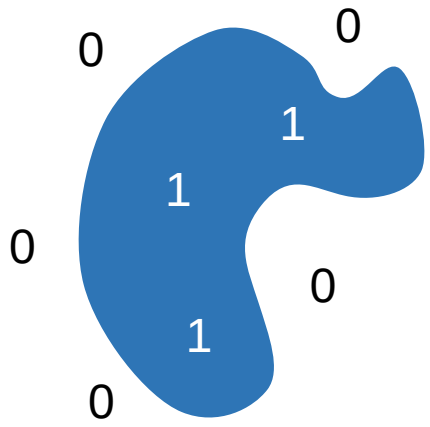


[Kazhdan et al. '06]

Poisson Surface Reconstruction

- How to construct indicator function?

$$\chi_M(p) = \begin{cases} 1 & \text{if } p \in M \\ 0 & \text{if } p \notin M \end{cases}$$

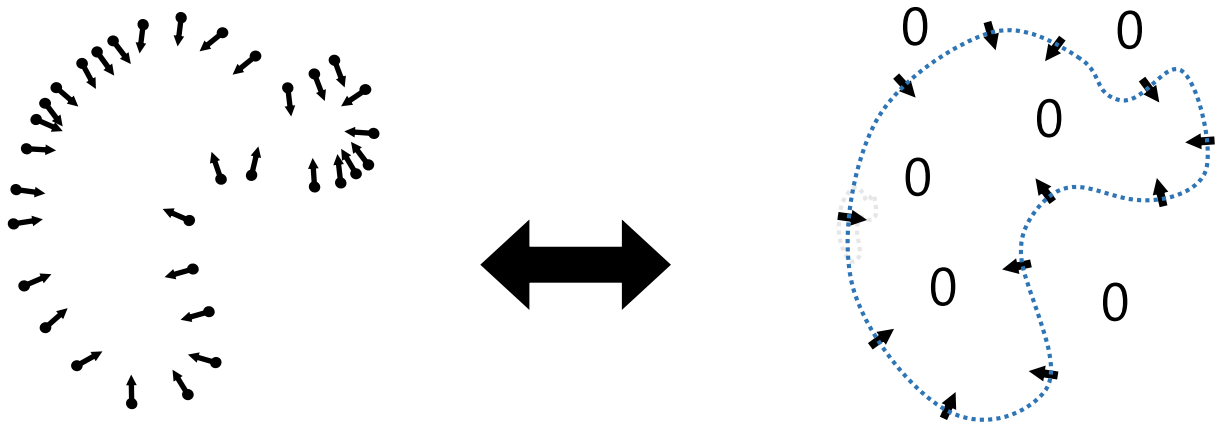


Indicator function
χ_M

[Kazhdan et al. '06]

Poisson Surface Reconstruction

- Relationship between point normal and gradient of indicator function



Oriented points

Indicator gradient
$\nabla \chi_M$

[Kazhdan et al. '06]

Poisson Surface Reconstruction

- Represent points by a vector field V
- Find function f whose gradient approximates V

$$\min_f \|\nabla f - V\|$$

- Transform to Poisson problem:

$$\nabla \cdot (\nabla f) = \nabla \cdot V \qquad \leftrightarrow \qquad \Delta f = \nabla \cdot V$$

- Solve as least squares fitting problem

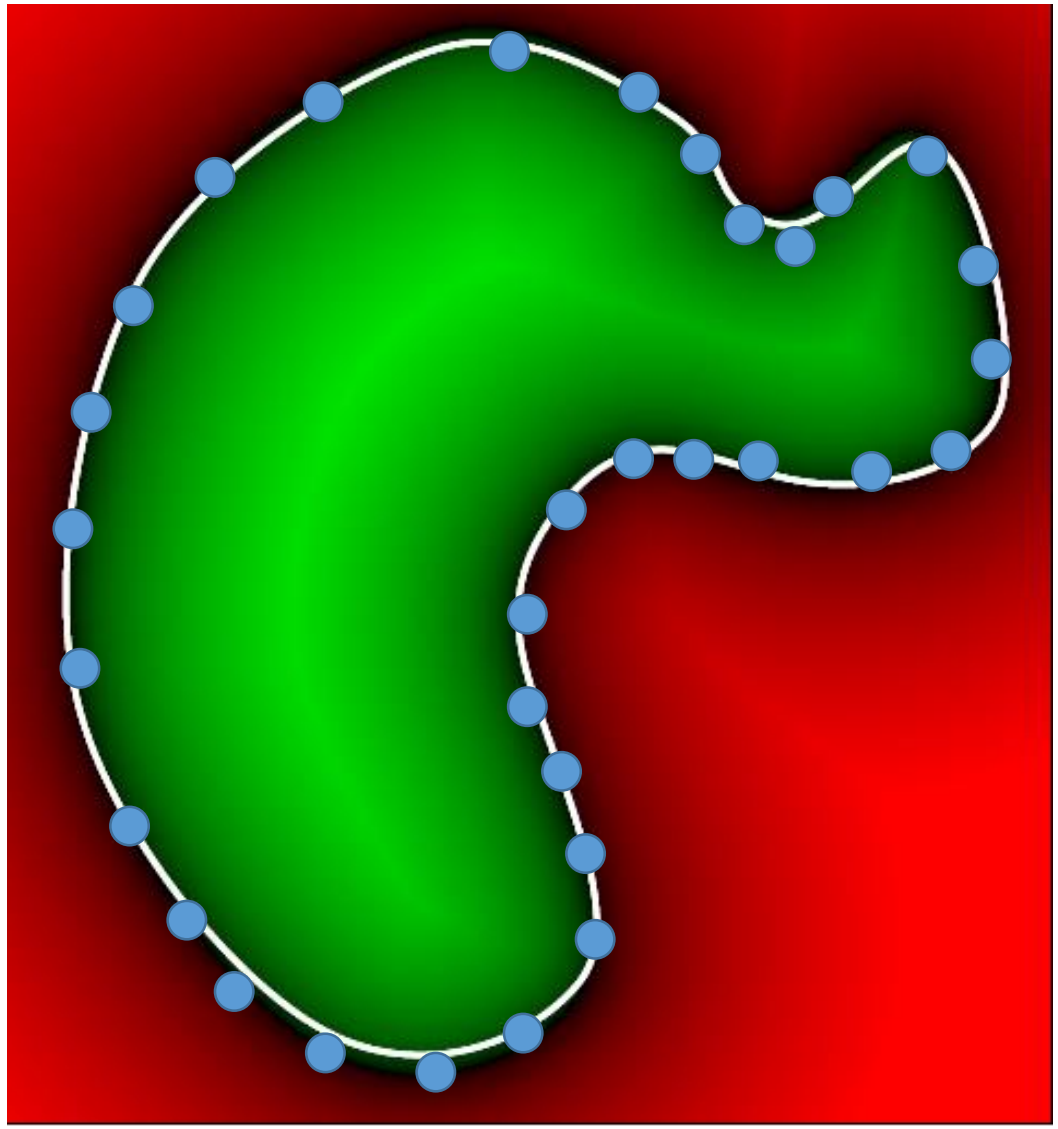
[Kazhdan et al. '06]

Points to Implicit to Mesh

- Fit a function
  $f: \mathbb{R}^3 \rightarrow \mathbb{R}$

with $f < 0$ outside and $f > 0$ inside

- To Mesh: Extract surface at zero-set $\{x: f(x) = 0\}$



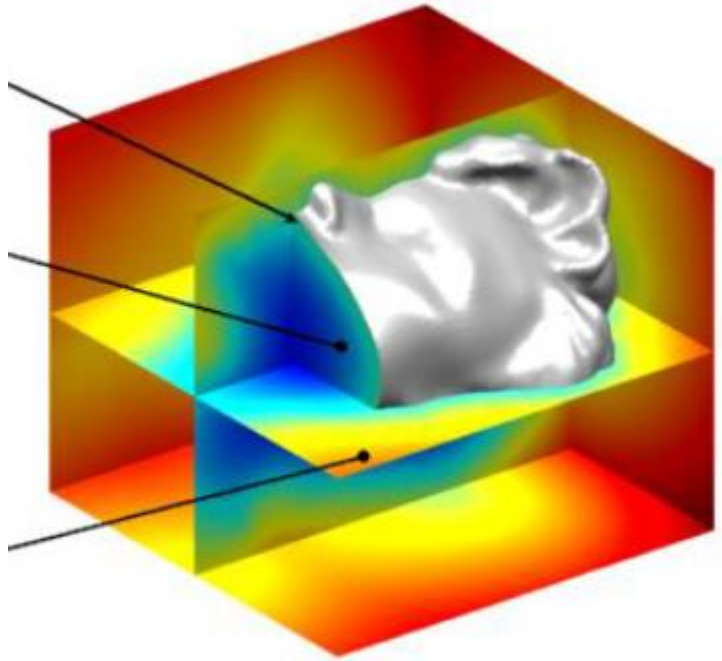
Implicit to Mesh

- Marching Cubes, Lorensen and Cline 1987
- Compute manifold mesh of level set

$f(x) = 0 \rightarrow$ surface

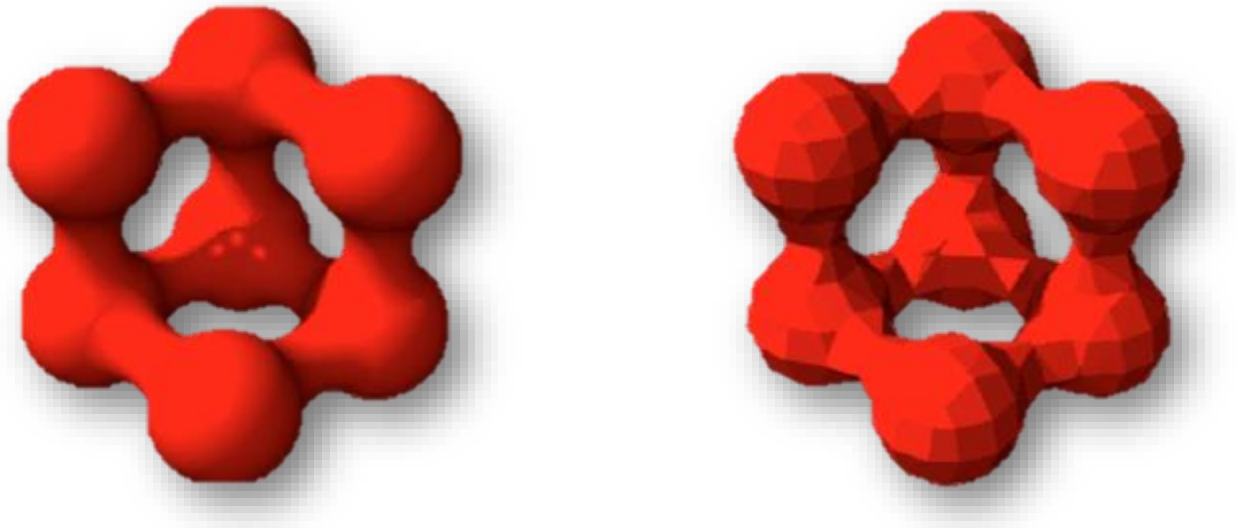
$f(x) < 0 \rightarrow$ inside

$f(x) > 0 \rightarrow$ outside



Marching Cubes

- Convert from implicit to explicit
- Given an implicit representation $\{x, \text{s.t. } f(x) = 0\}$, create a triangle mesh approximating the surface



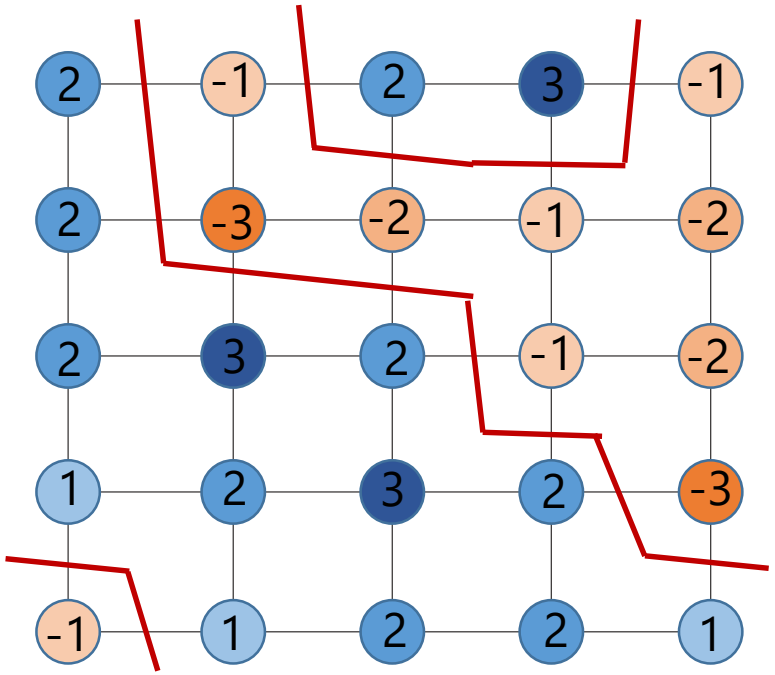
Marching Squares (2D)

- Given: uniform sampling on a 2D grid of an implicit function f
- Determine zero-crossings: edges with a sign switch

$$f(x_1) < 0, f(x_2) > 0 \Rightarrow$$
$$f\big(x_1 + t(x_2 - x_1)\big) = 0$$
$$\text{for some } 0 \leq t \leq 1$$

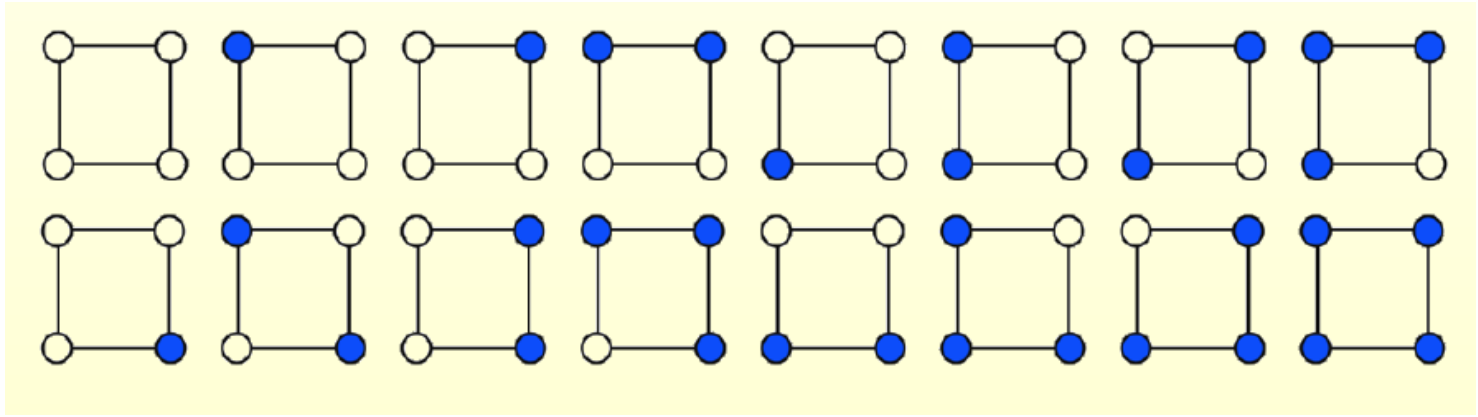
- Simplest way to compute t:
  assume f is linear between x_1, x_2

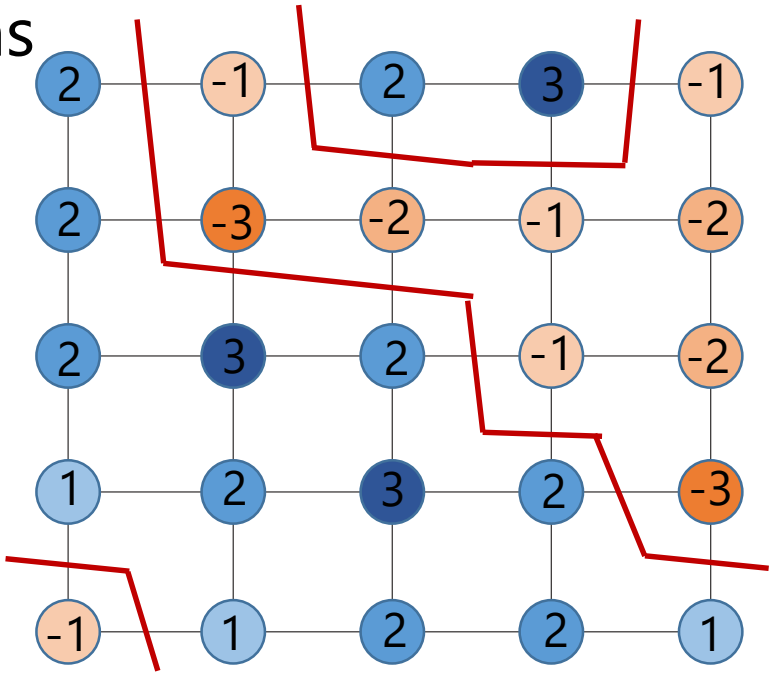
$$t = \frac{f(x_1)}{f(x_2) - f(x_1)}$$



Marching Squares (2D)

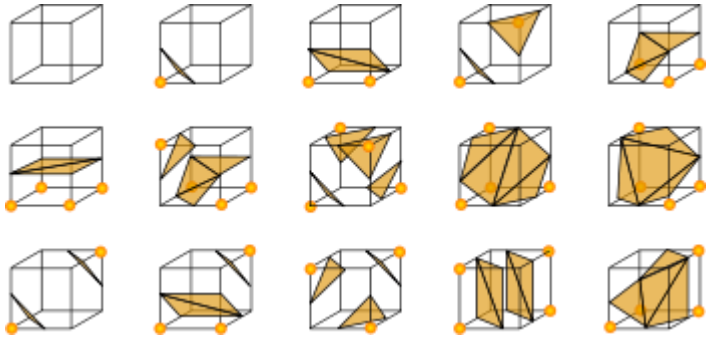
- Given: uniform sampling on a 2D grid of an implicit function f
- Determine zero-crossings
- Consider all possible inside/outside combinations
- Group to lookup table



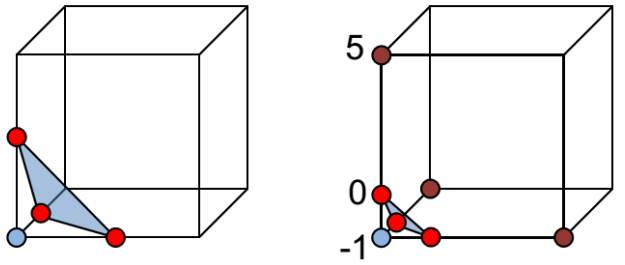


Marching Cubes (3D)

- Marching cubes lookup table



- Linear interpolation based on function values



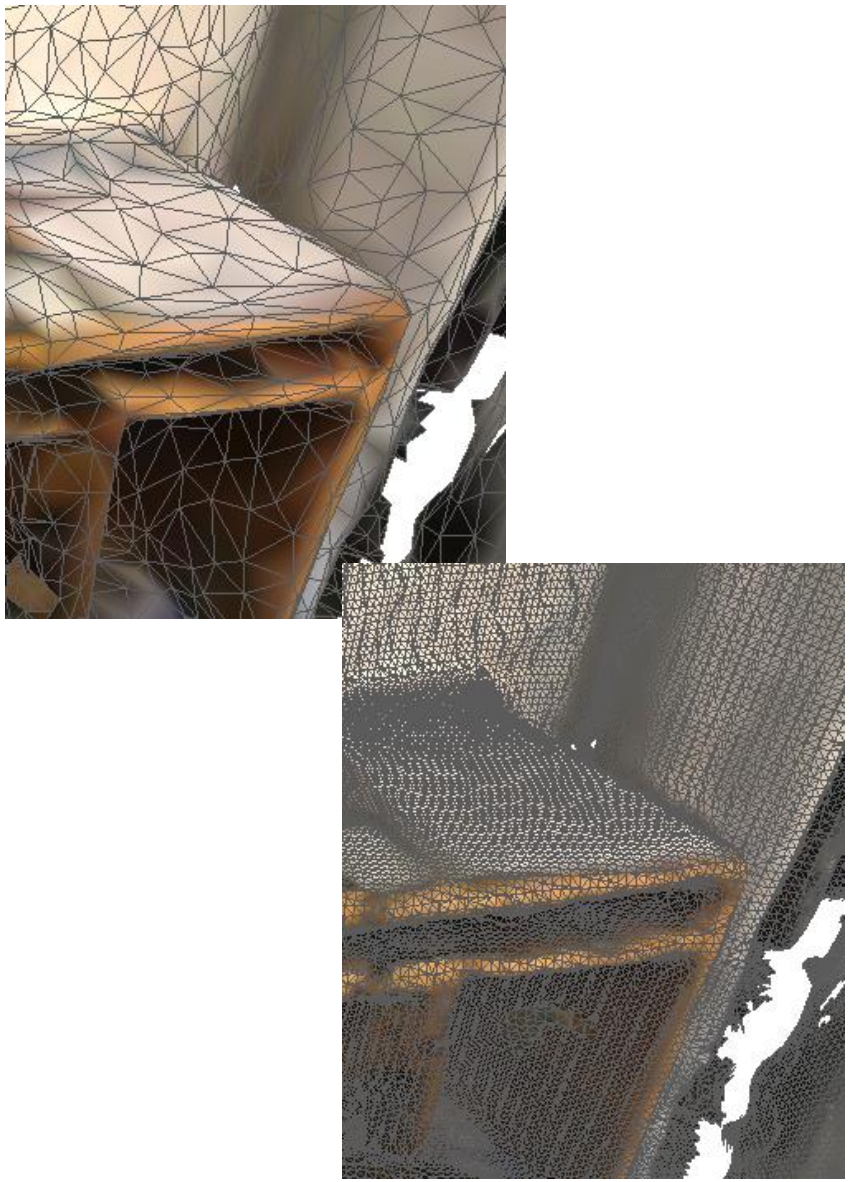
From table

Real intersections

Marching Cubes

- Multi-purpose, widely applicable
- Easy to compute and parallelizable
- Relatively simple to implement

- Can create badly shaped triangles (skinny)
- Many special cases (lookup table)
- No sharp features



Mesh to Points

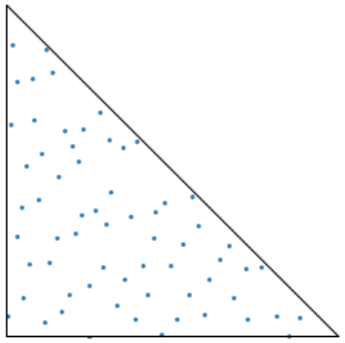
- Why?
- Some meshes can have bad triangles, bad connectivity
  e.g., often CAD models, amateur models
- Machine learning on points

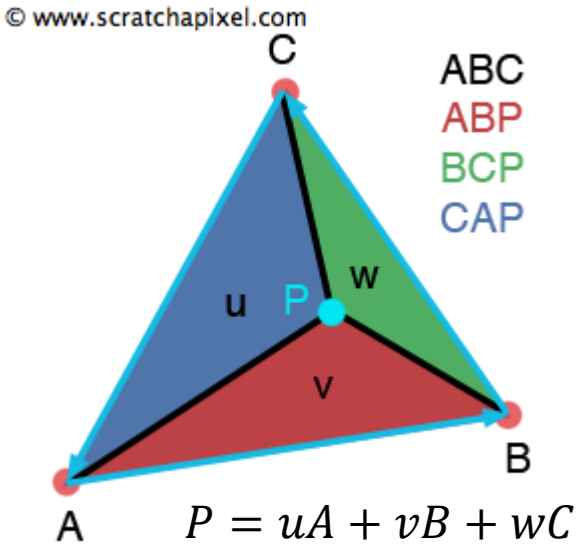
- How to sample the mesh?

Mesh to Points

- Simple approximation for uniform sampling
- Sample each triangle based on its surface area

- Sample each triangle uniformly: barycentric coordinates
- Sample random $r_1, r_2 \in [0,1]$:
  - $p = \left(1 - \sqrt{r_1}\right)A + \sqrt{r_1}(1 - r_2)B + \sqrt{r_1}r_2C$





$$P = uA + vB + wC$$
$$u + v + w = 1$$

Mesh to Points

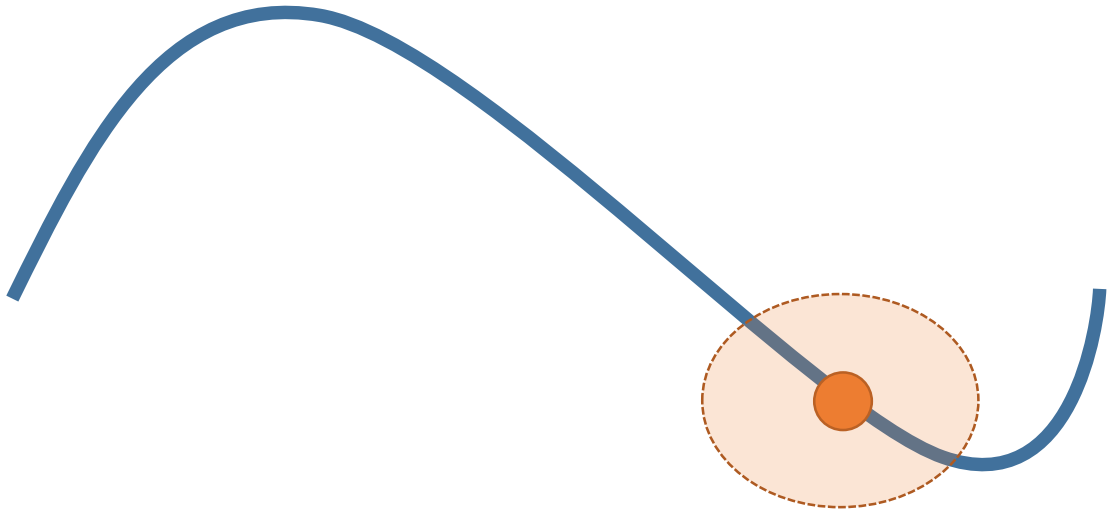
- Alternative: Farthest Point Sampling
- From an initial sample point set S, sample new point by finding the farthest point from all points in S. Iterate.

- Depends on notion of distance in domain
- Can weight adaptively (weighted distance)

- On mesh: discrete geodesic distance (path along edges)

Geometric Foundations

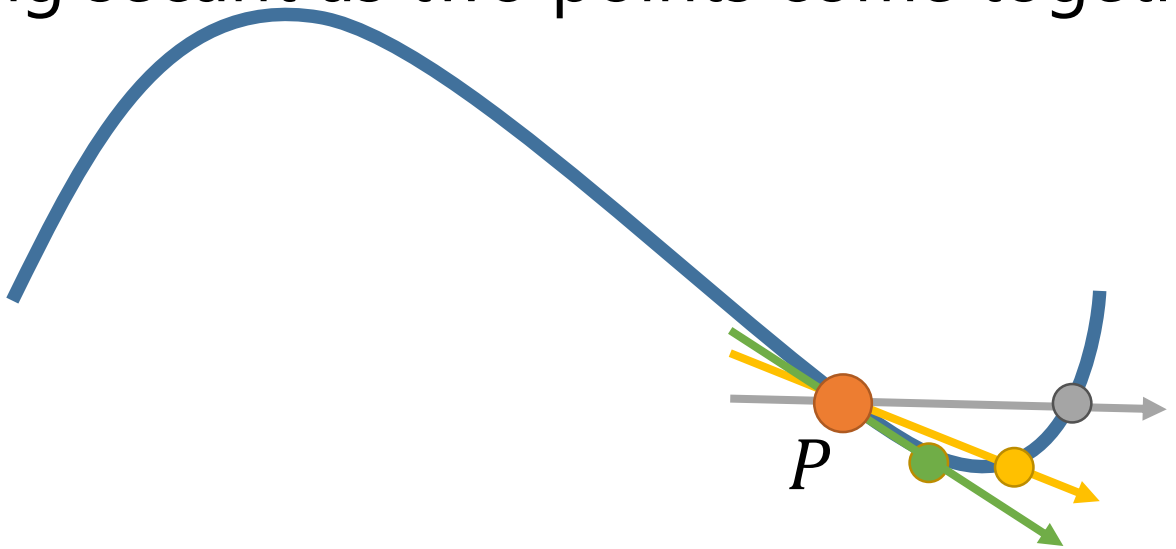
- How to describe the geometry of a local observation: point and its neighborhood



Geometric Foundations

- How to describe the geometry of a local observation: point and its neighborhood

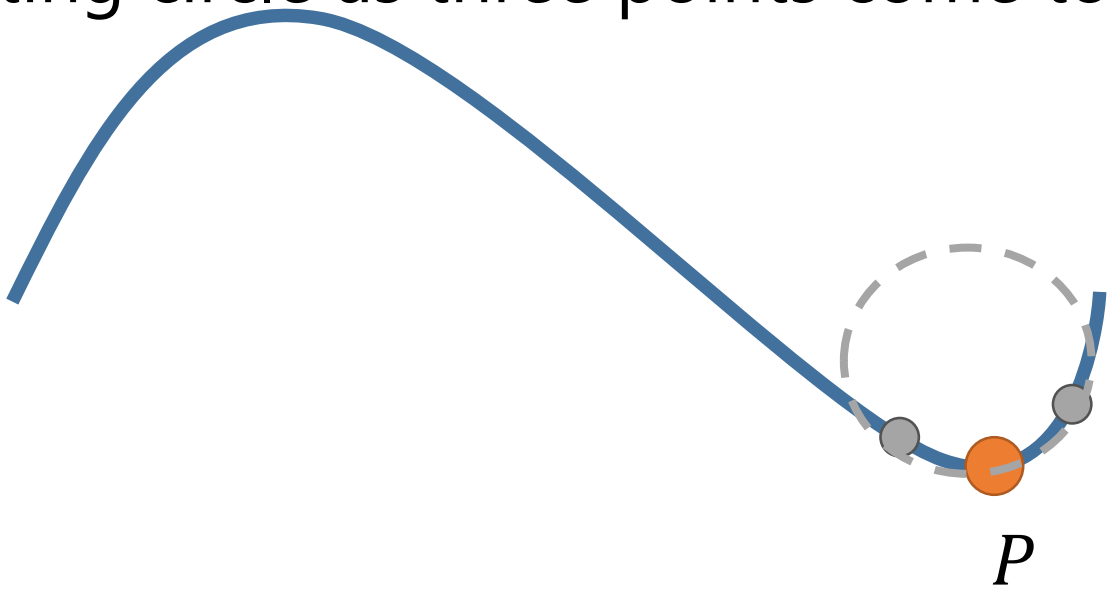
- Tangent: limiting secant as two points come together



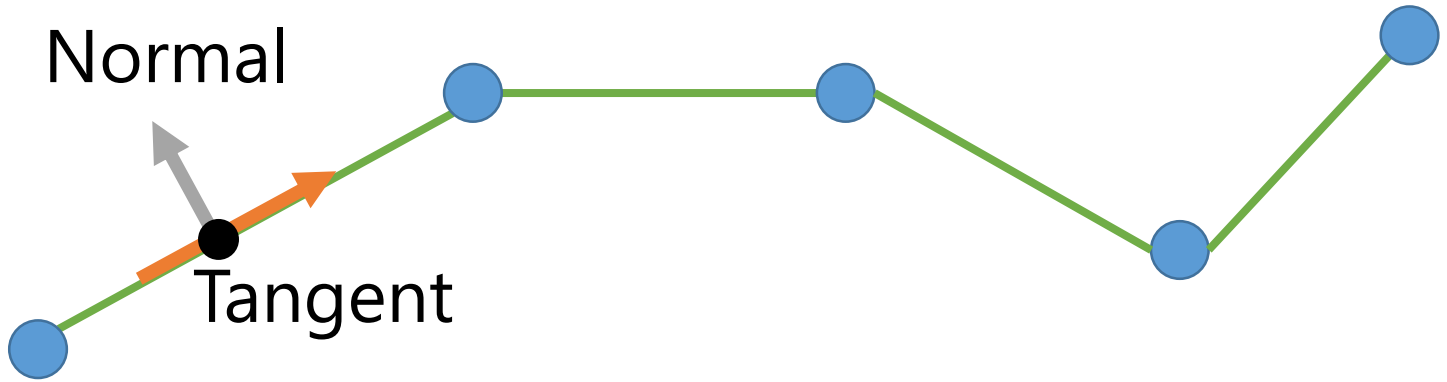
Geometric Foundations

- How to describe the geometry of a local observation: point and its neighborhood

- Curvature: limiting circle as three points come together

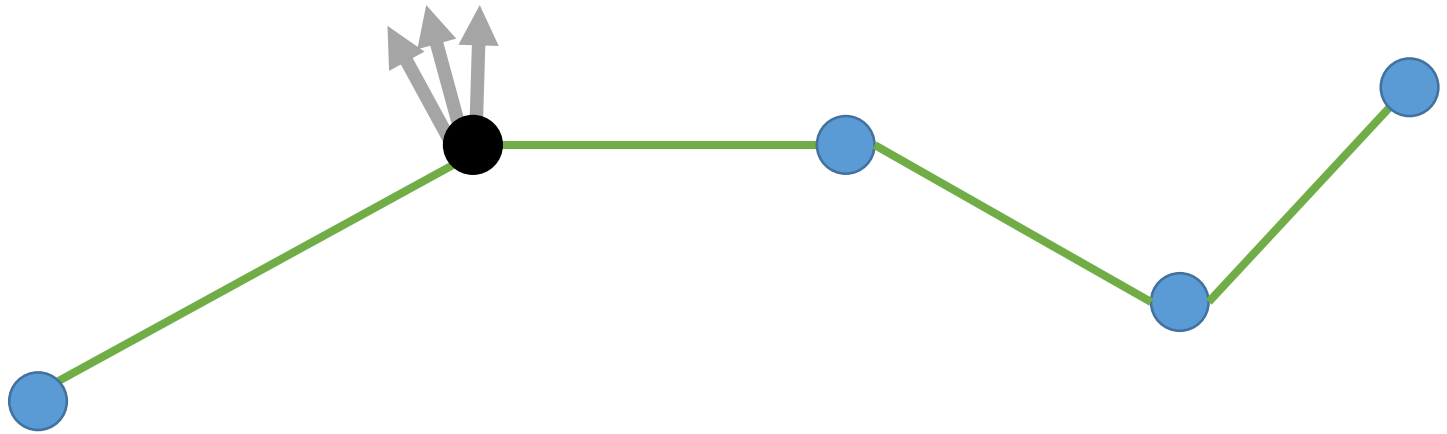


On Discrete Curves



On Discrete Curves

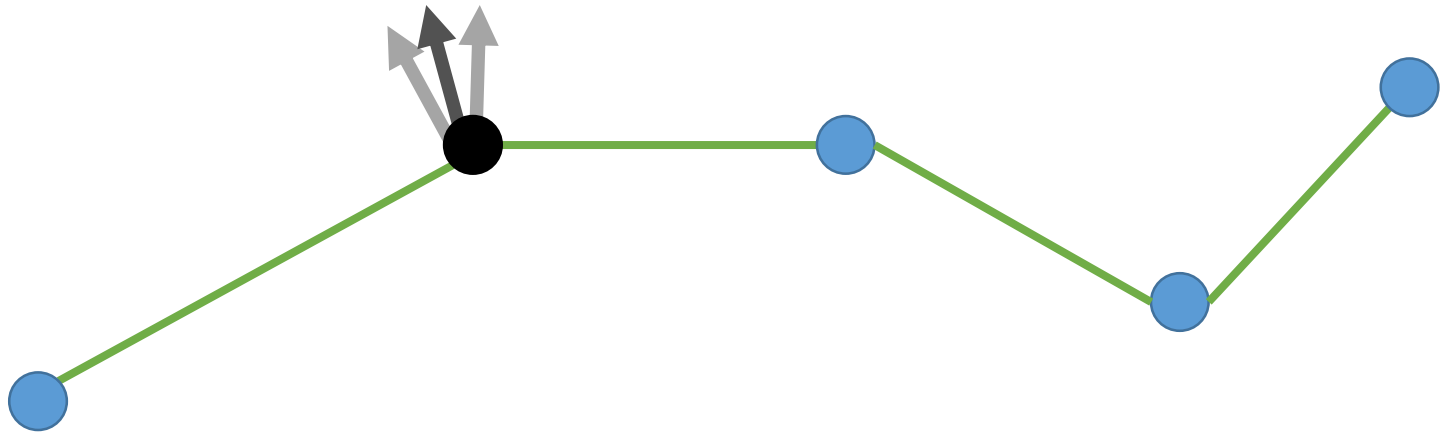
- What about at vertices?



On Discrete Curves

- What about at vertices?

- Can average adjacent normals

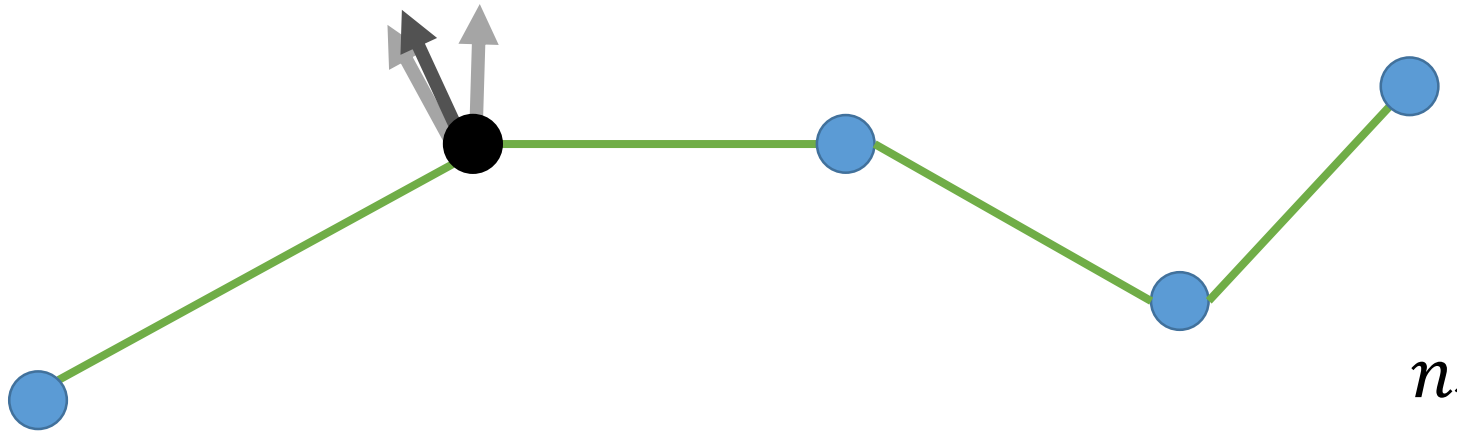


$$n_v = \frac{n_{e_1} + n_{e_2}}{\left\| n_{e_1} + n_{e_2} \right\|}$$

On Discrete Curves

- What about at vertices?

- Can average adjacent normal

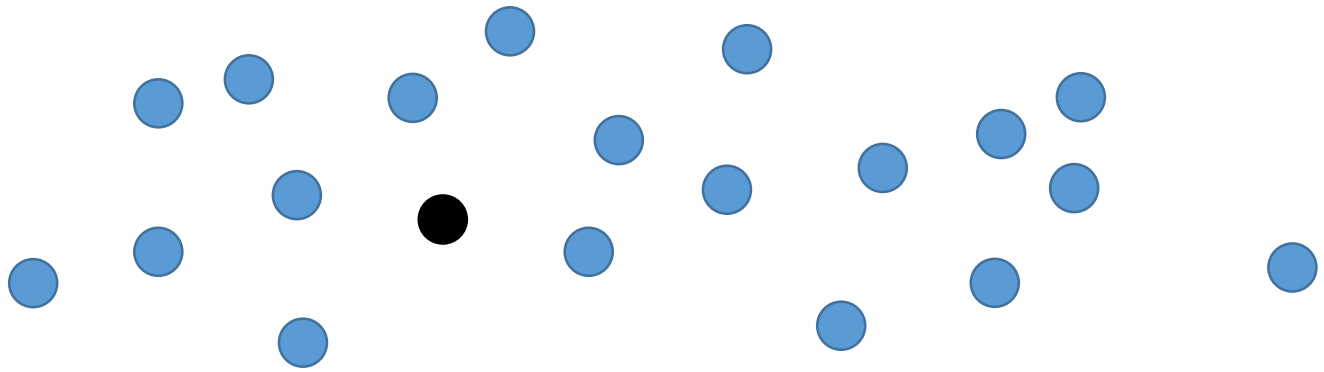


$$n_v = \frac{|e_1|n_{e_1} + |e_2|n_{e_2}}{\left\||e_1|n_{e_1} + |e_2|n_{e_2}\right\|}$$

- Weight by edge lengths

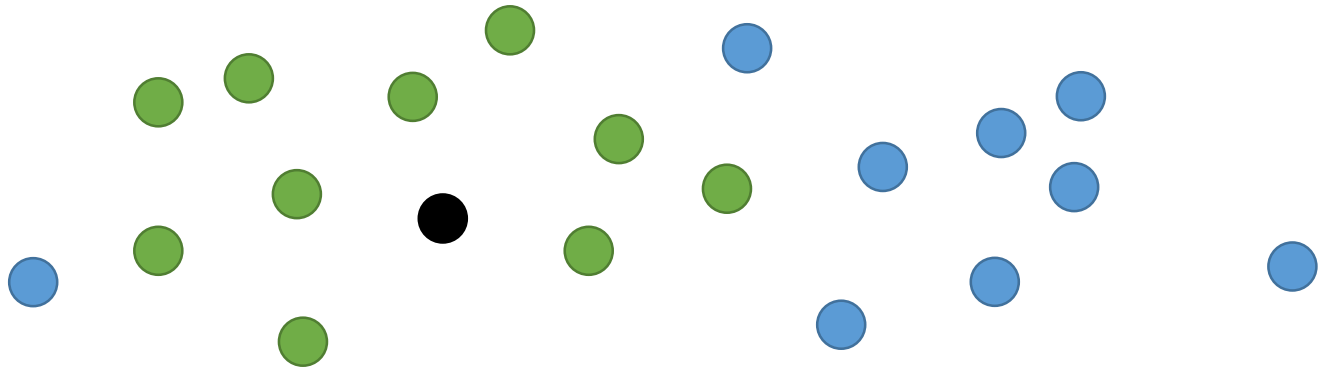
On Point Clouds

- Estimate normal by approximating the plane tangent to the surface
- Least-squares fitting problem



On Point Clouds

- Estimate normal by approximating the plane tangent to the surface
- Least-squares fitting problem
- Find neighborhood around point



Estimate plane by PCA of neighbors

Some Useful Software

- MeshLab (www.meshlab.net)
  - Viewing and processing meshes
- OpenMesh (www.openmesh.org)
  - Mesh processing
- CGAL (www.cgal.org)
  - Computational geometry