

Procedural Synthesis

Acknowledgment: Sylvain Lefebvre, Jeremie Dumas

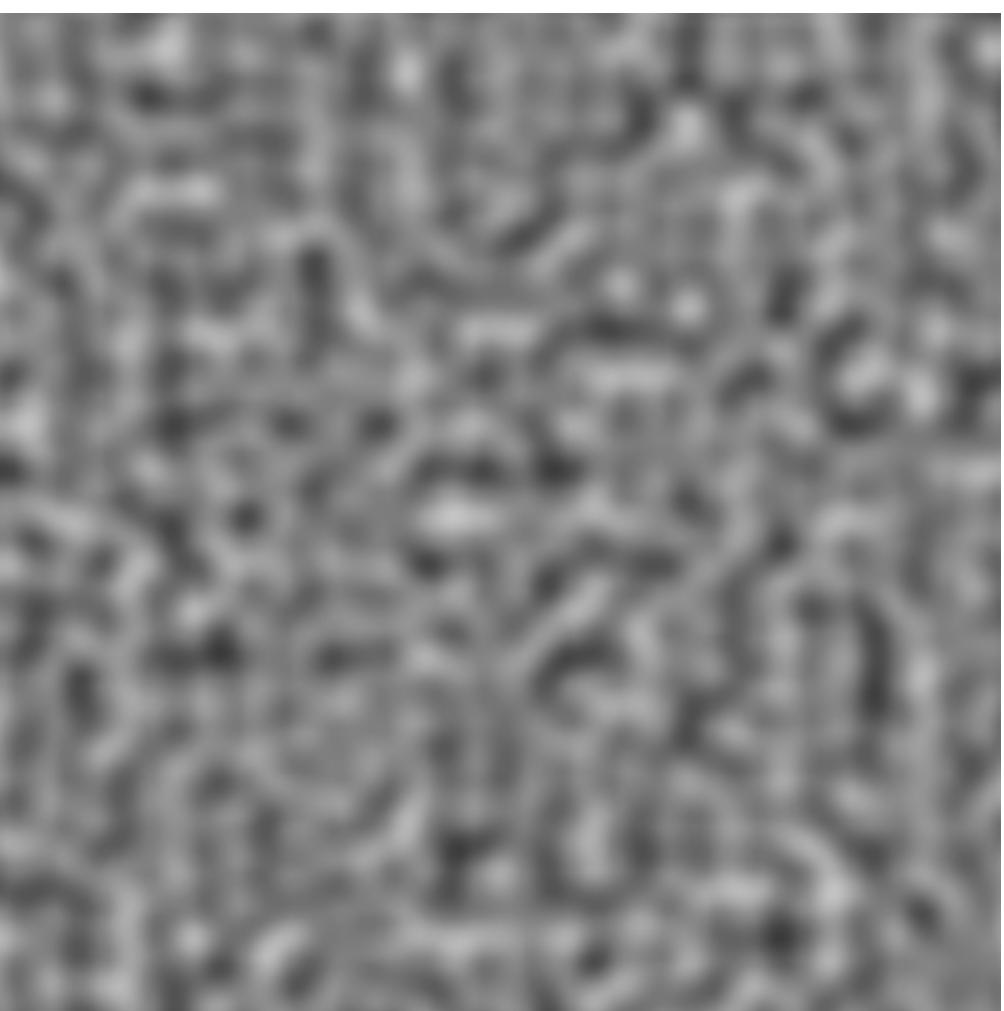


Inigo Quilez (<https://www.shadertoy.com/view/4ttSWf>)

Procedural Noise

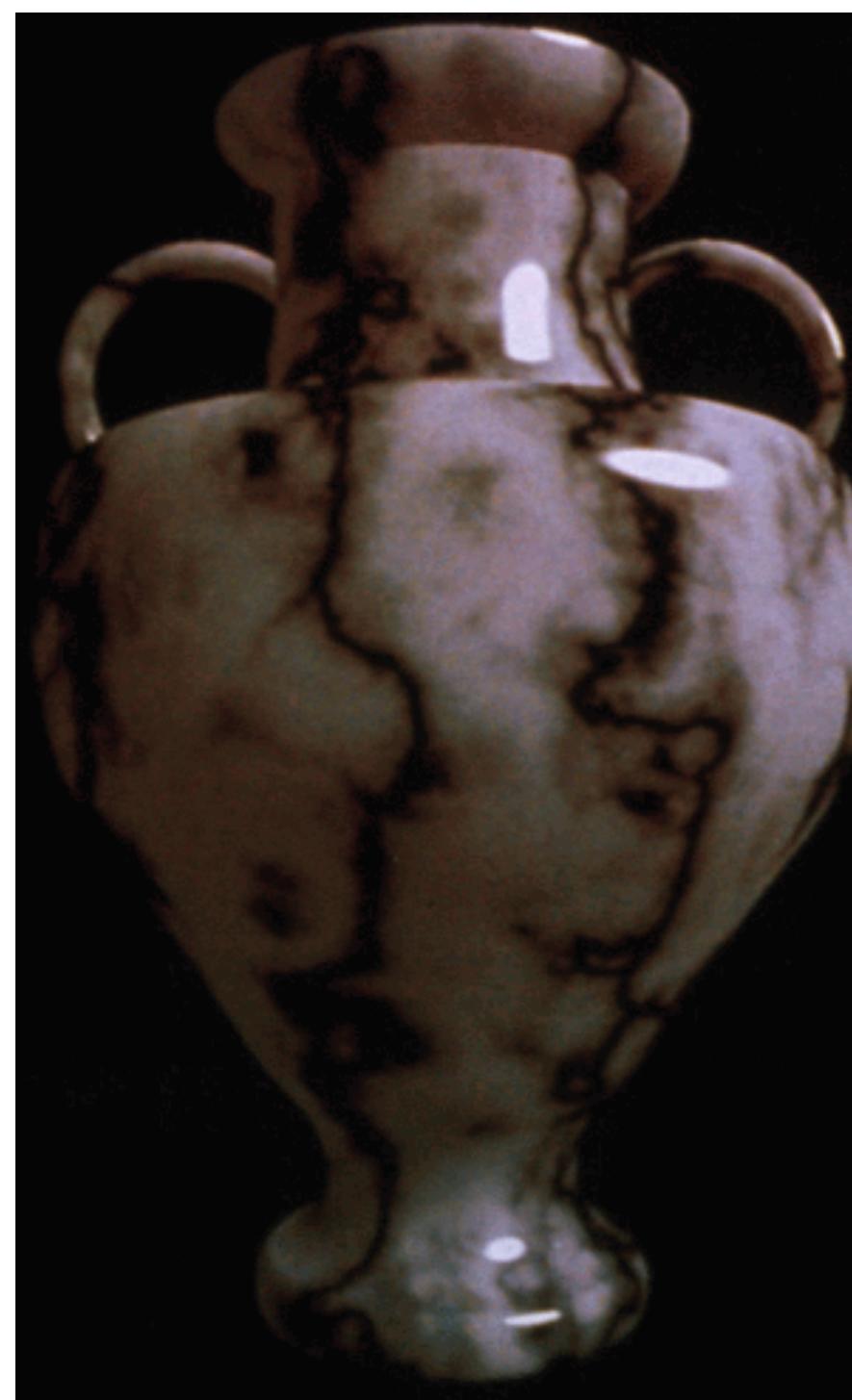
We start with noise functions

- Goal: create realistic “textures” at inexpensive costs.



Procedural noise

$F(\text{noise})$
→
colors,
normals,
roughness...

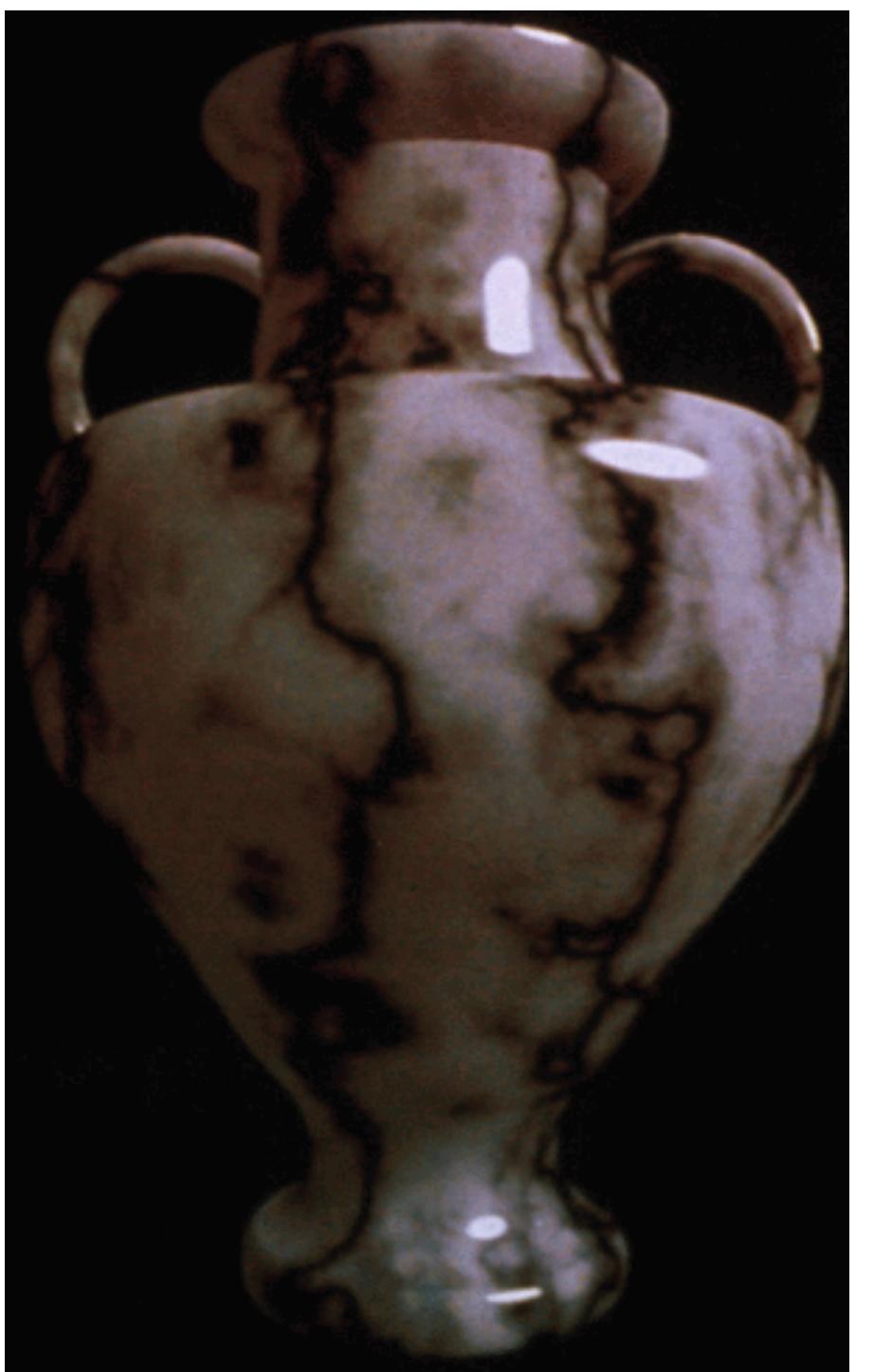
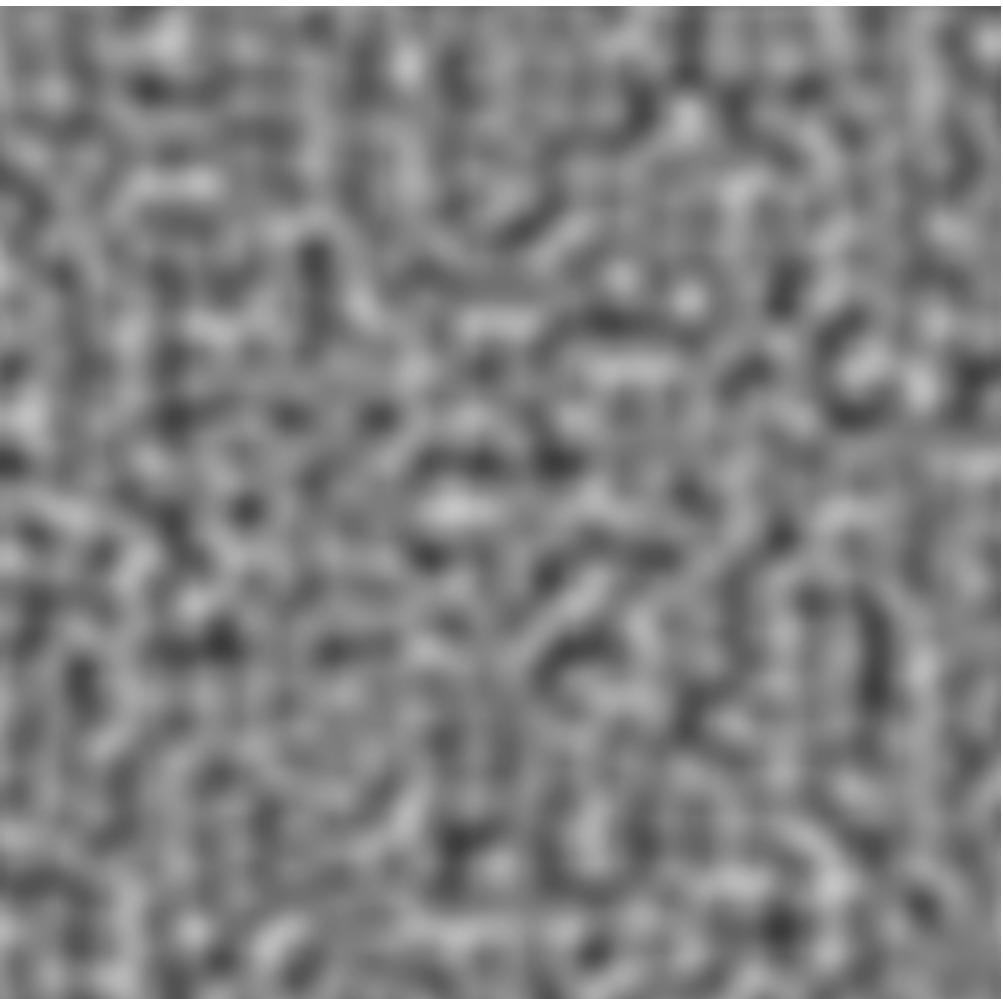


Procedural texture
[Perlin 1985]

What is a good noise function?

Randomly controlled primitive $F(p) \in \mathbb{R}$

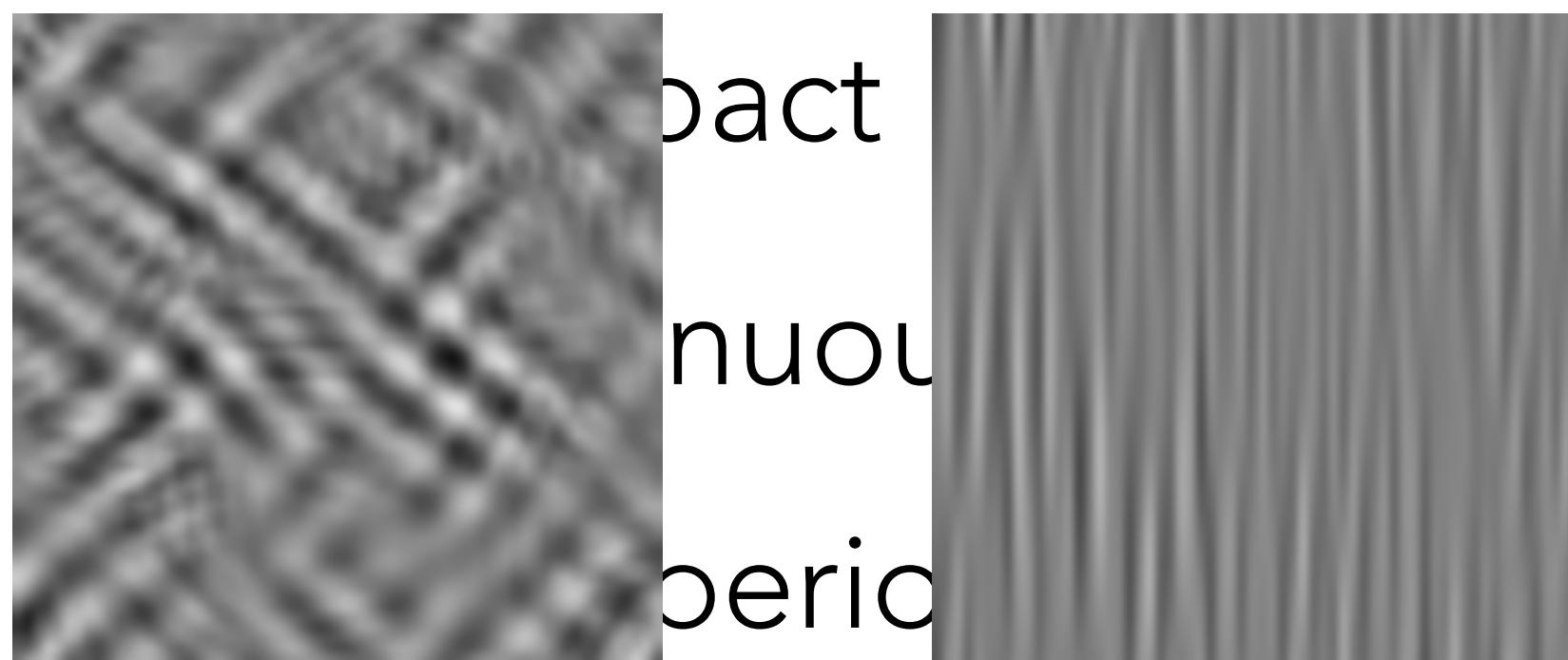
- ✗ White noise (salt)
- ✗ Smooth function
- ✓ Smooth function + salt



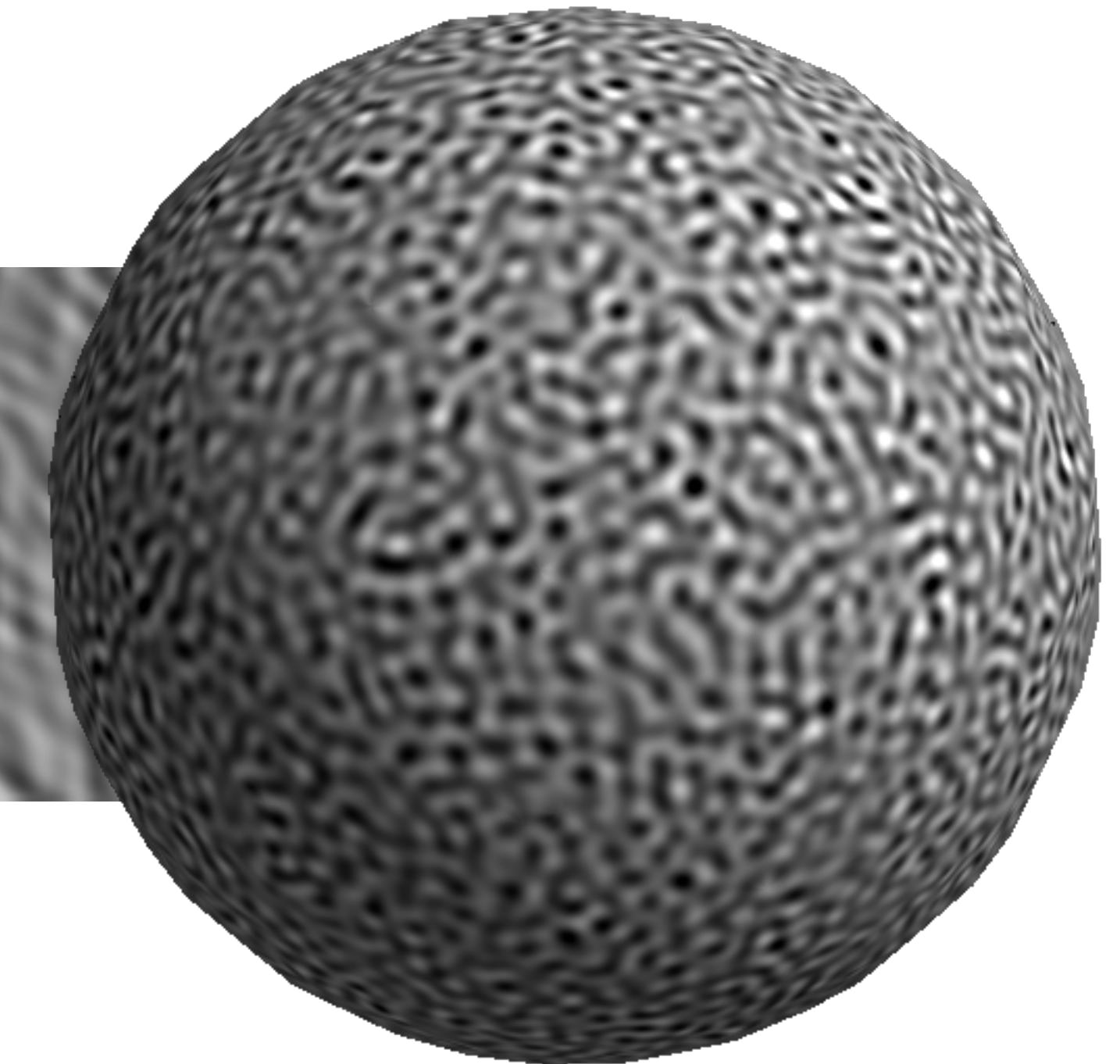
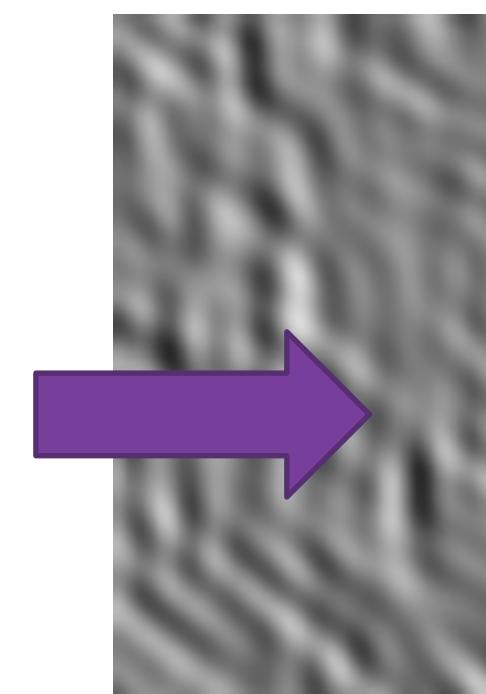
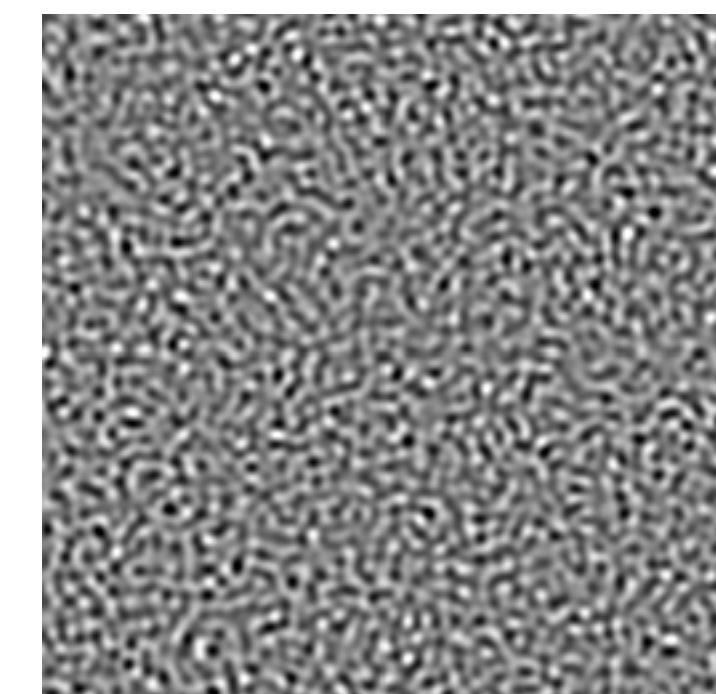
Perlin noise [85]

Noise Requirements

- We want noise with **controllable appearance**
- Other desirable noise properties:



compact
nuous
periodic



- Fast
- Applied to 3D objects: map to surface or 3D function

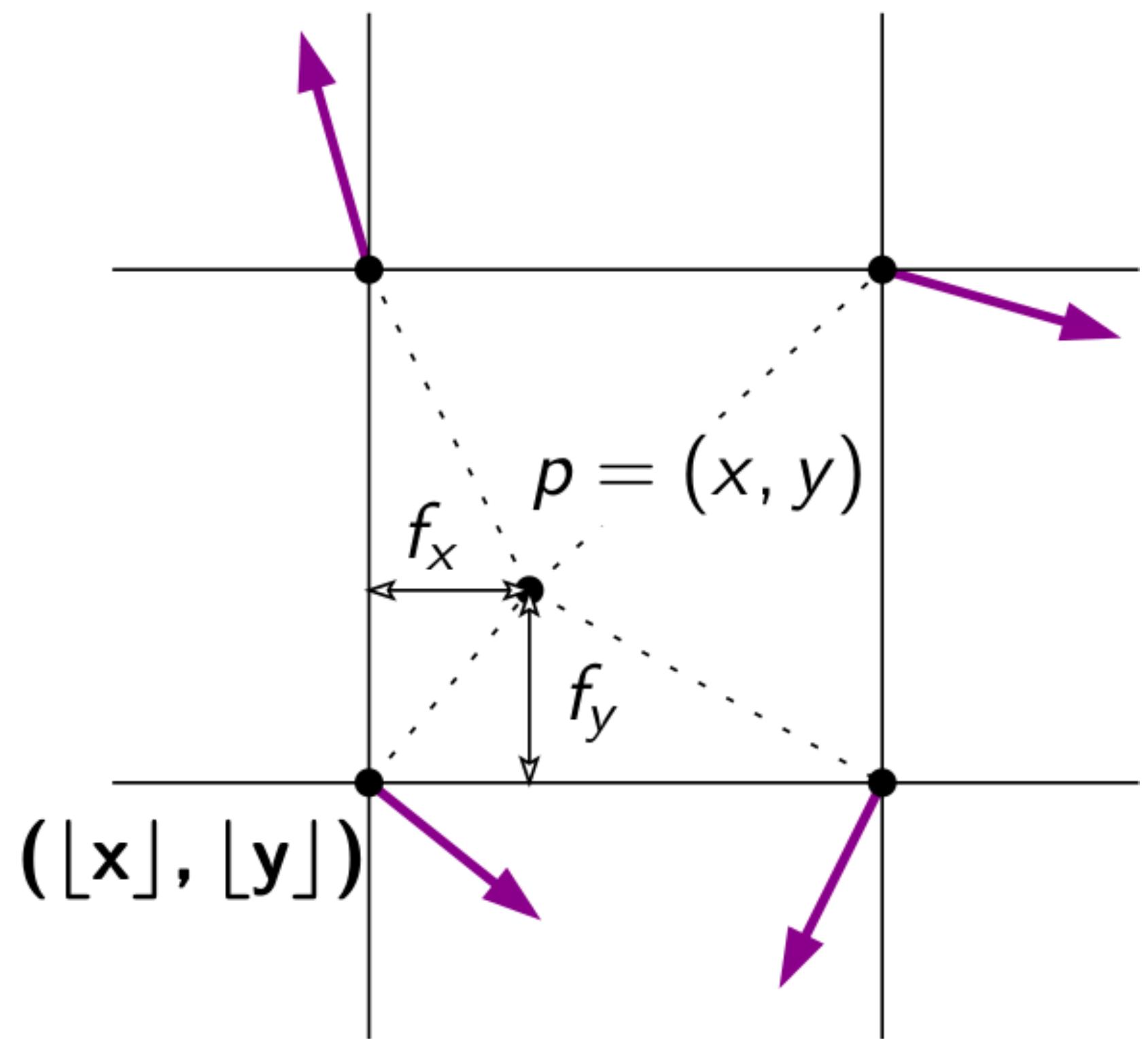


University
of Victoria

Computer Science

Perlin Noise

- Based on a regular lattice, with a 2D random vector \mathbf{v} defined on every corner
- Algorithm:
 - Given a point \mathbf{p} in 2D find the 4 lattice corners $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4$
 - Compute $\mathbf{v}(\mathbf{c}_i) \cdot (\mathbf{p}-\mathbf{c}_i)$ for every corner
 - Return the bilinear interpolation of the dot products evaluated at \mathbf{p}



```

// Function to linearly interpolate between a0 and a1
// Weight w should be in the range [0.0, 1.0]
float lerp(float a0, float a1, float w) {
    return (1.0 - w)*a0 + w*a1;

    // as an alternative, this slightly faster equivalent formula can be used:
    // return a0 + w*(a1 - a0);
}

// Computes the dot product of the distance and gradient vectors.
float dotGridGradient(int ix, int iy, float x, float y) {

    // Precomputed (or otherwise) gradient vectors at each grid node
    extern float Gradient[IYMAX][IXMAX][2];

    // Compute the distance vector
    float dx = x - (float)ix;
    float dy = y - (float)iy;

    // Compute the dot-product
    return (dx*Gradient[iy][ix][0] + dy*Gradient[iy][ix][1]);
}

```

```

// Compute Perlin noise at coordinates x, y
float perlin(float x, float y) {

    // Determine grid cell coordinates
    int x0 = int(x);
    int x1 = x0 + 1;
    int y0 = int(y);
    int y1 = y0 + 1;

    // Determine interpolation weights
    // Could also use higher order polynomial/s-curve here
    float sx = x - (float)x0;
    float sy = y - (float)y0;

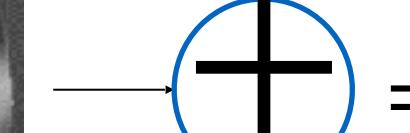
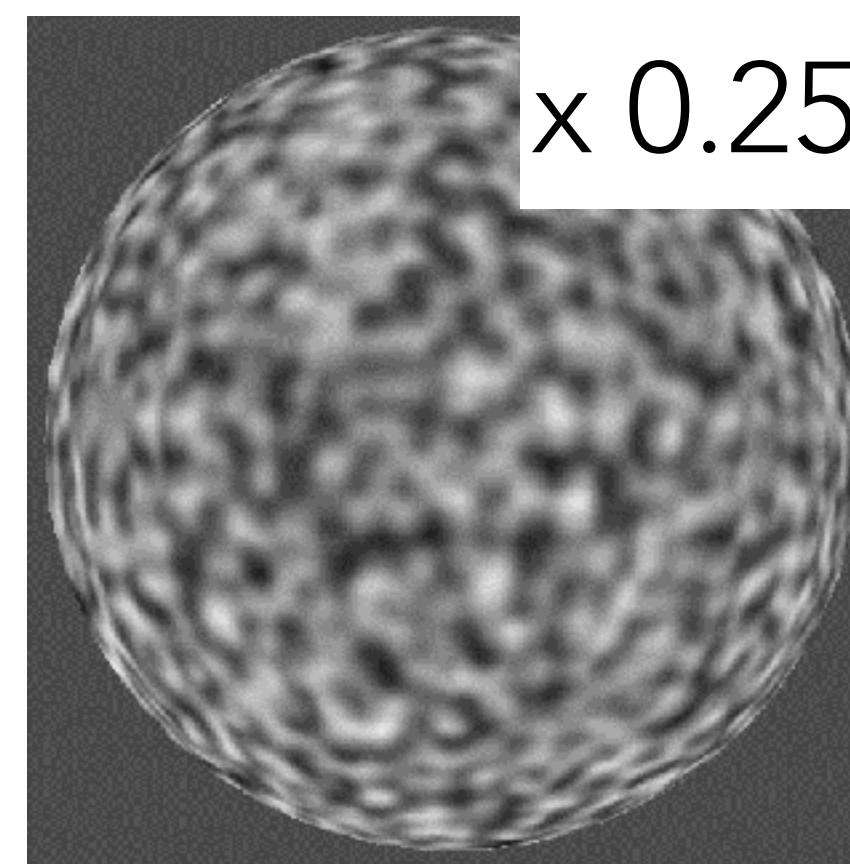
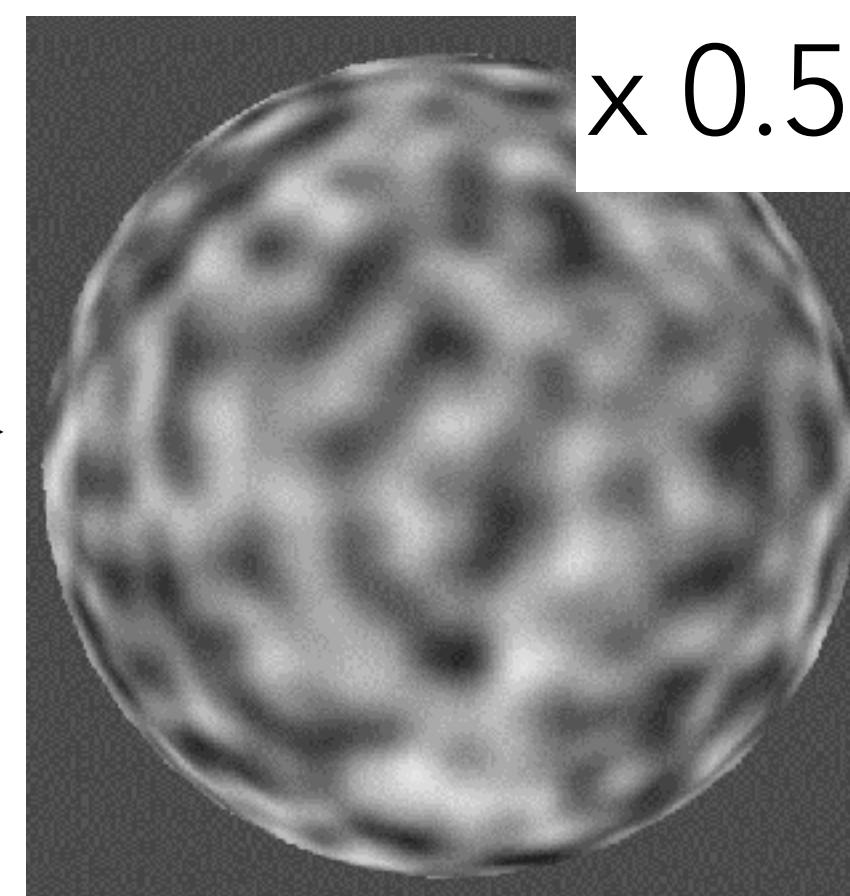
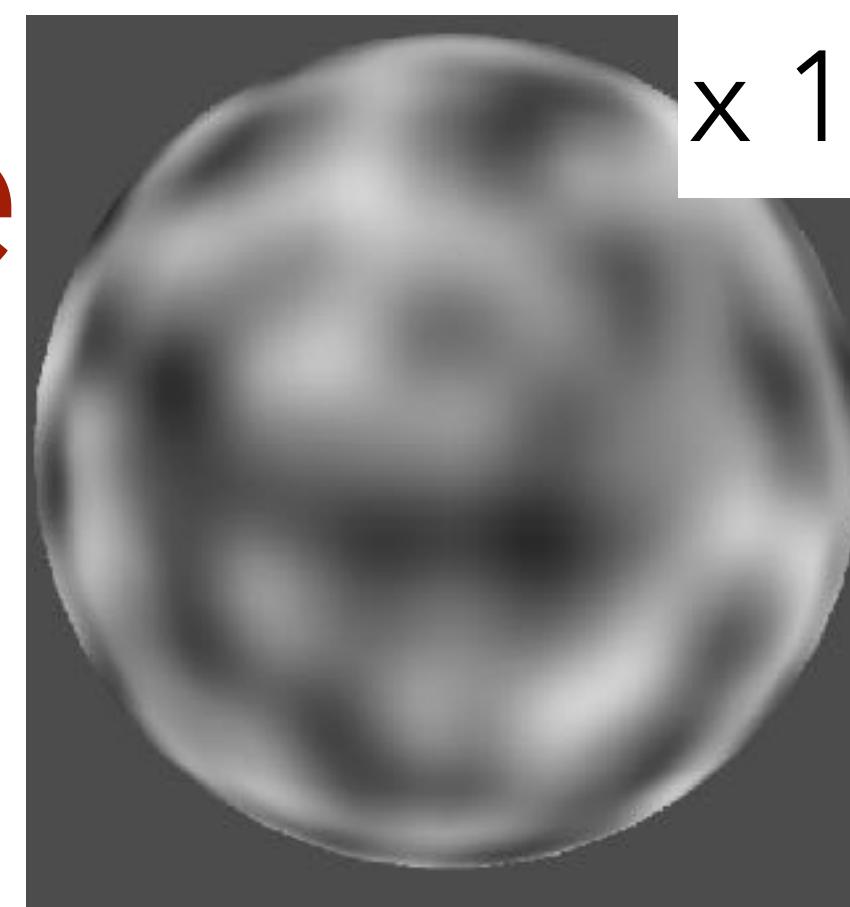
    // Interpolate between grid point gradients
    float n0, n1, ix0, ix1, value;
    n0 = dotGridGradient(x0, y0, x, y);
    n1 = dotGridGradient(x1, y0, x, y);
    ix0 = lerp(n0, n1, sx);
    n0 = dotGridGradient(x0, y1, x, y);
    n1 = dotGridGradient(x1, y1, x, y);
    ix1 = lerp(n0, n1, sx);
    value = lerp(ix0, ix1, sy);

    return value;
}

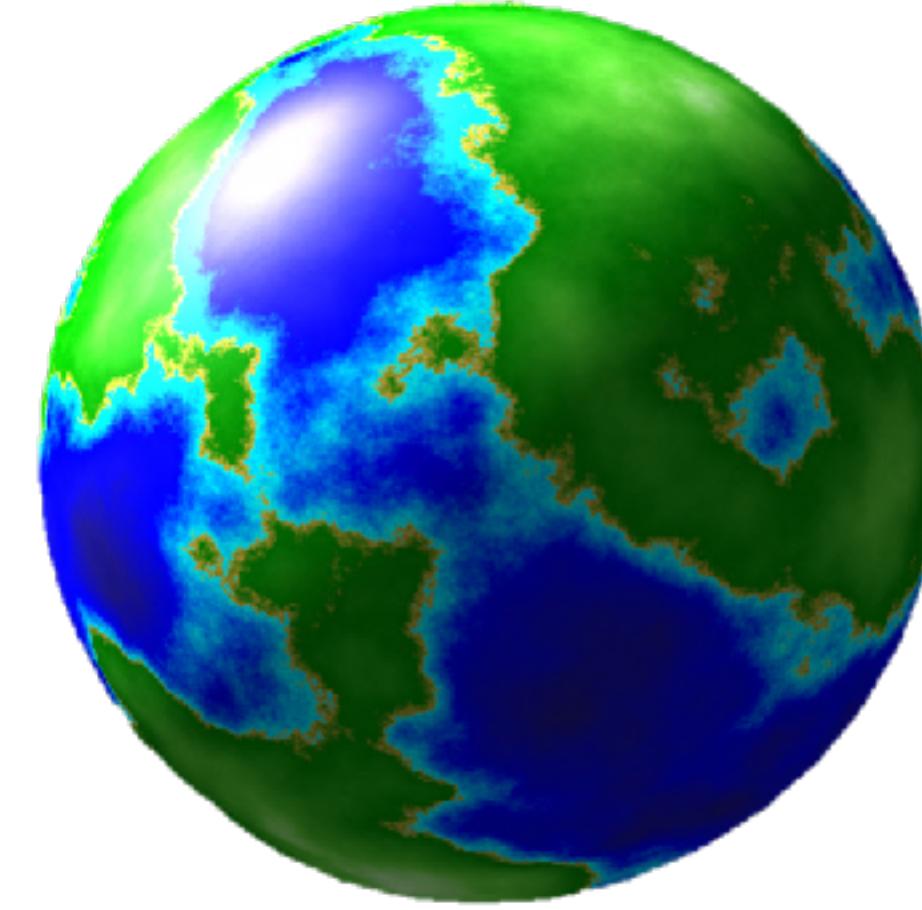
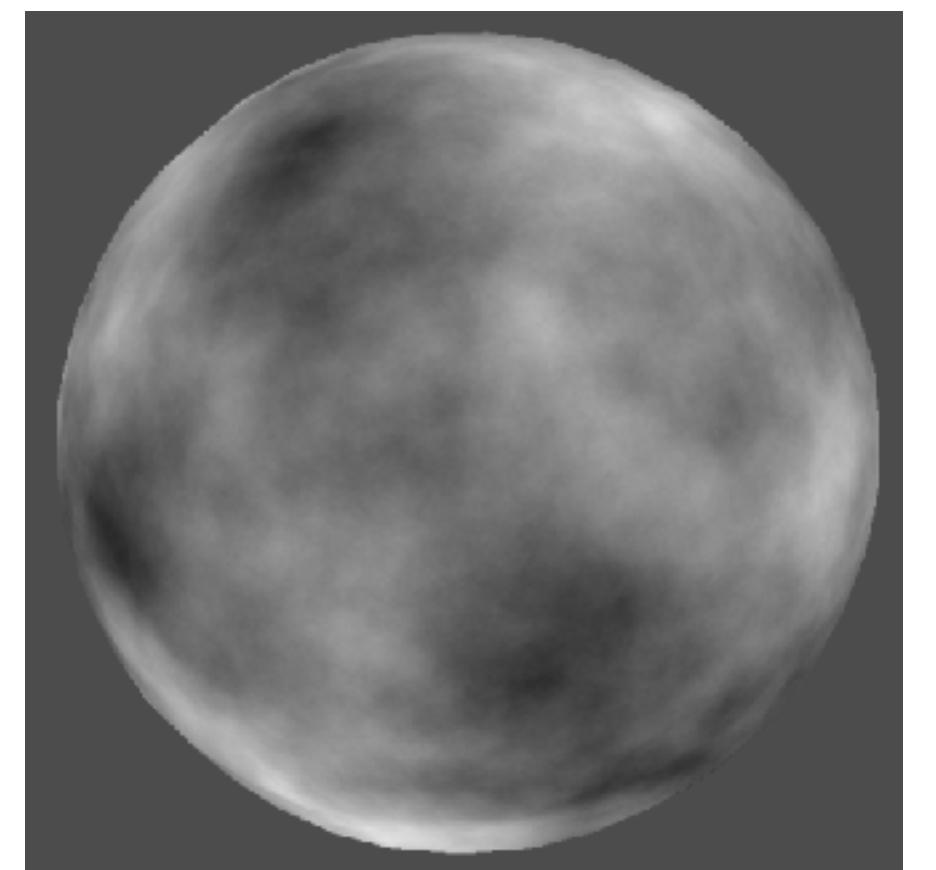
```

Combine

42
seed



=



and add colors!

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



University
of Victoria

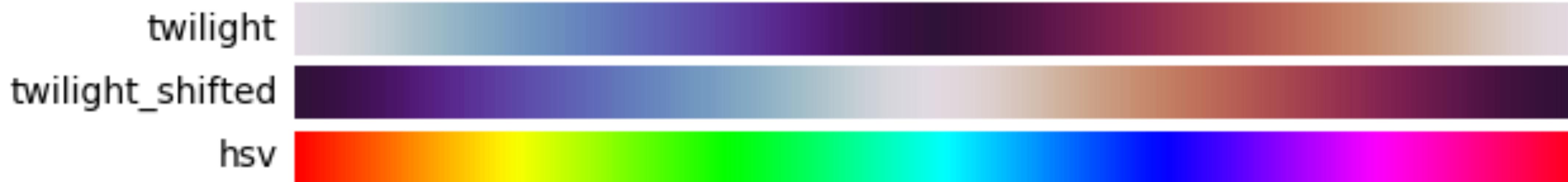
Computer Science

Color Maps

Perceptually Uniform Sequential colormaps



Cyclic colormaps



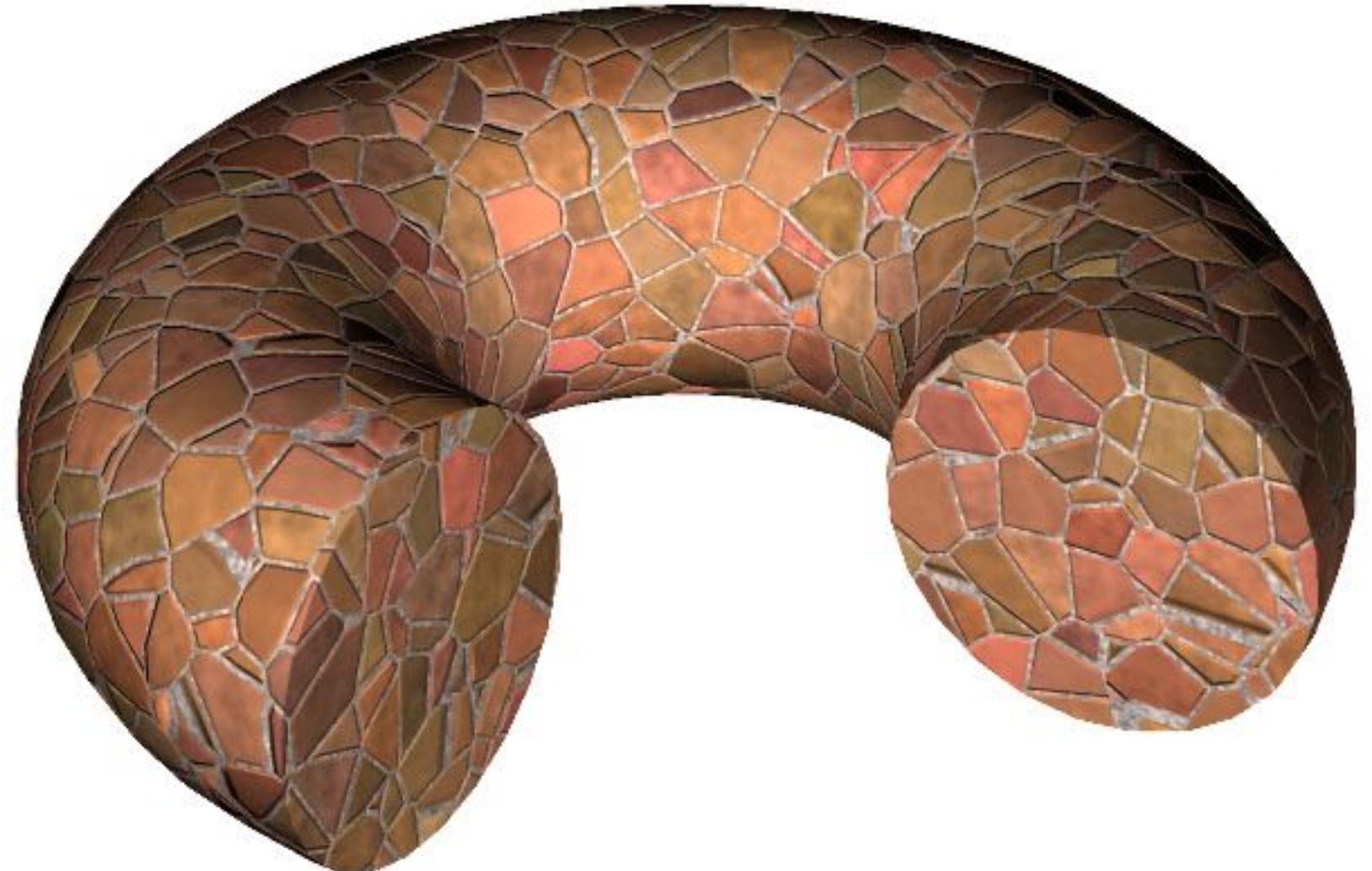
From: <https://matplotlib.org/tutorials/colors/colormaps.html>

CSC 305 - Introduction to Computer Graphics - Teseo Schneider

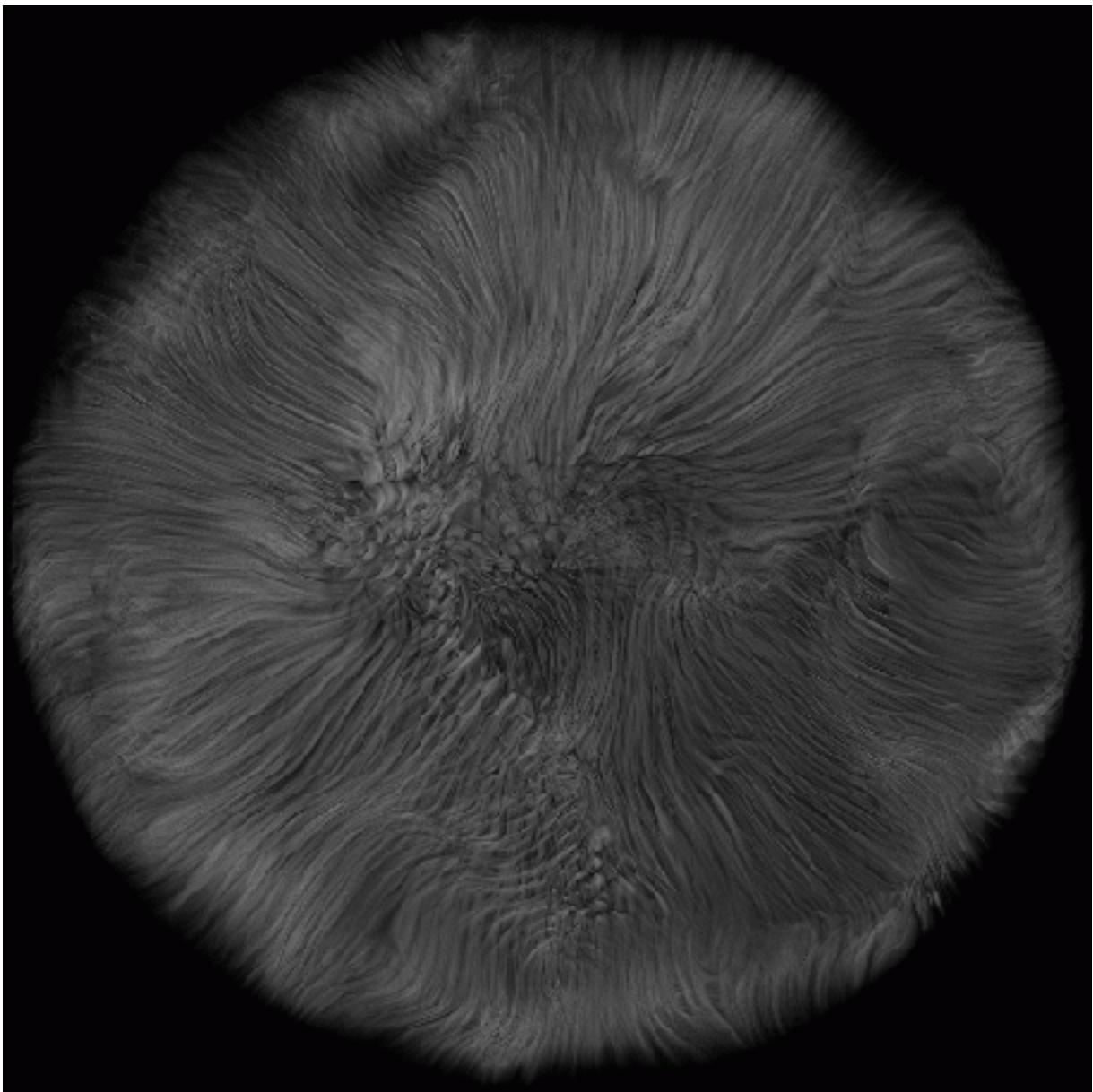
Why are Procedural Textures Popular?

- No need to store them
 - Reduce GPU transfer for rasterization
 - Reduce scene size for raytracing
- They can be evaluated at any point
 - computation-bound instead of memory-bound
- Infinite resolution (similar to vector graphics)

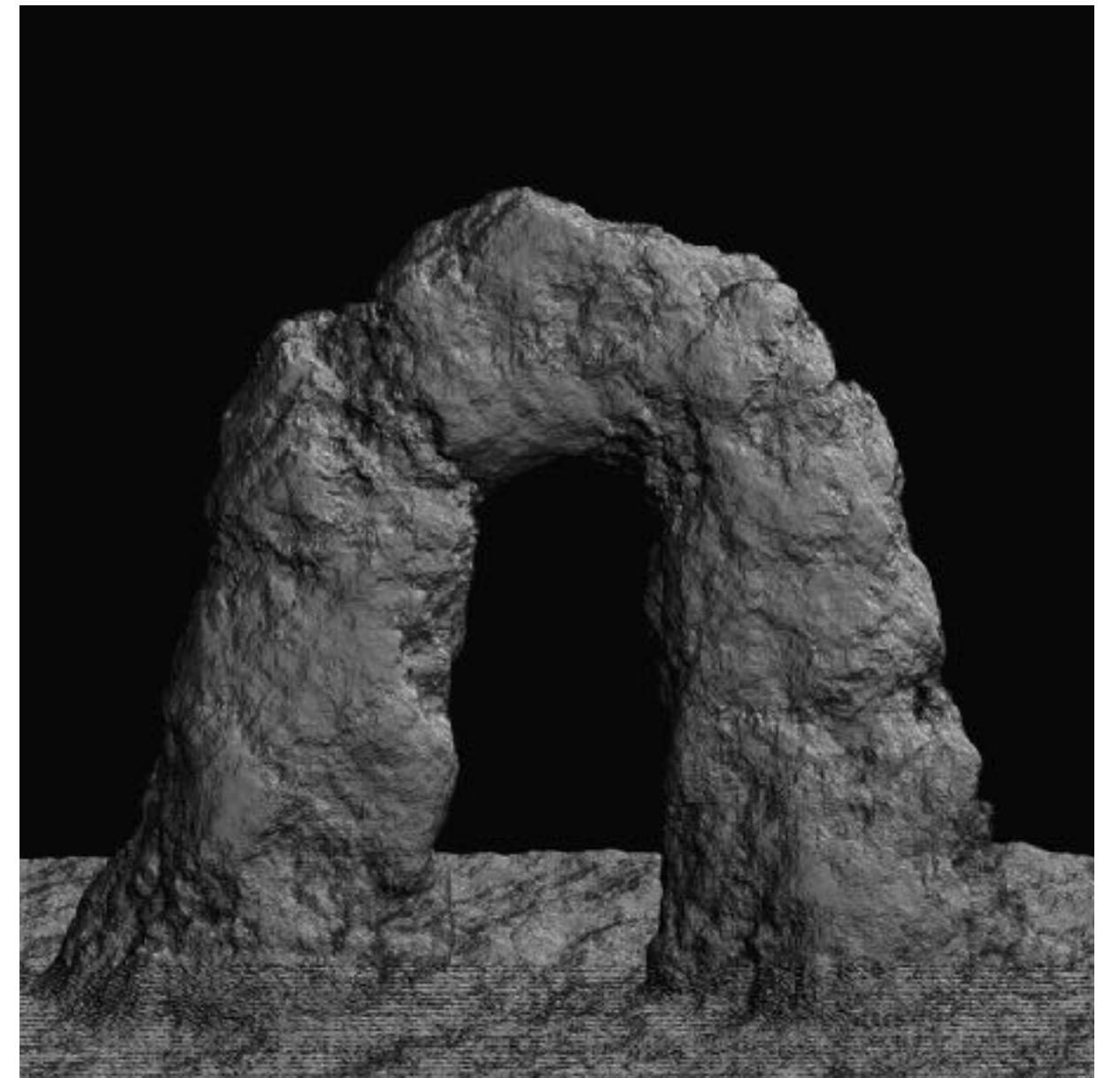
Solid Textures and Hypertextures



Cellular Noise [Worley 1996]



Hypertextures [Perlin 1989]



Implicit Modeling

Acknowledgement: Marco Tarini

CSC 305 - Introduction to Computer Graphics - Teseo Schneider

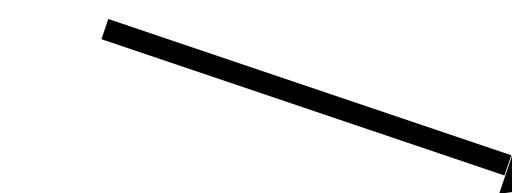


University
of Victoria

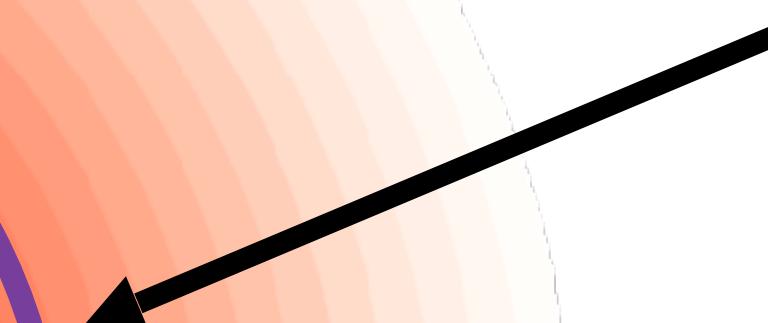
| Computer Science

$$S = \{x \in \mathbb{R}^n \mid f(x) = d\}$$

distance field

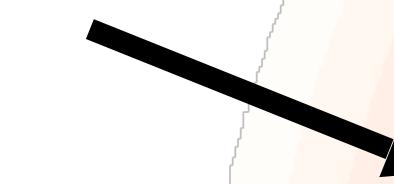


iso-surface



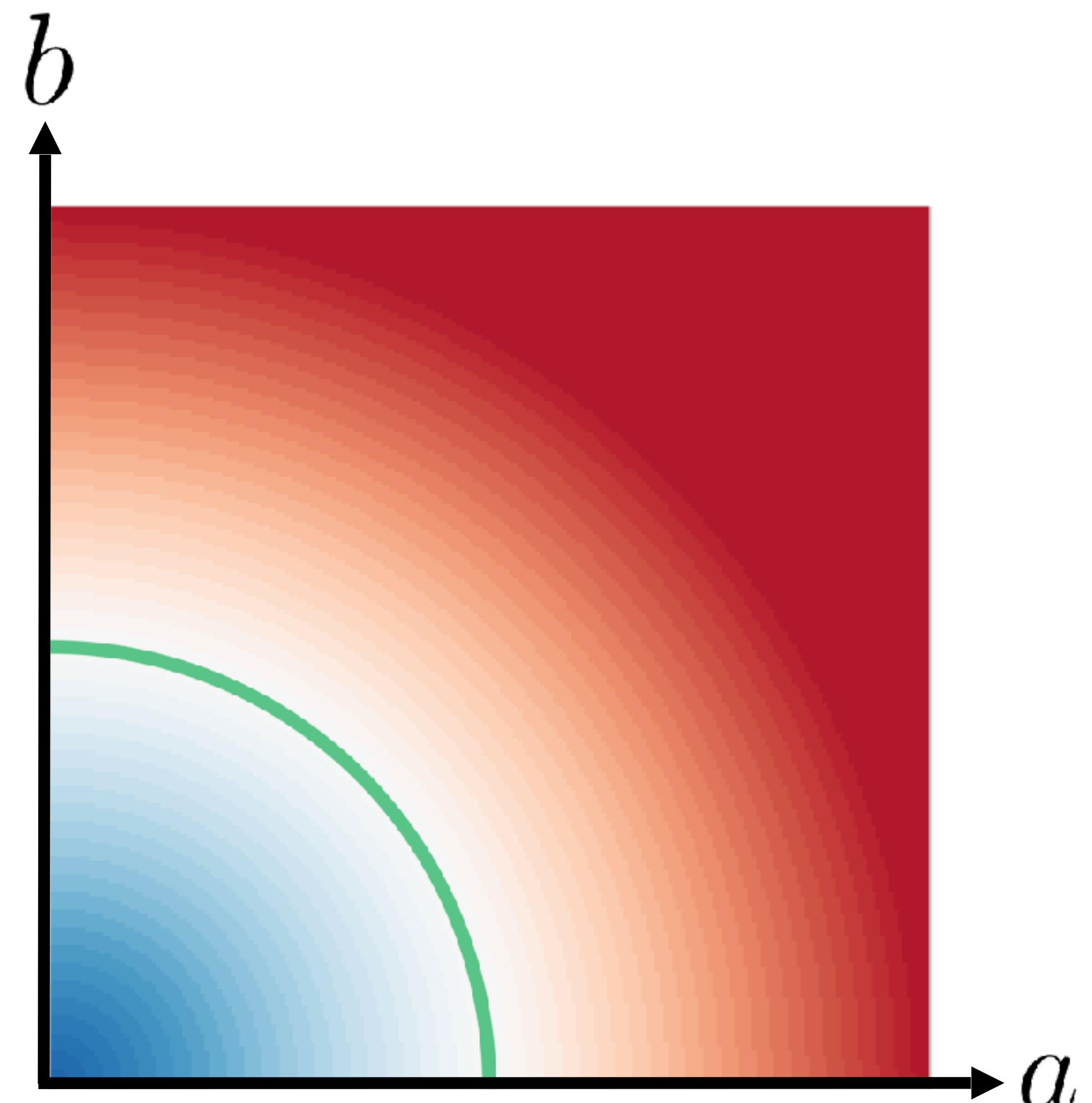
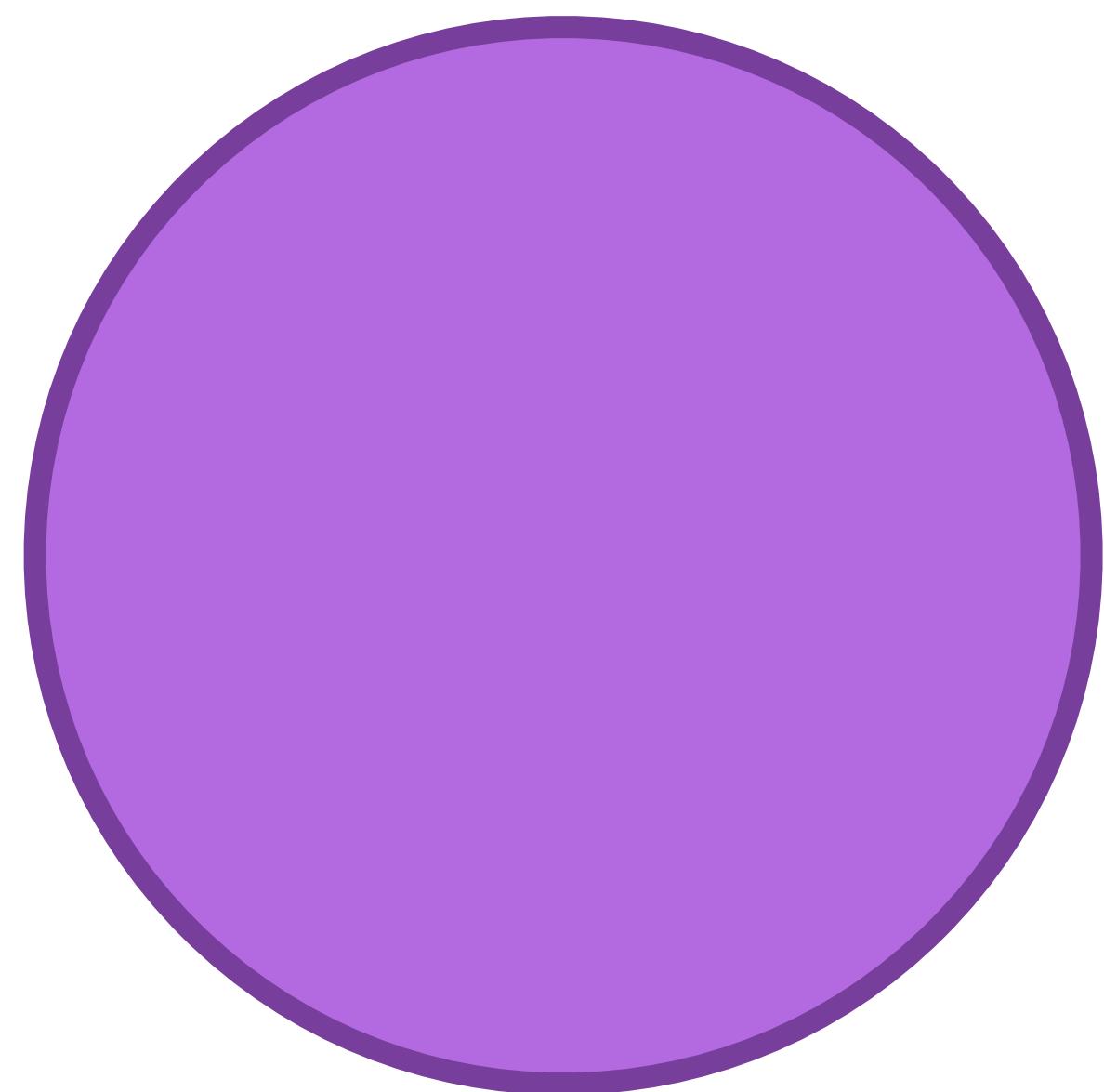
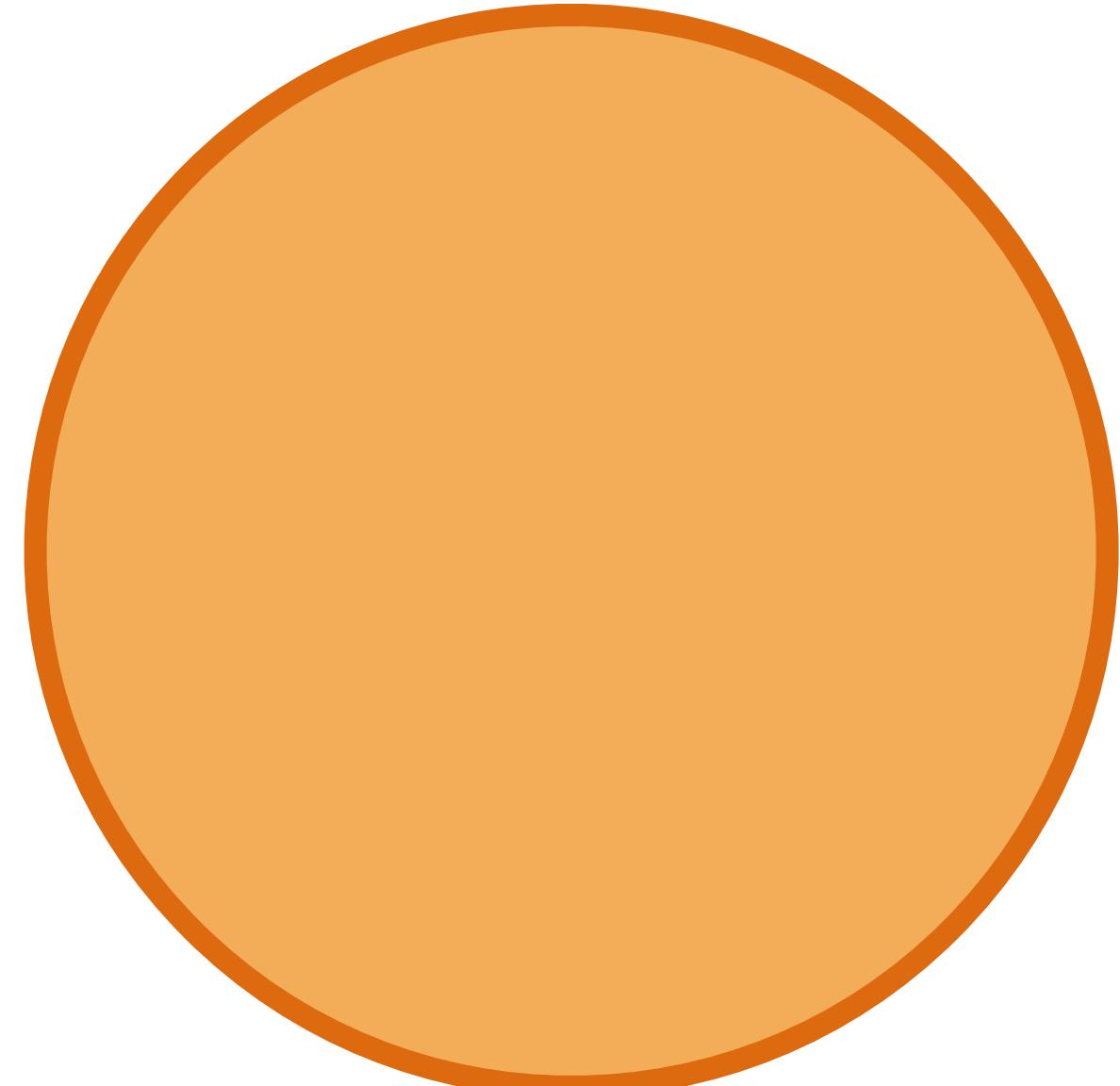
$$S = \{x \in \mathbb{R}^n \mid f(x) = d\}$$

distance field

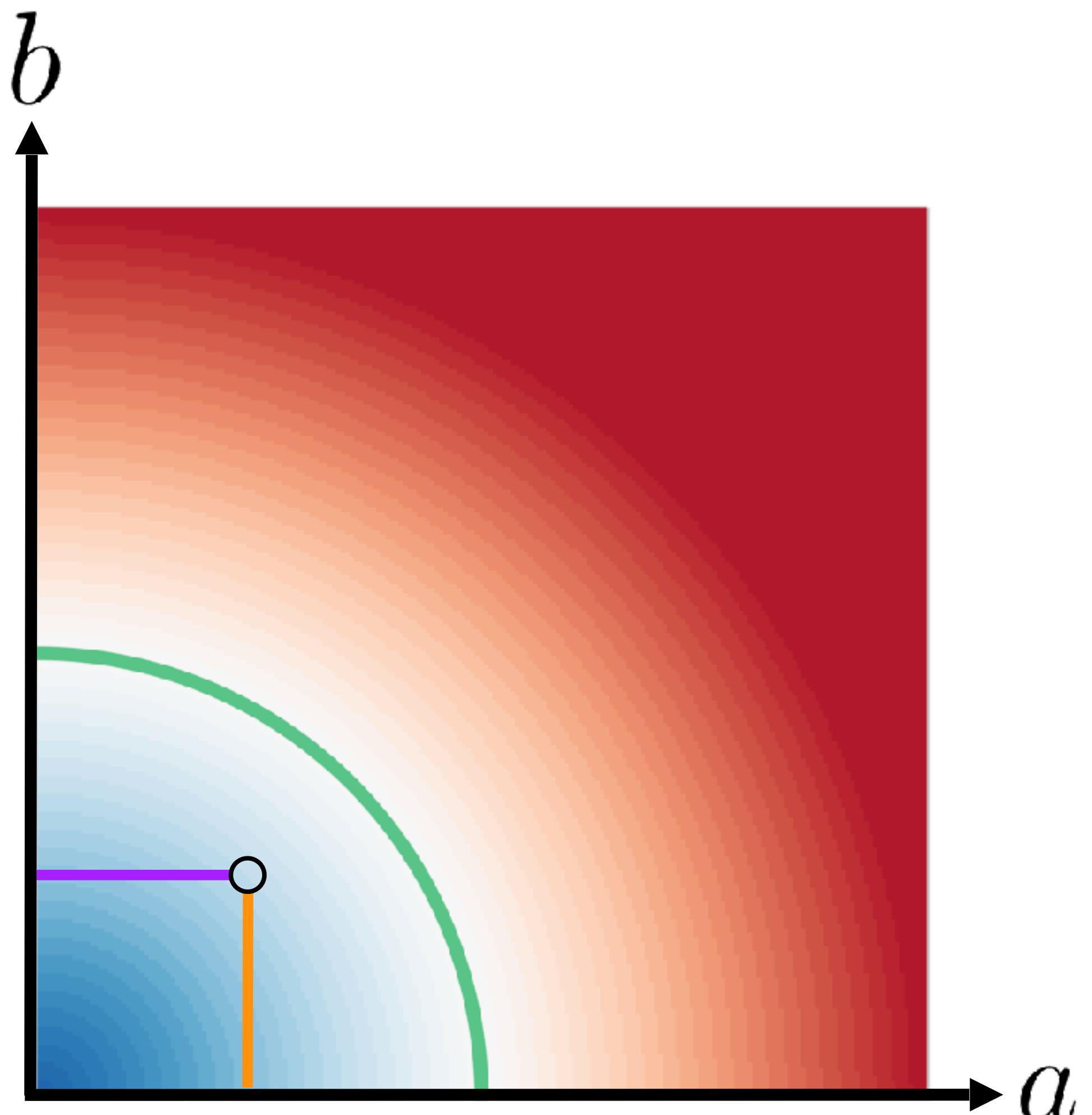
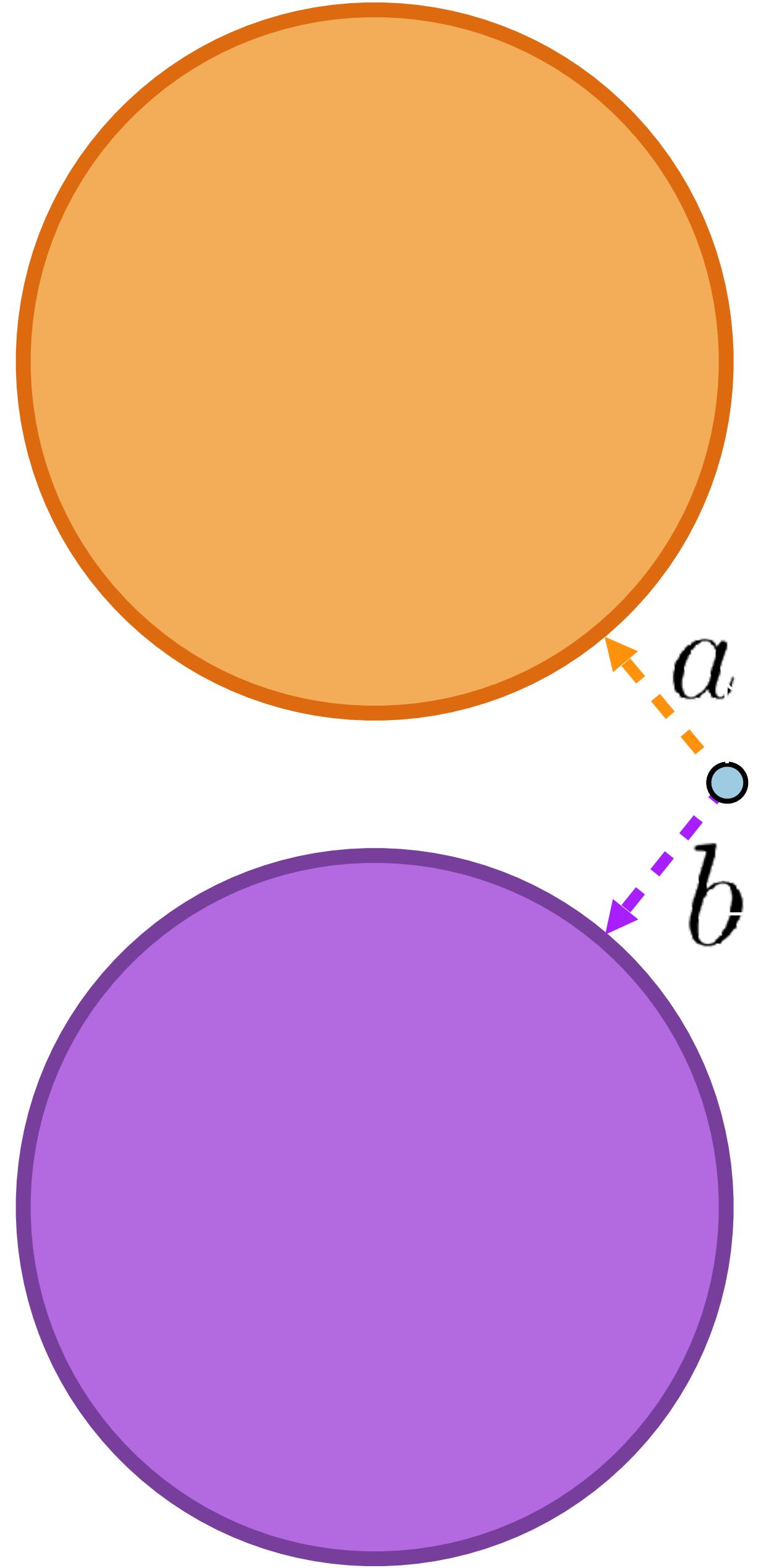


iso-surface

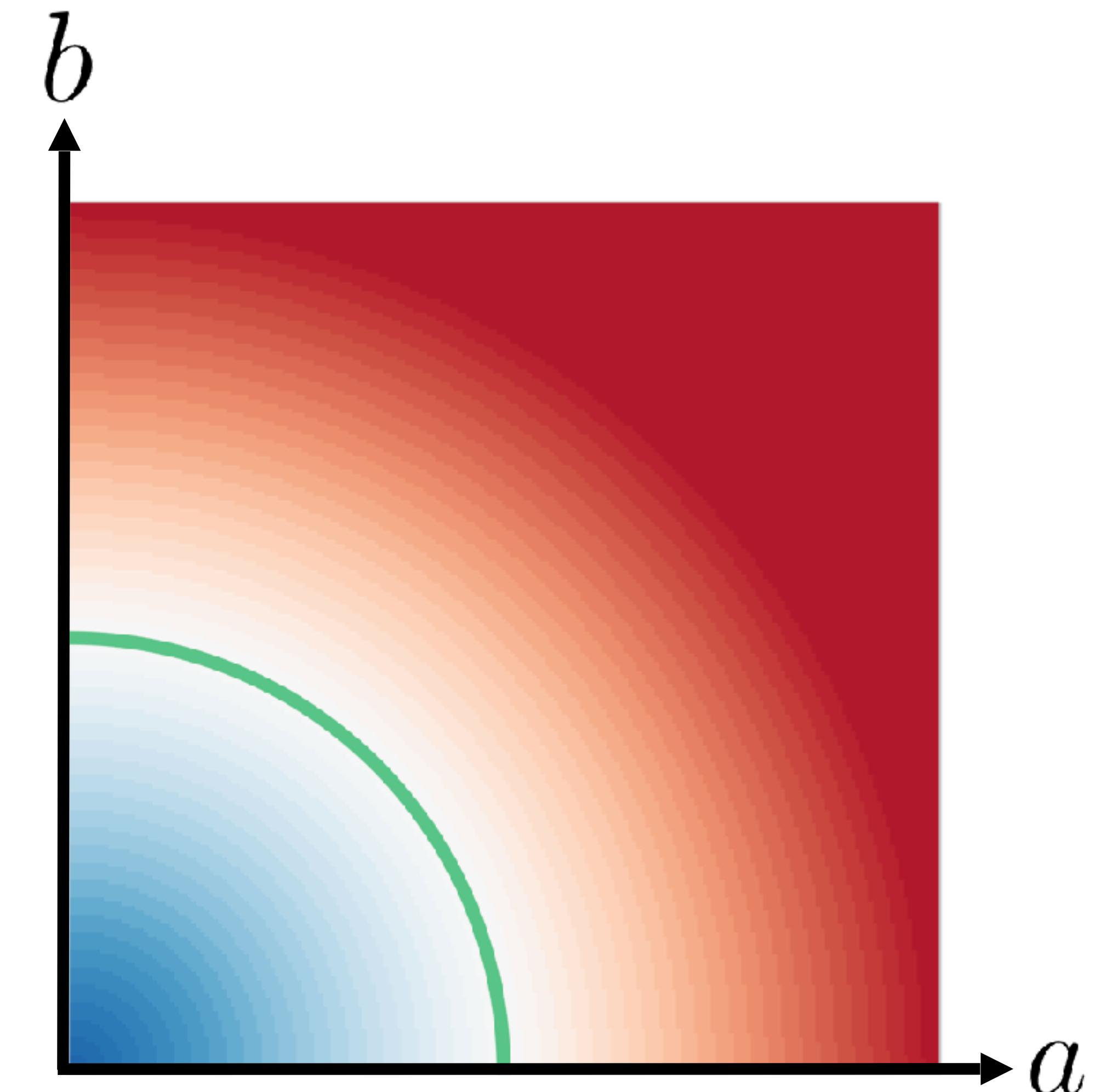
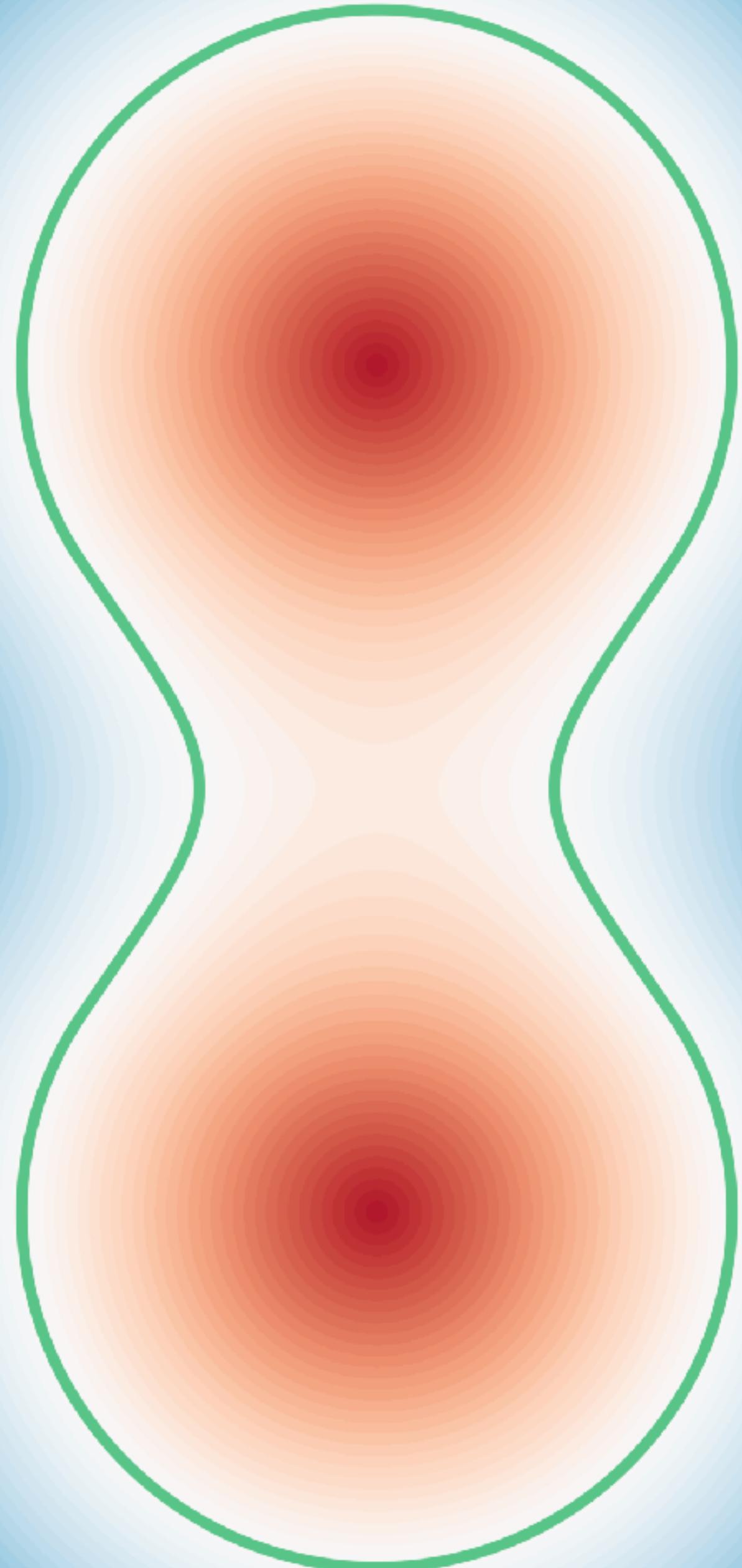




composition operator



composition operator



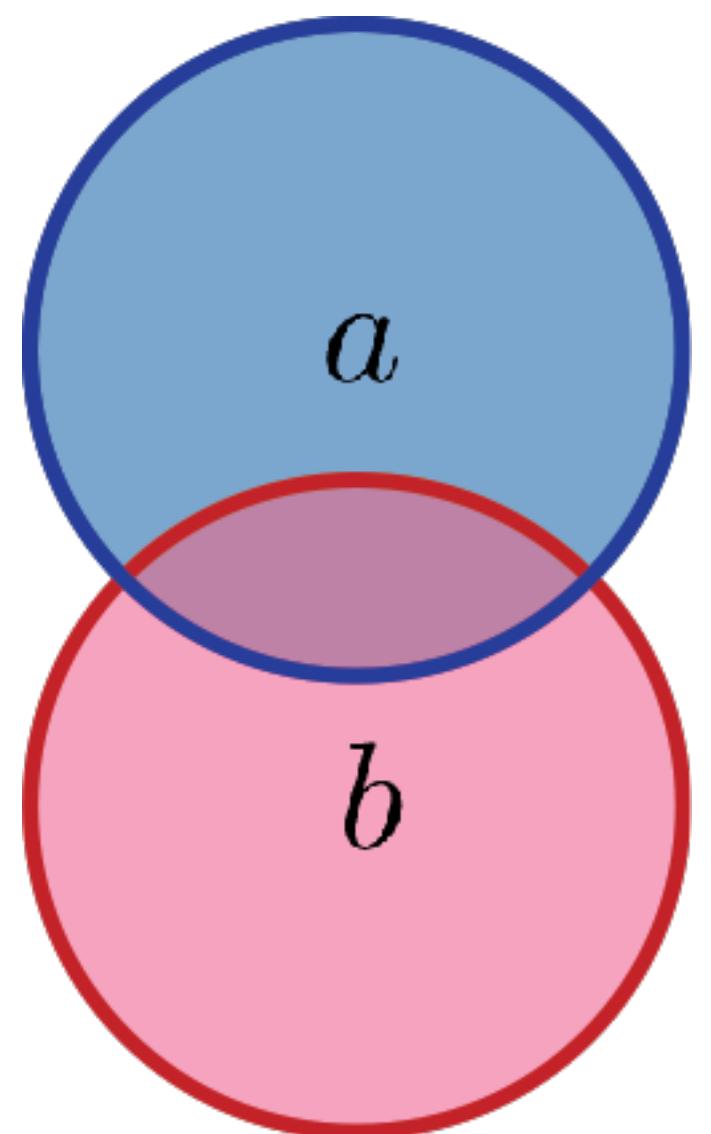
composition operator



University
of Victoria

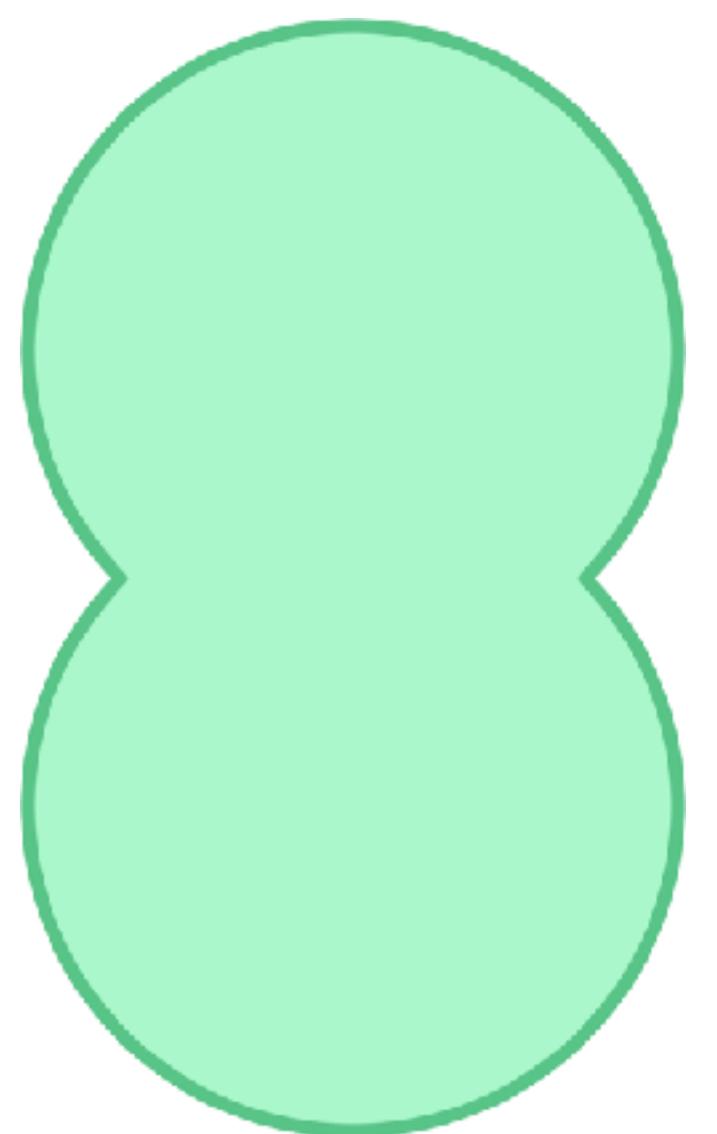
Computer Science

union
[Sabin 1968]

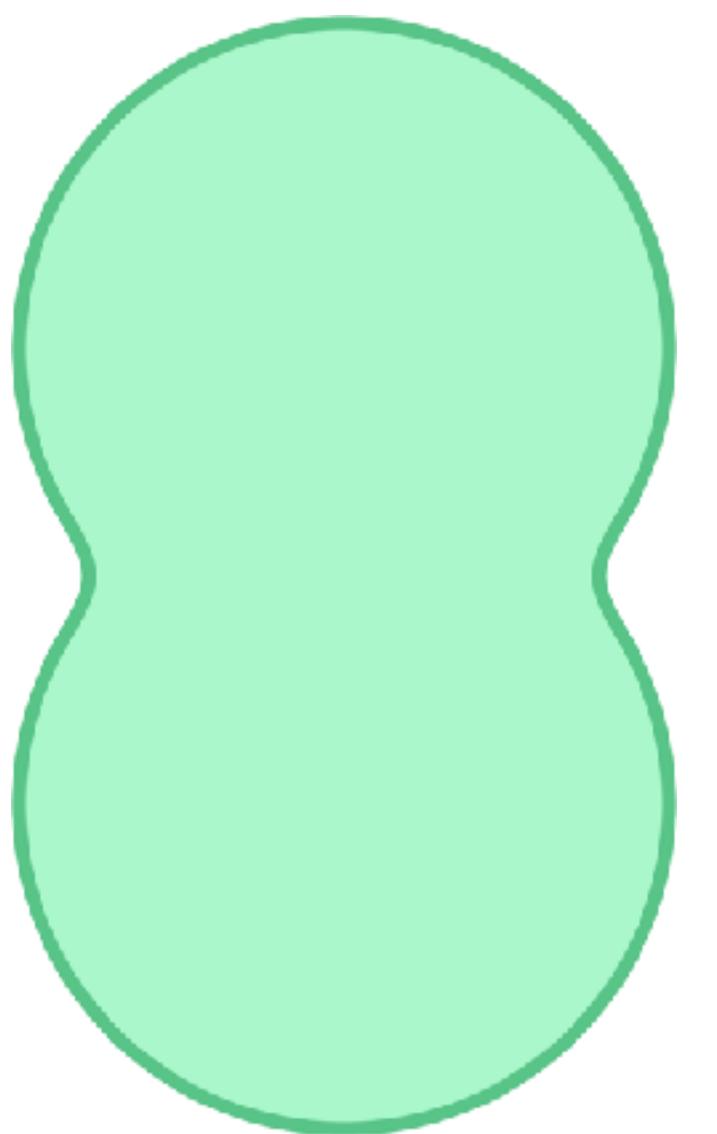


$$\max(a, b)$$

blend
[Blinn 1982]
[Ricci 1973]



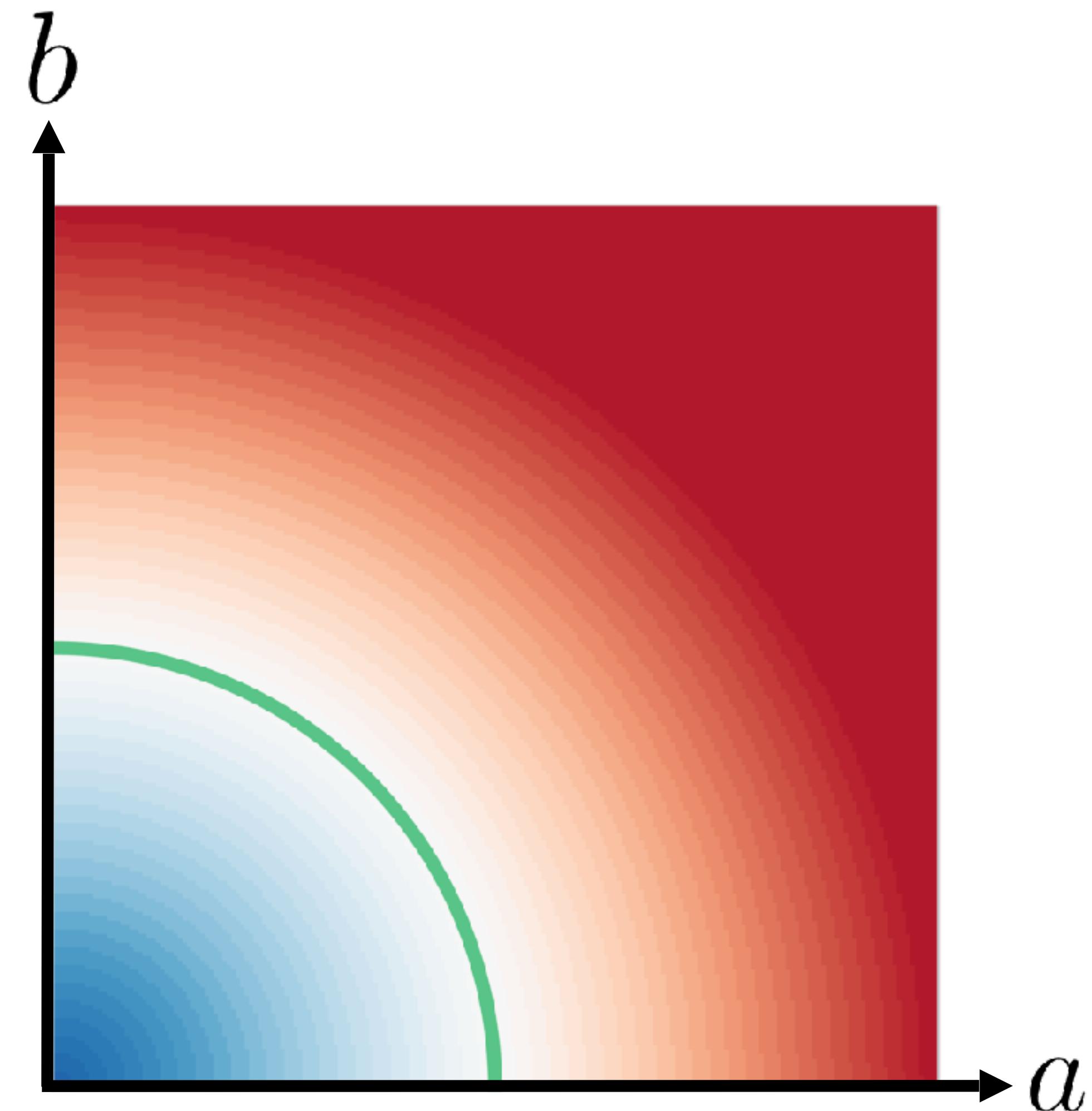
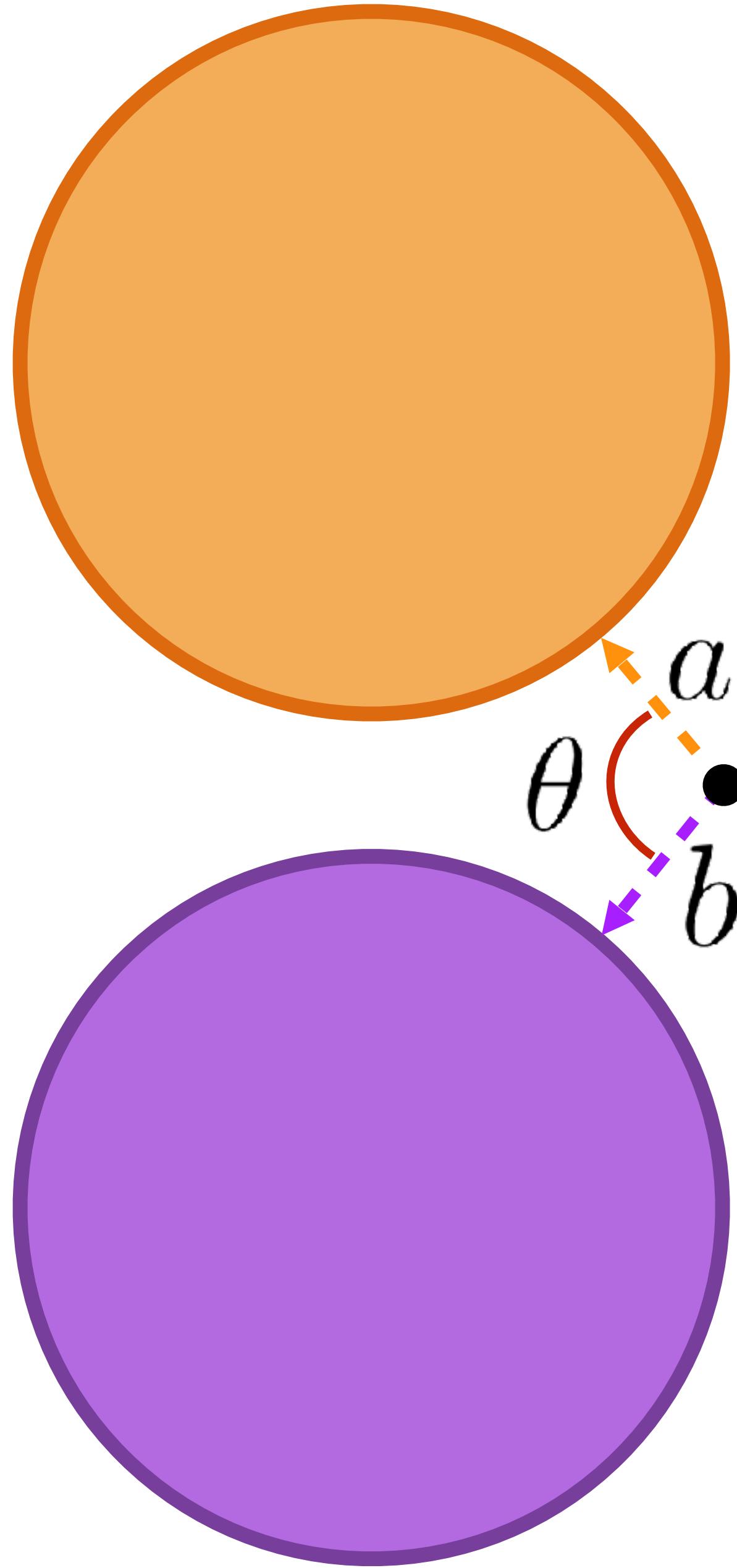
blend
[Blinn 1982]
[Ricci 1973]



contact & bulge
[Cani 1993]

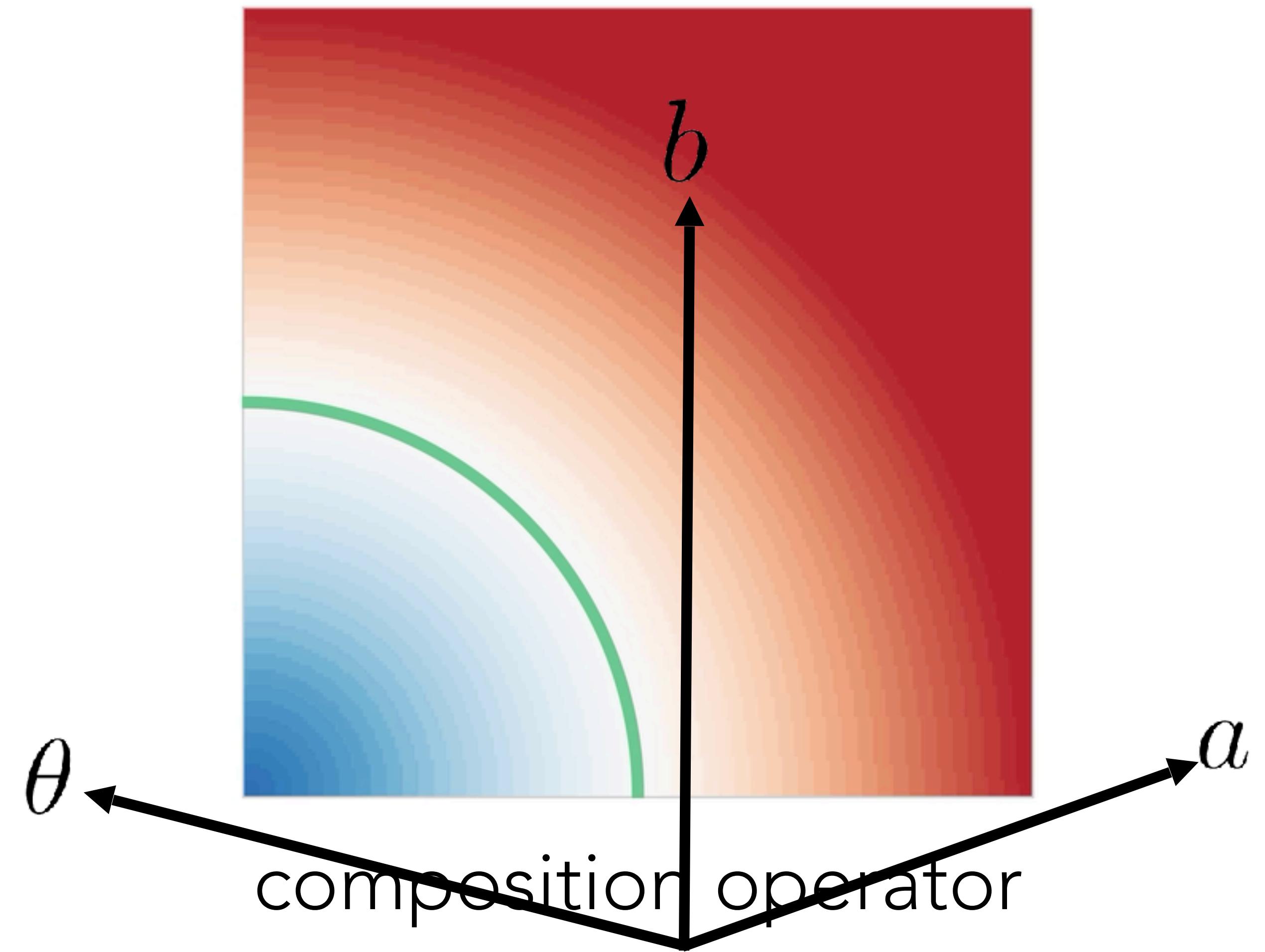
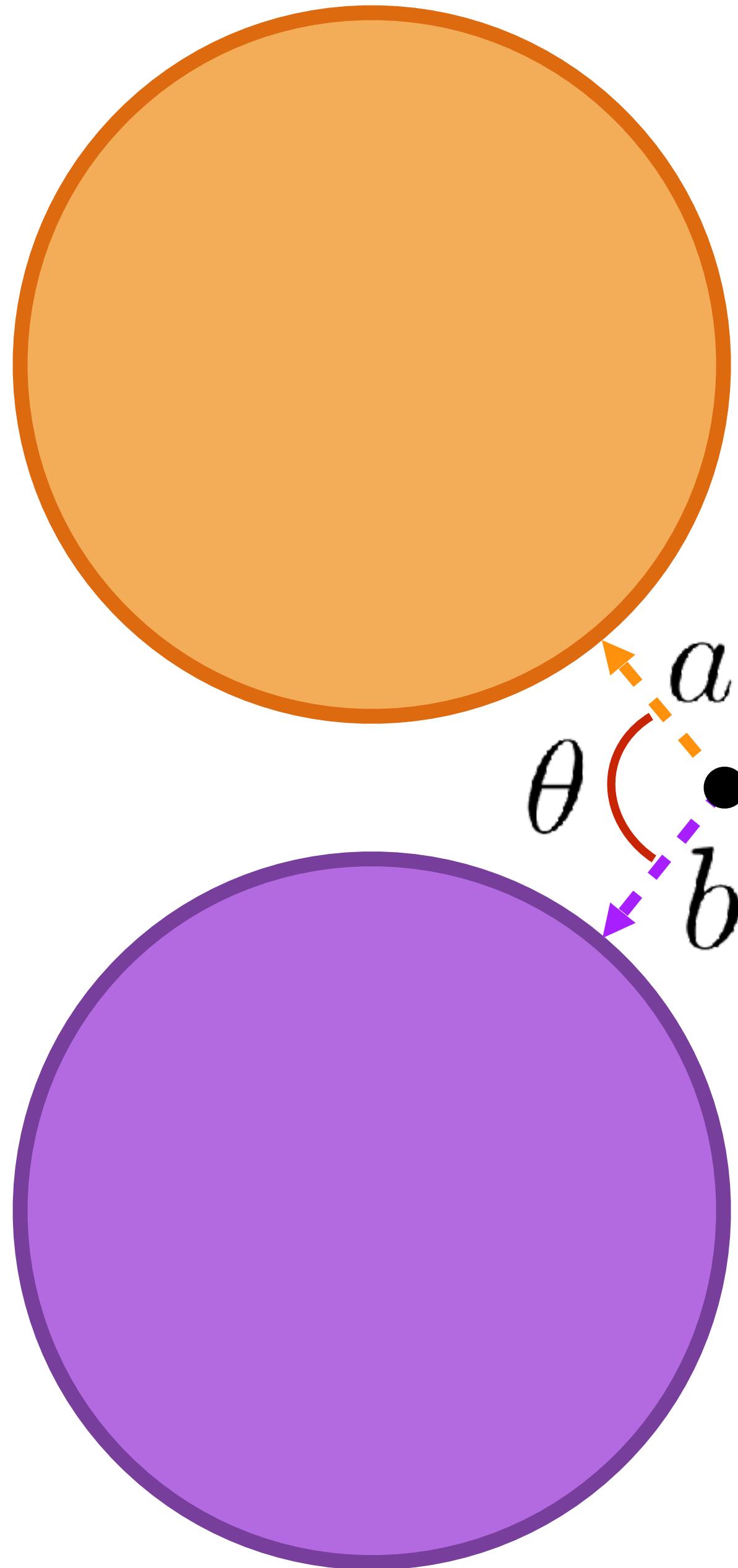
$$\begin{cases} a - b + 1 & \text{if } b > 1 \\ a + h(a, b) & \text{else} \end{cases}$$

[Gourmel et al. 2013]

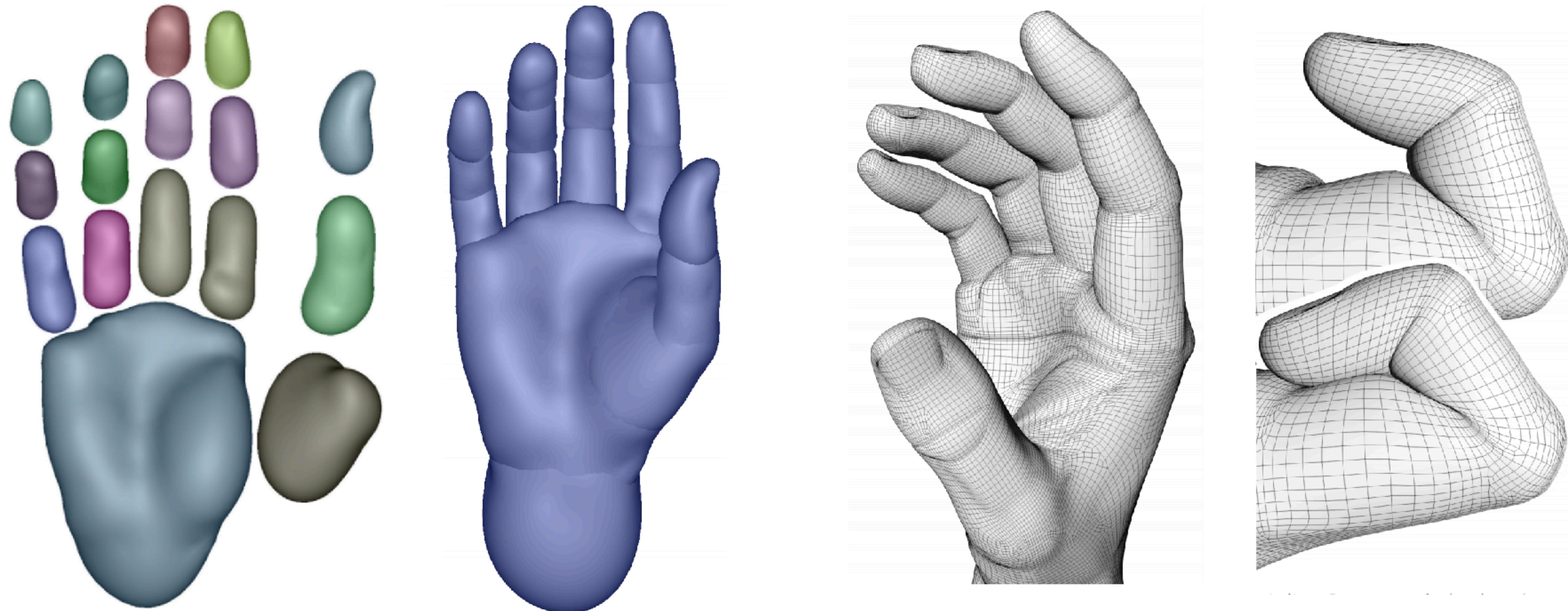


composition operator

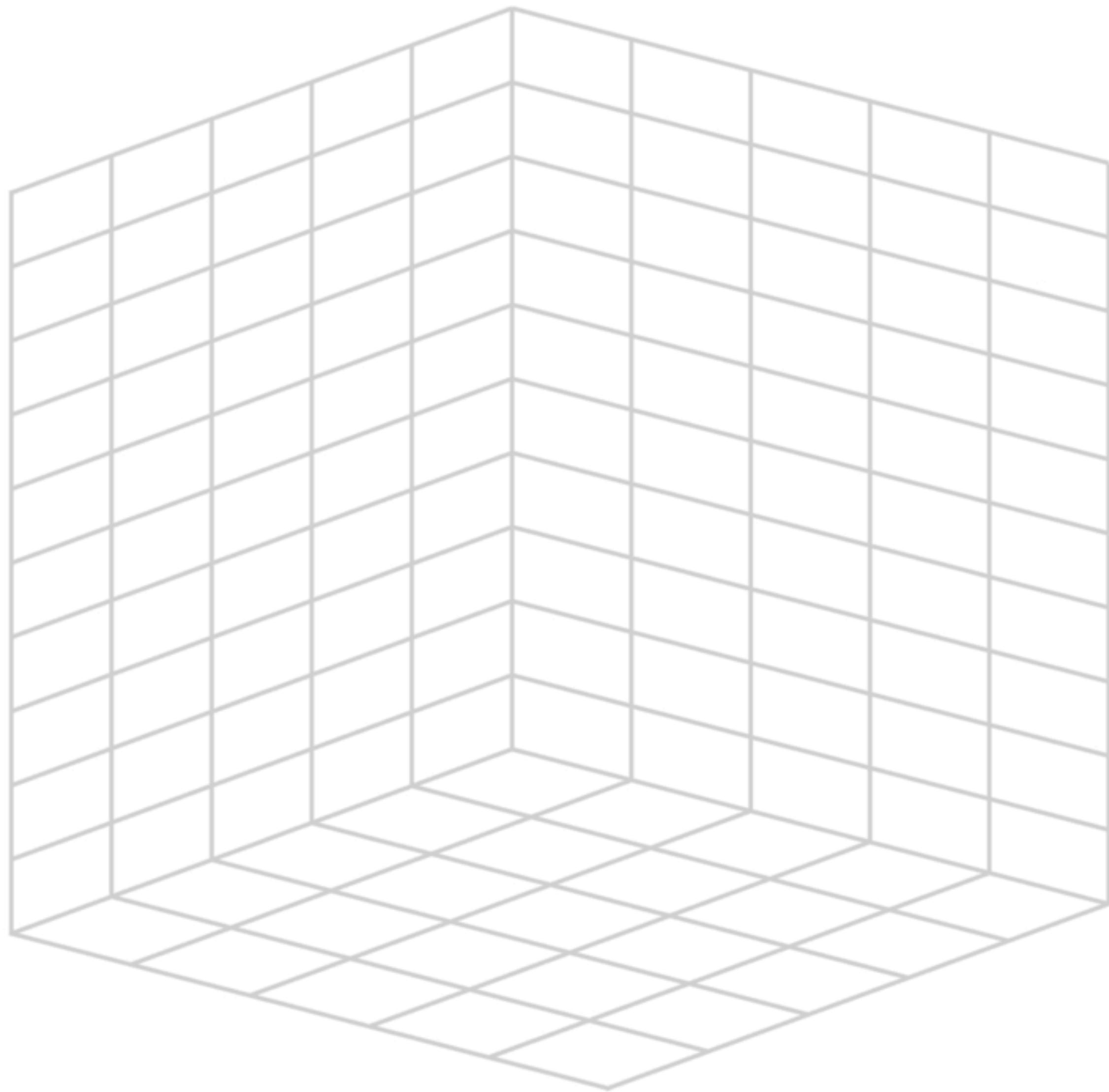
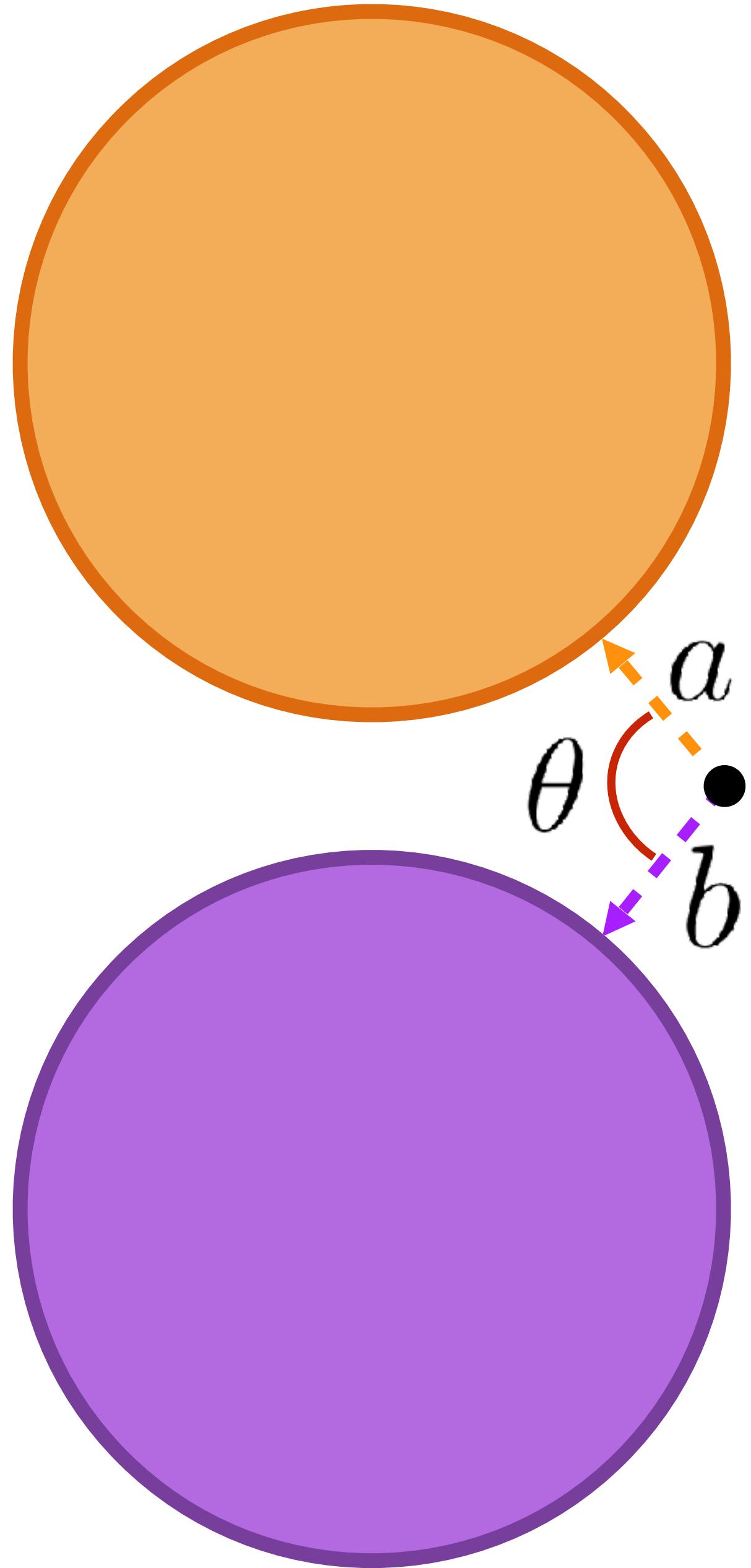
[Gourmel et al. 2013]



Implicit Skinning [Vaillant et al. 2013]

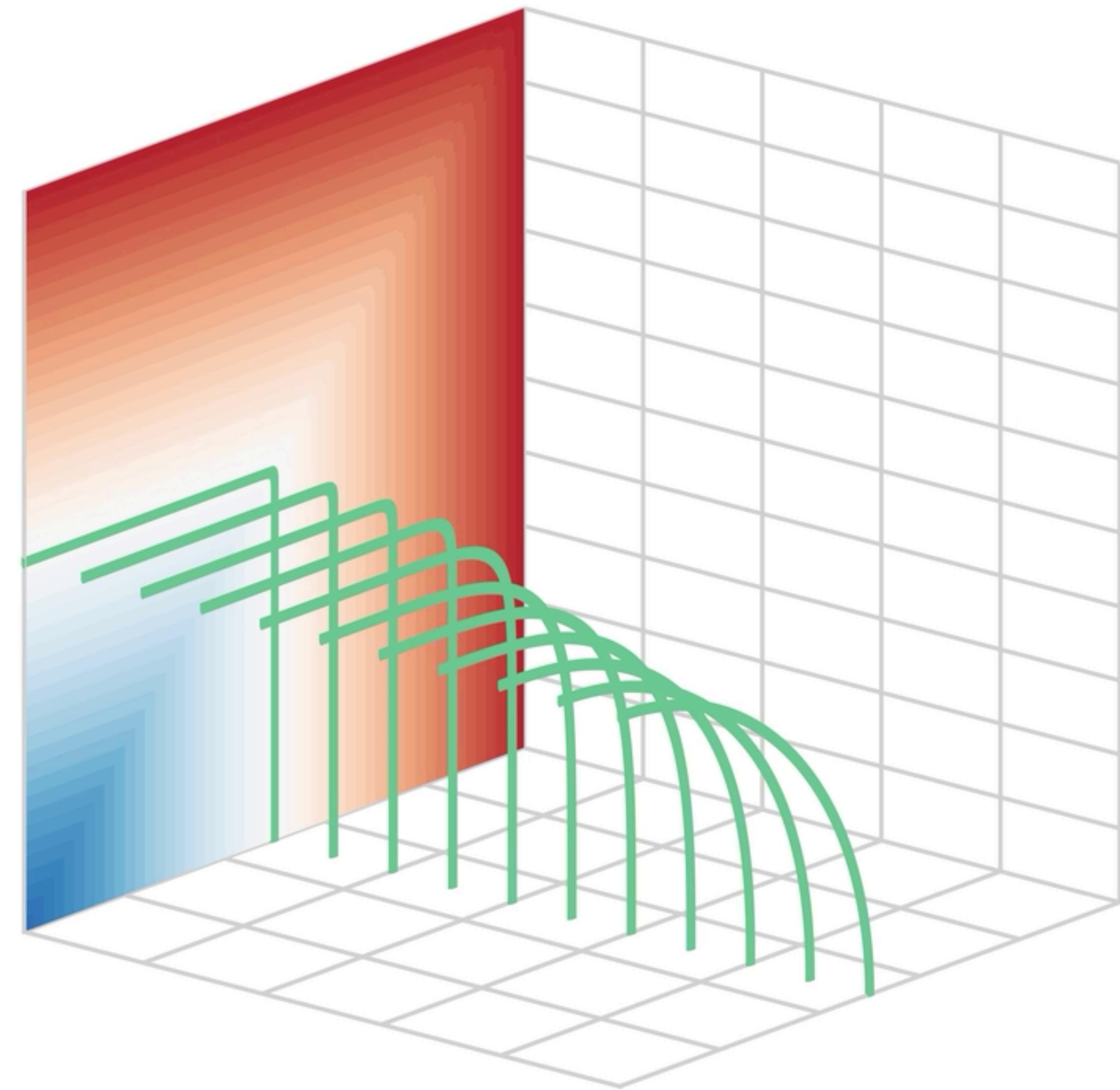
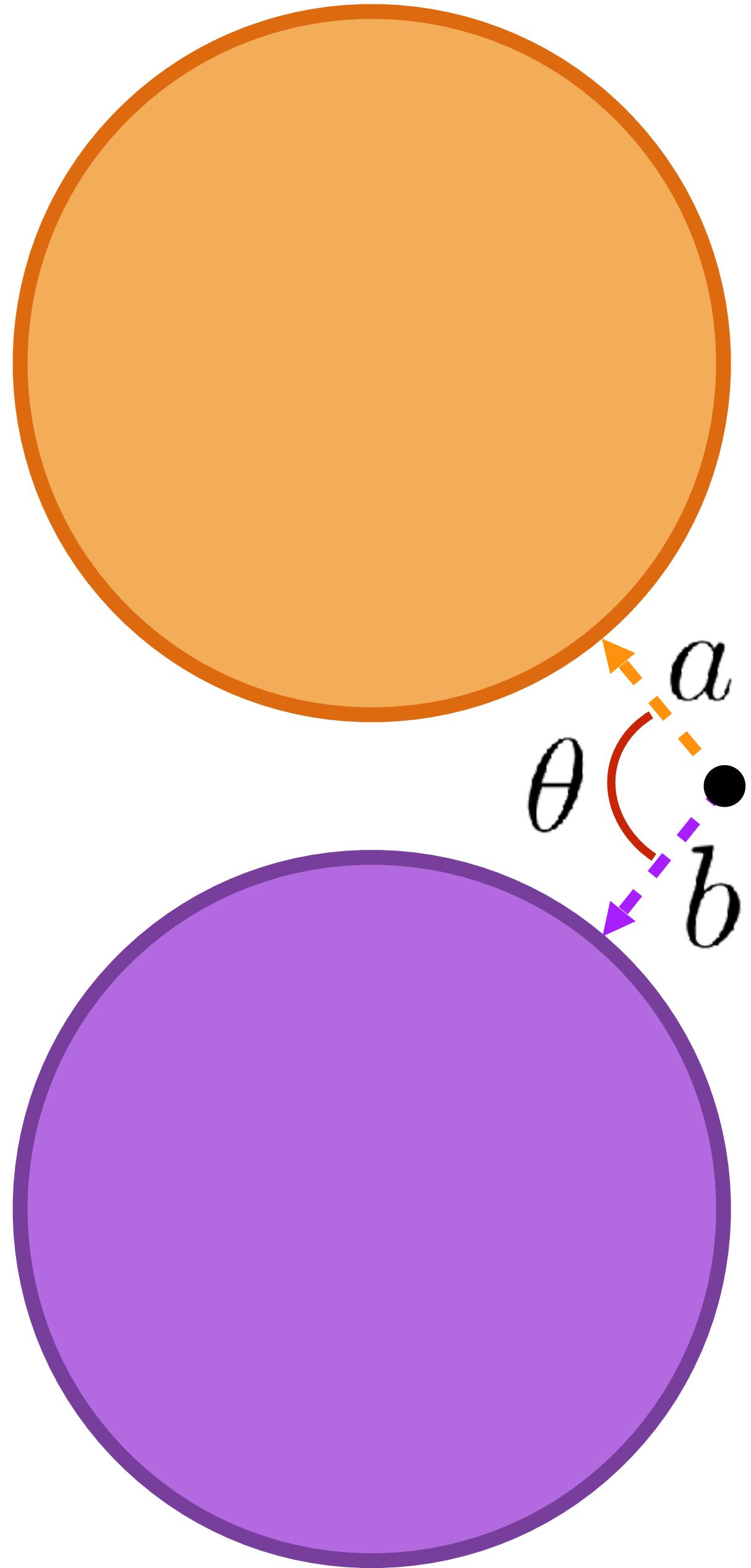


[Vaillant et al. 2014]



composition operator

[Vaillant et al. 2014]



composition operator

Implicit Model

- All points where $f(p) = 0$

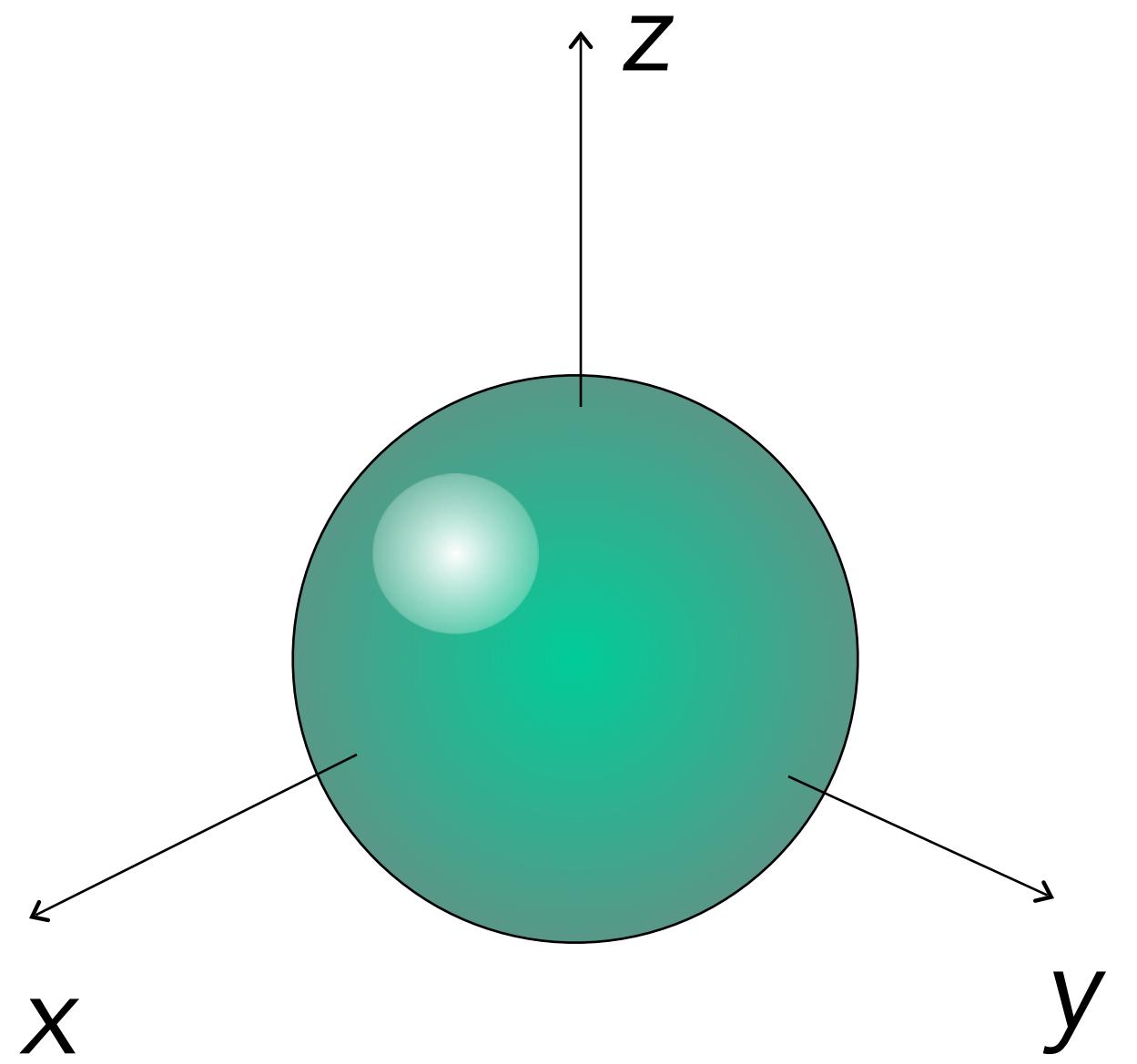
$$f : R^3 \rightarrow R$$

- It directly defines an inside and outside
 - $f(p) < 0 \iff p \text{ inside}$
 - $f(p) > 0 \iff p \text{ outside}$
 - $f(p) = 0 \iff p \text{ on the surface}$
- By construction, it defines a closed, watertight model.



Sphere

$$f \begin{pmatrix} x \\ y \\ z \end{pmatrix} = x^2 + y^2 + z^2 - R^2$$



Categories

- Algebraic surfaces: $f()$ is a polynomial
- Quadratic surfaces : $f()$ degree is 2,
 - aka «[quadrics](#)»
 - important class!
 - simple equations, good expressive power
 - e.g. perfect spheres
- Cubic surfaces: $f()$ degree is 3
- Higher-Order



Implicit Modeling: Pros and Cons?

Pros:

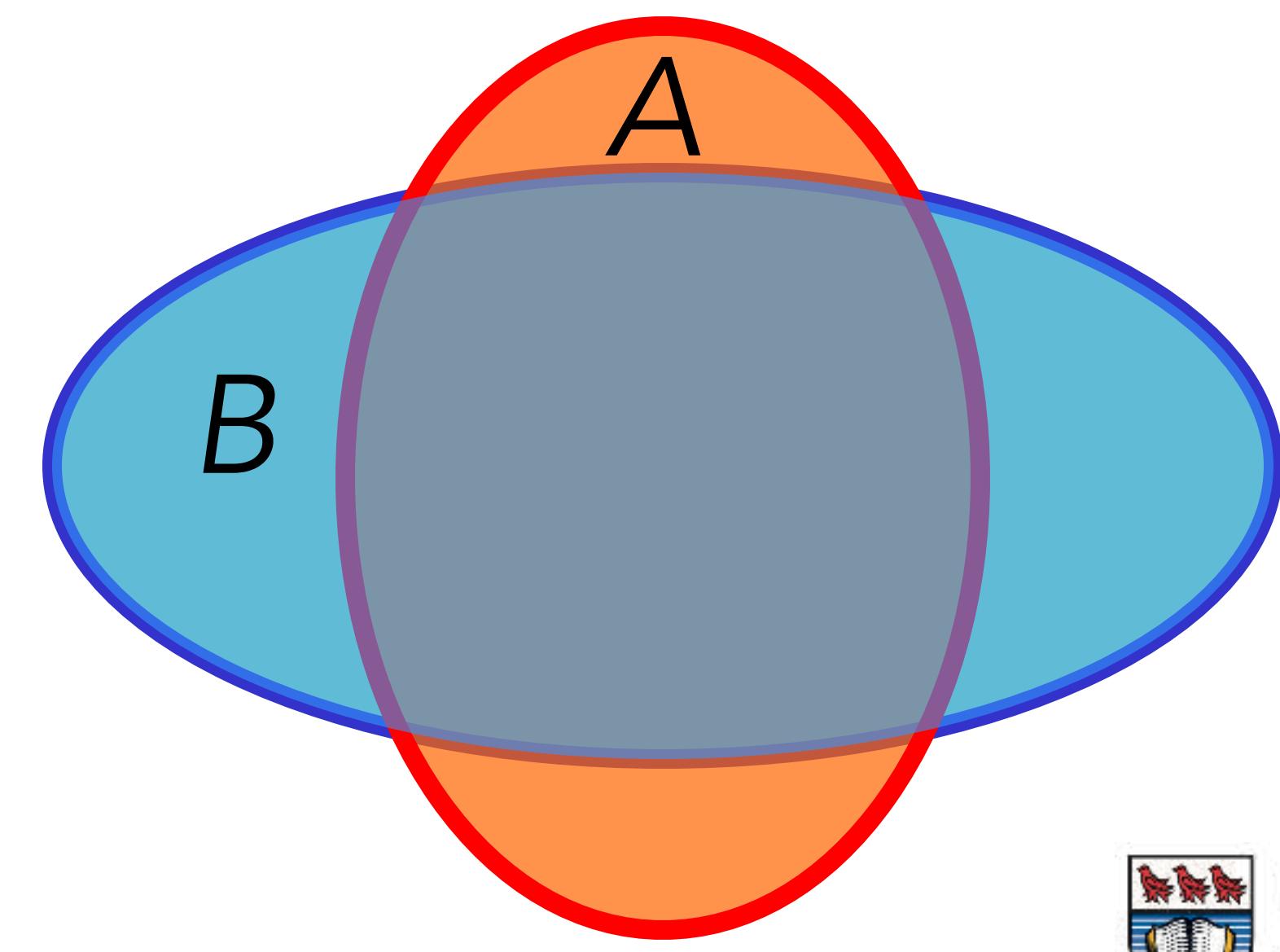
- Compact
- CSG! (see later)
- Good model for both fluids and solids
- Easy to render with ray marching

Cons:

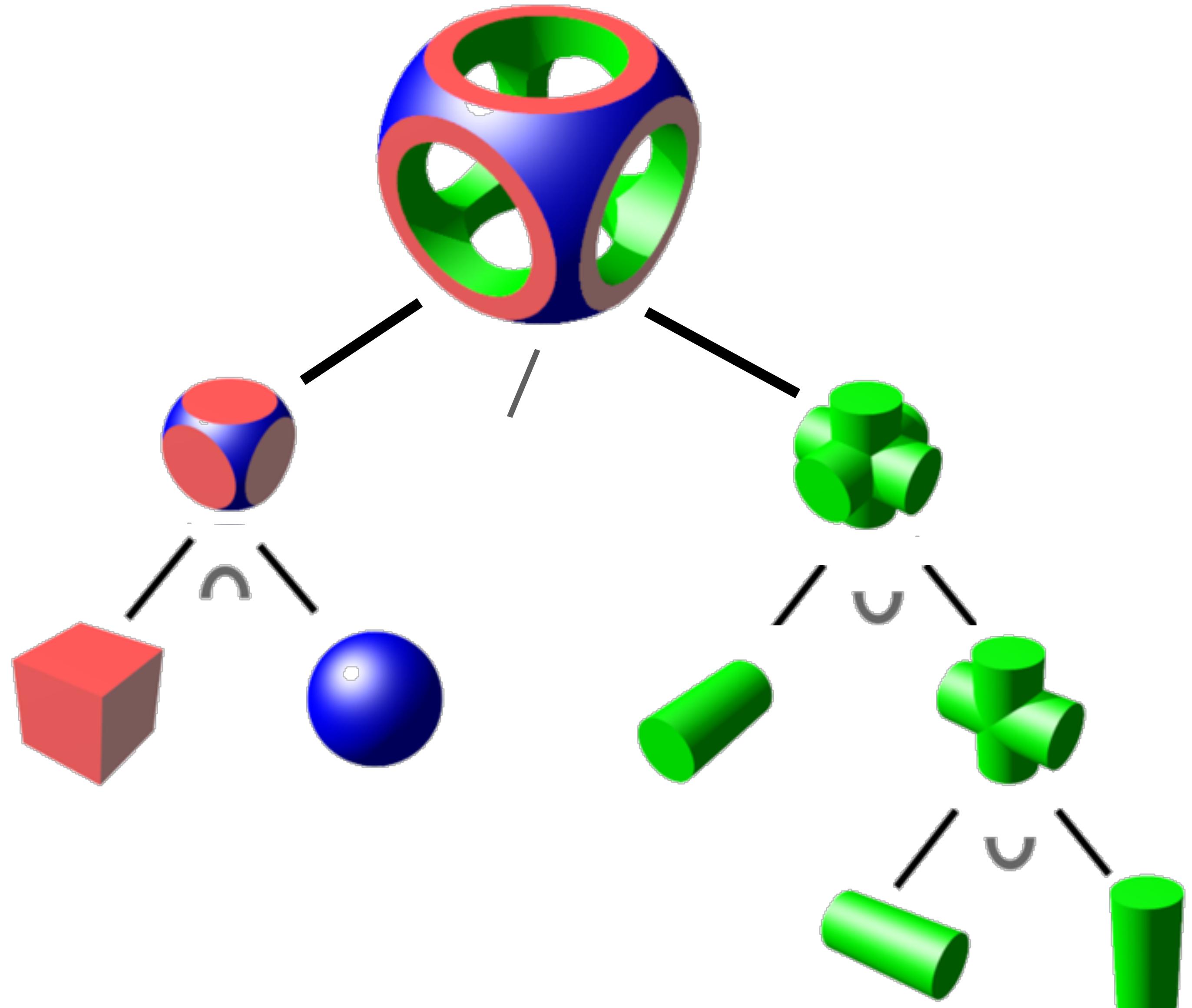
- Difficult to render for rasterization-based pipelines

Implicit Solid Modeling

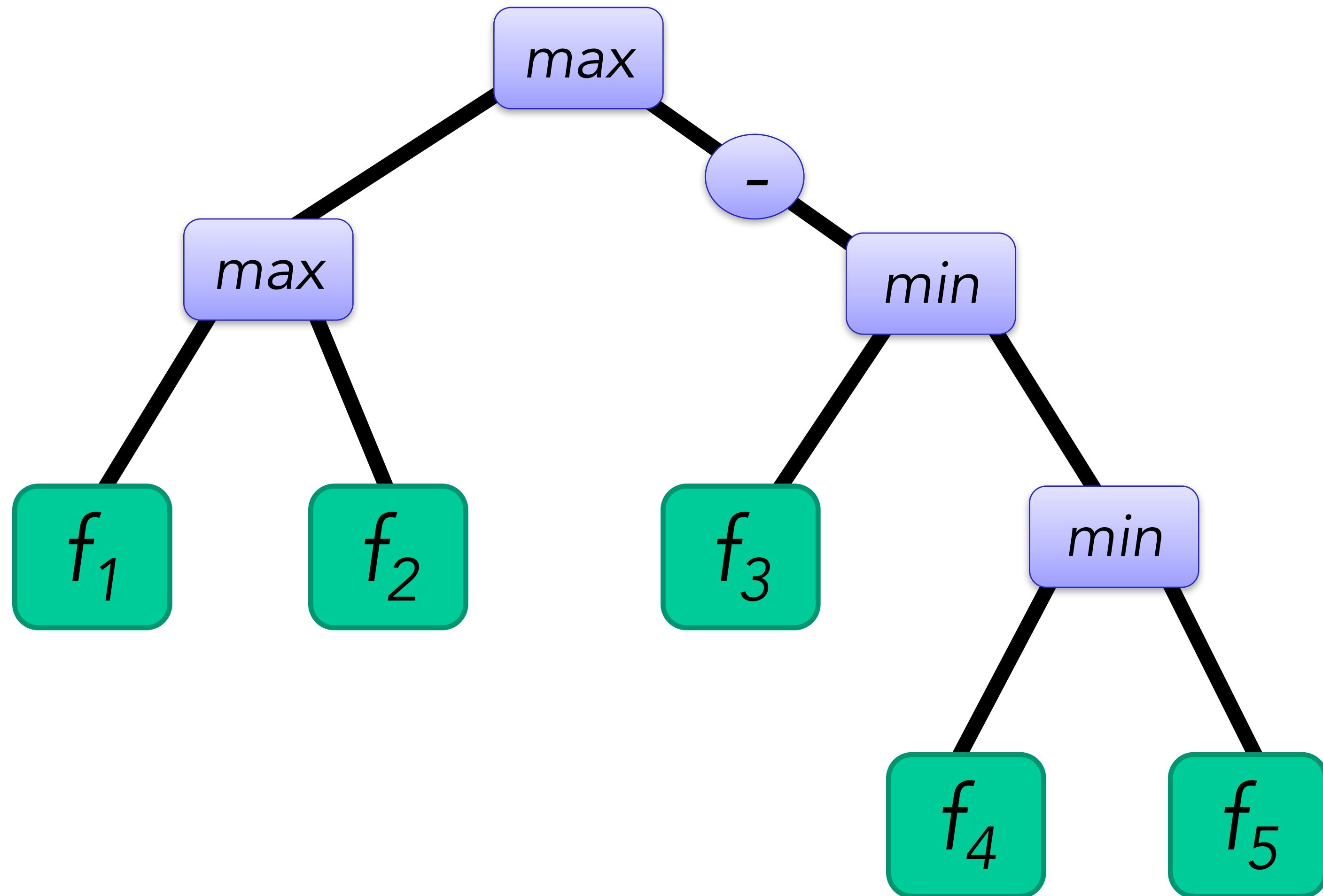
- Let A and B be two solid objects described implicitly by implicit functions f_A and f_B
- We can then define:
 - complement: $-f_A$
 - intersection: $\max(f_A, f_B)$
 - union: $\min(f_A, f_B)$
 - subtraction: $\max(f_A, -f_B)$



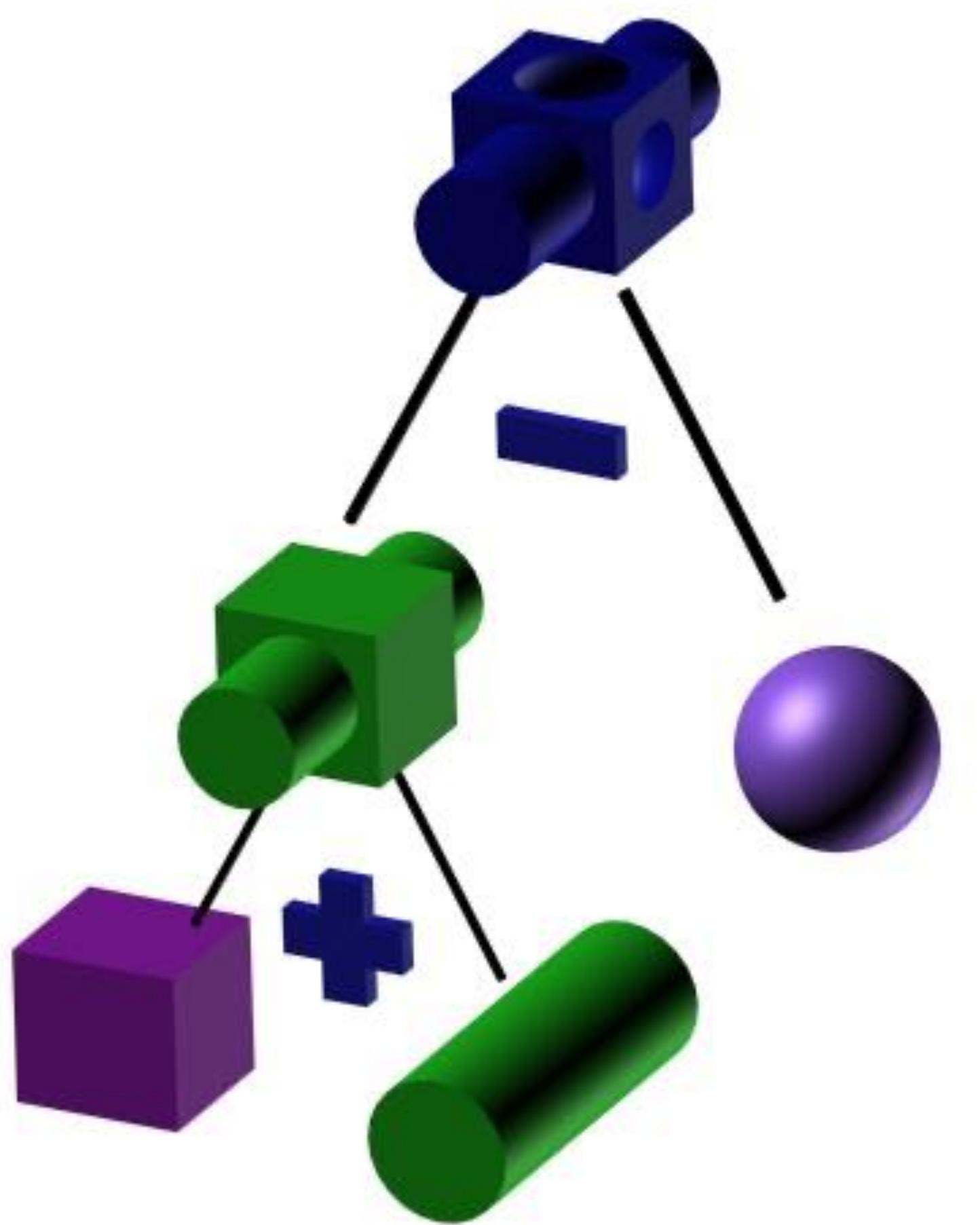
Constructive Solid Geometry (CSG)



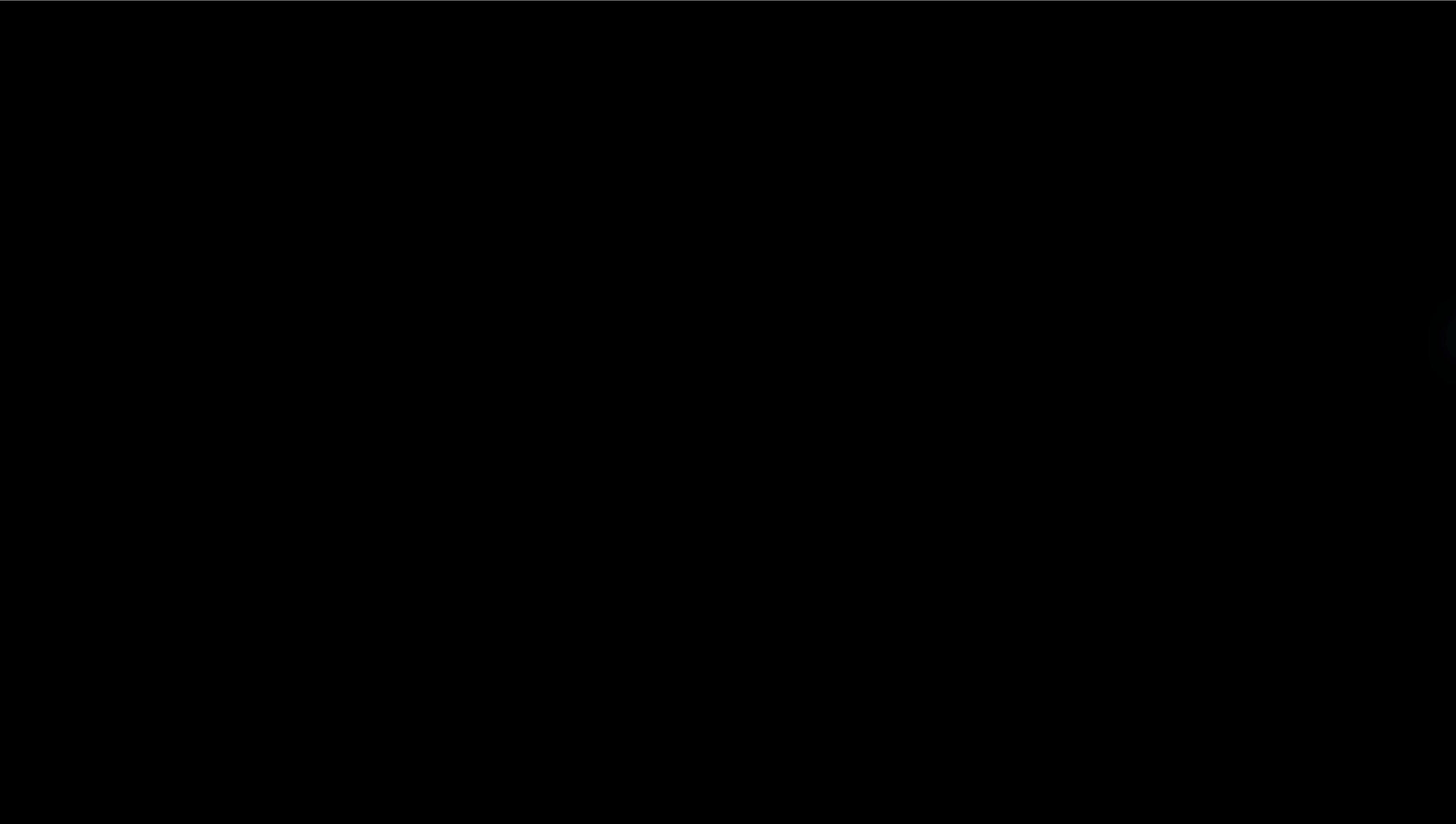
Geometric Solid Modelling



Constructive Solid Geometry CSG



Modeling Complex Scenes



<http://iquilezles.org/www/articles/raymarchingdf/raymarchingdf.htm>



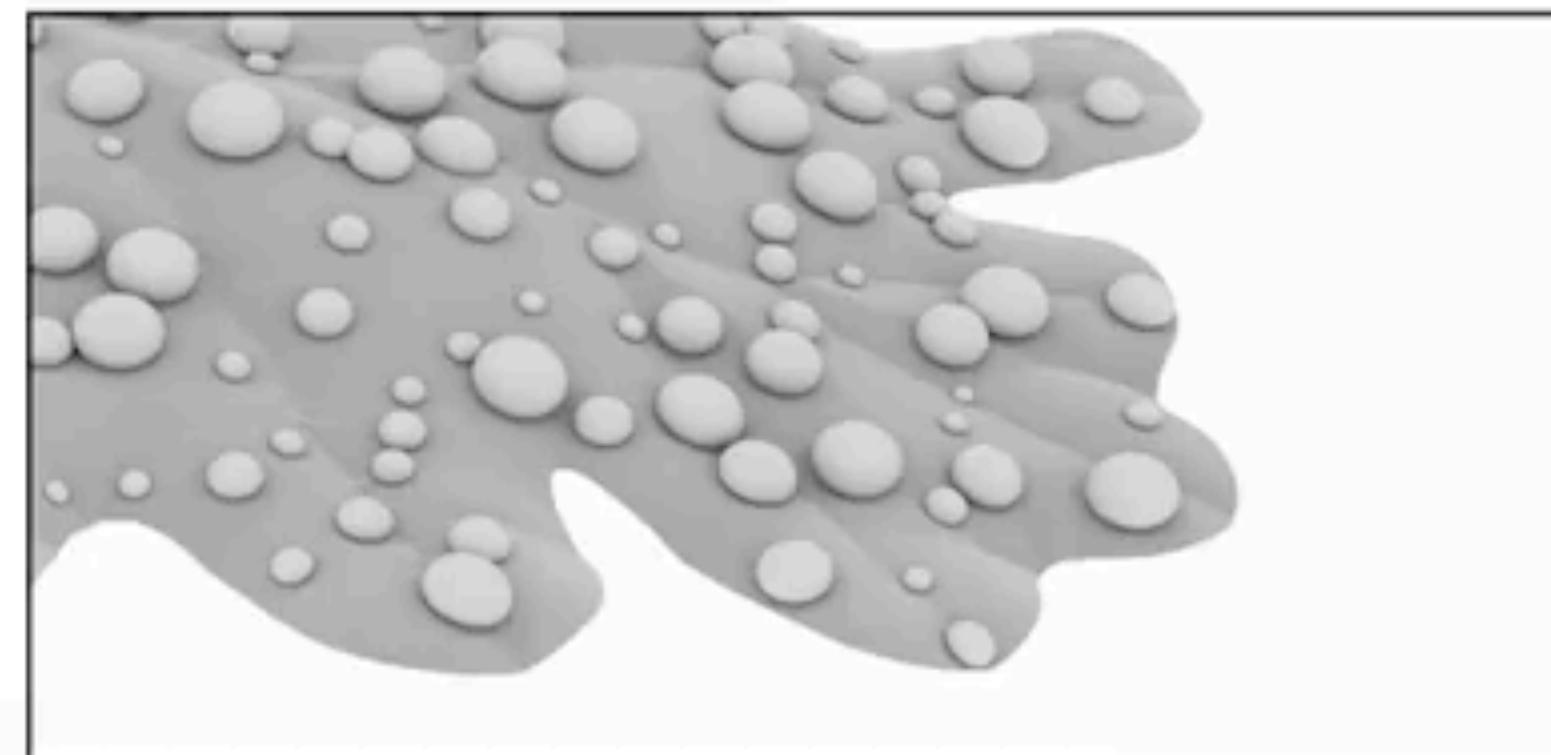
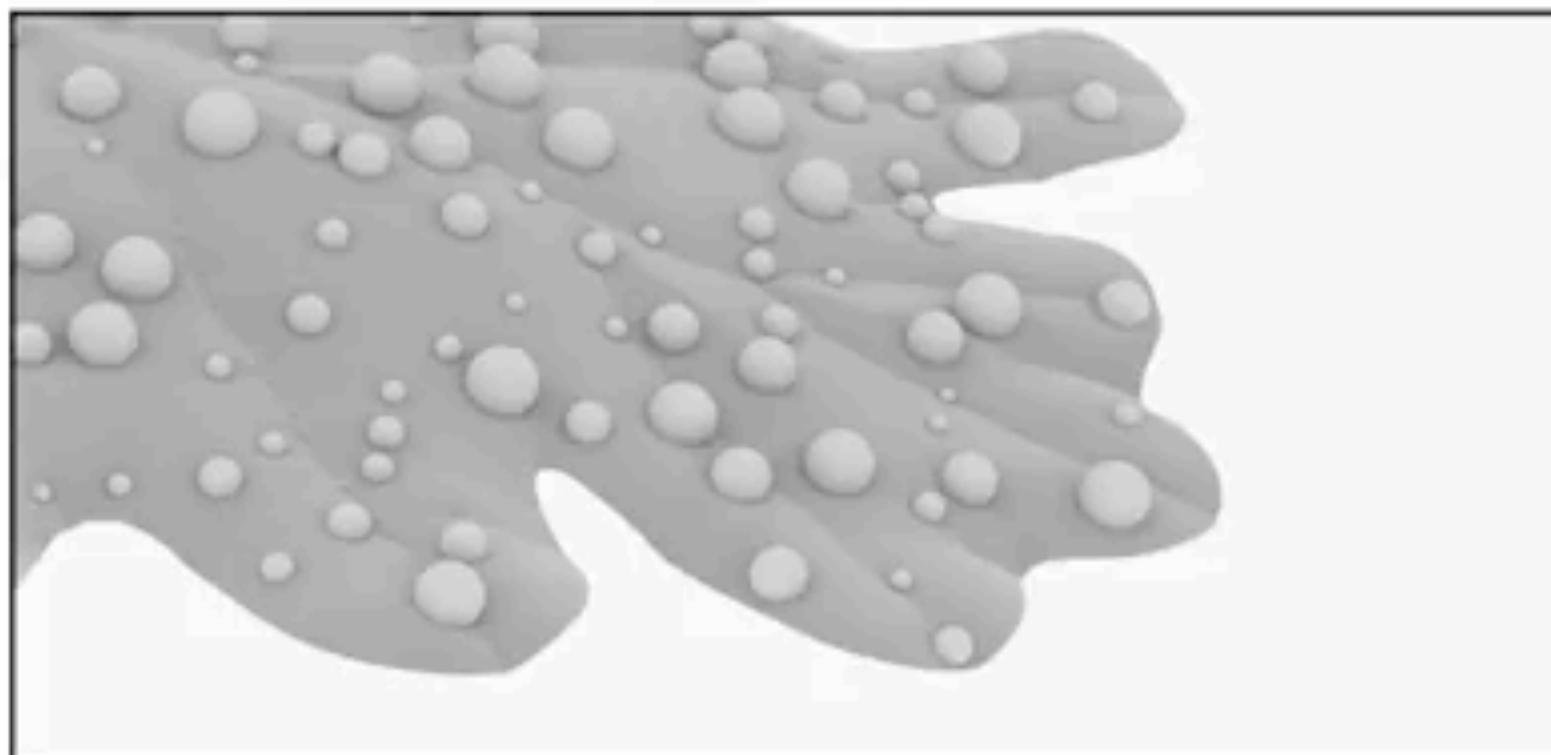
University
of Victoria

Computer Science



Sketch-Based Implicit Blending

Baptiste Angles^{1,2}, Marco Tarini³, Brian Wyvill¹, Loïc Barthe², Andrea Tagliasacchi¹



1. University of Victoria



2. Université de Toulouse, IRIT/CNRS



3. Università dell'Insubria, ISTI / CNR



Rendering Implicits

Ray Marching

Acknowledgement: Jamie Wong

<http://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/>

CSC 305 - Introduction to Computer Graphics - Teseo Schneider

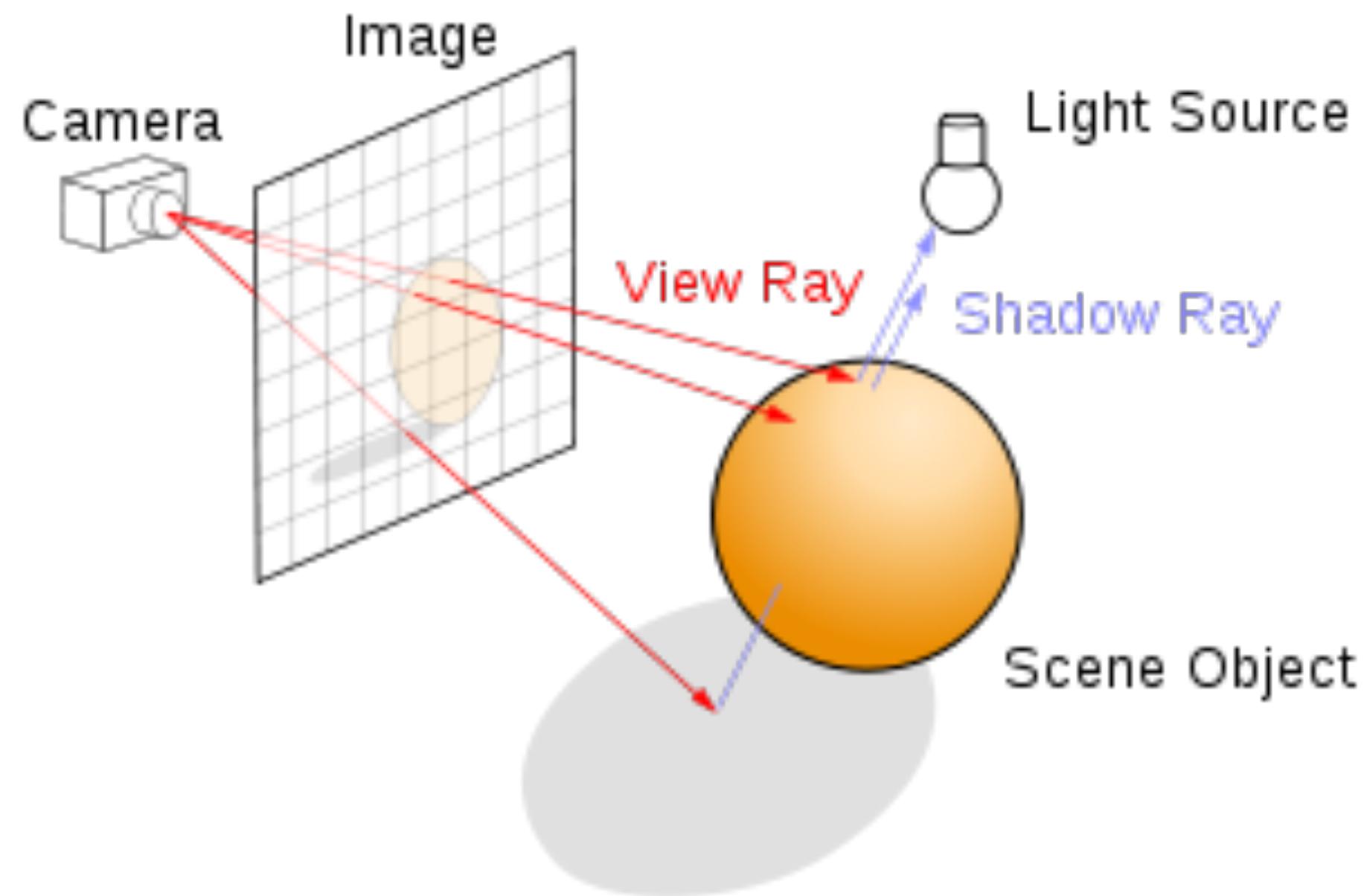


University
of Victoria

Computer Science

https://www.youtube.com/watch?list=PLVbS70ERPhCCGKc-MdKsH03R7o6TNbGoZ&time_continue=12&v=roZ-Cgxe9bU

Ray Marching



- Similar to ray tracing
- Since the implicit function might not be quadratic, we cannot find the intersection explicitly as we did for triangles and spheres

Image from Wikipedia

CSC 305 - Introduction to Computer Graphics - Teseo Schneider

Algorithm 1

- The simplest algorithm resembles gradient descent
 - You proceed on the ray by a fixed amount
 - You start to do bisection search if you get closer than a given distance to the surface

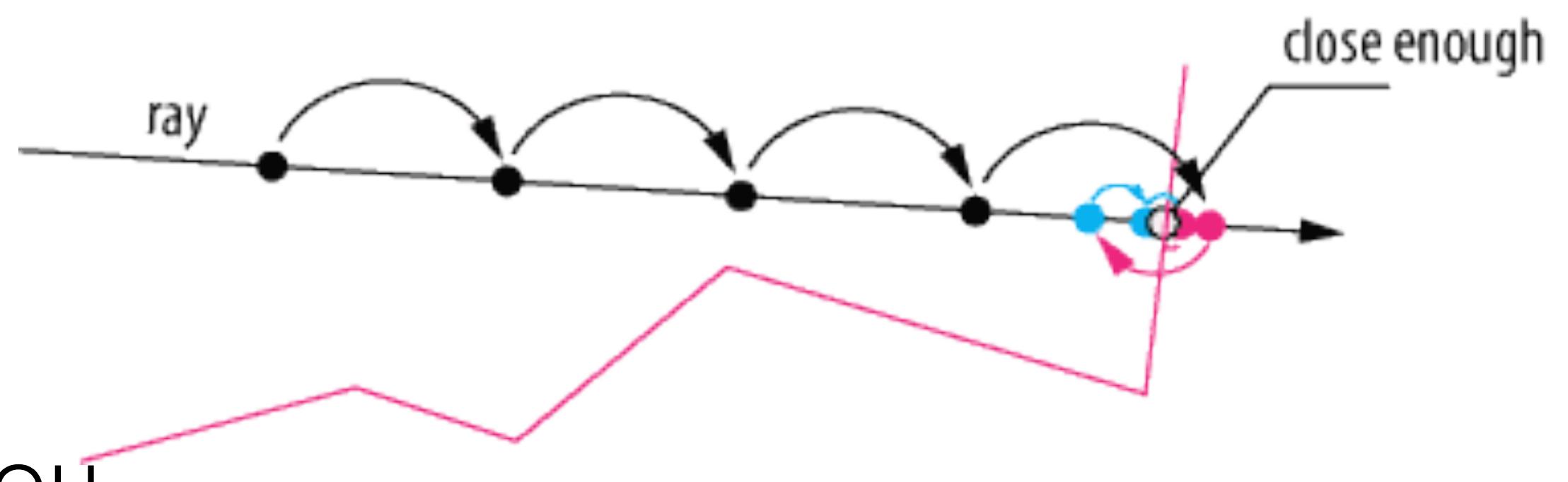
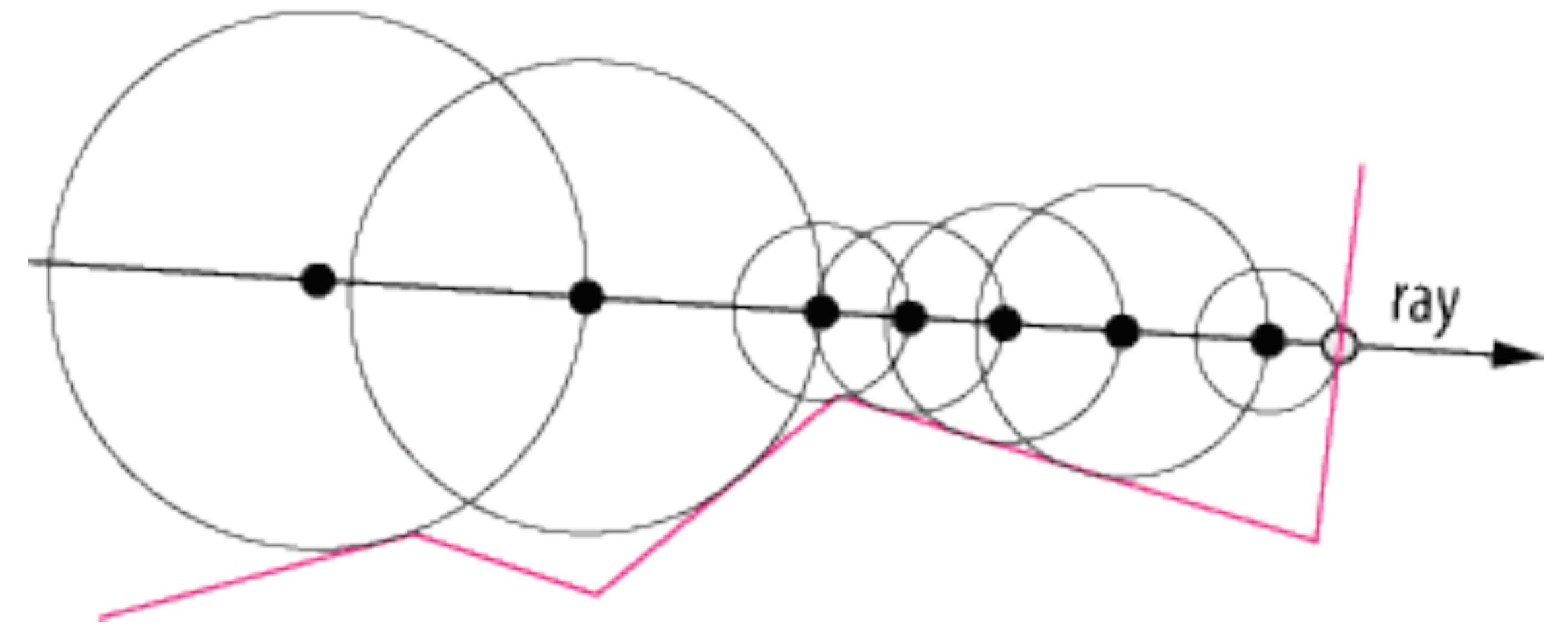


Image From: <https://computergraphics.stackexchange.com/questions/161/what-is-ray-marching-is-sphere-tracing-the-same-thing>

Algorithm 2: “Sphere Tracing”

- Instead of taking a constant step, you check the closest point on the surface, and you move by that amount
- Can be done exactly with a distance field
- It can be a conservative estimate for more general cases



Tutorials and Examples

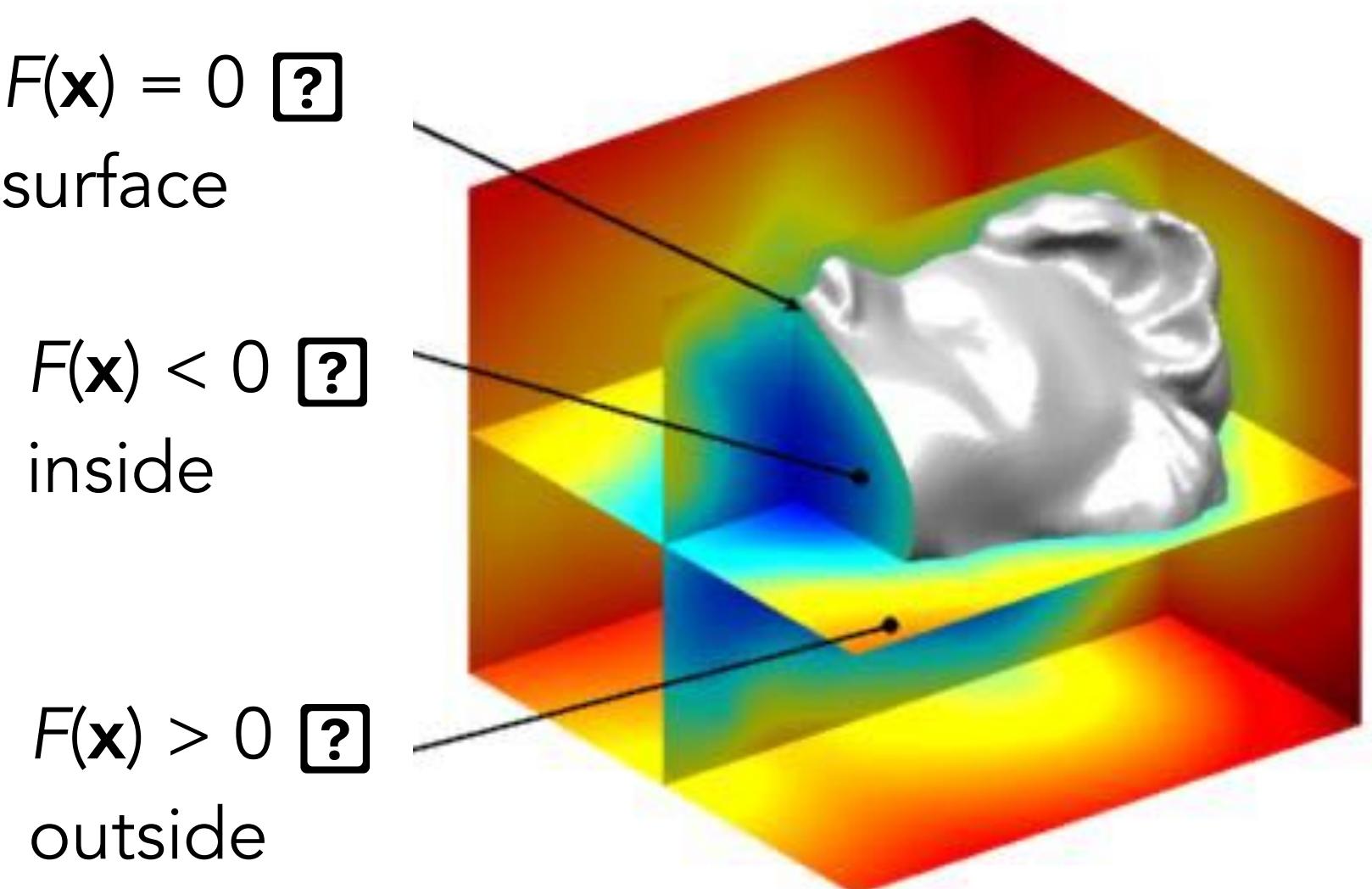
- shadertoy.com
- <http://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/#the-raymarching-algorithm>
- <https://computergraphics.stackexchange.com/questions/161/what-is-ray-marching-is-sphere-tracing-the-same-thing>

Rendering Implicits

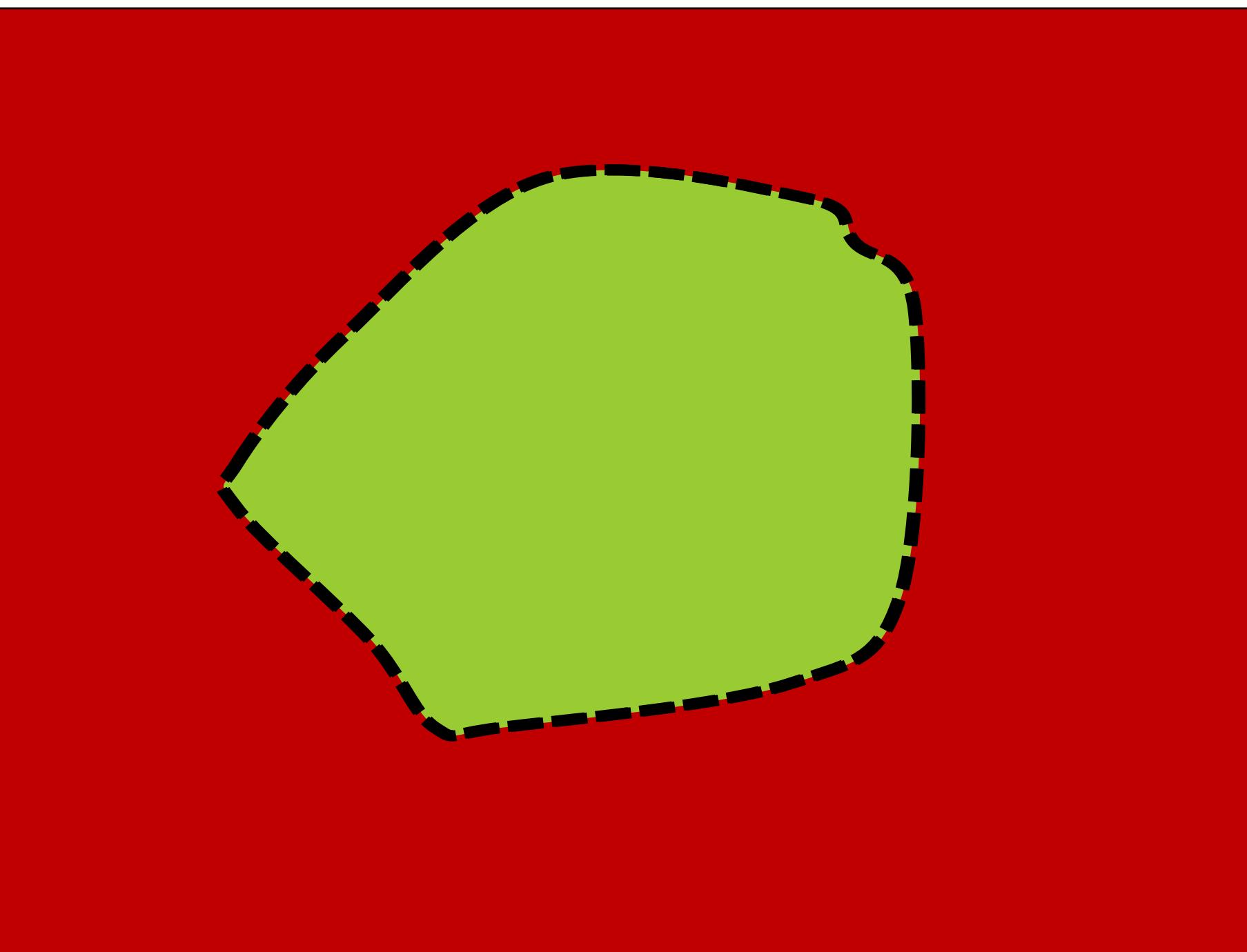
Explicit Meshing

Extracting the Surface

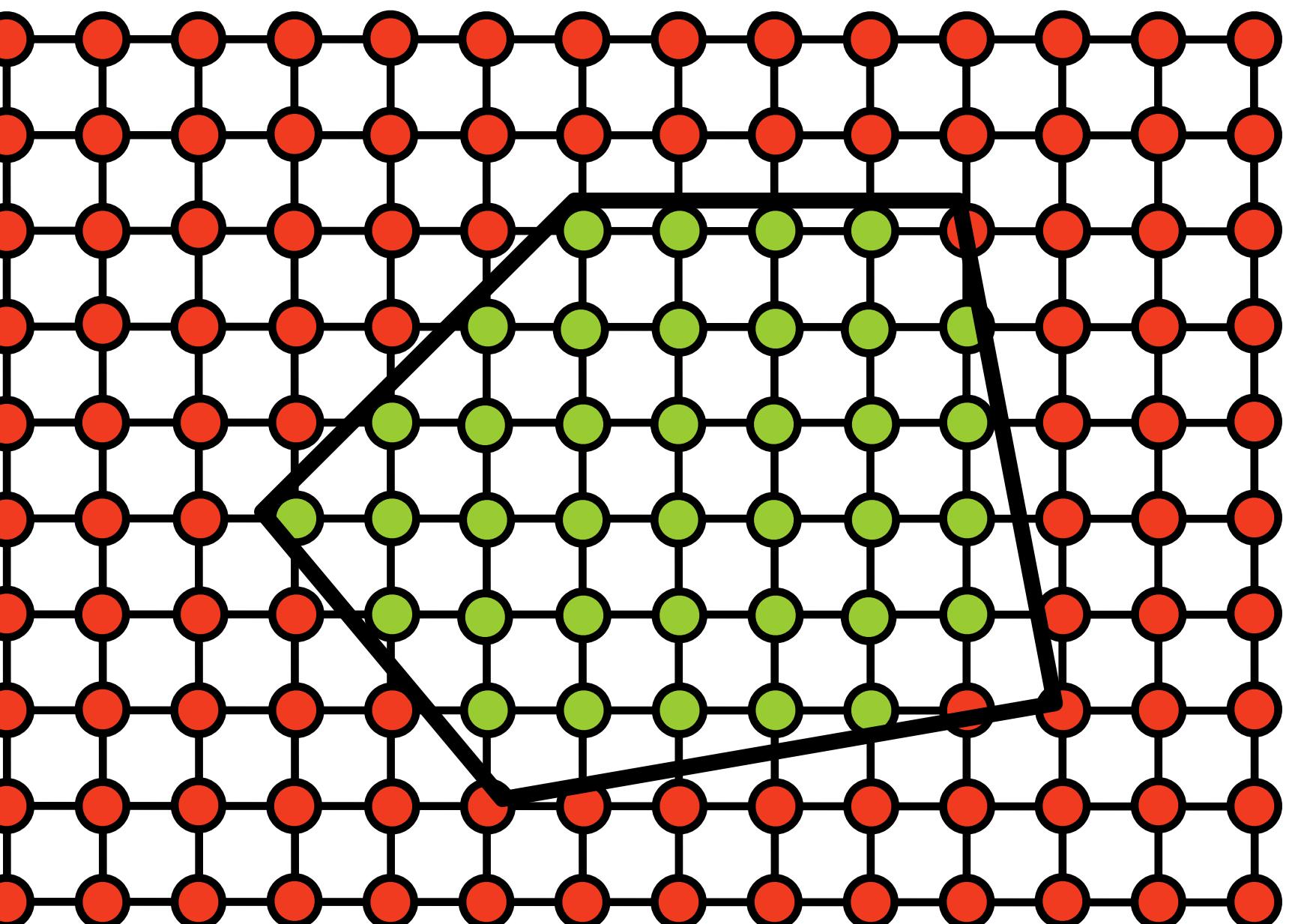
- Wish to compute a manifold mesh of the level set



Sample the SDF

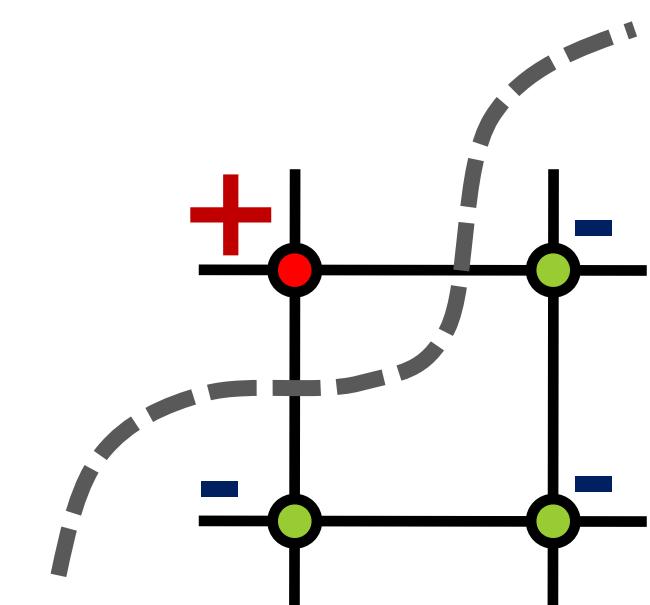


Sample the SDF

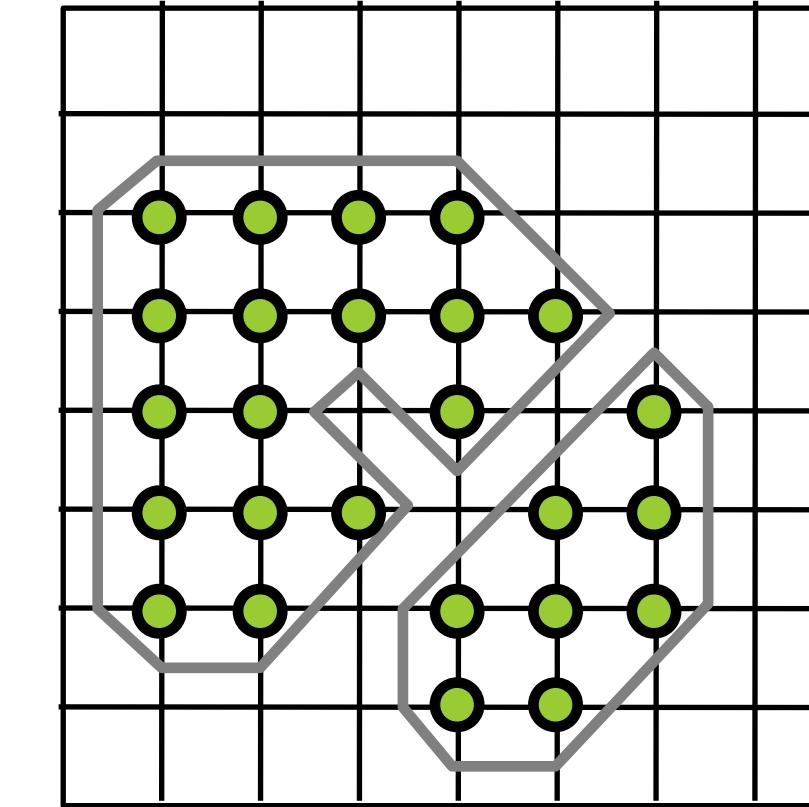
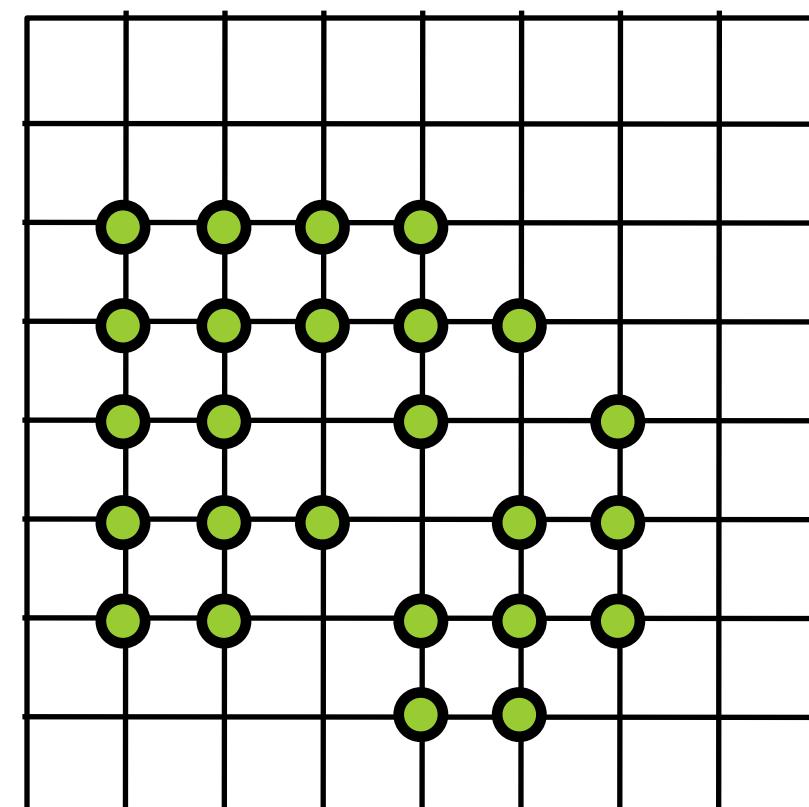


Tessellation

- Want to approximate an implicit surface with a mesh
- Can't explicitly compute all the roots
 - Sampling the level set is difficult (root finding)
- Solution: find approximate roots by trapping the implicit surface in a grid (lattice)

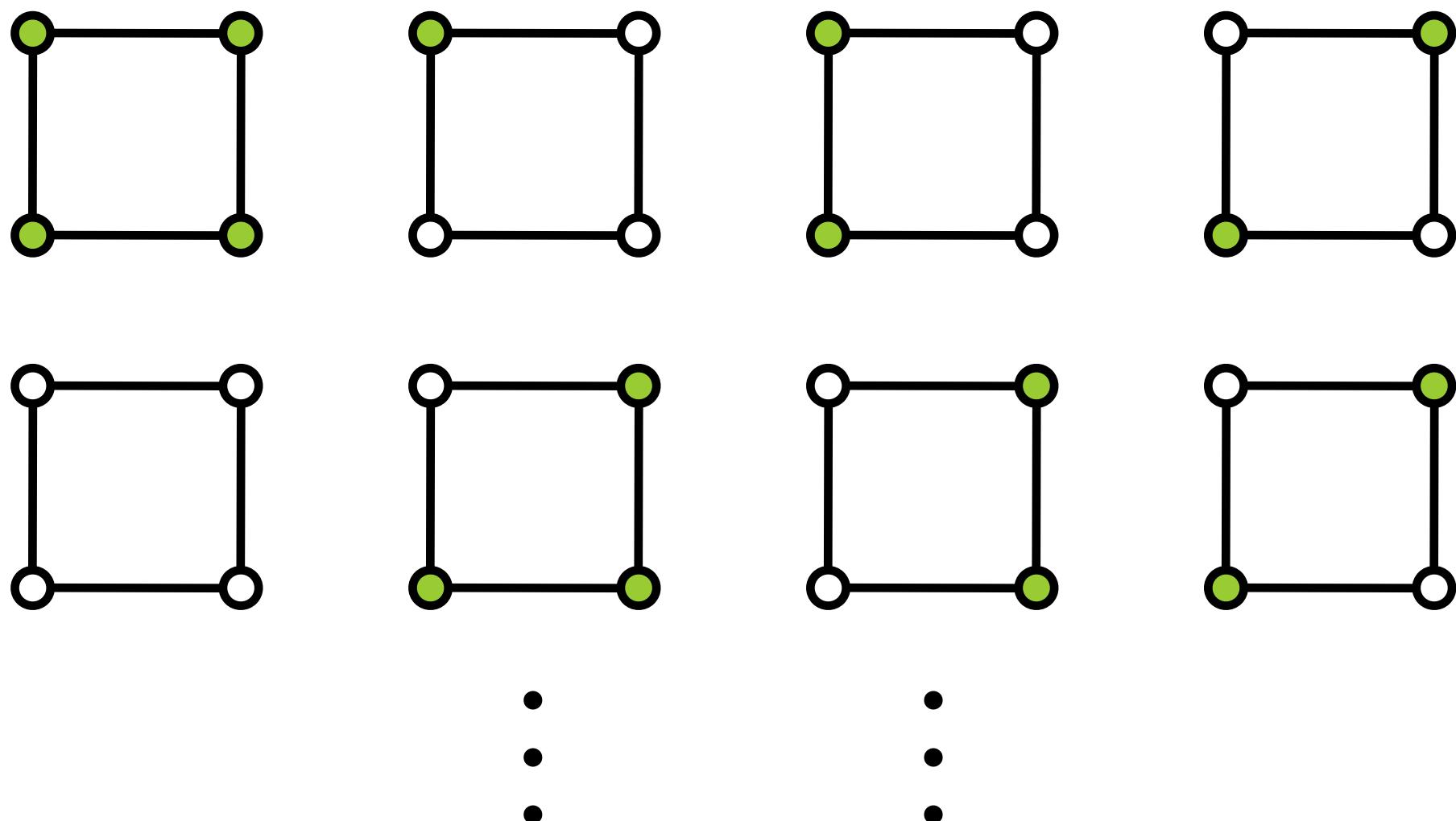


$$\bullet F(\mathbf{x}) < 0$$



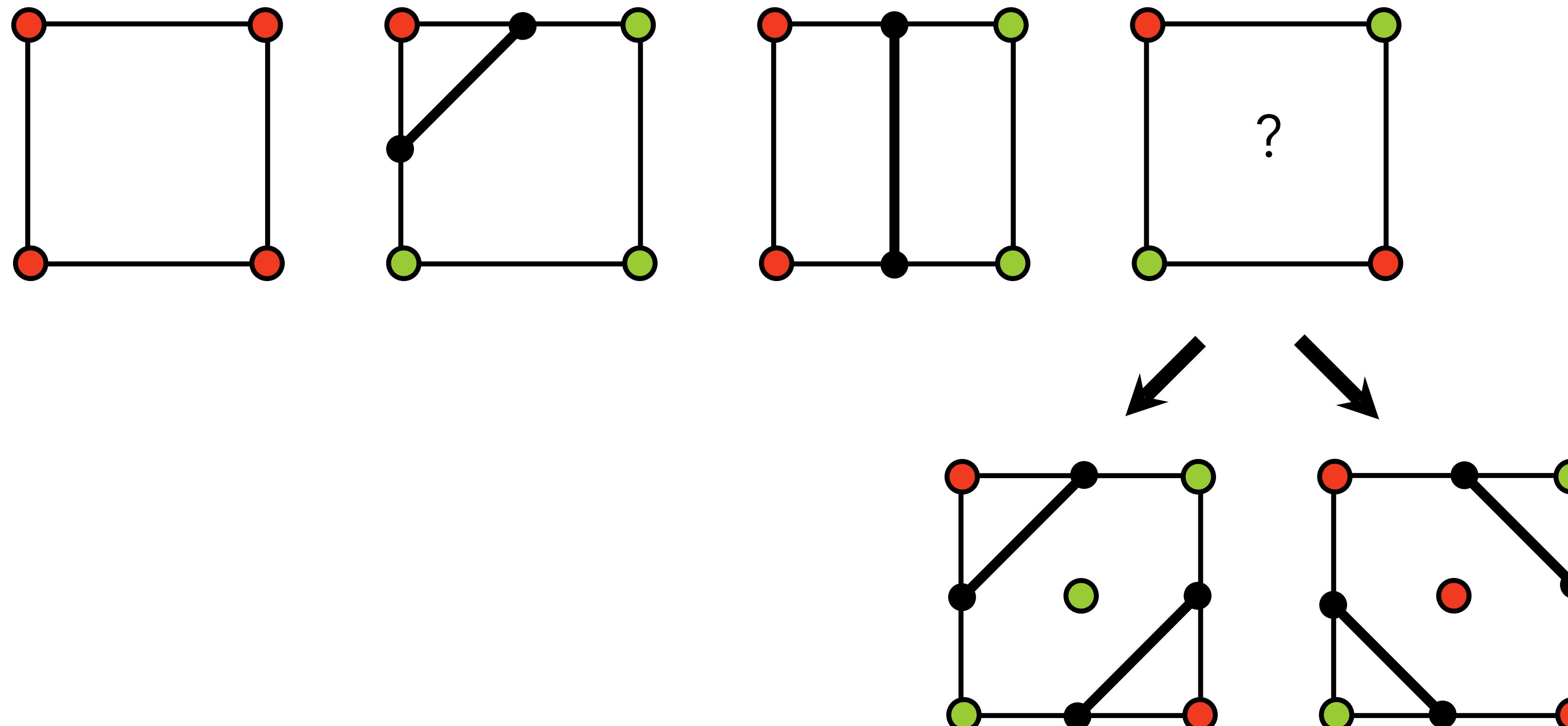
Marching Squares

- 16 different configurations in 2D
- 4 equivalence classes (up to rotational and reflection symmetry + complement)



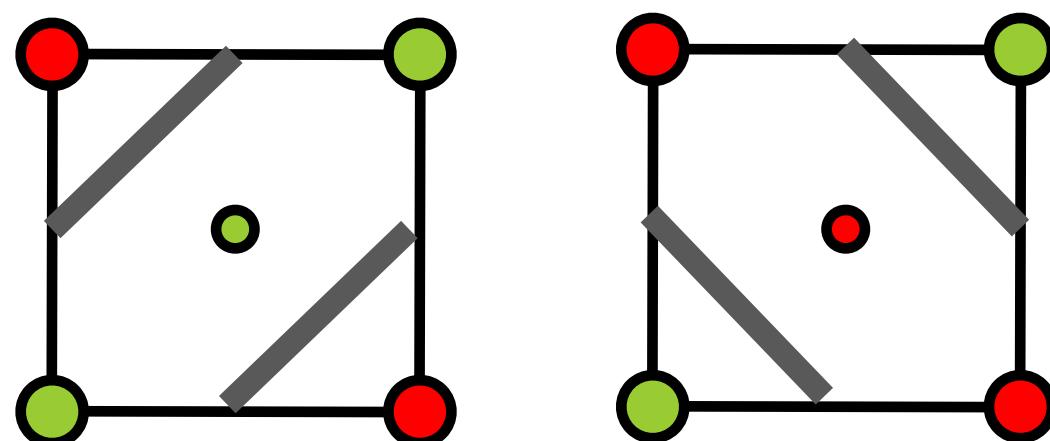
Tessellation in 2D

- 4 equivalence classes (up to rotational and reflection symmetry + complement)

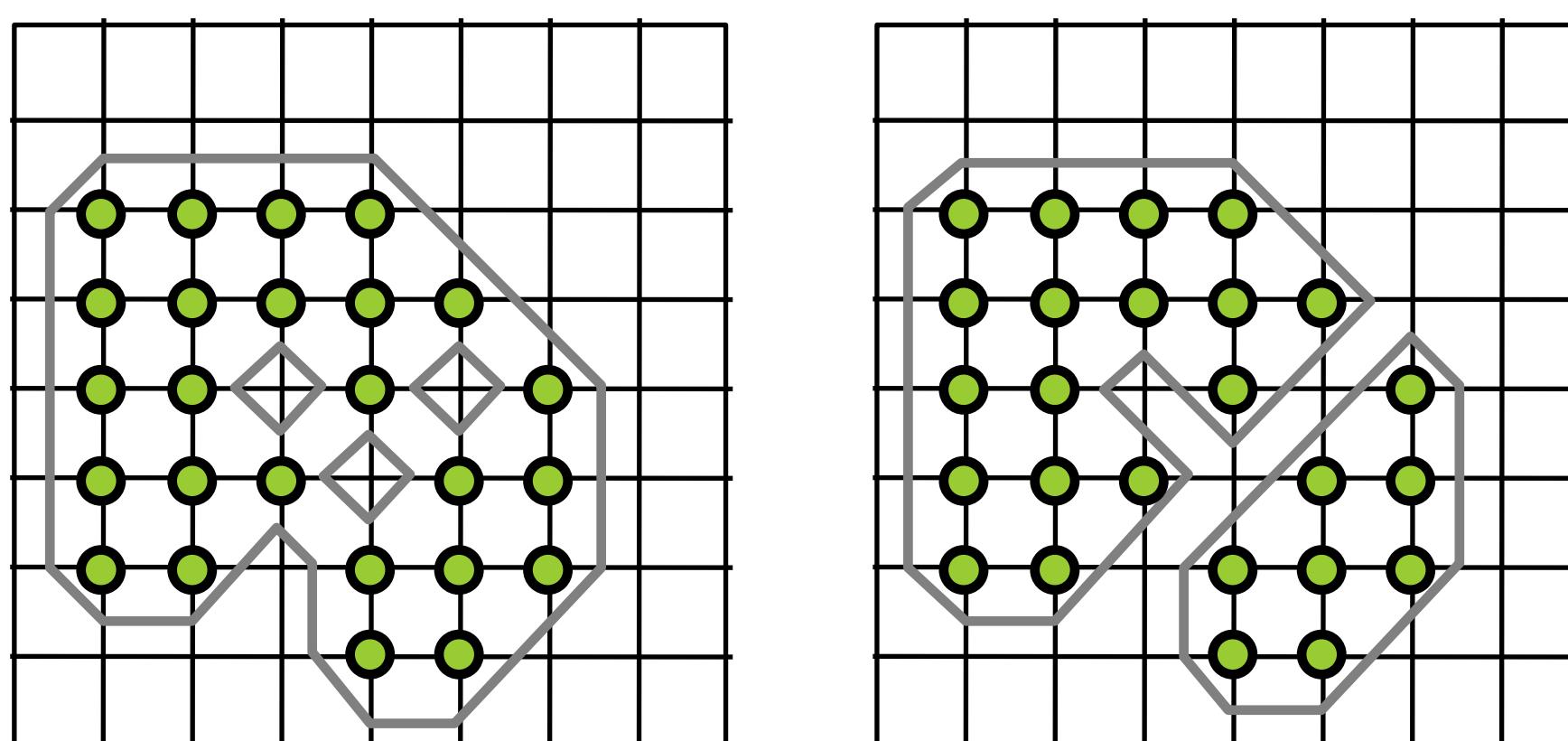


Tessellation in 2D

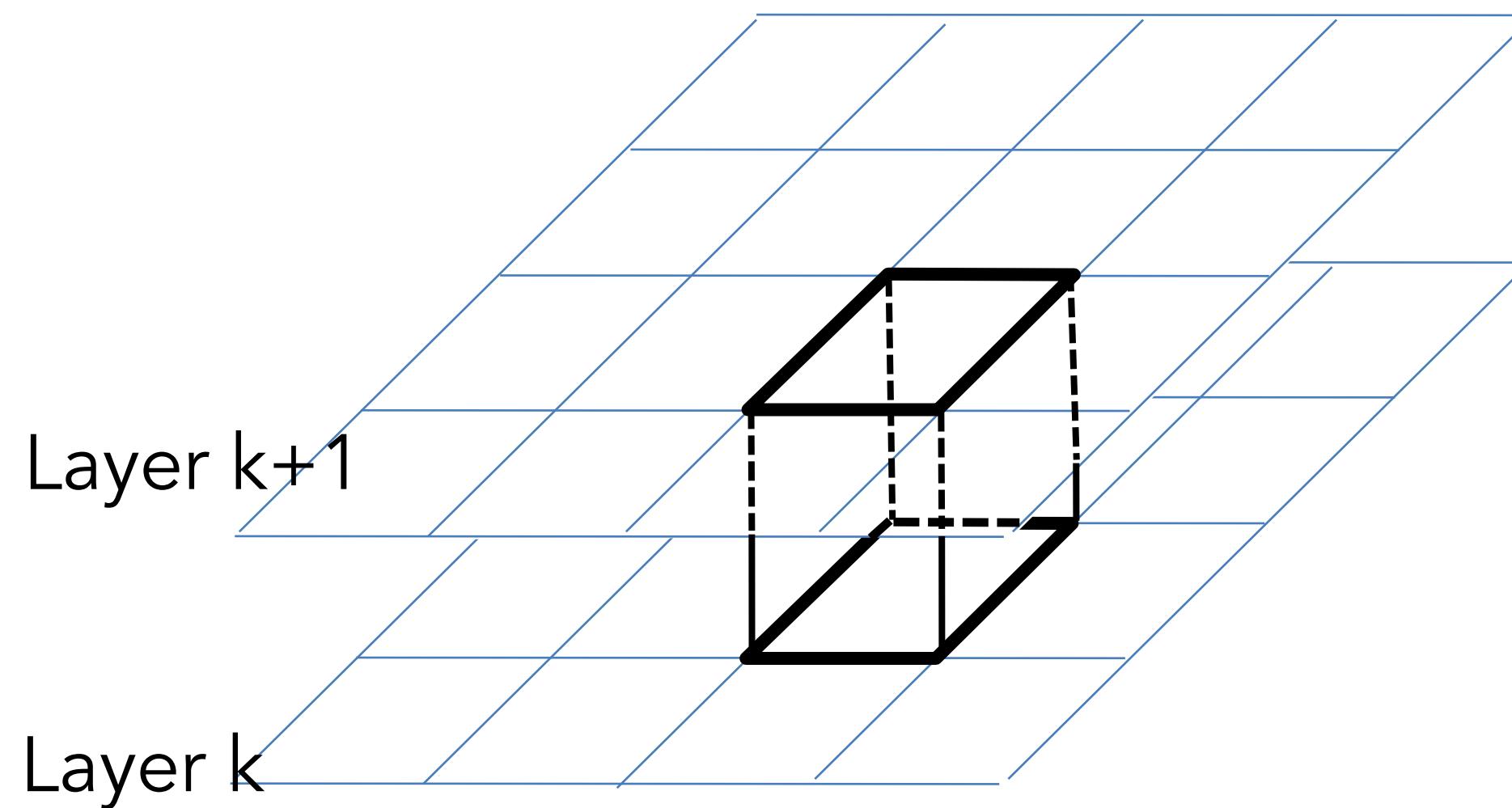
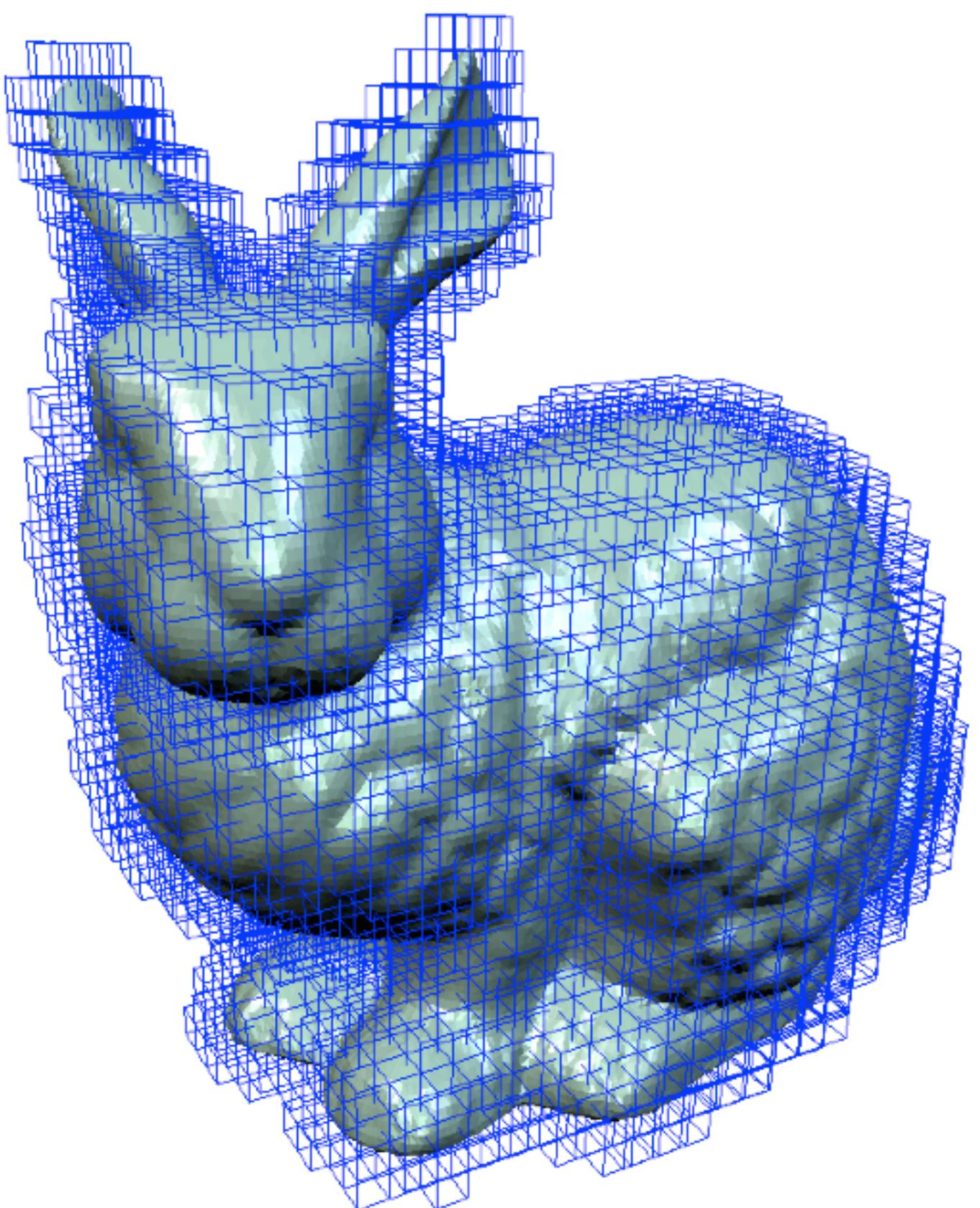
- Case 4 is ambiguous:



- Always pick consistently to avoid problems with the resulting mesh

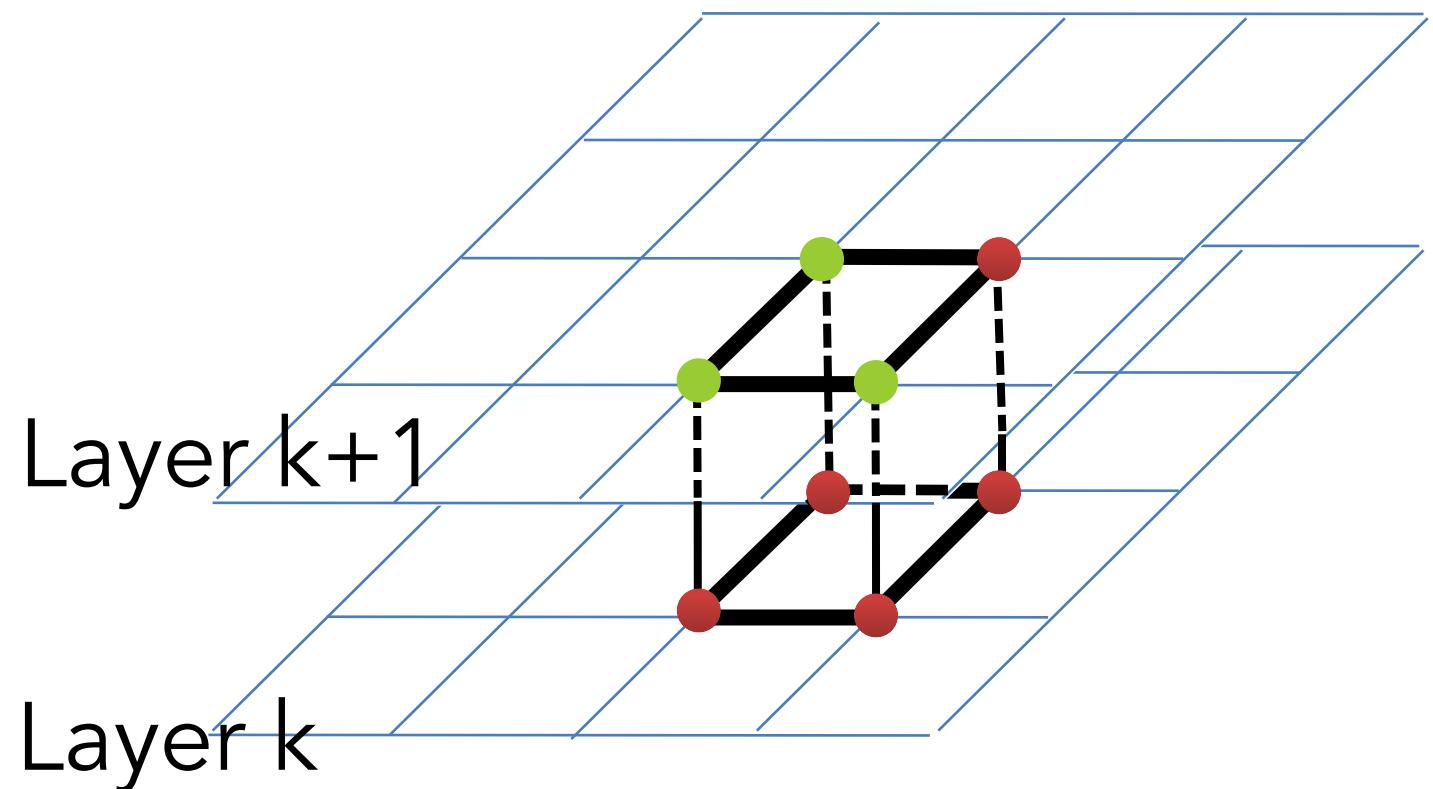


3D: Marching Cubes



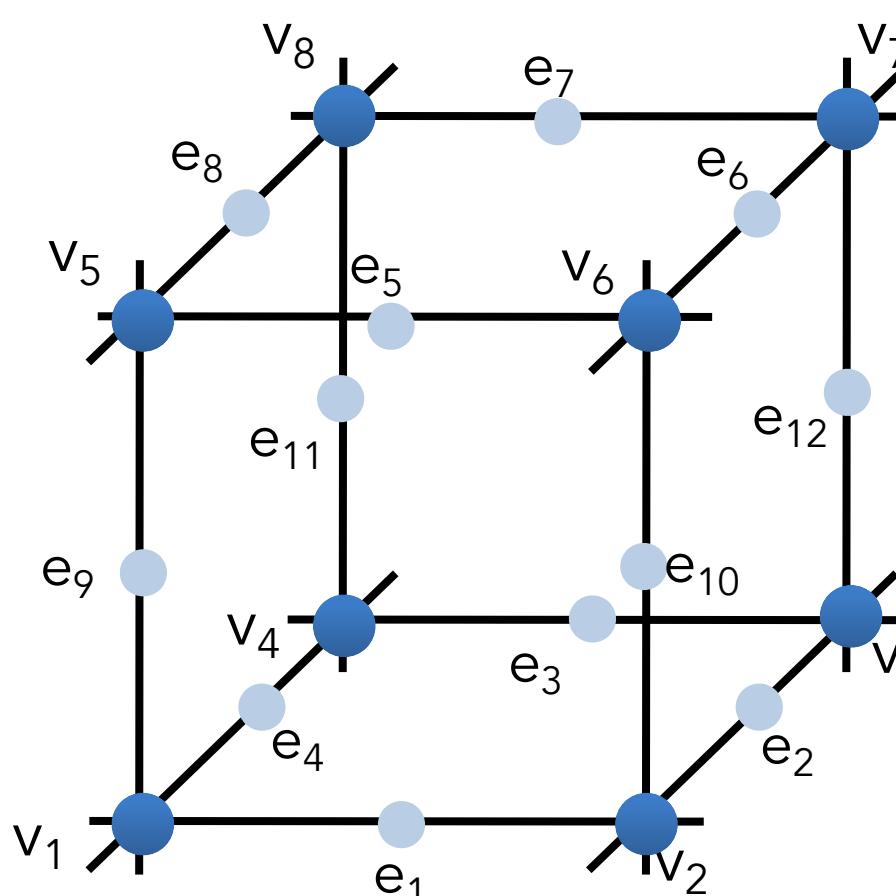
Marching Cubes

- Marching Cubes (Lorensen and Cline 1987)
 1. Load 4 layers of the grid into memory
 2. Create a cube whose vertices lie on the two middle layers
 3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)

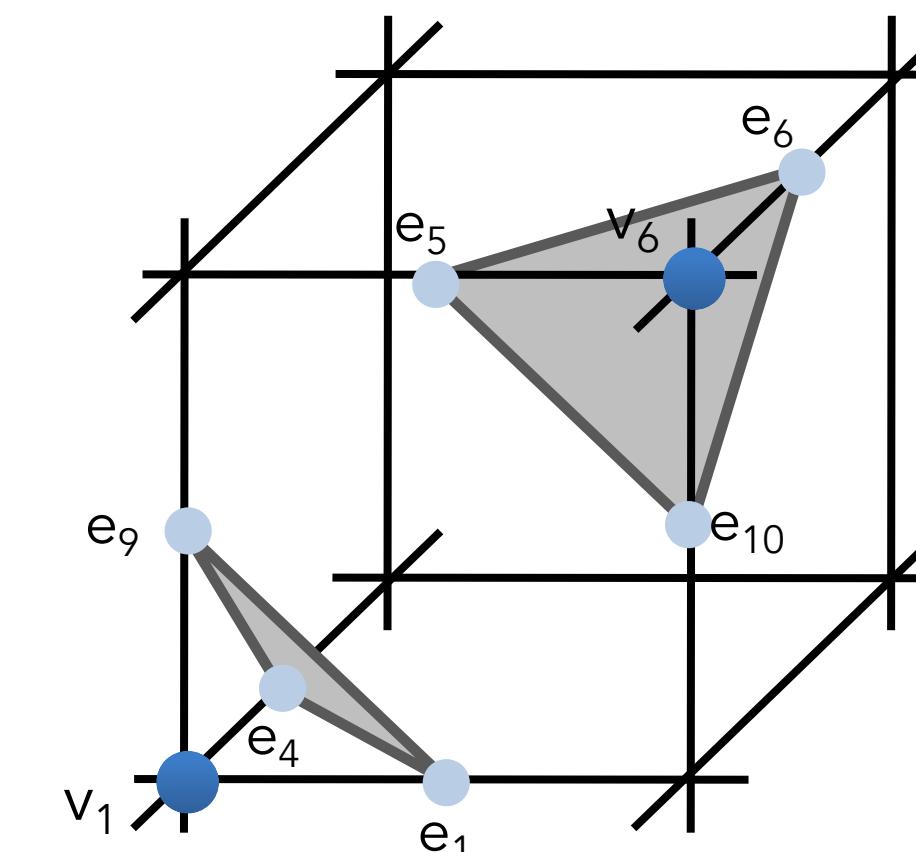


Marching Cubes

4. Compute case index. We have $2^8 = 256$ cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



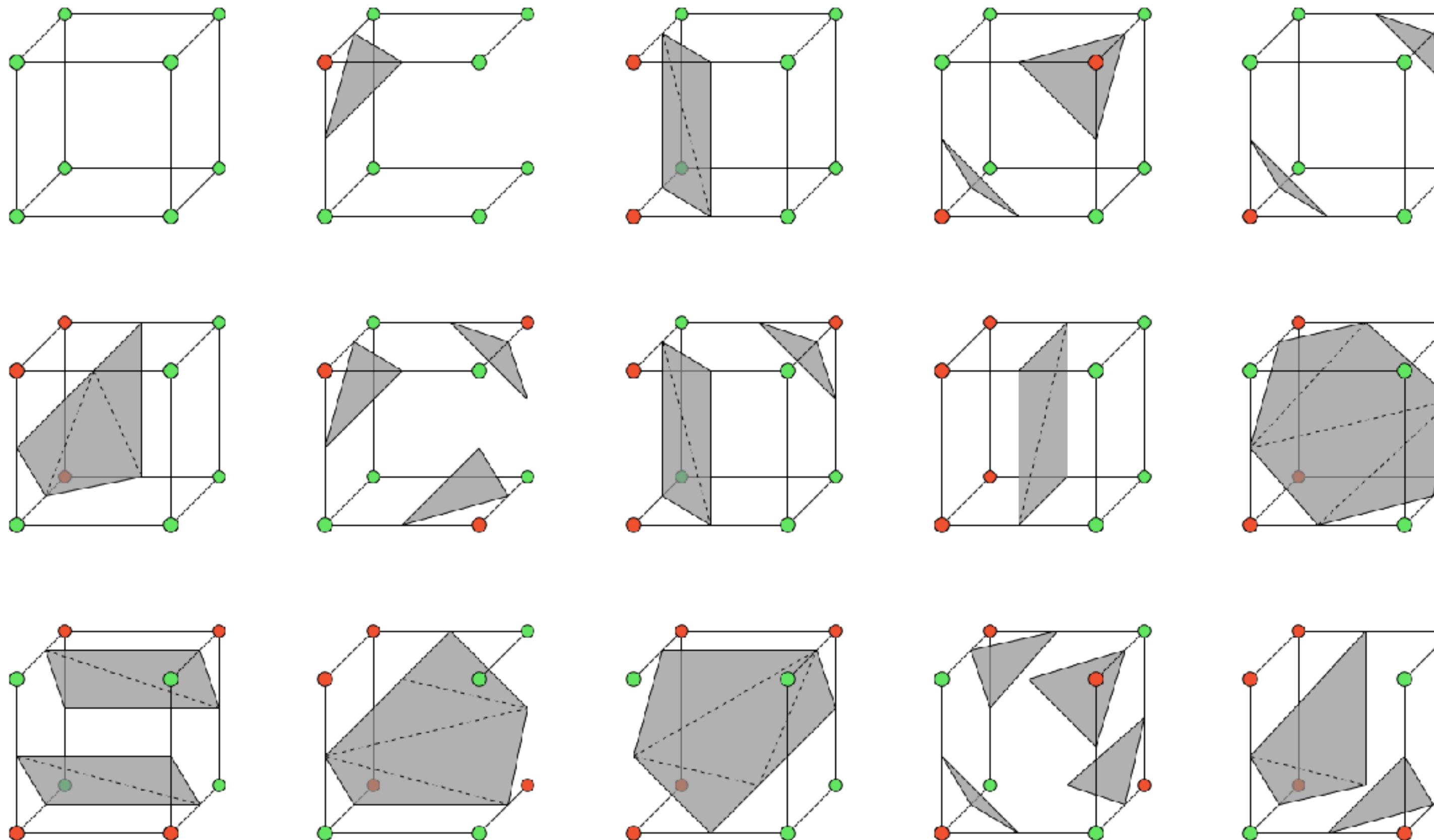
index = [v₁ | v₂ | v₃ | v₄ | v₅ | v₆ | v₇ | v₈]



index = [0 | 0 | 1 | 0 | 0 | 0 | 0 | 1] = 33

Marching Cubes

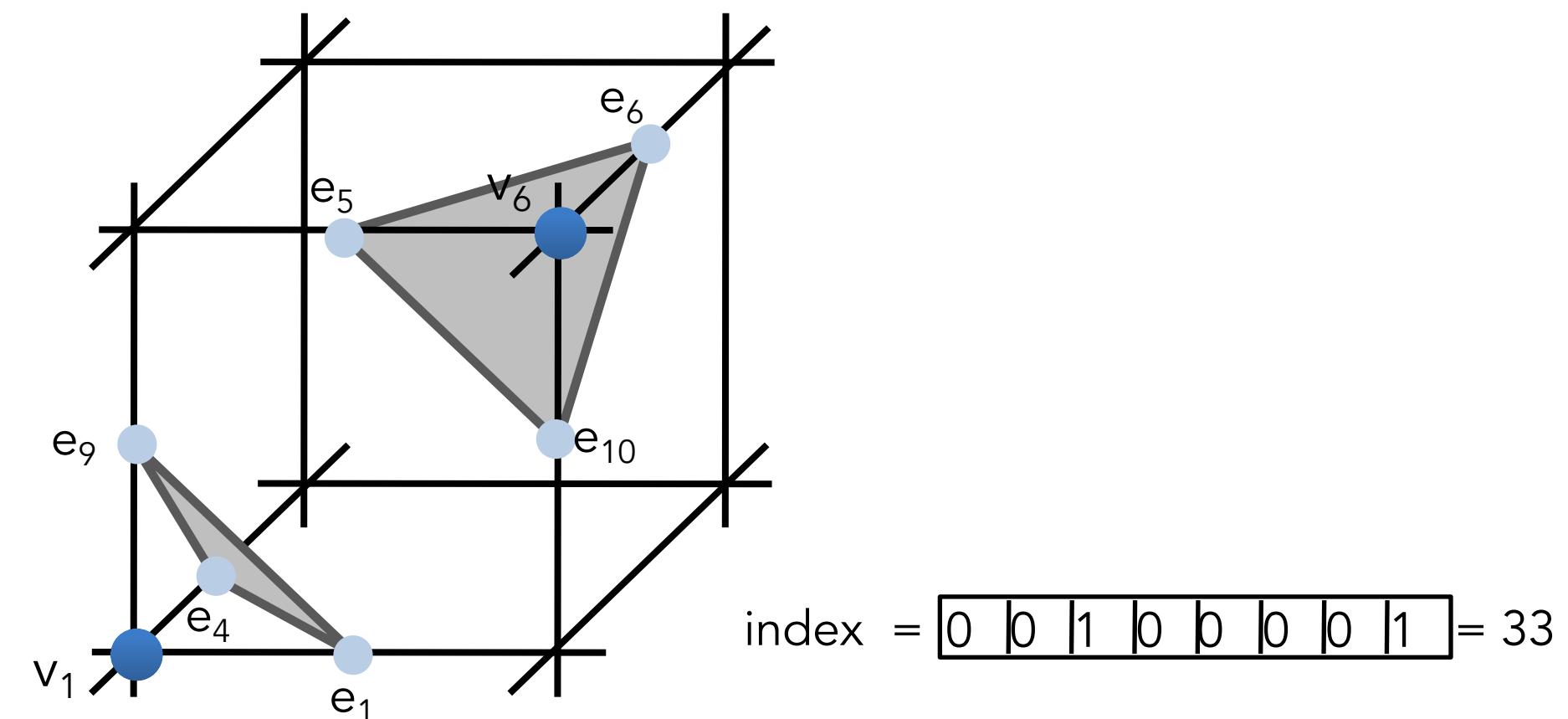
- Unique cases (by rotation, reflection and complement)



Tessellation

3D – Marching Cubes

5. Using the case index, retrieve the connectivity in the look-up table
 - Example: the entry for index 33 in the look-up table indicates that the cut edges are $e_1; e_4; e_5; e_6; e_9$ and e_{10} ; the output triangles are $(e_1; e_9; e_4)$ and $(e_5; e_{10}; e_6)$.

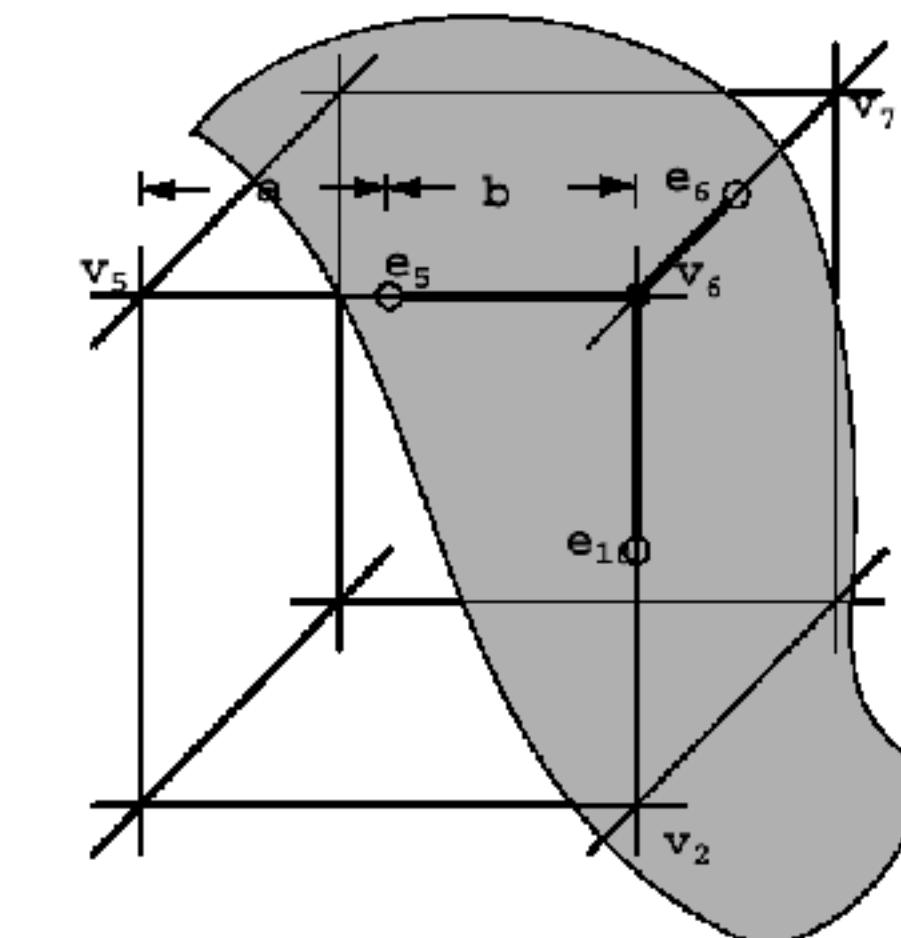


Marching Cubes

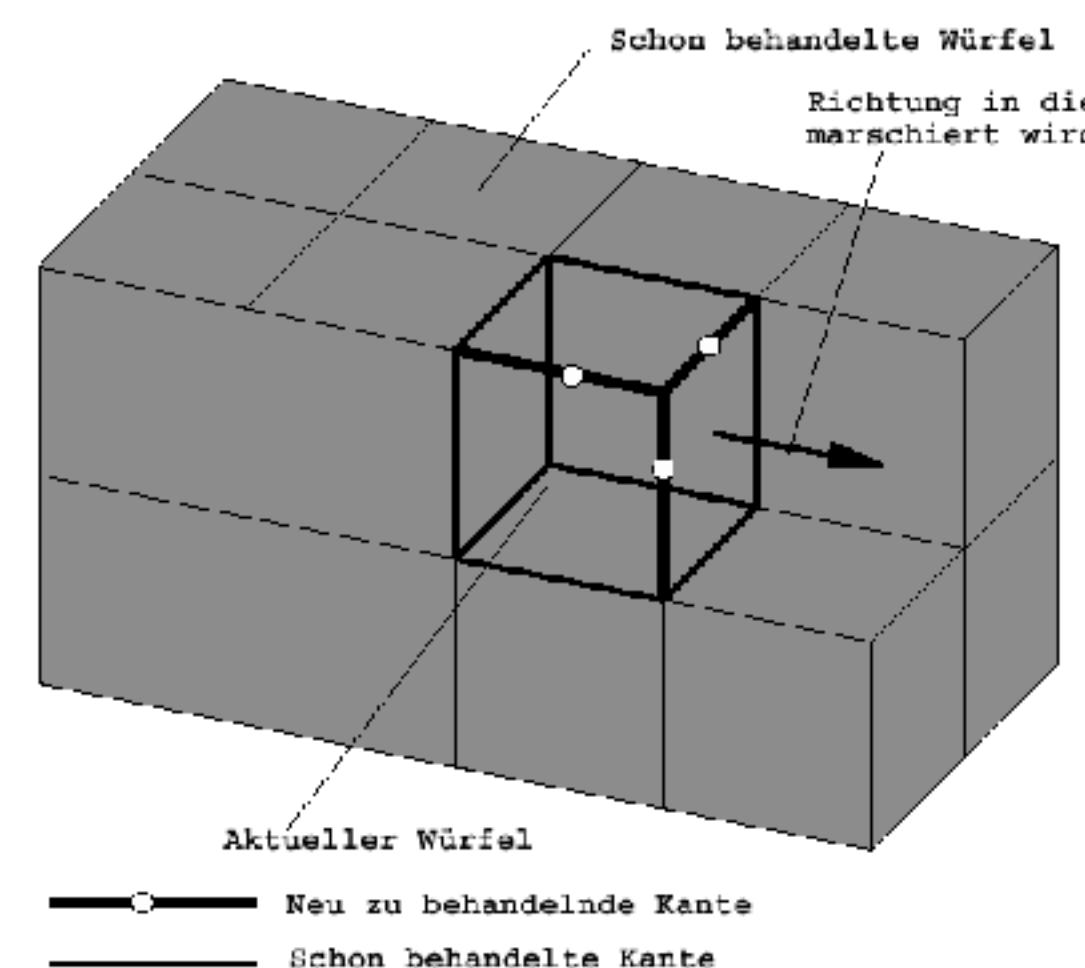
6. Compute the position of the cut vertices by linear interpolation:

$$\mathbf{v}_s = t\mathbf{v}_a + (1 - t)\mathbf{v}_b$$

$$t = \frac{F(\mathbf{v}_b)}{F(\mathbf{v}_b) - F(\mathbf{v}_a)}$$

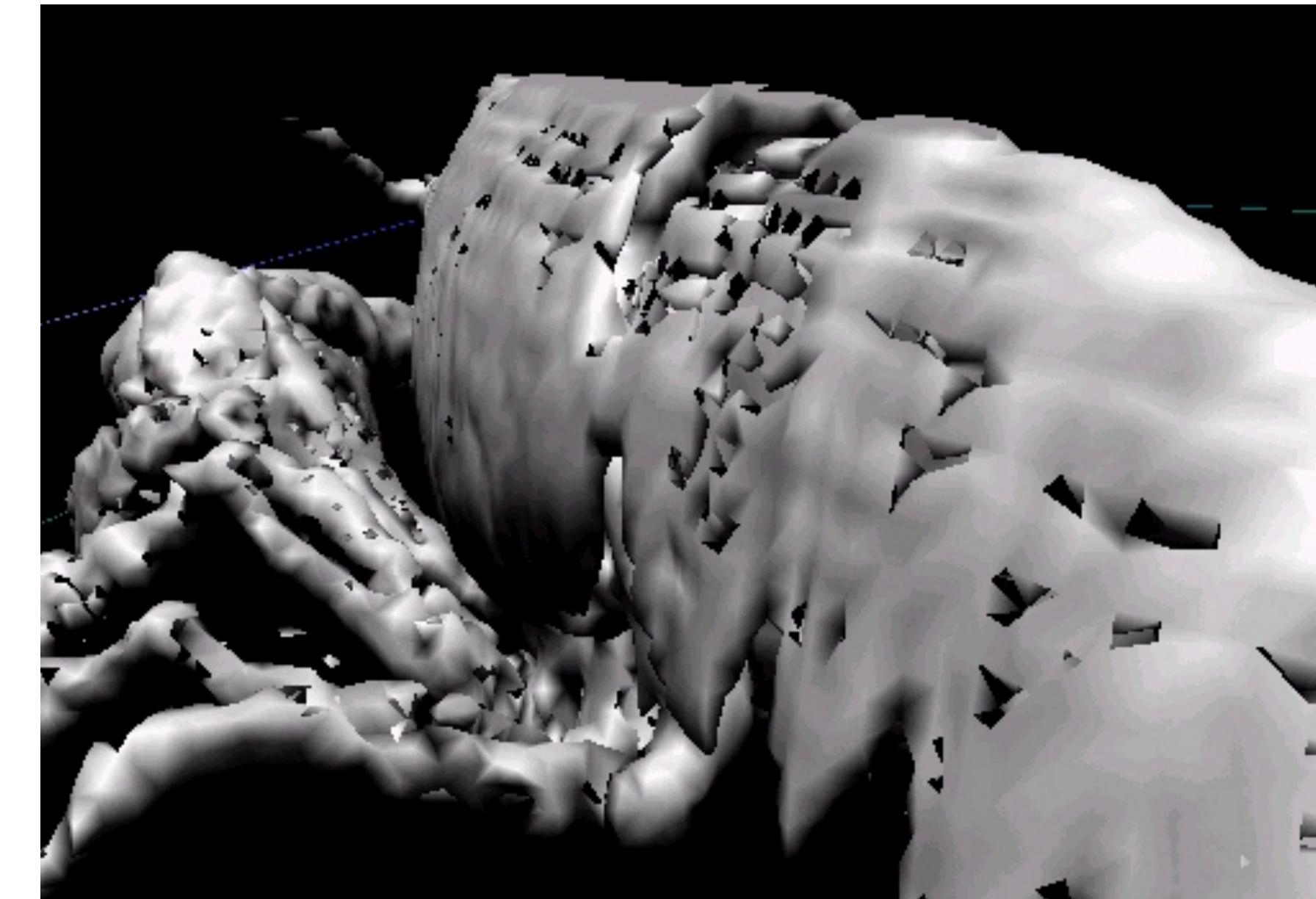
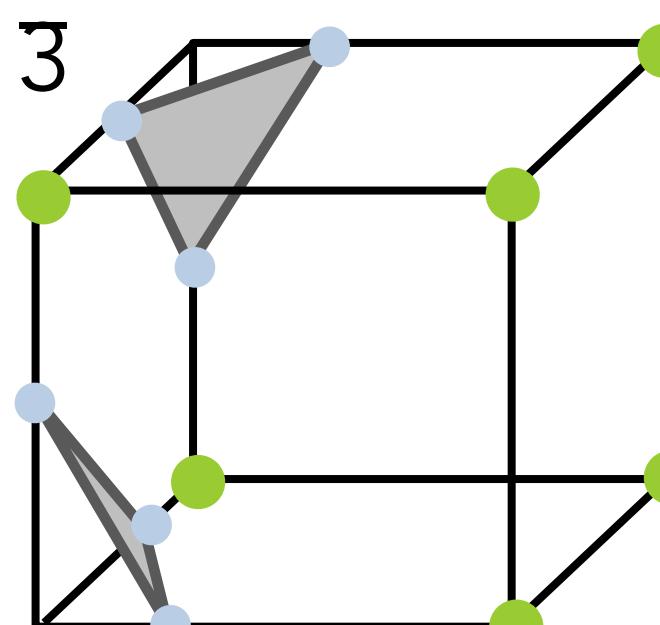
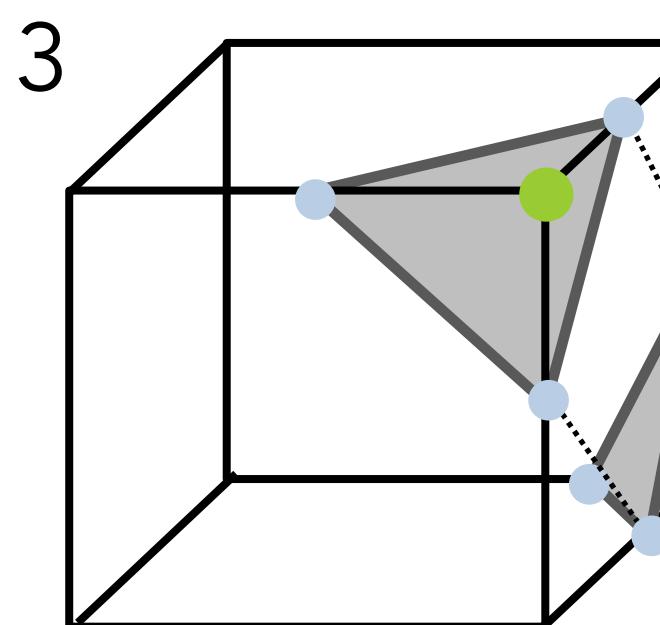


7. Move to the next cube



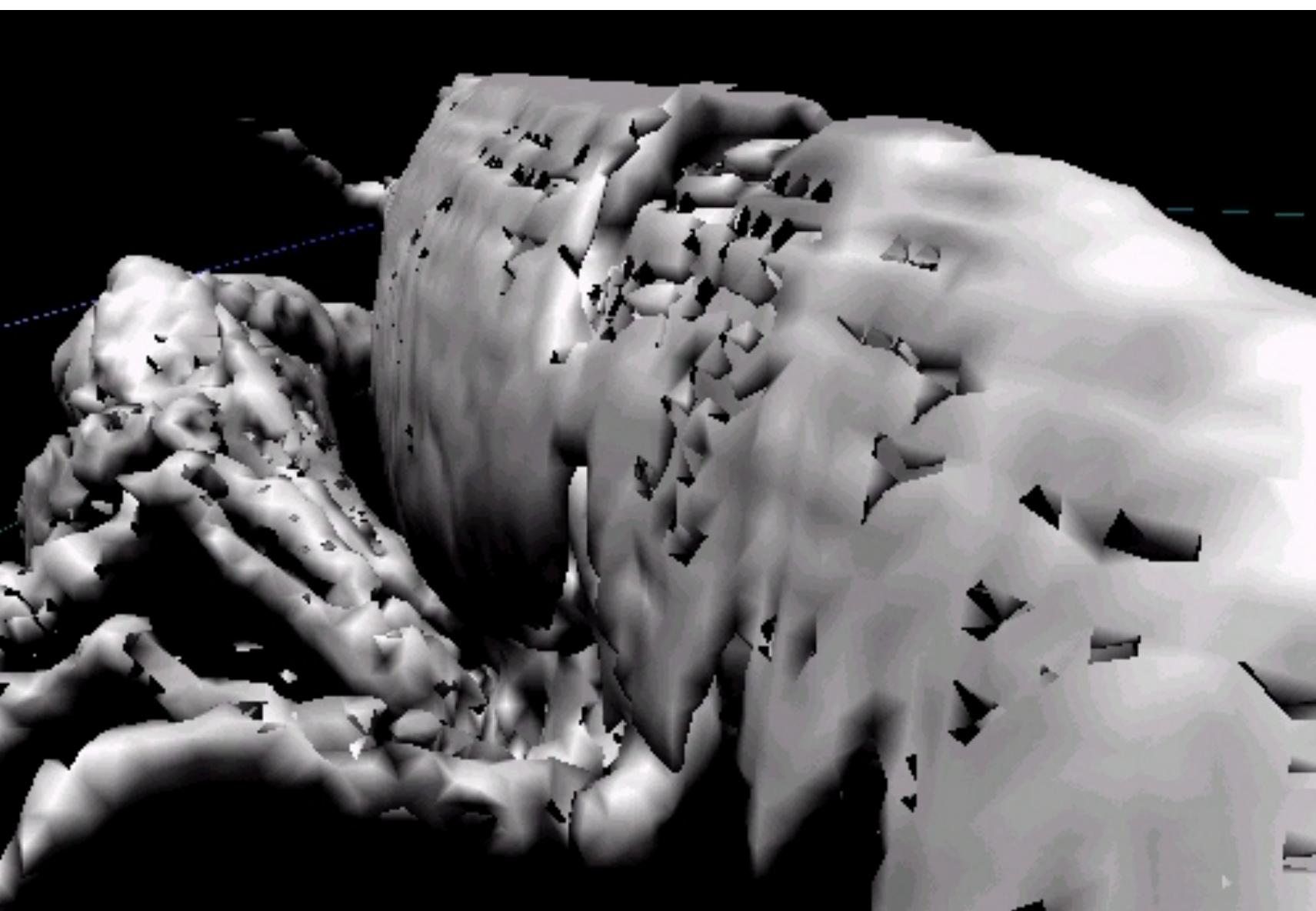
Marching Cubes – Problems

- Have to make consistent choices for neighboring cubes – otherwise get holes

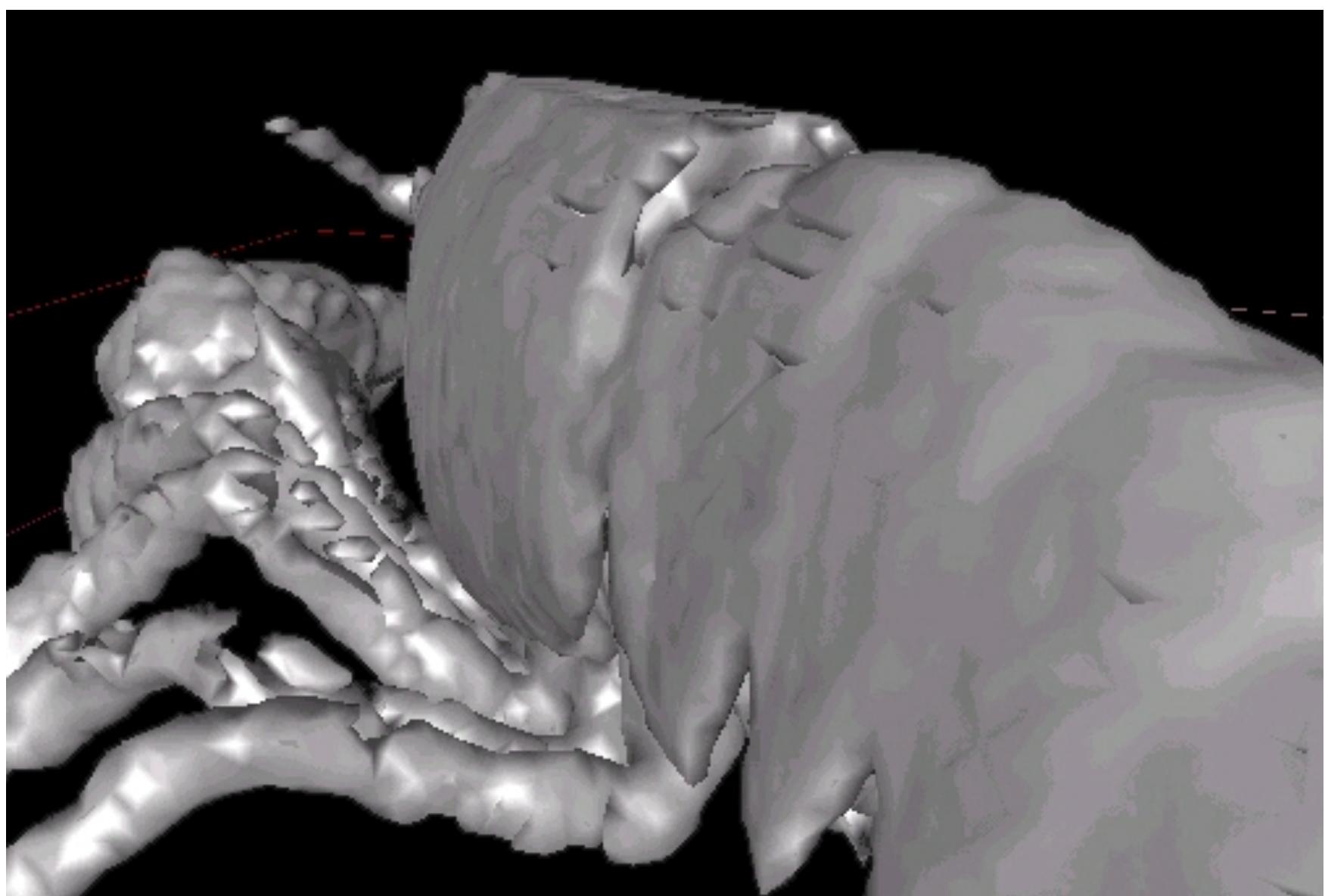


Marching Cubes – Problems

- Resolving ambiguities



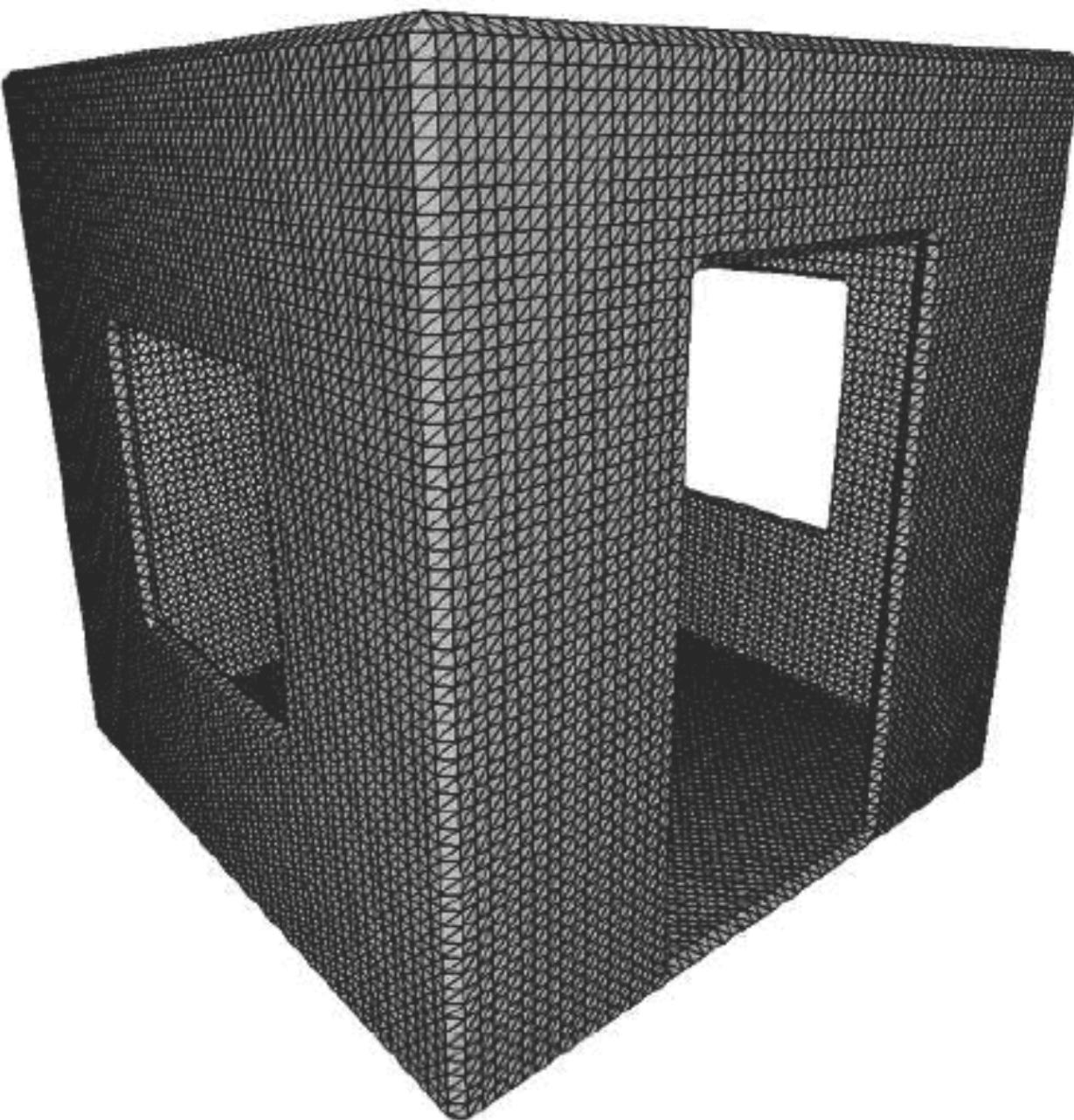
Ambiguity



No Ambiguity

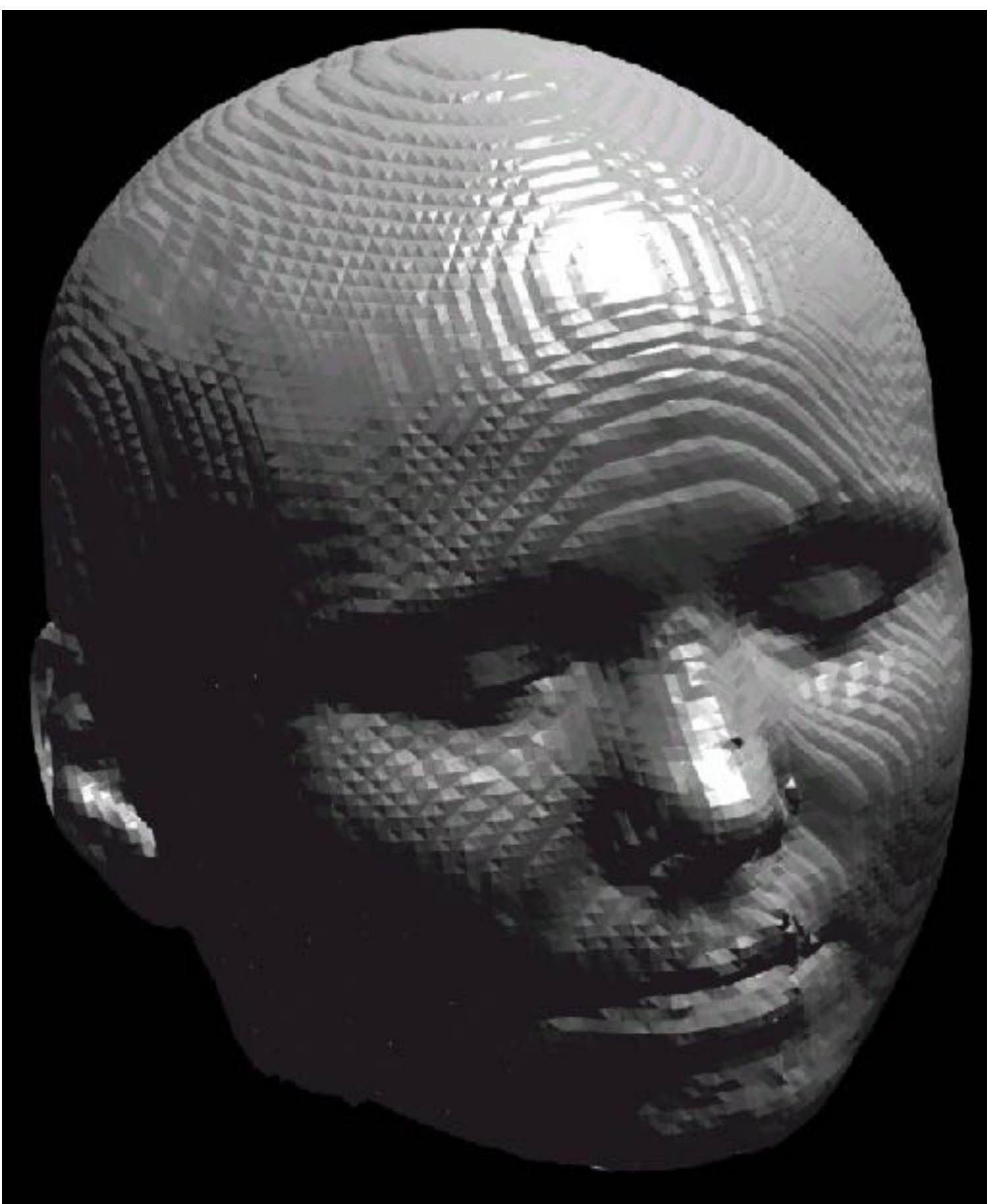
Marching Cubes – Problems

- Grid not adaptive
- Many polygons required to represent small features



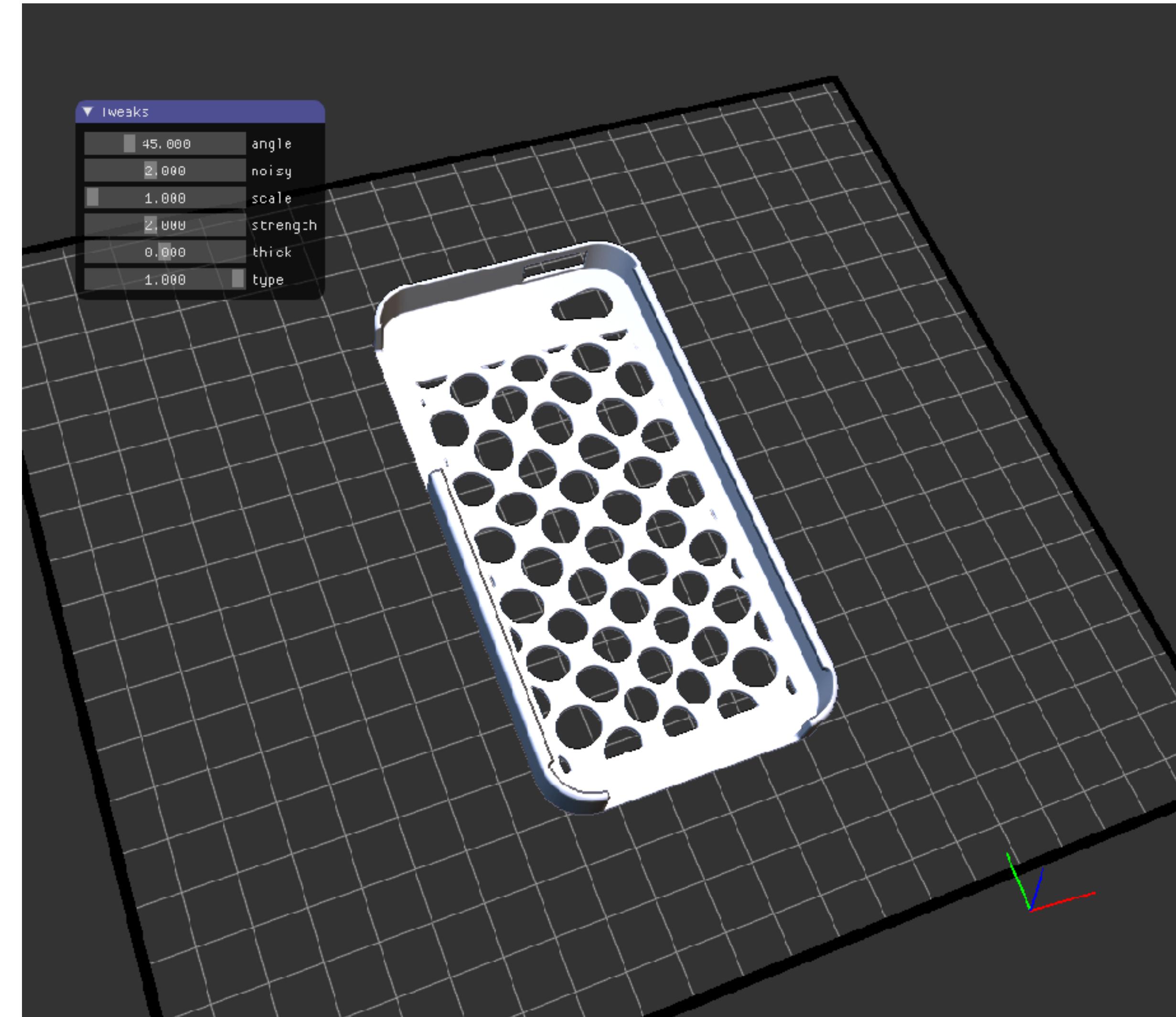
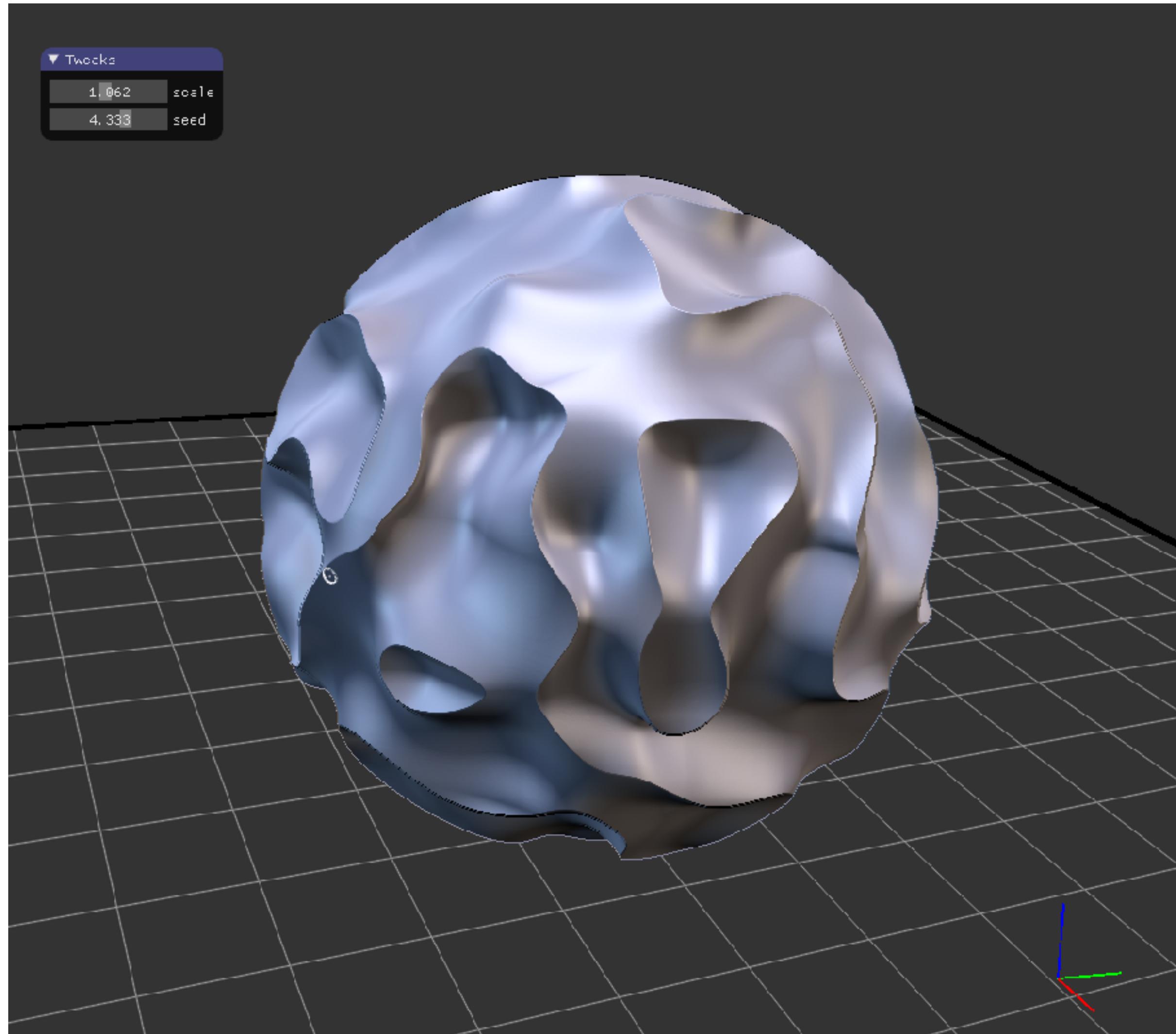
Images from: "Dual Marching Cubes: Primal Contouring of Dual Grids"
by Schaeffer et al.

Marching Cubes – Problems

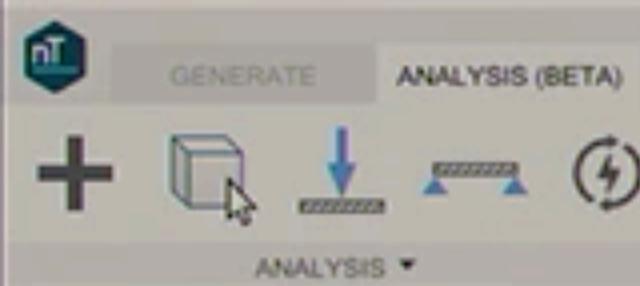


Implicit Modeling In Additive Manufacturing

IceSL

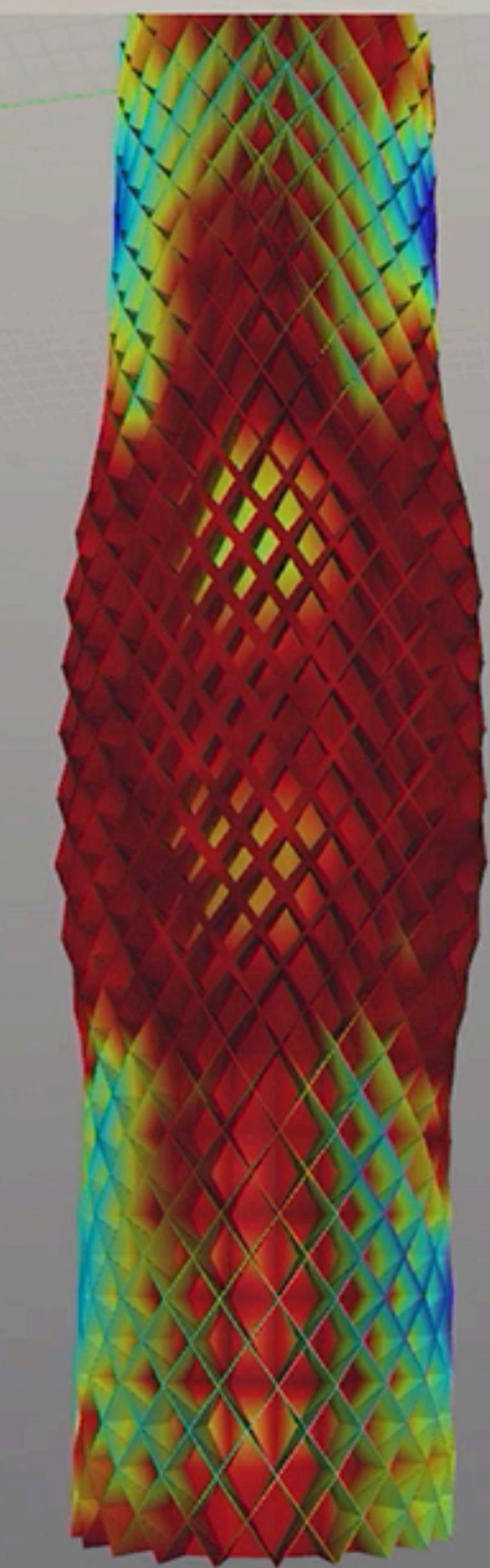


<http://shapeforge.loria.fr/icesl/>

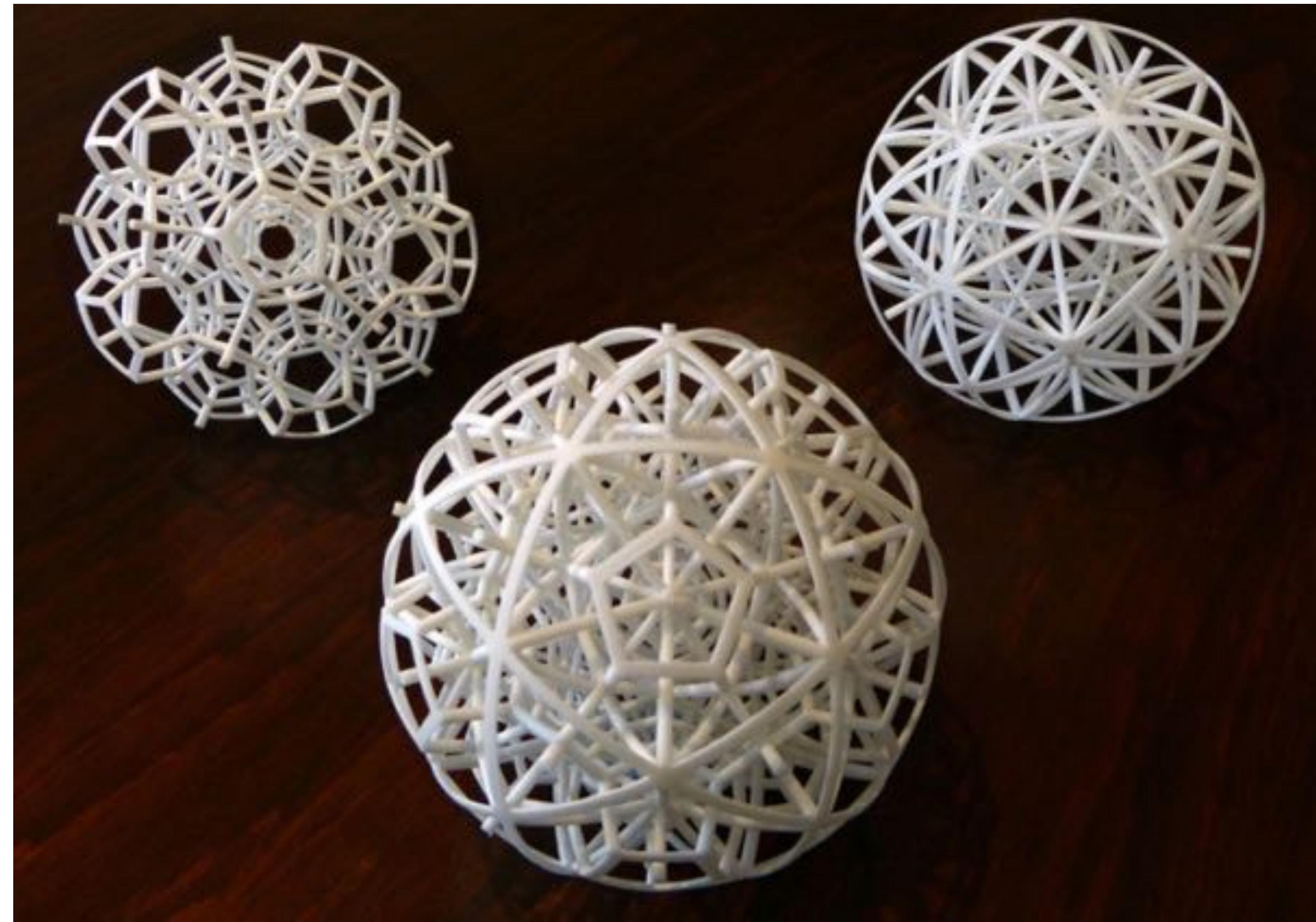


Analysis Objects

- ▼ Studies
 - ▼ Lattice Size Optimization
 - ▼ Study Objects
 - Single-Surface Conformal Lattice
 - ▼ Selection Objects
 - sheet_876
- ▼ Load Case
 - pressure 1
 - point constraint 1
- ▼ Results
 - all result



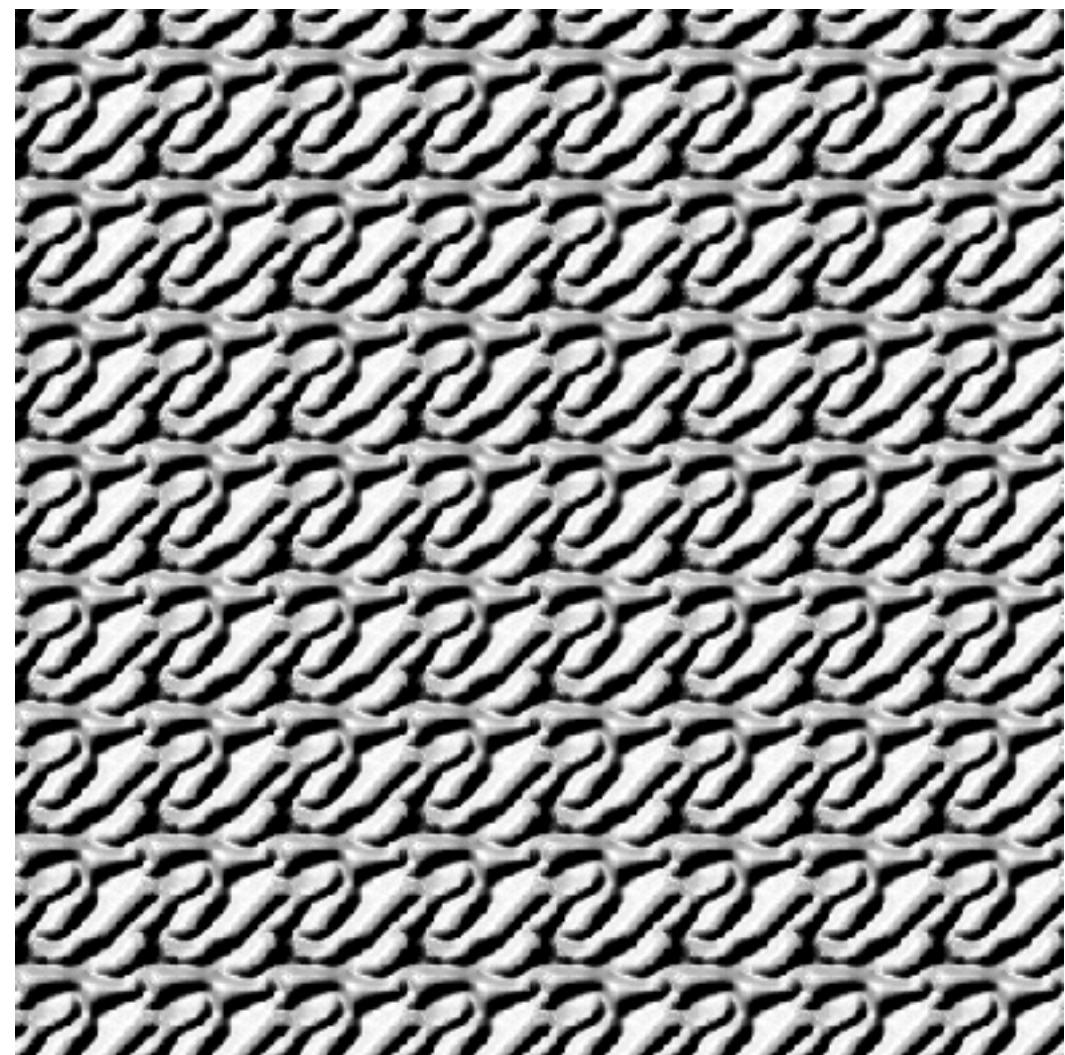
Tilings



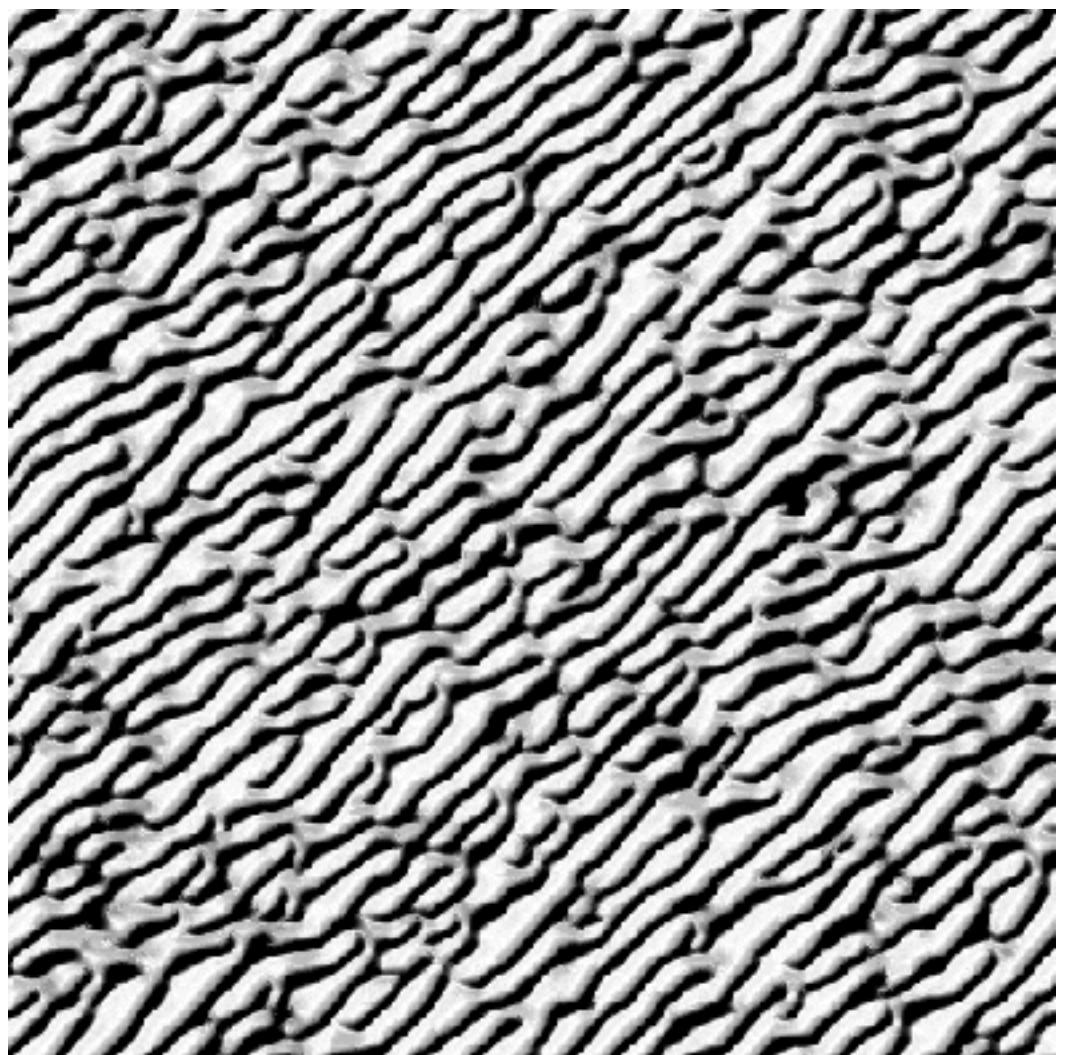
[Henry Segerman]

By-Example Texture Synthesis

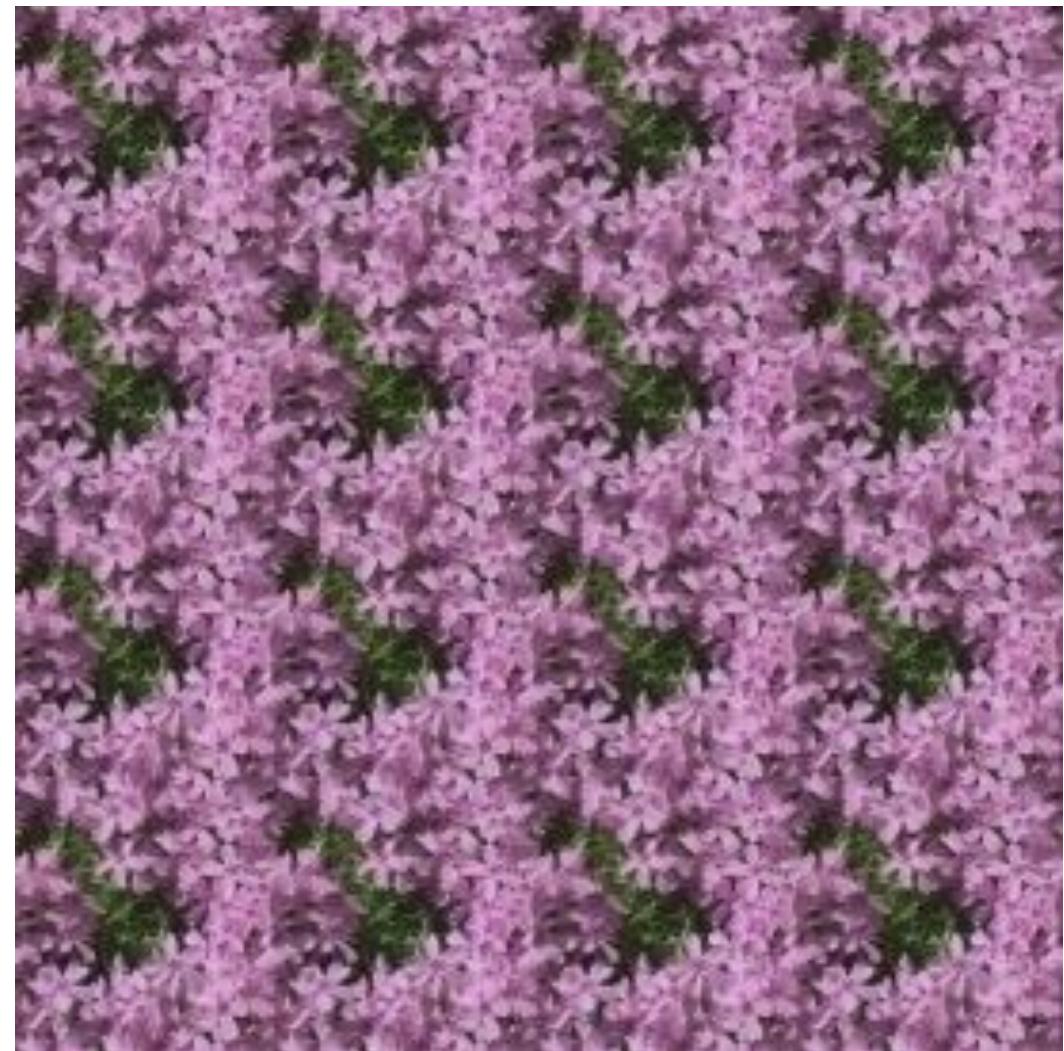
Texture synthesis from example



Tiling



Synthesis

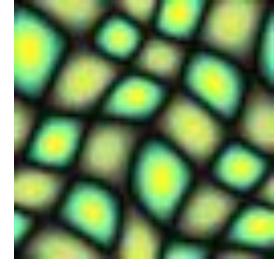


Tiling



Synthesis

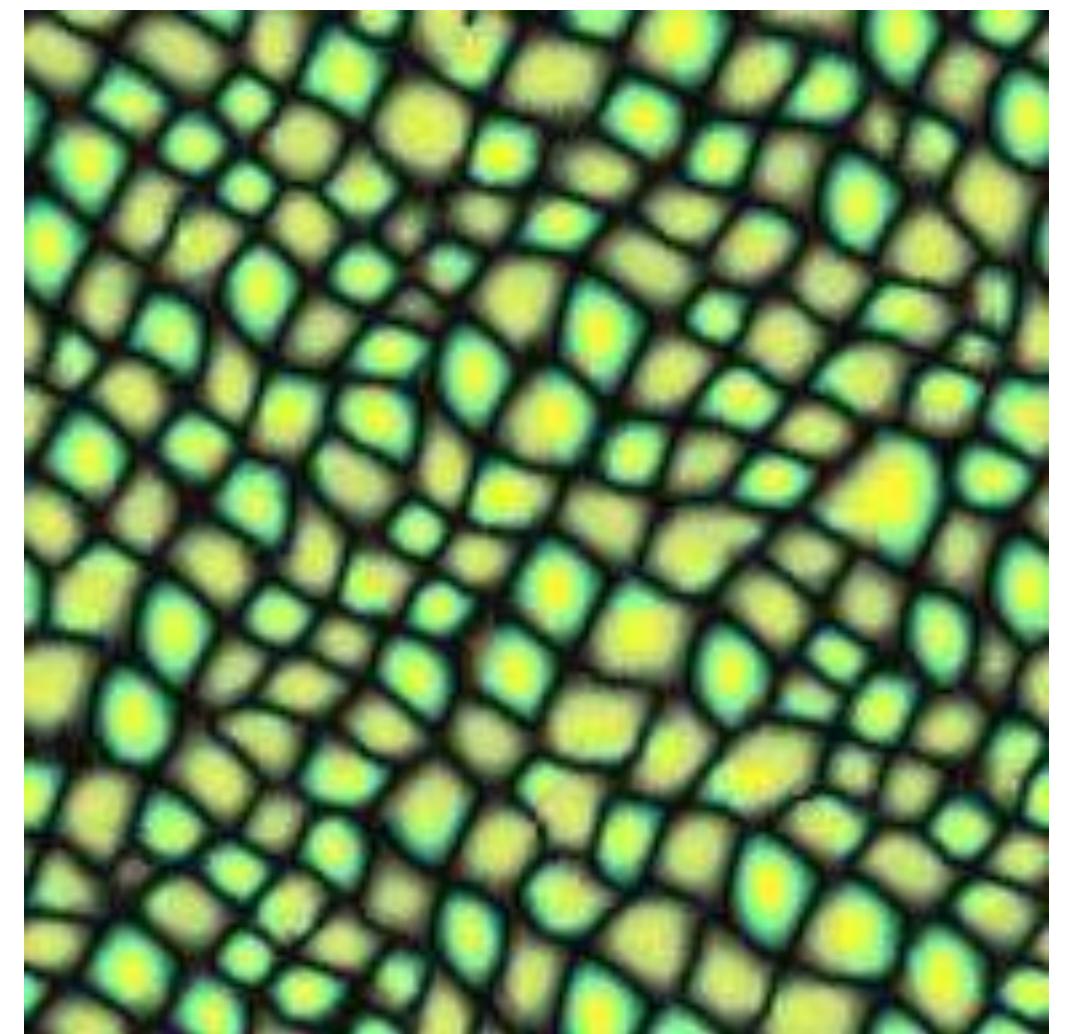
Basic Idea



Example Image



Random Initial Guess



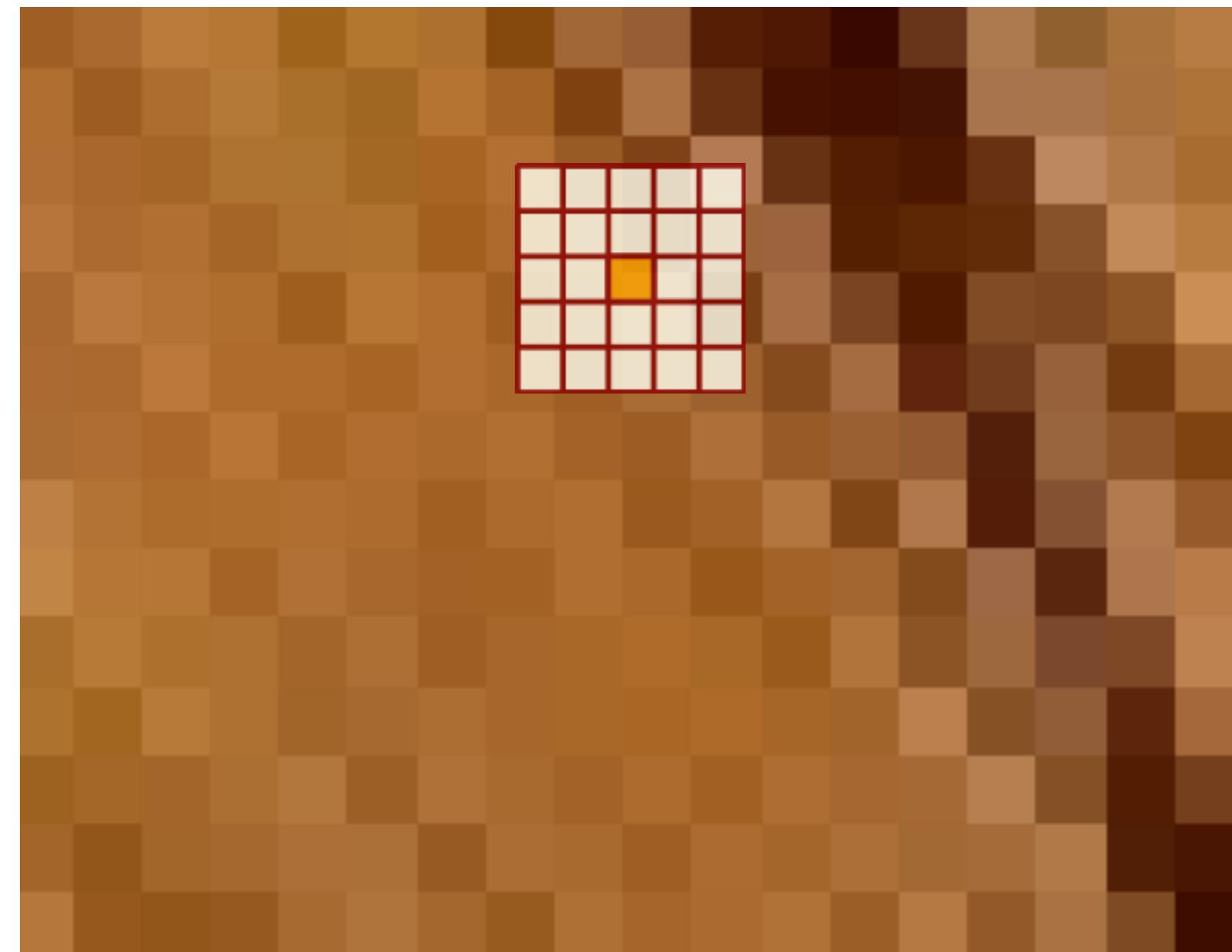
Final Result

Measuring Similarity

- Neighborhood Matching:



Exemplar



Texture being synthesized

What is a good exemplar?

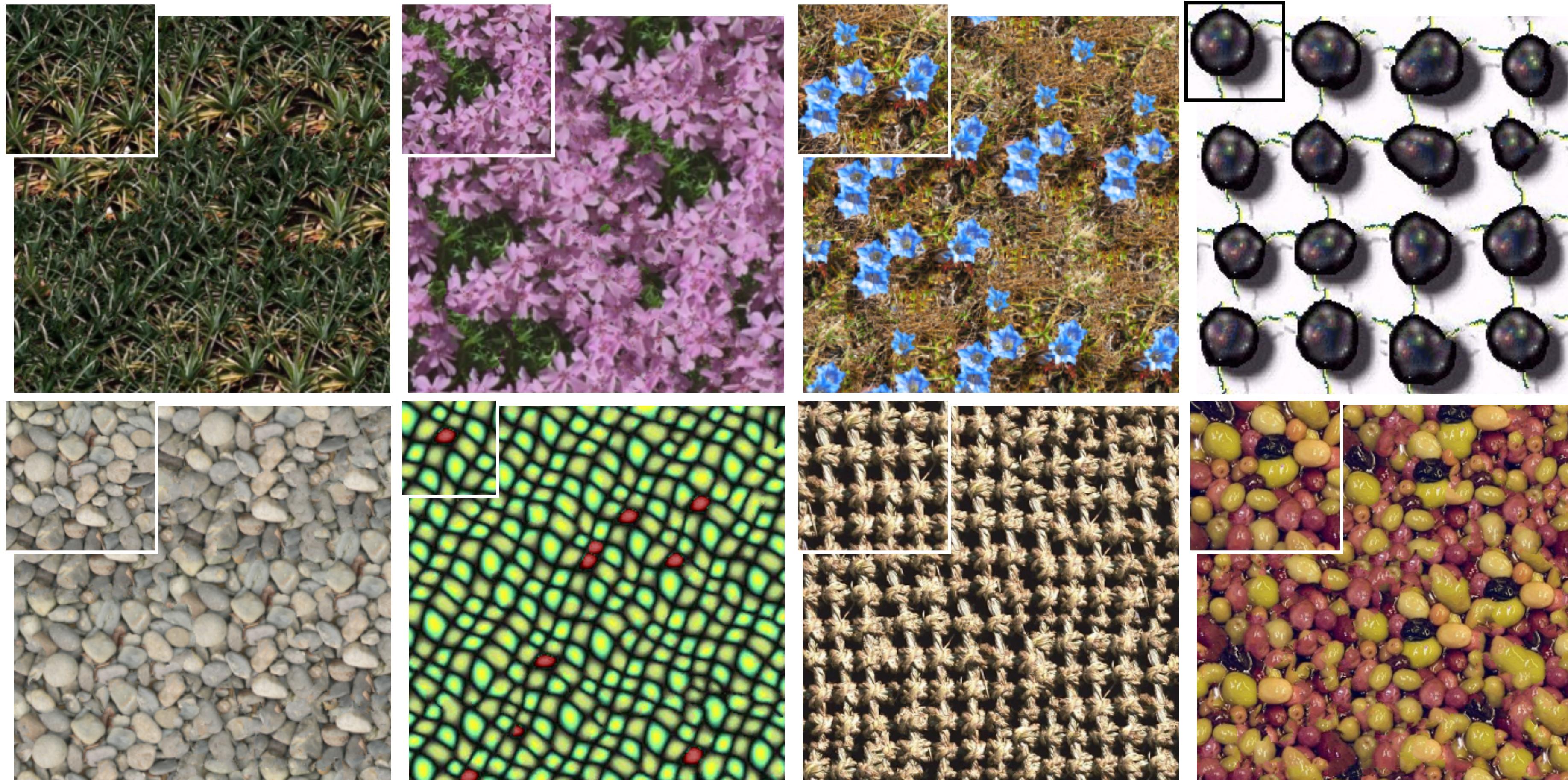


✓ Stochastic Image



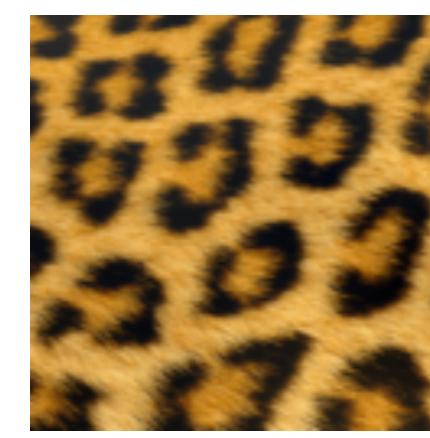
✗ Structured Image

Results

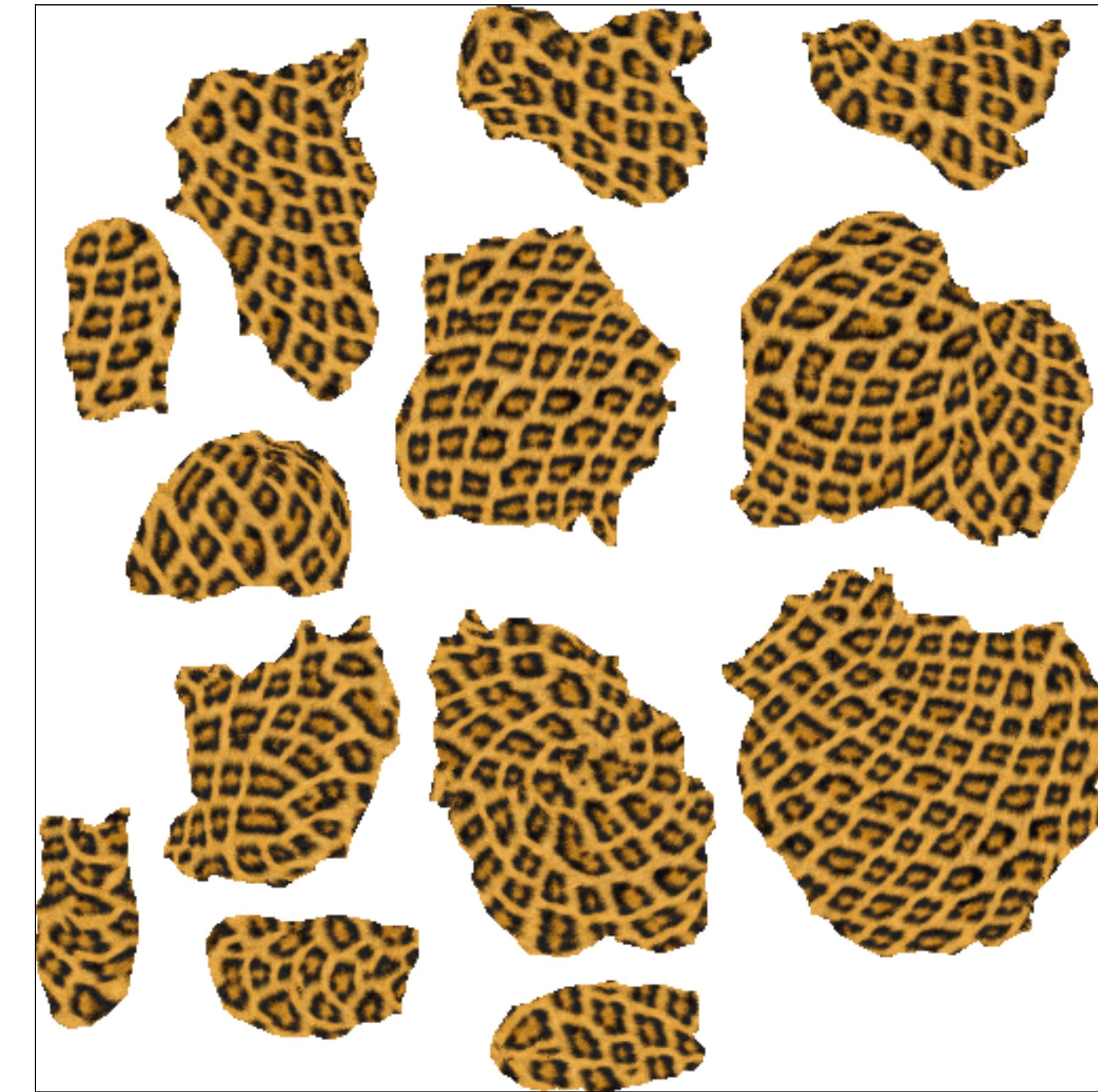


<http://research.microsoft.com/projects/ParaTexSyn/>

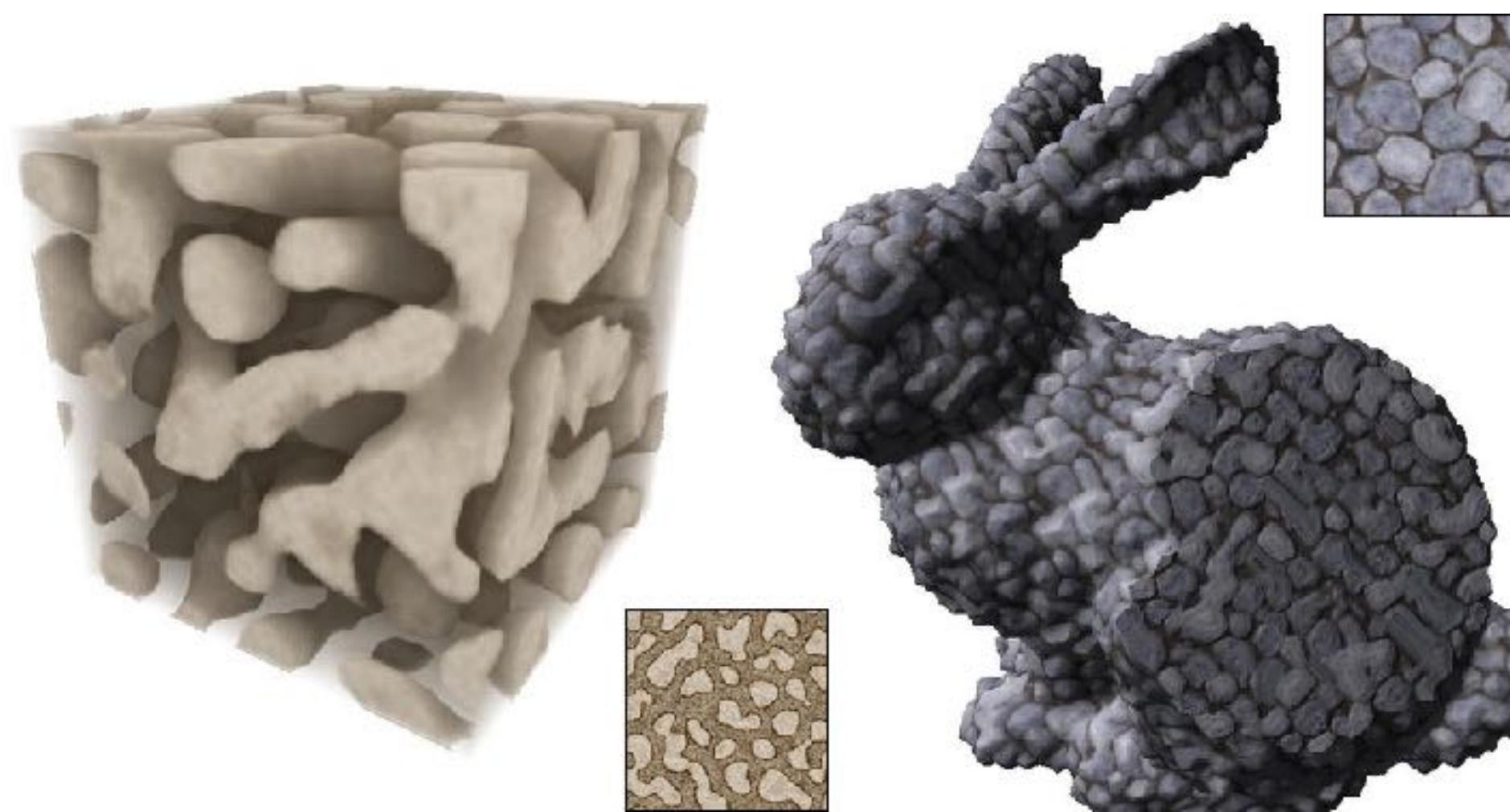
Surface Texture Synthesis



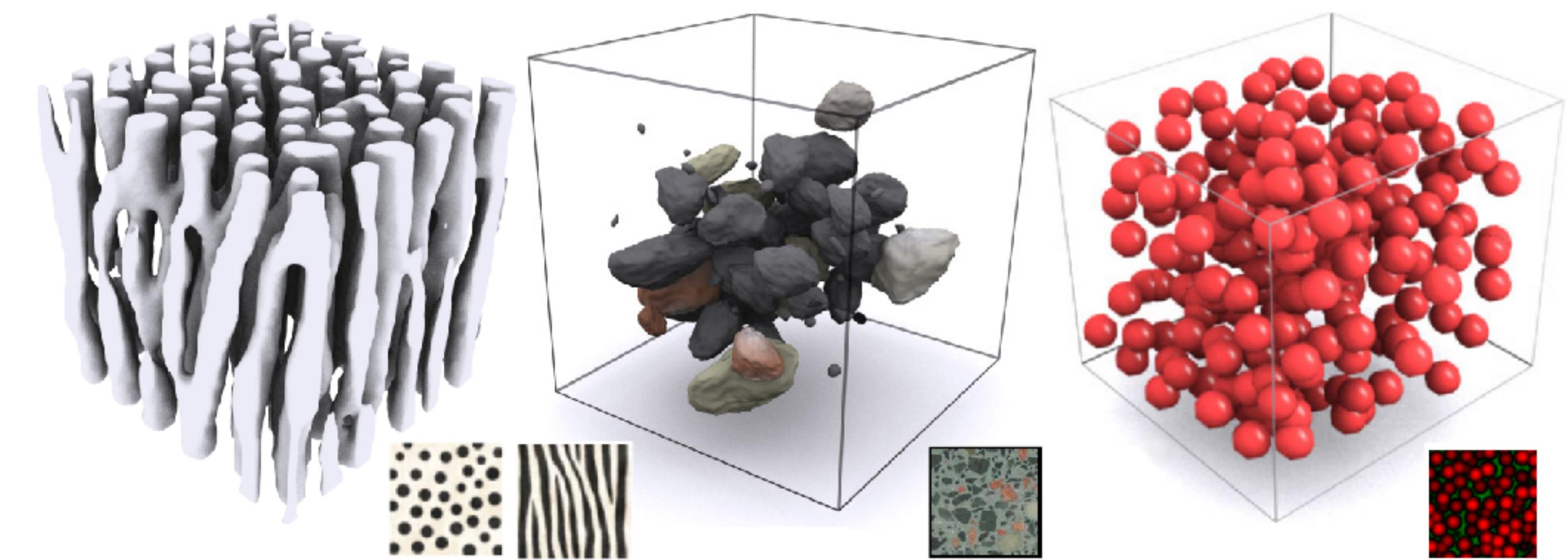
Exemplar



By-Example Solid Textures

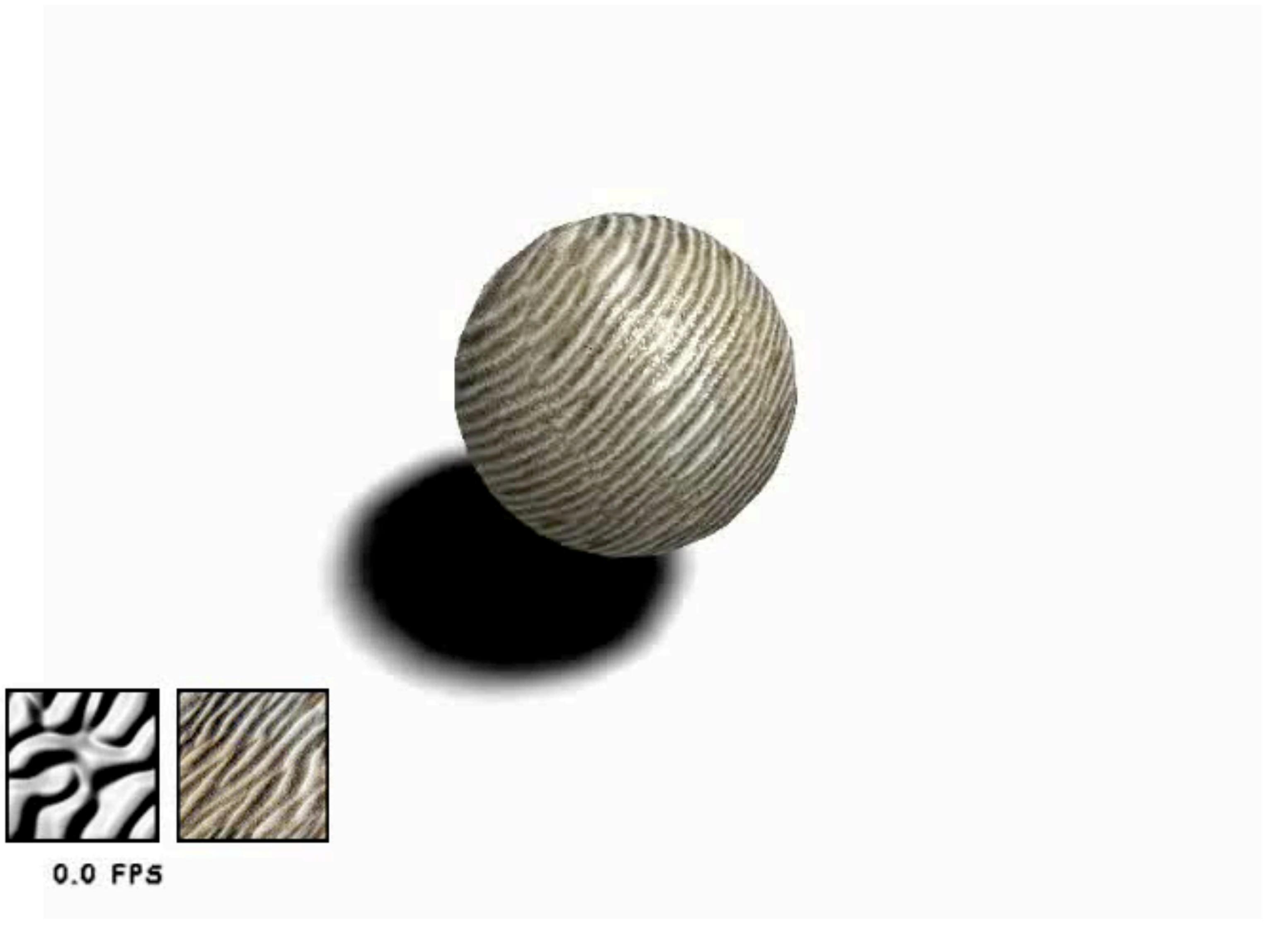


Solid Texture Synthesis [Kopf07]



Lazy Solid Texture Synthesis [Dong08]

By-Example Solid Textures



Lazy Solid Texture Synthesis
[Dong et al. 2008]



University
of Victoria

Computer Science

References

Fundamentals of Computer Graphics, Fourth Edition

4th Edition by Steve Marschner, Peter Shirley
Chapter 22

<https://www.shadertoy.com/view/4dS3Wd>

<https://thebookofshaders.com/>

<http://www.iquilezles.org/>

State of the Art in Example-based Texture Synthesis

Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, Greg Turk
Eurographics 2009 - State of the Art Reports 2009

State of the Art in Procedural Noise Functions

Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, D.S. Ebert, J.P. Lewis, Ken Perlin, Matthias Zwicker
Eurographics 2010 - State of the Art Reports 2010



**University
of Victoria**

Computer Science