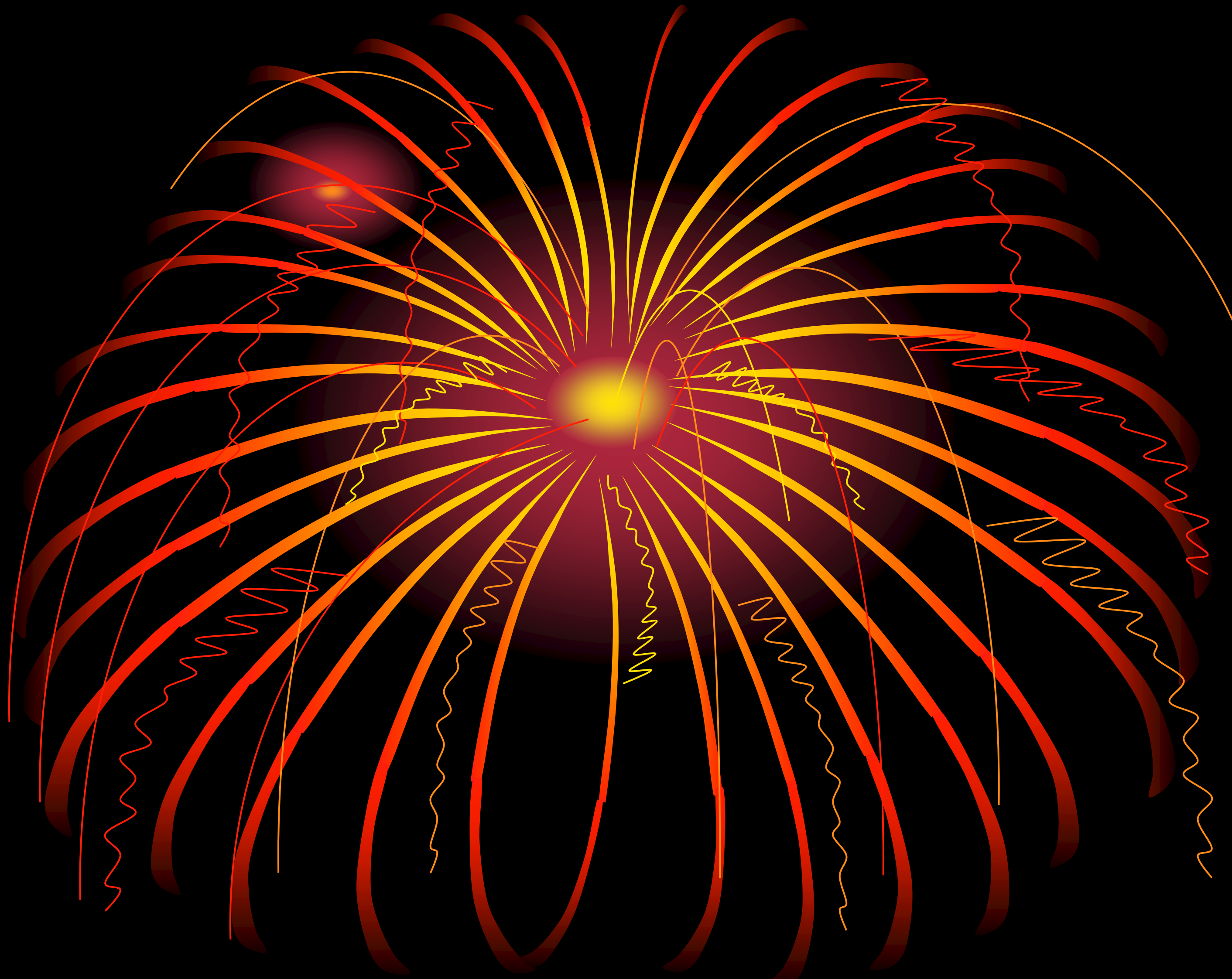
计算机图形学



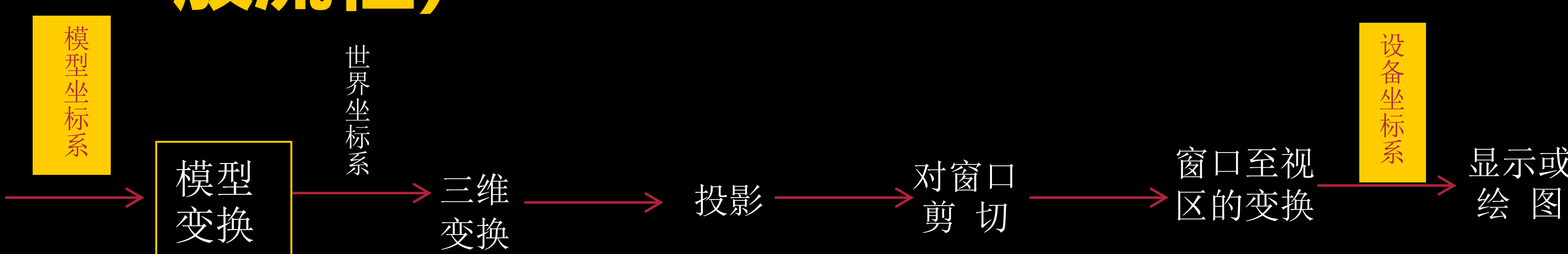
第一章 计算机图形学概述

1、图形的概念－－广义图形、计算机图形概念

计算机图形学是研究如何在计算机环境下描述、交互处理和绘制图形的一门学科。

2、计算机图形学的应用

3、计算机图形的生成过程（三维图形显示的一般流程）



图形的显示流程

第三章 计算机图形系统及硬件基础

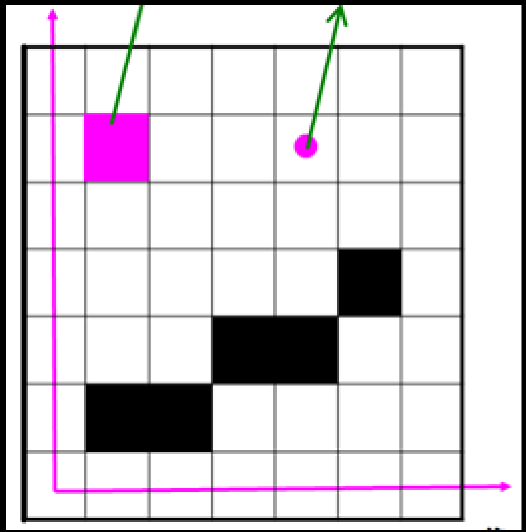
- 1、计算机图形系统的构成及基本功能

(计算功能/存储功能/输入功能/输出功能/交互功能)

- 2、三种显示器的基本工作原理

(CRT(随机扫描式、光栅扫描式)、液晶、等离子)

- 3、图形绘制设备、输入设备



第四章 基本光栅图形算法

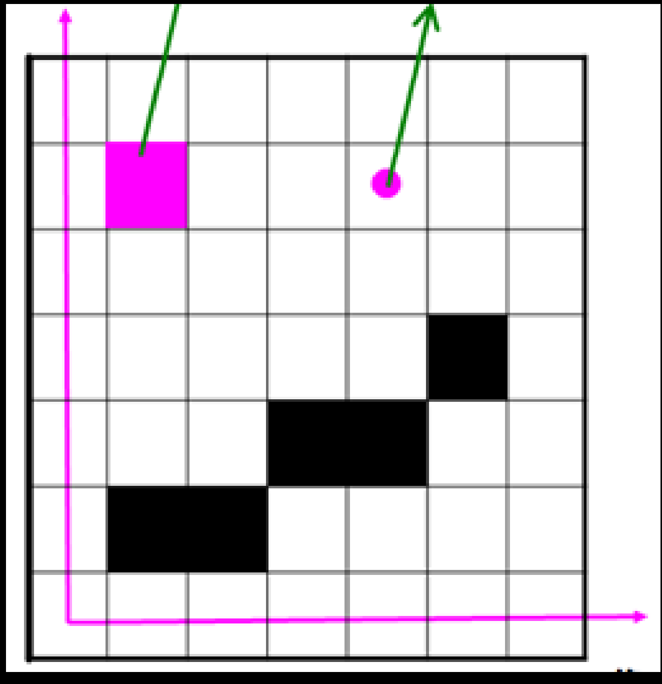
- 1、直线生成算法

  (DDA算法、正负法和Bresenham算法)

- 2、圆弧生成算法

  (正负法和Bresenham算法)

- 3、多边形的填充
- 4、区域填充



4.1.1生成直线的DDA算法

设直线的起点为(x_s,y_s),终点为（x_e,y_e），并设(x_s,y_s)和（x_e,y_e）都是整数（做求整处理）。

令$\Delta x= x_e-x_s$,$\Delta y= y_e-y_s$。则直线的参数方程是

$$\begin{cases} x = x_s + \Delta x t \\ y = y_s + \Delta y t \end{cases} \quad 0 \le t \le 1 \qquad (4.1)$$

目标是能快速地求出能很好地表示直线的像素。

❖**提高速度的方法之一是:乘法用加法实现，**用等步长计算直线上的点 。

即每一个点坐标都可以由前一个坐标变化一个增量得到。

$$\begin{aligned} x_{i+1} &= x_i + \Delta x \cdot \Delta t \\ y_{i+1} &= y_i + \Delta y \cdot \Delta t \end{aligned} \qquad \Delta t = t_{i+1} - t_i \qquad (4.2)$$

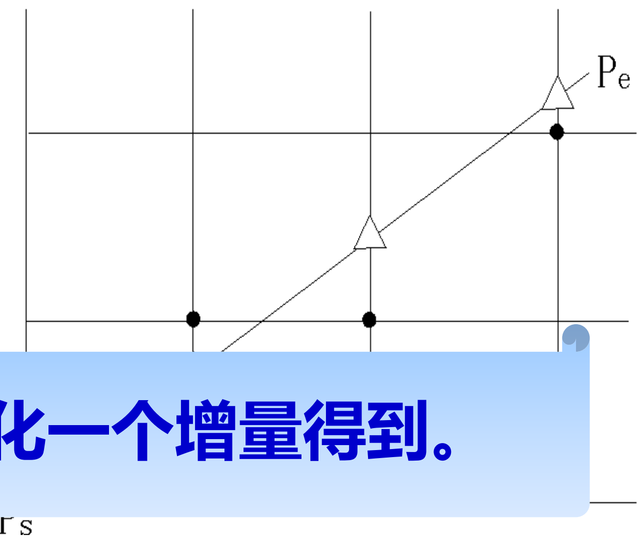


图4.2　图中黑圆点表示用DDA法生成的直线

正负法算法的具体实现

设直线的起点和终点分别为(x_s, y_s)和(x_e, y_e)，直线方程为：

$$\frac{x-x_s}{x_e-x_s}=\frac{y-y_s}{y_e-y_s}$$

$$F(x,y)=ax+by+c=0$$

其中 $a=y_s-y_e$ ，$b=x_e-x_s$ ，$c=x_sy_e-y_sx_e$

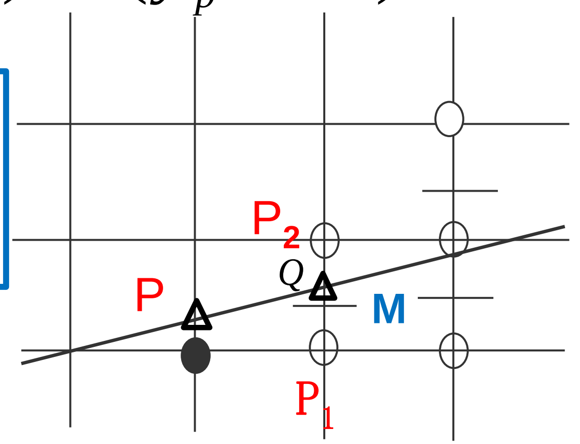
$F(x,y)=0$、 $F(x,y)>0$和 $F(x,y)<0$

分别对应于点(x,y)在**直线上**、**直线上方**和**直线下方**

所以，要判断M在直线的上方还是下方，只需把M代入，判断它的符号即可。
构造**判别式**：

$$d=F(M)=F(x_p+1,y_p+0.5)=a(x_p+1)+b(y_p+0.5)+c$$

d<0，Q 在 M 上方，取 P_2为下一像素
d>0，取 P_1为下一像素，
d=0，可在两个点中任取一个，约定取右下方的点



4.1.3 Bresenham算法

令$m=\triangle y/\triangle x$，考虑$0\le m\le 1$

$$y = m(x - x_s) + y_s$$

点斜式方程 $\boxed{y - y_1 = m(x - x_1)}$

x方向增加1，y方向增加m

所以有，$y_{i+1}=y_i+m(x_{i+1}-x_i)=y_i+m$ （4.5）

由起点$(x_s，y_s)$,可求得直线上的点$(x_i，y_i)$, i=1,2,3,…, 其中 $x_1= x_s, y_1= y_s$

用像素点$(x_i$，round$(y_i))$来表示直线上的点：

round(y_i)表示最靠近y_i的整数。令 $y_{i,r}$ =round(y_i)，即用**像素点$(x_i, y_{i,r})$**来表示直线上的点。

如何快速地求出直线上的点？

设B点是**直线上的点**，其坐标为$(x_{i+1}，y_{i+1})$，则**像素点$(x_{i+1}, y_{i+1,r})$**只能从C或D点中选。

设A为CD边的中点，若B在A点上面则应取D点作为$(x_{i+1}, y_{i+1,r})$, 否则应取C点。

令 $\varepsilon(x_{i+1})=y_{i+1}-y_{i,r}-0.5$

$\varepsilon(x_{i+1})$用来判断取C或D

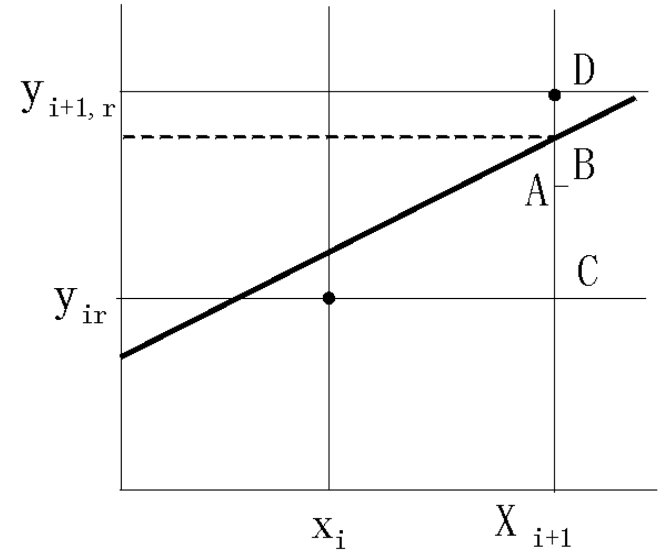


图4.4 ε(x)的几何意义

4.2圆弧生成算法

本节仅考虑圆心在原点的情况

- ◆正负法
- ◆Bresenham法
- ◆多边形逼近法

4.2.1正负法

设圆的圆心在$(0,0)$,半径为R,则圆的方程为

$F(x,y)=x^2+y^2-R^2=0$

❖ 当点(x,y)在圆内时，$F(x,y)<0$，向圆外走一步。

❖ 当点(x,y)在圆外时，$F(x,y)>0$，向园内走一步。

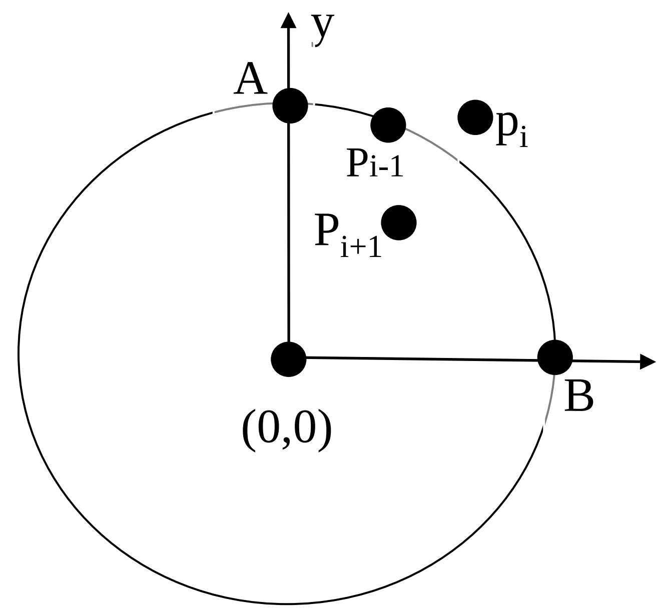


图4.8 对弧AB上点的取法

4.2.2 Bresenham 生成圆弧的算法

假设圆心$(0,0)$为原点

考虑AB弧的画法，显示一个整圆时，只要在显示AB上任一点(x,y)时，同时显示在圆周上其它七个对称点

$(y,x), (y,-x),(x,-y),(-x,-y),(-y,-x),(-y,x),(-x,y)$

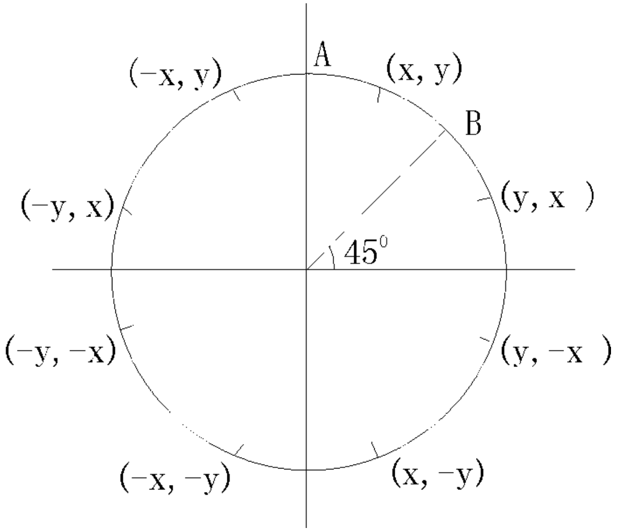


图4.9 七个对称点

从A点开始寻找弧AB上要用的点

设P_{i-1}是已选中的一个表示圆弧上的点，下一个点应从 H_i 或 L_i 中选择。设 H_i 和 L_i 两点的坐标分别为(x_{hi},y_{hi})和(x_{li},y_{li}) 考虑应选哪一个

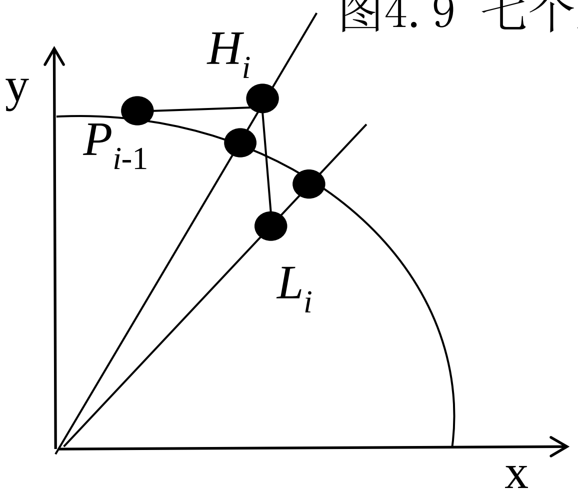


图4.10 两个候选点

基本思路： 通过比较临近像素点到圆弧的距离，设法求出该距离的递推关系，并通过符号判别像素取舍。

第四章 基本光栅图形算法

- 1、直线生成算法
- 2、圆弧生成算法
- 3、多边形的填充

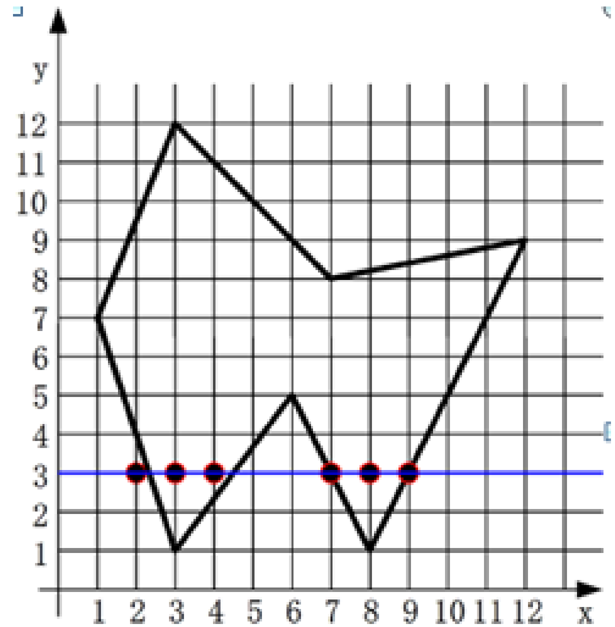
  3.1 多边形的两种表示方法（顶点表示/点阵表示）
  3.2 多边形填充的扫描线算法（基本思想、性质、奇异点的处理、数据结构和实现步骤）
- 4、区域填充

  区域的概念及两种表示方法(内点表示/边界表示)

4.3.2多边形填充的扫描线算法

从该例可以看出，算法的核心是需要按X递增顺序排列交点的X坐标序列，由此可得扫描线算法步骤：

- 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值（Ymin和Ymax）
- 2.从Ymin到Ymax，每次用一条扫描线进行填充
- 3.对一条扫描线的填充过程可分为：
  - 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X值递增顺序排序。
  - c.交点配对：12、34等，每对就代表扫描线与多边形的一个相交区间
  - d.区域填色：把这些相交区间内的像素置成不同于背景色的填充色。



5. 扫描线算法的数据结构与实现步骤

该数据结构由边y筒ET和边的活化链表AEL(Active Edge List)两部分组成。

ET和AEL中的多边形的边由四个域组成：

y_{max}　边的上端点的y坐标；

x　在ET中为边的下端点的x坐标，在AEL中是边与扫描线交点的x坐标

Δx　边的斜率的倒数，即$\frac{1}{m_{i_r}}$

next　指向下一条边的指针。

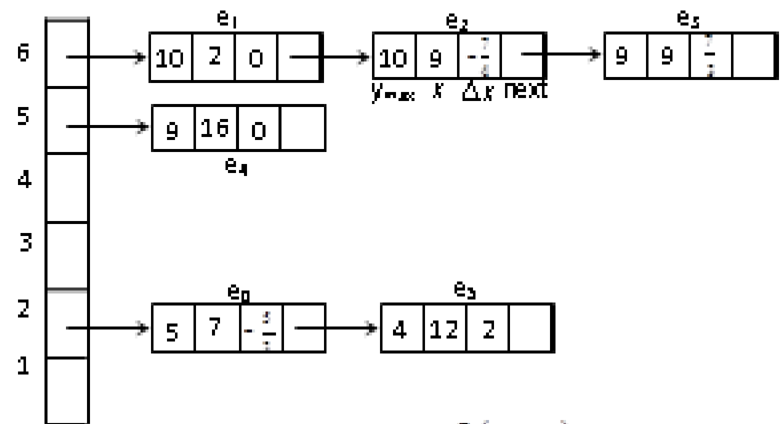


图 4.19 边y筒

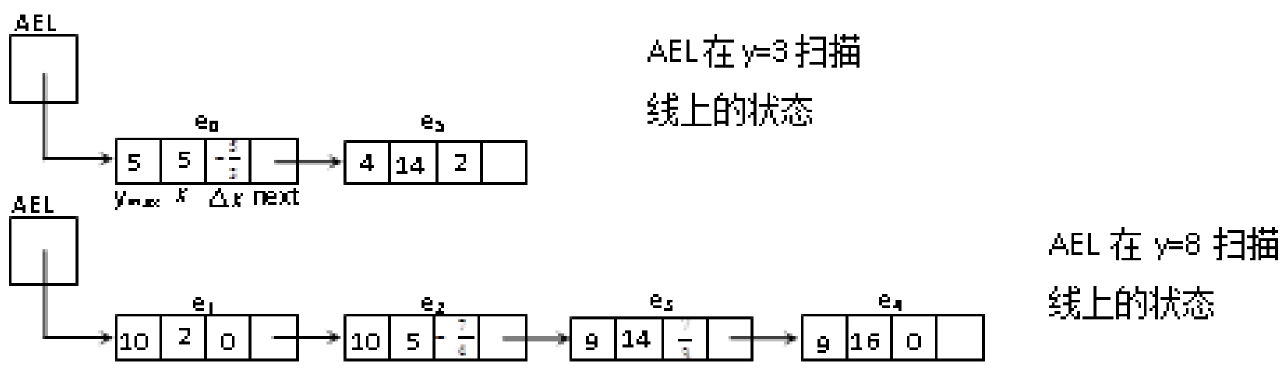


图 4.20 边的活化链表

边y筒ET数据结构

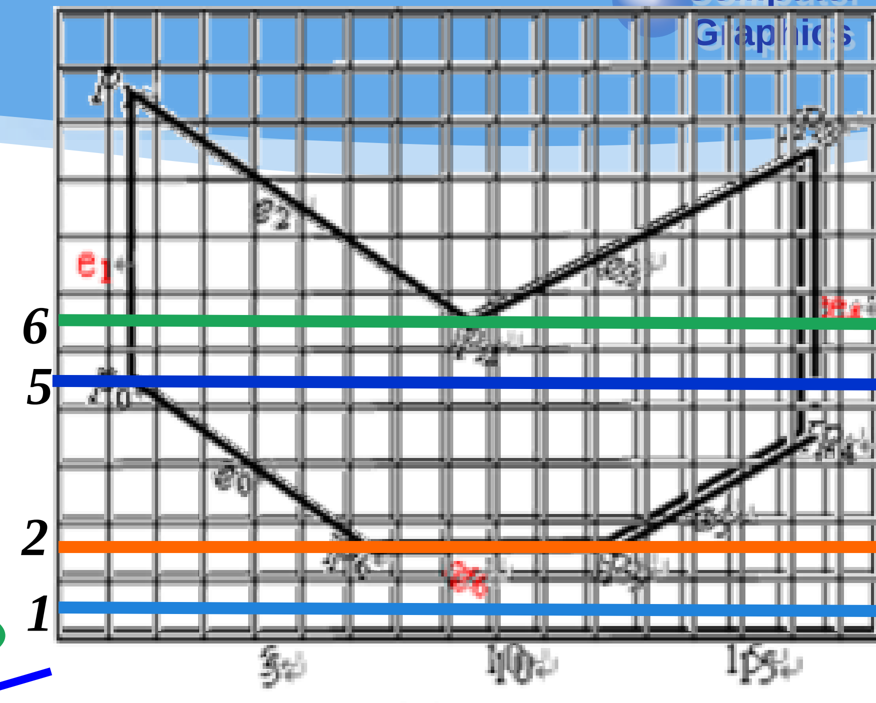
边的y筒ET是按边下端点的纵坐标*y*对非水平边进行分类的指针数组，数组的长度等于扫描线的总数。**下端点的纵坐标y=i的边归入第i类。**

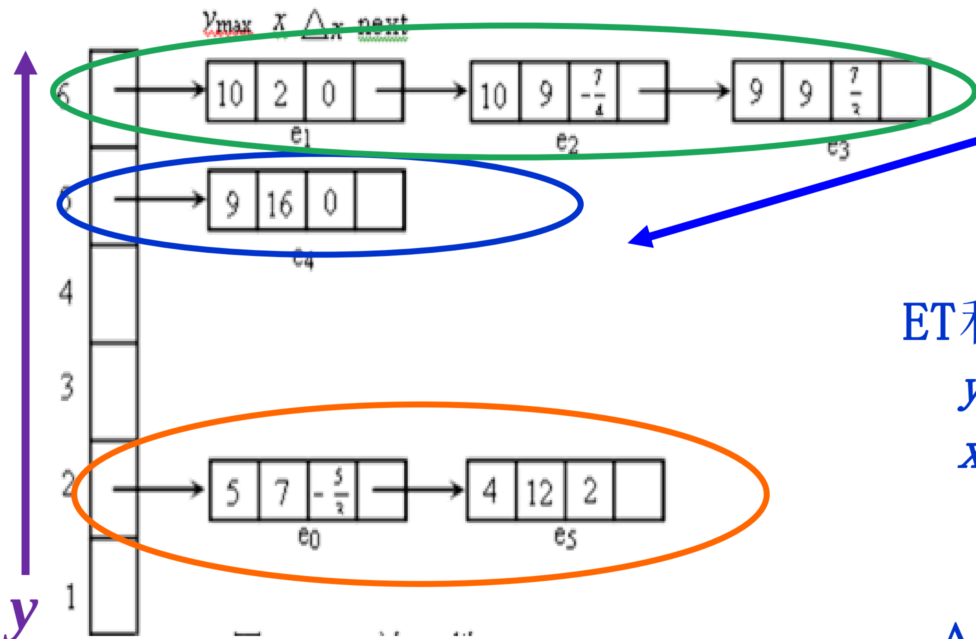
同一类中，各边按*x*值递增的顺序排列成行。

对于右图中的多边形：

$[P_0P_1P_2P_3P_4P_5\ P_6]=$

$[(2,5)\ (2,10)\ (9,6)\ (16,11)\ (16,4)\ (12,2)\ (7,2)]$





ET和AEL中的多边形的边由四个域组成：

y_{max} 边的上端点的*y*坐标；

x 在ET中为边的下端点的*x*坐标，在AEL中是边与扫描线交点的*x*坐标

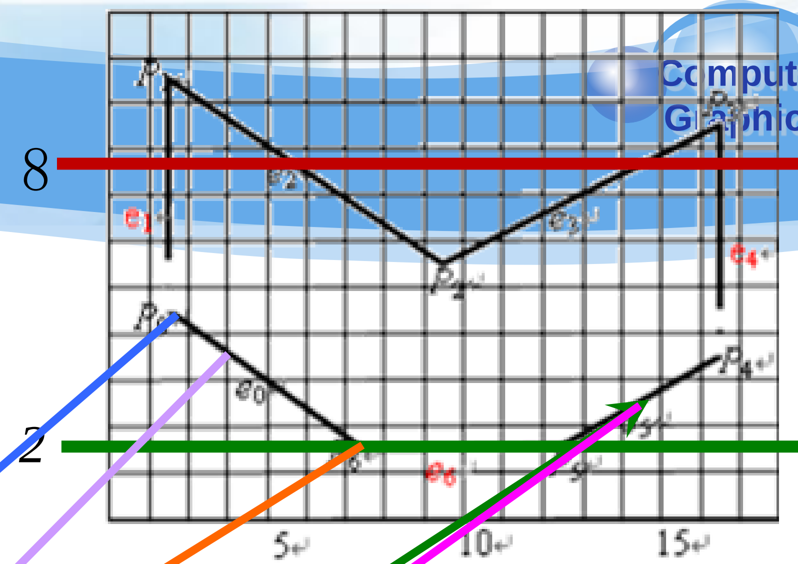
Δx 边的斜率的倒数

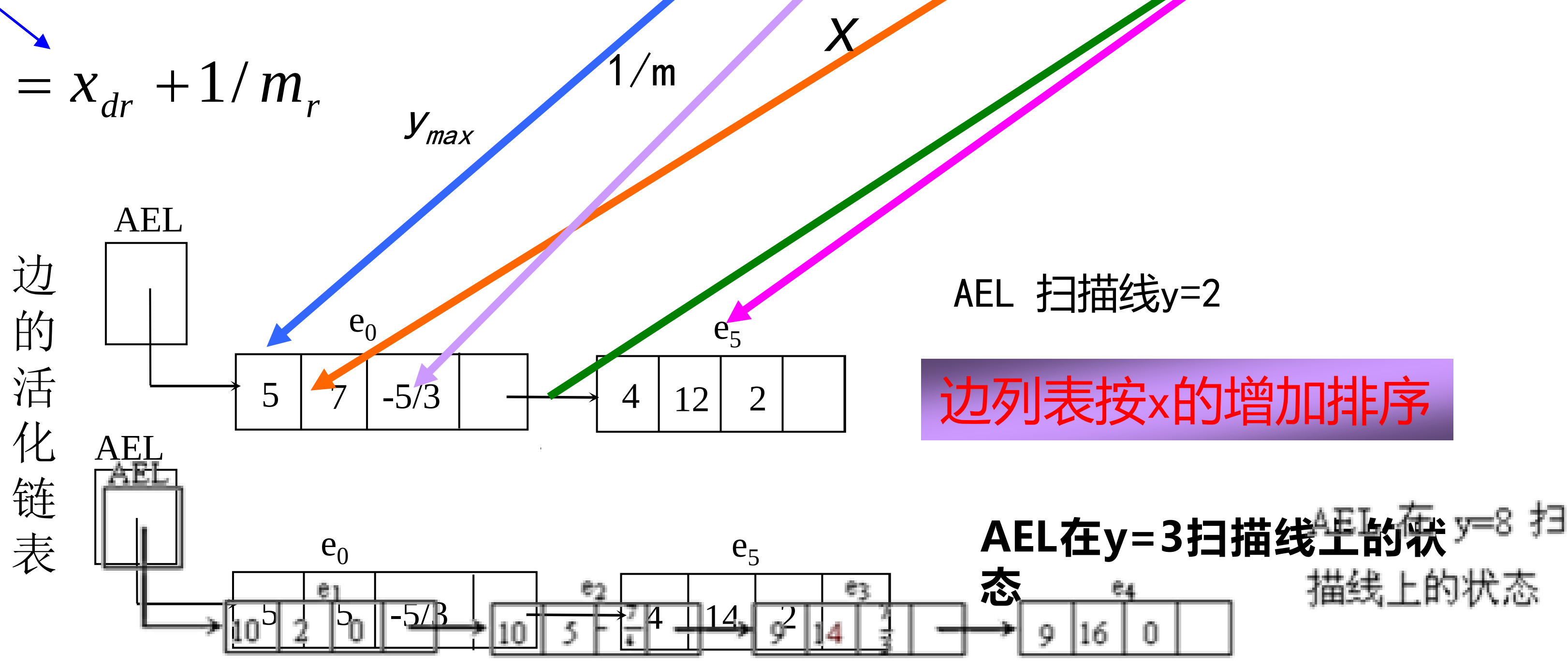
next 指向下一条边的指针。

边的活化链表AEL数据结构

边的活化链表AEL由**与当前扫描线相交的所有多边形的边**组成，它记录了多边形边沿扫描线的交点序列，并根据递推关系式不断地更新交点序列

$$x_{er} = x_{dr} + 1/m_r$$





扫描线算法的步骤：

- **步骤1:(AEL初始化)将活化链表AEL设置为空。**
- **步骤2:(y初始化)取扫描线纵坐标y的初始值为ET中非空元素的最小序号,**在图4. 19中y=2。
- **步骤3：按从下到上的顺序对纵坐标值为y的扫描线(当前扫描线)执行下列步骤，直到边的活化链表AEL都变成空为止。**

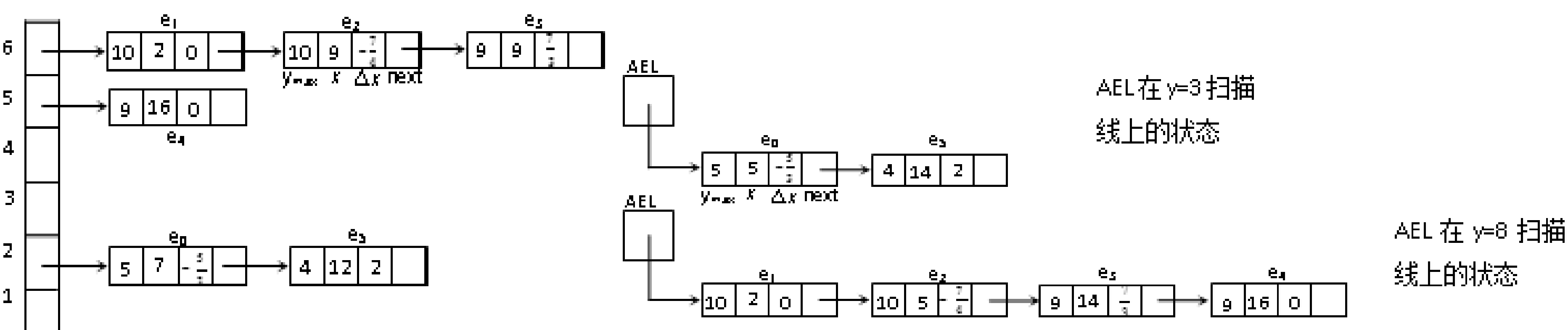
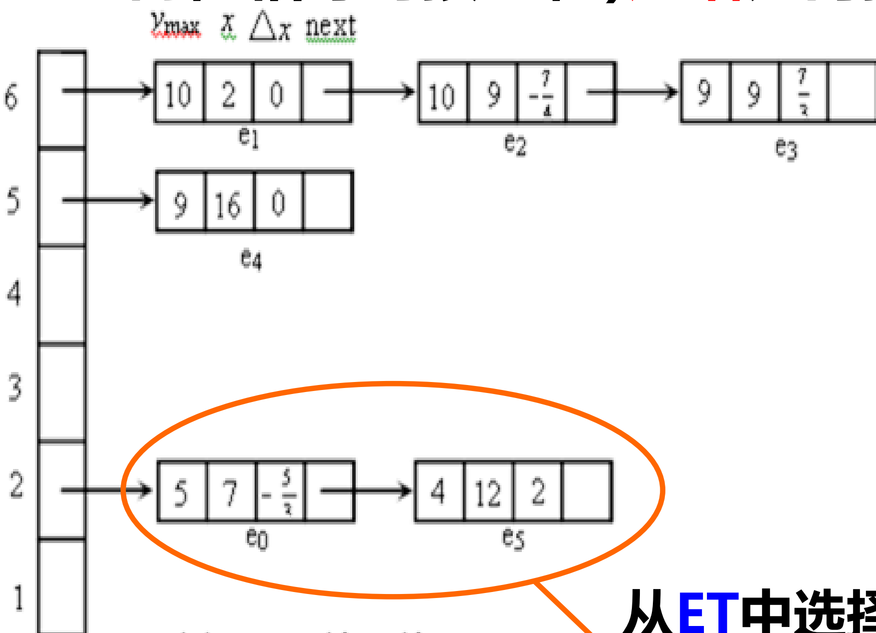
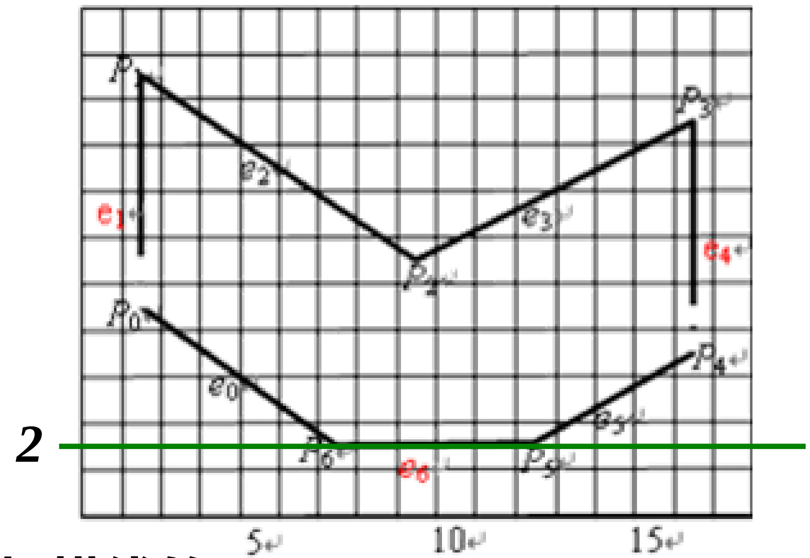


图 4.19 边y筒　　图 4.20 边的活化链表

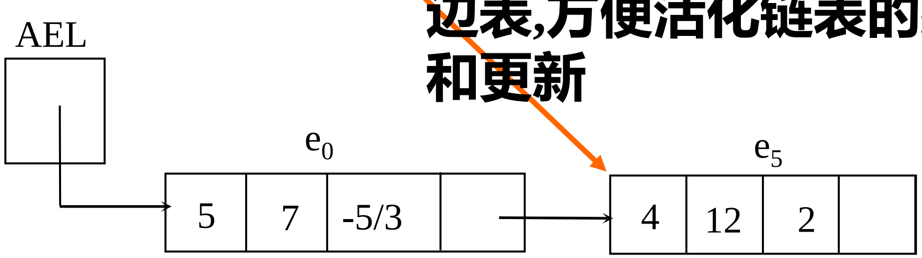
扫描线算法的具体步骤：

- ①如边ET中的第y类元素非空，则将属于该类的所有边从ET中取出并插入活化链表AEL中，AEL中的各边按照x值(当x的值相等时按Δx值)递增方向排序。





从ET中选择当前扫描线的边表,方便活化链表的建立和更新

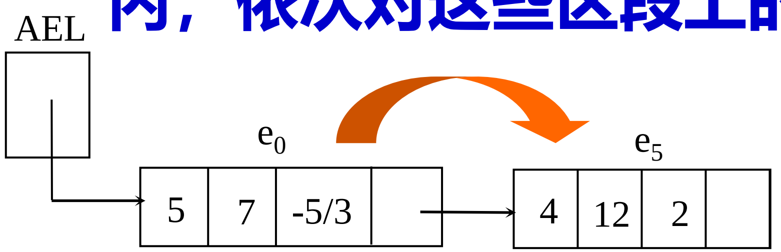


AEL在y=2扫描线上的状态

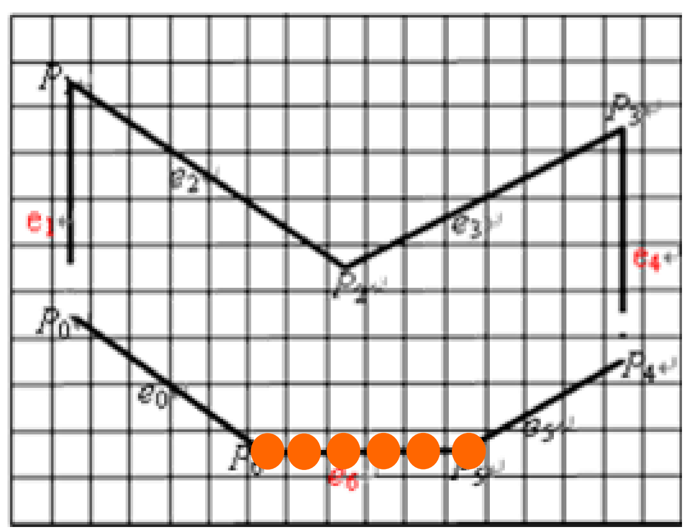
扫描线算法的具体步骤：

- ②若相对于当前扫描线，其活化链表AEL非空，则将AEL中的边两两依次配对，即第1，2边为一对，第3，4边为一对，依此类推。

每一对边与当前扫描线的交点所构成的区段位于多边形内，依次对这些区段上的点(像素)按多边形属性着色。

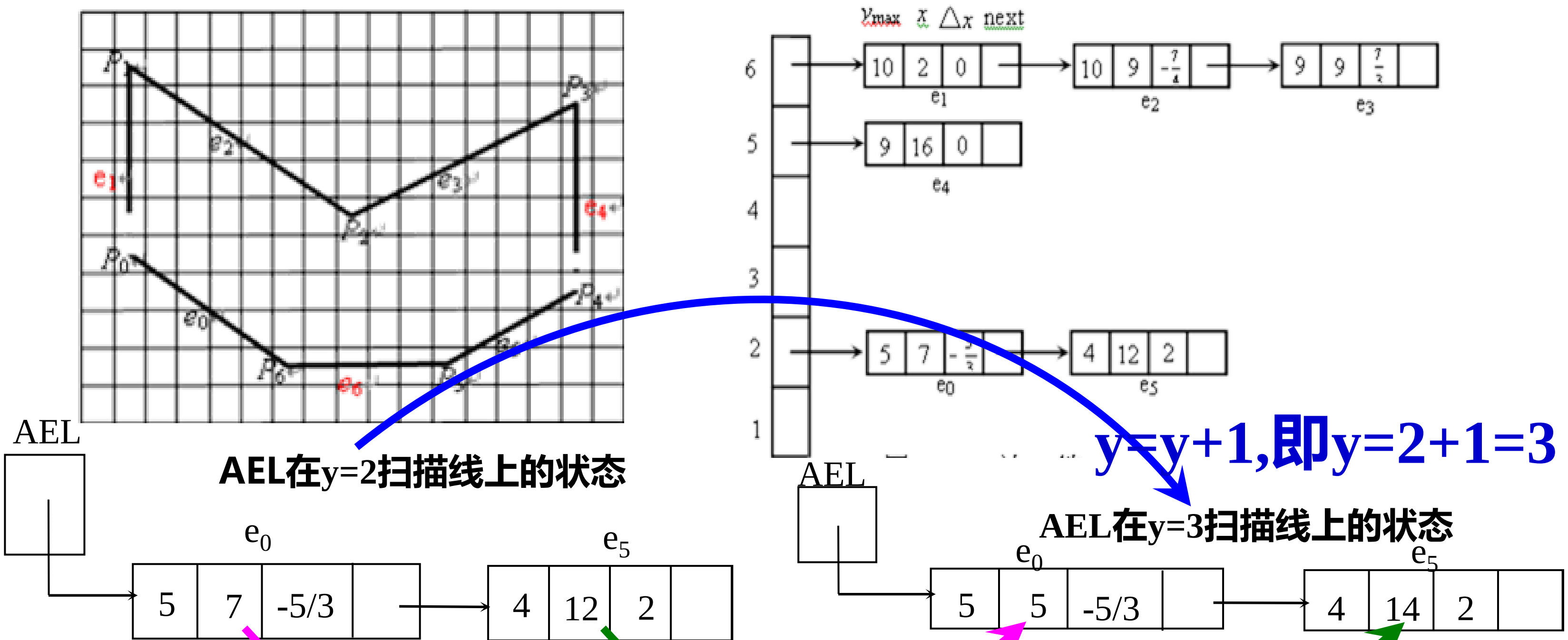


AEL在y=2扫描线上的状态



扫描线算法的具体步骤：

- ③将活化链表AEL中满足$y=y_{max}$的边删去。
  (表明一条边已经结束)
- ④将边ET剩下的每一条边的x域累加Δx，即$x=x+\Delta x$。
  (求得下一条扫描线与边的交点x，利用了边的连续性)
- ⑤将当前的扫描线的纵坐标值y累加，即$y=y+1$
  (求下一条扫描线)



e_0 :m=-5/3,所以x变为7-5/3=5

e_5 :m=2,所以x变为12+2=14

第四章 基本光栅图形算法

- 1、直线生成算法
- 2、圆弧生成算法
- 3、多边形的填充

3.1 多边形的两种表示方法（顶点表示/点阵表示）
3.2 多边形填充的扫描线算法（基本思想、性质、奇异点的处理、数据结构和实现步骤）
3.3 边缘填充算法
3.4 边界标志算法

- 4、区域填充

区域的概念及两种表示方法(内点表示/边界表示)、简单种子填充算法、扫描线种子填充算法

第五章 变换和裁剪

- 1、几何变换

平移变换、放大缩小变换、旋转变换及其实现函数，齐次坐标的概念；

- 2、裁剪

2.1 Sutherland-Cohen算法(基本思想及算法的具体实现)

2.2 Cyrus-Beck算法及梁-Barsky算法(基本思想及算法的具体实现)

5.5 推导以直线ax+by+c=0为对称轴的二维对称变换矩阵

直线变为y=(-a/b)x+(-c/b)，即直线的斜率为-a/b，直线的截距为-c/b，整个变换过程可分以下几个步骤完成：

- a）沿y轴，平移直线使之通过原点，平移量为c/b，变换矩阵：
- b）绕Z轴旋转-θ（θ=arctg(-a/b)），使直线与x轴重合，变换矩阵：
- c）做关于x轴的对称变换，变换矩阵：
- d）绕Z轴回旋θ，变换矩阵：
- e）沿y轴，平移直线，平移量为-c/b，变换矩阵：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{c}{b} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\frac{c}{b} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \begin{bmatrix} \frac{b^2-a^2}{a^2+b^2} & \frac{-2ab}{a^2+b^2} & \frac{-2ac}{a^2+b^2} \\ \frac{-2ab}{a^2+b^2} & \frac{a^2-b^2}{a^2+b^2} & \frac{-2bc}{a^2+b^2} \\ 0 & 0 & 1 \end{bmatrix}$$

第五章 变换和裁剪

- 1、几何变换

平移变换、放大缩小变换、旋转变换及其实现函数，齐次坐标的概念；

- 2、裁剪

2.1 Sutherland-Cohen算法(基本思想及算法的具体实现)

2.2 Cyrus-Beck算法及梁-Barsky算法(基本思想及算法的具体实现)

5.3.1 Sutherland-Cohen算法

Sutherland–Cohen算法分成两部分：

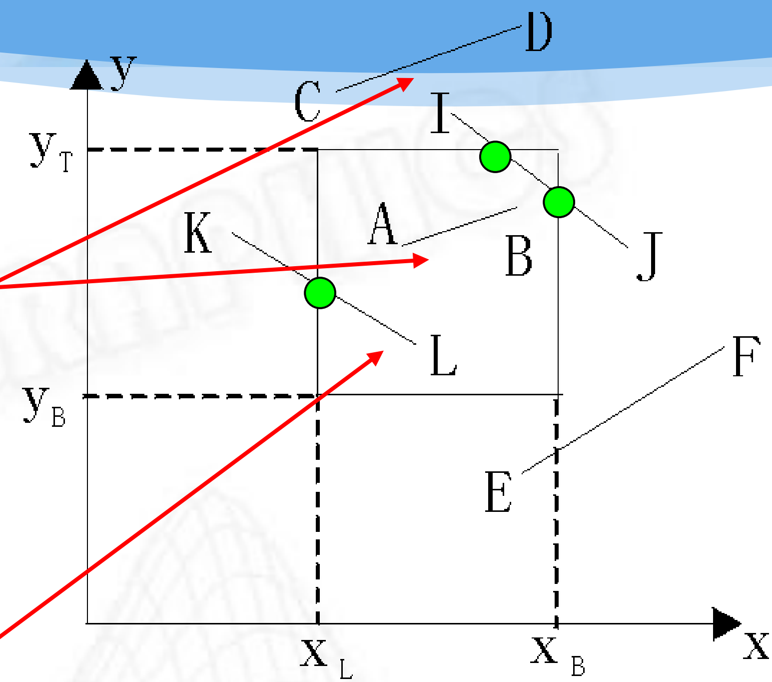
第一步，判定：

1) 完全在窗口内的直线段，称为

完全可见的线段；

2) 完全在窗口外的线段，称为

完全不可见线段。



第二步，处理不能断定为完全可见或完全不可见的线段。

*这时需要计算出直线段和窗口边界的一个交点，这个交点把直线分成两段，其中一条为完全不可见的线段,被抛弃；对余下部分再作第一步的判断。

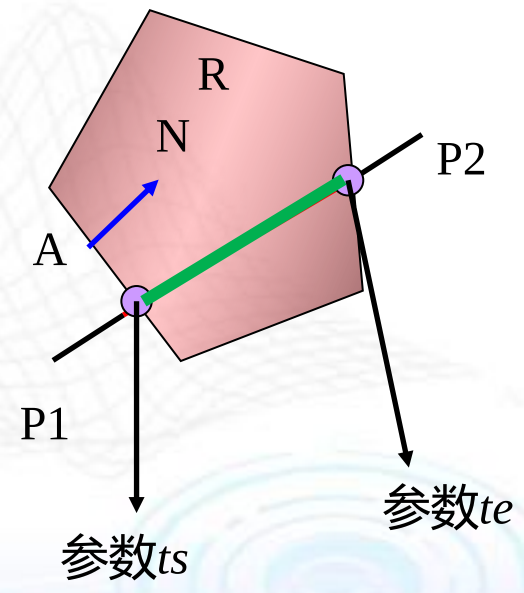
*重复上述过程，直到直线段余下的部分可用第一步的判断得出肯定的结论为止。

5.3.2 Cyrus-Beck算法

3、基本思想

$$P(t) = (P_2 - P_1)t + P_1 \qquad （0 \le t \le 1） \qquad （5.17）$$

对于线段P_1P_2的参数方程表示，如果能判断出线段进入多边形时候的参数t_s和线段退出多边形时的参数t_e，则t_st_e之间的线段为裁剪完毕后的结果。



5.3.2 梁友栋-Barsky算法

当凸多边形是矩形窗口，且矩形的边平行于坐标轴时，Cyrus-Beck算法可简化为梁友栋-Barsky算法。

表5.1 梁友栋-Barsky算法所用的量

对于窗口的每条边，表5.1列出了其内法向量 $\boldsymbol{N}_i$，该边上一点 A_i，从 A_i 指向线段起点 P_1 的向量 $\boldsymbol{P}_1-\boldsymbol{A}_i$，以及线段与该边（或延长线）的交点参数。

由于每个法向量只有一个非零分量，所以：

任意一个向量与法向量求内积，相当于给出该向量的相应分量。

边	内法向量 $\boldsymbol{N}_i$	边上一点 $\boldsymbol{A}_i$	P_1 - $\boldsymbol{A}_i$	$t=-\dfrac{\boldsymbol{N}\cdot(P_1-A)}{\boldsymbol{N}\cdot(P_2-P_1)}$
左边 $x=XL$	(1，0)	(XL，y)	(x_1-XL, y_1-y)	$\dfrac{(x_1-XL)}{-(x_2-x_1)}$
右边 $x=XR$	(-1，0)	(XR，y)	(x_1-XR, y_1-y)	$\dfrac{-(x_1-XR)}{x_2-x_1}$
下边 $y=YB$	(0，1)	(x，YB)	(x_1-x, y_1-YB)	$\dfrac{(y_1-YB)}{-(y_2-y_1)}$
上边 $y=YT$	(0，-1)	(x，YT)	(x_1-x, y_1-YT)	$\dfrac{-(y_1-YT)}{y_2-y_1}$

Liang, Y.D., and Barsky, B., "A New Concept and Method for Line Clipping", ***ACM Transactions on Graphics***, 3(1):1-22, January 1984

5. 3. 2 梁友栋-Barsky算法

设

$\Delta x=x2-x1$，

$\Delta y=y2-y1$， 令
$$\begin{cases} r_L=-\Delta x, & s_L=x_1-x_L, \\ r_R=\Delta x, & s_R=x_R-x_1, \\ r_B=-\Delta y, & s_B=y_1-y_B, \\ r_T=\Delta y, & s_T=y_T-y_1, \end{cases}$$

由表5.1可得交点的参数 t_k

$$t_k=s_k/r_k \quad , \quad k=L,R,B,T$$

实际就是判断Ni·(p2-p1)>0还是<0

上述终点组和起点组的特征分别表现为$r_k>0$和$r_k<0$，其中k对应于相应的裁剪边界(k=L、R、B、T,分别对应于左、右、下、上边界)沿 P_1P_2方向前进。$r_k>0$时，将进入k边界的**外侧**；$r_k<0$时，将进入k边界的**内侧**。若$r_k=0$时$s_k<0$,线段完全不可见，算法结束，否则就继续处理其他边。

实际是在平行情况下，
判断Ni·(p1-Ai)<0则表明整条直线在外侧

第六章 三维空间的观察

- 1、投影

  1.1 两种投影变换即透视和平行投影的概念及公式

  1.2 任意坐标系到观察坐标系的变换；

- 2、视见体规范化、窗口到视口的变换、连续变换的处理

第七章 人机交互绘图技术

- 1、基本的交互任务

（输入、输出、定位、定值、选择、拾取字符串）

- 2、人机交互输入模式

（请求模式、样本模式、事件模式、混合）

- 3、常见辅助交互技术

（几何约束、拖拽、三视图、结构平面等）

第八章 隐藏线和隐藏面的消除

- **1、可见面判断的有效技术**－－边界盒、后向面的概念、非垂直投影转换成垂直投影；
- **2、基于窗口的子分算法、基于多边形的子分算法；**
- **3、z缓冲器算法及其扫描线算法；**
  - **3.1 两种算法的基本思想**
  - **3.2 扫描线算法的数据结构、具体步骤**
- **4.优先级排序算法和光线投射算法；**

第九章 简单光照明模型

- 1、简单光照明模型（光源、材质、简单光照明模型）
- 2、光滑明暗处理技术

Gouraud明暗处理技术——对多边形顶点处光亮度做双线性插值

Phong明暗处理技术——对多边形顶点处法向量做双线性插值

第十章 Bézier曲线曲面

- 1、曲线曲面的基础知识
- 2、Bézier曲线

2.1 Bézier曲线的定义
2.2 Bézier曲线的性质(端点、端点的切线和曲率、仿射不变性、凸包性、交互能力、变差缩减性和保凸性)

考试题型：

- 简答题（**5**个）
- 综合题（**4**个）
- 计算题（**2**个）

分数计算：

平时**+**上机**30%**，期末考试**70%**

预祝考试顺利