计算机图形学A

1. 概述

计算机图形学——研究如何利用计算机生成、处理和显示图形的一门学科

图形——形成视觉印象中的客观或想象中的对象。

基于线条信息的表示，如工程图、等高线图

明暗图，真实感图形

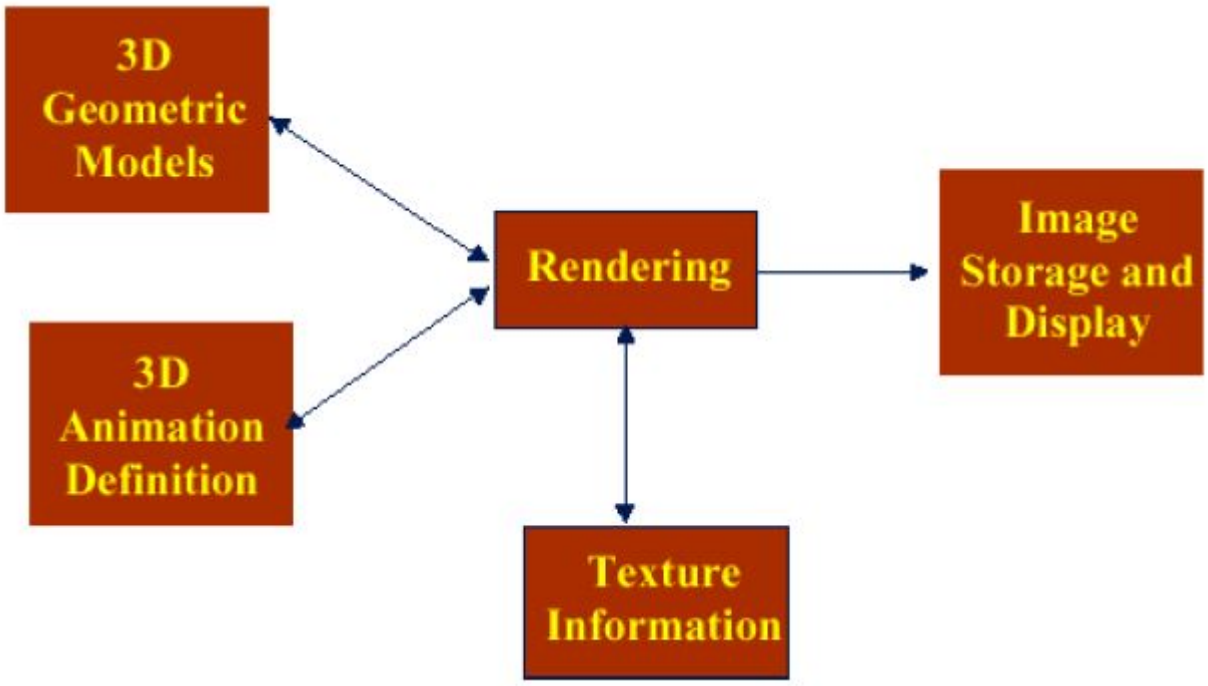
构成图形的要素：刻画形状的点线面体等几何要素；反映物体表面属性或材质的灰度颜色等非几何要素

图形的表示方法

点阵法：枚举出图形中所有点，强调图形由点组成及其点的属性（如颜色），如像素图或图像

参数法：由图形的形状参数（方程或分析表达式的系数，线段端点坐标等）和属性参数（颜色，线型等）来表示图形

图形学处理过程：三维几何建模+三维动作捕捉+纹理信息→着色绘制（透视投影+图像生成）→图像储存和显示



三维图形的基本问题——二维屏幕上如何显示三维物体，投影；如何表示三维物体，利用各种用于形体表示的理论、模型和方法；如何反应遮挡关系，消除隐藏面和隐藏线；如何产生真实感图形，建立光照明模型，开发真实感图形绘制方法

三维图形的显示流程

图形(Graphics)与图像(Image)——图像单纯指计算机内按位图形式存在的灰度或彩色信息；图形含有几何属性，或者说更强调场景的几何表示，是由场景的几何模型和景物的物理属性共同组成的。

2. 图形硬件设备与系统

显示设备发展

1950年，MIT旋风一号图形显示器，类似示波器显示简单图形

50年代末，MIT林肯实验室，旋风型上开发SAGE空中防御系统，用激光笔在屏幕上指点，交互图形学诞生

1963，Ivan Sutherland，SKETCHPAD，一个人机通信的图形系统

60年代中期，随机扫描显示器，较高分辨率和对比度，动态性能良好；闪烁且需要刷新，需要缓冲存储器和高速处理器来帮助刷新

60年代后期，储存管式显示器，不需要刷新和缓存，价格低，分辨率高；不能动态显示图形，不适合交互

80年代，光栅显示器

90年代，液晶显示器

GPU，头盔，立体显示器，显示阵列...

图形绘制算法

1960s，解决可见性问题——隐藏线消除算法，隐藏面消除算法，可见性排序算法

1970s，光栅化图形学——漫反射光照，镜面反射光照，曲面表面，纹理映射，Z-buffer隐藏面消除算法，反混淆

1980s早期，全局光照明——光线追踪法，辐射度方法，绘制方程

1980s后期，照片真实感绘制技术——层次绘制树，高级绘制语言，真实感绘制工具

1990s早期，非照片真实感绘制——体绘制，印象绘画风格，自动钢笔画插图，美术绘制

光栅扫描显示系统：电子束按照固定的扫描顺序进行扫描N条扫描线，每条扫描线M个像素，因此显示器分辨率为$\boldsymbol{M} \times \boldsymbol{N}$。

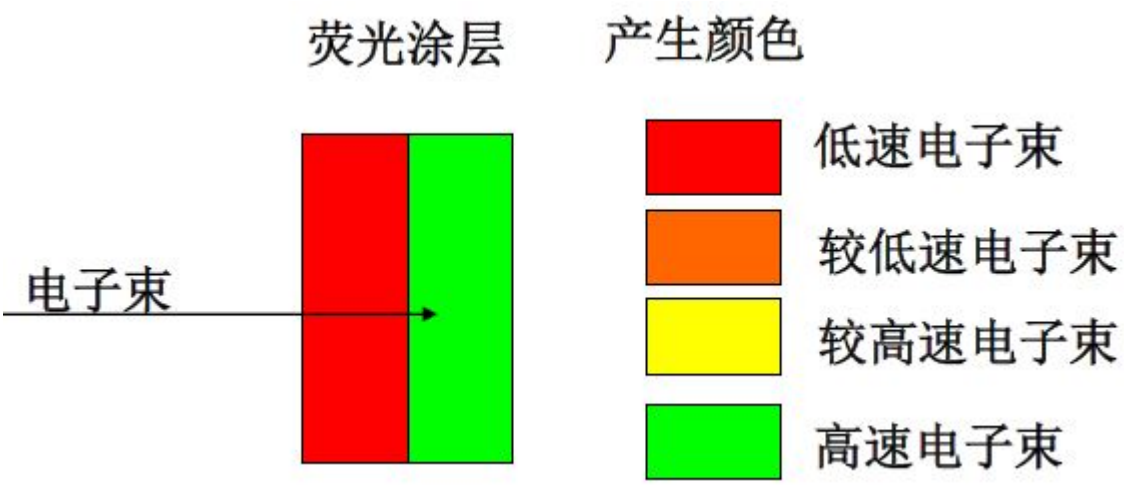
逻辑部件——帧缓冲储存器+视频控制器+显示处理器+CRT

优点——成本低，易于绘制填充图形，色彩丰富，刷新频率与图形复杂度无关，图形易于修改

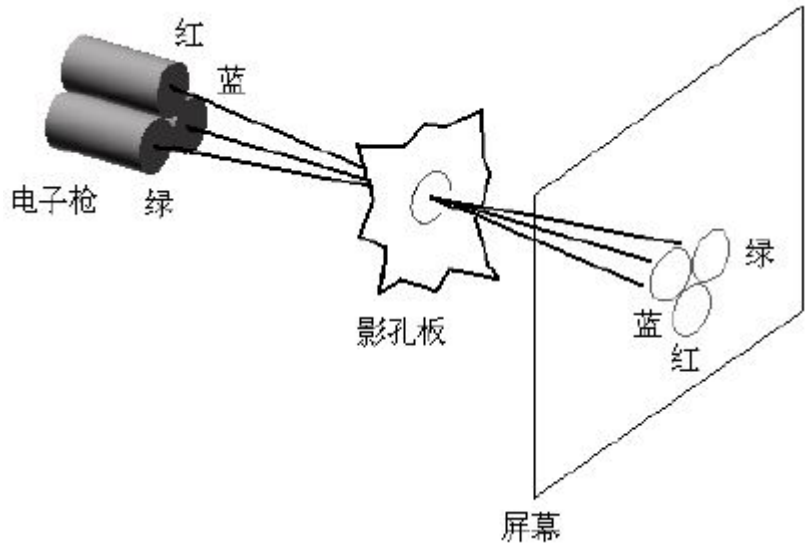
缺点——需要扫描转换，容易发生混淆

阴极射线管图形显示器——电子枪（发射电子束）+聚焦系统（电磁场使电子束变细）+加速电极+偏转系统（最大偏转角是重要性能）+荧光屏（电子束轰击荧光涂层发光）

射线穿透法：两层荧光涂层，一层红光一层绿光，电子束轰击穿透荧光层的深浅决定了产生的颜色。成本低，价格便宜；颜色只有四种，图形质量差。



影孔板法：显示屏从外到内为外层玻璃，荧光涂层，影孔板。影孔板可以精确定位像素，每个孔对应一个像素，显示时，三支电子枪照射一个影孔，通过调节电子束可以控制RGB的亮度。



阴极射线管显示器基本指标——最大偏转角，点距（影孔板两个孔之间的距离），分辨率

LCD显示器：利用液晶的特殊属性——液体的力学性质和固体的光学性质。当液晶收到电压影响时，光的折射角发生变化，产生色彩。

屏幕从内向外分别为——背光光源，偏光板，液晶体（在这里光源色彩发生变化），彩色滤光片，偏光板，外层玻璃

LCD显示器基本指标——可视角度，点距（两个液晶颗粒或光点之间的距离），分辨率

Frame Buffer——存储屏幕上像素的颜色值，帧缓存中单元数目与显示器上像素数目相同，单元与像素一一对应。显示颜色的种类与每个单元的位数相关。

高分辨率和真彩模式产生的问题——需要大容量帧缓存，并且要求帧缓存存取速率很快。解决方法一是查色表，二是隔行扫描。

查色表：一维线性表，每一项对应一种颜色，每个单元n位，则查色表长度为2^n位，查色表使得帧缓存不增加的情况下，可以大范围挑选颜色。

显存中某位的值→查色表地址→得到颜色→屏幕上对应位置显示该颜色

3. 图形变换与投影

齐次坐标：原本的n维向量用一个n+1维向量来表示。$(x, y) \rightarrow (x_h, y_h, h)$。

$$(x, y) \quad \rightarrow \quad (x_h, y_h, z_h)$$
$$x_h = hx,\ y_h = hy,\ z_h = h \neq 0$$

标准齐次坐标，取h=1，(x, y, 1)

二维变换的矩阵表示——便于变换合成，便于硬件计算

平移变换

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{记为} = T(t_x, t_y) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

旋转变换

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{记为} = R(\theta) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

缩放变换

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{记为} = S(s_x, s_y) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

关于x轴的对称变换

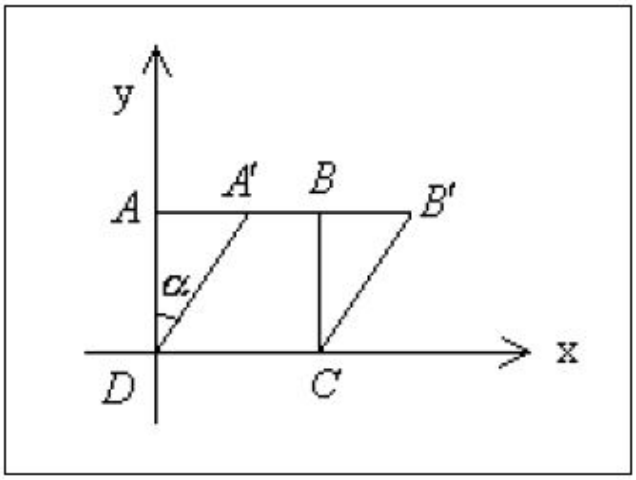
$$SY_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

关于y轴的对称变换

$$SY_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

以y轴为依赖轴的错切变换，以y=0为参考轴

$$\begin{cases} x' = x + sh_x y \\ y' = y \end{cases}$$



$$SH_y(sh_x) = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $sh_x = \tan a$

以x轴为依赖轴的错切变换

$$SH_x(sh_y) = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

仿射变换：几何变换的一种一般形式，会保持平行直线的平行性

$$Af = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

二维图形显示流程：世界坐标系→世界坐标系内的变换→关于窗口的裁剪→窗口到视区的变换→设备坐标系→扫描转换→显示或储存图像

世界坐标系（用户坐标系）：真实的坐标系

局部坐标系：以待绘制物体为参考的坐标系

设备坐标系（屏幕坐标系）：屏幕上的坐标系

窗口：在世界坐标系中指定的矩形区域，用来指定待显示的图形

视区：设备坐标系上指定的矩形区域，用来指定窗口内的图形在屏幕上显示的大小及位置

三维几何变换

三维齐次坐标：$(x,y,z)\rightarrow(x_h,y_h,z_h,h),\ x_h=hx,\ y_h=hy,\ z_h=hz,\ h\neq 0$

标准三维齐次坐标：$(x,y,z)\rightarrow(x,y,z,1)$

三维变换的一般形式

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

三维图形的基本研究内容——投影；三维形体表示；消除隐藏线与隐藏面；建立光照模型、开发真实感图形绘制方法

三维图形显示流程：三维世界坐标系→三维裁剪→投影→窗口到视区的变换→三维设备坐标系

照相机模型：选定投影类型，设置投影参数（拍摄方向），三维裁剪（取景），投影和显示（成像）

视见体：三维裁剪的窗口。定义窗口→发出射线→形成观察空间→确定前后裁剪面→形成视见体

投影：将n维的点变换为小于n维的点

投影中心（COP），对应电影放映机；投影面，不经过投影中心，对应电影屏幕；投影线，从投影中心向物体上各个点发出的射线。

平面几何投影：投影面是平面，投影线是直线。

透视投影：COP与投影平面间距离有限。产生近大远小的视觉效果，由它产生的图形深度感强，看起来更加真实。

平行投影：COP与投影平面之间的距离无限，可以视为极限情况下的透视投影。投影方向与投影平面正交——正投影；否则——斜投影。

观察坐标系VRC——照相机所在的坐标系，用来简化和加速投影变换

坐标原点VRP：聚焦参考点在投影平面上的投影

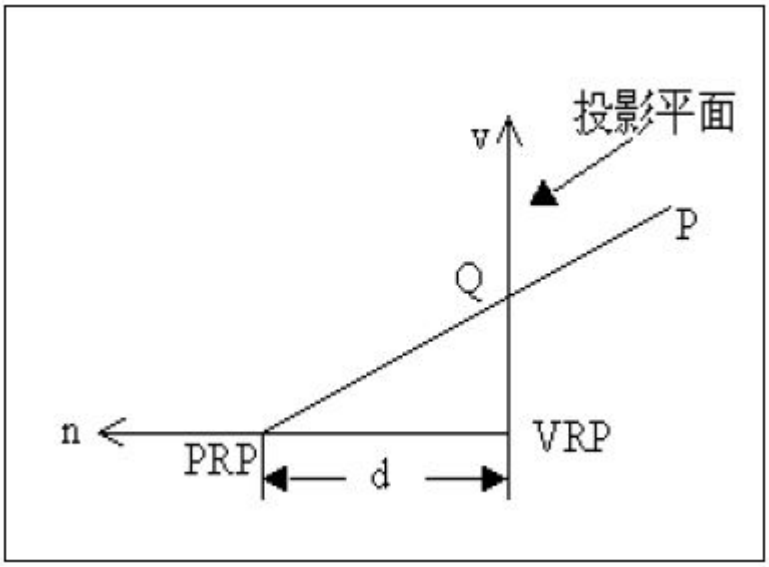
n轴：投影平面的法向（照相机镜头方向）

v轴：观察正向VUP在投影平面的投影（照相机向上的方向）

u轴：u = v x n

投影变换——观察坐标系中的投影变换

透视投影



$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{bmatrix}$$

$$Q = M_{per} \bullet P$$

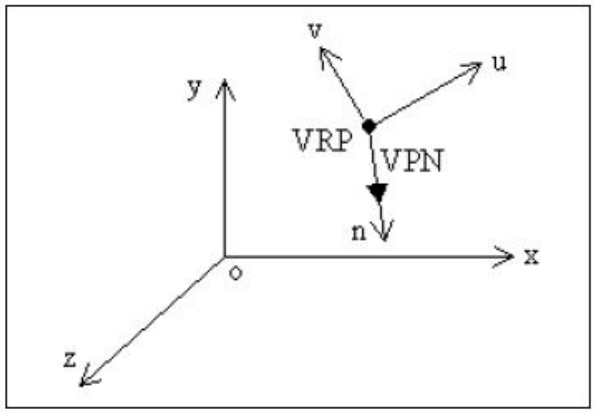
平行投影变换

$$M_{ort} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q = M_{ort} \bullet P$$

世界坐标系到观察坐标系的变换

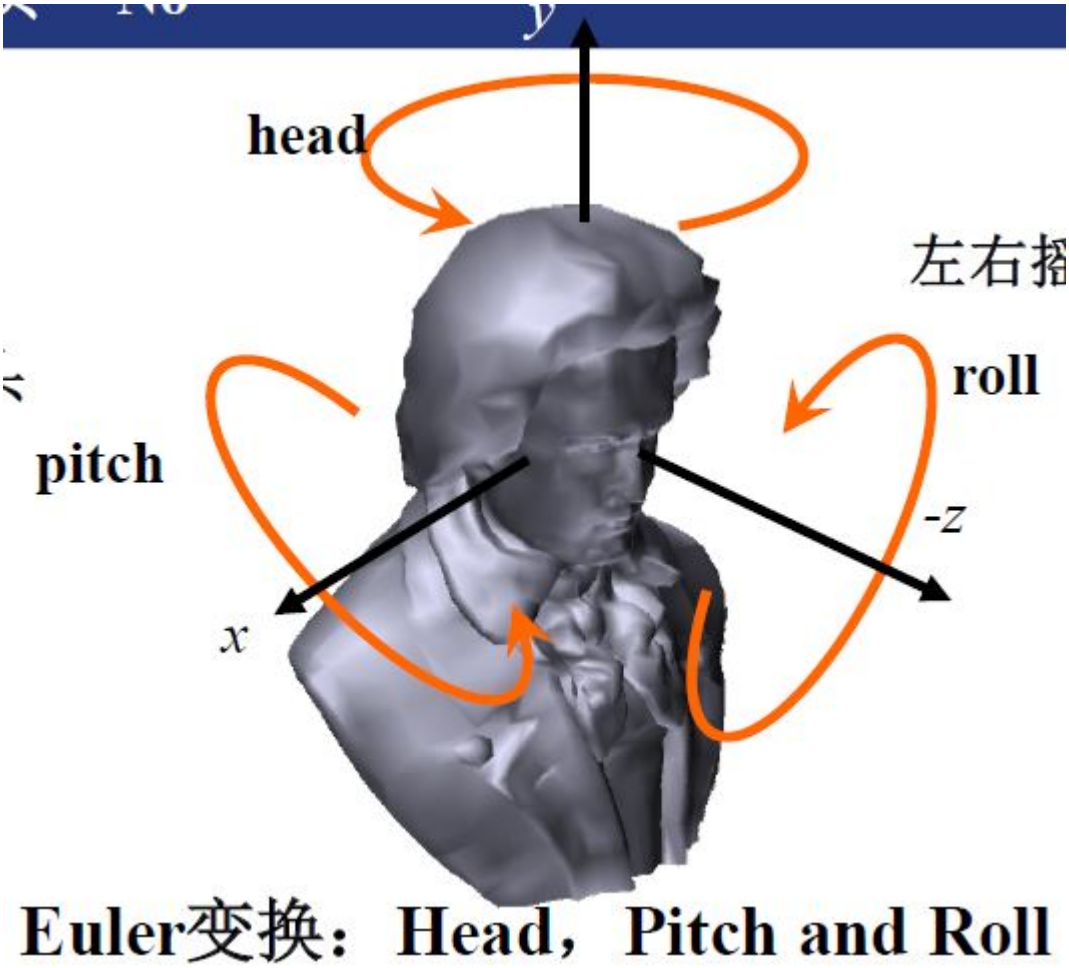
$$\begin{cases} n = \dfrac{VPN}{\|VPN\|} \xrightarrow{\text{记为}} [n_x, n_y, n_z] \\ u = \dfrac{VUP \times VPN}{\|VUP \times VPN\|} \xrightarrow{\text{记为}} [u_x, u_y, u_z] \\ v = n \times u \xrightarrow{\text{记为}} [v_x, v_y, v_z] \end{cases}$$



$$M_{WC \to VRC} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera操作——Euler变换：一种使物体朝向一指定方向的有效方法

$$E(h,p,r) = R_z(r)R_x(p)R_y(h)$$



Gimbal lock现象，当一个自由度丧失时，比如$p = \frac{\pi}{2}$时，E只依赖于一个角，此例子中为$(r + h)$

四元数——在表示旋转和朝向方面优于Euler角，具有表示紧凑，朝向插值稳定的优点，常用于表示各种camera的旋转变换。

三维图形显示流程：模型坐标系→模型变换→世界坐标系→观察变换→观察坐标系→关于视见体裁剪→投影→投影平面→窗口至视区的变换→设备坐标系→显示

关于何时裁剪：三维裁剪（投影之前进行裁剪），只对可见物体进行投影变换，裁剪复杂；二维裁剪（投影之后进行裁剪），裁剪容易，但需要对所有物体进行投影变换。

规范视见体——简化三维裁剪过程。平面投影的规范视见体是半立方体，透视投影的规范视见体是四棱台。

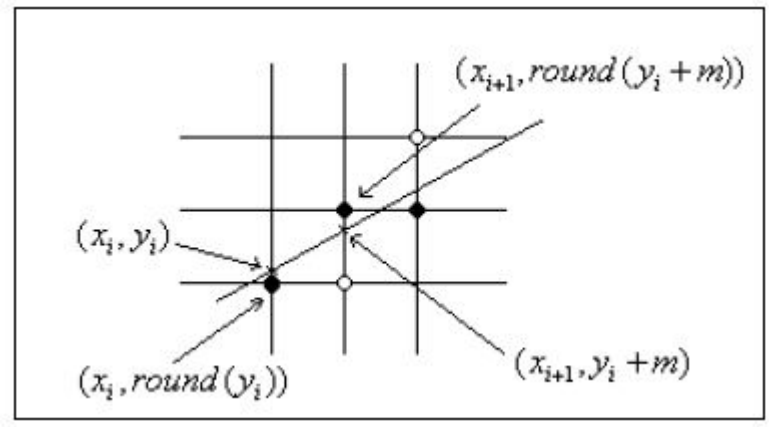
三维裁剪的方法：将齐次坐标转为三维坐标，在三维空间内裁剪，裁剪容易但需要做坐标转换，并且一些有理曲线只能用齐次坐标表示，无法进行转换；直接在四维齐次坐标空间中裁剪，不需要转换但复杂

4. 光栅图形学——直线与平面图形

直线段的扫描转换算法——数值微分法(DDA)，中点画线法，Bresenham算法

DDA算法——增量算法，沿直线的x坐标计算y坐标，效率低且不利于硬件实现（有浮点数取整运算）

对于浮点数，$round(a) = int(a + 0.5)$
$(x_i, y_i) \to (x_i + 1, y_i + m)$，$m$为直线斜率
$y_{i+1} = mx_{i+1} + b = m(x_i + 1) + b = y_i + m$
$(x_i, round(y_i))$即为当前点的坐标



注意！这里只适用于$|k| \leq 1$的情况，x每次+1，y至多+1，如果k>1，则需要将xy互换

中点画线法——判断距离直线最近的下一个像素点是右方的点还是右上方的点。判断方法：根据这两个重点在直线的哪一侧判断。

直线方程：$F(x,y)=ax+by+c=0,\ a=y_0-y_1,\ b=x_1-x_0,\ c=x_0y_1-x_1y_0$
直线上方的点：$F(x,y)>0$，直线下方的点：$F(x,y)<0$
构造判别式：$d=F(M)=F(x_p+1,y_p+0.5)$，根据d的正负即可判定下一个像素
d的更新：若$d\geq 0$，那么下一个像素的判别式$d_1=F(x_p+2,y_p+0.5)=d+a$
若$d<0$，那么下一个像素的判别式$d_2=F(x_p+2,y_p+1.5)=d+(a+b)$

改进方法：d只有在初值会包含小数（来自于0.5），之后的所有增量都是整数值，那么可以用2d来替代d，之后所有操作都用2倍，这样算法中就只有整数，并且不包含乘除法。

Bresenham算法——过各行各列像素中心构造虚拟网各县，按照直线起止顺序计算直线和各垂直网格线的交点，之后根据误差项d的符号确定该列最近的像素。

$d_0=0,\ d_{i+1}=d_i+k$，若$d\geq 1$，则$d\leftarrow d-1$
若$d>0.5,\ (X_i,Y_i)\rightarrow(X_{i+1},Y_{i+1})$
若$d<0.5,\ (X_i,Y_i)\rightarrow(X_{i+1},Y_i)$

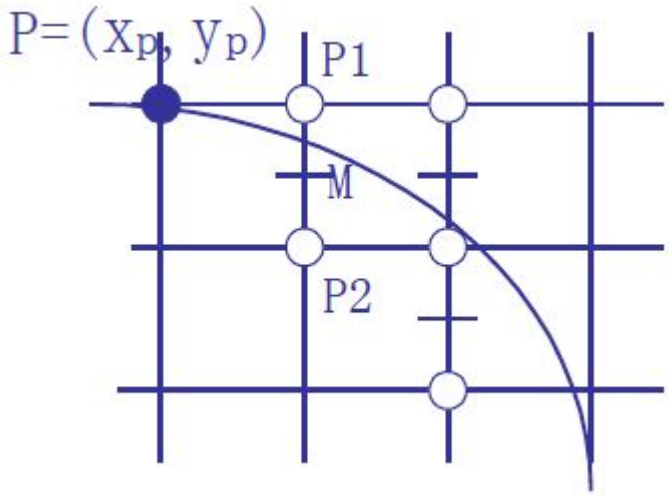
为了方便计算，令e=d-0.5，e的更新方式与d相同，用e的符号来进行判断。

圆弧（椭圆）扫描转换算法——直接离散方法，中点画线算法，多边形逼近法（内接正多边形法，等面积正多边形法），正负法

直接离散方法——需要进行开根或者三角函数运算，计算量太大，不可取。

中点算法——与直线类似

圆弧方程：$F(x,y)=x^2+y^2-R^2=0$，切线斜率$m\in[-1,0]$
中点判别式：$d=F(M)=F(x_p+1,y_p-0.5)$
若$d\geq 0$，那么取右下P_2，下一个像素的判别式$d_1=F(x_p+2,y_p+0.5)=d+a$
若$d<0$，那么取正右P_1，下一个像素的判别式$d_2=F(x_p+2,y_p+1.5)=d+(a+b)$



内接正多边形逼近法——将圆转化为其内接正n边形，之后利用直线中点算法即可。首先在圆上确定一点作为起始点，之后按照下式递推计算出n边形所有顶点。

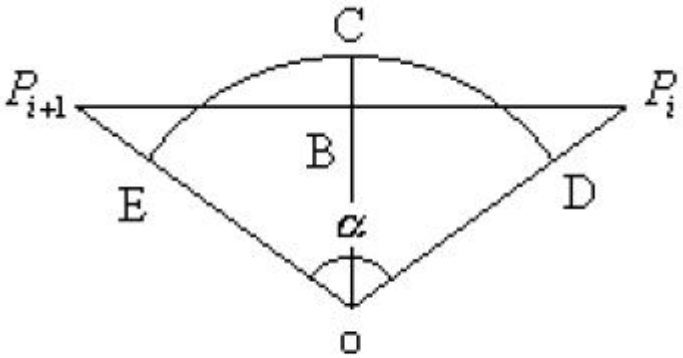
$$x_{i+1} = x_i \cos\alpha - y_i \sin\alpha$$
$$y_{i+1} = x_i \sin\alpha + y_i \cos\alpha$$

问题在于如何确定n。给定最大误差δ，确定$n = \frac{2\pi}{\alpha}$。

$$R - R\cos\frac{\alpha}{2} \le \delta \to \cos\frac{\alpha}{2} \ge \frac{(R-\delta)}{R} \to \alpha \le 2\arccos\frac{R-\delta}{R}$$

等面积多边形逼近法——求多边形径长，由δ确定α。要求

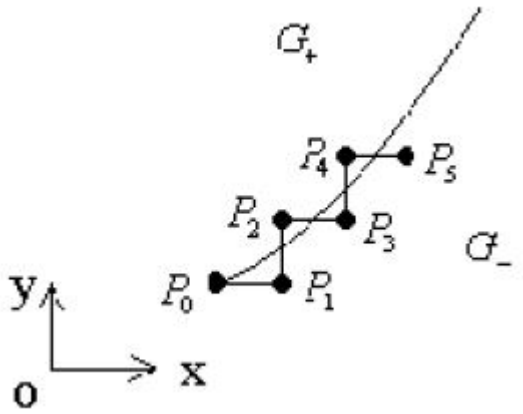
$$S_{扇形ODCE} = S_{\triangle OP_iP_{i+1}}$$



正负法——顺着圆的边交错地行走，步长Δ，如果当前点与初始点在同侧则前进Δy，在异侧则前进Δx.

判别式：$D(P_i) = F(P_i) \cdot F(P_1) = F(x_i, y_i) \cdot F(x_0 + \Delta x, y)$

若$D(P_i) \ge 0$，则前进Δy；若$D(P_i) < 0$，则前进Δx



线宽控制——像素复制方法，移动刷子方法，填充图形方法（等距线方法）

像素复制方法——直接在水平或竖直方向上复制。效率很高，实现非常简单。

缺点：线段两端要么水平要么竖直，不自然；折线处会出现缺口；圆弧的宽度不一致；宽度不合要求，不是在法线上的宽度；奇偶数像素的效果不同，不对称。

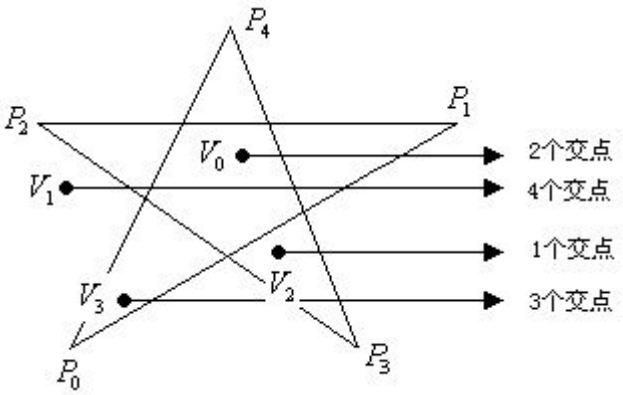
移动刷子方法——线宽与线段的斜率有关，动态地计算线宽

等距线方法——填充等距线之间的部分。线宽均匀，端口与边垂直，生成的图形质量高；但一致性难以保证，比如椭圆。

扫描转换矩形——简单地转换即可，对于共享边界属于谁的问题，采用左闭右开下闭上开的原则。

扫描转换多边形——逐点判断法（射线法，累计角度法，编码法，程序都简单，速度慢，效率低），扫描线算法，边缘填充法

射线法——从像素点发出射线，计算与多边形边相交的次数，若为偶数次说明在外面，否则说明在里面。



累计角度法——从像素点向多边形各个顶点发出射线，计算形成的有向角之和

$$\text{若} \sum_{i=0}^{n} \theta_i = 0, \text{则} v \text{在多边形外}$$

$$\text{若} \sum_{i=0}^{n} \theta_i = \pm 2\pi, \text{则} v \text{在多边形内}$$

编码方法——累计角度方法的离散方法

a. 预处理，测试点是否在边上
b. 以v为远点建立局部坐标系，对多边形P的所有顶点进行编码
c. 边编码——PiP(i+1):ΔPiP(i+1)=I(P(i+1))-I(Pi)
d. 计算编码和。若为0，则在外；若为+/-4，则在内；若为2则需要特殊处理

扫描线算法（有序边表算法）——只能处理非自交多边形，若出现自交，必须将其切为几个不自交的多边形再填充。效率远高于逐点填充法，但对于各种表的维持和排序开销大，适合软件实现而不适合硬件实现

填充边界像素：如果交点落在像素上，左边填充，右边部填充；其他情况下交点在左边，向右取整填充，在右边，向左取整填充。

水平边属于特殊情况，直接扔掉不考虑。

活性边：与当前扫描线相交的边

活性边表：按交点X的增量顺序存放在一个链表中，活性边表的每个节点存放对应边的有关信息。存放内容——ymax本条边的上端点的y，X当前扫描线与边的交点，deltax当前扫描线到下一条扫描线之间的X的增量（边的斜率的倒数），nextEdge与本条边相邻的下一条边

边的连贯性：某条边与当前扫描线相交，也可能与下一条扫描线相交

扫描线的连贯性：当前扫描线与各边的焦点顺序也与下一条扫描线与各边的焦点顺序相同或类似

1. 如果ET中当前类非空，则将其中的边取出插入AET
2. 如果有新的边，则把新边表NET[i]中的边节点，用插入排序法插入活性边表AET，使之按x坐标增序排序
3. 遍历AET，把配对交点之间的区间（左闭右开）上的各个像素填充
4. 遍历AET，把Y_max=i的节点删除并把Y_max>i的结果点的x递增deltax
5. 重复各扫描线过程

边缘填充法——对于每条扫描线和每条多边形边的交点，将该扫描线上交点右方的所有像素取补，对多边形每条边进行相同的操作而无关顺序。

原理：用求余运算替代扫描线算法中的排序运算。$\bar{M} = 0xFF\cdots F - M$

算法简单，适用于具有帧缓存的图形系统；但对于复杂图形，一个像素可能被多次访问，I/O量很大。

以扫描线为中心的边缘填充：
1. 将每一条扫描线上所有像素着色\bar M
2. 遍历所有扫描线，发现和某边交点则从该交点向右求余
以边为中心的边缘填充
1. 将绘图窗口背景设置为\bar M
2. 对每一条非水平边，从该边上每个像素起向右求余

扫描转换扇形——类似多边形转换，但需要分情况讨论直线段和圆弧段的相交顺序。根据直线与圆弧的交点P1P2的位置不同分为4 x 4种情况

种子填充算法——从内部的一个点出发，通过四连通或者八连通找到下一个点，如此搜索地填充图形。可以用于填充带内孔的平面

缺点：递归搜索效率低。

解决方法：扫描线种子填充算法

```
1．栈顶像素出栈
2．沿扫描线向左向右填充直至遇到边界
3．检查上下扫描线，选相应区间最左像素所谓种子像素入栈
```

以图像填充区域——建立起绘图空间与纹理（图像）空间的一一映射；或建立区域局部空间与图像空间的一一映射

反混淆（抗锯齿）

混淆（走样）——用离散量表示连续量引起的失真，具体体现为阶梯状边界，图形细节失真，狭小图形遗失（动画序列中产生闪烁）

提高分辨率——方法简单有效，但代价很大

加权反走样——将像素视为若干个“小像素”，每个小像素具有一个权值，根据图形覆盖的小像素加权得到该像素的颜色。一般权函数采用高斯分布，在实际应用中需要进行离散化，如下

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

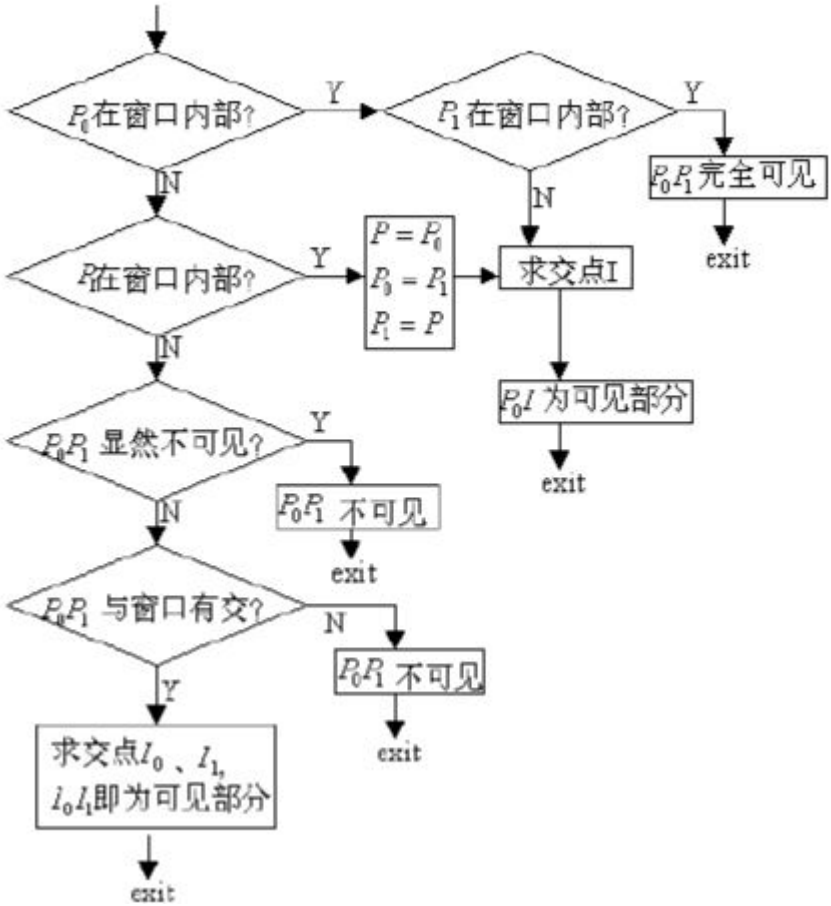
5. 光栅图形学2——二维裁剪与三维图形

二维裁剪——判断图形元素是否落在窗口区域内，只显示图形内容，其它地方不进行显示。

先扫描转换，再裁剪，在设备坐标系中进行，算法效率不高，适合求交难度大的

直线段裁剪的点裁剪——直接求交算法，Cohen-SutherLand算法（编码算法），Liang-Barsky算法

直接求交算法——直线与窗口的边都写为参数形式，判断直线是否全部或部分在窗口内，若是则直接解方程计算交点。



C-S算法——对于直接求交法对完全课件和显然不可见的判断进行优化

编码：$C_tC_bC_rC_l$对应上下左右。初始时编码置0；若$y > y_{max}$，则$C_t = 1$；若$y < y_{min}$，则$C_b = 1$；若$x > x_{max}$，则$C_r = 1$；若$x > x_{min}$，则$C_l = 1$

若两个端点的编码都为0则完全 可见；若两个端点编码求与后非0，则线段显然不可见；其他情况下根据编码令线段与窗口边界求交，验证在各段的小线段是否满足。

最坏情况下，一个线段需要求交四次（有一些多）。根据交点编码快速判断线段完全可见和显然不可见。适用于大窗口和特别小窗口。

Liang-Barsky算法——点集间的交集即为裁减结果，将二维裁剪简化为一维裁剪。

对于一维裁剪，线段为P0P1，诱导窗口为Q0Q1，假设P0P1长度为一个单位，Q0为t0，Q1为t1。那么

P_0P_1至少部分可见的充要条件：$\max\{0, t_0\} \le \min\{1, t_1\}$
并且可见部分的参数区间为：$[\max\{0, t_0\}, \min\{1, t_1\}]$

诱导窗口的计算：假设直线l与窗口左右上下四条边的交点为LRTB，那么P0P1的可见部分为
$VW = P_1P_1 \cap LR \cap TB$

Lyrus-Beck算法（参数化算法）——L-B算法的推广，裁剪窗口可以是凸多边形。

线段的参数表示：$P(t) = (P_2 - P_1) * t + P1$

凸多边形的性质：P(t)在凸多边形内的充要条件是，对于凸多边形边界上任意一点A和该点处内法向量N，都有
$N \cdot (P(t) - A) > 0$.

k边形中可见线段参数区间的解：

$$N_i \cdot (p(t) - A_i) \geq 0,\ i = 0, 1, \cdots, k, 0 \leq t \leq 1$$

线段可见的交点参数

$$t_1 = \max\{0, \max\{t_i :\ N_i \cdot (P_2 - P_1) > 0\}\}$$
$$t_2 = \min\{1, \min\{t_i :\ N_i \cdot (P_2 - P_1) < 0\}\}$$

若$t_1 \leq t_2$，则$[t_1, t_2]$是可见线段的交点参数区间；否则线段不可见

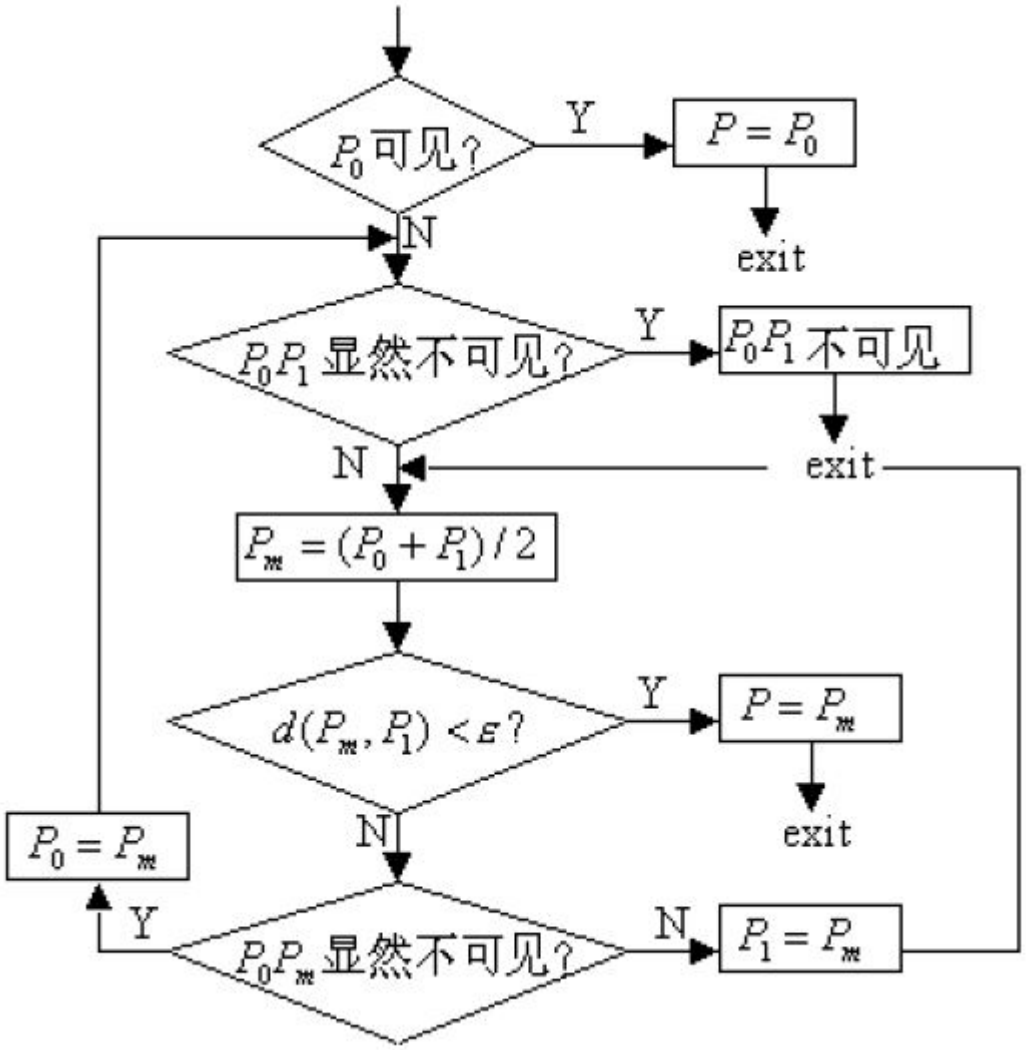
当凸多边形为矩形窗口且矩形的边与坐标轴平行时，算法退化为L-N算法。

Nicholl-Lee-Nicholl算法——消除C-S算法中多次求交的情况，对2D平面进行更细的划分。

首先先像C-S算法中一样，对区域进行细分，分为九格；在需要求交时，从P0向窗口四角引出射线，把平面分为了四个有意义的区域，判断P1在哪个区域中（利用斜率判断）即可判断P0P1需要与哪条边求交。

该方法效率较高，但仅适合二维矩形窗口。

中点分割法——从P0出发找到距离P0最近的可见点，从P1出发找到矩P1最近的可见点。二分地进行查找



二分查找效率较高，算法过程中只用到加法和除2（右移），适合硬件，适合并行。

多边形裁剪——Sutherland-Hodgman算法，Weiler-Atherton算法（内裁减）

S-H算法——将多边形关于矩形窗口的裁剪分解为多边形关于窗口四边所在直线的裁剪，采用流水线式处理，依次进行左上右下的裁剪。

该方法采用流水线方式，适合硬件实现，并且可以推广到任意凸多边形裁剪窗口和三维凸多面体裁剪窗口。

W-A算法——可以进行任意多边形（凹凸带孔）窗口或待裁剪多边形的裁剪，地位对等，被裁减多边形为主多边形，裁剪窗口为裁剪多边形。

主多边形和裁剪多边形将二维平面分为四部分，其中内裁减（$A \cap B$）为待求裁剪。内裁减的边界，是主多边形边界和裁剪多边形的边界交替出现的。

进点：主多边形边界进入裁剪多边形边界的内裁减的顶点；出点：裁剪多边形边界进入主多边形边界的内裁减的顶点

算法步骤：建立定点表，求取交点，裁剪，对奇异情况进行裁剪

奇异情况：与裁剪多边形边重合的主多边形的边不参与求交点；对于顶点落在了裁剪多边形的边上的主多边形的边，如果落在裁剪边的内侧，则该顶点算作交点，否则落在裁剪边外侧，该顶点不看作交点。

字符串裁剪

基于字符串——只保留完全可见的字符串

基于字符——只保留完全可见的字符

基于构成字符的最小元素——点阵字符使用点裁剪，矢量字符使用线裁剪

表示形体的模型

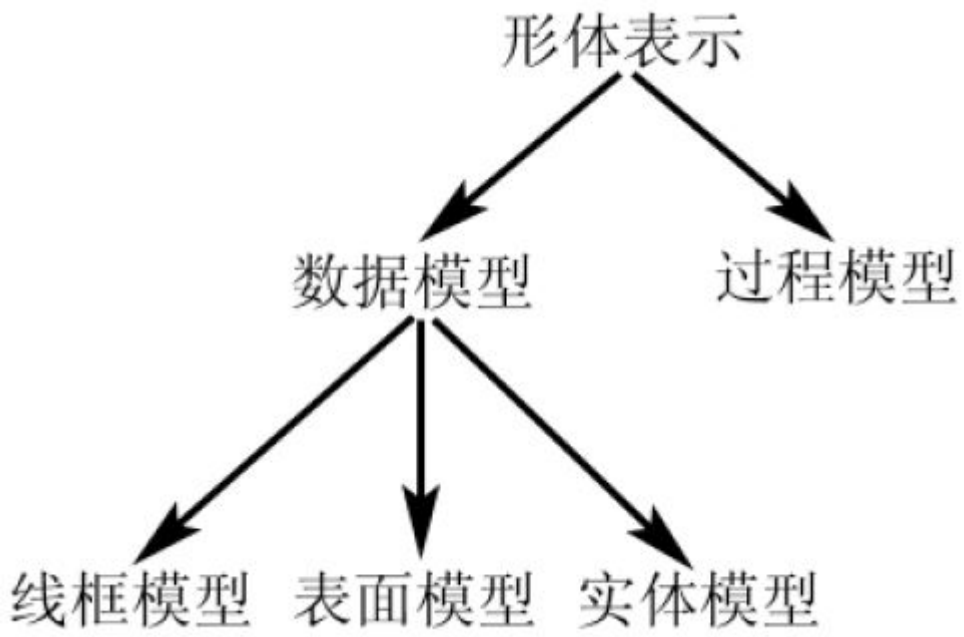
数据模型——完全以数据描述

线框模型，将形体表示为一组轮廓线的集合，简单处理速度快，高度抽象，不适合真实感显示

表面模型，将形体表示为一组平面的集合，适合真实感显示

实体模型，描述实体，包含较多信息，如几何信息，拓扑信息等

过程模型——以一个过程和相应的控制参数描述，比如粒子系统，L系统



实体的定义——具有一定的形状，具有封闭的边界（表面），内部连通，占据有限的空间，经过运算后仍然是有效的实体

内点，边界点，取内点运算i，取闭包运算c，正则运算r

正则点集——$r \cdot A$是A的正则点集；若$r \cdot A = A$则A是正则点集

Euler's Formula——对于简单的没有洞的多边形

$$\text{顶点数} V - \text{边数} E + \text{面数} F = 2$$

推广后的Euler's Formula

$$V - E + F - \#Holes\_in\_faces = 2(\#Components - \#Holes\_thru\_Object)$$

实体可计算条件——正则点集，表面是二维流形

二维流形——其上任意一点存在充分小的领域与圆盘同构（存在连续的一一映射）

正则集合运算——$A\ op^*\ B = r \cdot (A\ op^*\ B)$

正则并，正则交，正则差

特征表示——用一组特征参数表示一组类似的物体，特征包括形状特征，材料特征等。适合工业标准件的表示。

空间分割表示——空间位置枚举，八叉树，单元分解表示

空间位置枚举——同样大小的立方体粘合表示物体

八叉树表示——不同大小的立方体粘合表示物体

单元分解表示——多种体素粘合表示物体

均分立方体——用三维数组C[i][j][k]表示物体，为0表示没有物体，为1表示被物体占据

优点：可以表示任何物体，容易实现物体的集合运算，容易计算物体整体性质；缺点：存储空间大，没有边界信息，几何变换困难，只是物体的非精确表示

八叉树——自适应地分割

建树：根节点对应整个物体空间；节点被完全占据，标记为F，返回，没有物体，标记为E，返回；部分占据，标记为P，之后将这一空间分割为八个，对每个子立方体进行同样的处理。

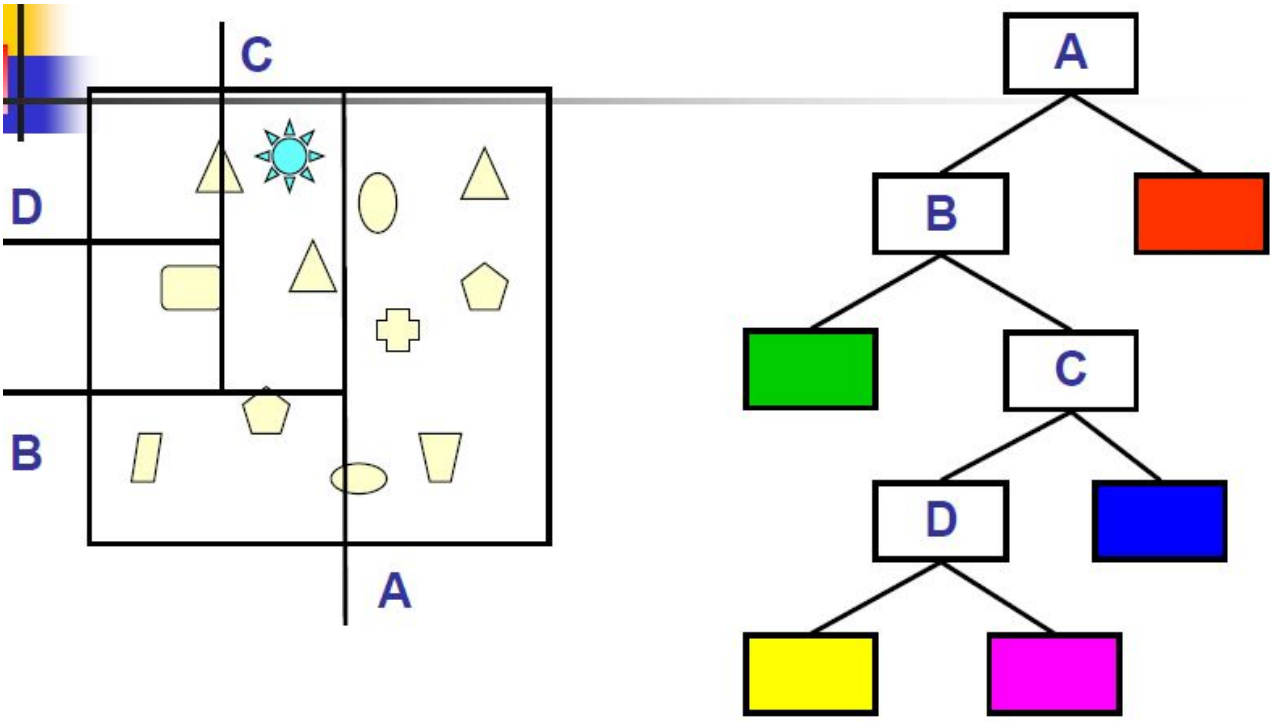
优点：可以表示任何物体，容易实现集合运算，容易计算物体体积，存储空间较少；缺点：没有边界信息，不适合图形显示，难于集合变换，非精确表示。

单元分解表示——将单一体素分割为多种体素。

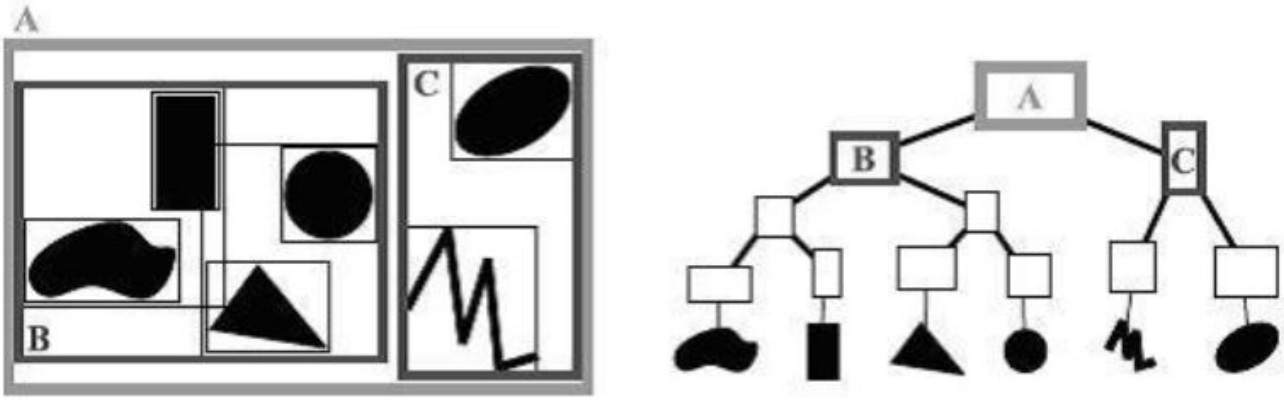
空间层次划分——均匀分割，KD-tree，BVH

均匀分割——创建速度快，实现简单，但对于不均匀的图元遍历效率低下

KD-Tree——遍历速度快，创建速度慢



BVH——适合动态场景的表示及碰撞检测。基于图元，创建过程快，便利效率不高。



空间分割表示，优点：表示简单容易实现几何变换，基本体素按需选择，物体可以精确表示；缺点：物体表示不唯一，有效性不保证

推移表示（SWEEP体）——将物体A沿轨迹推移得到的物体B

平移SWEEP——将二维区域沿着适量方向推移

旋转SWEEP——将二维区域绕旋转轴旋转一周

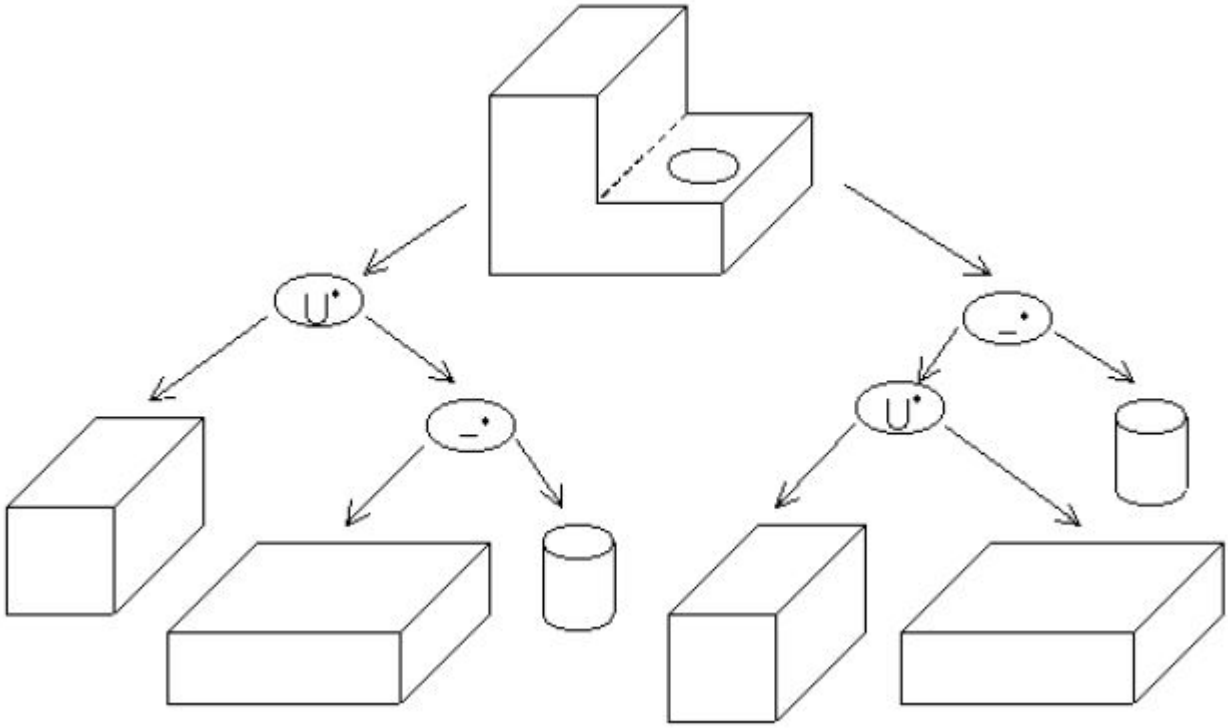
广义SWEEP——轨迹任意，物体任意，推移过程物体可以变形

优点：简单直观，适合作为图形输入；缺点：对几何运算不封闭，作几何变换困难

边界表示——确定了物体的边界也就确定了物体本身。

优点：精确表示物体，表示能力强，容易几何变换，适合显示处理；缺点：表示复杂，难以保证有效性，集合运算复杂。

构造实体几何表示（CSG树）——叶节点为基本体素，中间节点为正则集合运算



优点：简单直观，保证物体有效性，可以用来构造物体，容易计算整体性质；缺点：不能直接用于显示，表示不唯一，求交计算麻烦。

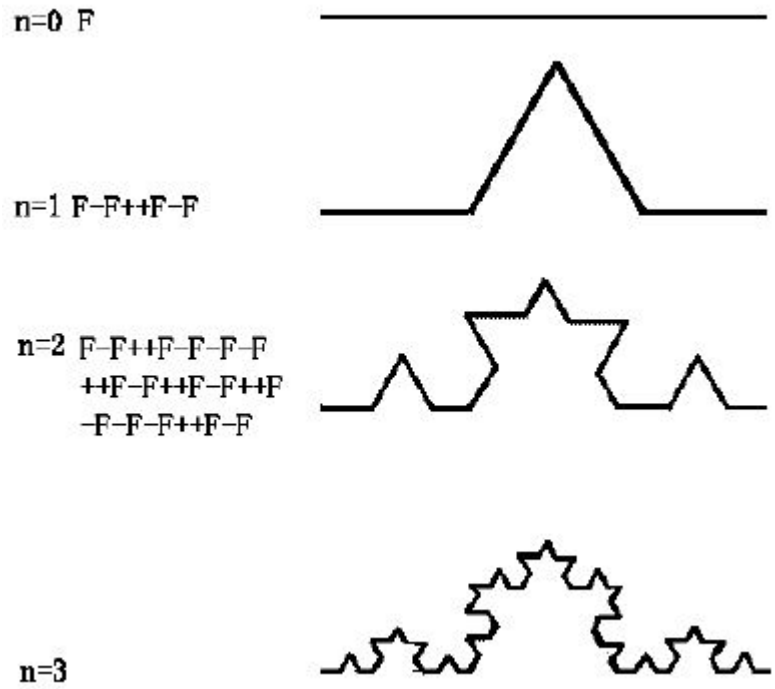
不规则形体的建模方法——迭代函数系统，基于文法的模型，粒子系统

IFS（迭代函数系统）——分形。具备的基本特征——形态不规则形，结构精细性，局部与整体自相似性，维数的非整数性，生成的迭代性。

DOL系统（确定的上下文无关的L系统）——定义三元组，V为字母表，V*为V上所有单词组成的单词表，w为一个非空单词，称为公理，P为产生式集合

任意$\alpha \in V$，存在$x \in V^*$，使得$\alpha \to x$，有且只有如果没有明显的声称式，则令$\alpha \to x$

Koch雪花——V：{F,+,-}；w：F；P：F→F-F++F-F；解释：F向前画线，+右转a，-左转b



粒子系统——略

6. 光栅图形学3——曲线与曲面

参数曲线基础

$P = \vec{P}(t),\ t \in [a,b]$，规范参数区间为$[0,1]$

正则点——导数不为0

正则曲线——其上所有点都是正则点的曲线

斜率——直线的倾斜程度；一个坐标变量关于另一个坐标变量的变化率

切矢量——$P'(t) = [x'(t),\ y'(t),\ z'(t)]^T$，坐标变量关于参数的变化率

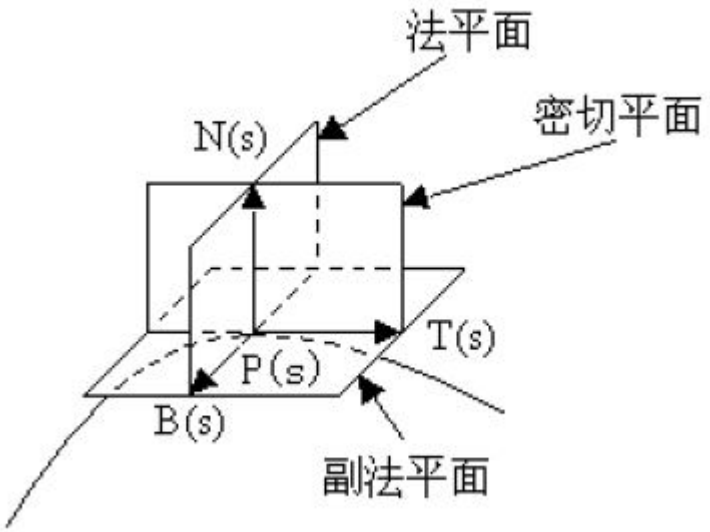
弧长——$s = \lim_{n\to\infty} L(n) = \lim_{n\to\infty} \sum_{i=1}^{n} |P_{i-1}P_i| = \int_0^t |\frac{dP(t)}{dt}|dt$

弧长参数——如果$\frac{ds}{dt} = |\frac{dP(t)}{dt}| > 0$，则s=s(t)是关于参数t的单调函数，因而s=s(t)存在反函数，结论：曲线$P = P(t)$可以用弧长参数$P = P(s)$表示

单位切矢量——$T(s) = \frac{dP(s)}{ds}$

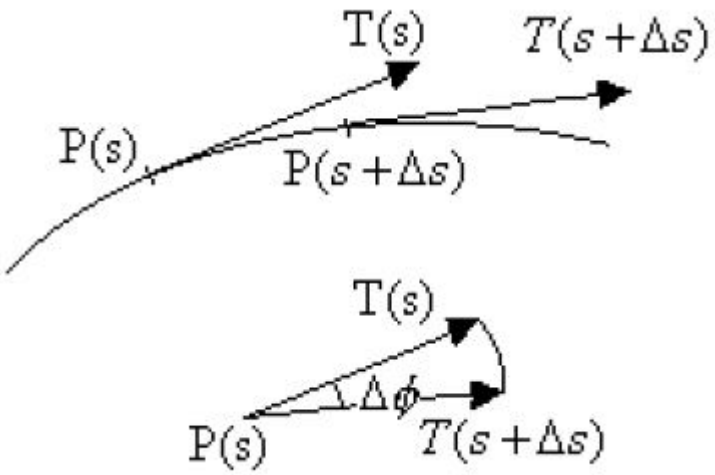
主法矢量——$N(s) = \frac{T'(s)}{|T(s)|}$，N(s)与T(s)垂直

副法矢量——$B(s) = T(s) \times N(s)$



曲率——$k(s)=\lim_{\Delta t\to 0}|\frac{\Delta\phi}{\Delta t}|$，衡量曲线的弯曲程度

曲率半径——$p(s)=\frac{1}{k(s)}$



C^n连续（参数连续性）——传统的，严格的连续性。曲线$P=P(t)$在t_0处n阶参数连续，如果它在$t=t_0$处n阶左右倒数存在并且满足

$$\frac{d^kP(t)}{dt^k}|_{t=t_0^-}=\frac{d^kP(t)}{dt^k}|_{t=t_0^+},\ k=0,1,\cdots,n$$

几何连续性——参数连续性不能客观准确地度量参数曲线的光滑性

GC^0，0阶几何连续，$P=P(t)$在$t=t_0$处0阶几何连续，如果它在t_0处连续

GC^1，1阶几何连续，$P=P(t)$在$t=t_0$处1阶几何连续，如果它在t_0处GC^0，并且切矢量方向连续

GC^2，2阶几何连续，$P=P(t)$在$t=t_0$处2阶几何连续，如果它在t_0处GC^1，并且副法矢量方向连续，曲率连续

$$P(t_0^-)=P(t_0^+)\Rightarrow GC^0$$
$$GC^0,\ P'(t_0^-)=\alpha\cdot P'(t_0^+)\ (\alpha>0\text{为一常数})\Rightarrow GC^1$$
$$GC^1,\ B(t_0^-)=B(t_0^+),\ k(t_0^-)=k(t_0^+)\Rightarrow GC^2$$

参数多项式曲线

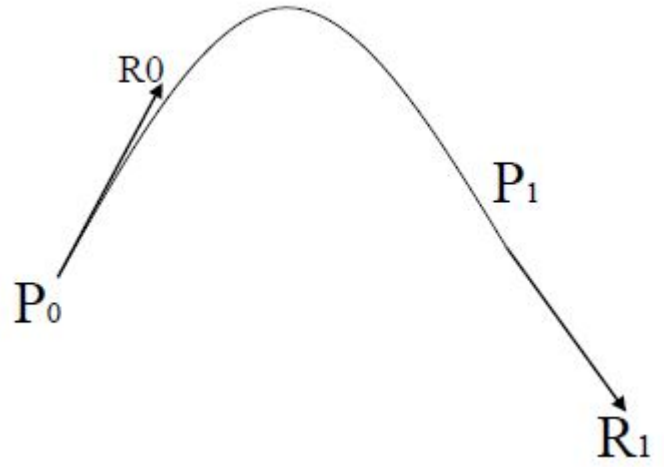
$$P(t)=\begin{bmatrix}x(t)\\y(t)\\z(t)\end{bmatrix}=\begin{bmatrix}x_0 & x_1 & \cdots & x_n\\y_0 & y_1 & \cdots & y_n\\z_0 & z_1 & \cdots & z_n\end{bmatrix}\begin{bmatrix}1\\t\\\vdots\\t^n\end{bmatrix}\overset{记作}{=}C\cdot T,\ t\in[0,1]$$

矩阵表示——$G=[G_0,G_1,\cdots,G_n]$为集合矩阵，其中G_i为控制顶点，M为基矩阵，$M\cdot T$确定了一组基函数。

$$C=G\cdot M$$
$$P(t)=G\cdot M\cdot T,\ t\in[0,1]$$

三次Hermite曲线——给定四个矢量P0,P1,R0,R1，称满足以下条件的三次多项式曲线P(t)为Hermite曲线

$$P(0)=P_0,\ P(1)=P_1$$
$$P'(0)=R_0,\ P'(1)=R_1$$



解法（解不唯一）

按照定义将条件写为矩阵形式：

$$G_H\cdot M_H\cdot T|_{t=0}=G_H\cdot M_H\cdot[1\ \ 0\ \ 0\ \ 0]^T=P_0$$
$$G_H\cdot M_H\cdot T|_{t=1}=G_H\cdot M_H\cdot[1\ \ 1\ \ 1\ \ 1]^T=P_1$$
$$G_H\cdot M_H\cdot T'|_{t=0}=G_H\cdot M_H\cdot[0\ \ 1\ \ 0\ \ 0]^T=R_0$$
$$G_H\cdot M_H\cdot T'|_{t=1}=G_H\cdot M_H\cdot[0\ \ \ 2\ \ 3]^T=R_1$$

合并为矩阵：

$$G_H\cdot M_H\cdot\begin{bmatrix}1&1&0&0\\0&1&1&1\\0&1&0&2\\0&1&0&3\end{bmatrix}=[P_0\ \ P_1\ \ R_0\ \ R_1]\overset{取值为}{=}G_H$$

解出一个基矩阵为：

$$M_H=\begin{bmatrix}1&1&0&0\\0&1&1&1\\0&1&0&2\\0&1&0&3\end{bmatrix}^{-1}=\begin{bmatrix}1&0&-3&2\\0&0&3&-2\\0&1&-2&1\\0&0&-1&1\end{bmatrix}$$

基函数为：

$$M_H\cdot T=\begin{bmatrix}1-3t+2t^3\\3t^2-2t^3\\t-2t^2+t^3\\-t^2+t^3\end{bmatrix}=\begin{bmatrix}G_0(t)\\G_1(t)\\H_0(t)\\H_1(t)\end{bmatrix}$$

三次Hermite曲线的形状控制——控制顶点，调节切矢量，几何变换（等价于对控制顶点的变换）

Bezier基函数——以下n次多项式

$$BEZ_{i,n}(n)=C_n^it^i(1-t)^{n-i},\ t\in[0,1]$$

正性：$BEZ_{i,n}(t)\geq 0,\ t\in[0,1]$

权性：$\sum_{i=0}^{n}BEZ_{i,n}(t)=1,\ t\in[0,1]$

对称性：$BEZ_{i,n}(t)=BEZ_{n-i,n}(1-t)$

降阶公式：$BEZ_{i,n}(t)=(1-t)BEZ_{i,n-1}(t)+tBEZ_{i-1,n-1}(t)$

升阶公式：$BEZ_{i,n}(t)=\frac{i+1}{n+i}BEZ_{i+1,n+1}(t)+\frac{n+1-i}{n+1}BEZ_{i,n+i}(t)$

导数：$BEZ'_{i,n}(t)=n(BEZ_{i-1,n-1}(t)-BEZ_{i,n-1}(t))$

积分：$\int_0^1 BEZ_{i,n}(t)dt=\frac{1}{n+1}$

最大值：在$t=\frac{i}{n}$处取最大值

Bezier曲线——$P(t)=\sum_{i=0}^{n}P_i\cdot BEZ_{i,n}(t),\ t\in[0,1]$

端点位置：$P(t)|_{t=0}=P_0,\ P(t)|_{t=1}=P_n$

端点切矢量：$P'(t)|_{t=0}=n(P_1-P_0),\ P'(t)|_{t=1}=n(P_n-P_{n-1})$

仿射不变性

凸包性：Bezier曲线位于控制顶点的凸包内

三次Bezier曲线的矩阵表示

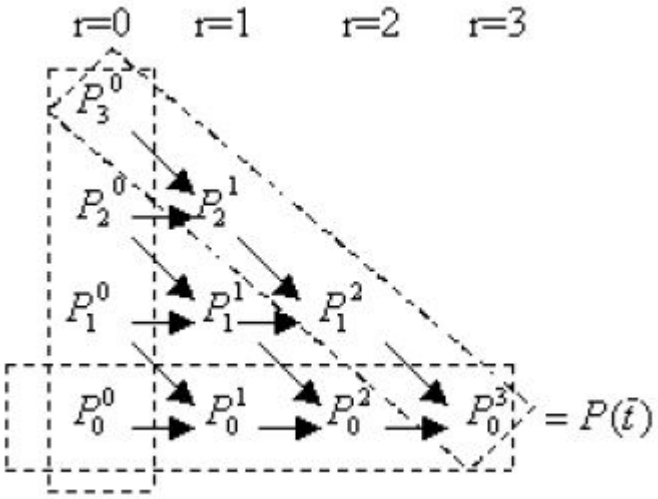
$$P(t)=\sum_{i=0}^{3}P_i\cdot BEZ_{i,3}(t)=[P_0,P_1,P_2,P_3]\begin{bmatrix}BEZ_{0,3}(t)\\BEZ_{1,3}(t)\\BEZ_{2,3}(t)\\BEZ_{3,3}(t)\end{bmatrix}$$

$$=G_{BEZ}\bullet\begin{bmatrix}C_3^0(1-t)^3\\C_3^1t(1-t)^2\\C_3^2t^2(1-t)\\C_3^3t^3\end{bmatrix}=G_{BEZ}\bullet\begin{bmatrix}1&-3&3&-1\\0&3&-6&3\\0&0&3&-3\\0&0&0&1\end{bmatrix}\begin{bmatrix}1\\t\\t^2\\t^3\end{bmatrix}$$

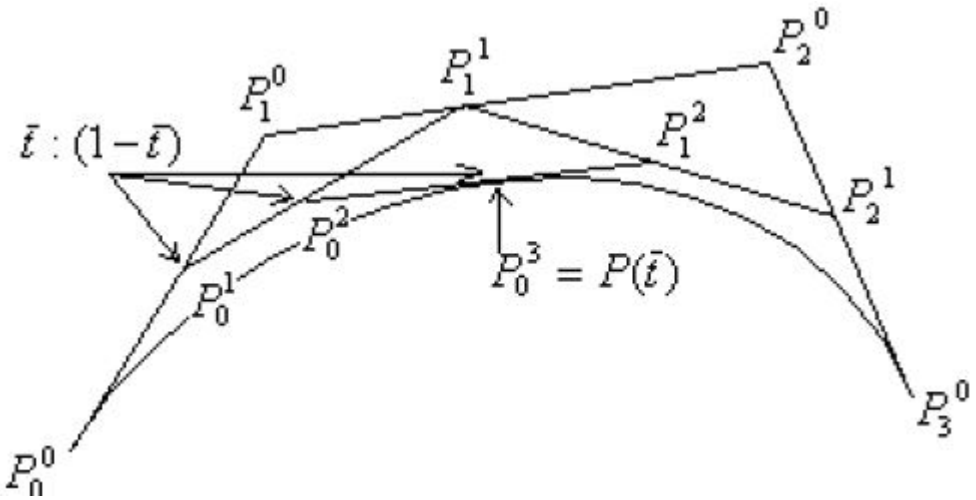
$$=G_{BEZ}\bullet M_{BEZ}\bullet T$$

de Casteljau算法（分割法）——给定参数$\bar{t}$，迭代计算出$P(\bar{t})$

$$P_i^r=P_i,\qquad r=0$$
$$P_i^r=(1-\bar{t})\cdot P_i^{r-1}+\bar{t}\cdot P_{i+1}^{r-1},\quad r=1,2,\cdots,n;\ i=0,1,\cdots,n-r$$



分割定理：P=Q+R，Bezier曲线被分割为两段，得到的两段曲线仍然是Bezier曲线。




```
DisplayBezier(Vector P[], int n, float delta):
1. if dis(P, n) <= delta                //达到精度则显示
2.   then display curve P
3. BezierCurveSplit(P, Q, R, n) //取t=0.5，将P分割为Q和R
4. DisplayBezier(Q, n, delta)   //分别画Q，R
5. DisplayBezier(R, n, delta)
```

参数多项式曲面——$P(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}a_{ij}u^iv^j=U^TAV,\ (u,v)\in[0,1]\times[0,1]$

等参数曲线：将一个参数固定，另一个自由变化。u曲线$P=P(u,v_0)$，v曲线$P=P(u_0,v)$

矩阵分解：$P(u,v)=U^T\cdot M_u^T\cdot G\cdot M_v\cdot V$，其中$G=(G_{ij})_{i,j=0}^{m,n}$

Bezier曲面——$P(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{i,j}BEZ_{i,m}(u)BEZ_{j,n}(v),\ (u,v)\in[0,1]\times[0,1]$

四个边界对应四条Bezier曲线

$P(0,0)=P_{0,0},\ P(0,1)=P_{0,n},\ P(1,0)=P_{m,0},\ P(1,1)=P_{m,n}$

凸包性：Bezier曲面在其控制顶点的凸包之内

de Casteljau算法——给定(u_0,v_0)，计算该点的值

$$\begin{cases} P_{i,j}^{0,0}=P_{i,j} & i=0,1,\cdots m,\ j=0,1,\cdots n \\ P_{i,j}^{r,s}=(1-u_0)P_{i,j}^{r-1,s}+u_0P_{i+1,j}^{r-1,s} & r=1,\cdots m,\ i=0,1,\cdots m-r \\ P_{i,j}^{r,s}=(1-v_0)P_{i,j}^{r,s-1}+v_0P_{i,j-1}^{r,s-1} & s=1,\cdots n,\ j=0,1,\cdots n-s \end{cases}$$

分割定理

$$P:\left\{P_{i,j}\right\}_{i=0,j=0}^{m,n}$$

$$G:\left\{G_{i,j}\right\}_{i=0,j=0}^{m,n}=\left\{P_{0,0}^{i,j}\right\}_{i=0,j=0}^{m,n}$$

$$Q:\left\{Q_{i,j}\right\}_{i=0,j=0}^{m,n}=\left\{P_{0,j}^{i,n-j}\right\}_{i=0,j=0}^{m,n}$$

$$R:\left\{R_{i,j}\right\}_{i=0,j=0}^{m,n}=\left\{P_{i,0}^{m-i,j}\right\}_{i=0,j=0}^{m,n}$$

$$W:\left\{W_{i,j}\right\}_{i=0,j=0}^{m,n}=\left\{P_{i,j}^{m-i,n-j}\right\}_{i=0,j=0}^{m,n}$$