**光栅图形学算法总结**

姓名：王新钰

学号：201701101121

班级：电信17-4

**目录**

[一、光栅图形学概述 1](#_Toc35971232)

[二、重点算法总结 1](#_Toc35971233)

[1.直线段的扫描转换算法 1](#_Toc35971234)

[1.1 DDA画线算法 1](#_Toc35971235)

[1.2 中点画线算法 1](#_Toc35971236)

[1.3 Bresenham算法 2](#_Toc35971237)

[2 圆弧扫描算法 2](#_Toc35971238)

[2.1 中点画圆法 2](#_Toc35971239)

[2.2 Bresenham画圆算法 2](#_Toc35971240)

[2.3 生成圆弧的正负法 3](#_Toc35971241)

[3.多边形扫描转换与区域填充算法 3](#_Toc35971242)

[3.1 多边形的扫描转换算法 3](#_Toc35971243)

[3.2 区域填充算法 3](#_Toc35971244)

[4.反走样算法 4](#_Toc35971245)

[4.1 走样现象 4](#_Toc35971246)

[4.2 反走样算法 4](#_Toc35971247)

[5.裁剪算法 4](#_Toc35971248)

[5.1 直线裁剪算法 4](#_Toc35971249)

[5.2 多边形裁剪算法与文字裁剪算法 6](#_Toc35971250)

[6.消隐算法 6](#_Toc35971251)

[6.1 Z-buffer算法 6](#_Toc35971252)

[6.2 区间扫描线算法 6](#_Toc35971253)

[6.3 Warnock算法 6](#_Toc35971254)

[参考资料 7](#_Toc35971255)

一、光栅图形学概述

随着光栅显示器的出现，为了在计算机上处理、显示图形，需要发展一套与之相适应的算法，也就是光栅图形学算法，该算法详细的介绍在本课程中的第二、三和第四讲，光栅图形学算法属于计算机图形的底层算法。在本课程中，涉及到的重点内容有：直线段的扫描转换算法、圆弧扫描算法、多边形的扫描转换与区域填充算法、裁剪算法、反走样算法和和消隐算法。

二、重点算法总结

1.直线段的扫描转换算法

在数学上，直线上的点有无穷多个。但当在计算机光栅显示器屏幕上表示这条直线时需要做一些处理。而怎么处理数学上的直线到光栅显示器显示的直线的算法，就是直线段的扫描转换算法。确定最佳逼近于该直线的一组象素，并且按扫描线顺序，对这些象素进行写操作。从图元的数学/参数表示形式(由用户按需要指定）转换成适于光栅系统显示所需要的点阵表示形式

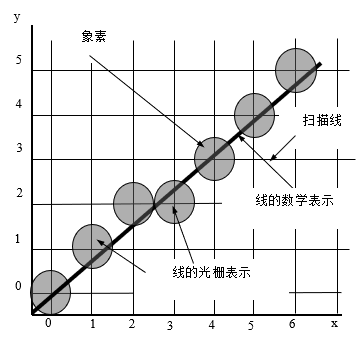


图 1 直线扫描算法

1.1 DDA画线算法

DDA画线算法也就是数值微分法。该算法引进图形学中一个很重要的思想：增量思想。

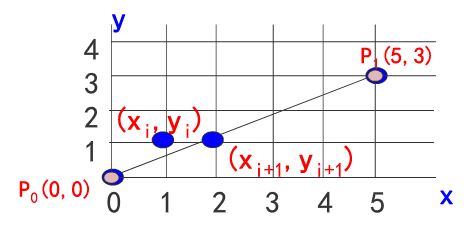
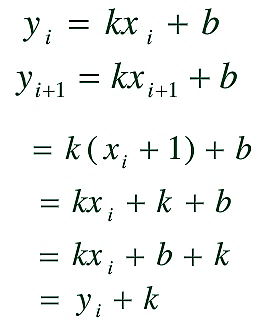
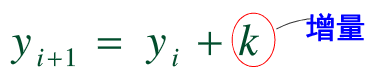


图 2 DDA画线算法







这个式子的含义是：当前步的y值等于前一步的y值加上斜率k这样就把原来一个乘法和加法变成了一个加法。

1.2 中点画线算法

1.2.1 中点画线算法初步

直线的一般方程为：

F(x,y)=0

Ax+By +C=0

其中:A=-(△y);B=(△x);C=-B(△x)

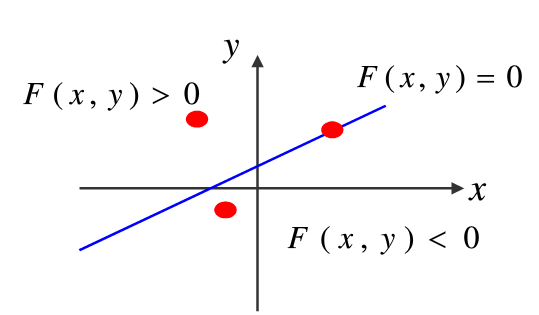


图 3 中点画线算法

对于直线上的点:F(x,y)=0

对于直线上方的点:F(x,y)>0

对于直线下方的点:F(x,y)<0

每次在最大位移方向上走一步，而另一个方向是走步还是不走步要取决于中点误差项的判断。

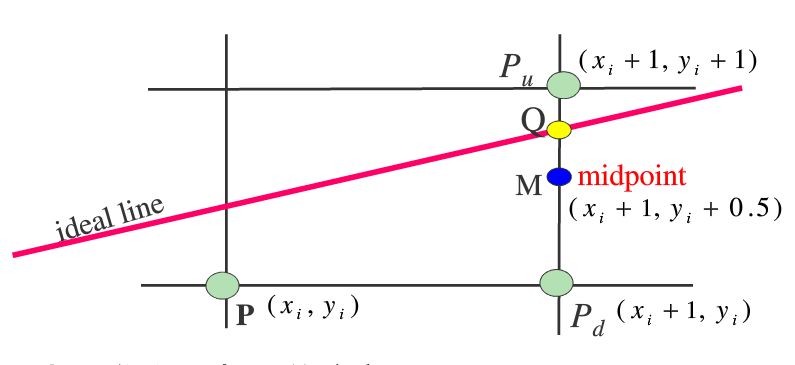
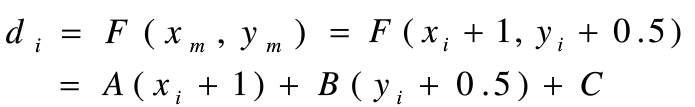


图 4 中点画线算法-2

构造



这样，只需要判断d就可以确定取点。

1.2.2 中点画线算法改进

由于中点画线算法计算量大，因此有改进的增量式中点画线算法。

1.3 Bresenham算法

DDA把算法效率提高到每步只做一个加法。中点算法进一步把效率提高到每步只做一个整数加法。Bresenham提供了一个更一般的算法。该算法不仅有好的效率，而且有更广泛的适用范围。

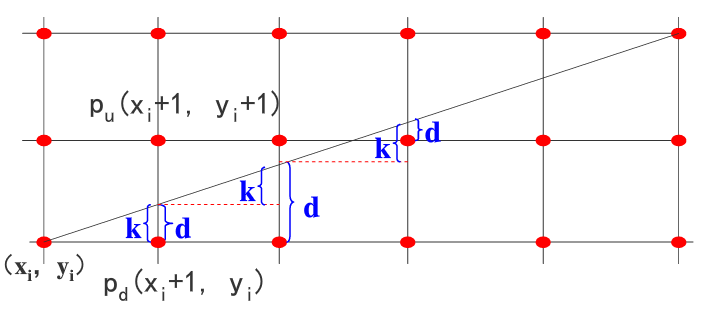
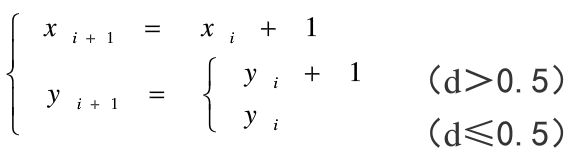


图 5 Bresenham算法

该算法的基本思想是：通过各行、各列像素中心构造一组虚拟网格线，按照直线起点到终点的顺序，计算直线与各垂直网格线的交点，然后根据误差项的符号确定该列象素中与此交点最近的象素。



该算法也有一系列的改进算法，从而减少了计算量。

算法步骤：

1输入直线的两端点P0(x,y)和P1(x1,y1)｡

2.计算初始值△x､△y､e=△x､x=xo､y=y

3绘制点(x,y)｡

4.e更新为e+2△y,判断e的符号｡若e>0,则(x,y)更新为(x+1,y+1),同时将e更新为e-2△x;否则(x,y)更新为5当直线没有画完时,重复步骤3和4｡否则结束｡

2 圆弧扫描算法

2.1 中点画圆法

利用圆的对称性，只须讨论1/8圆。

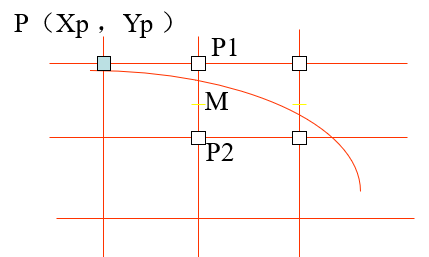


图 6 中点画圆法

P为当前点亮象素，那么，下一个点亮的象素可能是P1(Xp+1,Yp)或P2(Xp +1,Yp -1)。构造函数：

F（X，Y）=X2 + Y2 - R2 ；则

F（X，Y）= 0 （X,Y）在圆上；

F（X，Y）< 0 （X,Y）在圆内；

F（X，Y）> 0 （X,Y）在圆外。

设M为P1､P2间的中点,M=(Xp+1,Yp-0.5)

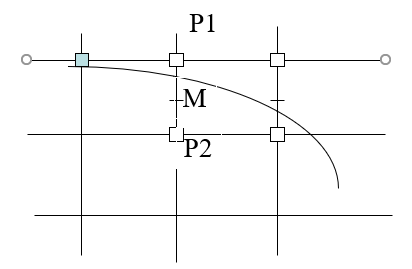


图 7 中点画圆法-2

有如下结论：

F(M)<0 -----M在圆内---取P1

F(M)>= 0---M在圆外---取P2

2.2 Bresenham画圆算法

原理与直线算法类似，都是用一系列离散的点来描述一个圆。

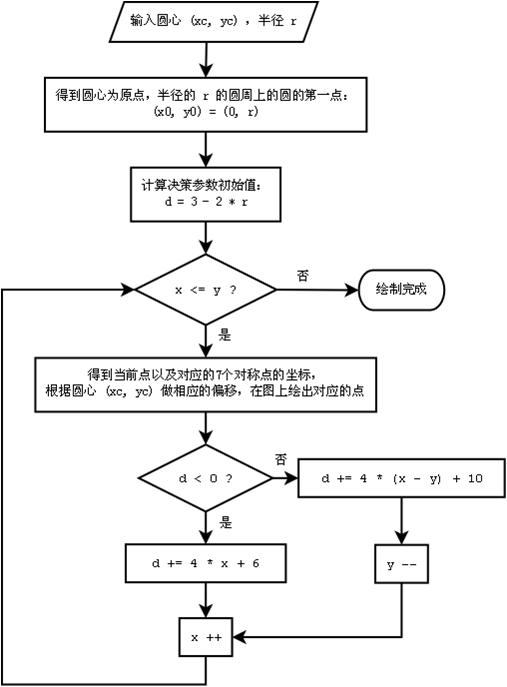


图 8 算法流程

2.3 生成圆弧的正负法

设圆的方程为F(x,y)=X2+Y2 -R2=0;

假设求得Pi的坐标为(xi,yi)；

则当Pi在圆内时-> F(xi,yi)<0 -> 向右-> 向圆外

Pi在圆外时-> F(xi,yi)>0 -> 向下-> 向圆内

即求得Pi点后选择下一个象素点Pi+1的规则为：

当F(xi,yi) ≤0 取xi+1 = xi+1，yi+1 = yi;

当F(xi,yi) ＞0 取xi+1 = xi， yi+1 = yi - 1;

这样用于表示圆弧的点均在圆弧附近，且使F(xi,yi) 时正时负，故称正负法。

3.多边形扫描转换与区域填充算法

3.1 多边形的扫描转换算法

3.1.1 X扫描线的算法

X-扫描线算法填充多边形的基本思想是按扫描线顺序，计算扫描线与多边形的相交区间，再用要求的颜色显示这些区间的像素，即完成填充工作。区间的端点可以通过计算扫描线与多边形边界线的交点获得。

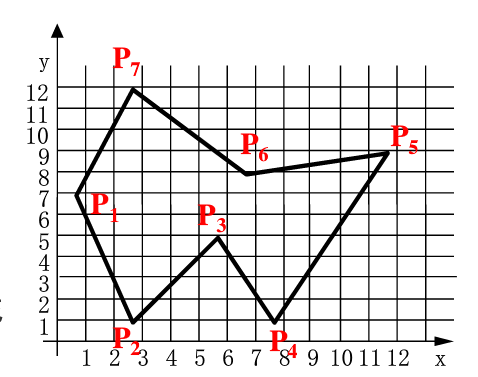


图 9 X线扫描算法

算法的核心是按X递增顺序排列交点的X坐标序列。由此，可得到X-扫描线算法步骤如下：

（1）确定多边形所占有的最大扫描线数，得到多边形顶点的最小和最大y值。

（2）从ymin到ymax 每次用一条扫描线进行填充。

（3）对一条扫描线填充的过程可分为四个步骤：

**a、求交**：计算扫描线与多边形各边的交点

**b、排序：**把所有交点按递增顺序进行排序

**c、交点配对**：第一个与第二个，第三个与第四个

**d、区间填色：**把这些相交区间内的像素置成不同于背景色的填充色

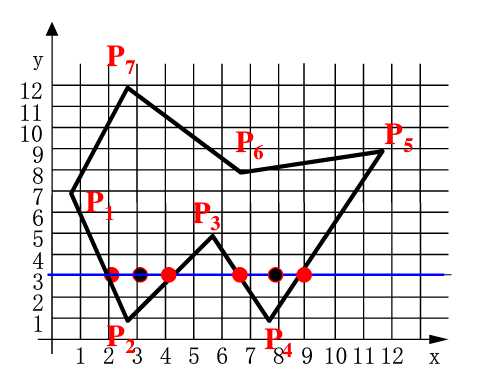


图 10 X线扫描算法-2

3.1.2改进的X扫描线的算法

传统的X扫描线的算法因为求交的计算量是非常大的所以提出了改进算法。

该算法主要是引进了新的数据结构来进行改进，由于不是重点内容，不做详细总结。

3.2 区域填充算法

区域，指已经表示成点阵形式的填充图形，是象素的集合。

区域填充是指将区域內的一点（常称种子点）赋予给定颜色然后将这种颜色扩展到整个区域内的过程。

区域可采用内点表示和边界表示两种表示形式：

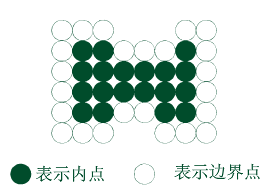


图 11 区域填充：内点和外点

**内点表示：**枚举出区域内部的所有像素，内部的所有像素着同一个颜色，边界像素着与内部像素不同的颜色

**边界表示：**枚举出边界上的所有像素，边界上的所有像素着同一个颜色，内部像素着与边界像素不同的颜色

区域可分为4向连通区域和8向连通区域。

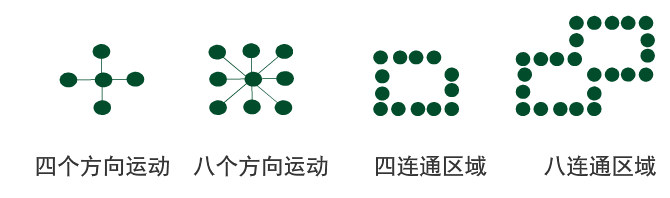


图 12 连通区域

3.2.1 简单的种子填充算法

种子填充算法的原理是：假设在多边形区域内部有一像素已知，由此出发找到区域内的所有像素，用一定的颜色或灰度来填充。假设区域采用边界定义，即区域边界上所有像素均具有某个特定值，区域内部所有像素均不取这一特定值，而边界外的像素则可具有与边界相同的值。考虑区域的四向连通，即从区域上一点出发，可通过四个方向，即上、下、左、右移动的组合，在不越出区域的前提下，到达区域内的任意像素。

3.2.2 使用栈结构来实现简单的种子填充算法

算法原理如下

种子像素入栈，当栈非空时重复执行如下三步操作：

(1)栈顶像素出栈

(2)将出栈像素置成要填充色

(3)按左､上､右､下顺序检查与栈像素相邻的四个像素，若其中某个像素不在边界且未置

4.反走样算法

4.1 走样现象

是光栅图形产生的阶梯形（锯齿形）二是图形中包含相对微小的物体时，这些物体在静态图形中容易被丢弃或忽略，小物体由于“走样”而消失。

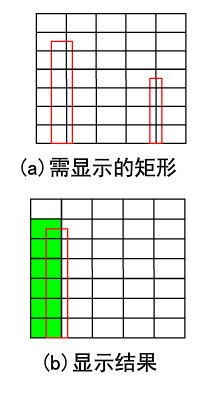


图 13 走样现象

简单地说，如果对一个快速变化的信号采样频率过低，所得样本表示的会是低频变化的信号：原始信号的频率看起来被较低的“走样”频率所代替。

4.2 反走样算法

4.2.1 非加权区域采样方法

根据物体的覆盖率(coverage)计算像素的颜色。覆盖率是指某个像素区域被物体覆盖的比例

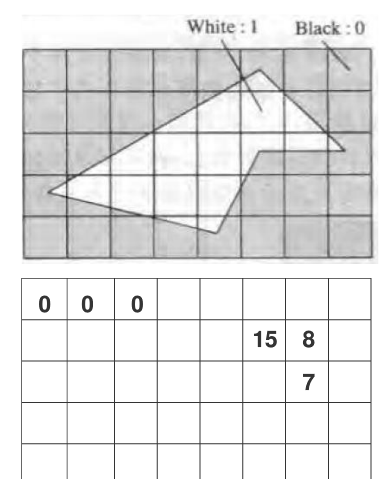


图 14 非加权区域采样算法

4.2.2 加权区域采样方法

这种方法更符合人视觉系统对图像信息的处理方式，反走样效果更好。

利用权函数积分求相交区域面积，用它乘以像素可设置的最大亮度值，即可得到该像素实际显示的亮度值。**采用离散计算方法。**

5.裁剪算法

5.1 直线裁剪算法

5.1.1 Cohen-Sutherland算法

本算法又称为编码裁剪算法，算法的基本思想是对每条直线段分三种情况处理：

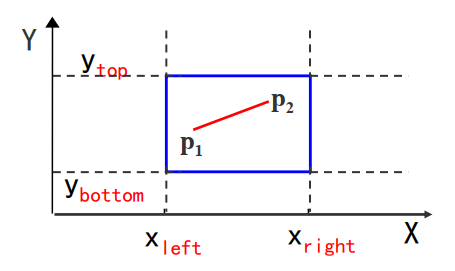
（1）若点p1和p完全在裁剪窗口内，简取

图 15 第一种情况

(2) 若点p1(x1,y1)和p2(x2,y2)均在窗口外，且满足四个条件之一：简弃

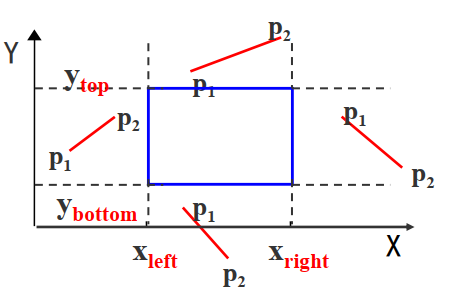


图 16 第二种情况

（3）如果直线段既不满足“简取”的条件，也不满足“简弃”的条件需要对直线段按交点进行分段，分段后判断直线是“简取”还是“简弃。

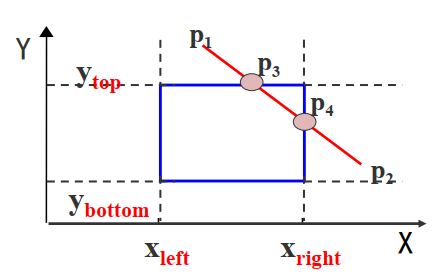


图 17 第三种情况

**编码规则：**

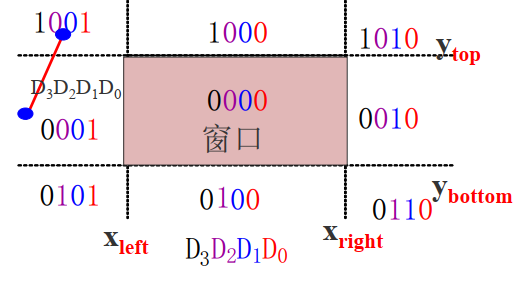
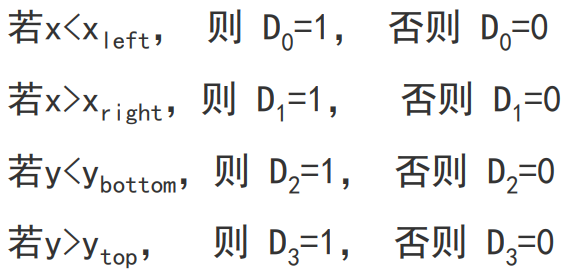


图 18 编码规则示意图



根据这个规则,D0对应窗口左边界,D1对应窗口右边界,D2对应窗口下边界,D2对应窗口上边界｡经过二进制运算后,决定取舍｡

有以下规则：

**code1|code2=0 简取**

**code &code2≠0 简弃**

若都不成立，则由直线段与窗口边界交点处分段后再进行取舍。

5.1.2中点分割法

该算法首先对直线段的端点进行编码。核心思想是二分逼近。有三种情况：

1、完全在窗口内

2、完全在窗口外

3、和窗口有交点

前两种情况比较简单，其中有交点的情况要根据中点是否处于窗口内来进行讨论。

1、若中点不在窗口内，则把中点和离窗口边界最远点构成的线段丢掉，以线段上的另一点和该中点再构成线段求其中点

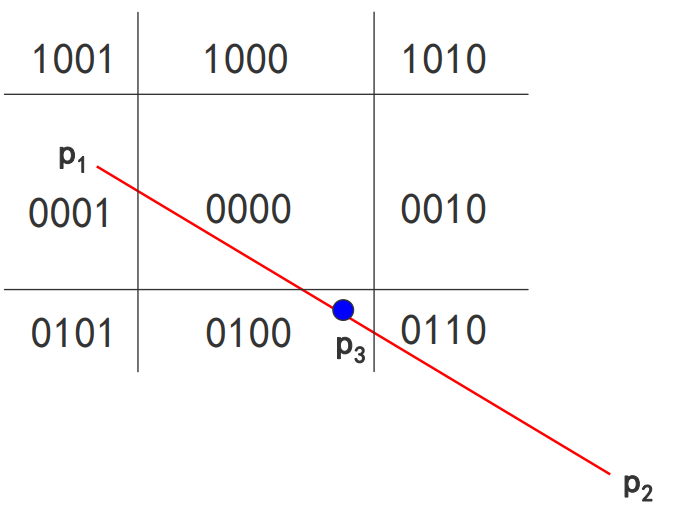


图 19 中点不在窗口内

2、如中点在窗口内,则又以中点和最远点构成线段,并求其中点,直到中点与窗口边界的坐标值在规定的误差范围内相等

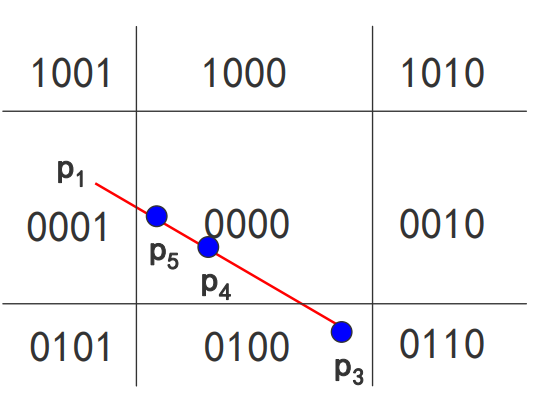


图 20 中点在窗口内

5.1.3 Liang-Barsky裁剪算法

上世纪80年代，梁友栋先生提出了著名的 Liang-Barsky算法至今仍是计算机图形学中最经典的算法之一，也是写进国内外主流《计算机图形学》教科书里的唯一一个以中国人命名的算法。该算法这里不做详细描述

5.2 多边形裁剪算法与文字裁剪算法

多边形裁剪算法的输出是裁剪后的多边形边界的顶点序列，主要有Suther and-Hodgeman多边形裁剪。

文字裁剪主要有以下几种：串精度裁剪、字符精度裁剪、笔划/像素精度裁剪。

6.消隐算法

当我们观察空间任何一个不透明的物体时，只能看到该物体朝向我们的那些表面，其余的表面由于被物体所遮挡我们看不到。如果把可见和不可见的线都画出来，对视觉会造成多义性。要消除二义性，就必须在绘制时消除被遮挡的不可见的线或面，习惯上称作消除隐藏线和隐藏面，简称为消隐。

按消隐对象分类，有线消隐和面消隐；按照消隐空间分类，有物体空间的消隐算法和图像空间的消隐算法；主流算法是图像空间的消隐算法；

6.1 Z-buffer算法

该算法有帧缓冲器和深度缓冲器｡对应两个数组intensity(x,y)——属性数组(帧缓冲器)存储图像空间每个可见像素的光强或颜色depth(x,y)—深度数组(Z-buffer)存放图像空间每个可见像素的z坐标｡

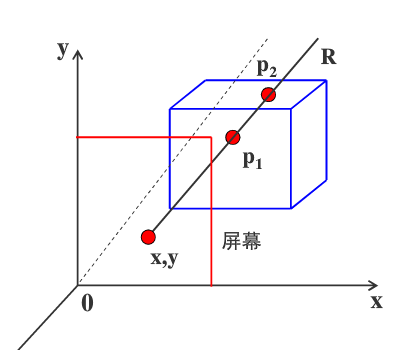
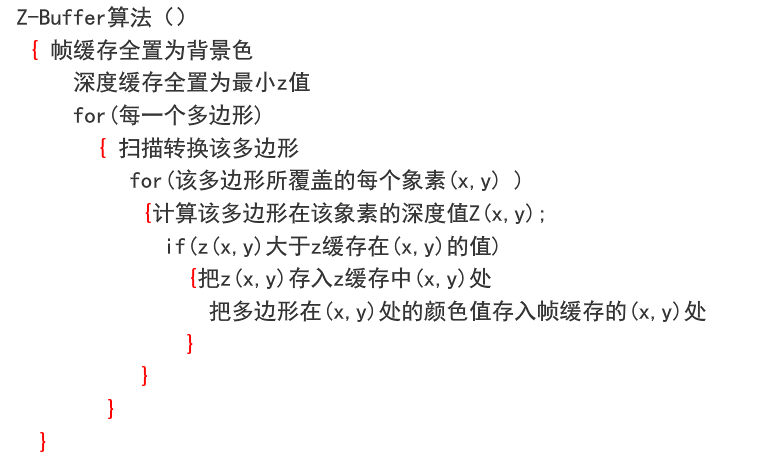


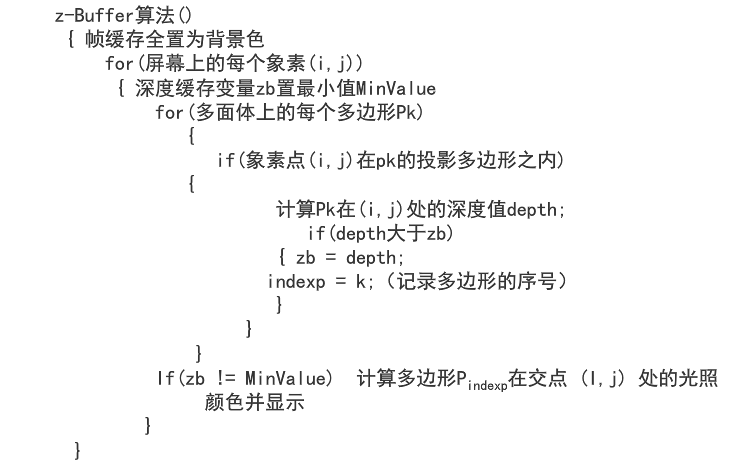
图 21 Z-buffer 算法

比较p1和p2的z值将最大的z值存入z缓冲器中,显然,p1在p2前面,屏幕上(x,y)这一点将显示p1点的颜色｡

算法实现：



改进的算法：



6.2 区间扫描线算法

首先要有投影多边形,然后求交点,然后交点进行排序排序,排序的结果就分成了一个个区间,然后在每个区间找当中的个像素(i,j),在(i,j)处计算每个相关面的z值,对相关深度值z进行比较,其中最大的一个就表示是可见的｡整个这段区间就画这个z值最大面的颜色｡

对于小区间有以下几点

(1)小区间上没有任何多边形,如[a4,a5],用背景色显示。

(2)小区间只有一个多边形,如[a1,a2],显示该多边形的颜色。

(3)小区间上存在两个或两个以上的多边,比如[a6,a7],必须通过深度测试判断哪个多边形可见。

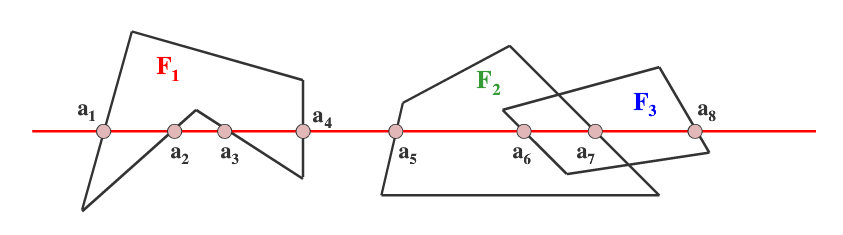


图 22 区间扫描线算法

6.3 Warnock算法

Warnock算法的重要性不在于它的效率比别的算法高，而在于采用了分而治之的思想，利用了堆栈的数据结构把物体投影到全屏幕窗口上，然后递归分割窗口，直到窗口内目标足够简单，可以显示为止。

主要有以下几种情况

(1)窗口中仅包含一个多边形

(2)窗口与一个多边形相交,且窗口内无其它多形

(3)窗口为一个多边形所包围

(4)窗口与一个多边形相分离

**算法步骤：**

(1)如果窗口内没有物体则按背景色显示

(2)若窗口内只有一个面,则把该面显示出来

(3)否则,窗口内含有两个以上的面,则把窗口等分成四个子窗口｡对每个小窗口再做上述同样的处理这样反复地进行下去

(4)窗口内含有两个以上的面,则把窗口等分成四个子窗口｡对每个小窗口再做上述同样的处理｡这样反复地进行下去, 把四个子窗口压在一个堆栈里后进先出。

参考资料

整理自中国大学MOOC课程提供的内容