图形学基础算法

BresenHam画直线算法

算法基本思想

- 对DDA算法做优化
- 引入误差项

算法步骤

假设直线方程为y = (dy/dx)x

综上所述，**Bresenham** 画线算法的迭代公式如下：

i. 对于 $(x_0, y_0) = (0, 0)$，初值 $e_0 = 2dy - dx$

ii. 若 $e_i \geq 0$，则选用 **T** 点：

$$y_{i+1} = y_i + 1$$

$$e_{i+1} = e_i + 2dy - 2dx$$

若 $e_i < 0$，则选用 **S** 点：

$$y_{i+1} = y_i$$

$$e_{i+1} = e_i + 2dy$$

算法特点

- 易于硬件实现

扫描种子填充算法

算法基本思想

现以图 2.25 中一个采用边界定义的区域（允许内部含有空洞或孔）为例说明。

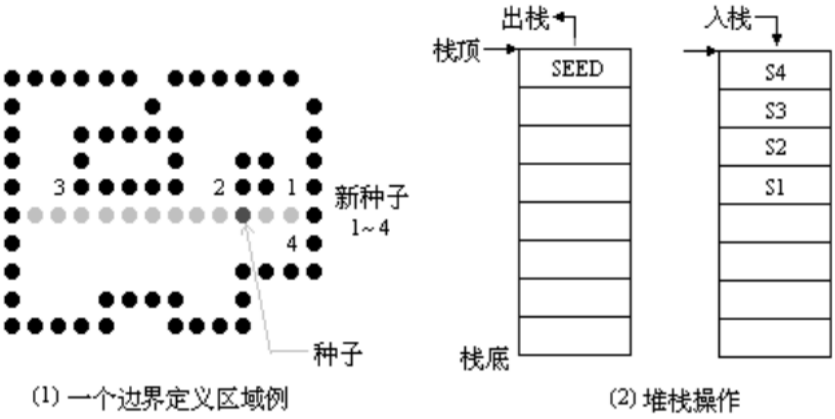


图 2.25　扫描线种子填充算法示意图

假定算法一开始，栈顶存放有种子 seed。

当栈非空时，重复执行如下三个步骤：

步骤一、栈顶像素出栈(Pop)，并置成填充色。

步骤二、沿着这像素所在的扫描线，对左、右像素填色，直至遇到边界像素为止。该区间内最左和最右的像素记为 x_L 和 x_R。

步骤三、在当前区间 $x_L \le x \le x_R$ 中，检查上、下两条扫描线是否为边界或已填充过。若不是，则把每一区间的最右(或最左)像素取作为种子并入栈(Push)。

算法特点

- 假定区域采用边界定义，边界采用特定值，区域内像素均不采用此值。
- 思想或八项扩展，直到遇到边界为止。易于实现。但递归深度大，重复信息多。
- 通过堆栈操作实现

Cohen-Sutherland算法

应用场景-(窗口-视口变换)

裁剪**(Clipping)**意指切掉一个指定区域（窗口，window）外的图形部分而只保留该区域内的图形。这是计算机图形学的基本算法之一。

通常，裁剪后的图形被显示在屏幕上的另一个指定区域（视口，viewport）中。这一处理称为“窗口－视口的映射变换”。如图 2.7 所示。

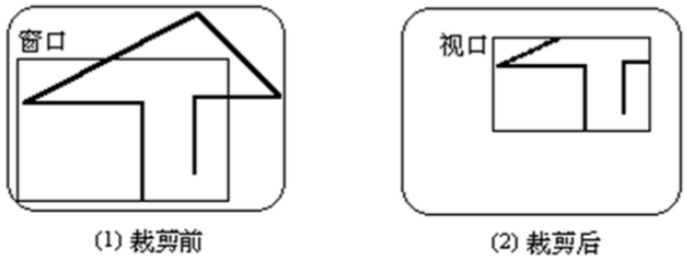


图 2.7　裁剪示意图

计算机绘图能使你在一个较小的屏幕上画出任意大小的复杂图形。这是因为它的窗口－视口变换(Window-Viewport transformation)功能，就像照相机的“Zoom 变焦”或“Pan 摇晃”能够观察照片的任何细节。

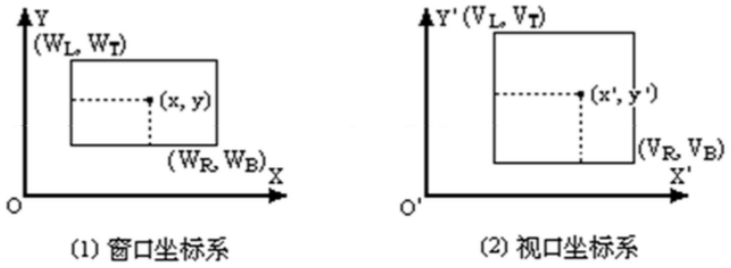


图 2.14　窗口－视口变换

编码基础

为了有效地判断并计算出给定线段在窗口内的图形部分，该算法采用对线段端点的二进制编码方式。区代码由 4 位二进制数组成，见下：。

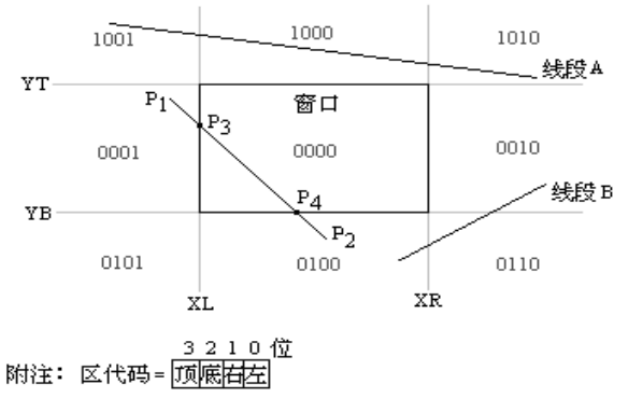


图 2.9　Cohen-Sutherland 编码裁剪法示意图

算法步骤

- 给定线段P1P2的端点所在的区域代码code1和code2
- 若code1=code2=0，则该线段完全在窗内，要保留它
- 若code1&code2不等于零，则说明线段完全在窗口外，应抛弃之
- 若code1&code2等于零，可求出交点
- 在对他的各个子线段重复执行以上步骤
  算法改进
- 按左右底顶(或相反)的边界线顺序进行计算
- 为了适合硬件实现，使用中点法主次确定可见段

几何造型(边界表示法)

算法概念

8.3.1　边界表示法(B-rep 实体)　　三 PP46-48

定义 8-3-1：顾名思义，B-rep(Boundary representation，边界表示法)是基于实体的边界信息，它将体表面拆成若干个“面—边—顶点”的层次表示。

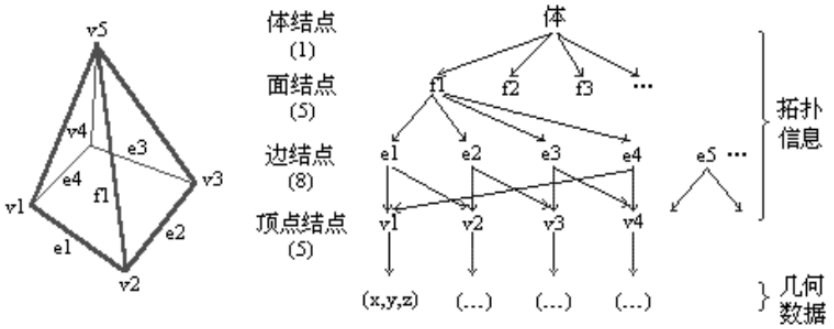


图 8.10　四面体的边界表示法

算法优缺点

综上所述，B-rep 实体造型的主要优缺点是：

- 信息丰富，实体的面、边、顶点及其关系的表示具有实用性。为了加快诸如求交计算、消隐处理等算法的执行速度，支持交互操作、图形显示或其他目的，这些数据都是很重要的。
- 适合增删点、边、环等的局部操作（即 **Euler** 操作）。
- 但从另一角度来看，B-rep 所存储的数据量大，而且有冗余度。在表示上要比下一节介绍的 CSG 实体模型麻烦得多。
- 对于空间寻址不是有效的。例如，确定一个给定点是否在体内的算法往往是十分复杂的。
- 需有专用程序来检查数据有效性。不适合反复的求交计算（如“并、交、差”）。

因此，有些以 B-rep 为主的造型系统（如 Spatial Technology 公司 ACIS 系统）中，常与 CSG 配合使用。

几何造型(构造性实体几何)

算法概念

定义 8-3-2：CSG(Constructive Solid Geometry)的字面意义是“构造性实体几何”，通常也称 体素造型 。CSG 意味着一个实体可通过体素的布尔运算来构造。

它源自于七十年代初美国 Rochester 大学研制的 PADL-1(Part and Assembly Description Language, 零件及装配描述语言)几何造型系统。

更一般地讲，CSG 实体可看作为一棵有序的二元树：

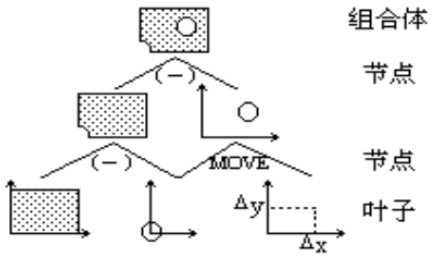


图 8.15　CSG 树示意图

算法优缺点

综上所述，CSG 模型的结构较简单、存储量小，实体表示精确，易于管理维护。但也有下列的缺点：

- **CSG** 树的层次较深，因而系统开销较大，处理速度慢。
- 体素种类较少，有些简单实体也需多次执行布尔运算才能被组合成。布尔运算与体素形状有密切的联系，各类算子并不是通用的。运算结果的封闭性有时也不能完全保证。
- 局部修改难，难于获取模型中的边信息。不能描述诸如雕塑体之类的形状；对于两个实体相互“粘合”产生的悬面，CSG 一般也处理不了。

曲线和曲面

Bezier曲线性质

1. 端点：$P(0) = P_0$，$P(1) = P_3$

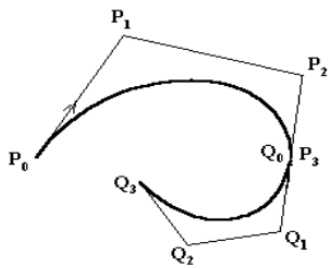
   Bézier 曲线通过特征多边形的始点和终点，而中间两点起着吸铁石的作用。

2. 切线方向：$P'(0) = 3(P_1 \quad P_0)$

   $P'(1) = 3(P_3 \quad P_2)$

   在两端点处的切线方向与特征多边形的第一条边和最后一条边的走向一致。

判则：n 次 Bezier 曲线在端点处的 r 阶导数，只与（r + 1）个相邻点有关，而与更远的点无关。（注：这里相邻点是指“包括端点本身在内”）



3. 反序性/对称性：若特征多边形各顶点的位置不变而输入次序反向（即 $P_3, P_2, \ldots P_0$），则该曲线保持不变。这适合于交互操作。
4. 其他公有性质：凸包性、几何不变性、保凸性、变差缩减性。（三 PP100-102）

参数曲线和曲面的一般性质

- 几何不变性
- 凸包性
- 透视变化不变性
- 保凸性
- 变差缩减性

Roberts基本判则

若物体上一个面的(外)法矢指向视点，则该面是可见的。

深度缓存(Z-缓存)算法

算法步骤

深度缓存算法的执行过程如下：

步骤一、初始时，深度缓存数组所有的元素 ZB[i, j]均被设置为最小 Z 值，如某机器最小值-2^{63}。

步骤二、对于屏幕上的每个像素 (i, j), $1 \leqslant i \leqslant m$ 和 $1 \leqslant j \leqslant n$，若它位于第 k 个（$1 \leqslant k \leqslant K$）面 F_k 的投影内，则计算该面上对应像素点的深度值 Z_{ij}：

$$Z_{ij} = -\frac{ai + bj + d}{c}$$，这里假定 F_k 平面方程为 $ax + by + cz + d = 0$

倘若 Zij > ZB[i, j], 这表示该面更接近于观察者，则要替换 ZB[i, j]中的深度值：

$$ZB[i, j] = Z_{ij}$$

同时将帧缓存中的对应元素 FB[i, j]修改为 F_k 的颜色值 C_{Fk}：

$$FB[i, j] = C_{Fk}$$

依此类推，直至所有的面处理完成。

算法优缺点

- 简单，不需要预排序
- 易于硬件实现
- 允许多个处理器进行并行处理
- 能用于处理对透明物体的消隐

Gourand光照模型算法

算法步骤

在图像空间中，**Gouraud** 光照模型（双线性亮度插值法）计算各像素的光强如下：

(1) 只在多边形的顶点处，按下式计算光强：(只考虑环境光和漫反射光)

$$\mathbf{I = K_a I_a + K_d I_L \cos\theta}$$

而顶点的法矢等于各邻面法矢的平均值：

$$\vec{N}_A = \frac{1}{n}\sum_{i=1}^{n}\vec{N}_i$$

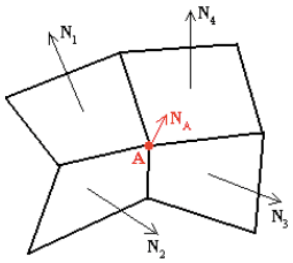


图 **12-12**　求法矢的平均值

(2) 对于多边形边上和内部的各点，用顶点明暗度的线性插值计算出：

$$\text{令 } \Delta I_{AB} = \frac{I_B - I_A}{y_B - y_A}, \quad \Delta I_{AD} = \frac{I_D - I_A}{y_D - y_A}$$

$$\text{故 } I_S = I_A + \Delta I_{AB} \bullet \Delta y, \quad I_T = I_A + \Delta I_{AD} \bullet \Delta y$$

对于多边形内部的各点，用两边端点光强的线性插值法计算出：

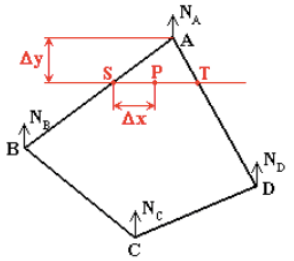


图 **12-13**　双线性亮度插值法

算法优缺点

- 算法简单，计算量小
- 不能再现高光
- 适用于粗糙表面

Phong光照模型算法

算法步骤

在图像空间中，**Phong** 光照模型（双线性法矢插值法）计算各像素的度如下：

① 顶点的法矢等于各邻面法矢的平均值，即 $\frac{1}{n}\sum_{i=1}^{n}\vec{N}_i$

对于多边形上和内部的各点(像素)，用顶点法矢的线性插值。计算公式如下：

令 $\Delta N_S = \frac{N_A - N_B}{y_A - y_B}$，　$\Delta N_T = \frac{N_A - N_D}{y_A - y_D}$

故 $N_{S,\,y+1} = N_{S,\,y} + \Delta N_S$，　$N_{T,\,y+1} = N_{T,\,y} + \Delta N_T$

令 $\Delta N_P = \frac{N_T - N_S}{x_T - x_S}$

故 $\Delta N_{x+1,\,P} = \Delta N_{x,\,P} + \Delta N_P$

注意：这里，法矢 Np 就是实际曲面上该点的切平面法矢的近似值

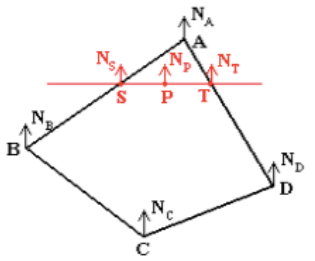


图 12-14　双线性法矢插值法

② 在多边形所有的点处按下式计算光强：(环境光+漫反射光+单向反射光)

$$I = K_a I_a + K_d I_L \cos\theta + K_s \cos^n \alpha$$

算法优缺点

- 真实感强
- 计算量大
- 适用于光滑表面

与Gourand对比

Gouraud 和 Phong 光照模型两者比较如下：

Gouraud Shading	Phong Shading
双线性光强插值法	双线性法向插值法
适用于漫反射光（粗糙表面）	适用于镜面反射光（特别是高光的金属表面）
马赫带效应较重	马赫带效应较轻
生成多面体真实感图象效果差	高光域准确
速度较快（算法简单、计算量小）	速度较慢（计算量较大）

Ray-Tracing光照跟踪算法

基本思想

Ray-Tracing 算法的基本思想如下：

（**1**）在物体空间中，假定视点取在 Z 轴上，XOY 平面取作为投影屏幕(其与显示器屏幕的像素布局一一对应)，通过跟踪多个光源对各像素的贡献而计算出它们的色彩明暗度。

（**2**）当光线遇到某一物体（不是背景或光源）时，如果光线对该交点处的光强贡献不趋于零而且跟踪深度并不很深，则应继续跟踪光线。

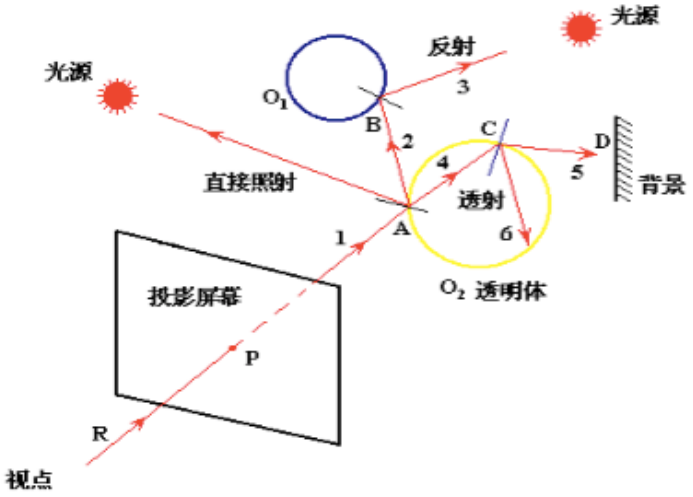


图 12-20　光线跟踪法

算法步骤

步骤一、 从视点穿过某像素 P 发出一条射线 R，它逆着光线方向跟踪所有光源在这个可见点（即离视点最近的交点）上产生的色彩明暗度：

$$I_A = I_{局部} + I_{全局}$$

其中 $I_{局部}$和 $I_{全局}$的具体计算按步骤二。

步骤二、 首先计算出可见点处的曲面法矢（并存储它，以供备用），然后查找表面数据表（其含有表面的颜色属性、粗糙度、反射率、透明度等）进行计算：（Hall 整体光照模型）

(1) $I_{局部} = K_a I_a + I_L(K_d \cos\theta + K_s \cos^{ns}\alpha + K_t \cos^{nt}\beta)$

即包括环境光、漫反射光、镜面反射光和透射光。

(2) $I_{全局} = K_R I_R + K_T I_T$

即由其他物体反射或折射到视线的反射或透视方向的光。

步骤三、判断可见点是否处在阴影(shadow)中：从该点向光源引射线。若射线与某

个不透明的物体相交，则该点在阴影中（此时只取环境光），而返回到步骤一。

算法优缺点

光线跟踪法的特点/优缺点：

光线跟踪法是目前最常用的高级渲染技术之一。

a. 同时处理消隐和渲染，并有透明效果和阴影生成。

b. Ray-Tracing 本质上是一种递归算法（层次等于物体个数），效率极高。但求交运算量大，也没有考虑物体表面的光辐射，。

算法加速方法：

事实上每条射线只与少数几个物体相交。若预先将物体按空间位置适当地组织起来（如采用包围体），则可缩小搜索范围，避免不必要的求交运算。

- 包围体(Bounding Box/Sphere)。仅当射线与根结点有交时，才进行它与子结点求交

在光线跟踪算法中，跟踪光线的终止条件有以下四个：

1) 被跟踪的光线与光源相交；
2) 被跟踪的光线遇到背景；
3) 被跟踪的光线对某交点处的光强贡献趋于零；
4) 光线跟踪的深度已经很深了。

参考资料

- UCAS计算机图形学课程