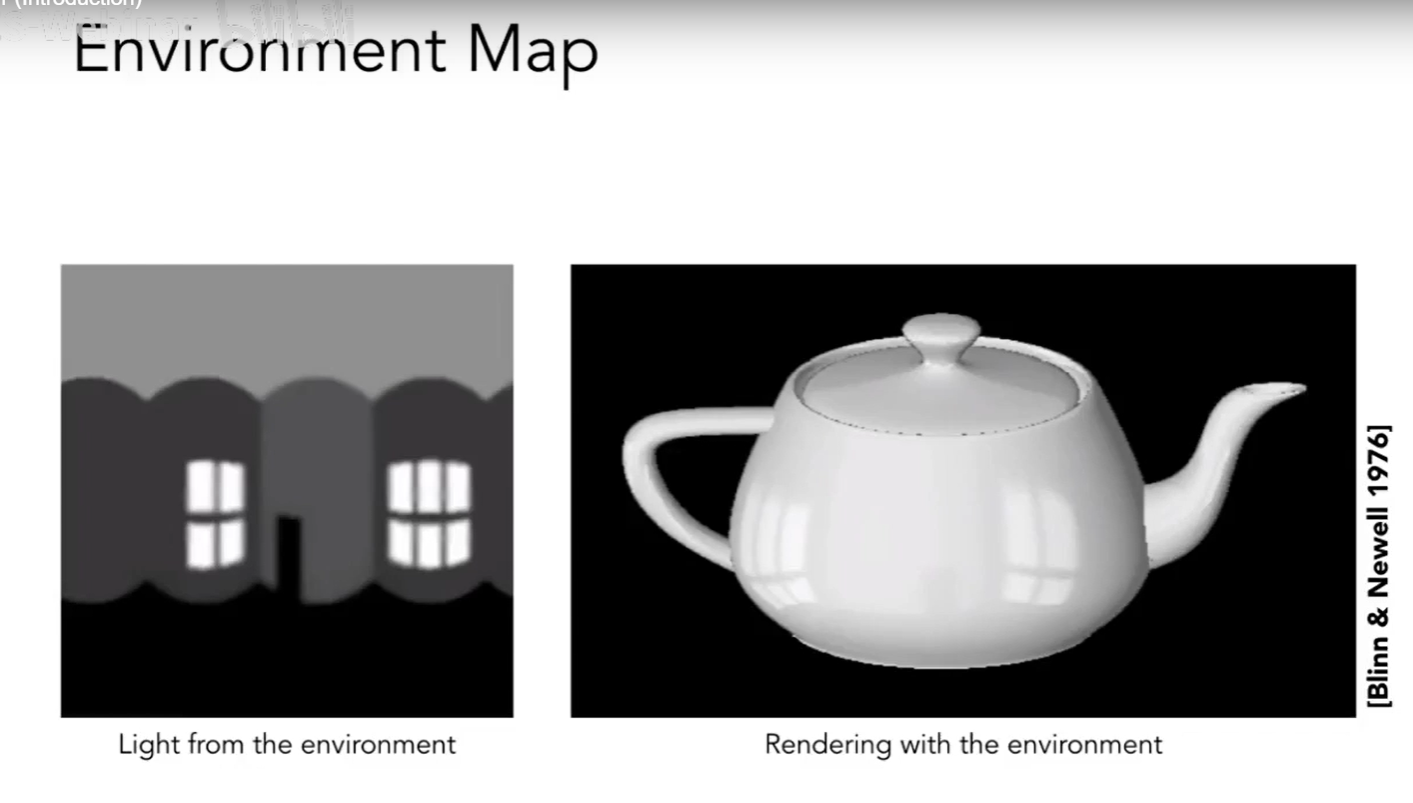
什么是纹理？

现代GPU中，纹理=内存+多次查询（滤波），可以理解成一块数据，可以做不同类型的查询的数据



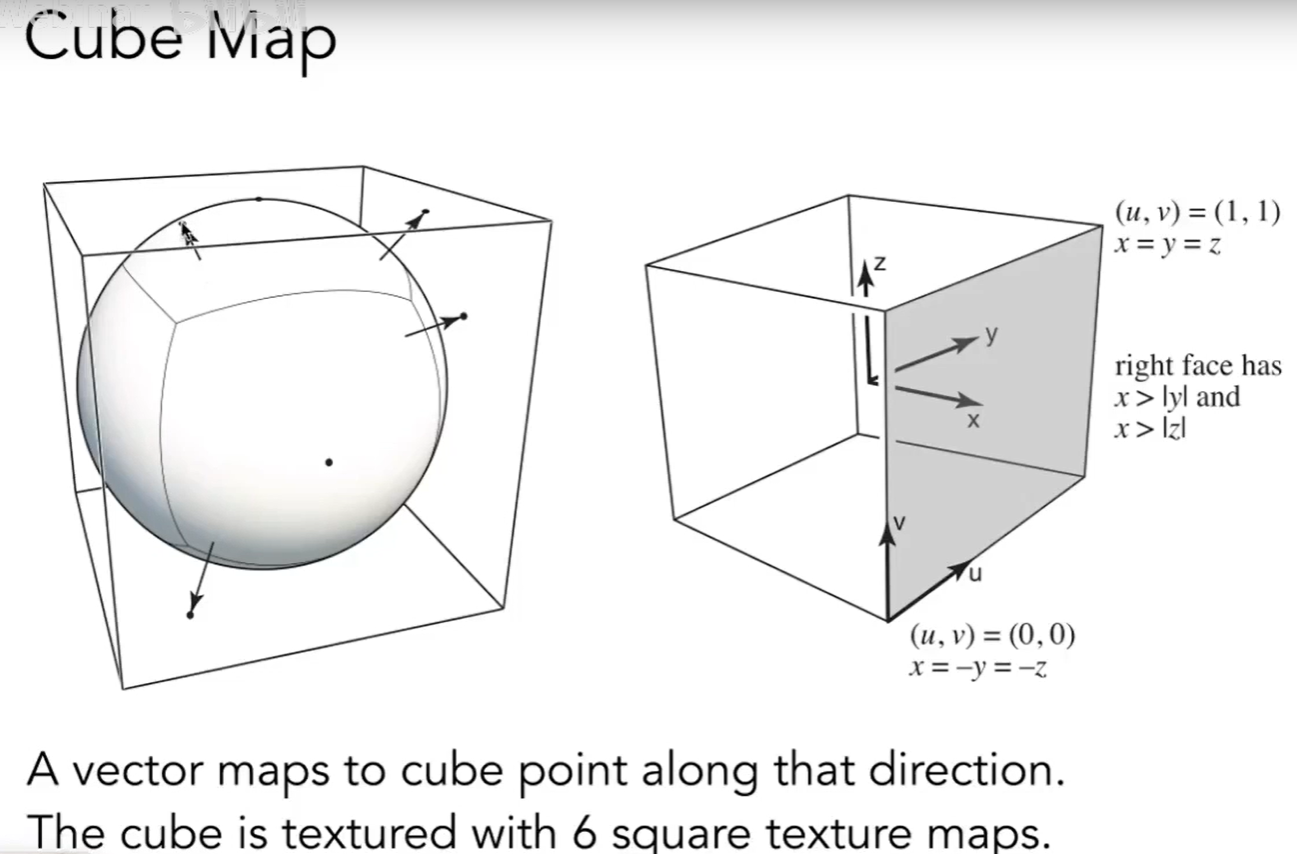
环境光的反射（假设光源都在无限远处）



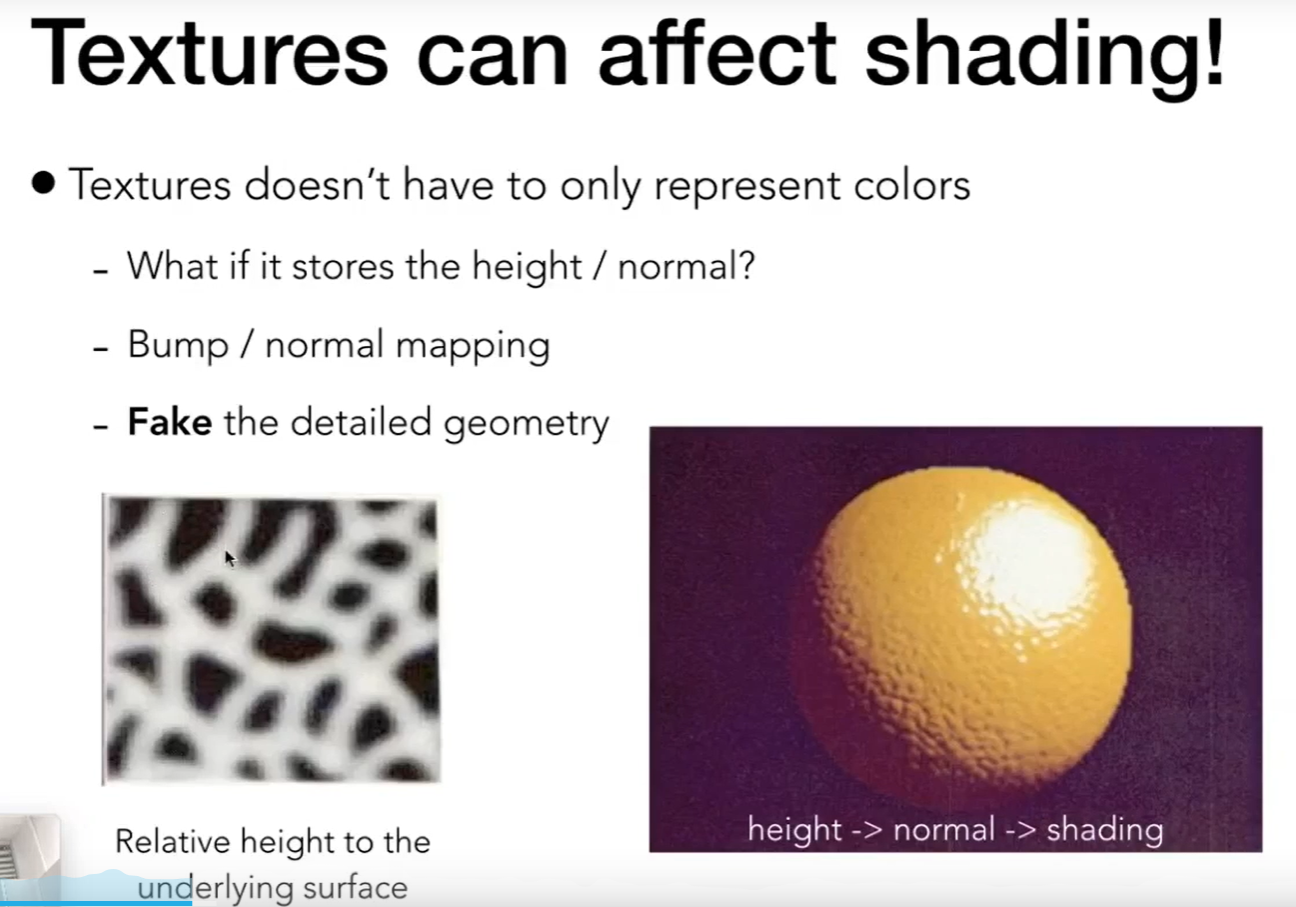
对于这样一个球形物体，环境光可以直接记录在球面上，并且展开（spherical map）

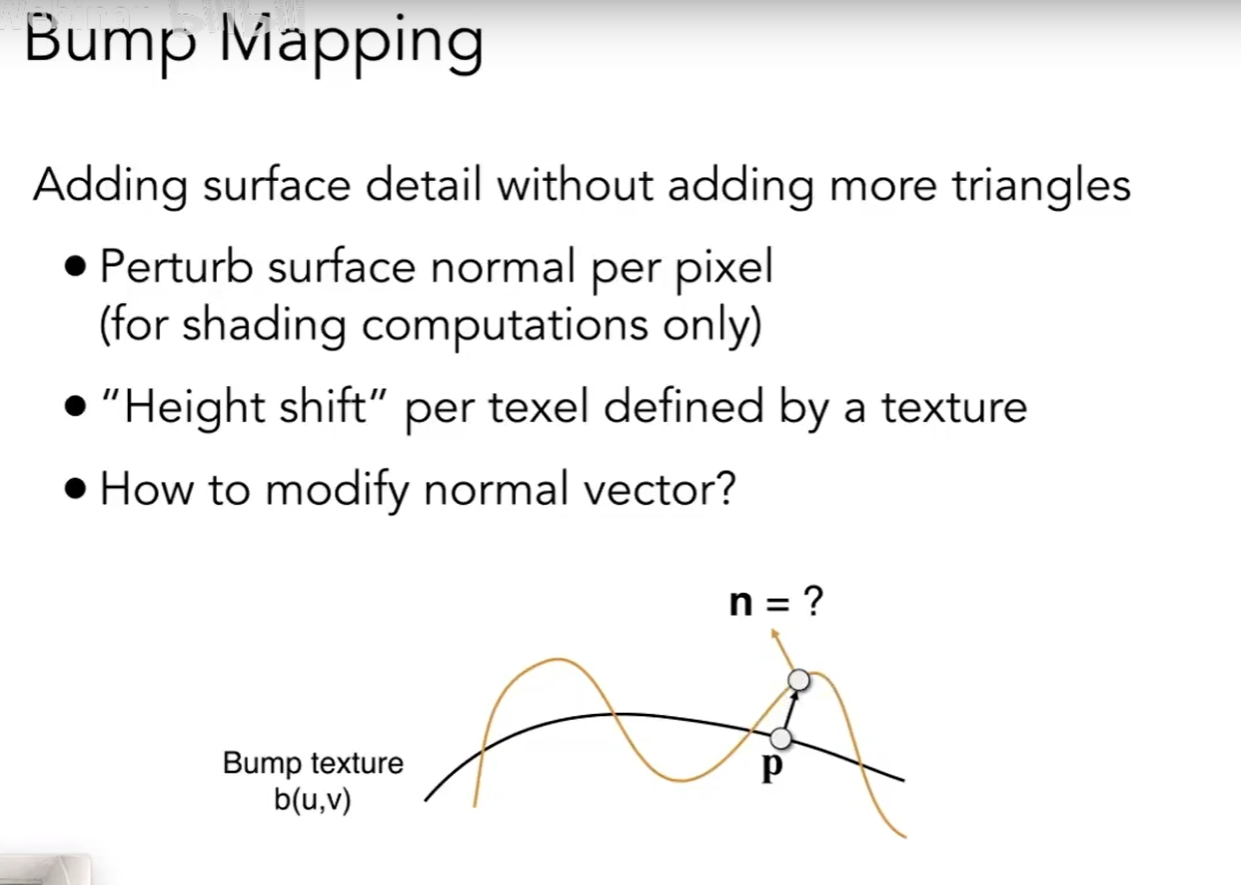
但是这样会在顶部有扭曲问题（就像在世界地图，纬度高的地方相对较小）

解决方案（cube map）：假设这个球外面有一个包围盒，然后把光照信息投射到球面上。扭曲更小。



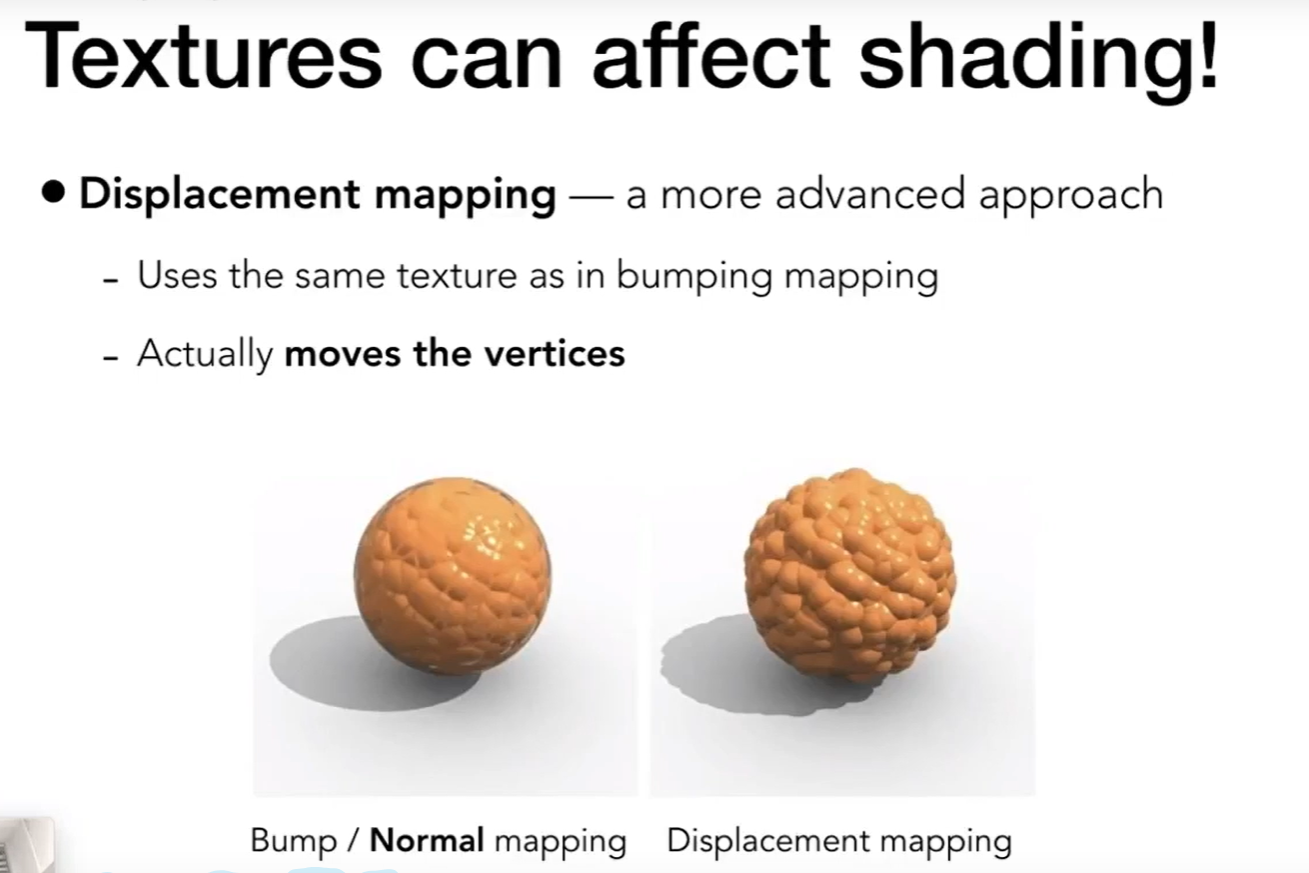
纹理还可以记录纹理上任意一点的相对高度，相对高度的变化，法线就会发生变化，从而着色发生变化，产生明暗对比，从而呈现出凹凸不平的视觉效果。而实际上并没有改变几何图形的复杂度。





如何去计算法线如何变化？

1. 假设任何一个点原本是平面，发现是向上的，法线是(0, 1)
2. 通过求导，找到这个点的切线
3. 切线逆时针旋转90°得到法线（x, y对换，加上符号，再归一化）

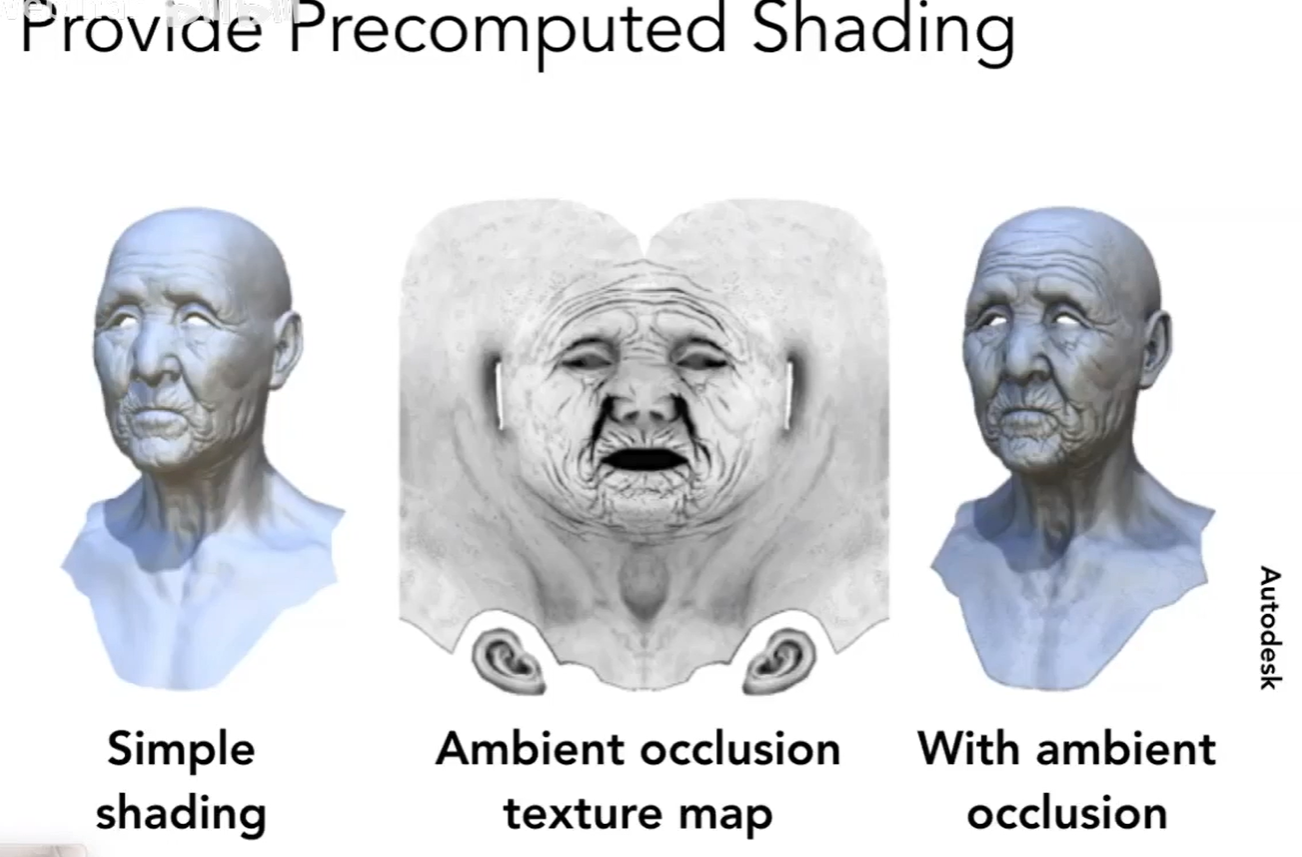


凹凸贴图：实际上没有改变几何形状，可以看到边缘仍然是圆形，投影也是圆形

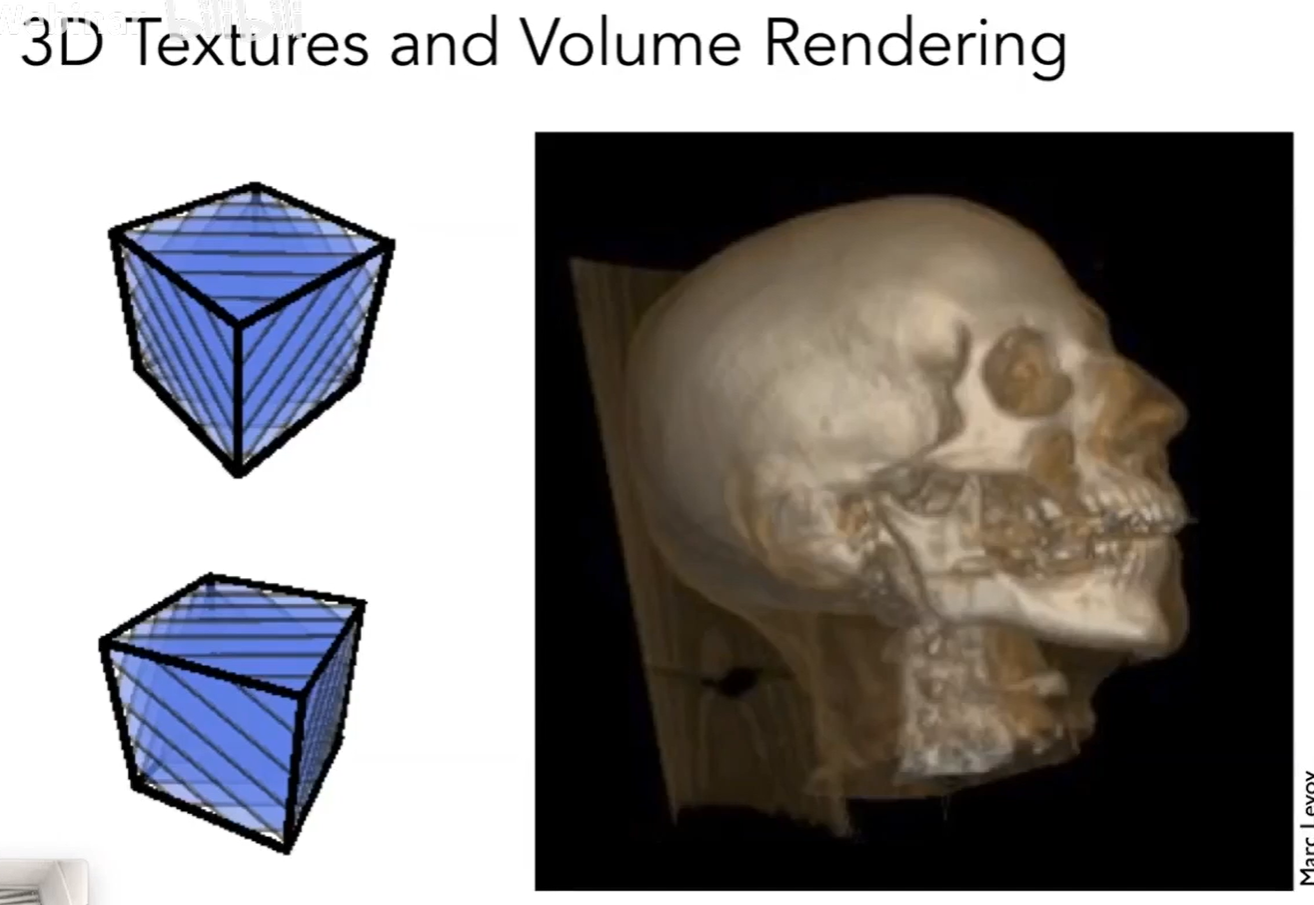
位移贴图：真的移动了顶点的位置。边缘和投影都变化了。结果更好，但是代价是要求这个模型的三角形要足够精细，能够比纹理定义的频率还要高。

纹理还可以记录之前算好的信息

右图：眉毛部分可以投影一部分阴影，算shading的时候比较复杂，但是这部分可以计算好了之后写进纹理贴图，将着色结果（左图）乘以环境光遮蔽贴图得到右图的结果。

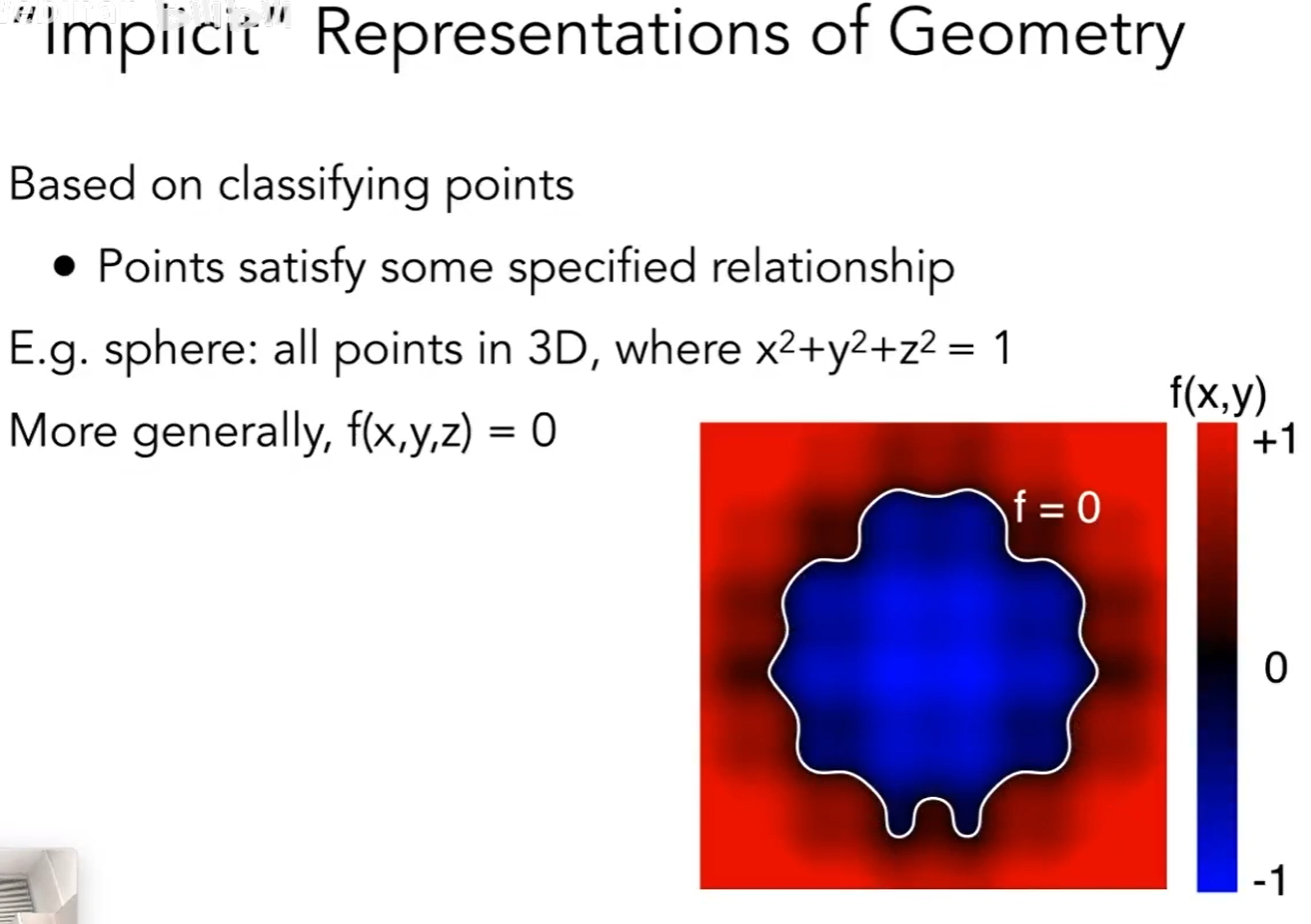


体渲染：三维空间的信息，任何一个点的密度，记录下来做一个渲染（体渲染）

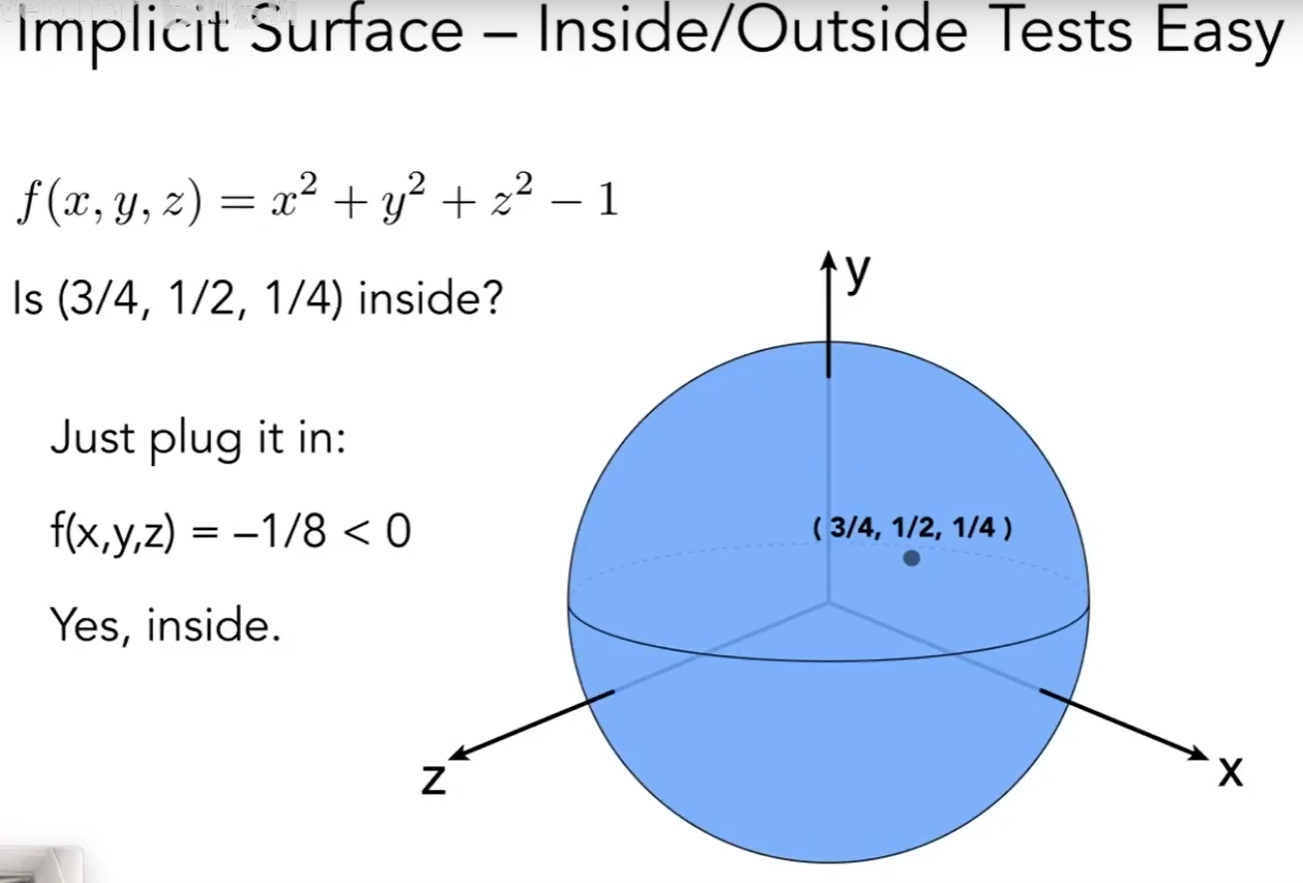


表示不同几何的不同方式

**隐式**：不会具体告诉你点在哪儿，而是告诉你这些点满足什么位置关系。例如单位球可以写成一个方程式表示。

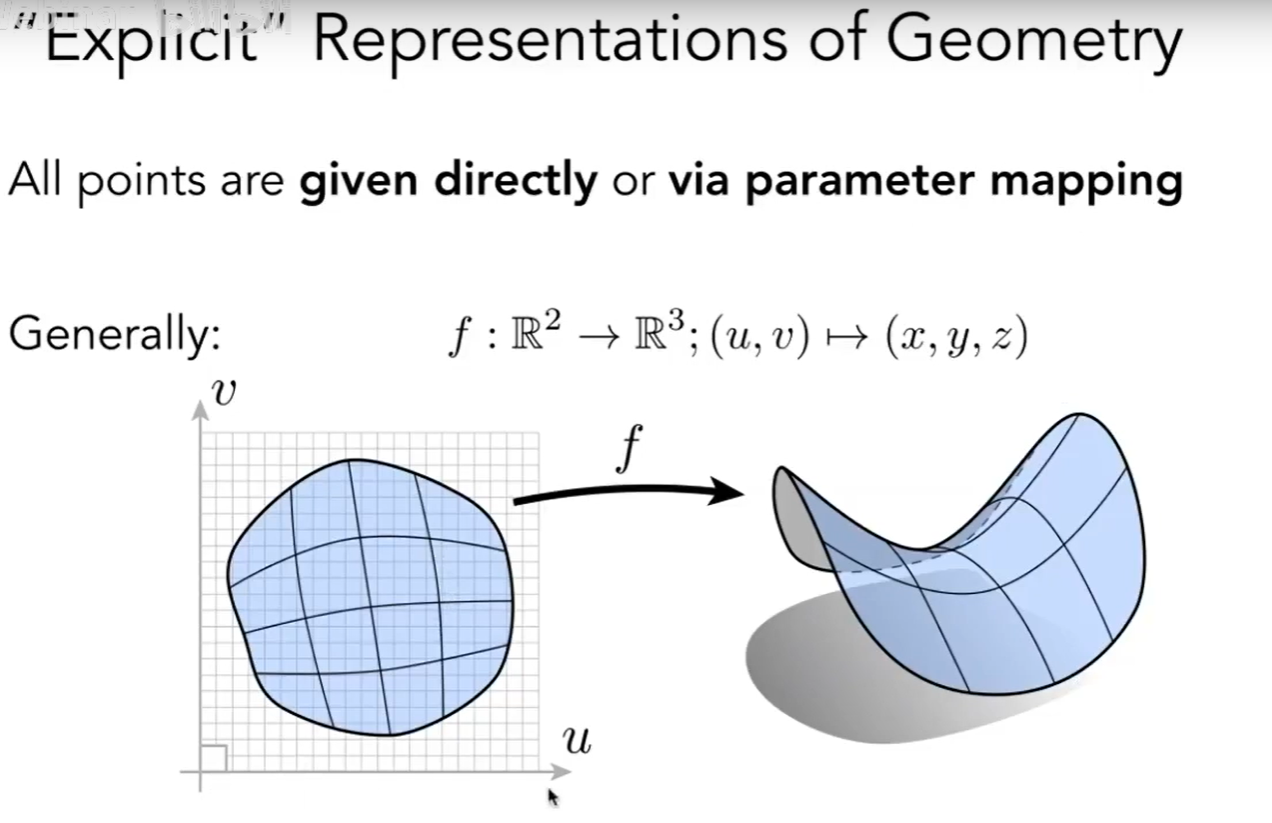


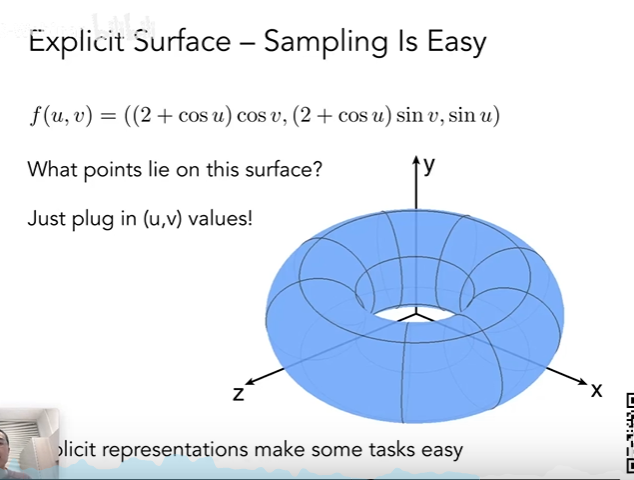
好处，很容易判断一个点在不在这个物体内/外



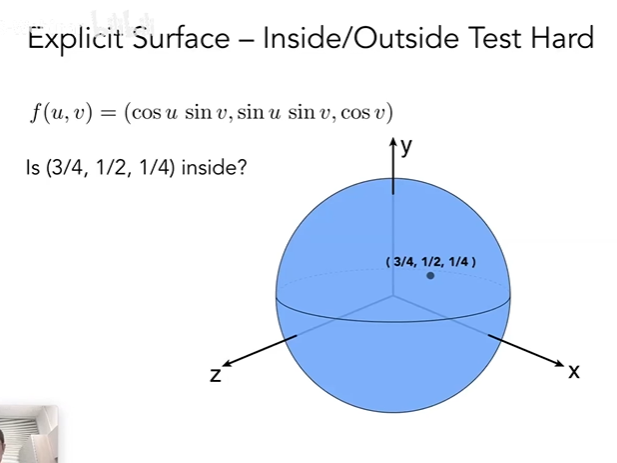
**显式**：一种是**直接给出**所有的点；另一种是通过**参数映射**的方法

参数映射：定义一个函数，从二维点映射到三维的点。

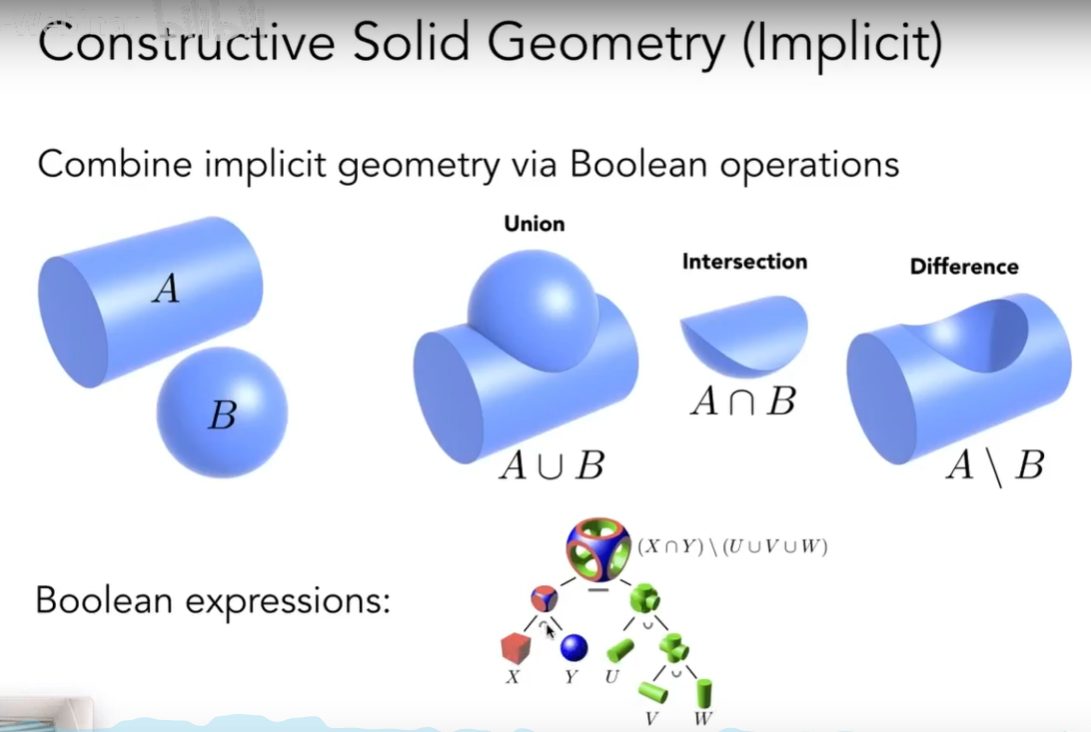




对显式而言，判断一个点在物体内部还是外部比较难

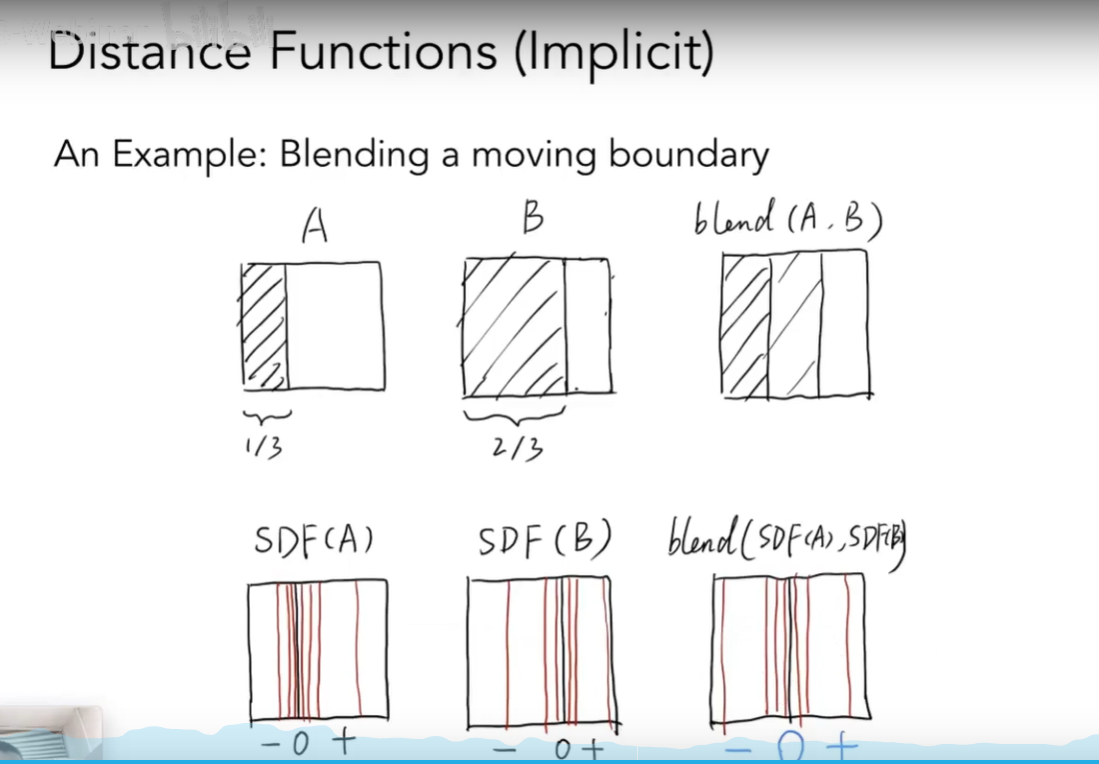


**Constructive Solid Geometry**: 通过一系列基本的几何图形的运算组合成复杂的几何图形

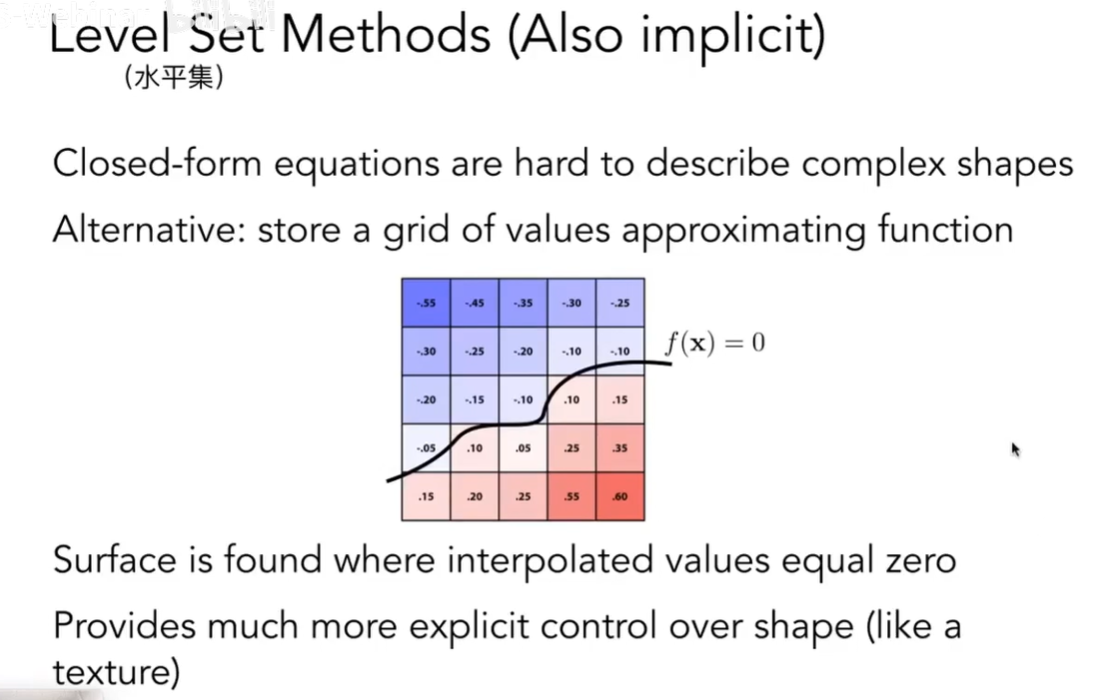


**距离函数**表示：对任何一个点，都不直接描述它的表面，而是描述空间中的任意一个点，到你想要表示的那个点，他们之间的最小距离。这个距离是可正可负的。

这一部分没有太听懂：分别计算A,B的距离函数，黑色表示被遮盖的部分，白色表示露出部分。（下方几张图）黑白交接距离函数是0，越向左边（黑色）去越小，越向右边（白色）去越大。如果将A, B的距离函数合起来就得到了他们合起来的边界是在中间的



**水平集**：如何将blend到的结果恢复成表面？找出所有f(x)=0的位置，得到表面。函数的表示是写在格子上的



如果我们有一个三维的纹理表示人体内部的密度，可以通过找密度函数等于特定值的所有点，这样就能找到一个表面

