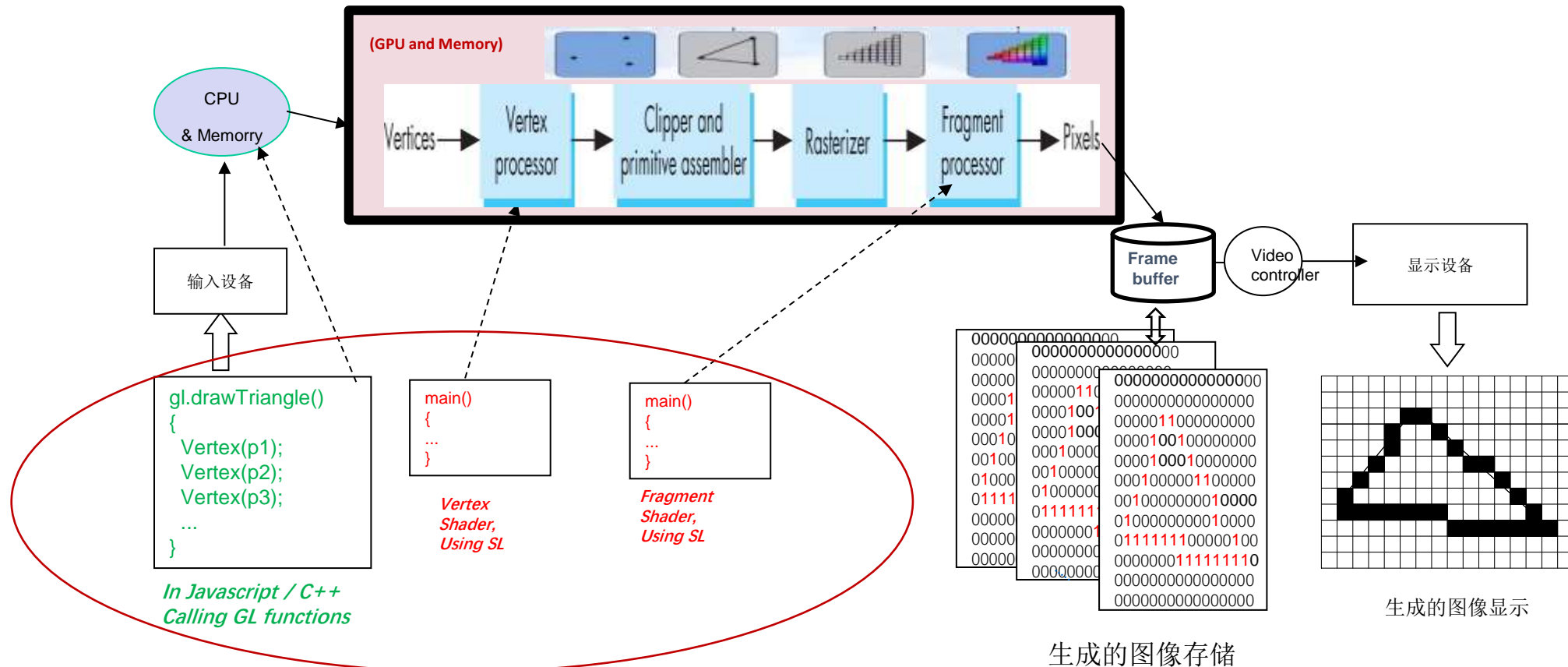




Recap

The Programmable Rendering Pipeline 可编程渲染管线的架构

- Performance is achieved by using GPU rather than CPU, GPU does all rendering,
- Programmer Control GPU through shaders
 - ◆ Application's job (run at CPU) is to send data to GPU
 - ◆ Vertex Shader run at per vertex, fragment shader run at per fragment concurrently

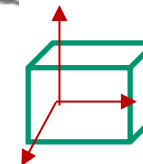
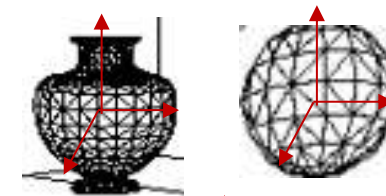


Modeling/Geometry

Room?

Modeling建模(Geometry几何)
~一般CPU完成~

Objects and Material Representation Data,



MC

Interaction and Animation 交互和动画

SC/DC



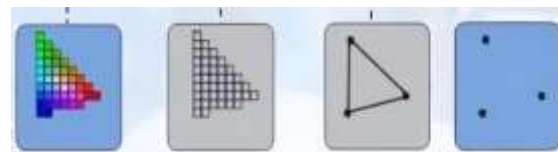
colored Image 2D

Rendering渲染,
~GPU完成~

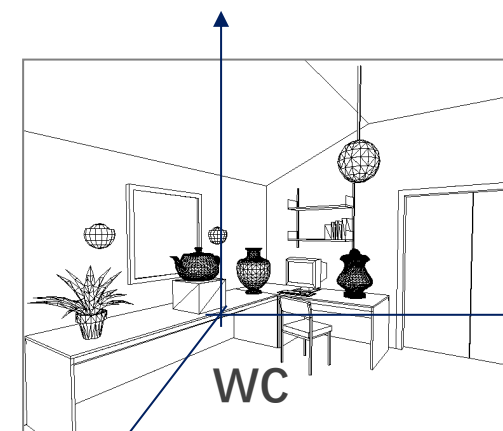
片元处理:
Fragment

图元组装和光栅化
Rasterization

顶点处理:
Vertex



VC
camera



Scene 3D

Scene description: objects, light etc.



Outline

- **Overview of Modeling/Geometry**
 - 图形的表示
 - 图形对象构成, 基本几何元素, 实体的定义,
 - 造型技术
- **Irregular Shape Modeling/Natural Modeling 不规则形体造型/自然造型**
 - Fractal geometry分形几何
 - Particle system粒子系统
- **Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型***
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - B-reps(Boundary Representation)边界表示*
 - Mesh* 网格
 - Spline Curve and Surface 样条曲线曲面*



The University of New Mexico

Overview of Modeling/Geometry

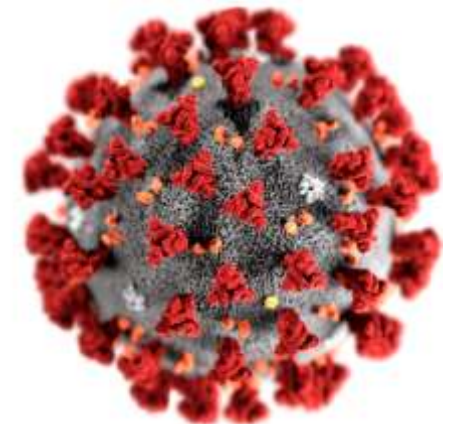
- Problem: 研究如何在计算机中“表示”不同的图形对象？



Maya中的直升机模型



Xfrog3.0生成的挪威云杉



Overview of Modeling/Geometry (cont.)

- Geometry Representation(几何表示)
 - Implicit(隐式),
 - Explicit(显式), (Parametric参数 ~显式)

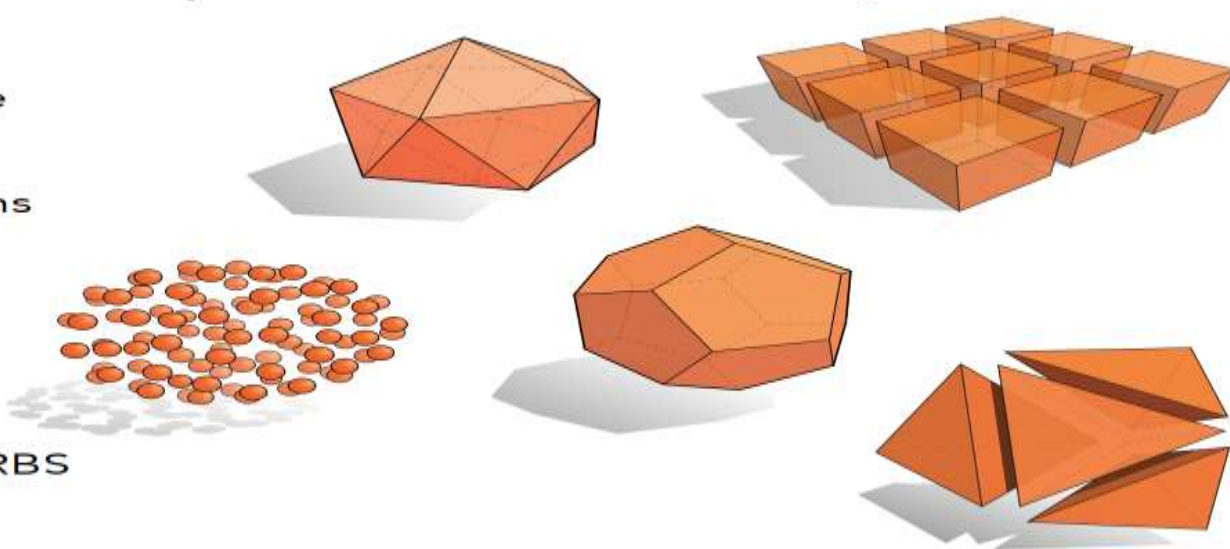
Many Ways to Represent Geometry

Implicit

- algebraic surface
- level sets
- distance functions
- ...

Explicit

- point cloud
- polygon mesh
- subdivision, NURBS
- ...



Each choice best suited to a different task/type of geometry

GAMES101 32 Lingqi Yan, UC Santa Barbara

Overview of Modeling/Geometry (cont.)

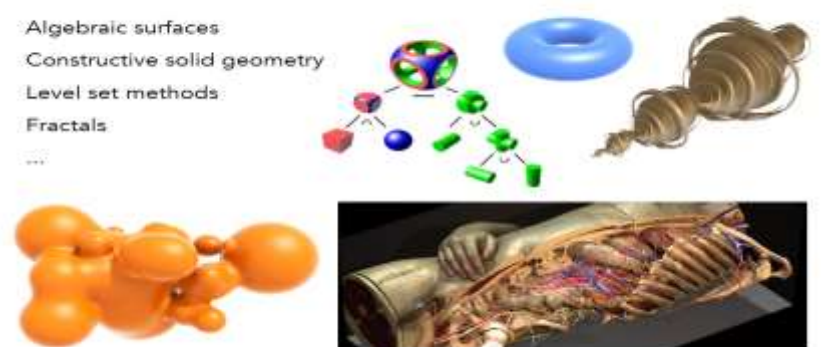
- Geometry Representation

- Implicit Representations (隐式表示): 满足特定关系的点集

如: 代数表面, 构造实体几何, 水平集level set, 分型fractal

Many Implicit Representations in Graphics

- Algebraic surfaces
- Constructive solid geometry
- Level set methods
- Fractals



GAMES101 42 Lingqi Yan, UC Santa Barbara

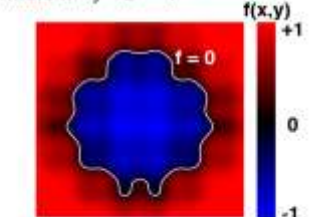
"Implicit" Representations of Geometry

Based on classifying points

- Points satisfy some specified relationship

E.g. sphere: all points in 3D, where $x^2+y^2+z^2 = 1$

More generally, $f(x,y,z) = 0$



GAMES101 33 Lingqi Yan, UC Santa Barbara

Constructive Solid Geometry (Implicit)

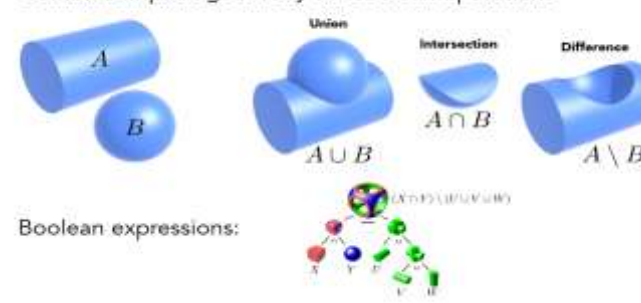
Combine implicit geometry via Boolean operations

Union: $A \cup B$

Intersection: $A \cap B$

Difference: $A \setminus B$

Boolean expressions:




GAMES101 44 Lingqi Yan, UC Santa Barbara

Fractals (Implicit)

Exhibit self-similarity, detail at all scales

"Language" for describing natural phenomena

Hard to control shape!



Overview of Modeling/Geometry (cont.)

• Implicit Representation (隐式表示举例)

- Much more robust(健壮)
- In general, we cannot solve for points that satisfy

Two dimensional curve(s) $g(x,y)=0$

- All lines $ax+by+c=0$,
- Circles $x^2+y^2-r^2=0$

Three dimensions $g(x,y,z)=0$ defines a surface

Intersect two surface to get a curve

Implicit Representations - Pros & Cons

Pros:

- compact description (e.g., a function)
- certain queries easy (inside object, distance to surface)
- good for ray-to-surface intersection (more later)
- for simple shapes, exact description / no sampling error
- easy to handle changes in topology (e.g., fluid)

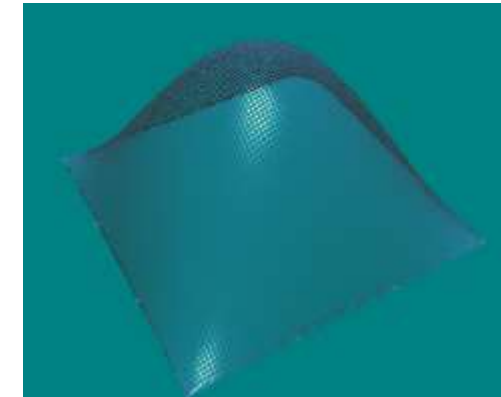
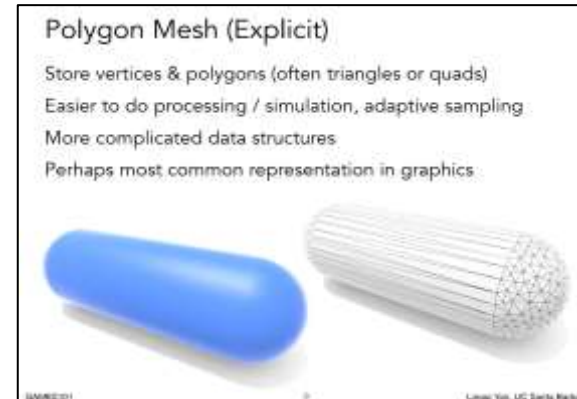
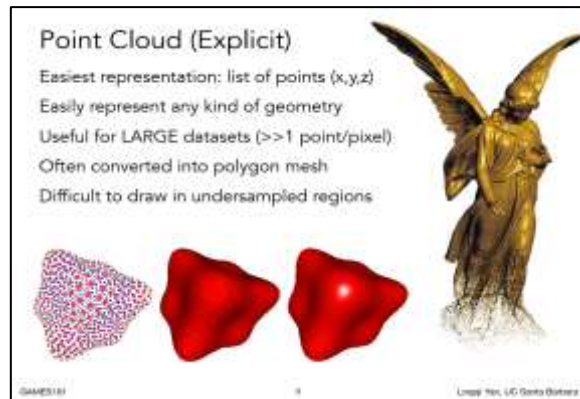
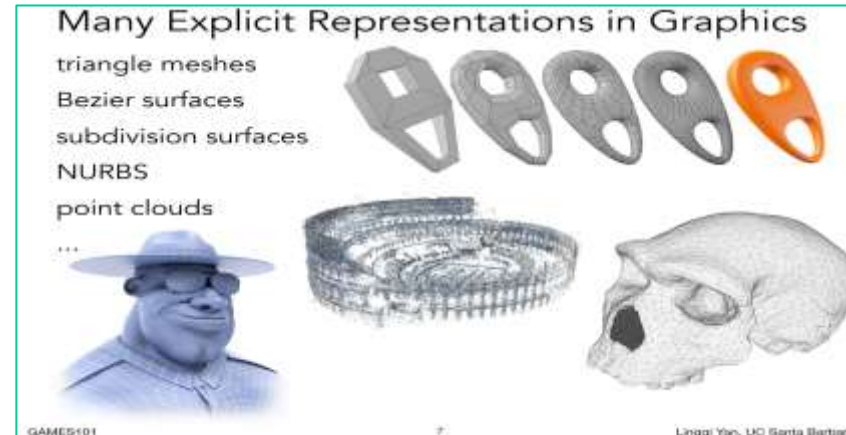
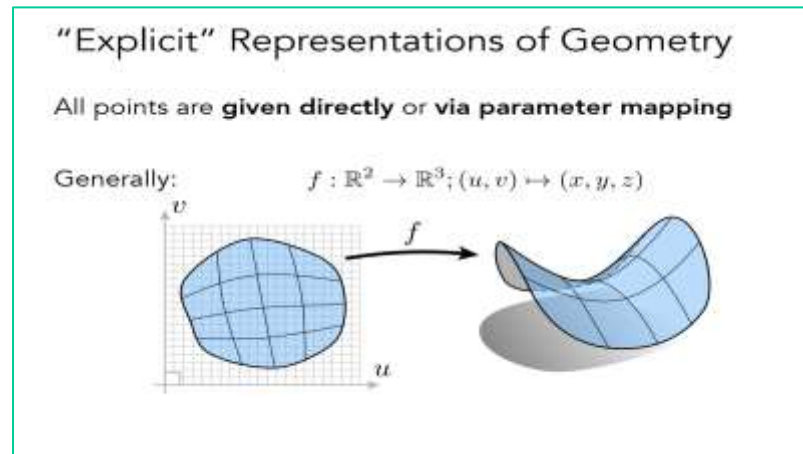
Cons:

- difficult to model complex shapes

Overview of Modeling/Geometry (cont.)

- Geometry Representation

- **Explicit Representations (显式表示):** 直接给出点 或 通过参数映射得到顶点属性
如: 点云point Cloud, 多边形网格(mesh), 参数曲线曲面(spline)



Overview of Modeling/Geometry (cont.)

- Explicit Representation (显式表示 举例)
 - Easy to Model complex shapes
 - Cannot represent all curves: Vertical lines, Circles

- Most familiar form of curve in 2D $y=f(x)$
- Extension to 3D $y=f(x), z=g(x)$
 - The form $z = f(x,y)$ defines a surface

- Parametric Sphere 参数球面

$$x(\theta, \phi) = r \cos \theta \sin \phi$$

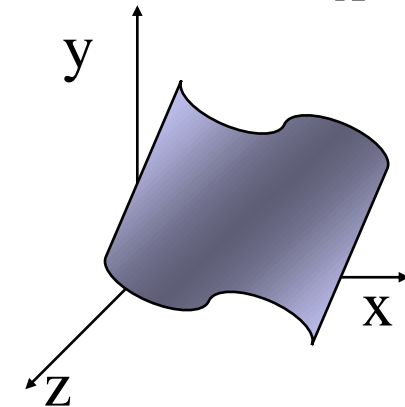
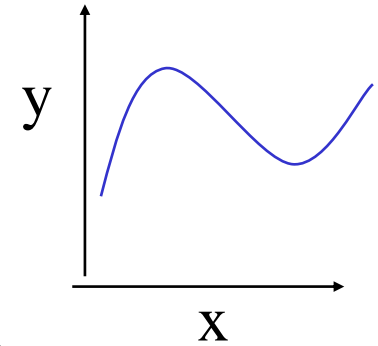
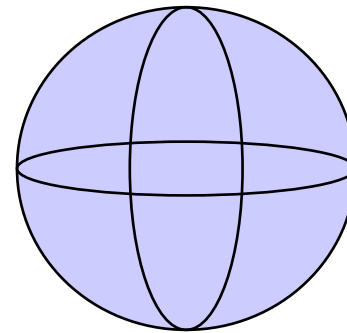
$$y(\theta, \phi) = r \sin \theta \sin \phi$$

$$z(\theta, \phi) = r \cos \phi$$

θ constant: circles of constant longitude

ϕ constant: circles of constant latitude

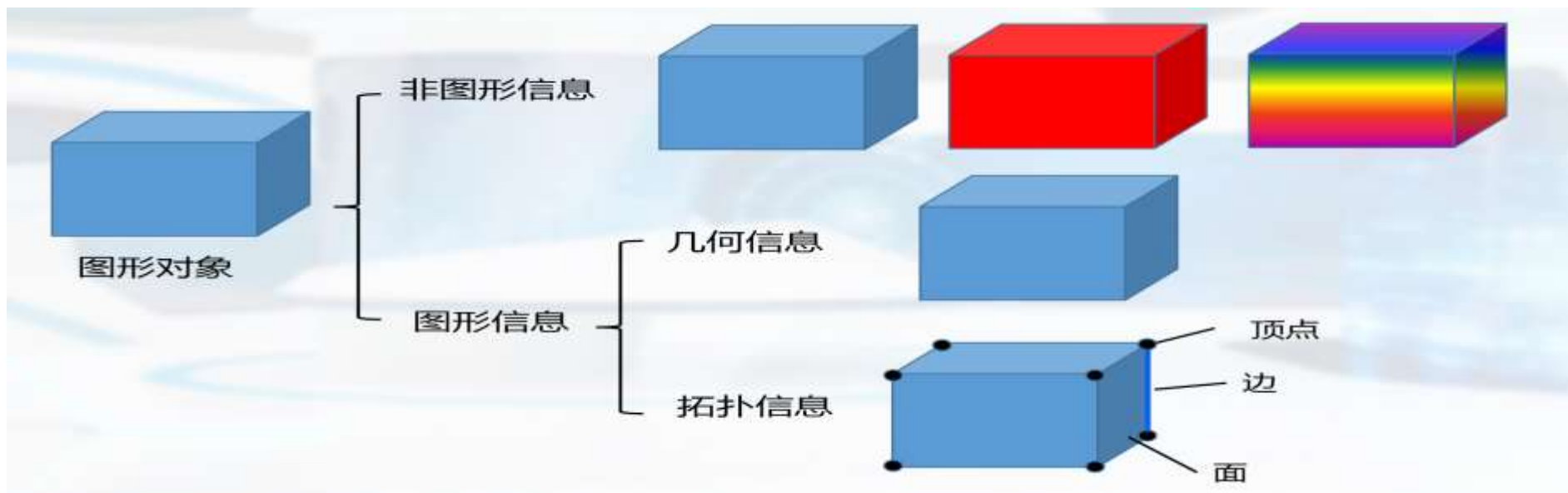
$$360 \geq \theta \geq 0, \quad 180 \geq \phi \geq 0$$



Overview of Modeling/Geometry (cont.)

➤ 图形对象的构成

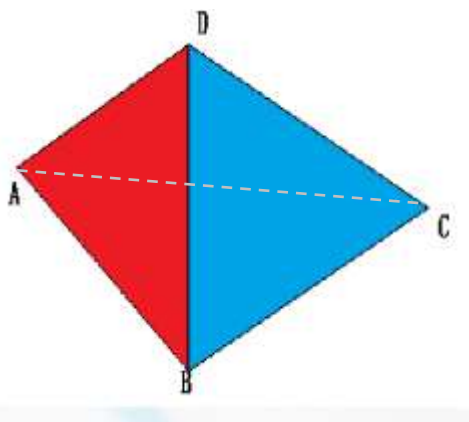
- **非图形信息**: 线型, 颜色, 亮度, 质量, 比重, 体积等。
- **图形信息**: 对象和构成它的点、线、面的位置和相互关系, 几何尺寸等。
 - **几何信息**: 形体在欧氏空间中的位置和大小。
 - **拓扑信息**: 形体各分量的数目和相互连接关系。



Overview of Modeling/Geometry (cont.)

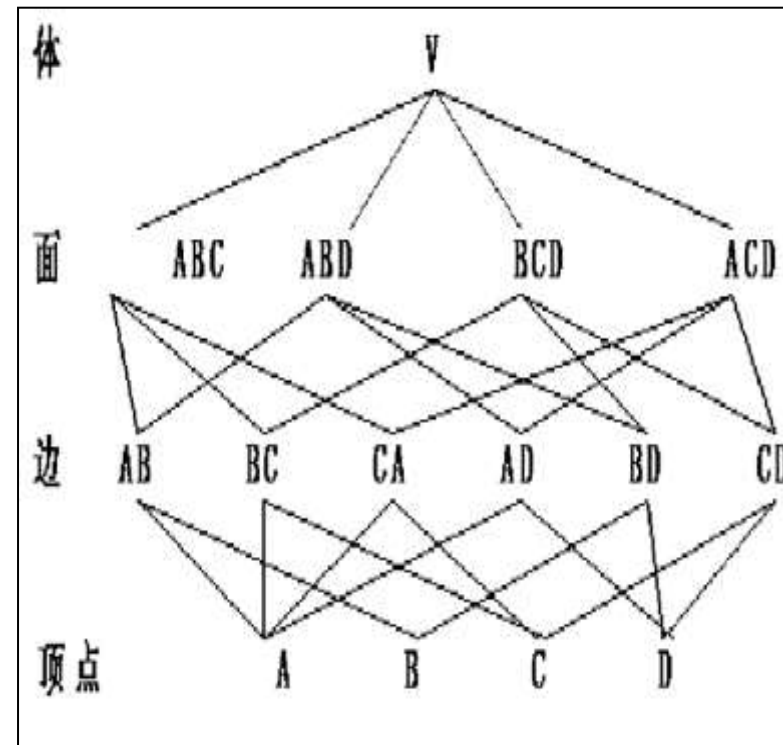
➤ 图形对象的构成 (cont.)

➤ 图形信息 = 几何信息 (顶点表、边表、多边形表) + 拓扑信息



四面体 ABCD

顶点表		边表		面表	
A	x_1, y_1, z_1	AB	A, B	ABC	AB, BC, AC
B	x_2, y_2, z_2	BC	B, C	ABD	AB, BD, AD
C	x_3, y_3, z_3	CA	C, A	BCD	BC, CD, BD
D	x_4, y_4, z_4	AD	A, D	ACD	AC, CD, AD



Overview of Modeling/Geometry (cont.)

➤ 图形的基本几何元素*

点

边

(面/环)

体

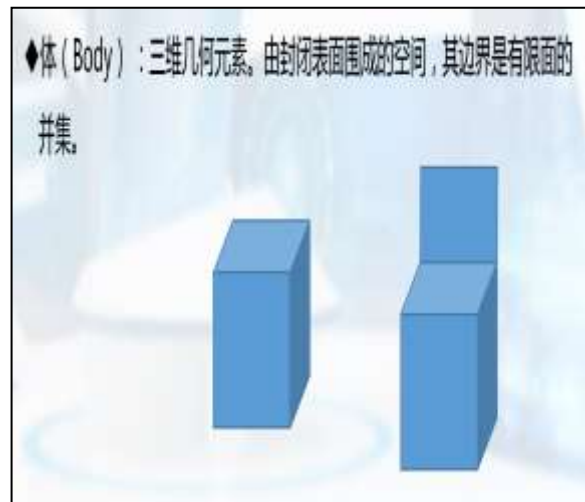
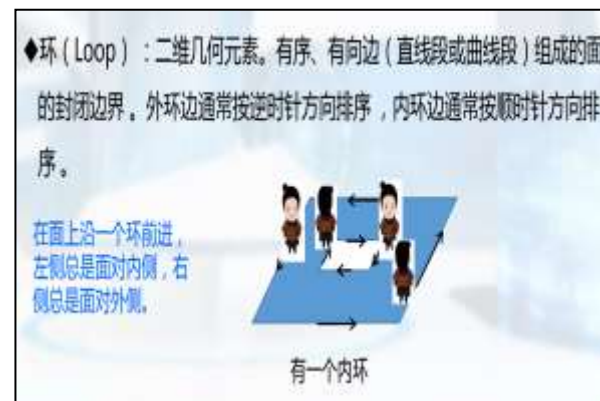
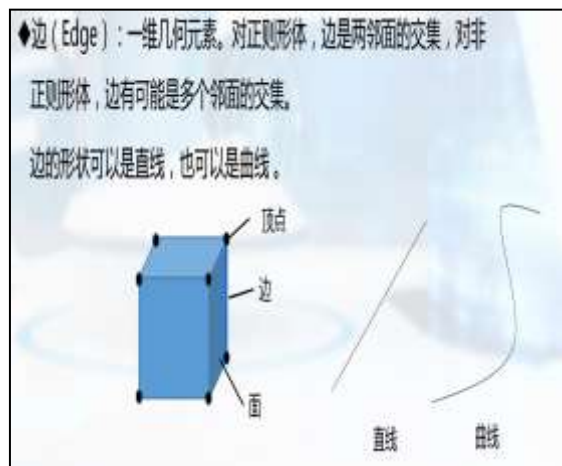
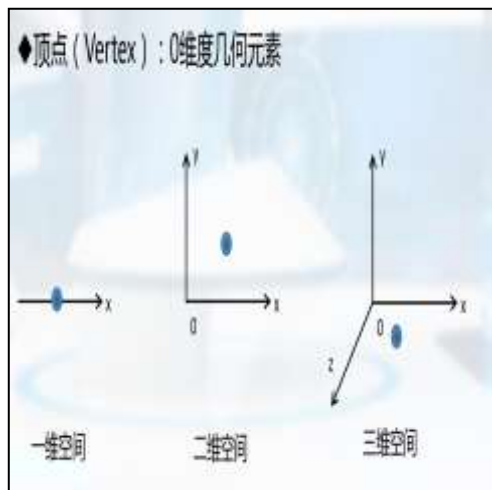
请问: 图形是几维的? (图形维数不是坐标空间的维数, 与坐标系无关, 而是图形向量维数/几何性质)

0维 ~ P

1维 ~ P(u)

2维 ~ P(u,v)

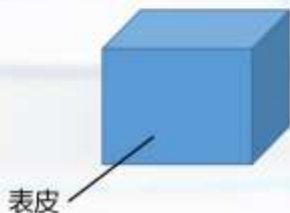
3维 ~ P(u,v,n)



Overview of Modeling/Geometry (cont.)

➤ 实体的定义

三维空间中的物体是一个内部连通的三维点集。形象地说，是由其内部的点集及紧紧包着这些点的表皮组成的。

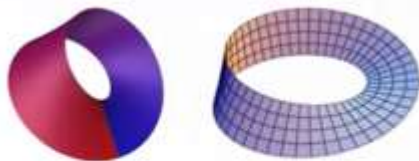


物体表面的性质：

- (1) **连通性**：位于物体表面上的任意两个点都可用实体表面上的一条路径连接起来；
- (2) **有界性**：物体表面可将空间分为互不连通的两部分，其中一部分是有界的；
- (3) **非自相交性**：物体的表面不能自相交；
- (4) **可定向性**：表面的两侧可明确定义出属于物体的内侧或外侧；
- (5) **闭合性**：物体表面的闭合性是由表面上多边形网格各元素的拓扑关系决定的。



克莱茵瓶 (Klein Bottle)
自交且不可定向的封闭曲面



莫比乌斯带 (Mobius Band)
单边不可定向

实体的性质：

- (1) **刚性**：必须具有一定的形状；
- (2) **维数的一致性**：三维空间中，一个物体的各部分均应是三维的；
- (3) **占据有限的空间**：体积有限；
- (4) **边界的确定性**：根据物体的边界能区别出物体的内部及外部；
- (5) **封闭性**：经过一系列刚体运动及任意序列的集合运算之后，仍然是有效的物体。

Overview of Modeling/Geometry (cont.)

➤ Modeling Technique 造型技术:

- 研究“如何建立恰当的模型以表示不同图形对象”的技术，也称为CAGD (Computer Aided Geometric Design) 计算机辅助几何设计
- “Best Representation Depends on the Task” 根据任务建立最好的表示

➤ 规则对象 (“几何形体”) 如点、直线、曲线、平面、曲面或实体

- => “几何造型” (以构造“几何形体模型”进行模拟)



➤ 不规则对象 (“自然形体”): 如山、水、树、草、云、烟等自然界丰富多彩的对象

- => “过程造型” (即构造“简单模型”并用少量易于调节参数进行模拟)





Outline

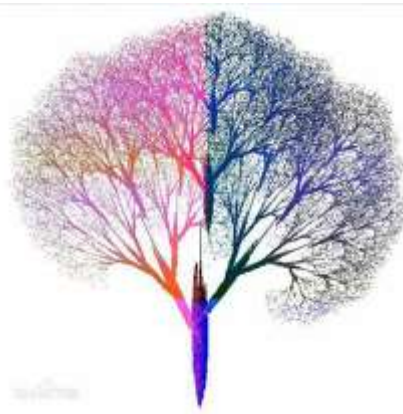
- Overview of Modeling
 - 图形的数学表示
 - 图形对象构成,基本几何元素,实体的定义
 - 图形形体分类及对应的造型技术*
- Irregular **Shape** Modeling/Natural Modeling 不规则形体造型/自然造型
 - Fractal geometry分形几何
 - Particle system粒子系统
- Regular **Shape** Modeling/Solid Modeling 规则形体造型/实体造型*
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - ***B-reps(Boundary Representation)边界表示****
 - Mesh* 网格
 - Spline Curve and Surface 样条曲线曲面*

Natural Modeling- Fractal Geometry



- 分形几何(Fractal Geometry)

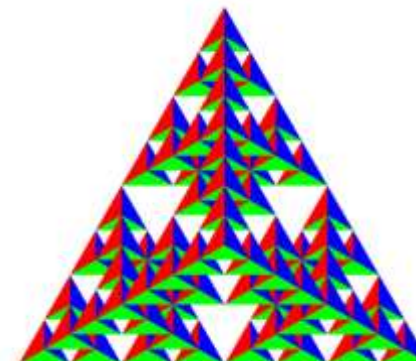
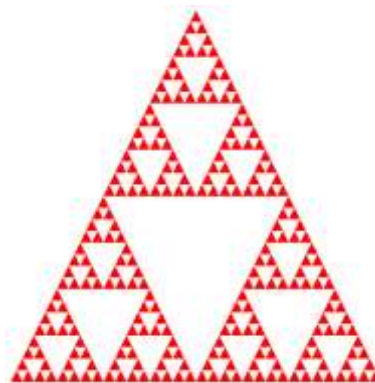
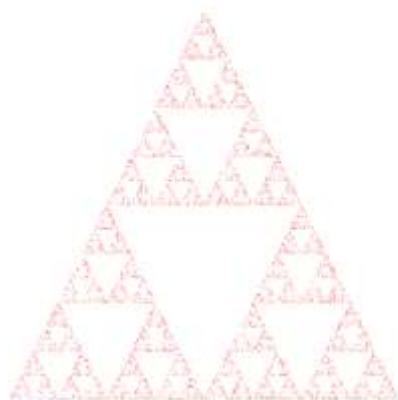
- 著名数学家 Mandelbrot 曼德尔布诺特/芒德勃罗, 于1980创立的分形几何理论
- **Fractal**: 不规则的、破碎的、分数的,
- 分形图形是一个粗糙或零碎的几何形状, 可以分成数个部分, 且每一部分都 (至少近似地) 是整体缩小后的形状”, 即具有**自相似的性质**, 具有以**非整数维形式**充填空间的形态特征。



Natural Modeling- Fractal Geometry (cont.)

- 分形几何(Fractal Geometry) 实例
 - 门格海绵, 科赫雪花, 龙骨曲线, ...

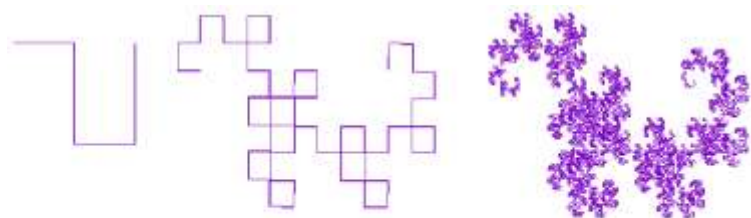
Ref: [angle8Ecode/02/gasket.html](http://angle8Ecode/02/gasket*.html)*



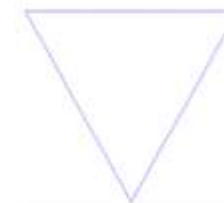
蕨菜叶



门格海绵



一次、五次、六次迭代的龙骨曲线



科赫Koch雪花曲线

Natural Modeling- Fractal Geometry (cont.)

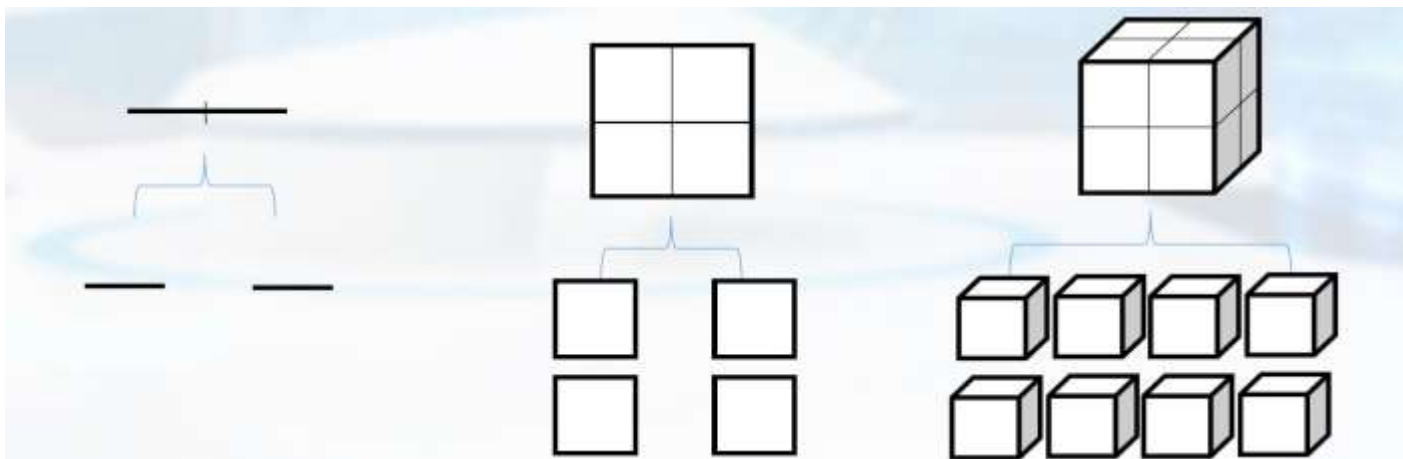
• 分形维数/分数维数/相似性维数

- 如果某图形是由把原图缩小为 $1/a$ 的相似的 b 个图形所组成

$$a^D = b$$

- 指数 D 就是分形维数，也称为相似性维数。
- D 可以是整数，也可以是分数(分数维)

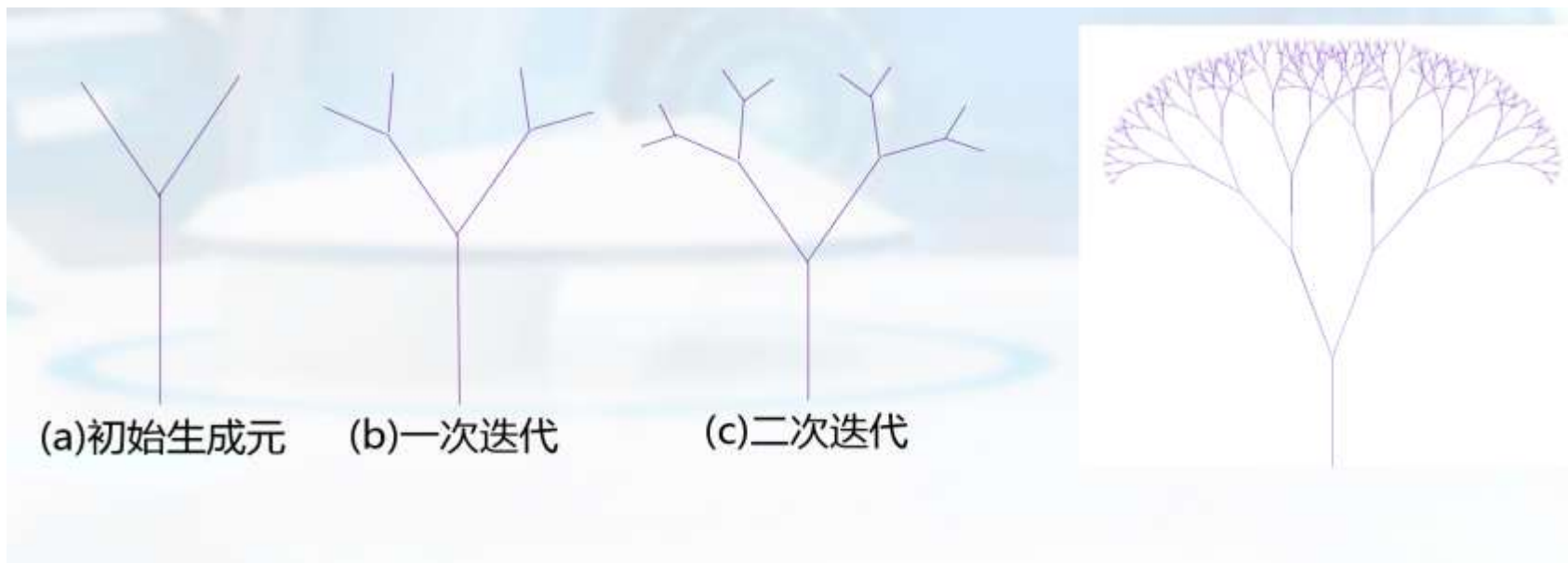
其线段、正方形、立方体分别被等分为 2^1 、 2^2 和 2^3 个相似的子图形，其中的指数1、2、3，正好等于与图形相应的经验维数。



Natural Modeling- Fractal Geometry (cont.)

• 分数维造型

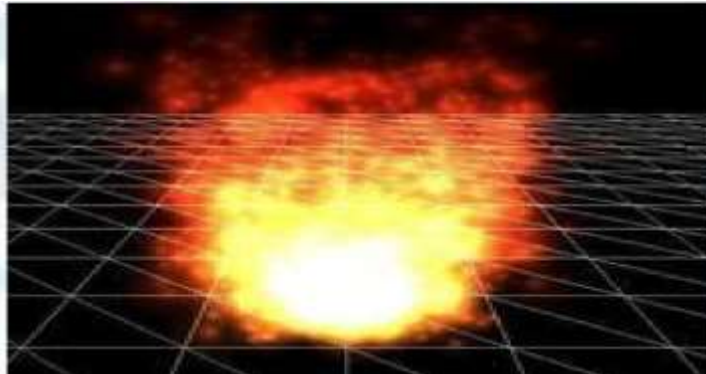
- 生成过程: 为产生物体局部细节指定一重复操作。例如:
- 确定性自相似分形法:
 - 初始生成元(initiator): 一个几何形状
 - 生成元(generator): 初始生成元的每部分用一个模型替代, 多次迭代后生成分形图



Natural Modeling- Particle Systems

- particle systems 微粒系统

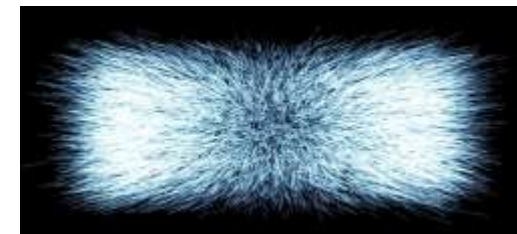
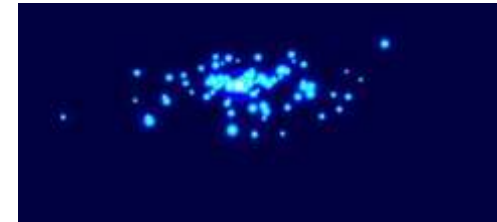
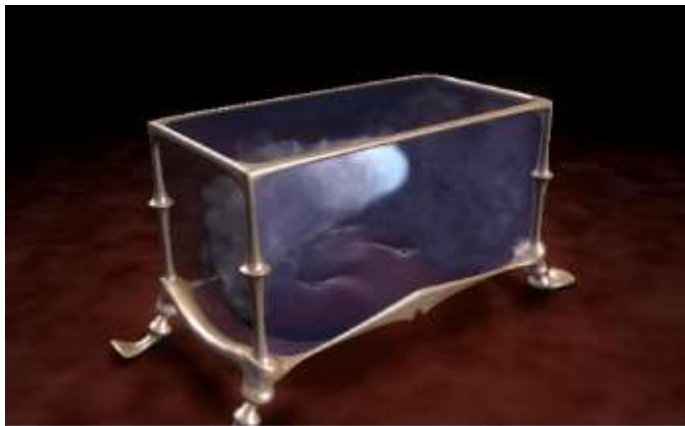
- 经常使用粒子系统模拟的现象有火、爆炸、烟、水流、火花、落叶、云、雾、雪、尘、流星尾迹或者象发光轨迹这样的抽象视觉效果等等。



Natural Modeling- Particle Systems(cont.)

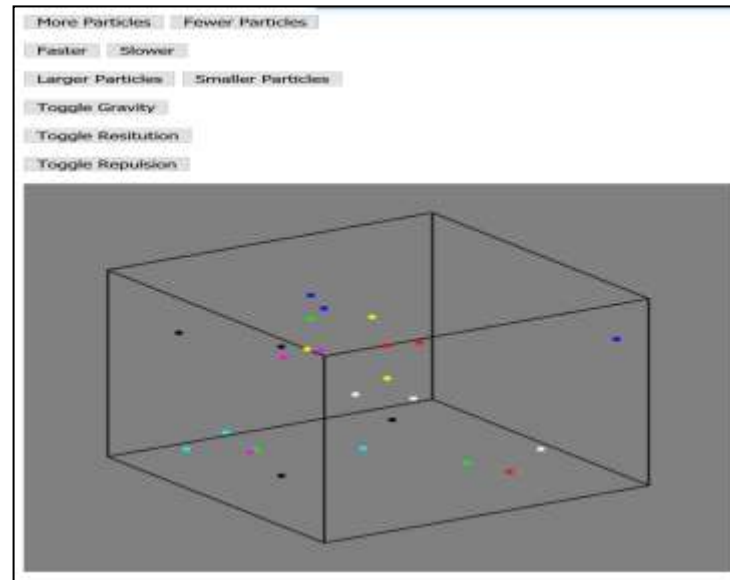
• particle systems微粒系统(cont.)

- 微粒系统是一组分散的微小物体集合，这些微小物体 的大小和形状可随时间变化，也就是说它们可以按照某种算法运动变化。
- 尤其擅长描述随时间变化的物体，可以用于模拟自然景物或模拟其它非规则形状物体展示“流体”性质。
- 微粒运动的模拟方式：随机过程模拟、运动路径模拟、力学模拟。



Natural Modeling- Particle Systems(cont.)

- particle systems 微粒系统(cont.)
 - 生成的两要素：粒子本身的造型 和 粒子的运动方式
 - 生成的两个过程：模拟和渲染
 - 模拟粒子的碰撞，重力，粒子吸引力等物理运动规律
 - Ref: angle8ECode/10/particleSystem.html





Outline

- Overview of Modeling
 - 图形的数学表示
 - 图形对象构成,基本几何元素,实体的定义
 - 图形形体分类及对应的造型技术*
- Irregular Shape Modeling/Natural Modeling 不规则形体造型/自然造型
 - Fractal Geometry分形几何
 - Particle System粒子系统
- **Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型***
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - ***B-reps(Boundary Representation)边界表示****
 - Polygon Mesh*
 - Spline Curve and Surface 样条曲线曲面*

Solid Modeling

规则形体/实体模型的三类表示：

1. 构造实体几何 (Constructive Solid Geometry, CSG)

它将实体表示成立方体、长方体、圆柱体、圆锥体等基本体素的组合，可以采用并、交、差等运算构造新的形体。

2. 空间分割表示 (Space-Partitioning Representation, SP)

用来描述物体的内部性质，将包含一物体的空间区域划分成一组小的、非重叠的、连续实体（通常是立方体），由这些小实体集合构成。

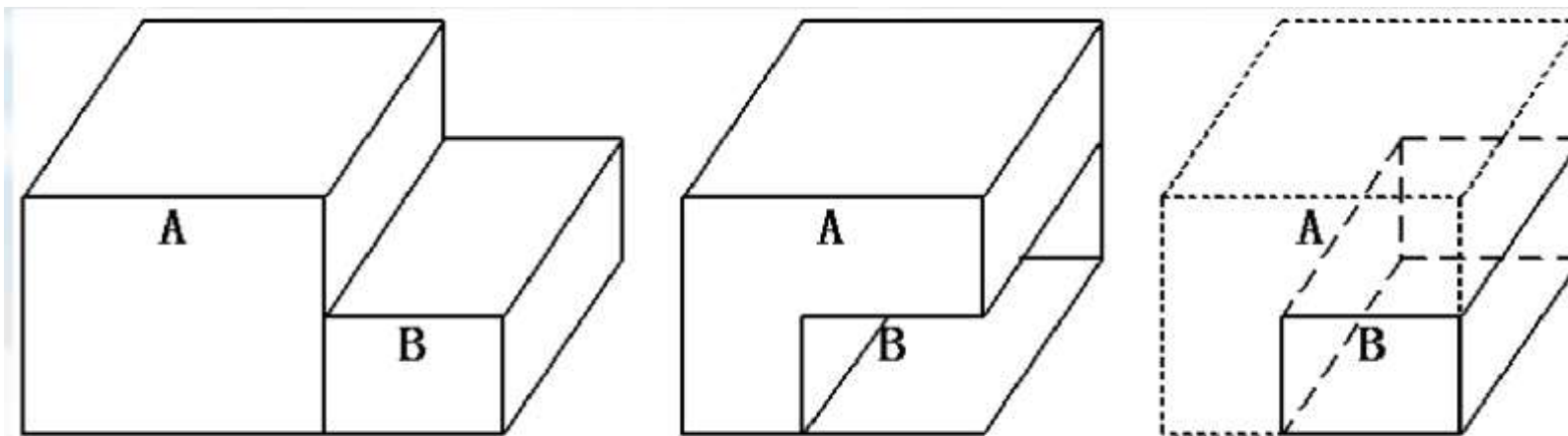
3. 边界表示 (Boundary Representation, B-reps)*

即用一组曲面（含平面）来描述物体，这些曲面将物体分为内部和外部。边界表示具体又包括多边形表面模型和扫描表示两种。

Solid Modeling -1.CSG

• 1)构造实体几何(CSG)

- 构造实体几何表示 (Constructive Solid Geometry, CSG)
- 它将实体表示成立方体、长方体、圆柱体、圆锥体等基本体素的组合,
- 可以采用并、交、差等运算, 构造新的形体。



(a) A, B形体的并

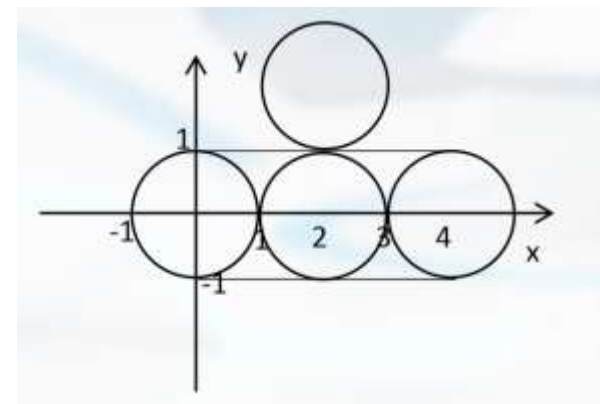
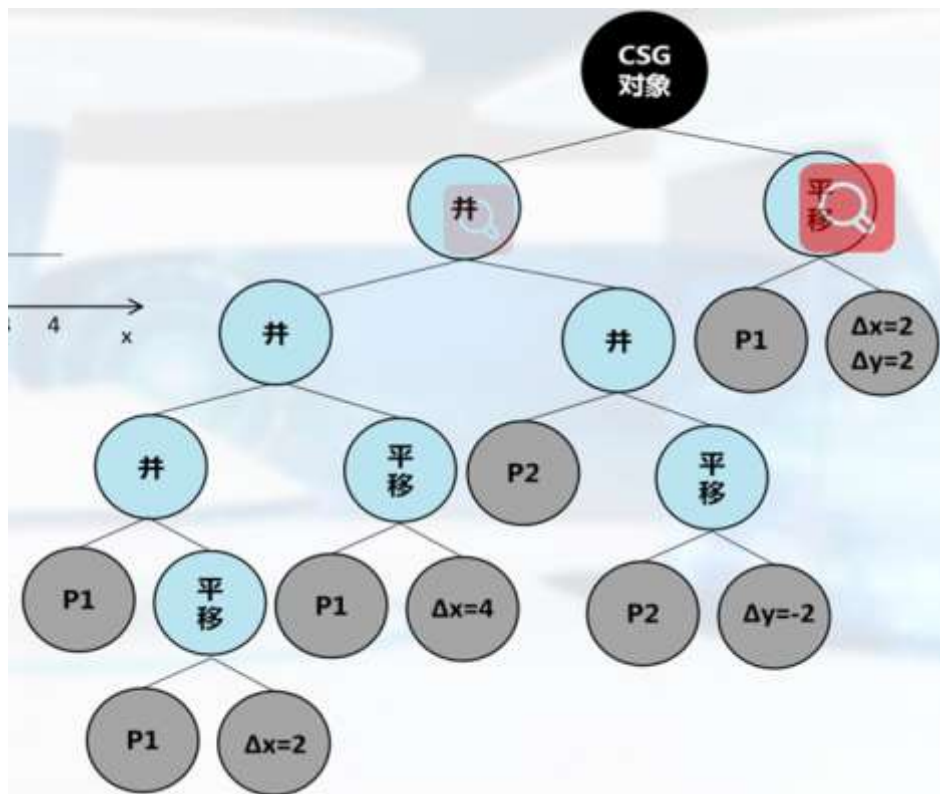
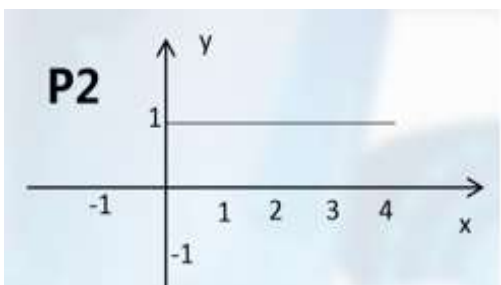
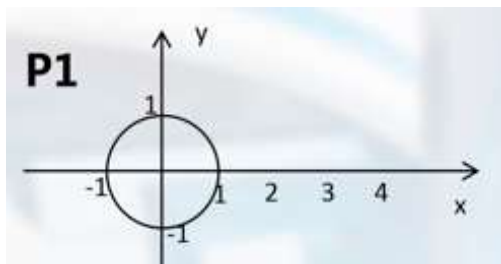
(b) A, B形体的差

(c) A, B形体的交

Solid Modeling -1.CSG(cont.)

• 1)构造实体几何表示(CSG)~二维实例

- 集合运算的实现过程可以用一棵二叉树(称为CSG树)
- 叶子结点:基本体素或变换参数
- 非叶子结点:施加于子结点的正则集合算子或几何变换定义
- 根结点:最终结果~实体~



Solid Modeling -1.CSG(cont.)

1)构造实体几何表示(CSG)-优缺点分析

优点:如果体素设置比较齐全,通过集合运算就可构造出多种不同的符合需要的实体。

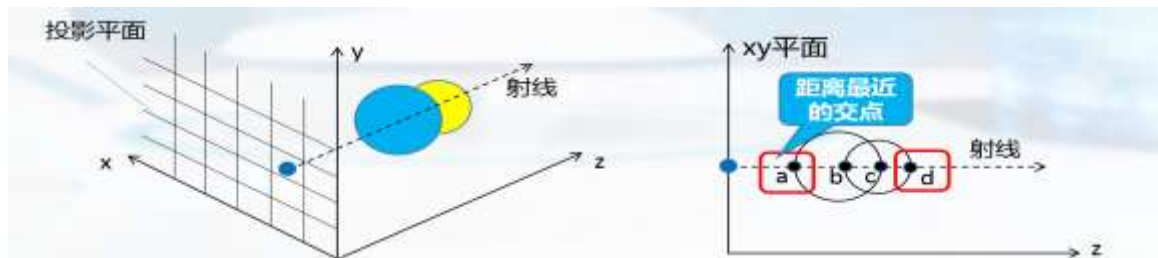
缺点:当用户输入体素时,主要是给定体素的有关参数,然后由系统给出该体素的表面方程,再由系统进行集合的求交运算,最后得到生成的实体。这里存在两个问题:

- 1)一是集合运算的中间结果难以用简单的代数方程表示,求交困难;
- 2)CSG树不能显式地表示形体的边界,因而无法直接显示CSG树表示的形体。

CSG找边界显示代价大,边界求法可采用“光线投射法”来直接找。

光线投射算法的核心思想:

- 从显示屏幕(投影平面)的每一象素位置发射一根光线(射线),
- 求出射线与距离投影平面最近的可见表面的交点和交点处的表面法矢量。
 - (1)将射线与CSG树中的所有基本体素求交,求出所有的交点;
 - (2)将所有交点相对于CSG树表示的物体进行分类,确定位于物体边界上的那部分交点;
 - (3)对所有位于物体边界上的交点计算它们在射线上的参数值并进行排序,确定距离最近的交点。并得到其所在基本体素表面的法矢量。
- 然后根据光照模型 计算出表面可见点的色彩和亮度,生成实体的光栅图形。



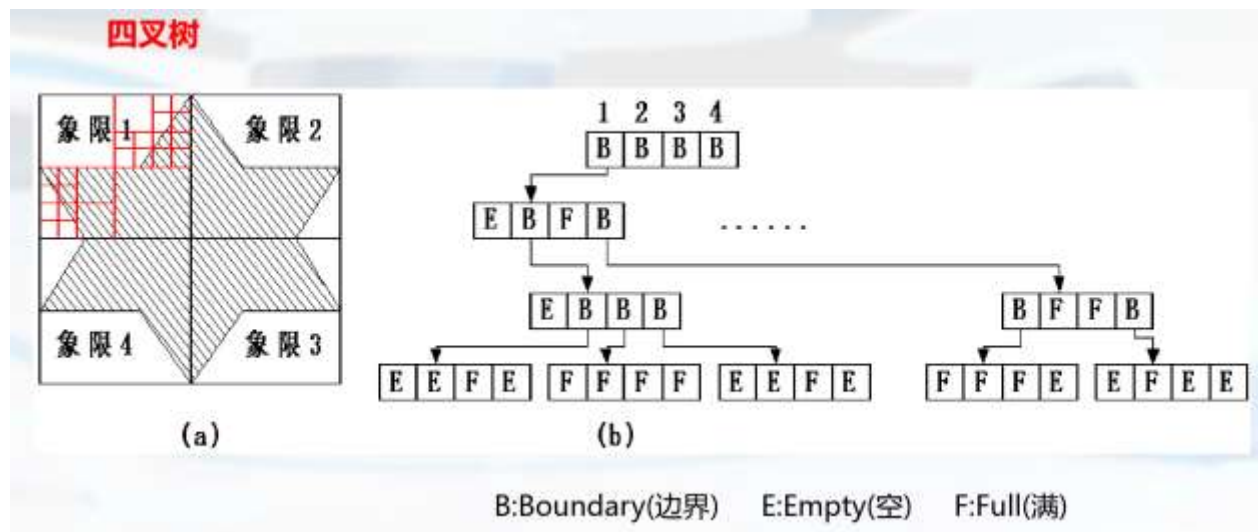
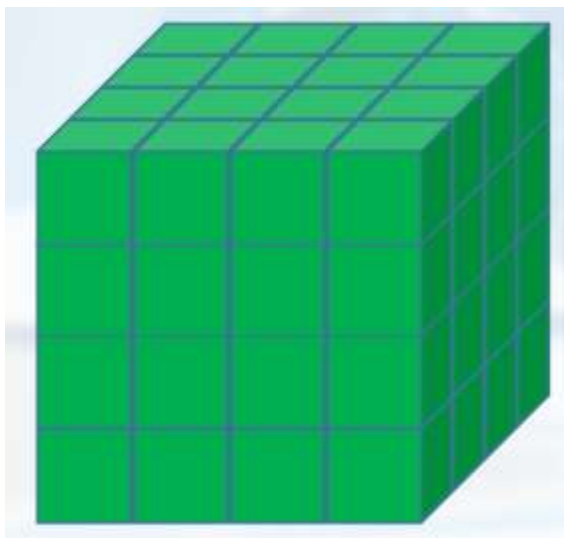
Outline

- Overview of Modeling
 - 图形的数学表示
 - 图形对象构成,基本几何元素,实体的定义
 - 图形形体分类及对应的造型技术*
- Irregular Shape Modeling/Natural Modeling 不规则形体造型/自然造型
 - Fractal Geometry分形几何
 - Particle System粒子系统
- **Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型***
 - CSG(Constructive Solid Geometry)构造实体几何
 - **SP(Space-Partitioning Representation)空间分割表示**
 - ***B-reps(Boundary Representation)边界表示****
 - Polygon Mesh*
 - Spline Curve and Surface 样条曲线曲面*

Solid Modeling -2.SP

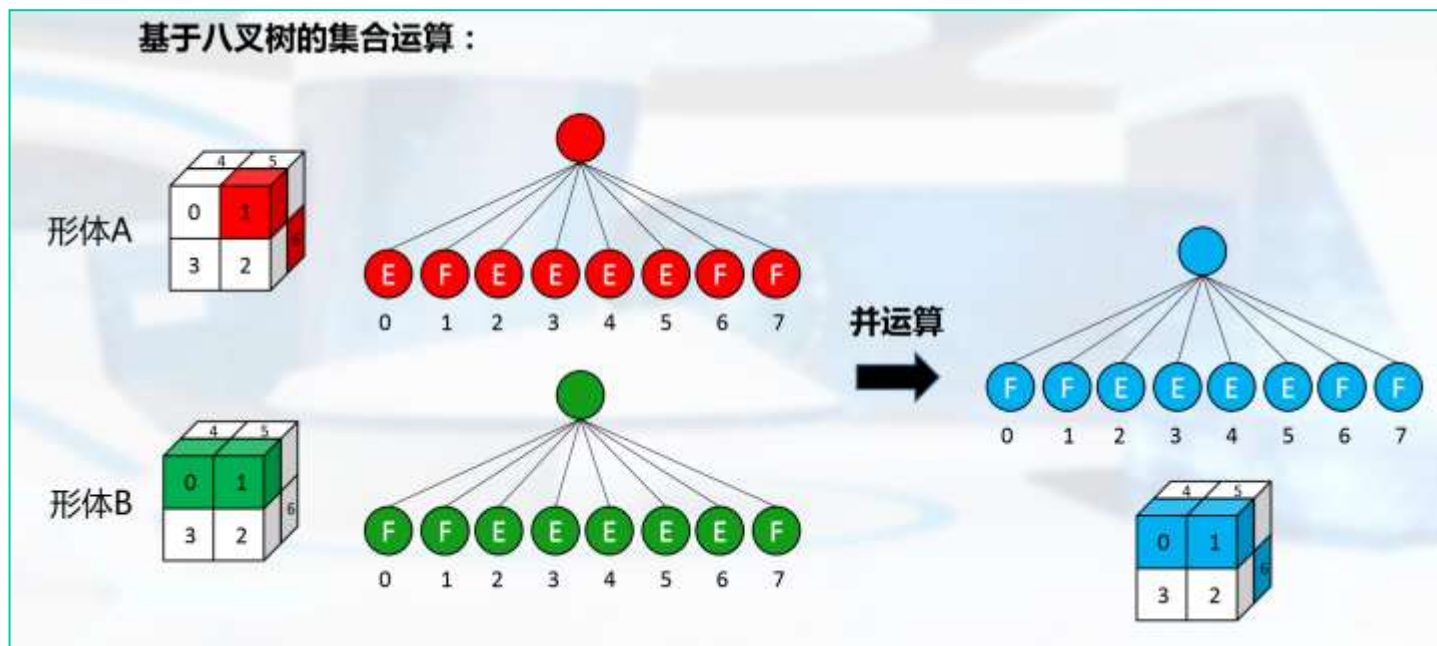
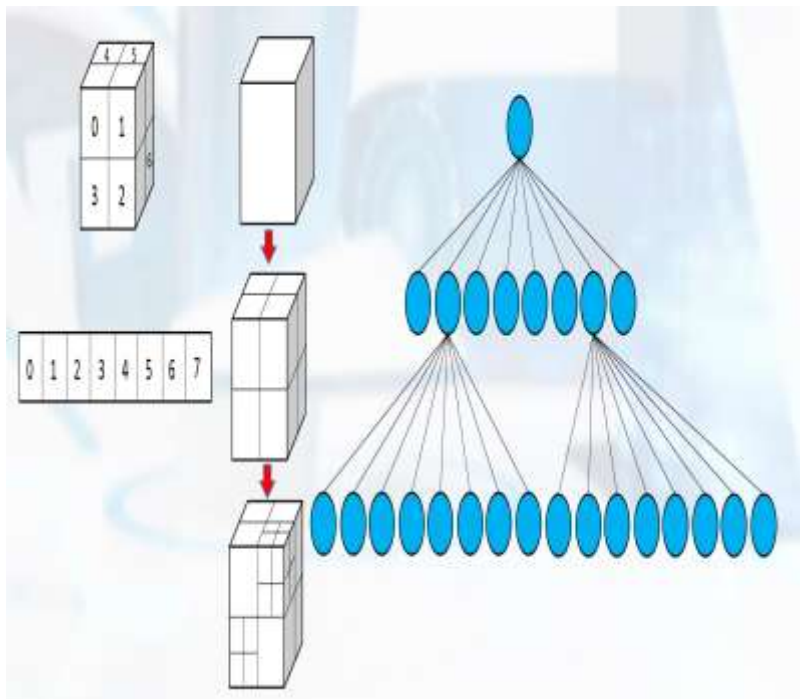
• 2.空间分割表示 (Space-Partitioning Representation)

- 用来描述物体的内部性质, 将包含一物体 的空间区域划分成一组小的、非重叠的、连续实体(通常是立方体): 空间位置枚举表示。
 - 将包含实体的空间分割为大小相同、形状规则(正方形或立方体)的体素, 然后, 以体素的集合来表示图形对象。
 - 用三维数组 $P[I][J][K]$ 表示物体, 数组中的元素与单位小立方体一一对应。



Solid Modeling -2.SP(cont.)

- 2.空间分割表示SP---八叉树法(octrees)
- 又称为分层树结构, 它对空间进行自适应划分, 采用具有层次结构的八叉树来表示实体。



注：B: Boundary, E: Empty, F: Full

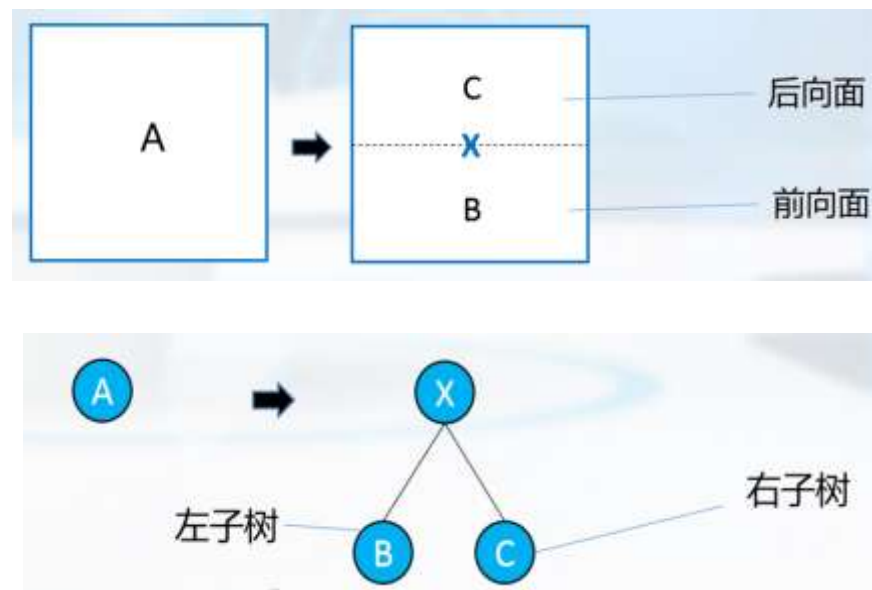
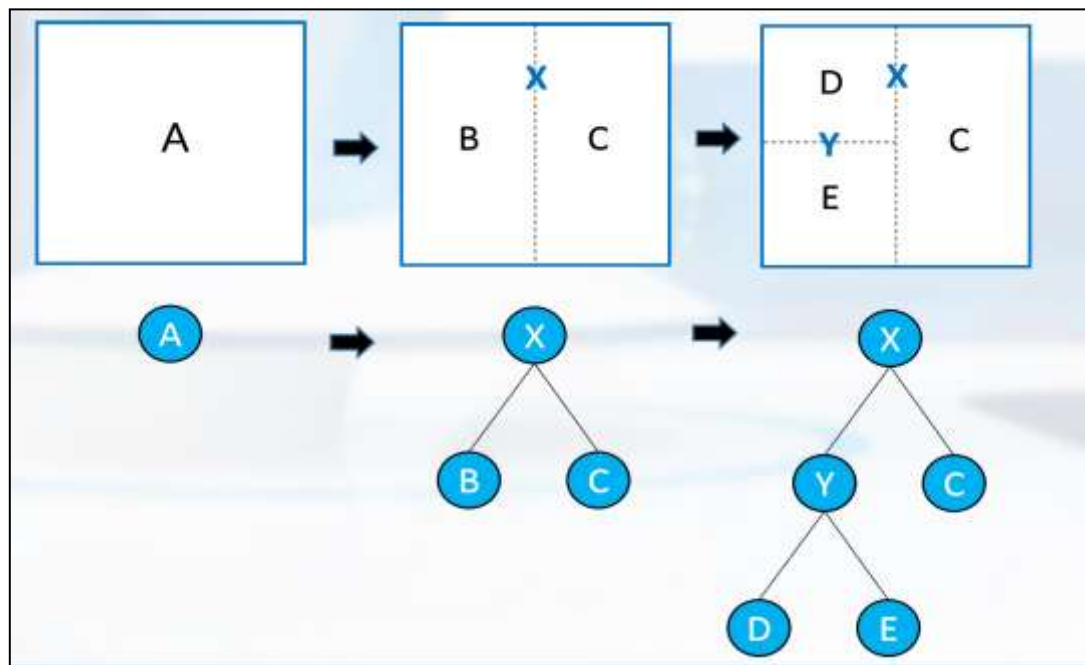
Solid Modeling -2.SP(cont.)

• 2)-空间分割表示(SP)---二叉空间分割法(BSP)

二叉空间分割(Binary Space Partitioning, BSP)每次将一实体用任一位置和任一方向的平面分为二部分来表示, 是高效的分割表示法。

自适应分割: BSP树可以减少场景树的深度,减少搜索时间;

有向超平面: BSP树可有效地识别前向面和后向面

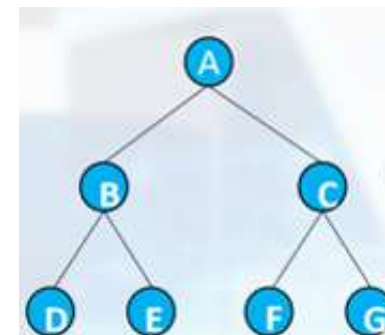
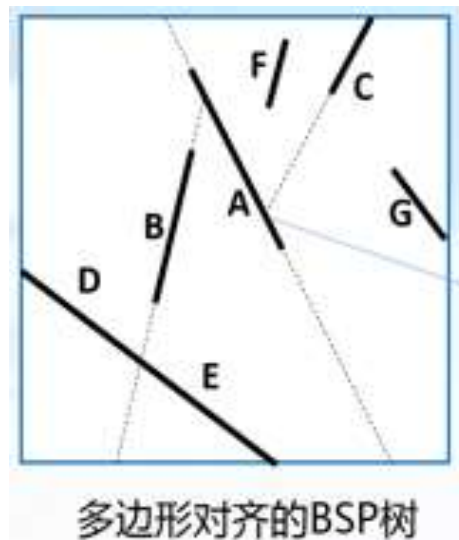
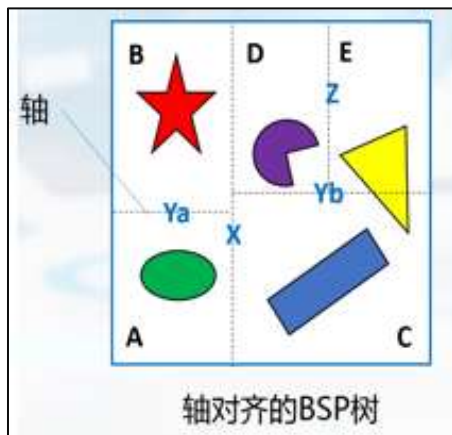


Solid Modeling -2.SP(cont.)

• 2)-空间分割表示(cont.)---二叉空间分割法(BSP)(cont.)

- BSP树的两种类型:

- 轴对齐BSP树: 只能对应粗排序.不常用。
- 多边形对齐BSP树: 对应相对视点的排序, 可与后面将要介绍的画家算法(一种消隐算法) 配合, 进行场景中物体的深度测试, 还可以作相交测试, 碰撞检测



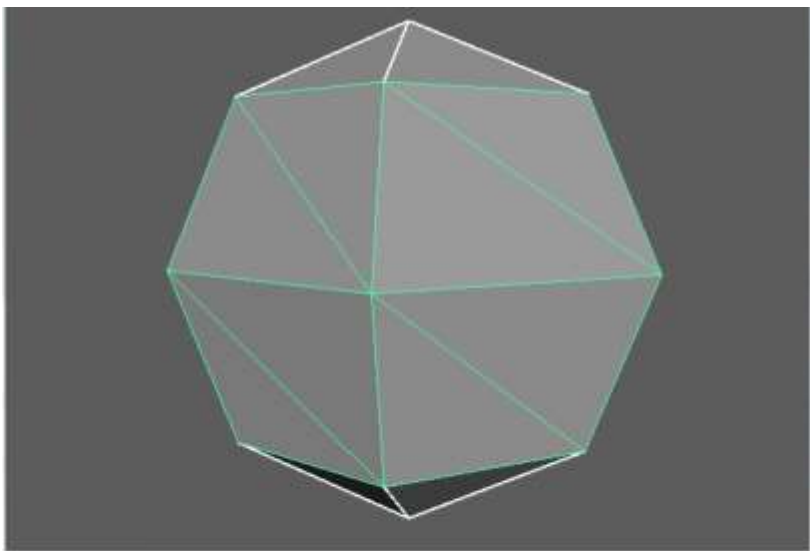


Outline

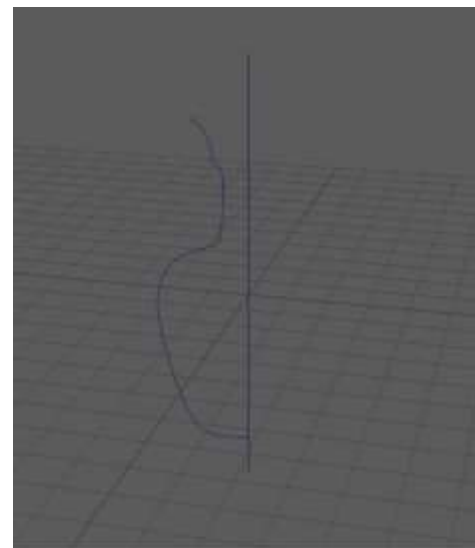
- Overview of Modeling
 - 图形的数学表示
 - 图形对象构成,基本几何元素,实体的定义
 - 图形形体分类及对应的造型技术*
- Irregular Shape Modeling/Natural Modeling 不规则形体造型/自然造型
 - Fractal Geometry分形几何
 - Particle System粒子系统
- **Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型***
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - **B-reps(Boundary Representation)边界表示***
 - **Mesh* 网格**
 - **Spline Curve and Surface 样条曲线曲面***

Solid Modeling -3.B-reps

- 3.边界表示 (Boundary Representation, B-reps)*
 - 用一组曲面 (含平面) 来描述物体! (被大量采用的造型方法)
 - 扫描表示模型
 - 多边形表面模型* (重点介绍和掌握)



多边形表面模型



扫描表示

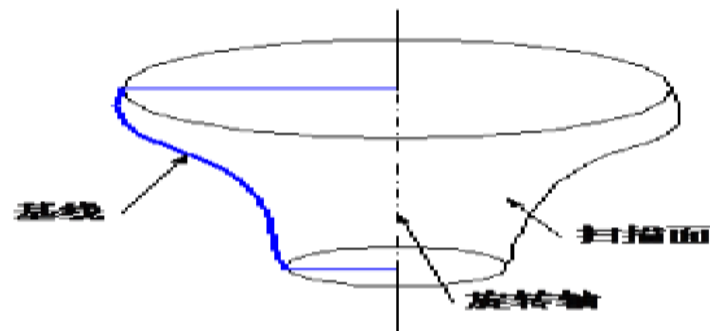


Solid Modeling -3.B-reps(cont.)

• 3. 边界表示(B-reps)---扫描表示法

- 将空间中的一个点、一条边或一个面沿某一路径扫描时，所形成的轨迹将定义一个“实体”。包含两个要素：
 - 一是作扫描运动的基本图形(基线):后面介绍样条曲线曲面生成
 - 二是扫描运动的方式, 如旋转扫描(圆形-轴对称), 非圆形扫描(平移, 螺旋), 广义扫描

➤ 旋转扫描



➤ 非圆形路径扫描

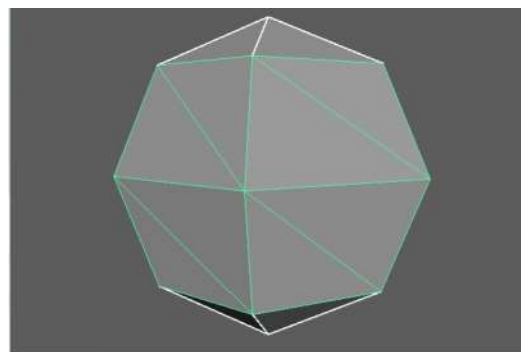


➤ 广义扫描法

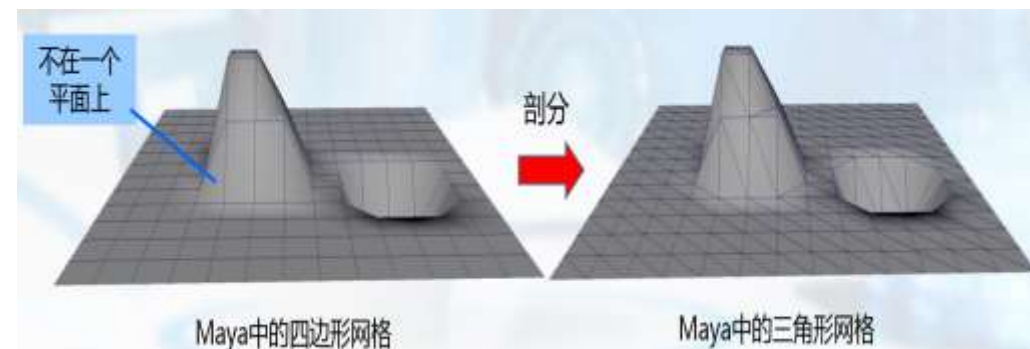


Solid Modeling -3.B-reps(cont.)

- **3. 边界表示(B-reps)---多边形表面模型Polygon Surface Model**
多面体:使用一组包围物体内部的平面多边形来描述**实体**。
三维形体的曲面边界通常用之模拟。常用三角形网格triangle mesh



多边形表面模型





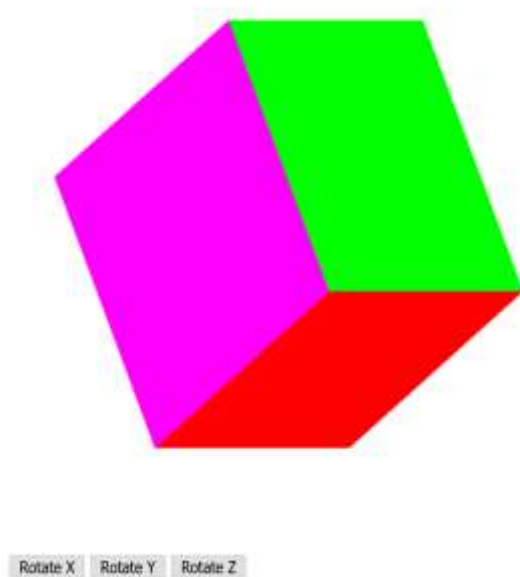
Outline

- Overview of Modeling
 - 图形的数学表示
 - 图形对象构成,基本几何元素,实体的定义
 - 图形分类及对应的造型技术*
- Irregular Shape Modeling/Natural Modeling 不规则形体造型/自然造型
 - Fractal Geometry分形几何
 - Particle System粒子系统
- **Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型***
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - **B-reps(Boundary Representation)边界表示***
 - **Mesh* 网格**
 - **Spline Curve and Surface 样条曲线曲面***

Mesh 网格(cont.)

网格mesh: 使用一组包围物体内部的平面多边形(常用三角形)来描述**实体**.

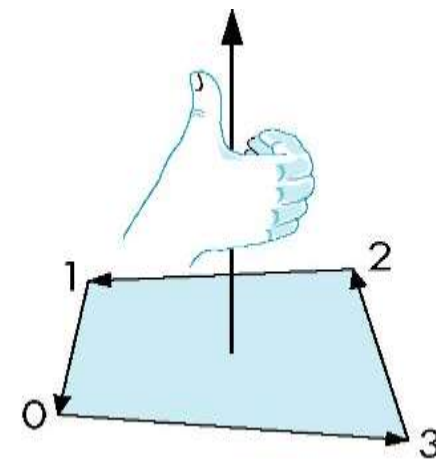
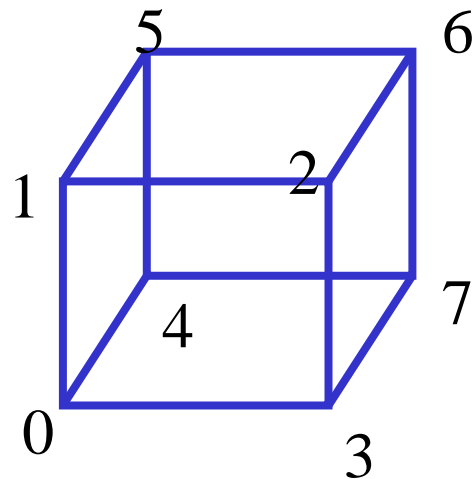
实例1:立方体规则网格 参见\Angle8E Code\04\cube.js, cube**v**.js



Mesh 网格(cont.)

➤实例1:立方体规则网格 参见\Angle8E Code\04\cube.js

```
colorCube ( )  
{  
    quad( 1, 0, 3, 2 ); //Z正-前  
    quad( 4, 5, 6, 7 ); //Z负-后  
    quad( 2, 3, 7, 6 ); //X正-右  
    quad( 5, 4, 0, 1 ); //X负-左  
    quad( 6, 5, 1, 2 ); //Y正-上  
    quad( 3, 0, 4, 7 ); //Y负-下  
}
```



• Inward and Outward Facing Polygons (内向面和外向面多边形)

- The order $\{v_4, v_5, v_6, v_7\}$ and $\{v_6, v_7, v_4, v_5\}$ are equivalent in that the same polygon will be rendered, describe *outwardly facing polygons* (外向面多边形)
- the order $\{v_5, v_4, v_7, v_6\}$ and $\{v_7, v_6, v_5, v_4\}$ are equivalent, describe *inward facing polygons* (内向面多边形)

➤ “right-hand rule” : counter-clockwise encirclement of outward-pointing normal
(右手法则: “逆时针方向” 指向该多边形的外向面的“外法向量方向”)

◆ OpenGL can treat inward and outward facing polygons differently

Mesh 网格(cont.)

实例1:立方体规则网格 参见\Angle8E Code\04\cube.js (cont.)

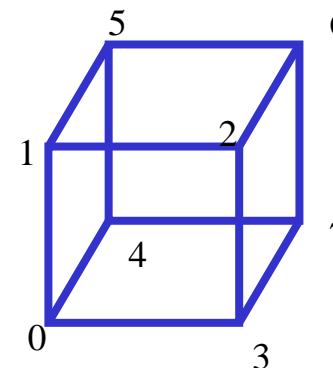
```
var vertices = [  
    vec3( -0.5, -0.5,  0.5 ),  
    vec3( -0.5,  0.5,  0.5 ),  
    vec3(  0.5,  0.5,  0.5 ),  
    vec3(  0.5, -0.5,  0.5 ),  
    vec3( -0.5, -0.5, -0.5 ),  
    vec3( -0.5,  0.5, -0.5 ),  
    vec3(  0.5,  0.5, -0.5 ),  
    vec3(  0.5, -0.5, -0.5 )  
];
```

```
var vertexColors = [  
    [ 0.0, 0.0, 0.0, 1.0 ], // black  
    [ 1.0, 0.0, 0.0, 1.0 ], // red  
    [ 1.0, 1.0, 0.0, 1.0 ], // yellow  
    [ 0.0, 1.0, 0.0, 1.0 ], // green  
    [ 0.0, 0.0, 1.0, 1.0 ], // blue  
    [ 1.0, 0.0, 1.0, 1.0 ], // magenta  
    [ 0.0, 1.0, 1.0, 1.0 ], // cyan  
    [ 1.0, 1.0, 1.0, 1.0 ] // white  
];
```

```
function colorCube( )  
{  
    quad( 1, 0, 3, 2 ); //Z正-前  
    quad( 4, 5, 6, 7 ); //Z负-后  
    quad( 2, 3, 7, 6 ); //X正-右  
    quad( 5, 4, 0, 1 ); //X负-左  
    quad( 6, 5, 1, 2 ); //Y正-上  
    quad( 3, 0, 4, 7 ); //Y负-下  
}
```

```
var quad(a, b, c, d)  
{  
    var indices = [ a, b, c, a, c, d ];  
    for ( var i = 0; i < indices.length; ++i ) {  
        points.push( vertices[indices[i]] );  
        colors.push( vertexColors[indices[i]] );  
    }  
}
```

```
gl.drawArrays( gl.TRIANGLES, 0, points.length);
```

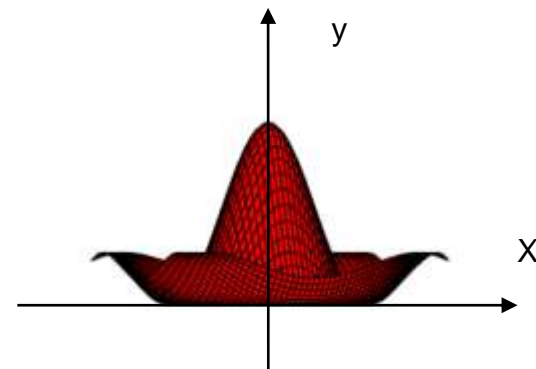
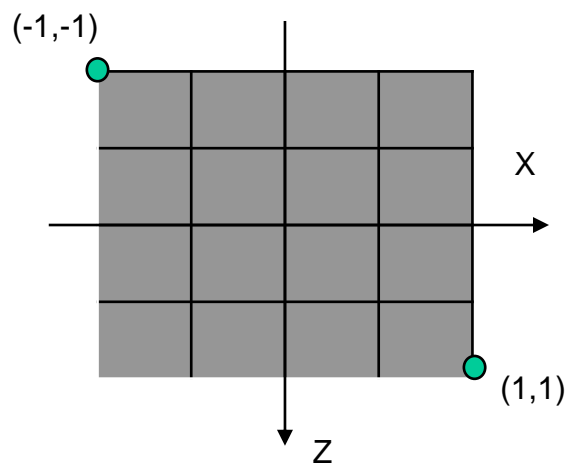
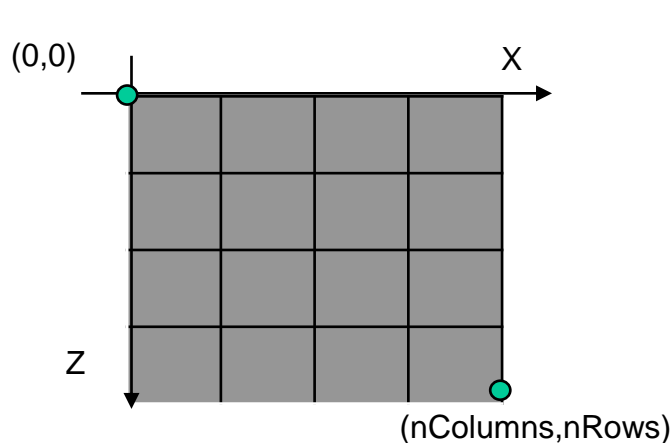


- 注意: 顶点顺序按外法向量方向
- 注意: 每个四边形生成两个三角形 将两个三角形的位置和颜色数据放入数组中

Mesh 网格(cont.)

➤ 实例2: 四边形规则网格模拟墨西哥帽子hat, 参见\Angle8E Code\05\ hat.*

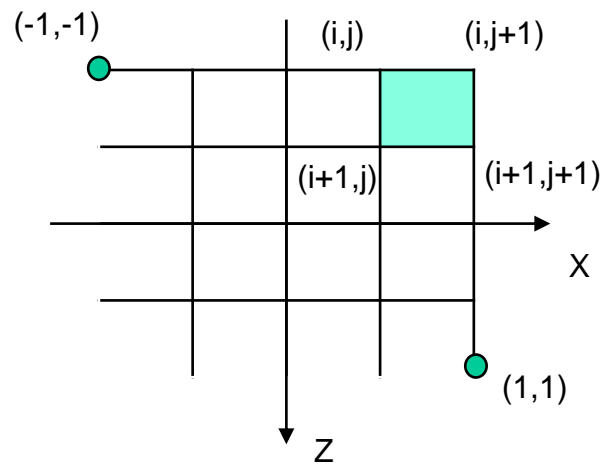
- Step1: 先将 $(0,0),(nRow,nColomn)$ 等距分割为 $(nColumns,nrows)$ 个方格,再将坐标规约到 $(-1,-1)(1,1)$
 - $Z_i = i * 2 / (nColumns - 1) - 1$ //例如下图中, $nColumns = 5$, 一般取奇数以保证图形对称
 - $X_j = j * 2 / (nRow - 1) - 1$ //例如下图中, $nRow = 5$, 一般取奇数以保证图形对称
- Step2: 然后根据 (x,z) , 计算 y 值(高程): $y = f(z,x)$, 存放到数组data中
 - $y = (\sin \pi r) / (\pi r)$, ($r \neq 0$, 非原点处, $r = \sqrt{x * x + z * z}$)
 - $y = 1$, ($r = 0$, 中心点)
- Step3: 用前两部计算出来每个顶点的 (x,y,z) , 将所有顶点按顺序放顶点缓存再发送GPU绘制三角形





The University of New Mexico

Mesh 网格(cont.)



➤ **实例2: 墨西哥帽, 参见\Angle8E Code\05\ hat.js (cont.)**

1: 先根据(z,x)的NDC坐标计算出y放data中

2: 将每个方格四个顶点位置按“逆时针”顺序, 放顶点位置数组中

3: 用三角扇(每四个顶点绘制两三角形)进行渲染

```
// 根据行列数, 确定每个顶点的X,Y,Z值。
// 行Z:  Z=(2*i/nRows-1.0);
// 列X:  X=(2*j/nColumns-1.0);
// 高度Y: 计算出r=2*PI*sqrt(Z*Z+X*X), 确定Y=sin(r)/r.
for(var i=0; i<nRows; i++) {
    var Z = (2*i/nRows-1.0);
    for(var j=0; j<nColumns; j++) {
        var X = (2*j/nColumns-1.0);

        //r范围 (-1, 1) 时, Y=sin(r)/r, 发现只有帽子中心部分显示出来
        //var r = Math.PI*Math.sqrt(X*X+Z*Z);

        //r范围 (-2pi,2pi)之间, Y=sin(r)/r, 帽子全显。
        var r = 2*Math.PI*Math.sqrt(Z*Z+X*X);

        if(r) data[i][j] = Math.sin(r)/r; else data[i][j] = 1;
    }
}
```

```
// vertex array of nRows*nColumns quadrilaterals (四边形)
// (two triangles/quad) from data
for(var i=0; i<nRows-1; i++) {
    for(var j=0; j<nColumns-1; j++) {
        pointsArray.push( vec4(2*i/nRows-1, data[i][j], 2*j/nColumns-1, 1.0));
        pointsArray.push( vec4(2*(i+1)/nRows-1, data[i+1][j], 2*j/nColumns-1, 1.0));
        pointsArray.push( vec4(2*(i+1)/nRows-1, data[i+1][j+1], 2*(j+1)/nColumns-1, 1.0));
        pointsArray.push( vec4(2*i/nRows-1, data[i][j+1], 2*(j+1)/nColumns-1, 1.0) );
    }
}
```

```
// draw each quad as two filled red triangles
// and then as two black line loops
for(var i=0; i<pointsArray.length; i+=4) {
    gl.uniform4fv(fColor, flatten(red));
    gl.drawArrays( gl.TRIANGLE_FAN, i, 4 ); //按三角扇图元显示四边形红色填充面
    gl.uniform4fv(fColor, flatten(black));
    gl.drawArrays( gl.LINE_LOOP, i, 4 ); //按线带图元显示四边形黑色边框
}
```



The University of Nottingham

Mesh 网格(cont.)

➤ 实例3: Wiresphere球体建模（四面体递归细分） ref:Angle8ECode/06/wireSphere.*

- ✓ step1.初始为在单位四面体, 有四个面, 四个顶点在单位球面上。
- ✓ Step2.取每条边上的中点P, 将每个面划分为4个小三角形面, 然后对新顶点进行归一化(即长度归约到1), 也就是把新顶点“撑”到单位球面上。
- ✓ step3.不断重复step2得到球面网格



```
function triangle(a, b, c) {
    pointsArray.push(a);
    pointsArray.push(b);
    pointsArray.push(c);
    index += 3;
}

function divideTriangle(a, b, c, count) {
    if ( count > 0 ) {

        var ab = normalize(mix( a, b, 0.5), true);
        var ac = normalize(mix( a, c, 0.5), true);
        var bc = normalize(mix( b, c, 0.5), true);

        divideTriangle( a, ab, ac, count - 1 );
        divideTriangle( ab, b, bc, count - 1 );
        divideTriangle( bc, c, ac, count - 1 );
        divideTriangle( ab, bc, ac, count - 1 );
    }
    else { // draw tetrahedron at end of recursion
        triangle( a, b, c );
    }
}

function tetrahedron(a, b, c, d, n) {
    divideTriangle(a, b, c, n);
    divideTriangle(d, c, b, n);
    divideTriangle(a, d, b, n);
    divideTriangle(a, c, d, n);
}
```

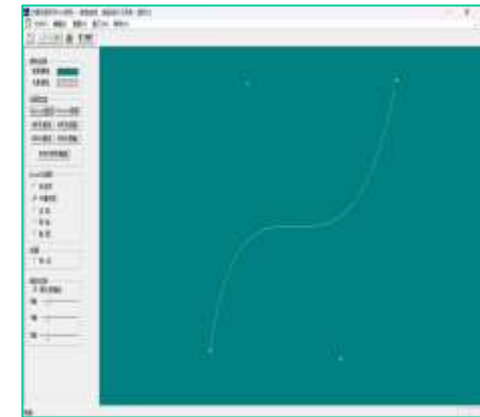
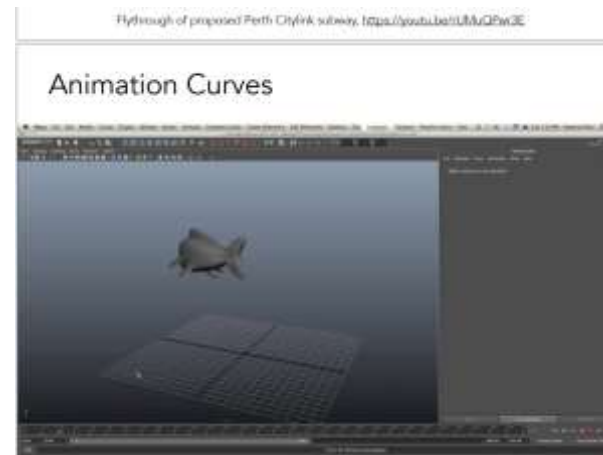


Summary

- Overview of Modeling
 - 图形的数学表示
 - 图形对象构成,基本几何元素,实体的定义
 - 图形形体分类及对应的造型技术*
- Irregular **Shape** Modeling/Natural Modeling 不规则形体造型/自然造型
 - Fractal geometry分形几何
 - Particle system微粒系统
- Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型*
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - ***B-reps(Boundary Representation)边界表示****
 - **Mesh* 网格**
 - **Spline Curve and Surface 样条曲线曲面***

Spline Curve and Surface

- 常常需要“交互”的设计“不规则”的曲线曲面
 - 相机路径，动画曲线，向量字体，造型边界表示，CAD，....



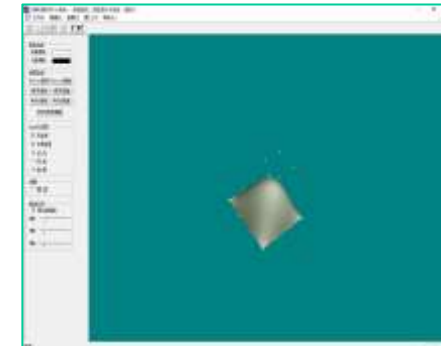
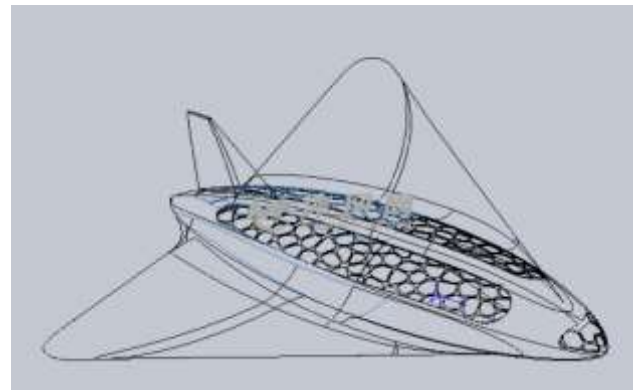
Vector Fonts

The Quick Brown
Fox Jumps Over
The Lazy Dog

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789



Baskerville font - represented as piecewise cubic Bézier curves



Spline Curve and Surface (cont.)

- **Spline** : curves designed under the control of a given set of points.

“样条”一词来源于 造船工程师 设计船体形状的“可弯曲的木条或者金属条”

- Spline

- a continuous curve constructed so as to pass through a given set of points and have a certain number of continuous derivatives.
- In short, a curve under control

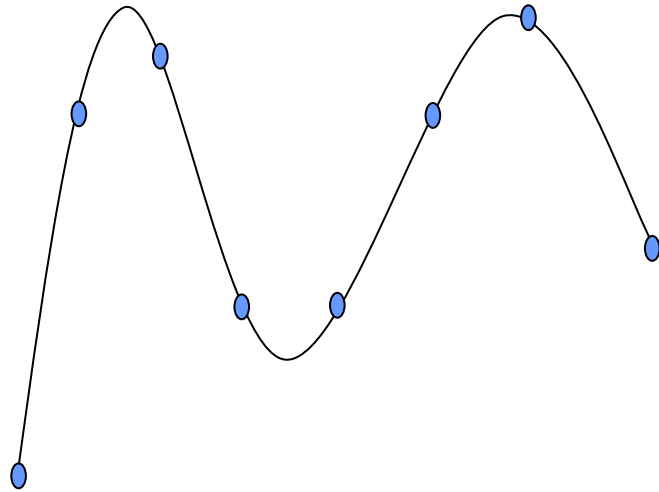


A Real Draftsman's Spline
<http://www.alatown.com/spline-history-architecture/>

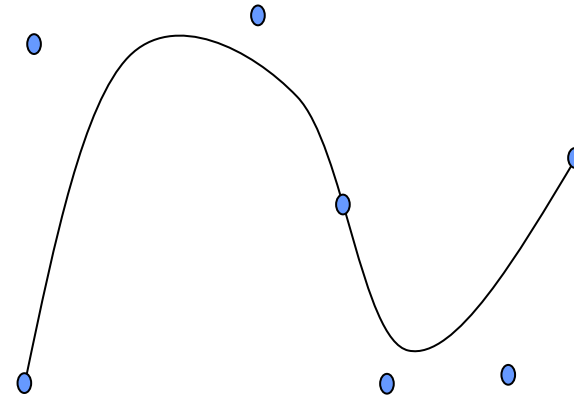
Spline Curve and Surface(cont.)

➤ the basic idea of constructing a curve is: fitting拟合

- 插值(interpolating)或逼近(approximating) 一组给定的点, 拟合出某函数(模型)表示该曲线



interpolating data point
(过已知插值点)



approximating data point
(逼近控制点)

Spline Curve and Surface(cont.)

➤ There are many ways to represent curves and surfaces:

✓ Want a representation that is Stable 稳定, Smooth 平滑, Easy to evaluate 易于求值? = Parametric Representation(参数表示) =

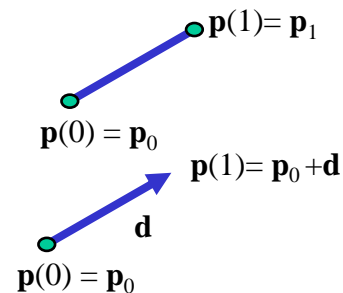
a. Parametric Lines: $p(u)$

- Line connecting two points p_0 and p_1
- normalize u to be over the interval $(0,1)$

$$p(u) = (1-u)p_0 + up_1$$

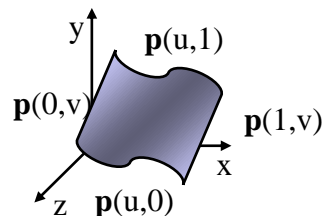
Ray from p_0 in the direction d

$$p(u) = p_0 + ud$$



b. Parametric Surfaces $p(u,v)$

➤ $p(u,v) = [x(u,v), y(u,v), z(u,v)]$



1. Parametric Planes 参数平面

point-vector form

$$p(u,v) = p_0 + uq + vr$$

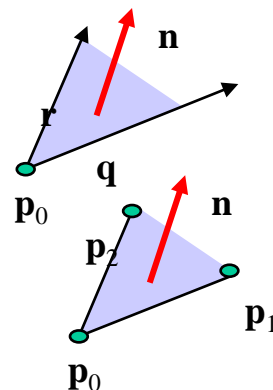
three-point form

$$q = p_1 - p_0$$

$$r = p_2 - p_0$$

$$n = q \times r$$

$$P(u,v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0)$$



2. Parametric Sphere 参数球面

$$x(\theta, \phi) = r \cos \theta \sin \phi$$

$$360 \geq \theta \geq 0$$

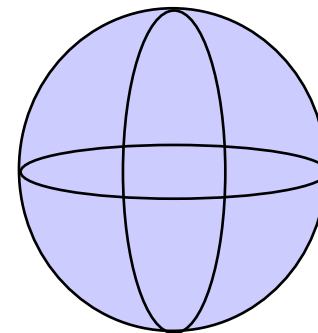
$$y(\theta, \phi) = r \sin \theta \sin \phi$$

$$180 \geq \phi \geq 0$$

$$z(\theta, \phi) = r \cos \phi$$

θ constant: circles of constant longitude

ϕ constant: circles of constant latitude



Spline Curve and Surface(cont.)

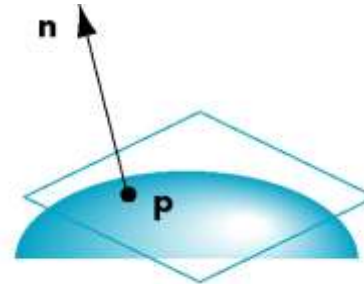
➤ Why Parametric (为什么参数化表示) ?

- easy to evaluate(易于求曲线/曲面上的点)
 - $p(u,v) = [x(u,v), y(u,v), z(u,v)]$ ($0 \leq u \leq 1, 0 \leq v \leq 1$)
- easy to differentiate (易于求导数, 求法向量等)
 - differentiate with respect to u and v to obtain the normal at any point p

$$\frac{\partial \mathbf{p}(u,v)}{\partial u} = \begin{bmatrix} \partial x(u,v) / \partial u \\ \partial y(u,v) / \partial u \\ \partial z(u,v) / \partial u \end{bmatrix}$$

$$\frac{\partial \mathbf{p}(u,v)}{\partial v} = \begin{bmatrix} \partial x(u,v) / \partial v \\ \partial y(u,v) / \partial v \\ \partial z(u,v) / \partial v \end{bmatrix}$$

$$\mathbf{n} = \frac{\partial \mathbf{p}(u,v)}{\partial u} \times \frac{\partial \mathbf{p}(u,v)}{\partial v}$$



Spline Curve and Surface(cont.)

➤ Why Polynomials(为什么用多项式函数拟合)?

- Easy to evaluate (求值)
- Continuous(连续) and differentiable(微分) everywhere

$$p(u) = \sum_{k=0}^L c_k u^k$$

$$x(u) = \sum_{i=0}^N c_{xi} u^i$$

$$y(u) = \sum_{j=0}^M c_{yj} u^j$$

$$z(u) = \sum_{k=0}^L c_{zk} u^k$$

- After normalizing u (规范化 U), each curve is written $\mathbf{p}(u)=[x(u), y(u), z(u)]^T$, ($1 \geq u \geq 0$)

Spline Curve and Surface(cont.)

➤ Why PC Curve is Cubic (为什么构造三次多项式函数拟合)？

$$p(u) = \sum_{k=0}^3 c_k u^k$$

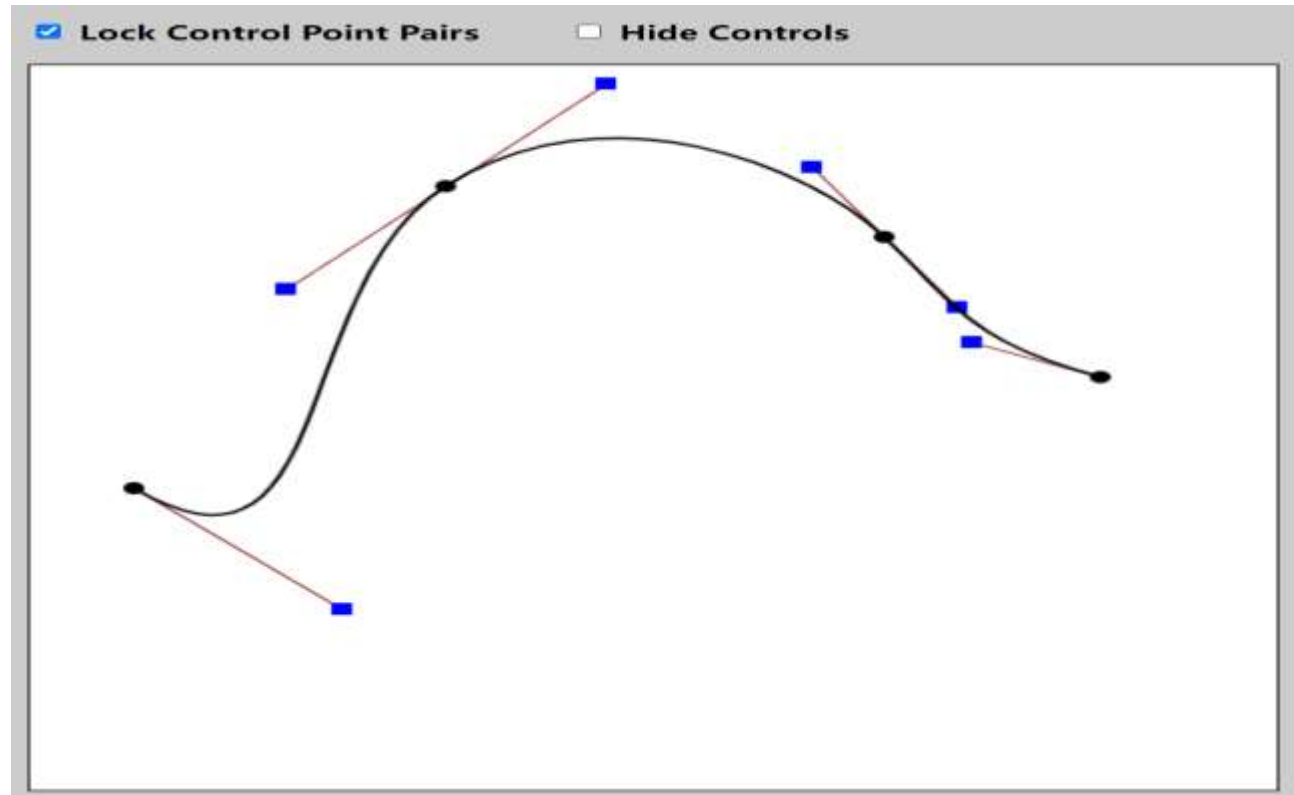
$$p(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 c_{ij} u^i v^j$$

- n=3, gives balance between ease of **evaluation** and **flexibility** in design
 - 拟合曲线的次数过高: 过拟合
 - 拟合曲线的次数太低: 欠拟合

Spline Curve and Surface (cont.)

➤ Segments and Continuity (分段与连续)

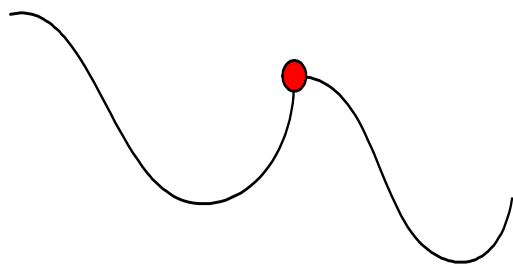
Bezier Curve Demos: <https://math.hws.edu/eck/cs424/notes2013/canvas/bezier.html>



Spline Curve and Surface(cont.)

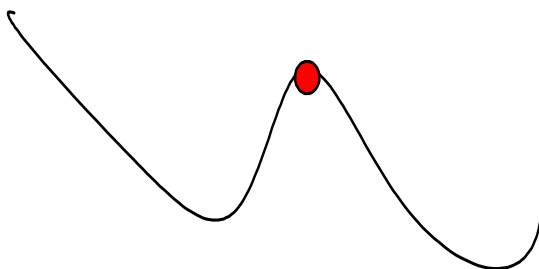
➤ Segments and Continuity(cont.)

- 0阶参数连续性 the parametric continuity C_0 : 两个相邻曲线段端点处重合。
- 1阶参数连续性 the parametric continuity C_1 : 两个相邻曲线段交点处重合, 并具有相同的一阶导数。一阶连续性对数字化绘图及一些设计应用已经足够。
- 2阶参数连续性 the parametric continuity C_2 : 两个相邻曲线段交点处重合, 并具有相同的一阶和二阶导数。二阶连续性对电影中的动画路径和精密CAD需求有用。



(a) 0阶连续性

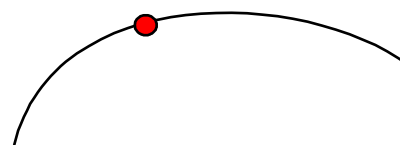
$$p_i(t_{i1}) = p_{(i+1)}(t_{(i+1)0})$$



(b) 1阶连续性

$$p_i(t_{i1}) = p_{(i+1)}(t_{(i+1)0})$$

$$\text{且 } p'_i(t_{i1}) = p'_{(i+1)}(t_{(i+1)0})$$



(c) 2阶连续性

$$p_i(t_{i1}) = p_{(i+1)}(t_{(i+1)0})$$

$$\text{且 } p'_i(t_{i1}) = p'_{(i+1)}(t_{(i+1)0})$$

$$\text{且 } p''_i(t_{i1}) = p''_{(i+1)}(t_{(i+1)0})$$

Spline Curve and Surface (cont.)

➤ Segments and Continuity (cont.)

- 0阶几何连续性, G^0

- 指相邻曲线段在连接点处位置重合。等价于 C^0

- 1阶几何连续性, G^1

- 指相邻曲线段在连接点处位置重合, 一阶成比例 (即方向相同, 大小不同)

- 2阶几何连续性, G^2

- 指相邻曲线段在连接点处位置重合, 一阶和二阶导数均成比例。

➤ C^n 连续保证 G^n 连续, 但反之不然。

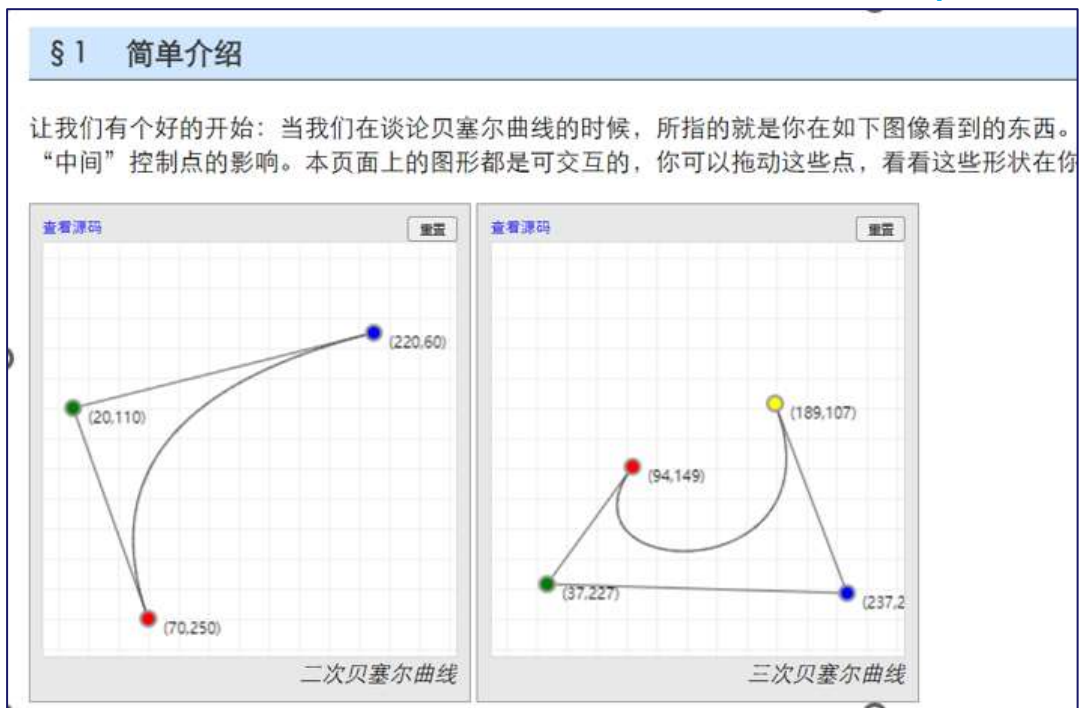
➤ C^n 连续的条件比 G^n 连续的条件要苛刻

Spline Curve and Surface(cont.)

- Bezier Curve 贝塞尔曲线:

- 常用的采用Approximation (逼近)方法构造的样条曲线

- 参“贝塞尔曲线入门”<https://pomax.github.io/bezierinfo/zh-CN/index.html#matrixsplit>



Spline Curve and Surface(cont.)

➤ Bezier Curve 贝塞尔曲线 (cont.)

Bezier曲线: 用一组基函数,对一组已知控制点进行线性组合的结果

$$p(t) = \sum_{k=0}^n P_k BEN_{k,n}(t) \quad t \in [0, 1]$$



Bernstein基函数BEN(t)具有如下形式:

$$BEN_{k,n}(t) = \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k} = C_n^k t^k (1-t)^{n-k}$$
$$k = 0, 1, \dots, n$$



$$C_n^k = \frac{n!}{k!(n-k)!} = \frac{n-k+1}{n} C_n^{k-1} \quad n \geq k$$

注意: 当 $k=0$, $t=0$ 时, $t^k=1$, $k!=1$ (即 $0^0=1, 0!=1$)

Spline Curve and Surface(cont.)

➤ Cubic Bezier Curve三次贝塞尔曲线

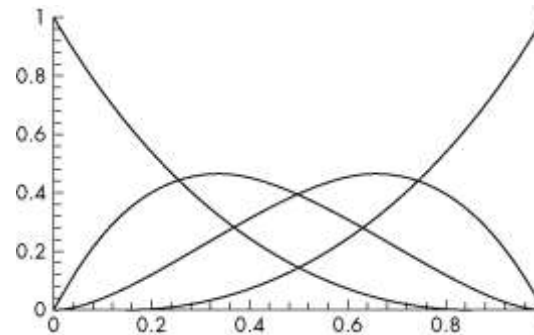
$$p(t) = \sum_{k=0}^3 P_k BEN_{k,3}(t)$$

$$= (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3 \quad t \in [0,1]$$

$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \quad t \in [0,1]$$

$$= T \cdot M_{be} \cdot G_{be}$$

$$\mathbf{b}(u) = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{bmatrix}$$



Note that all zeros are at 0 and 1 which forces the functions to be smooth over (0,1)

Spline Curve and Surface (cont.)

➤ Bezier Surface 贝塞尔曲面

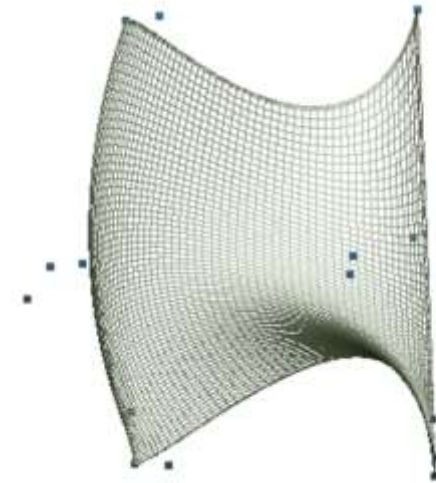
$$p(u, v) = \sum_{i=0}^m \sum_{j=0}^n P_{i,j} BEN_{i,m}(u) BEN_{j,n}(v)$$

$$(u, v) \in [0, 1] \times [0, 1]$$

$BEN_{i,m}(u)$ 与 $BEN_{j,n}(v)$ 是 **Bernstein** 基函数:

$$BEN_{i,m}(u) = C_m^i \cdot u^i \cdot (1-u)^{m-i}$$

$$BEN_{j,n}(v) = C_n^j \cdot v^j \cdot (1-v)^{n-j}$$



Spline Curve and Surface(cont.)

➤ Double Cubic Bezier Surface 双三次贝塞尔曲面

- 双三次Bezier曲面($m=n=3$), $(m+1)(n+1)=16$ 个控制点
Using same data array $\mathbf{P}=[p_{ij}]$ as with interpolating form

$$p(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 P_{i,j} BEN_{i,3}(u) BEN_{j,3}(v)$$

$$(u, v) \in [0, 1] \times [0, 1]$$

$$p(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 b_i(u) b_j(v) p_{ij} = u^T \mathbf{M}_B \mathbf{P} \mathbf{M}_B^T v$$

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \quad V = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$$

Patch lies in
convex hull

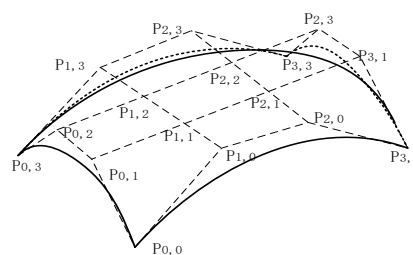
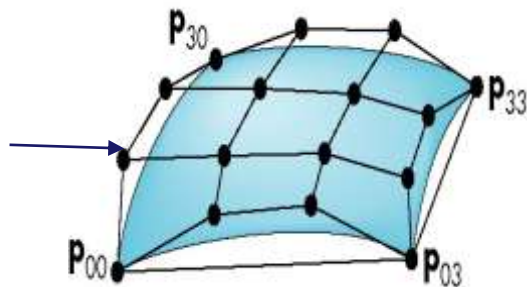


图 双三次Bezier曲面及其控制网格

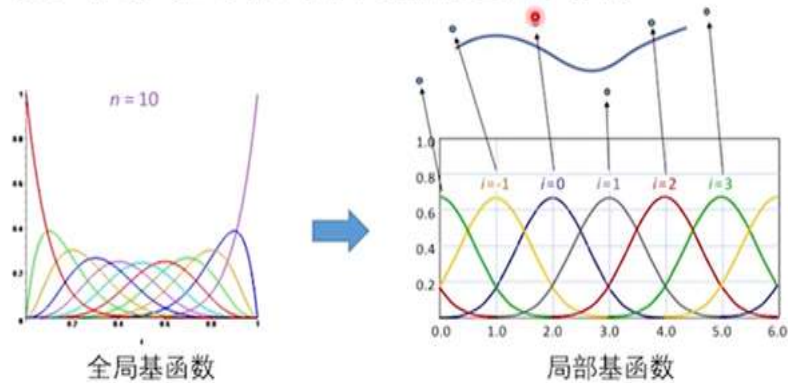
$$M_{be} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$P = \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} & P_{0,3} \\ P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\ P_{2,0} & P_{2,1} & P_{2,2} & P_{2,3} \\ P_{3,0} & P_{3,1} & P_{3,2} & P_{3,3} \end{bmatrix}$$

Spline Curve and Surface(cont.)

- 其它：B-SPLINE(B样条), NURBS(非均匀有理B样条)

- Bézier曲线、RBF函数：每个控制点上的权系数函数都是全局（定义在整个定义域）的
- B样条曲线：每个控制点上的权系数函数是局部定义的（定义在其参数节点附近的支集）



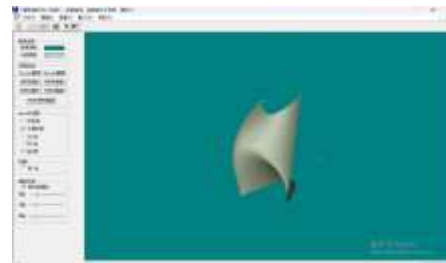
NURBS曲线

产品设计的工业标准

- NURBS曲线/曲面表达是当前的工业标准
 - 工业CAD软件的基本表达形式
 - 各种CAD系统的数据交换标准



- 3D建模软件：
 - 工业设计：AutoCAD, CATIA, SolidWorks, Rhino, ...
 - 动画设计：3DS Max, Maya, SoftImage, Cinema 4D, ...

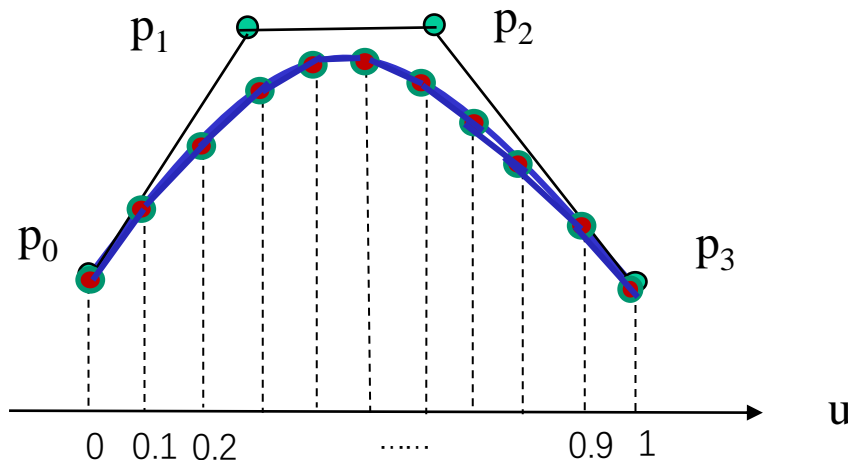


Spline Curve and Surface (cont.)

➤ Bezier Implementation Algorithm

1) 直接公式求值

思路: 通过等间距的一组参数值计算曲线上的点, 然后连接这些点, 使用折线来近似表示曲线。特点: 计算量大。



$$p(t) = \sum_{k=0}^3 P_k BEN_{k,3}(t)$$
$$= (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3 \quad t \in [0,1]$$

如: 等间距划分参数区间 $[0,1]$ $t=[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$
对每个参数取值 t , 计算基函数和控制点 P 的点乘得曲线上的点 (红点)

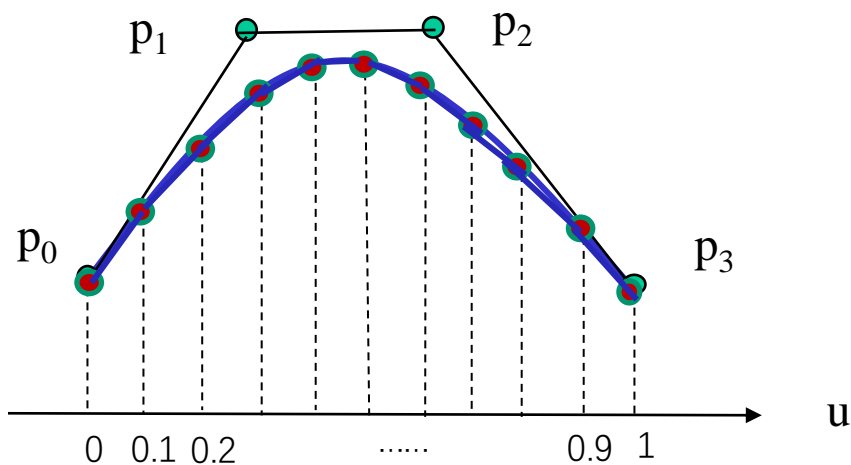
Spline Curve and Surface(cont.)

➤ Bezier Implementation Algorithm(cont.)

2) 多项式求值法 polynomial evaluation (Horner's method)

思路:通过等间距的一组参数值, 根据代数公式计算曲线上的点, 然后连接这些点, 使用折线来近似表示曲线。

特点:计算公式有优化, 但是只适合均匀网络, 容易累积数值误差



$$p(u) = c_0 + u(c_1 + u(c_2 + uc_3))$$

//3 multiplications for cubic

如: 等间距划分参数区间[0,1], $t = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$

先根据公式计算得到曲线上的点 (红点表示), 然后每两红点之间以直代曲, 绘制出曲线 (绿线表示)

Spline Curve and Surface(cont.)

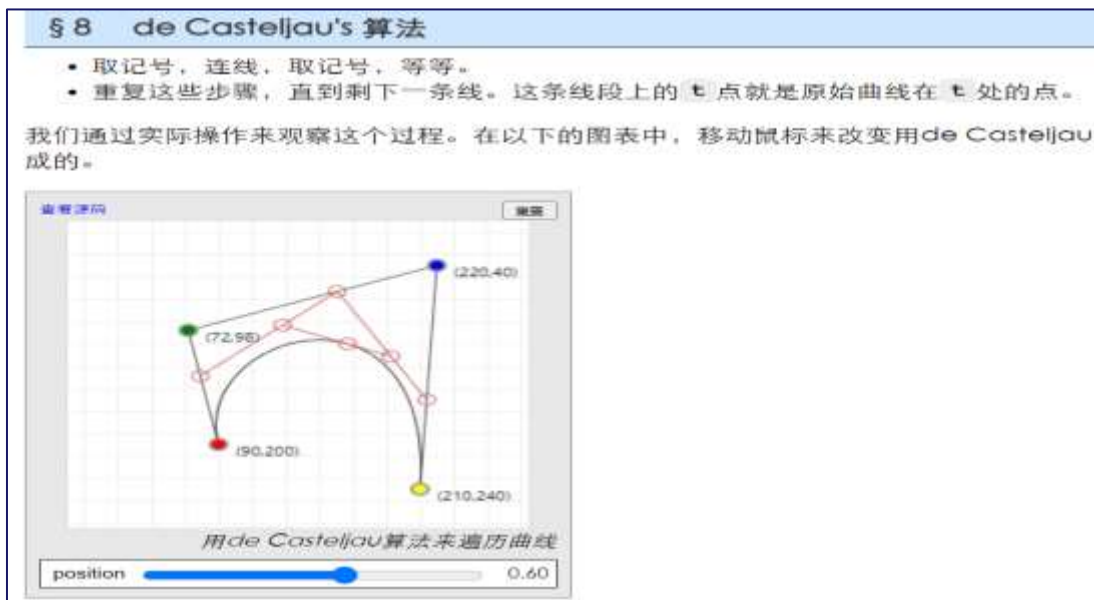
➤ Bezier Implementation Algorithm(cont.)

3) Cubic Bezier curve –De Casteljau(德•卡斯特里奥) Algorithm

参“贝塞尔曲线入门”<https://pomax.github.io/bezierinfo/zh-CN/index.html#matrixsplit>

思路：

对参数 t 取某值 t_1 ，对每两个控制点进行线性插值求得新控制点，并用新控制点构成的控制点集合继续进行插值，直到得到一个控制点，该点就是曲线上 $P(t_1)$ 值。



Paul de Casteljau
b. 1930

Spline Curve and Surface(cont.)



➤ Bezier Implementation Algorithm(cont.)

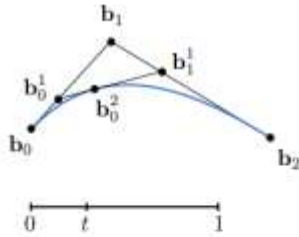
3) Cubic Bezier curve –De Casteljau Algorithm (cont.)

• 根据如下的三个控制点的递归计算过程，推导出来的就是贝塞尔公式！

- 参见 *GAMES101 Lecture11 geometry2*

Bézier Curve – Algebraic Formula

Example: quadratic Bézier curve from three points



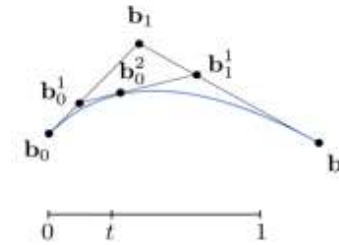
$$b_0^1(t) = (1-t)b_0 + tb_1$$

$$b_1^1(t) = (1-t)b_1 + tb_2$$

$$b_0^2(t) = (1-t)b_0^1 + tb_1^1$$

$$b_0^2(t) = (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2$$

Run the same algorithm for every t in $[0,1]$



$$\begin{aligned}
 p(t) &= \sum_{k=0}^2 P_k BEN_{k,2}(t) \\
 &= \boxed{(1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2} \quad t \in [0,1] \\
 &= (P_2 - 2P_1 + P_0)t^2 + 2(P_1 - P_0)t + P_0
 \end{aligned}$$

Spline Curve and Surface(cont.)

➤ Bezier Implementation Algorithm(cont.)

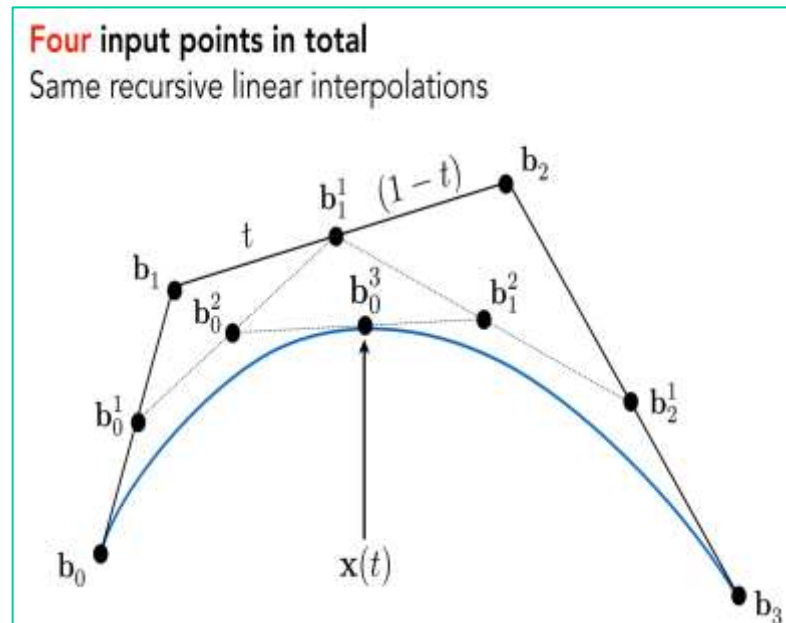
3) Cubic Bezier curve –De Casteljau Algorithm (cont.)

- 四个控制点绘制三次贝塞尔曲线示例如下：

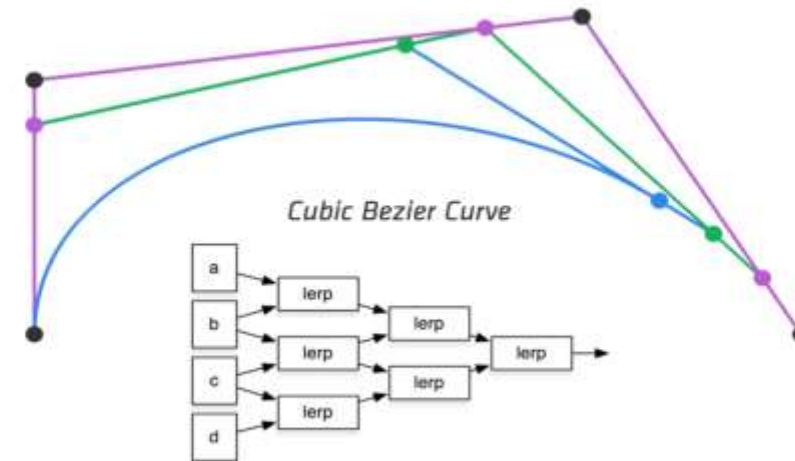
- 参见 *GAMES101 Lecture11 geometry2*



Paul de Casteljau
b. 1930



Visualizing de Casteljau Algorithm



Animation: Steven Wittens, Making Things with Maths, <http://acko.net>

Spline Curve and Surface(cont.)

➤ Bezier Implementation Algorithm(cont.)

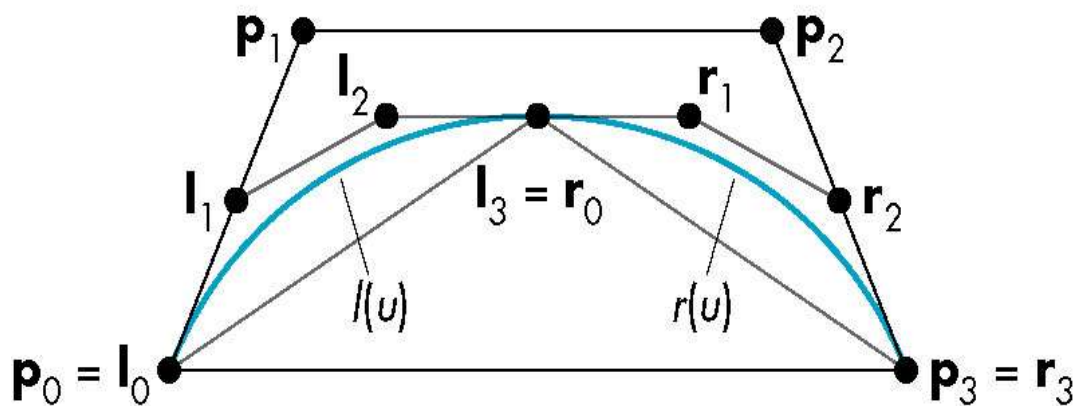
4) recursive subdivision 递归细分(“分割法”)

参“贝塞尔曲线入门”, <https://pomax.github.io/bezierinfo/zh-CN/index.html#matrixsplit>

特点: use the convex hull(凸包) property of Bezier curves to obtain **an efficient recursive method**

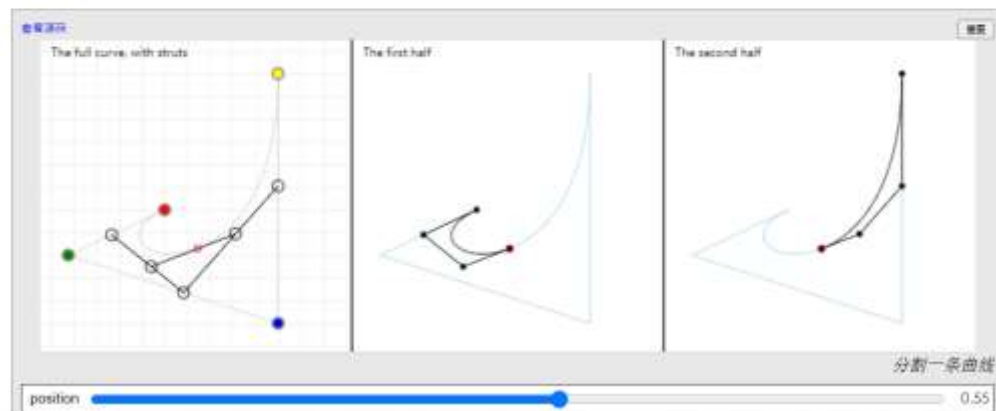
原理: 变差缩减性 the *variation diminishing property* : 将曲线 $P(u)$ 分割成两个独立的多项式 $l(u), r(u)$, $l(u)$ 和 $r(u)$ 的凸包都仍将位于 $P(u)$ 的凸包内。

思路: 不断将曲线一分为二, 重新计算所分更短两段的控制点, 当小段的控制点凸包“充分靠近”曲线时, 就用首末两点线段表示该曲线段。(保证绘制出线段而不是点, 与3) De Casteljau Algorithm方法的主要区别)



§ 10 分割曲线

使用 de Casteljau 算法我们也可以将一条贝塞尔曲线分割成两条更小的曲线, 二者拼接起来即可形成原来的曲线。当采法时, 该过程会给我们在 u 点分割曲线的所有点: 一条曲线包含该曲线上点之前的所有点, 另一条曲线包含该曲线上点之后的所有点。





The University of New Mexico

Spline Curve and Surface(cont.)

➤ Bezier Implementation Algorithm(cont.)

4) recursive subdivision 递归细分(“分割法”)

➤ 初始, 将 $p(u)$ 的凸包控制点(P_0, P_1, P_2, P_3)分成两组控制点(l_0, l_1, l_2, l_3), (r_0, r_1, r_2, r_3)

➤ 判断左边组控制点的扁平性($l_1 l_2$ 到 $l_0 l_3$ 的距离)或 $l_0 l_3$ 足够小)

如果距离足够小, 则可用线段($l_0 l_3$)代替该段曲线; 否则, 可将控制点组分成两部分并且继续判断两个新凸包的扁平性(递归调用)

➤ 判断右边组控制点的扁平性($r_1 r_2$ 到 $r_0 r_3$ 的距离)或 $r_0 r_3$ 足够小)

如果距离足够小, 则可用线段($r_0 r_3$)代替该段曲线; 否则, 可将控制点组分成两部分并且继续判断两个新凸包的扁平性(递归调用)

Start with Bezier equations $p(u) = \mathbf{u}^T \mathbf{M}_B \mathbf{p}$

$l(u)$ must interpolate $p(0)$ and $p(1/2)$

$$l(0) = l_0 = p_0$$

$$l(1) = l_3 = p(1/2) = 1/8(p_0 + 3p_1 + 3p_2 + p_3)$$

Matching slopes, taking into account that $l(u)$ and $r(u)$ only go over half the distance as $p(u)$

$$l'(0) = 3(l_1 - l_0) = p'(0) = 3/2(p_1 - p_0)$$

$$l'(1) = 3(l_3 - l_2) = p'(1/2) = 3/8(-p_0 - p_1 + p_2 + p_3)$$

➤ 四个方程联立求解可得到 $l(u)$ 的控制点(l_0, l_1, l_2, l_3)的计算公式,

➤ 同理, 对称得到 $r(u)$ 的控制点值(r_0, r_1, r_2, r_3)的计算公式,

Requires only shifts and adds!

$$l_0 = p_0$$

$$r_3 = p_3$$

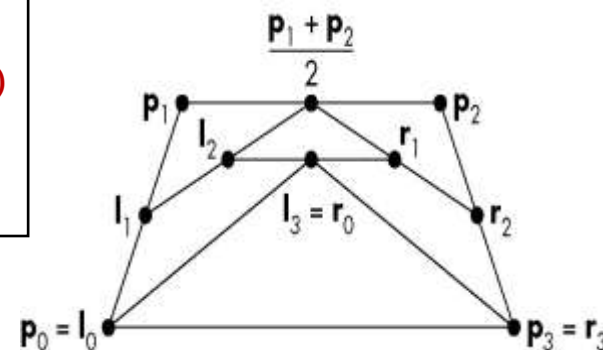
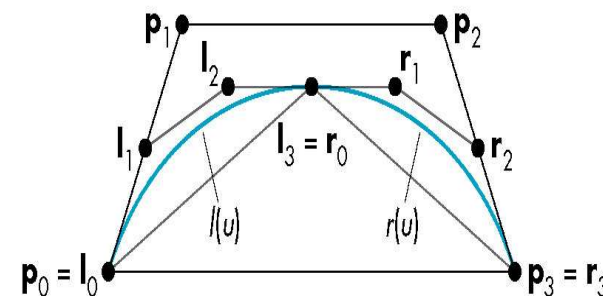
$$l_1 = 1/2(p_0 + p_1)$$

$$r_1 = 1/2(p_2 + p_3)$$

$$l_2 = 1/2(l_1 + 1/2(p_1 + p_2))$$

$$r_1 = 1/2(r_2 + 1/2(p_1 + p_2))$$

$$l_3 = r_0 = 1/2(l_2 + r_1)$$



Spline Curve and Surface(cont.)

➤ WebGL Example: Utah Teapot 犹他茶壶(略)

- Most famous data set in computer graphics
- Widely available as a list of 306 3D vertices and the indices that define 32 Bezier patches

- **Ref: *//Angle8ECode/11/teapot.****

- vertices.js: three versions of the data vertex data
- patches.js: teapot patch data



- teapot1: wire frame teapot by recursive subdivision of Bezier curves
- teapot2: wire frame teapot using polynomial evaluation





Summary

- **Overview of Modeling/Geometry**
 - 图形的数学表示
 - 图形对象构成, 基本几何元素, 实体的定义,
 - 造型技术
- **Irregular Shape Modeling/Natural Modeling 不规则形体造型/自然造型**
 - Fractal geometry分形几何
 - Particle system粒子系统
- **Regular Shape Modeling/Solid Modeling 规则形体造型/实体造型***
 - CSG(Constructive Solid Geometry)构造实体几何
 - SP(Space-Partitioning Representation)空间分割表示
 - B-reps(Boundary Representation)边界表示*
 - Mesh* 网格 (例子:立方体, 墨西哥帽, 球体)
 - Spline Curve and Surface 样条曲线曲面*(例:贝塞尔曲线曲面)