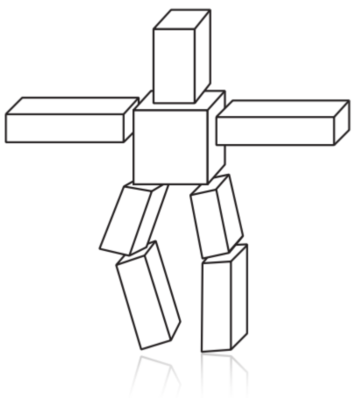
讲义六 - 几何学介绍

Computer Graphics: Rendering, Geometry, and Image Manipulation
Stanford CS248A, Winter 2023

增加模型的复杂性

变换

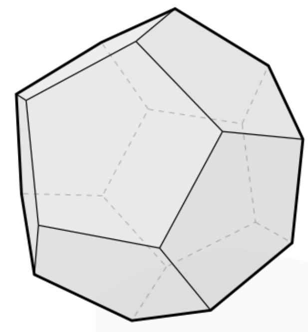


几何体

材料、光照等

什么是几何学?

1. 形状、尺寸、图案和位置的研究
2. 可测量某种数量(长度、角度等)的空间研究

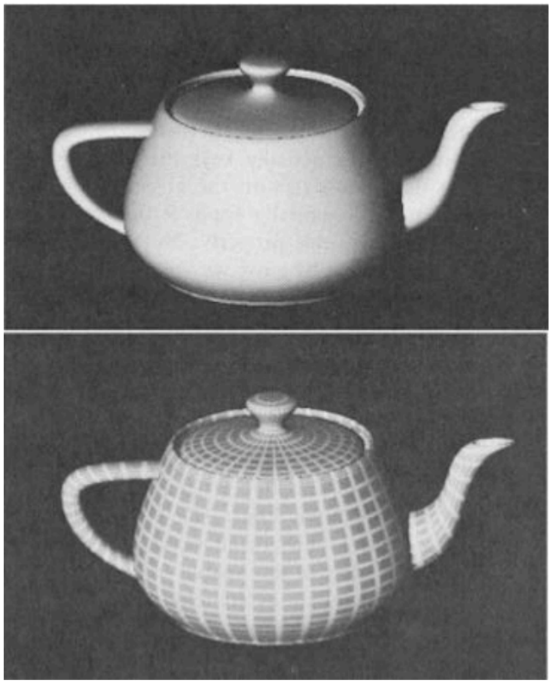
  

柏拉图：**...**从外形上来说地球就像一个那种由**12**块毛皮覆盖于其上的球体。

几何例子

一

原始犹他茶壶(**Utah Teapot**)照片
(现在放在山景城-**Mount View**-
计算机历史博物馆展览)



Matin Newell的早期茶壶渲染图
(**Martin**借助贝塞尔曲线在**1975**年生成了
茶壶模型渲染图)

二

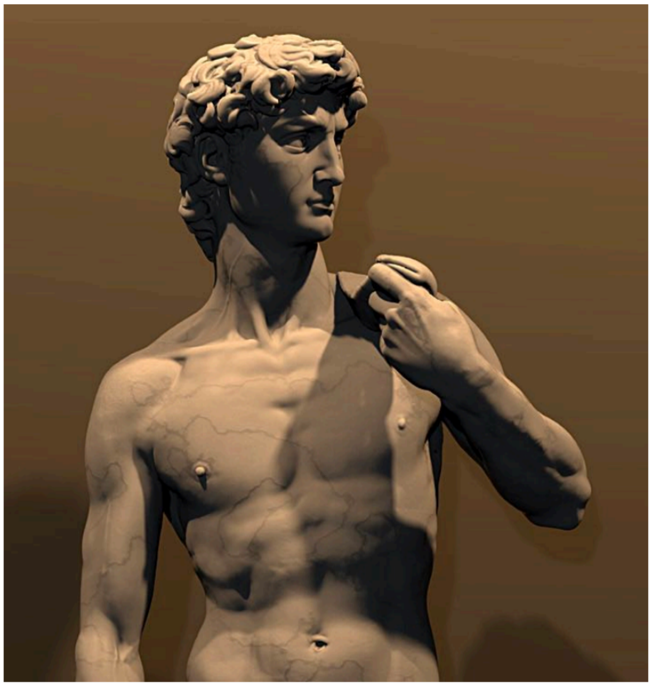
康奈尔盒子(Cornell Box)：初始版本生成于1984
(本图于1985年由Cohen和Greenberg渲染而成)

三

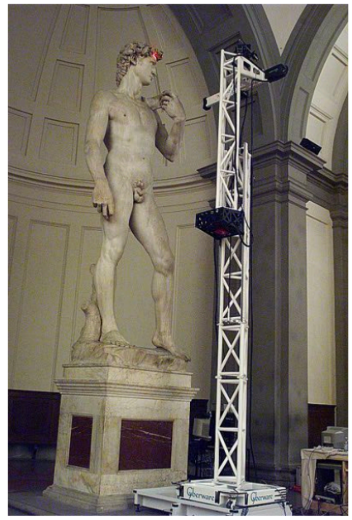
斯坦福**Bunny**兔
(**Mesh**借助激光扫描重建生成)

扫描的塑像图片
(塑像由**Greg Turk**在**Ave**大学的
一个商店中购买)

四

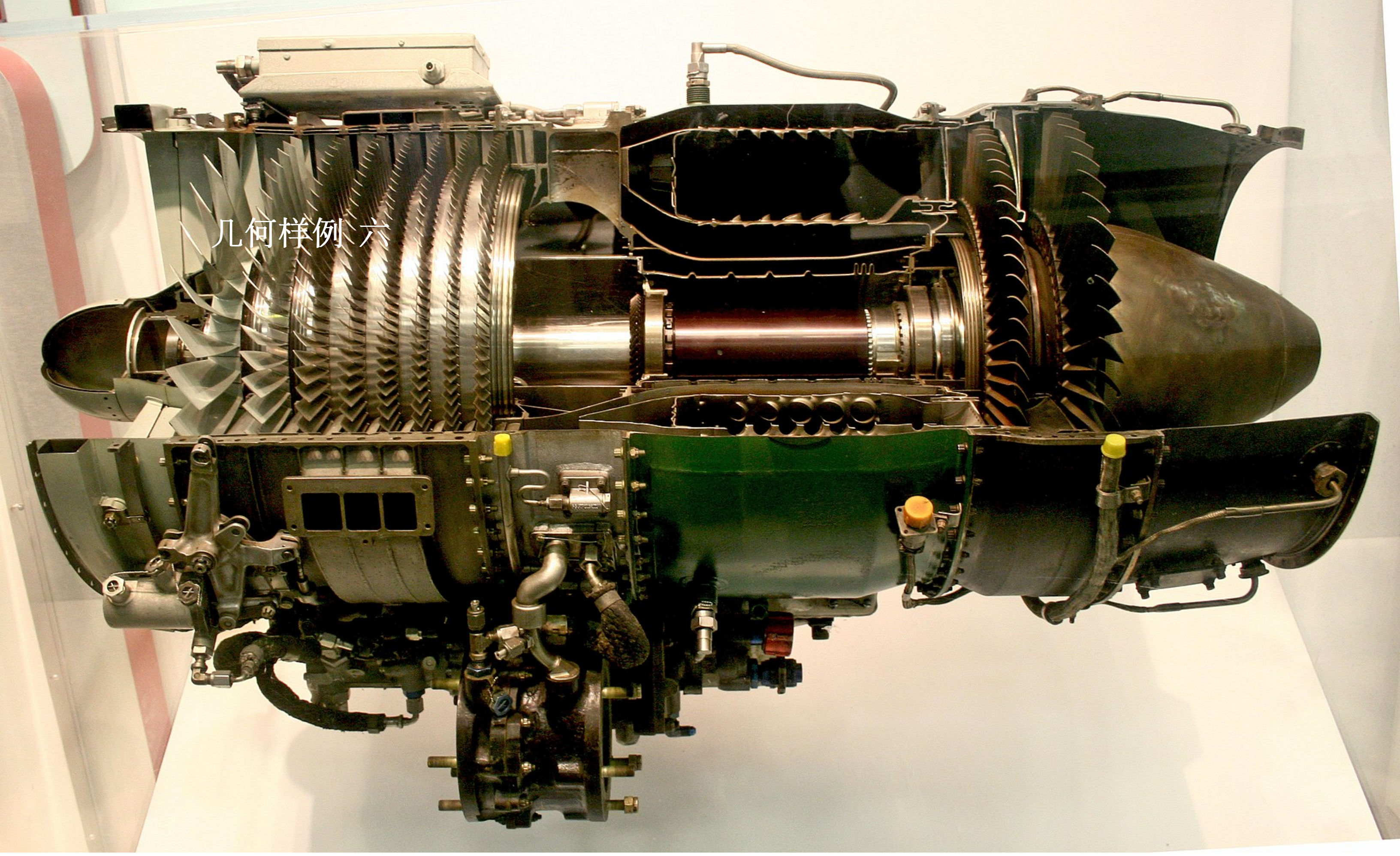


米开朗基罗的大卫雕像激光扫描
(斯坦福电子米开朗基罗项目,1999)



五

皮克斯(**Pixar**)的动画电影"**Brave**"中的卷发


几何样例 六

七

八

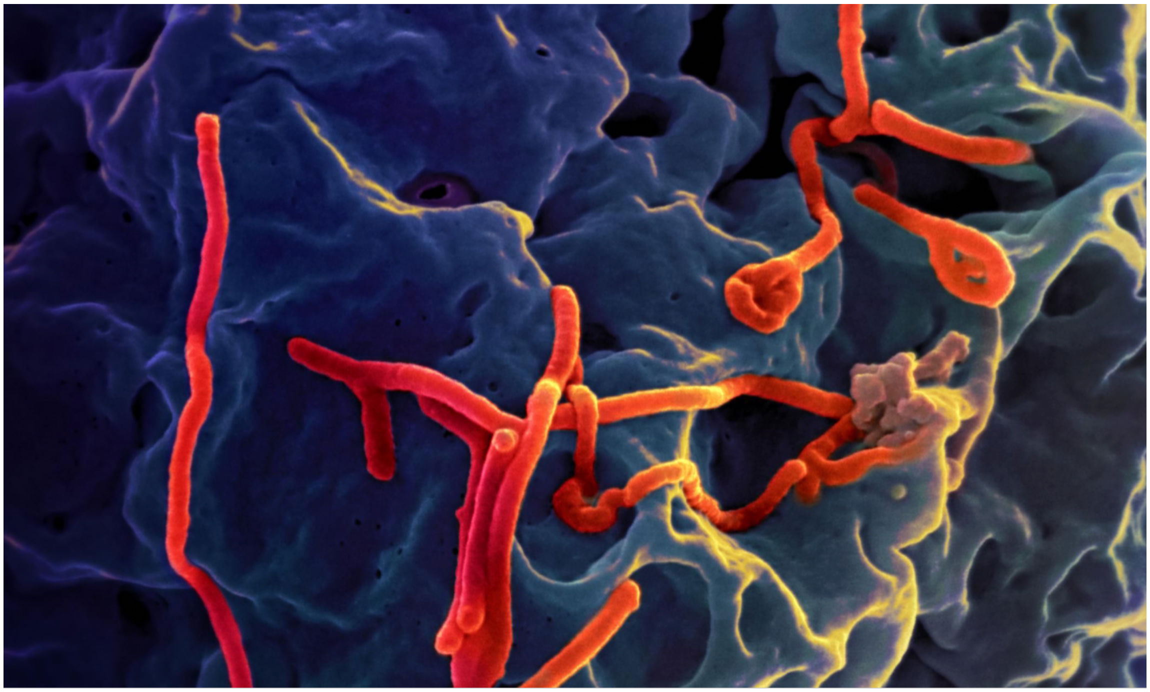
九

十

几何样例 十一

十二

十三



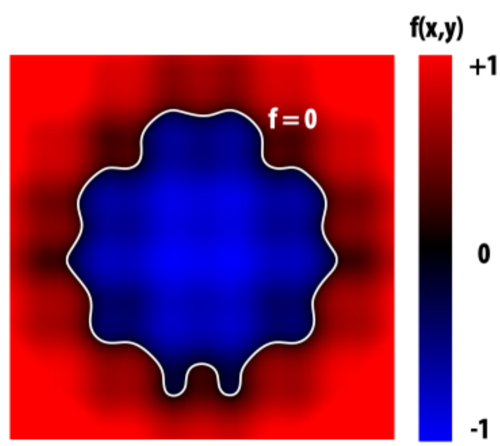
在计算机上编码几何体的最佳方式是什么?

以电子方式编码几何体的几种方式

- 显式(**EXPLICITY**)
  - 点云(**Point Cloud**)
  - 多边形几何网格(**Polygon Mesh**)
  - 曲面细分(**Subdivision**),**NURBS**
  - ...
- 隐式(**IMPLICITY**)
  - 水平集方法(**Level Set**)
  - 代数表面(**algebraic surface**)
  - **L-systems**
  - ...
- 每种选择最好适合于不同的几何任务/几何体类型

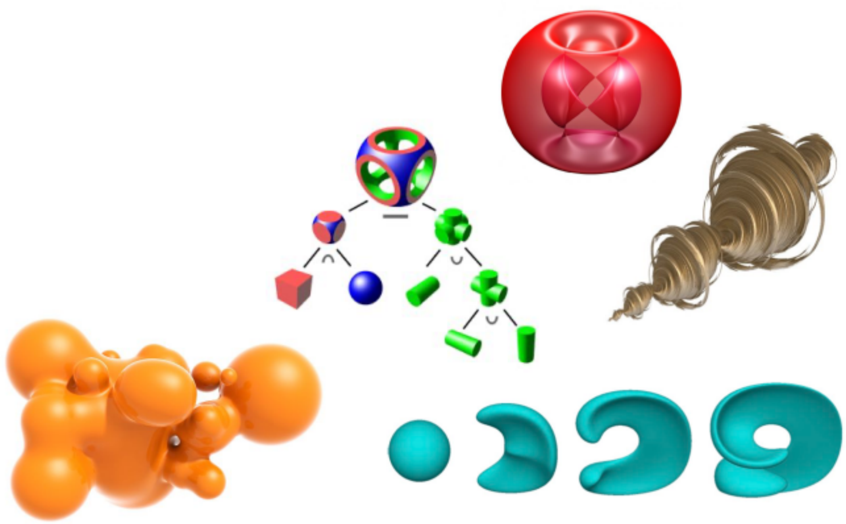
几何体的隐式表达方式

- 所以点(**Points**)不被直接知晓，但是满足某种关系
- 比如，单位球体就是由那种满足$x^2 + y^2 + z^2 = 1$的点构成
- 更通用的表达，$f(x, y, z) = 0$



图形学中的多种隐式表达

- 代数表面(algebraic surfaces)
- 可构造实体几何体(constructive solid geometry)
- 水平集方法(level set methods)
- 斑点状表面(blobby surfaces)
- 分形几何体(fractals)
- ...



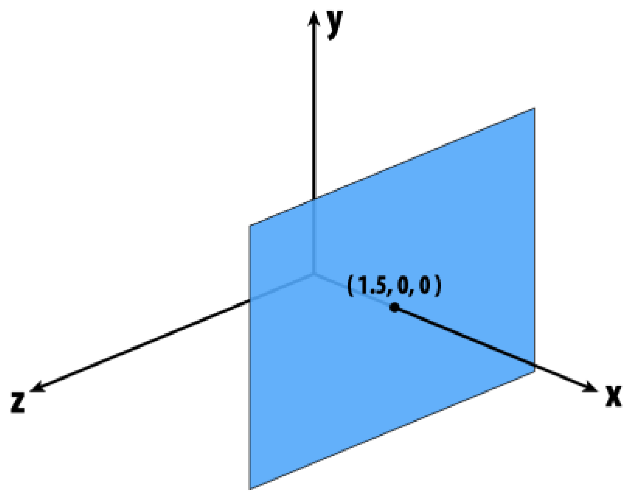
首先，让我们玩局游戏:

我正在思考隐形表面$f(x, y, z) = 0$

找出其上的任意一个点

要不要放弃?

我的函数是$f(x, y, z) = x - 1.5$(一个平面):



隐式表面使得一些任务难于处理（比如采样）

让我们再玩一局游戏:

我有一个新表面$f(x,y,z)=x^2+y^2+z^2-1$

我想看一下是否有一个点在其内

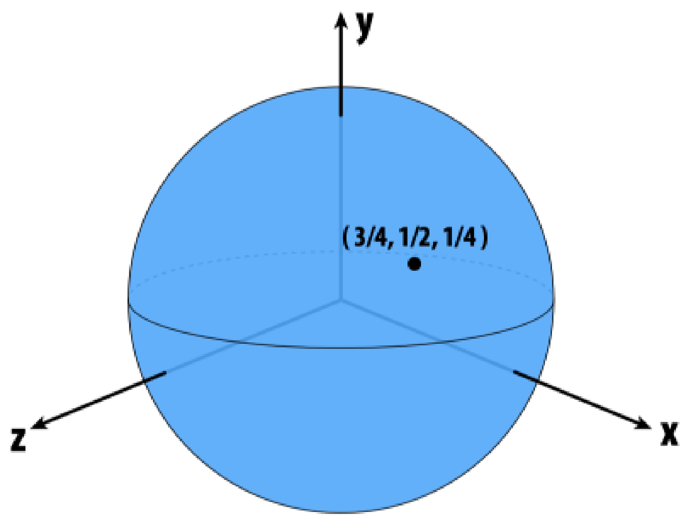
检查点是否在单位球体内?

点**(3/4,1/2,1/4)**如何?

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

$7/8<1$

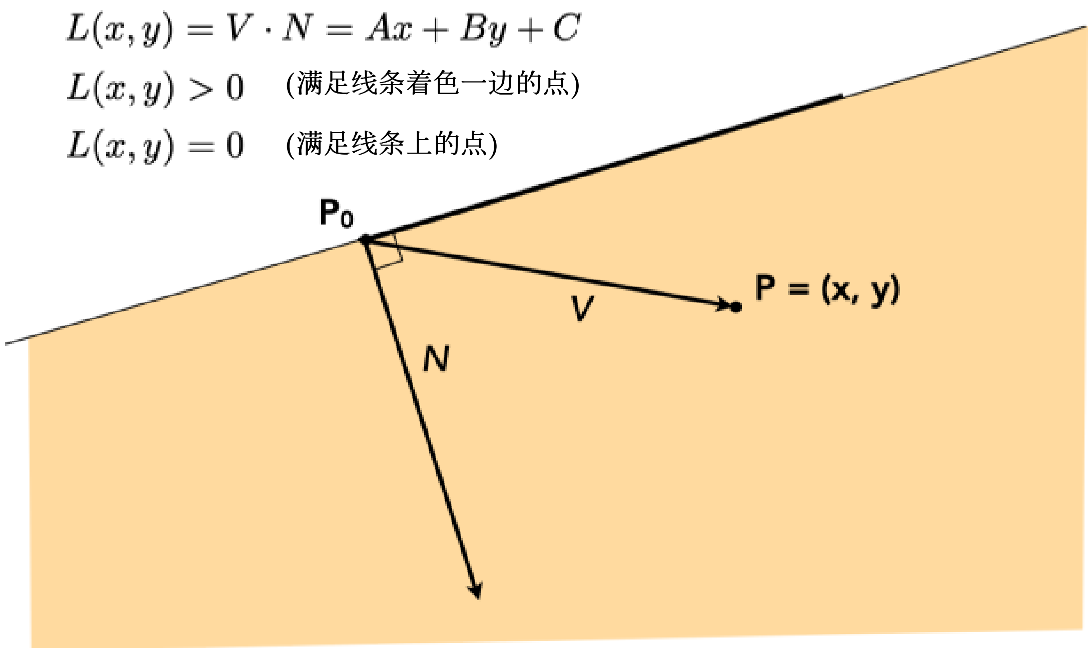
是！在球体内



隐式表面使得另一些任务变得容易（比如几何体内/外检测）

会议：线条Line的隐式表达

很容易验证点位于线条“正的”或是“负的”一边。



隐式表面使得一些任务难于处理（比如采样）

几何体的“显式”表达

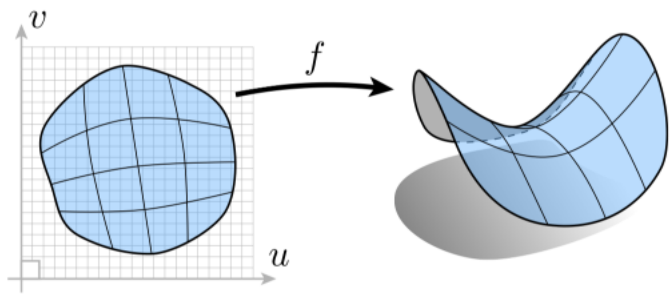
一

■ 所有的点被直接给出

■ 在球体上的点是

$$(\cos(u)\sin(v), \sin(u)\sin(v), \cos(v)),$$
$$\text{for } 0 \le u < 2\pi \text{ and } 0 \le v \le \pi$$

■ 更通用的表达： $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$

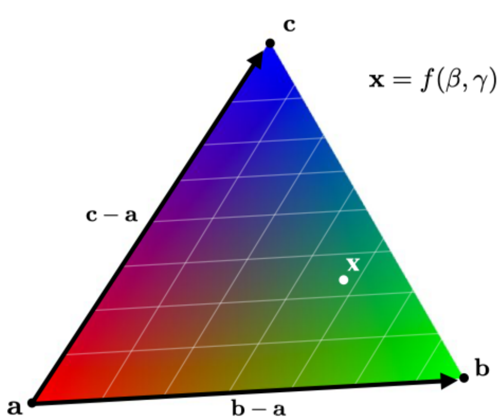


(可能存在很多这种映射，比如，每三角形一个)

二

■ 更通用的表达： $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$

■ 实例：一个三角形



$$\mathbf{x} = f(\beta, \gamma) = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

图形学中有很多这样的显式表达

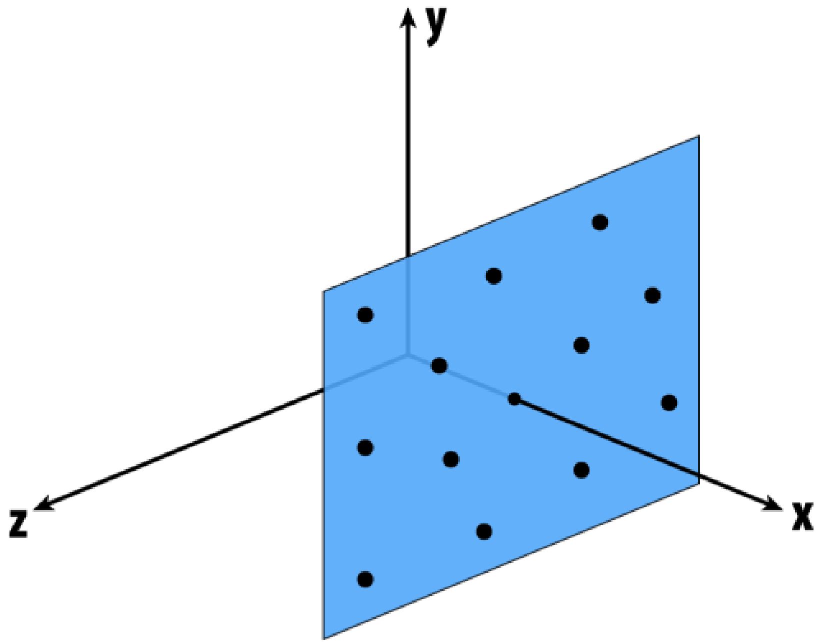
首先，让我们再玩一局游戏：

我会给出一个显式表面

你需要指出在其上的一些点

采样显式表面

表面为$f(u,v)=(1.5,u,v)$
只需将任意值(u,v)代入即可！



显式表面使得一些任务容易完成(比如采样)

我们再玩一个游戏:

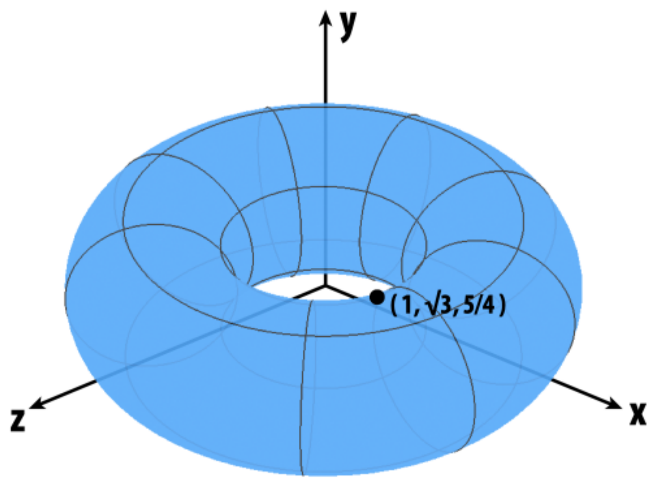
我有新表面$f(u, v)$

我想知道是否一个点位于几何体内

验证一下点是否位于圆环内?

表面为$f(u, v) = (2 + cos(u)cos(v), 2 + cous(u)sin(v), sin(u))$
点$(1, \sqrt[2]{3}, 5/4)$到底在不在圆环内?

. . . 不!!!



显式表面使得另一些任务难于完成(比如几何体内/外检测)

结论:

一些几何表达比另一些更有效——

依赖于要执行的任务

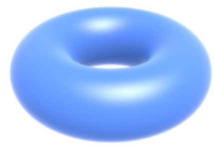
不同形式的几何表达可能会更适用于不同的几何体类型

让我们看一下图形学中常用的一些几何表达

代数表面(隐式表达)

- 表面是以x, y, z为变量("代数学变量")的多项式的零集
- 例子:

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



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

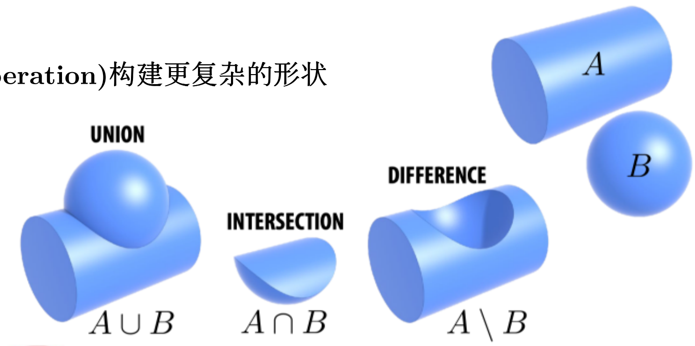
$(x^2+\frac{9y^2}{4}+z^2-1)^3 = x^2z^3+\frac{9y^2z^3}{80}$

- 更加复杂的形状又会如何?

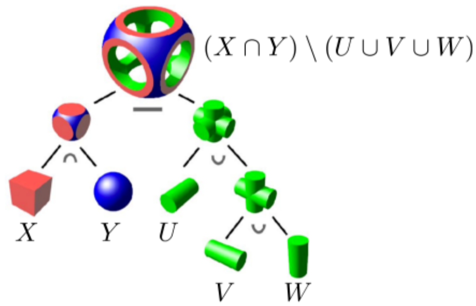
- 很难列出复杂形状的多项式?

可构建实体几何体(隐式表达)

- 借助布尔操作(**Boolean Operation**)构建更复杂的形状
- 基础操作:

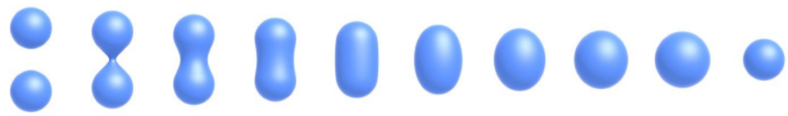


- 然后在基础操作之上构建更复杂的形状

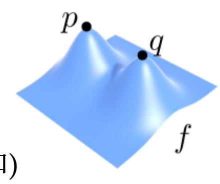


斑点状表面(隐式表达)

■ 不同于布尔操作(**Boolean Operation**),逐渐将表面混合在一起:

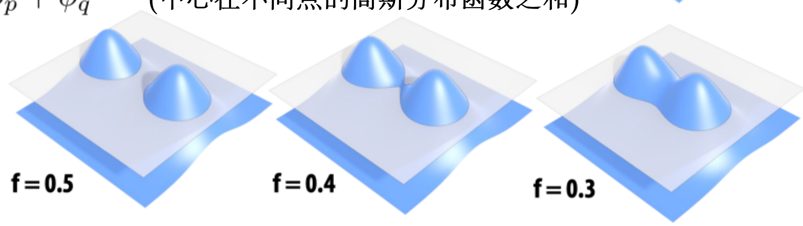


■ 在**2D**中更容易理解:



$\phi_p(x) := e^{-|x-p|^2}$(中心在$P$的高斯分布函数)

$f := \phi_p + \phi_q$ (中心在不同点的高斯分布函数之和)

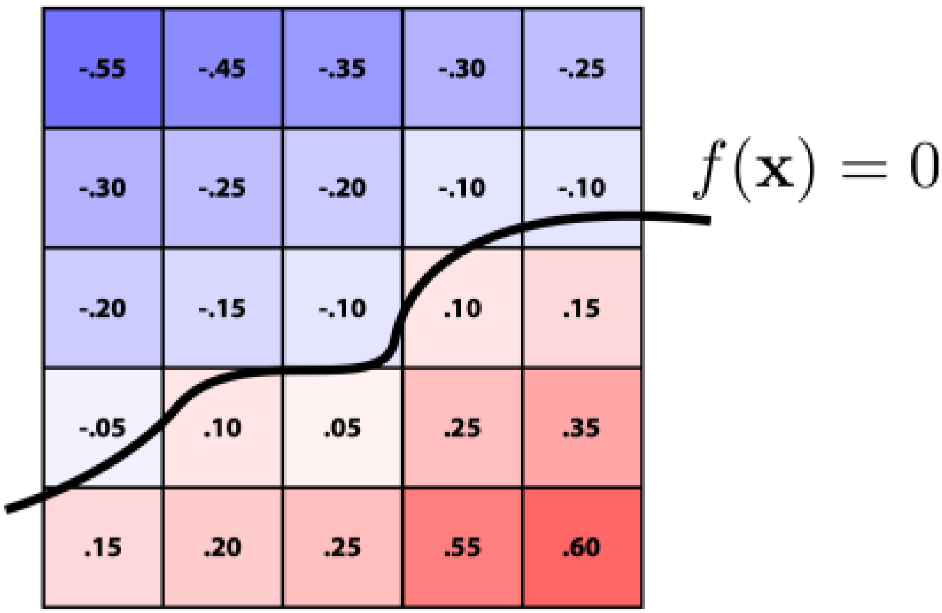


水平集方法-Level Set Methods(隐式表达)

■ 隐式表达具备一些良好的特征(比如，合并/划分)

■ 但是难于描述相近形式的复杂形状

■ 代替方法：存储近似函数的网格值



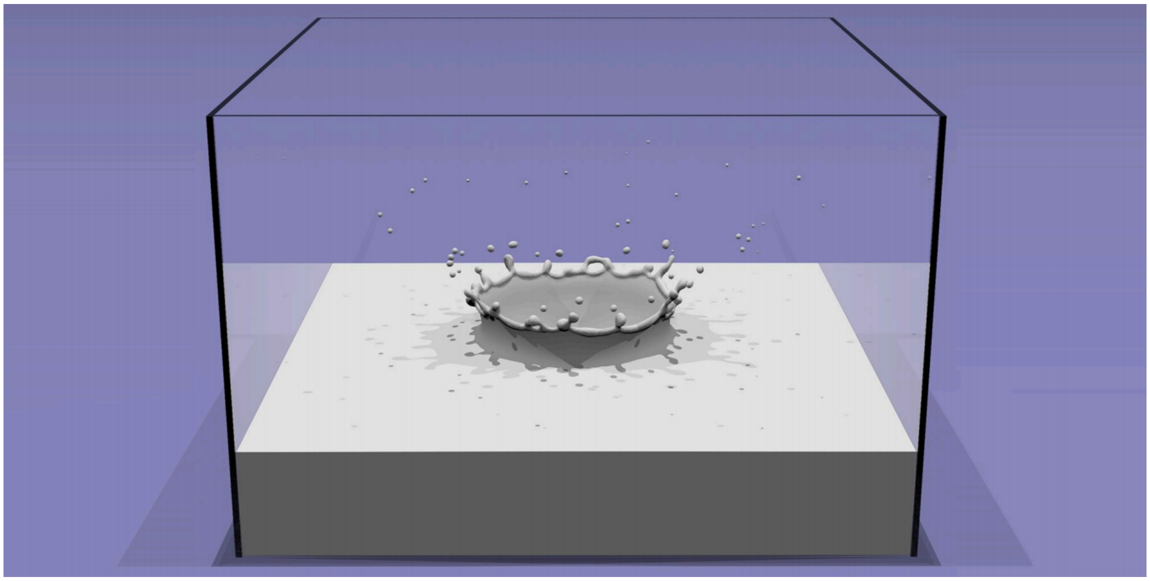
- 在插值后值为0的地方表面被发现
- 在形状上可以提供显式得多的控制(就像纹理)
- 总是需要复杂的过滤操作(trilinear,tricubic...)

应用于医学数据领域的水平集方法(CT,MRI,等等)

- 水平集编码，比如，恒定的组织密度

物理模拟中的水平集方法

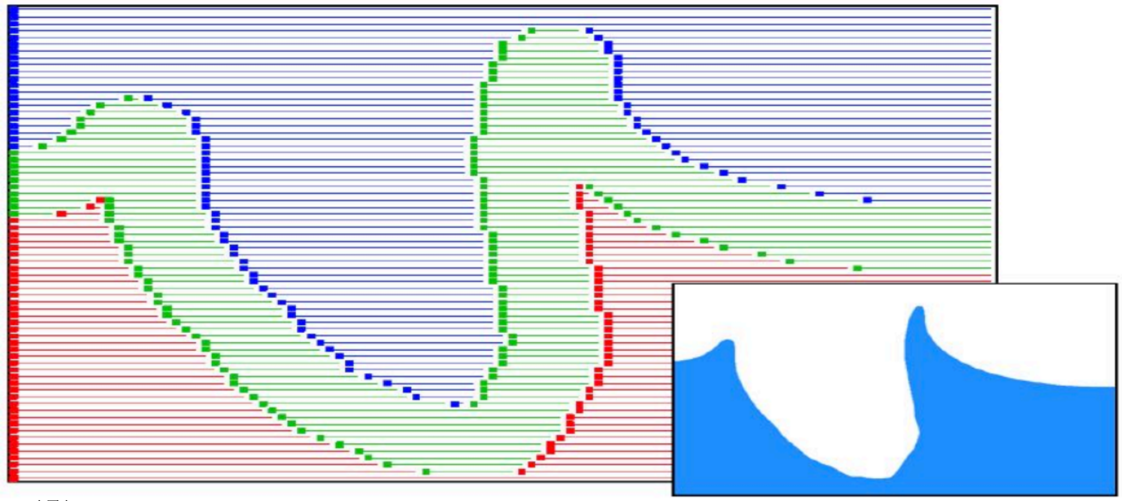
- 水平集编码到气液边界的距离



参考http://physbam.stanford.edu

水平集存储

- 缺点：用于**2D**表面的存储现在的复杂度为$O(n^3)$
- 通过仅存储表面附近的一个窄的距离带，可能减少存储代价



本图中：

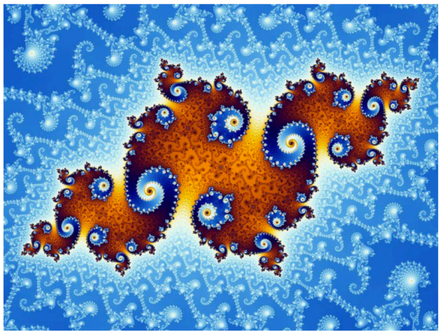
红色＝很明显位于水中

绿色＝存储用于编码表面的水平集值的区域

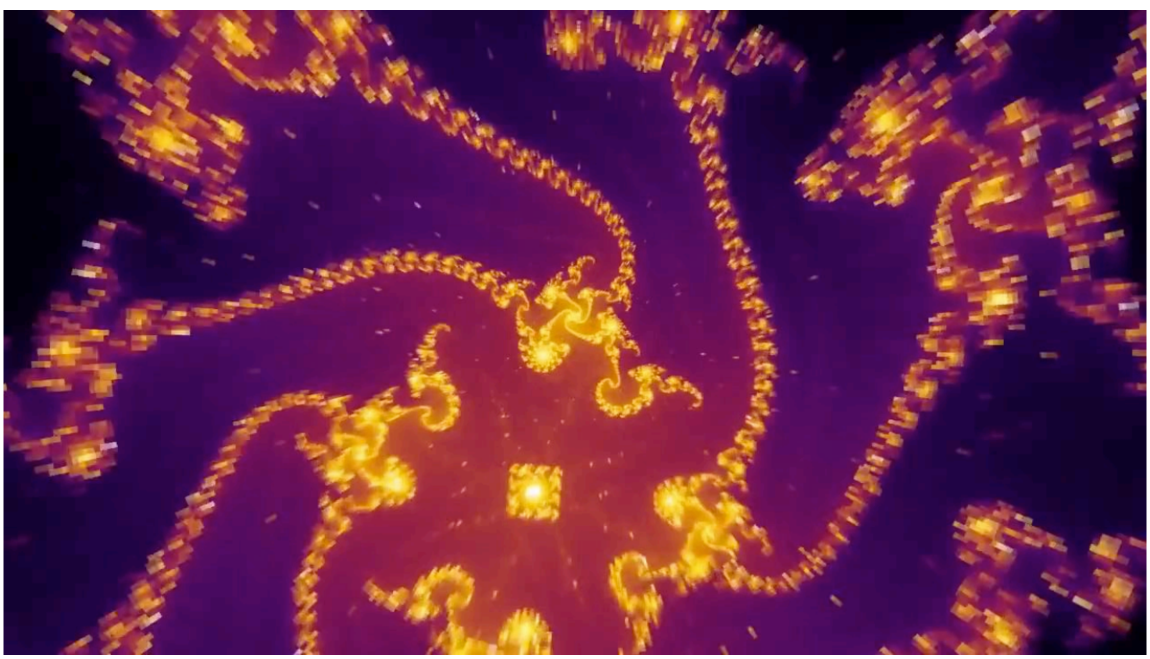
蓝色＝很明显不在水中

分形几何体(隐式)

- 无精确定义，表示出自相似性，在所有尺寸上都可以描述
- 用于描述自然现象的新“语言”
- 难于控制形状!



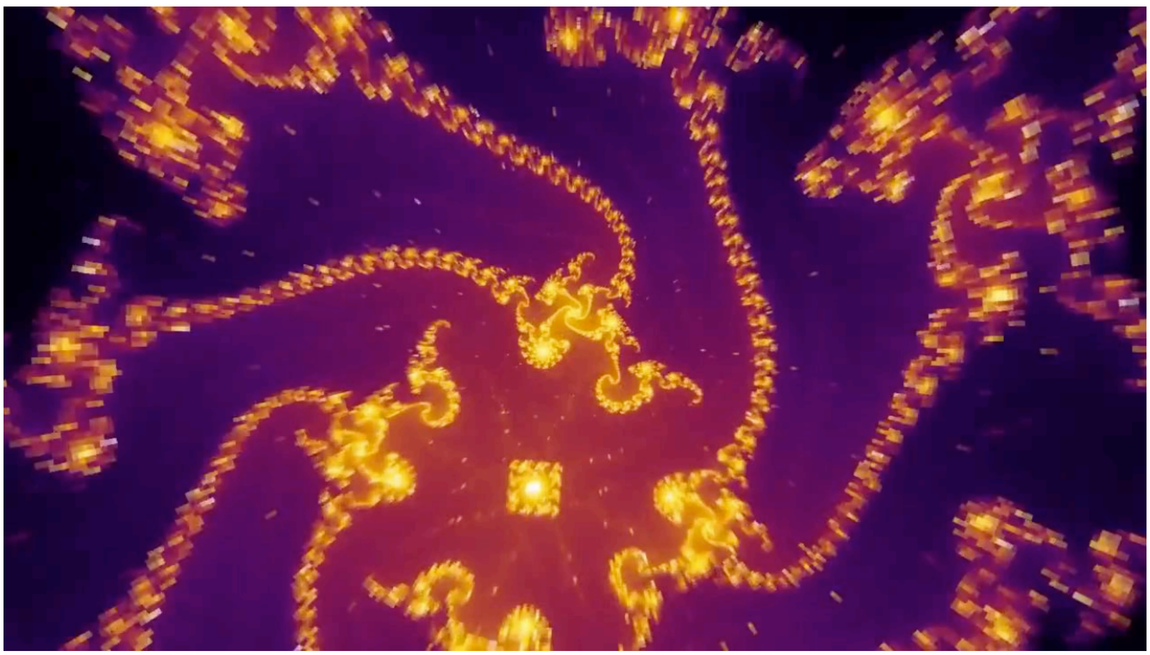
Mandlebrot集–放大



依据每个点分离和聚合得快速程度进行着色

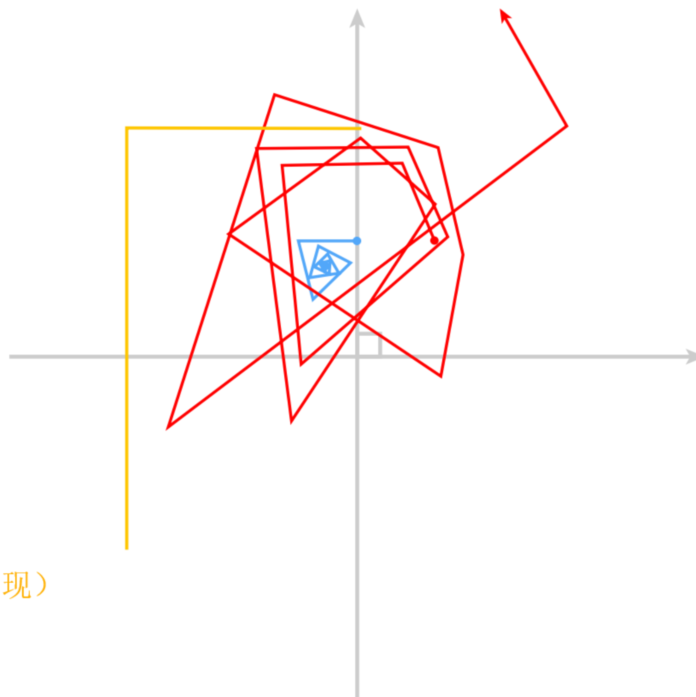
Mandlebrot集–定义

- 针对在平面上得每个点c:
  - 双倍其角度
  - 平方其量级
  - 叠加其初始点c
  - 重复



依据每个点分离和聚合得快速程度进行着色

Mandlebrot集–实例



起始点

(0, 1/2)　（聚合）

(0, 1)　（间歇出现）

(1/3, 1/2)　（分离）

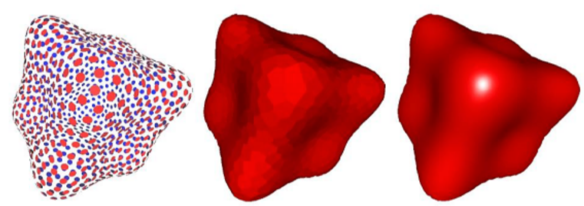
隐式表达–利与弊

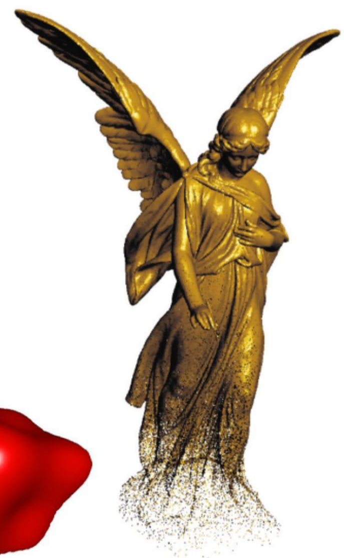
- 好处:
    - 描述可能非常紧凑(比如，多项式)
    - 易于明确一个点是否位于形状内(只需代入！)
    - 其它查询也可能易于执行(比如，到表面的距离)
    - 对于简单形状，存在确切描述/不存在采样错误
    - 易于处理拓扑学(topology)上的改变(比如，流体)
- 弊端:
    - 找出形状中的所有点是代价高昂的(比如，用于绘制时)
    - 非常难于建模复杂形状

显式表达又如何呢?

点云(显式表达)

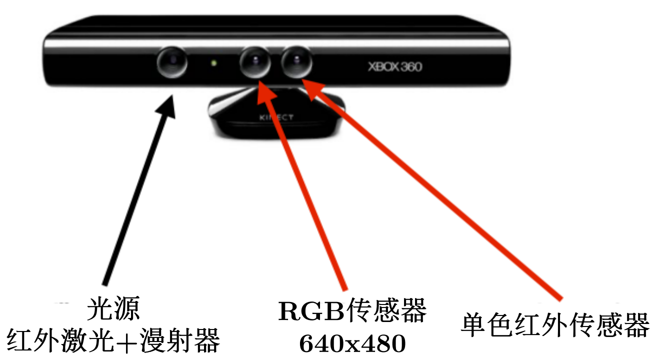
- 最简单的表达：点(x, y, z)清单
- 经常借助发现增强
- 轻松表达任何种类几何体
- 针对特大数据集十分有用(>>1点/像素)
- 对于低(欠)采样区域难以借助插值进行补充
- 难以进行处理/模拟/...





经由激光扫描形成的点云

另一个例子：Microsoft XBox 360 Kinect



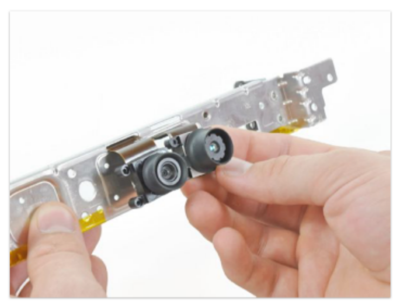
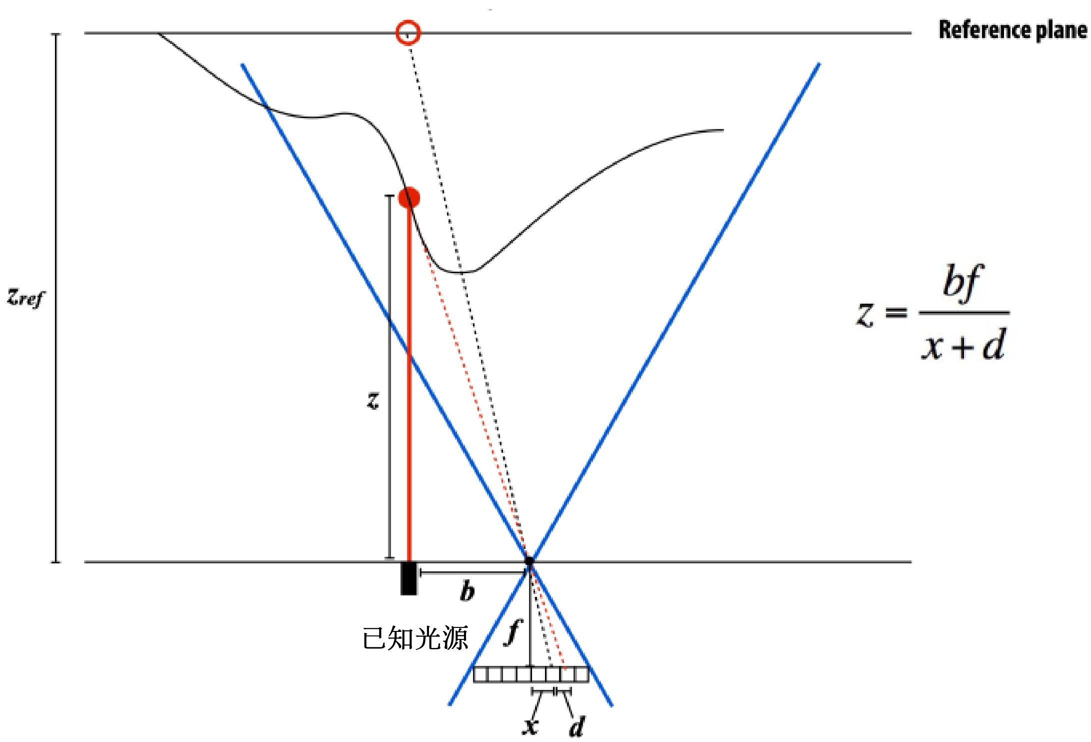


Image credit: iFixIt

结构化光照

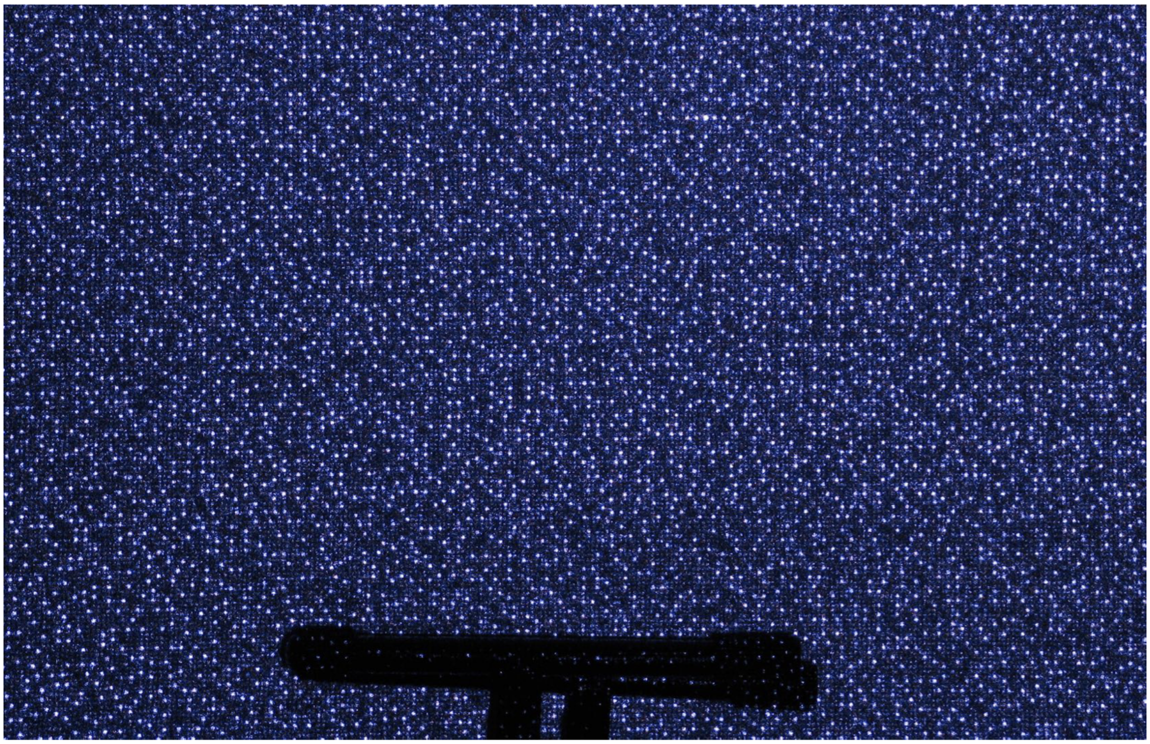
体系：一个光源发射已知光速 ＋ 一个相机测量场景外观
如果场景位于参考平面，相机将记录的图像就是明确的
(所记录图像的像素和场景点之间的对应关系是已知的)



单一点光源照明效率不高！
(必须使用点光源“扫描”场景获取深度，所以获取单一深度图像会有高延迟)

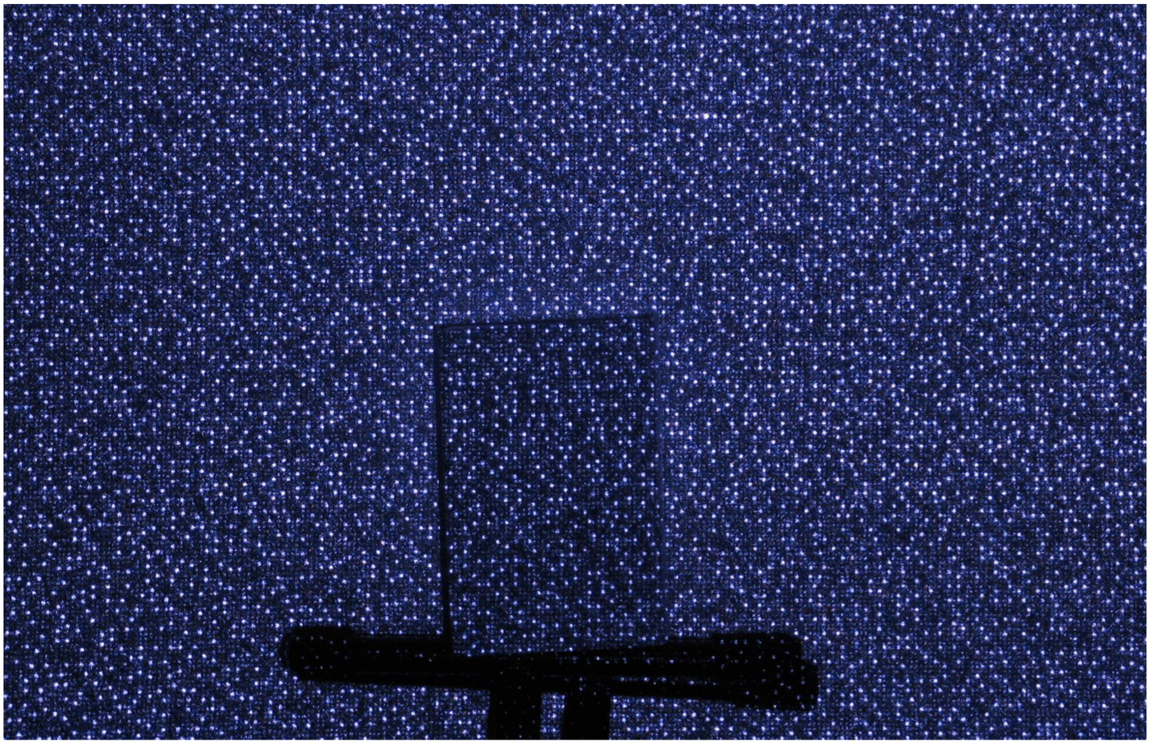
Kinect光源输出的红外图像

一



Credit: www.futurepicture.org

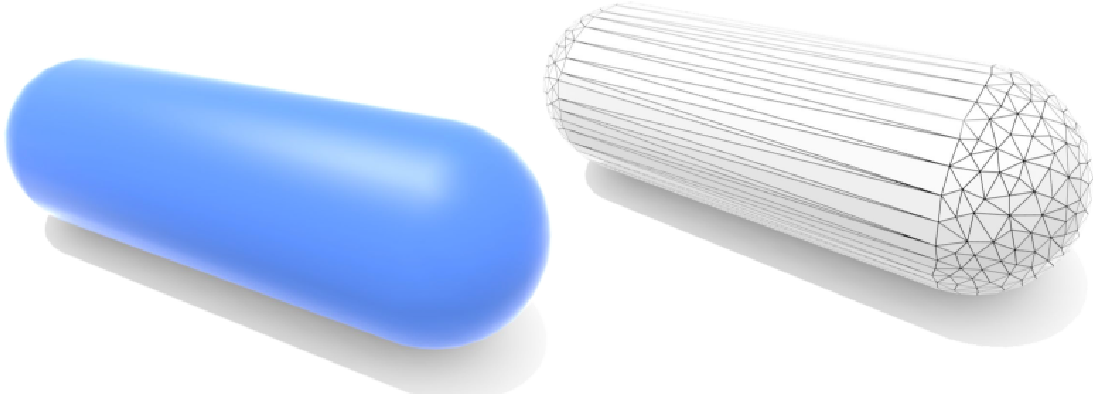
二



Credit: www.futurepicture.org

多边形网格(显式表达)

- 存储顶点和多边形（最常见的是三角形和四边形）
- 更易于处理/模拟，适应性采样（adaptive sampling）
- 更复杂的数据结构
- 可能为图形学中最常用的表达方式



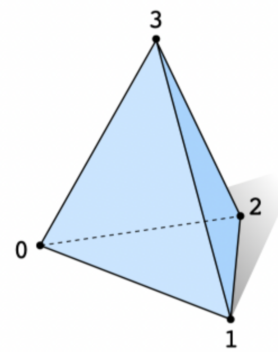
(在未来的课程中会见到更多的关于多边形网格的内容)

三角形网格(显式表达)

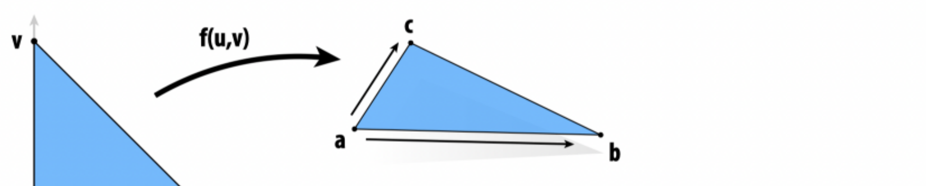
- 将顶点存储为坐标三元组(x,y,z)
- 将三角形存储为索引三元组(i,j,k)
- 例如，四面体：

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

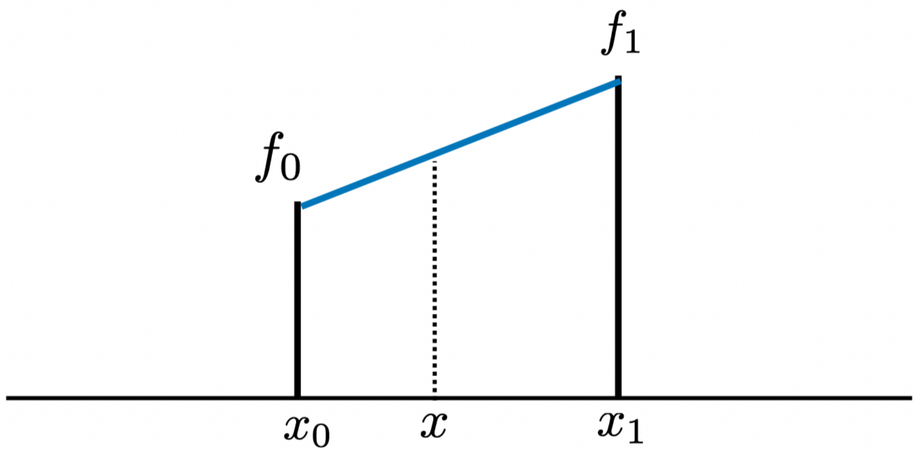


- 借助线性插值定义三角形内部的点



$$f(u,v) = a + u(b-a) + v(c-a)$$

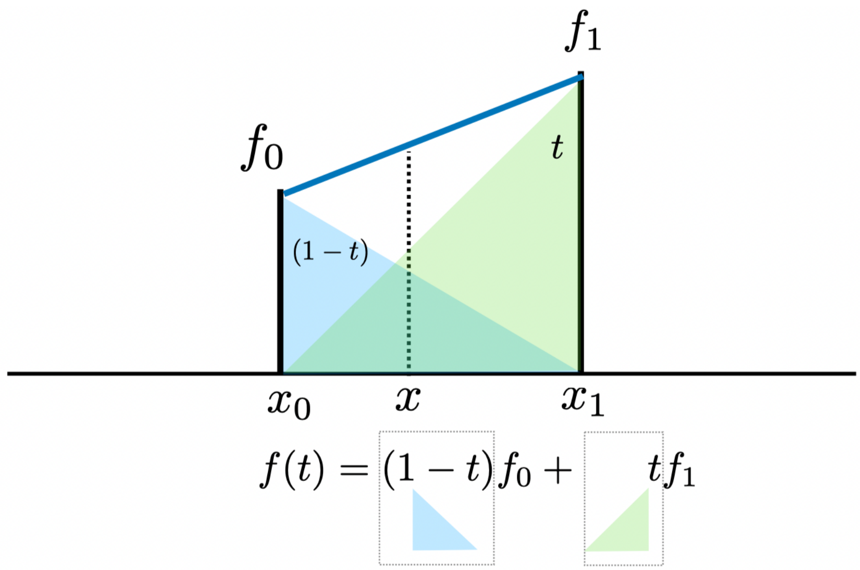
样本(在1D中)的线性插值



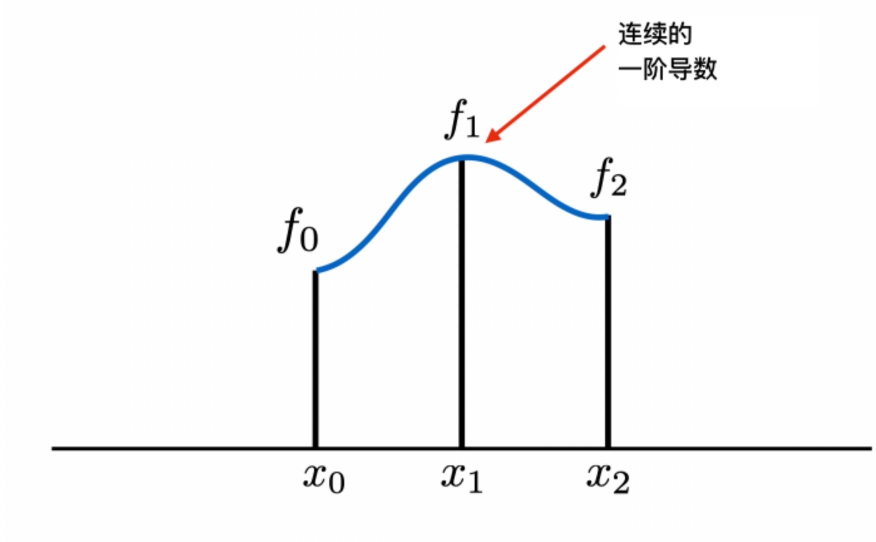
$$f(t) = (1-t)f_0 + tf_1$$

$$t = \frac{x - x_0}{x_1 - x_0}$$

可以将线性插值当作两个函数的线性组合

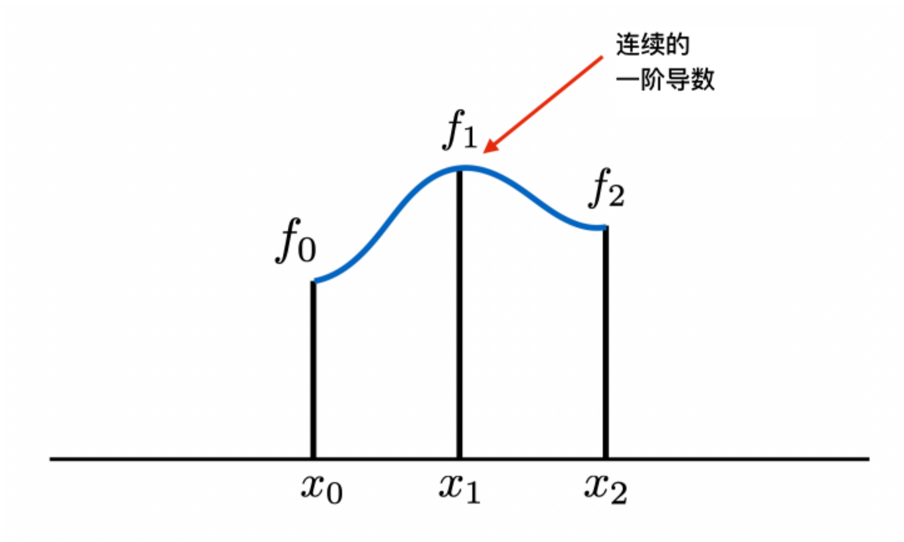


平滑插值?



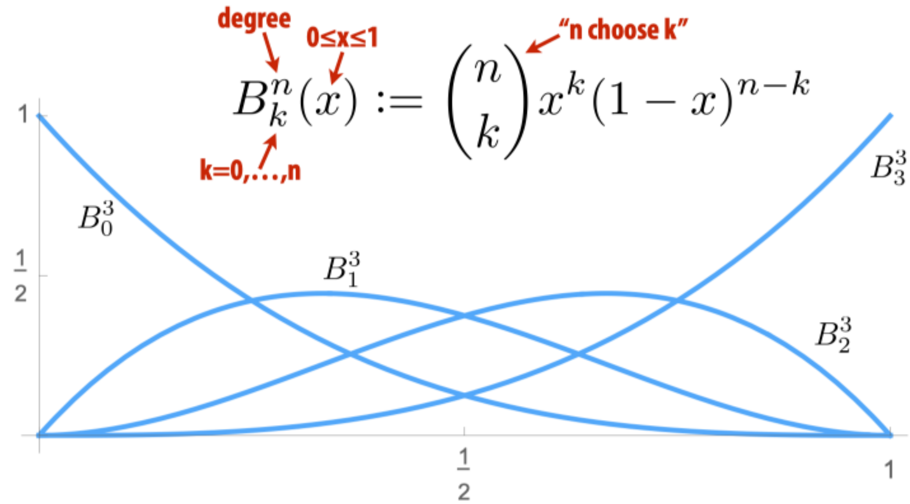
波恩斯坦基(Bernstein basis)

- 为什么限制我们只进行线性插值?
- 通过使用高等多项式会更具弹性
- 摒弃常见基$(1, x, x^2, x^3, \cdots)$,使用波恩斯坦基:



贝塞尔曲线(显式表达)

- 贝塞尔曲线也可以是以伯恩斯坦基表达的曲线:
- 通过使用高等多项式会更具弹性
- 摒弃常见基$(1, x, x^2, x^3, \cdots)$,使用波恩斯坦基:



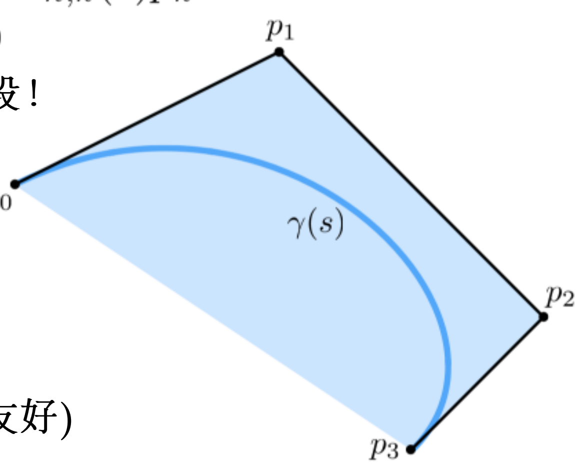
贝塞尔曲线(显式表达)

- 贝塞尔曲线也可以是以伯恩斯坦基表达的曲线:

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

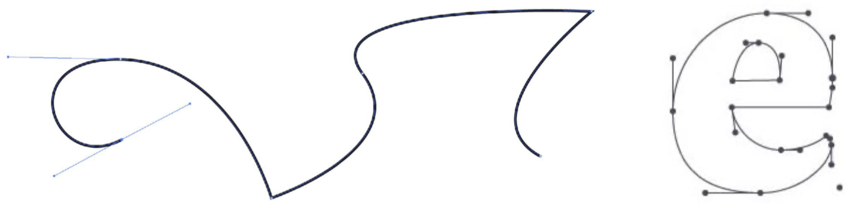
控制点

- 当n=1时，只是得到一条线段！
- 当n=3时，获得“立方贝塞尔曲线”
- 重要特征:
  1. 插值端点
  2. 到末端线段的切线
  3. 包含一个凸壳(对光栅化友好)



逐段的贝塞尔曲线(显式表达)

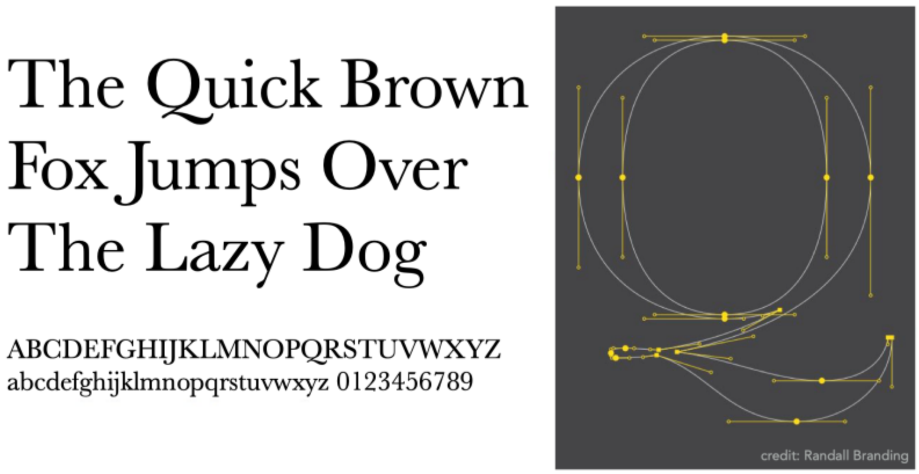
- 更多有趣的形状：逐段整合更多贝塞尔形状
- 广泛应用的技术（图解、字头、**SVG**、等等）



- 正式地，逐段贝塞尔曲线表达为：

$$\overset{\text{逐段贝塞尔}}{\gamma(u)} := \underset{\text{单一贝塞尔}}{\gamma_i}\left(\frac{u-u_i}{u_{i+1}-u_i}\right), \qquad u_i \leq u < u_{i+1}$$

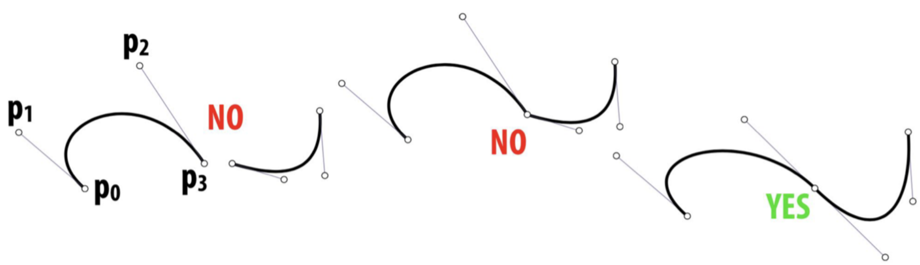
矢量字体



Baskerville字体 - 以立方贝塞尔样条的方式表达

贝塞尔曲线 – 切线连续性

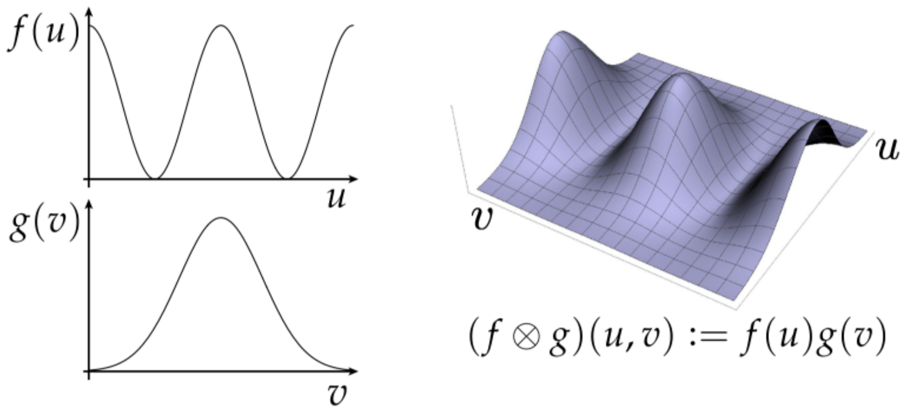
- 要获得“连续”曲线，应该将点和切线对齐



- 是的，但如何做?
- 每个曲线是一个立方等式：$au^3 + bu^2 + cu + d$
- 问题：存在存在多少约束? 对比 存在多少自由度?
- 问题：你能使用二次方贝塞尔完成这件事情吗? 线性贝塞尔呢?

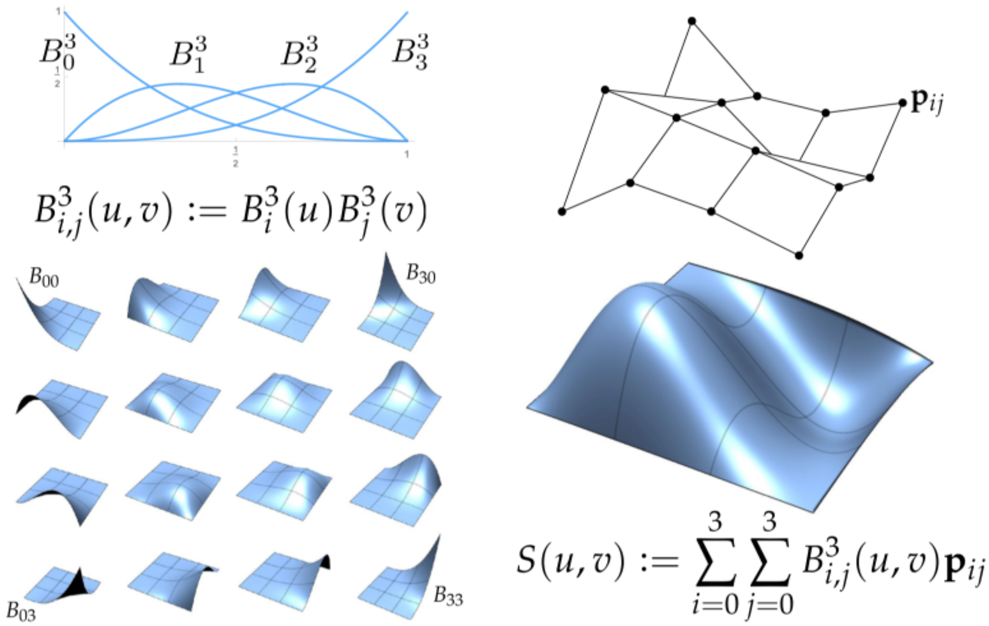
张量积(tensor product)

- 可以借助一对曲线获得表面
- 在任一点(u, v)的值，都借助坐标**u**上的曲线**f**和坐标**v**上的曲线**g**的乘积而获得（有时候又被称作“张量积”）



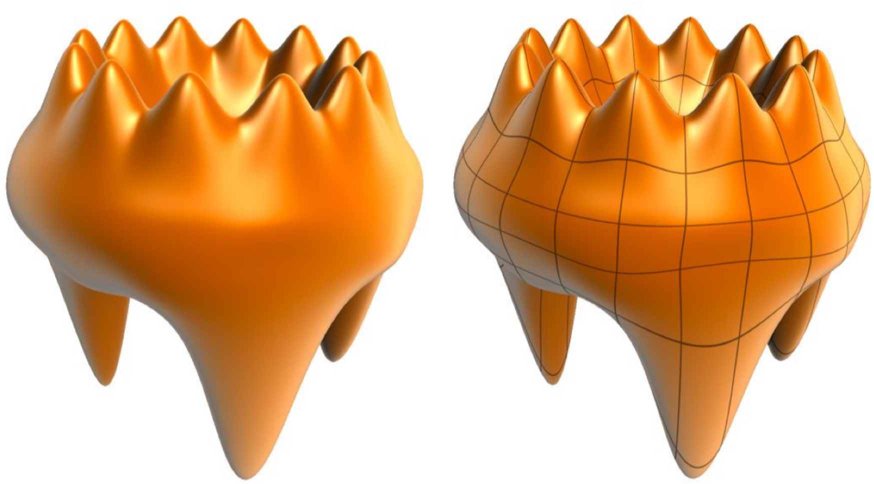
贝塞尔补丁(Bezier Patches)

- 贝塞尔补丁是伯恩斯坦基(Bernstein base)的张量积之和



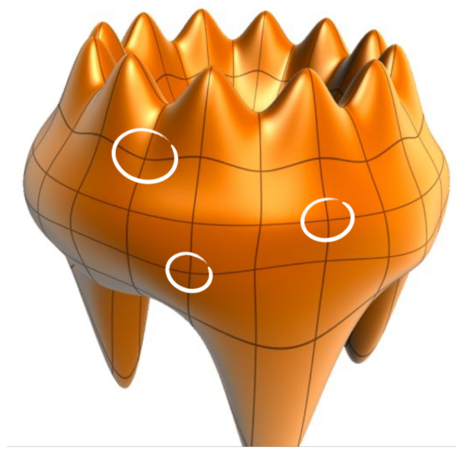
贝塞尔表面(Bezier Surface)

- 正如我们连接了贝塞尔曲线，也可以将贝塞尔补丁连接在一起获得贝塞尔表面：



- 非常易于绘制：只要将每块补丁划分为规则的(u, v)网格

贝塞尔补丁(Bezier Patch)十分简洁



- 注意，围绕每个顶点刚好有4个补丁
- 事实上，这产生了很大的约束性：
- 要生成有趣的形状（同时具备好的连续性），需要补丁具备更有趣的连接方式...

样条补丁方案

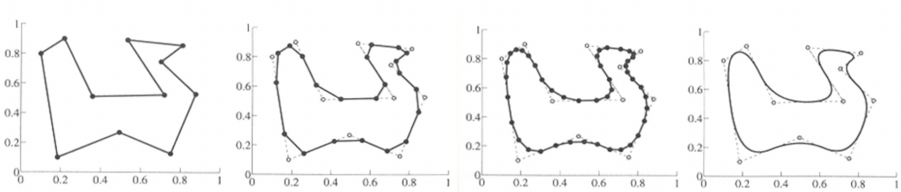
- 存在很多可选方法
- **NURBS, Gregory, Pm, polar...**
- 折衷：
    - 自由度
    - 连续性
    - 编辑的难度
    - 评估的代价
    - 通用型

- ...

- 如常：挑选适合工作的正确工具！

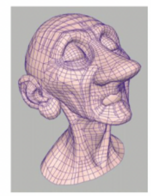
曲面细分(显式表达还是隐式表达?)

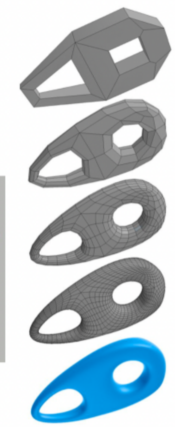
- 针对曲线或者表面可供选择的起始点：曲面细分
- 始于控制曲线
- 在每个边缘重点插入新的顶点
- 根据固定规则插入顶点位置
- 仔细挑选平均规则，获取平滑曲线
  - 一些曲面细分方案对应于总所周知的样条方案



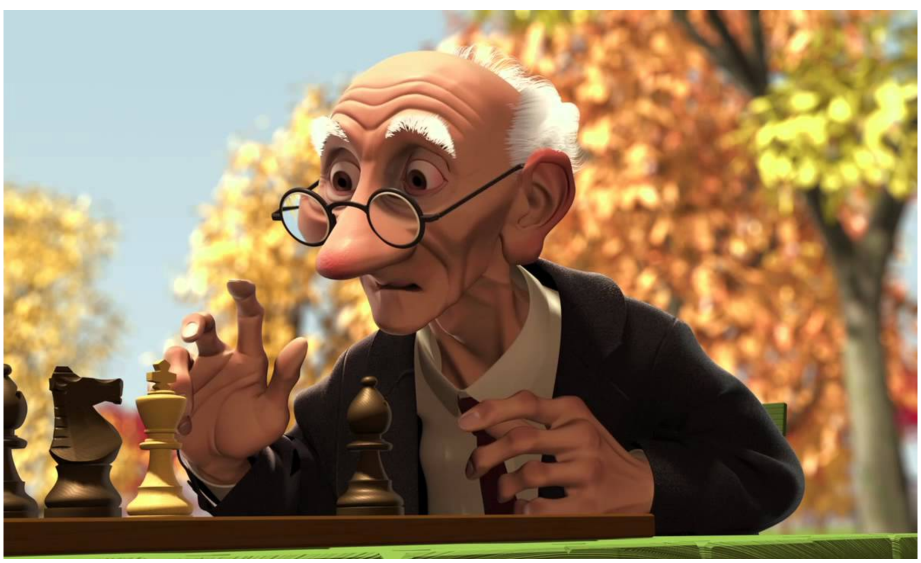
曲面细分(显式表达)

- 开始于粗粒度的多边形网格 （“控制笼子 -control cage”）
- 细分每个元素
- 经由局部平均化更新顶点
- 很多可能的规则：
  - **Catmull-Clark**（quads）
  - **Loop**（三角形）
  - ...
- 常见问题：
  - 选择插值还是近似?
  - 顶点处的连续性?
- 建模时比样条更简单；比逐点评估要难点





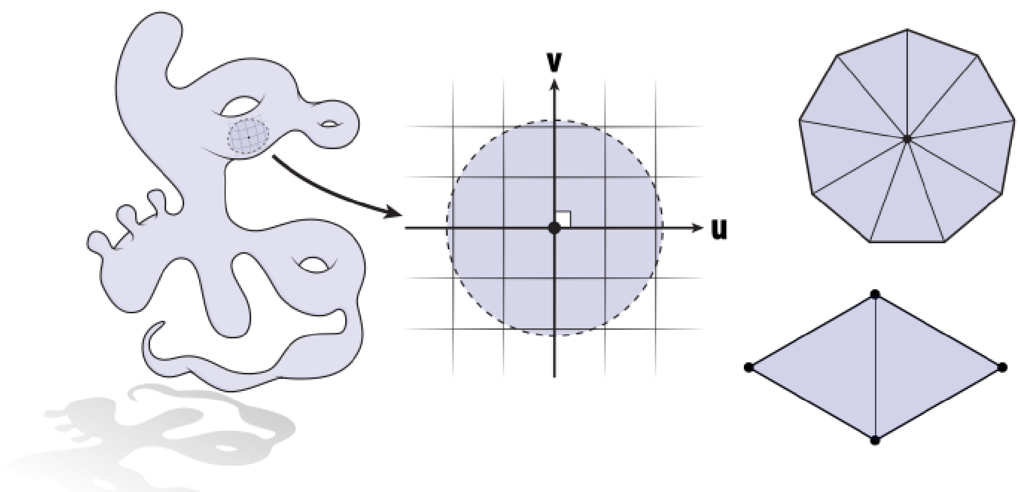
真实应用的曲面细分（皮克斯的“Geri’s Game”）



表面和流形(manifolds)

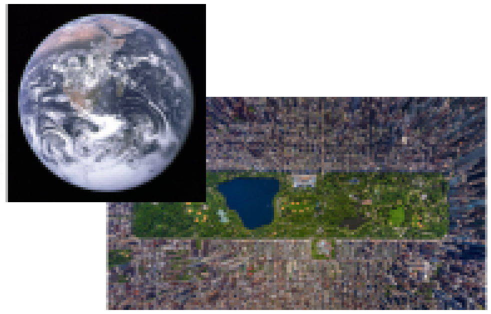
流形假设

- 现在我会介绍流形几何的基本思路
- 首先可能很难理解动机
  - 在下一课中这个概念将会变得清晰



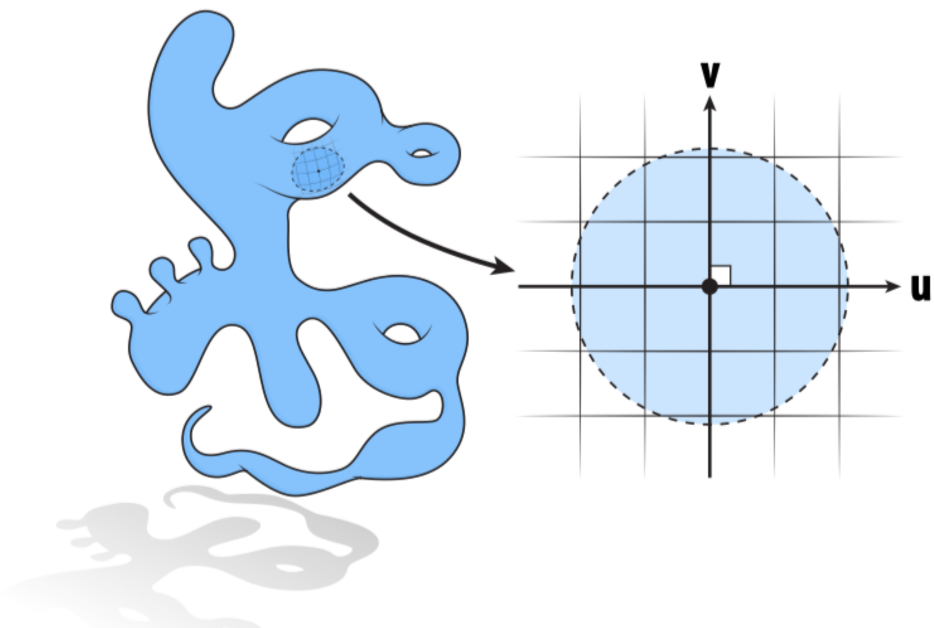
平滑表面

- 直观上来讲，表面是物体的“壳”或者范围边界
- （想象一下糖衣，不是巧克力那种）
- 表面为流形
  - 只要你放大得足够大（在任何点），都会看起来像一个平面（...或者可以轻松得被弄平整，不需要切削）
  - 比如.从空中观看地球 对比 从地面观看地球



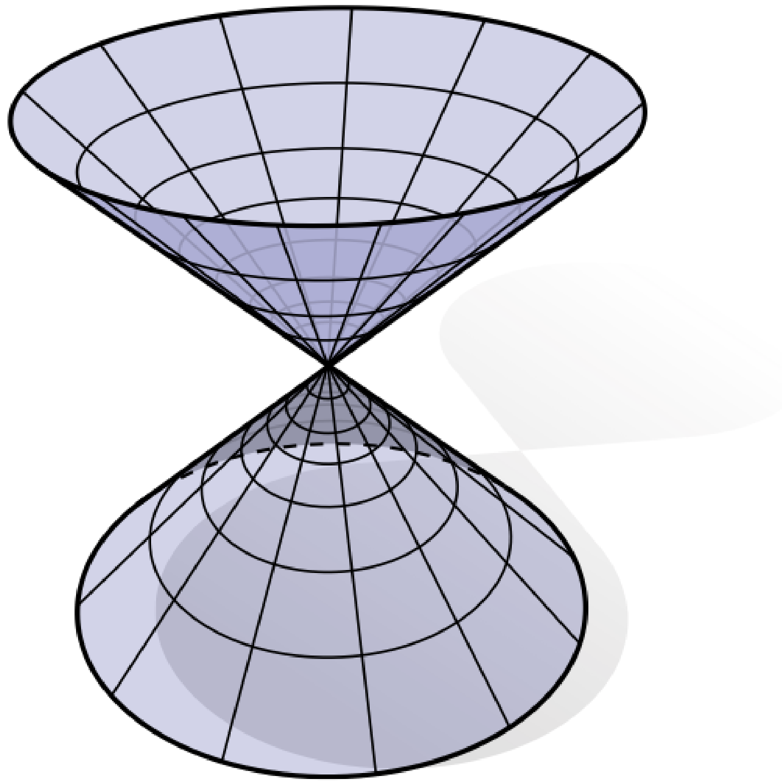
为何流形属性是有价值的?

- 让生活轻松，所有表面看起来都是相似的（至少局部）
- 会给出我们坐标！（至少局部）



不是每个形状都是流形?

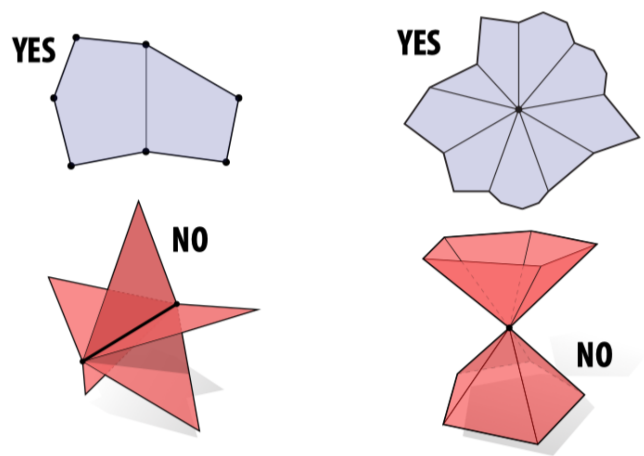
- 是，例如：



中部的点看起来怎么都不像平面，不管我们如何靠近！

流形多边形网格具有扇面，但不会有鱼鳍

- 对于多边形表面，只需要检查两个简单的条件：
  1. 每个边缘只包容两个多边形（但鱼鳍状边缘不是这样的）
  2. 包含每个顶点的多边形形成了单一的“扇面”?

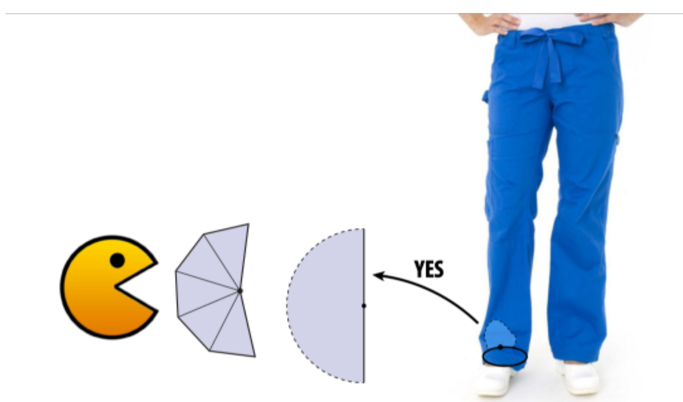


边缘是什么情形?

- 边缘是表面“结束”的地方
- 比如，裤子的脚踝和腰部
- 局部，看起来像半个碟子
- 整体上，每个边缘构成了一个环状
- 多边形网格：

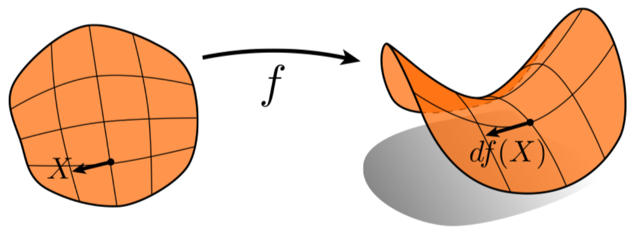
  边界的每个边缘包含一个多边形

  边缘顶点看起来像“packman”



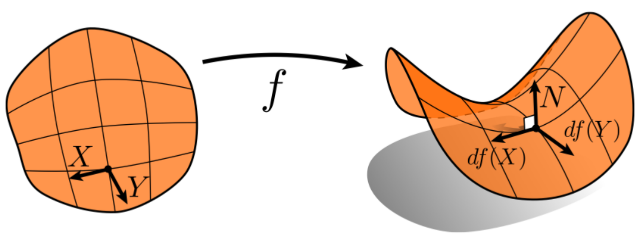
表面的测量

表面切线



表面法线(**N**)正交于所有的切线

$$N \cdot df(x) = 0 \; \forall X$$



一种法线常用的可视化方式

编码法线方向为RGB色彩，其值为灰度的差值

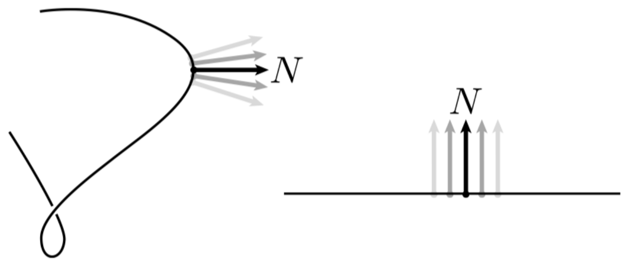
$R = 0.5 + 0.5\,N.x$

$G = 0.5 + 0.5\,N.y$

$B = 0.5 + 0.5\,N.z$

注意：伸缩和偏离法线值
以便我们可以表达法线的负值组件

曲率在法线上是变化的



曲率的半径