

# Lecture 10

## 1. Applications of textures

在现代 GPU 上，纹理等于 内存 + 随机查询（点查询/段查询），具体到功能，有以下的分类

- Environment lighting
- Store microgeometry
- Procedural textures
- Solid modeling
- Volume rendering

### 1. Environment



Light from the environment



Rendering with the environment

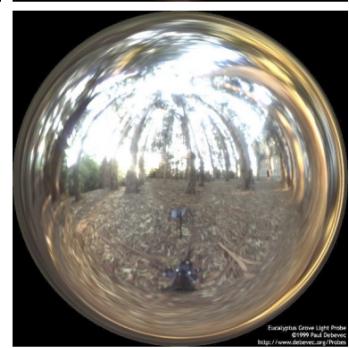
[Blinn & Newell 1976]

- 能将物体周围的光照正确地呈现在物体表面。可以将光照存储在物体表面

# Spherical Environment Map



*Hand with Reflecting Sphere.* M. C. Escher, 1935. lithograph



*Light Probes,* Paul Debevec

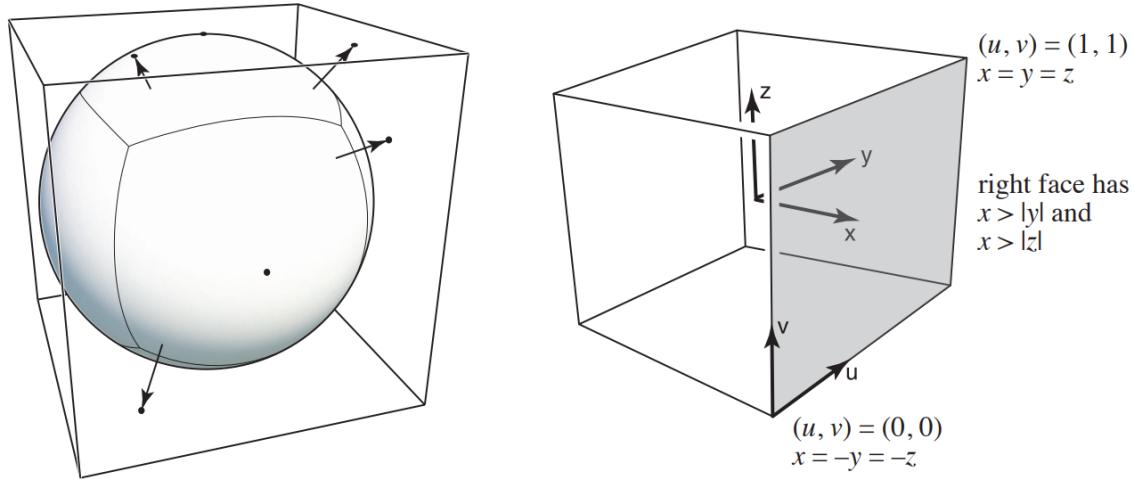
- 问题：由于球形的弧度问题，**顶部和底部容易失真**、



Prone to distortion (top and bottom parts)!

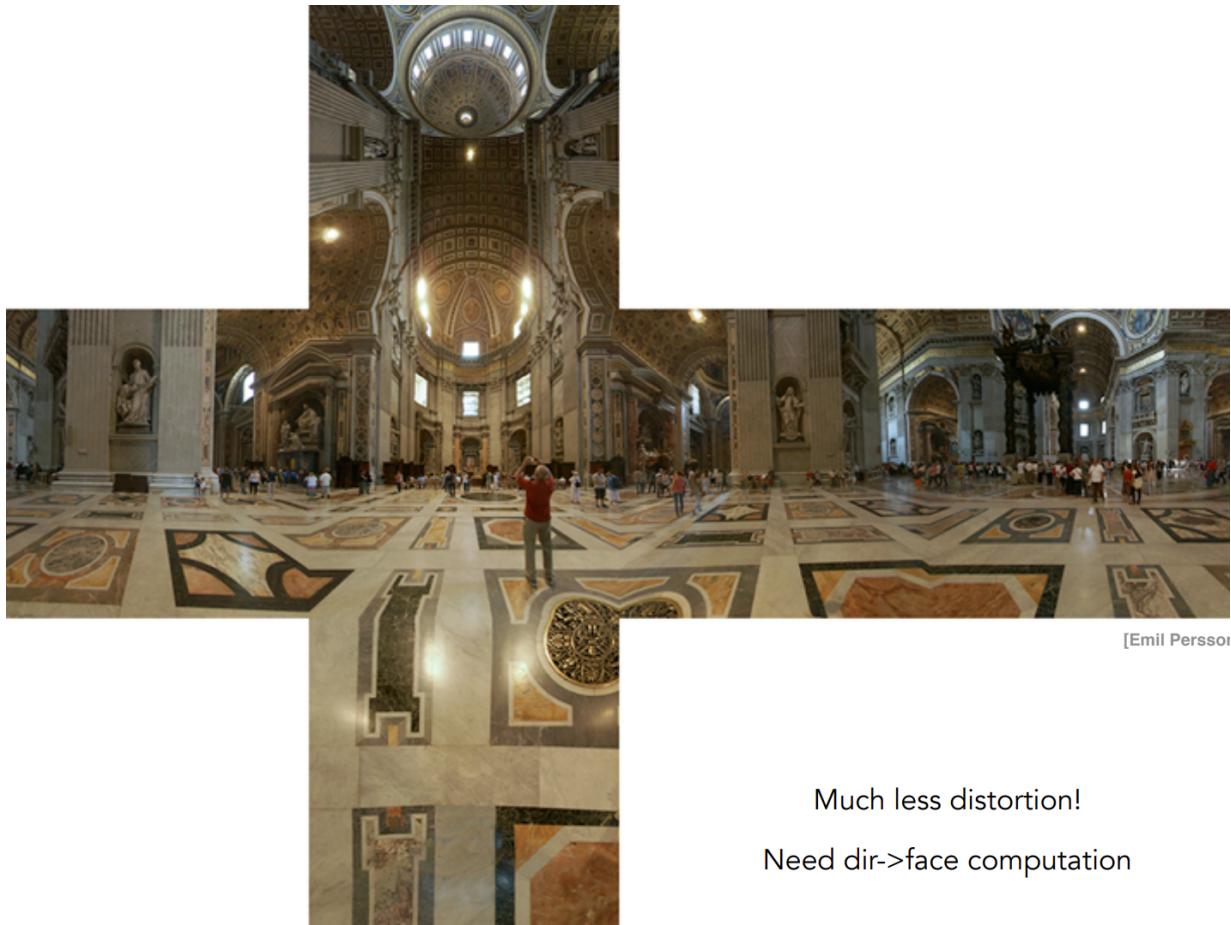
- 解决方式：使用正方体覆盖物体表面，用正方体存储光照

## Cube Map



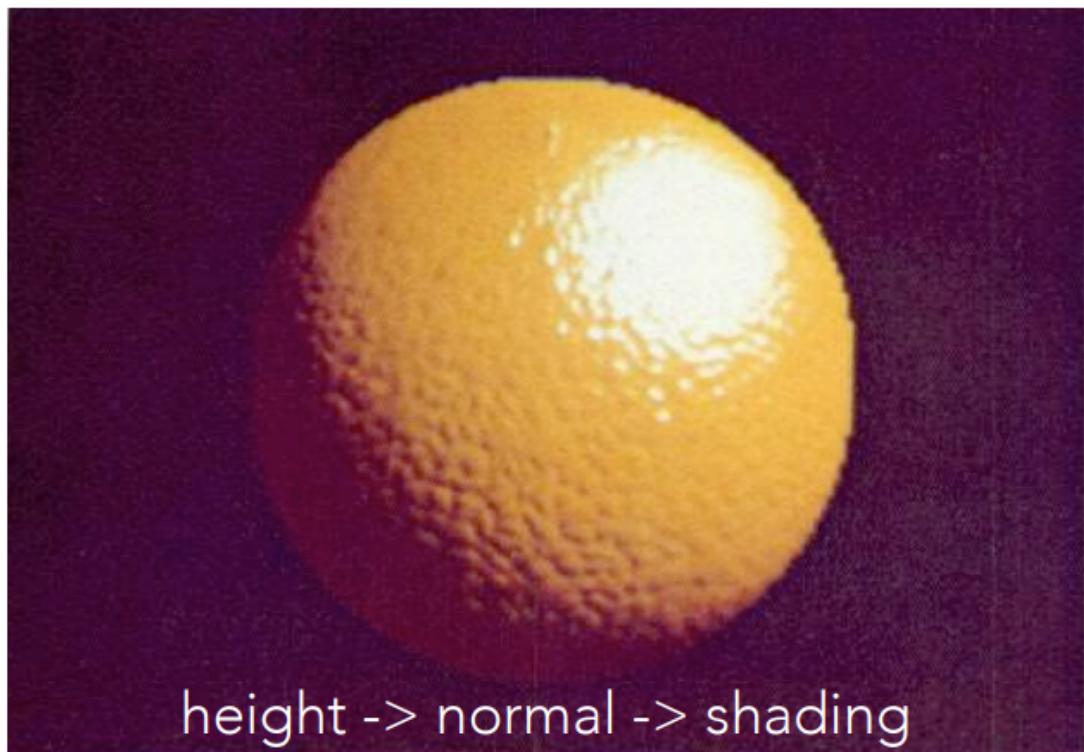
A vector maps to cube point along that direction.  
The cube is textured with 6 square texture maps.

- 效果：更少的失真 [Open: Pasted image 20240212131645.png](#)



## 2.Textures affect shading

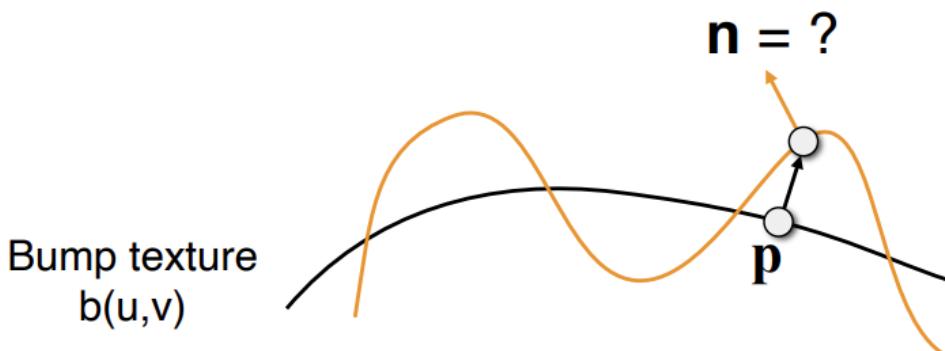
- 纹理的作用
  - 表示颜色
  - 表示任意点的相对高度，作为 **凹凸贴图 / 法线贴图** 的依据
  - **通过变化点的相对高度导致法线发生变化**，光照结果发生变化，使视觉产生变化，但其实其纹理没有发生变化。



◦

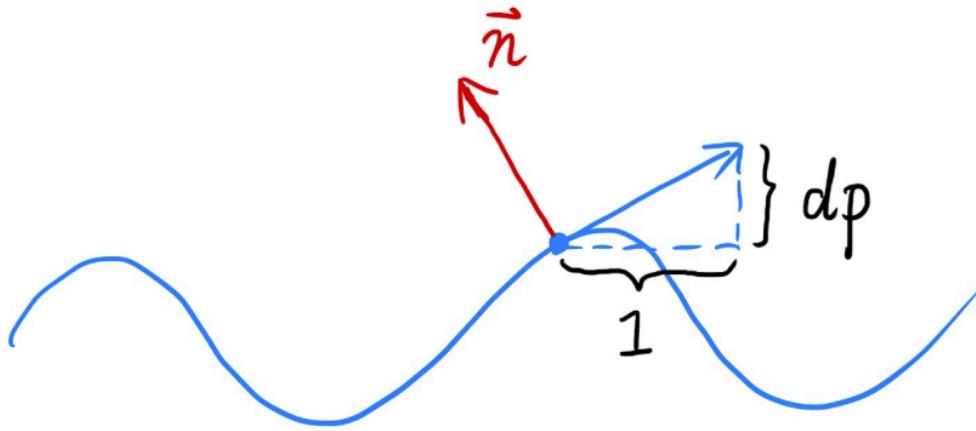
## 3.Bump Mapping

- 上文中纹理的作用，使得我们可以通过在不增加物体表面三角形数量的情况下，为渲染结果增加更多细节。
- 方式：**Perturb** 扰动表面单个像素的法线方向



- 2D (in flatland)

- 定义原表面法线方向  $n(p) = (0, 1)$
- 扰动后定义该点在曲线上的斜率为  $dp = c * [h(p + 1) - h(p)]$
- 扰动后定义该点的法线为  $n(p) = (-dp, 1). normalized()$

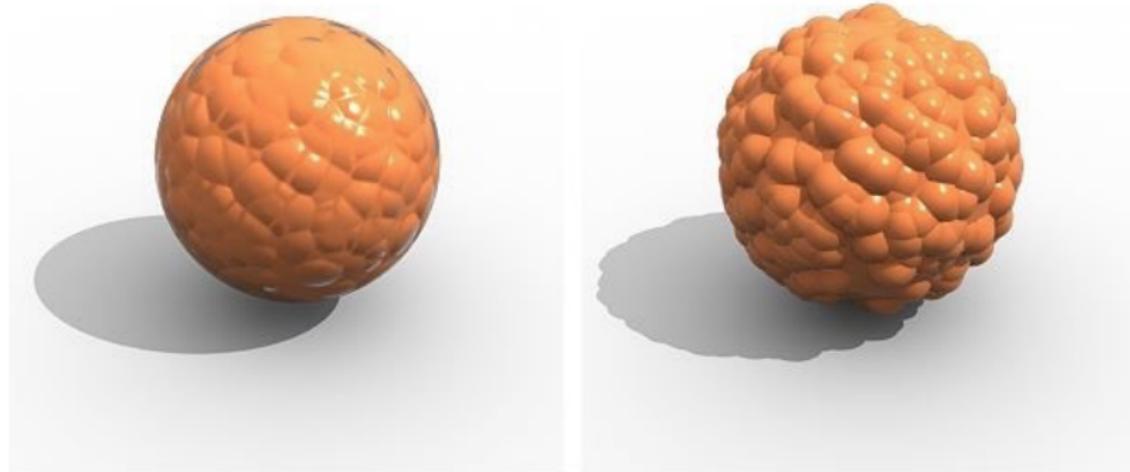


- 
- 3D
  - 定义原表面法线方向为  $n(p) = (0, 0, 1)$  **此处为计算方便，将所有点的坐标从原坐标系转换到局部坐标系中的  $(0, 0, 1)$**
  - 扰动后求  $(u, v)$  坐标系下两个方向的斜率
  - $dp/du = c1 * [h(u + 1) - h(u)]$
  - $dp/dv = c2 * [h(v + 1) - h(v)]$
  - 扰动后定义该点的法线为  
 $n = (-dp/du, -dp/dv, 1). normalized()$
  - **此时将 n 转换成原坐标系中的线段即可**

## 4. Displacement mapping

- 对比 Bump mapping, 其真正的移动了顶点, 所以相比于 Bump, 表面需要更多的三角形才能支撑其移动顶点后插值三角形内部点的准确

性。



Bump / **Normal** mapping

Displacement mapping

- 其他应用
  - 3D 噪声
  - 预计算着色
  - 3D 纹理与 **体积渲染**

## 2. Introduction to geometry

几何在物体上是多种多样的，包括汽车的车门结构，发动机的结构，衣物布料的结构，更甚至于一张城市的全景照片，其中包含的几何信息是

巨量的。



对于表示几何的方式，统称为

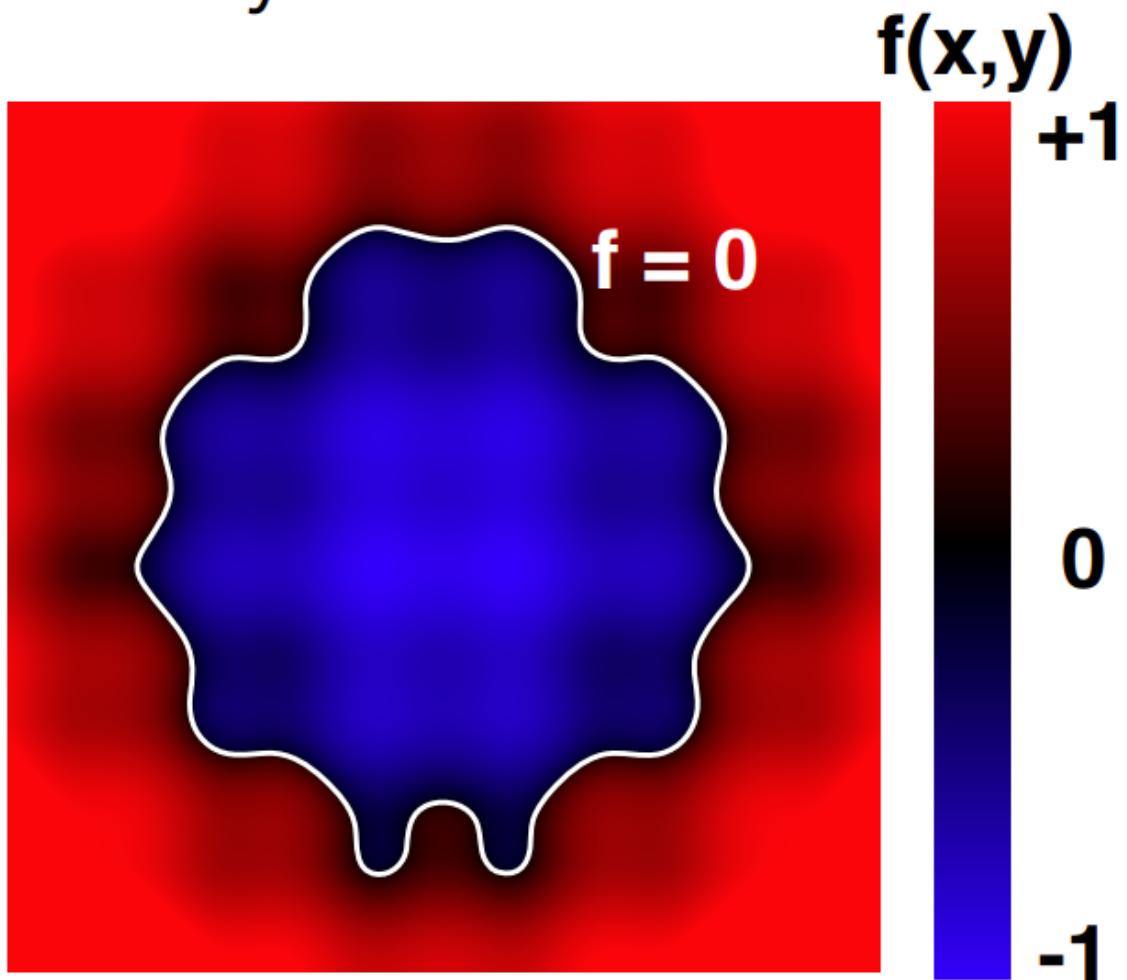
- Implicit 隐式函数表示
- Explicit 显示函数表示

## 1. Implicit

使用分类点的方法

- 点满足特定的关系，如函数或方程， $x^2 + y^2 + z^2 = 1$  就是一种

- 最一般的形式是  $f(x, y, z) = 0$ ，即下图



- 优点
  - 容易检测点是否在几何体外/内/上
- 缺点
  - 不容易采样，也不容易得到几何体的直观外形

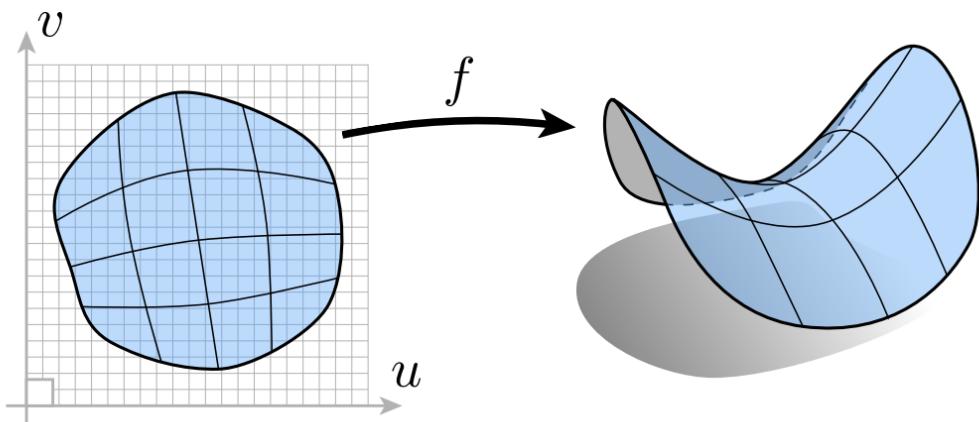
## 2. Explicit

使用点集合表示的方法

- 所有点都 **直接/通过参数映射** 的方式给出

Generally:

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^3; (u, v) \mapsto (x, y, z)$$



- 优点
  - 容易采样
- 缺点
  - 不容易判断点的几何体的关系

### 3. Some Implicit Representation

- Algebraic Surfaces 代数表面
  - 使用函数表达式表达一个几何体
  - 优点：准确
  - 缺点：无法表示复杂的几何体

Surface is zero set of a polynomial in  $x, y, z$



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



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



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



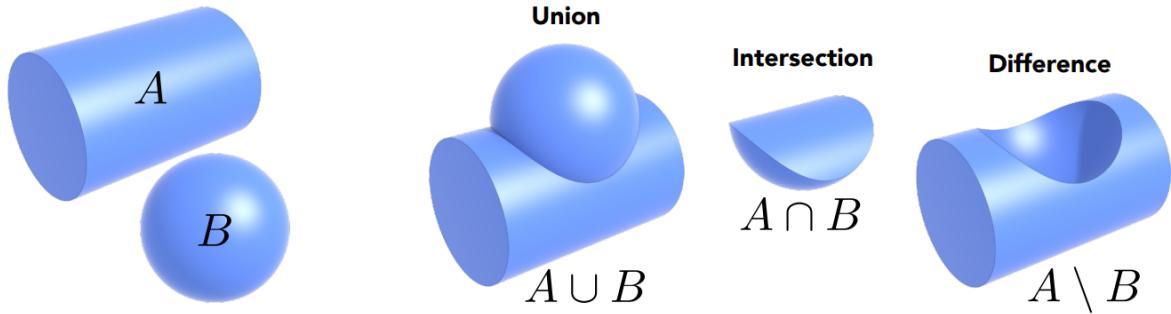
$$x^2 z^3 + \frac{9y^2 z^3}{80}$$

More complex shapes?

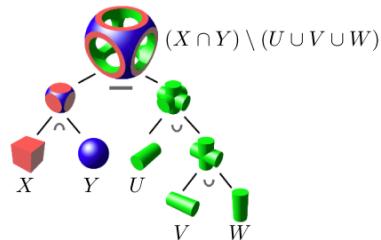
## 4. Constructive Solid Geometry

- 通过 并/交/差 的关系组合隐式几何形状

Combine implicit geometry via Boolean operations

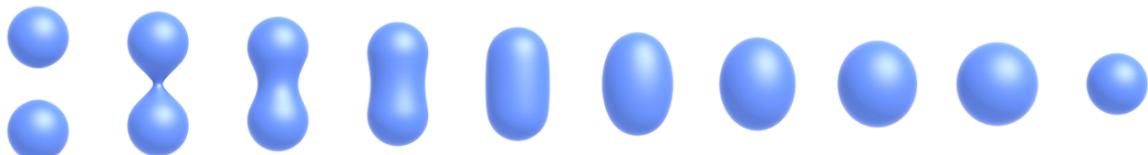


Boolean expressions:

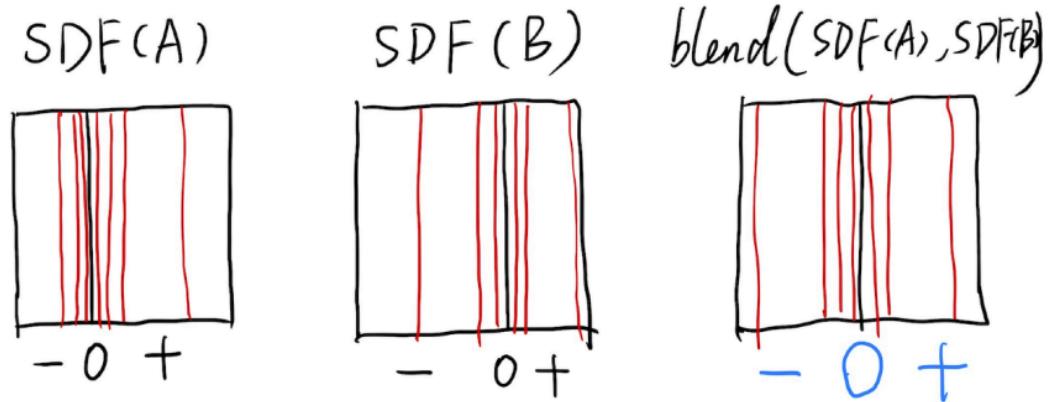


## 5. Distance Functions

- 使用逐渐融合表面的方式，组合物体
- 定义距离函数：任意点到指定平面最短的距离 (有向距离 **SDF**)
- 融合方式

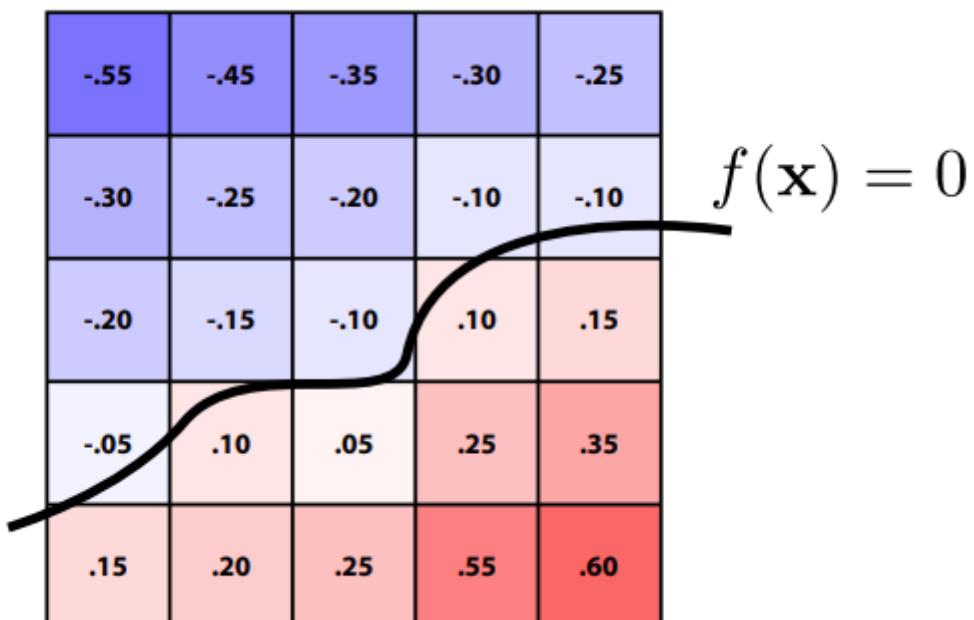


- SDF 通过有向距离的性质，**区分在边界/边界左边/边界右边**，通过叠加两个物体的 SDF 距离，可以实现融合



## 6. Level Set Methods

- 在一块区域内存储隐式函数的不同点的值，通过连接**值为0**的点，得到隐式函数的几何外形



## 7. Fractals

- 分型：指物体自身不断生成近似于原部分的形状

