



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



工程图学 (2)



内容

- 1、综述
- 2、直线的扫描转换
- 3、一般曲线的扫描转换



1、综述-图形的表示

处理图形我们应考虑以下问题：

1. 如何在计算机中表示图形？
2. 如何准备图形的数据？
3. 如何显示准备好的图形？
4. 如何实现人与图形的交互？

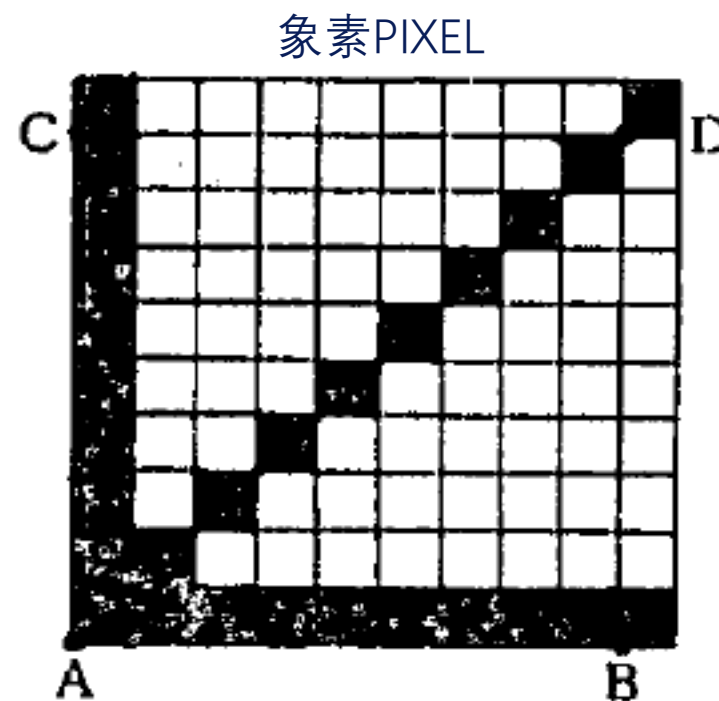
这里，图形是一个广义的概念，凡是可以在图形设备上输出的点、线、文本等的集合都可以称为是图形。

1、综述

- 光栅图形显示器可以看作由像素组成的矩阵，每个像素可以用一种或多种颜色显示，分别称为单色显示器或彩色显示器。
- 在光栅显示器上的图形，是一些具有一种或多种颜色的像素的集合。

光栅扫描显示器不是画线设备，而是画点设备，它不能精确的在两点之间画一条直线，而是由靠近这条直线的像素点表示。

直线算法追求的目标：选择直线的最佳光栅位置。



1、综述

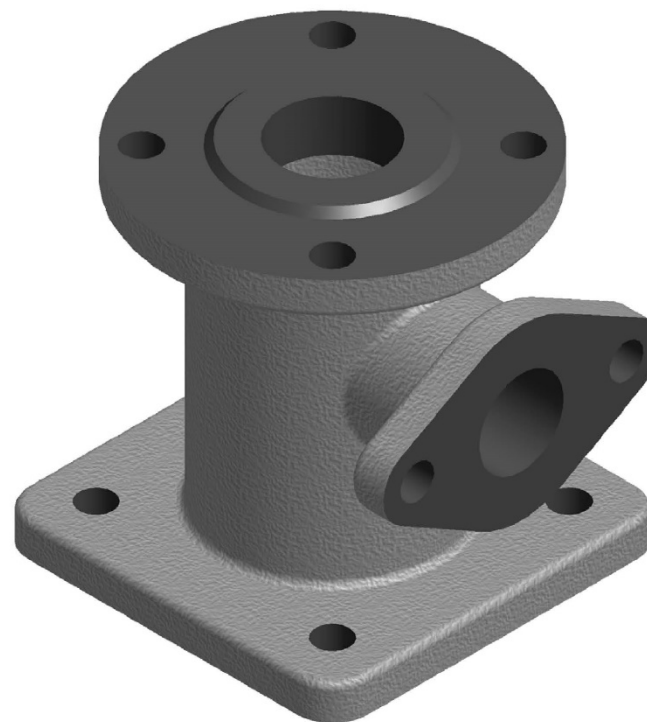
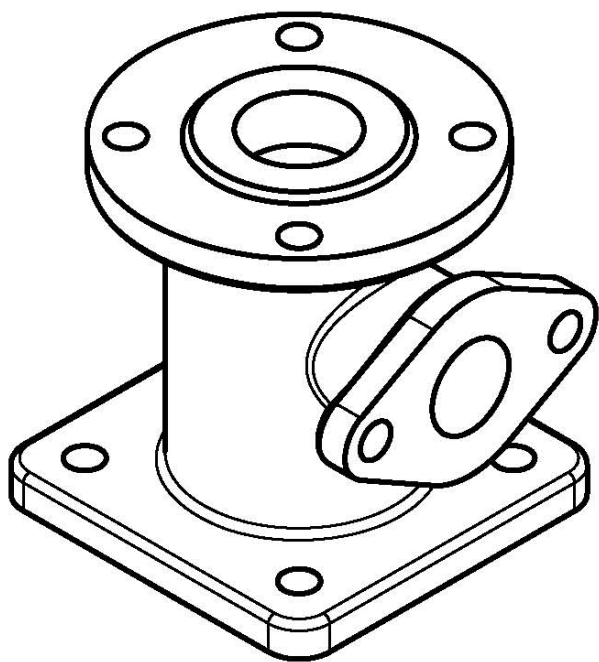
- 扫描转换（光栅化、插补）：用于显示一个图形而确定一个象素集合及其颜色的过程。
- 基本图形的扫描转换问题，包括：直线、圆、椭圆的扫描转换问题。
- 扫描转换步骤：
 - 确定有关像素
 - 用图形的颜色或其它属性，对像素进行某种写操作
- 扫描转换的主要工作是确定最佳逼近于图形的像素集
- 对于一维图形，在不考虑线宽时，用一个像素宽的直/曲“线”（即像素序列）来现实图形。

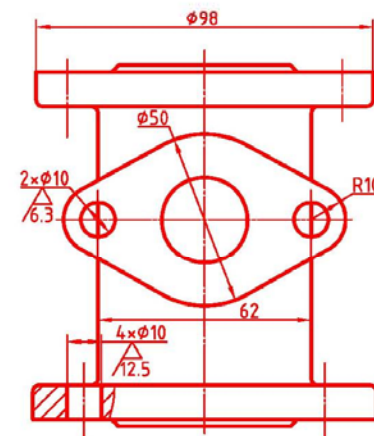
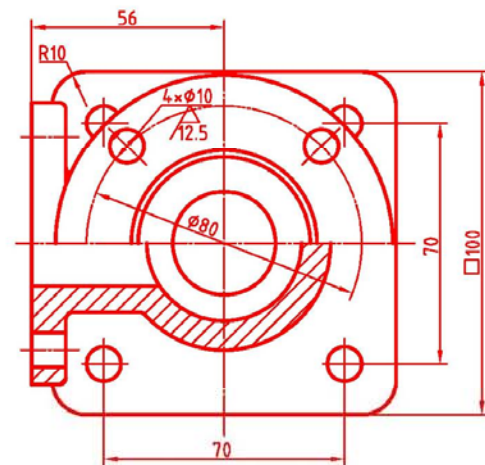
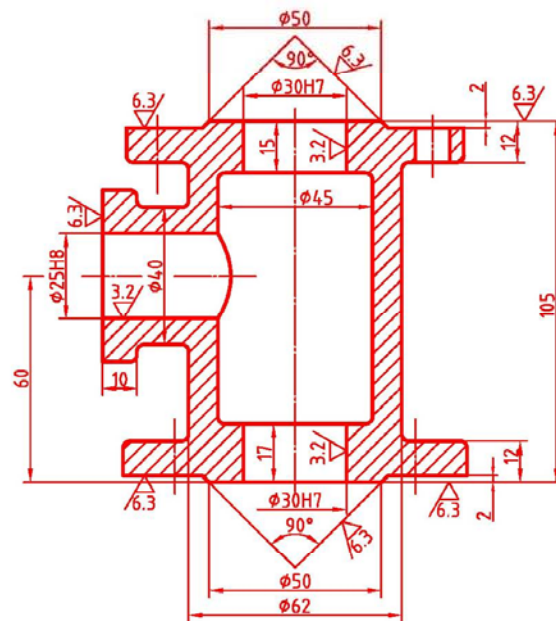
1、综述

两类图形：线条图形和点阵图形。

无论是哪一类，**点**是图形的基本几何元素。例如四边形，可以用四个角点表示，生成时用直线顺序连接四个角点。再例如曲线，也是由曲线上的点或控制多边形定义或曲线方程定义，通常用短直线逼近表示。

线条图形和点阵图形





其余

未注铸造圆角 R3

						HT200			北京邮电大学 自动化学院	
标记	处数	分号	更改文件号	(签名)	(年月日)				泵体	
设计	(签名)	(年月日)	审核	(签名)	(年月日)	(修改标记)	质量	比例		
						1:1			LT2.03	
						共	张	第	张	

表示图形的数据准备

图形最终是由点和如何表示这些点的绘图算法表示的。 这些信息在显示以前一般存储在数据库文件中，复杂的算法不仅数据库文件复杂，其存取算法也复杂。这些复杂数据库中的数据以各种方式组织在一起。例如，**环结构、二叉树结构、链表结构、四叉树结构等等。这些结构称为数据结构。** 数据库文件包含指针、子结构和其他非图的数据。

表示图形的数据准备

数据库及其存取算法是当前的一个研究领域。许多计算机图形的应用只是一些简单图形，这些图形的数据结构比较简单，设计很容易。如线性表、链表，这些简单数据结构也可以应付一些比较复杂的图形。

数据结构是一门计算机专业的课程，我们仅在用到时介绍。这一章讲图形的基本算法。

图形的显示

显示的图形分为二维和三维两种。二维图形比较简单，通常只需要做平移和比例等变换。最简单的只需作平移变换。例如，画圆程序（VB）：

```
Private Sub Form_click() '画圆
    Scale (0, 0)-(1600, 1200)
    pi = 3.14159
    x0 = 800: y0 = 600: r = 100: n = 36
    For i = 0 To n '循环
        ang = i * 2 / n * pi '圆心角
        x = x0 + r * Cos(ang)
        y = y0 - r * Sin(ang)
        If i = 0 Then
            PSet (x, y), RGB(255, 255, 0)
        Else
            Line -(x, y), RGB(255, 255, 0)
        End If
    Next i
End Sub
```

TC2.0的图形函数

Turbo C 提供了非常丰富的图形函数，所有图形函数的原型均在**graphics. h** 中

另外，使用图形函数时要确保有显示器图形驱动程序***BGI**，同时将集成开发环境options/Linker中的Graphics lib选为on，只有这样才能保证正确使用图形函数。

不同的显示器适配器有不同的图形分辨率。即是同一显示器适配器，在不同模式下也有不同分辨率。因此，在屏幕作图之前，必须根据显示器适配器种类将显示器设置成为某种图形模式，在未设置图形模式之前，微机系统默认屏幕为文本模式(80列，25行字符模式)，此时所有图形函数均不能工作。

设置屏幕为图形模式

TC2.0的图形模式和坐标系

利用Turbo C显示图形之前，必须先进入图形模式。进入图形模式的函数原形为：

```
void far initgraph (int far *drive, int far *mode, char far *path) ;
```

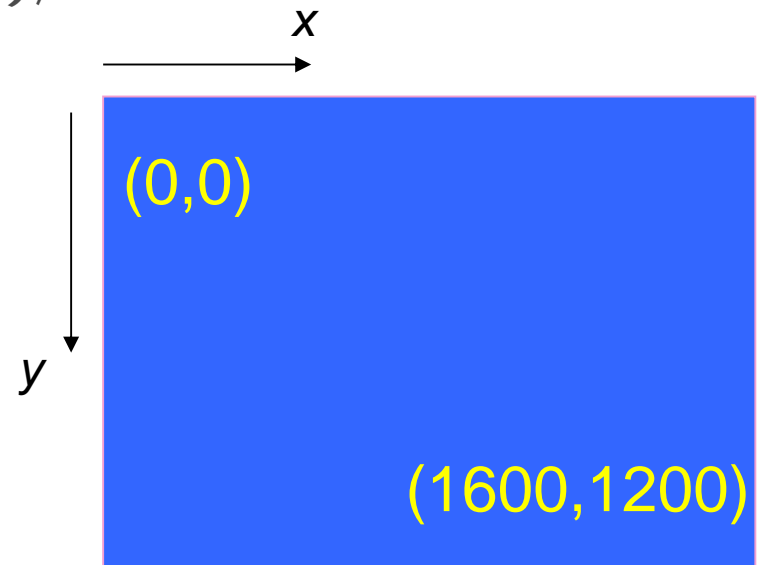
其中gdriver和gmode分别表示图形驱动器和模式，path是指图形驱动程序所在的目录路径图形驱动程序由Turbo C出版商提供，文件扩展名为.BGI。

有时编程者并不知道所用的图形显示器适配器种类，或者需要将编写的程序用于不同图形驱动器，Turbo C 提供了一个自动检测显示器硬件的函数，其调用格式为：

```
void far detectgraph(int *gdriver, *gmode);
```

其中gdriver和gmode的意义与上面相同

- ❖ 屏幕上每个象素根据它们的x，y坐标确定
- ❖ 进入图形模式之后，缺省的坐标系原点（0，0）在屏幕左上角。



内容

1、综述

2、直线的扫描转换

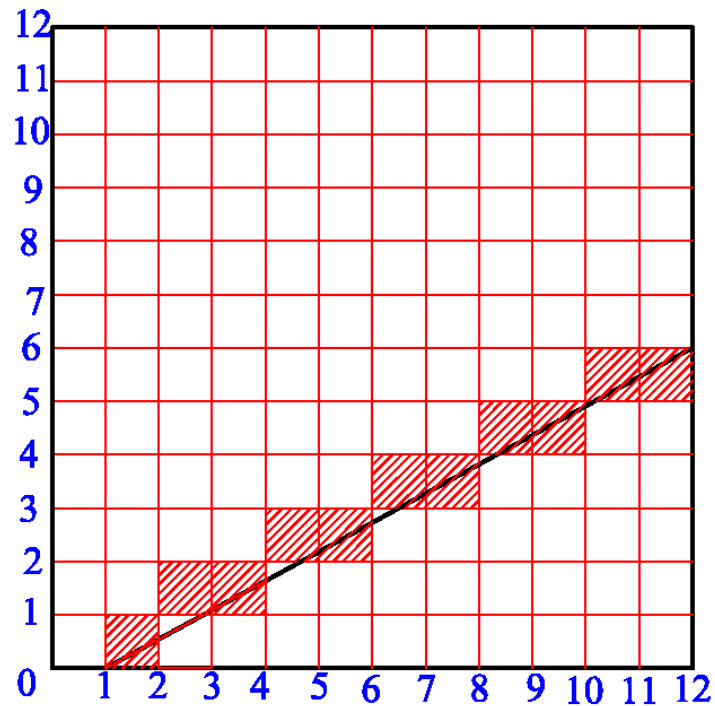
3、一般曲线的扫描转换



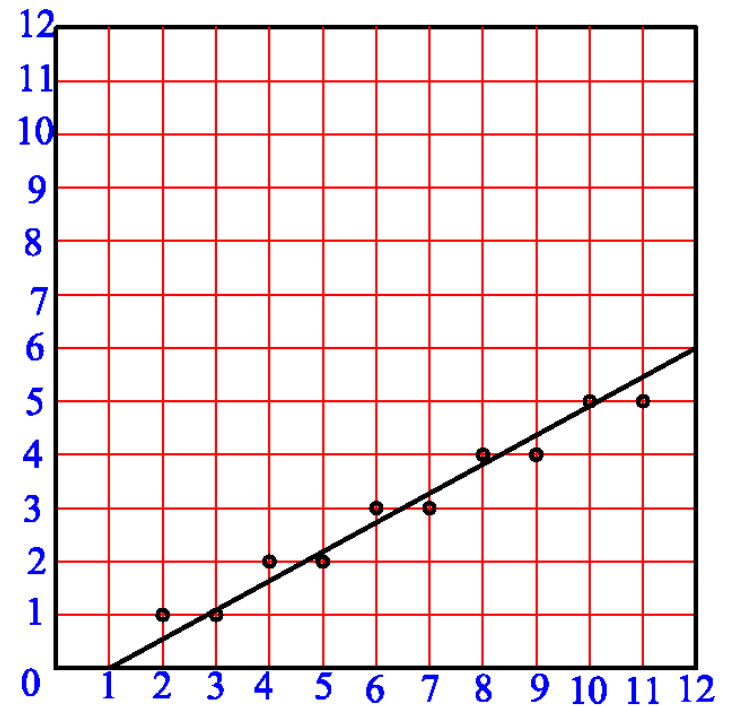
2、直线的扫描转换

- 理想的直线是没有宽度的，由无数点构成。
- 对直线进行光栅化时，只能在显示器的有限个像素中，确定一组最佳逼近的像素。
- 直线是最基本的图形，一个图形中可以包含很多直线，扫描转换算法的效率非常重要，关键是把乘法取消。
- 常用算法：
 - DDA数值微分法
 - 中点画线法
 - Bresenham布莱森汉姆算法

直线的近似表示

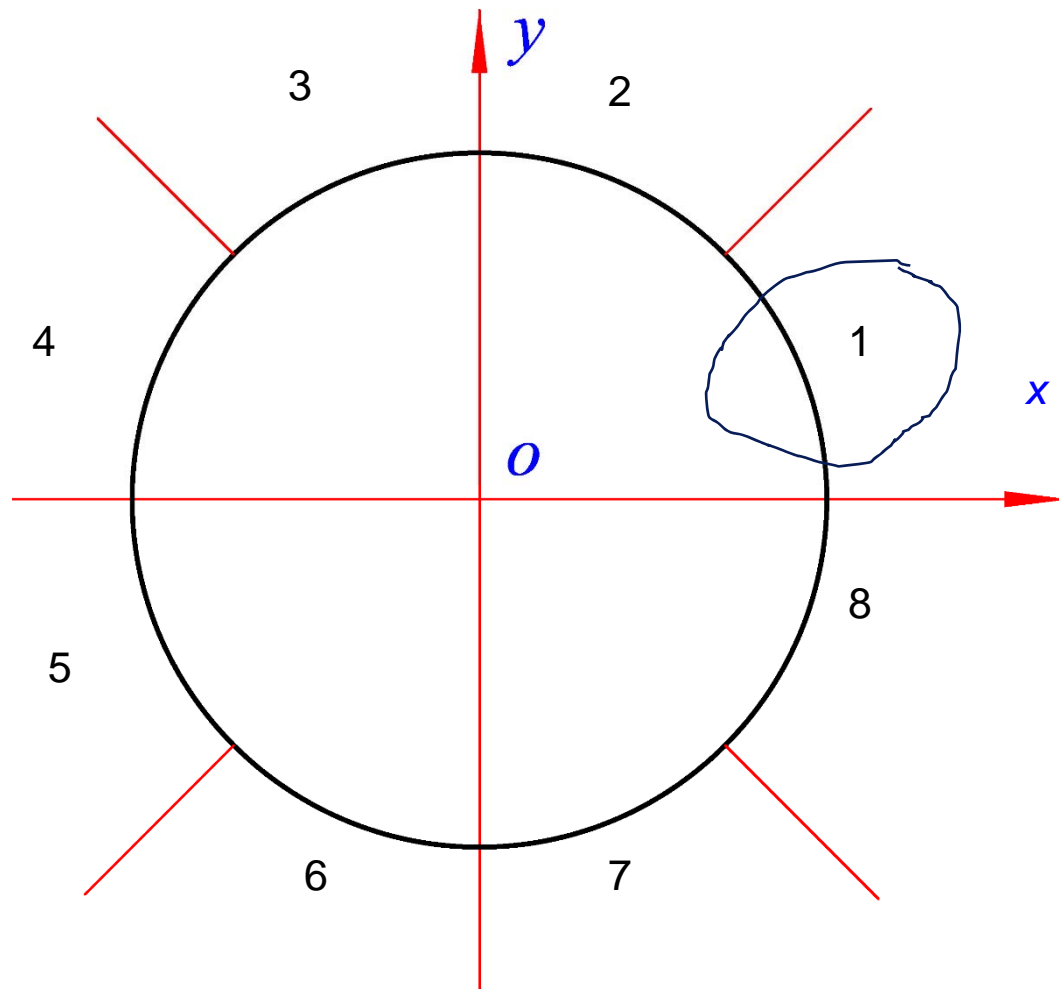


屏幕坐标系下的直线近似表示



数学坐标系下的直线近似表示

八分圆域



如何把数学上的一个点扫描转换一个屏幕像素点？

如： $p(1.7, 0.8)$ $\xrightarrow{\text{取整}}$ $p(1, 0)$ 直接取整

$p(1.7, 0.8)$ $\xrightarrow{+0.5}$ $p(2.2, 1.3)$ +0.5再取整
相当于四舍五入

$p(2.2, 1.3)$ $\xrightarrow{\text{取整}}$ $p(2, 1)$

2、直线的扫描转换

(1) 数值微分法 (DDA : Digital Differential Analyzer)

➤原理:

引入图形学一个很重要的思想

——增量思想

(1) 计算直线的斜率 $k = \Delta y / \Delta x$

(2) 从直线的起点出发, 计算: x 每增加一个步长 (设为 Δx), y 所增加的值:

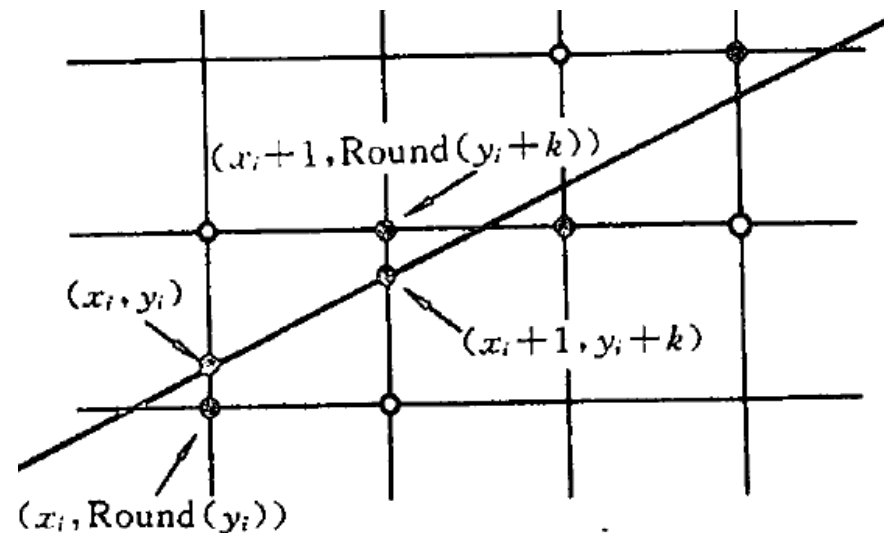
$$y_{i+1} = kx_{i+1} + b = k(x_i + \Delta x) + b$$

$$= kx_i + b + k\Delta x = y_i + k\Delta x$$

当 $\Delta x = 1$ 时, $y_{i+1} = y_i + k$

即 x 增加 1, y 增加斜率 k ($k \leq 1$)

简化方法



2、直线的扫描转换

(1) 数值微分法DDA

➤程序实现：（设由 x_0 、 y_0 到 x_1 、 y_1 画直线）

DDAline($x_0, y_0, x_1, y_1, color$)

int $x_0, y_0, x_1, y_1, color$;

{

int x ;

float d, dy, k, y ;

$dx = x_1 - x_0$;

$dy = y_1 - y_0$;

$k = dy/dx$;

$y = y_0$;

for($x = x_0; x \leq x_1; x++$)

{

putpixel($x, \text{int}(y + 0.5), color$);

$y = y + k$;

}

}

注意 $\text{int}(x)$ 是向下取整

通过 $x + 0.5$ 变成四舍五入

$\begin{cases} x = x + 1 \\ y = y + k \end{cases}$

2、直线的扫描转换

(1) 数值微分法DDA

➤总结:

■数值微分法的本质: 通过数值微分法, 对 x 和 y 各加一个小增量, 计算下一步的 x 、 y 值。

■增量算法: 在一个迭代算法中, 如果每一步的 x 、 y 值是用前一步的值加上一个增量来获得的, 此算法称为增量算法。

■DDA的优缺点: 算法简单, 但每一步都要对进行舍入取整(整数运算)。

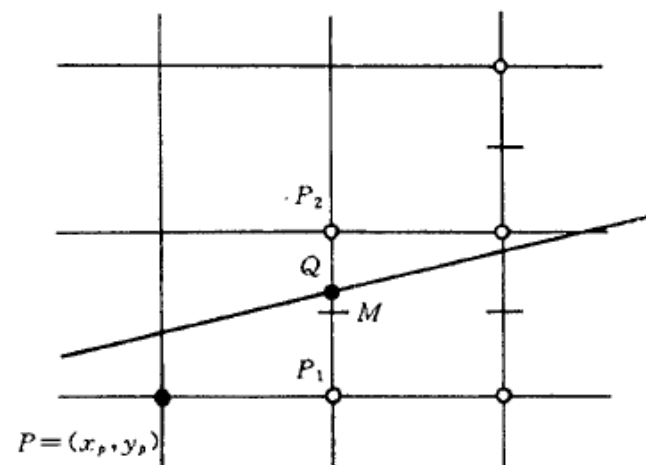
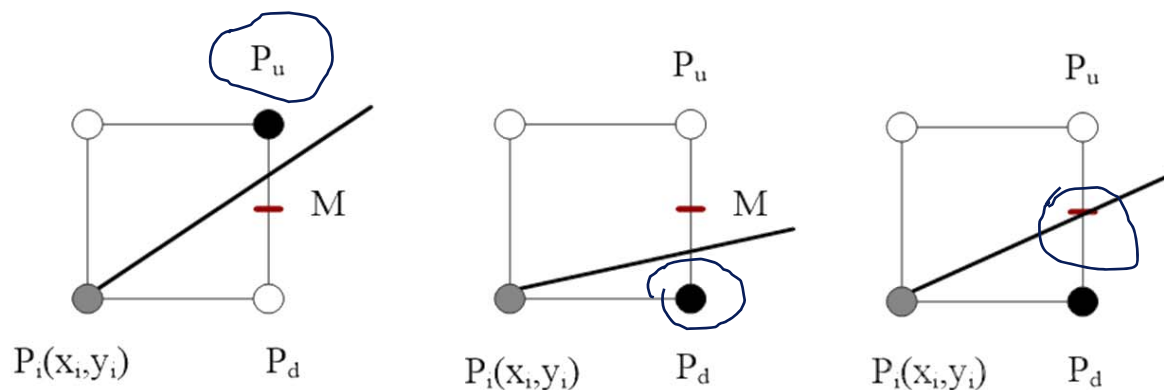
2、直线的扫描转换

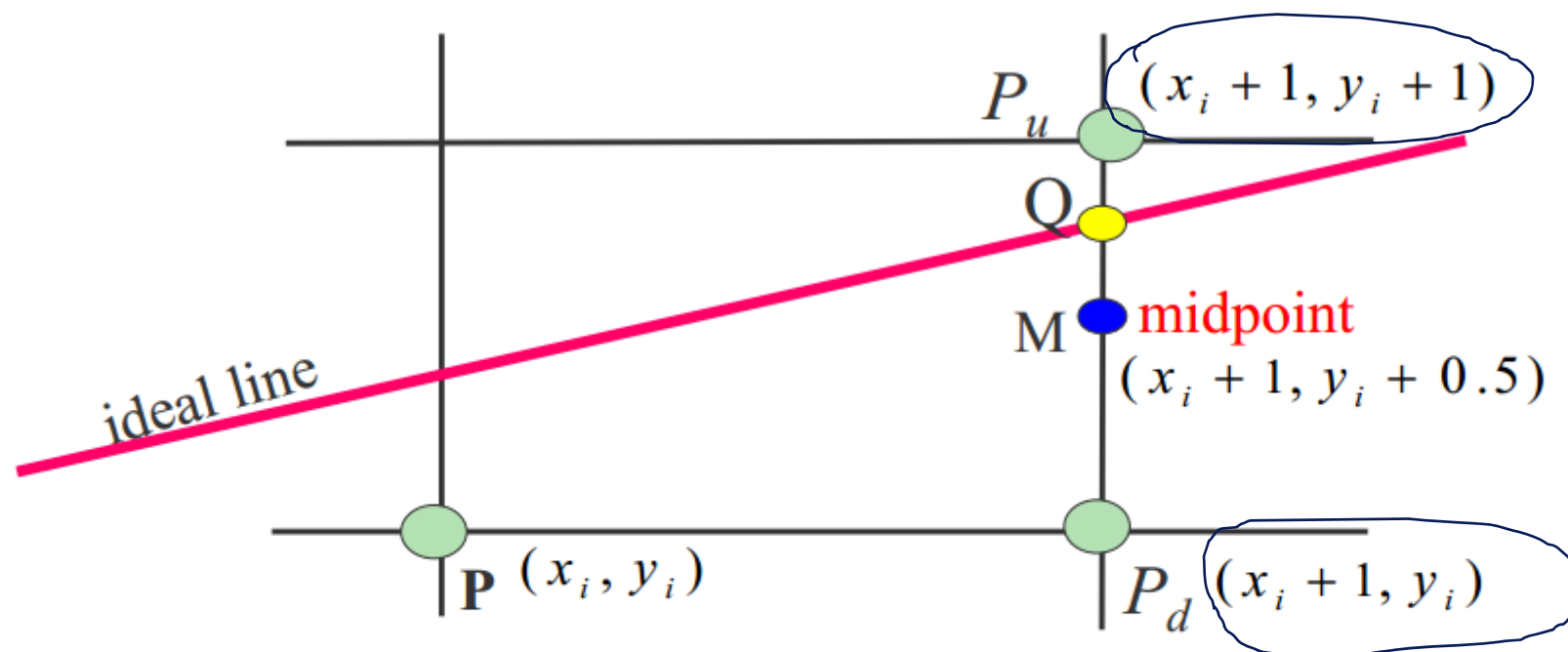
(2) 中点画线法

➤原理：通过判断靠近理想位置直线的上（**P2点**）、下（**P1点**）两个像素点间的**中点的位置**（在线的上方还是下方），来选取下面或上面的像素点。

➤假定斜率在0, 1之间。

第一象限1号域





当M在Q的下方，则 P_u 离直线近，应为下一个像素点

当M在Q的上方，应取 P_d 为下一点。

http://blog.csdn.net/weixin_38195506

2、直线的扫描转换

(2) 中点画线法

➤步骤:

1. 确定判别式

由直线方程 $ax+by+c=0$

得判别式 $F(x,y)=ax+by+c$

2. 确定 $F(x,y)$ 的符号区间

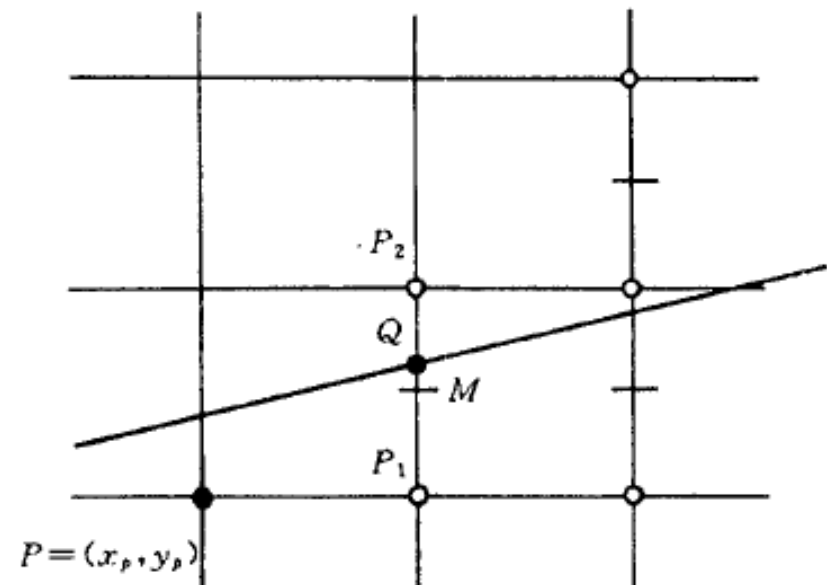
直线上的点 $F(x,y)=0$

直线上方的点 $F(x,y)>0$

直线下方的点 $F(x,y)<0$

$$(a=y_0-y_1; b=x_1-x_0; c=x_0y_1-x_1y_0)$$

$$M(x_p+1, y+0.5)$$



2、直线的扫描转换

(2) 中点画线法

3. 规定选择点：把上下两点间的中点坐标 $M(x,y)$ 代入判别式，

若 $F(x,y)>0$ ，取下方的 $P1(x_p+1,y_p)$ 点（中点 M 在实际点上方）

若 $F(x,y)<0$ ，取上方的 $P2(x_p+1,y_p+1)$ 点（中点 M 在实际点下方）

若 $F(x,y)=0$ ，两点中任意取一点（通常取 ≥ 0 、 < 0 ）

4. 计算 $F(x,y)$ 的值，确定下一步应选取的点。

2、直线的扫描转换

(2) 中点画线法——为了加速技术，采用增量方法

设从点 (x_0, y_0) 到 (x_1, y_1) 画直线，画线过程如下：

(1) 在 (x_0, y_0) 处，判别式的值为：

$$F_0 = F(x_0, y_0) = ax_0 + by_0 + c = 0$$

(2) 计算下一个中点处 $(x_0+1, y_0+0.5)$ 判别式的值：

$$\begin{aligned} F_1 &= F(x_0+1, y_0+0.5) = a(x_0+1) + b(y_0+0.5) + c \\ &= (ax_0 + by_0 + c) + a + 0.5b = F(x_0, y_0) + a + 0.5b \\ &= 0 + a + 0.5b = a + 0.5b \end{aligned}$$

取增量 $d_0 = F_1 - F_0 = a + 0.5b$

2、直线的扫描转换

(2) 中点画线法

若 $d_0 \geq 0$, 则取点 (x_0+1, y_0)

若 $d_0 < 0$, 则取点 (x_0+1, y_0+1)

① 若取点 (x_0+1, y_0) , 则下一个中点为 $(x_0+1+1, y_0+0.5)$, 判别式的值为:

$$F_2 = F(x_0+2, y_0+0.5) = a(x_0+2) + b(y_0+0.5) + c$$

$$= (ax_0 + by_0 + c) + 2a + 0.5b = F(x_0, y_0) + 2a + 0.5b$$

$$= 0 + 2a + 0.5b = 2a + 0.5b = d + a$$

$$d_0 \geq 0, \Delta_1 = F_2 - F_1 = a$$

2、直线的扫描转换

(2) 中点画线法

② 若取点 (x_0+1, y_0+1) ，则下一个中点为 $(x_0+1+1, y_0+0.5+1)$ ，判别式的值为：

$$F_2 = F(x_0+2, y_0+1.5) = a(x_0+2) + b(y_0+1.5) + c$$

$$= (ax_0 + by_0 + c) + 2a + 1.5b = F(x_0, y_0) + 2a + 1.5b$$

$$= 0 + 2a + 1.5b = 2a + 1.5b = d + a + b$$

$$d_0 < 0, \Delta_2 = F_2 - F_1 = (2a + 1.5b) - (a + 0.5b) = a + b$$

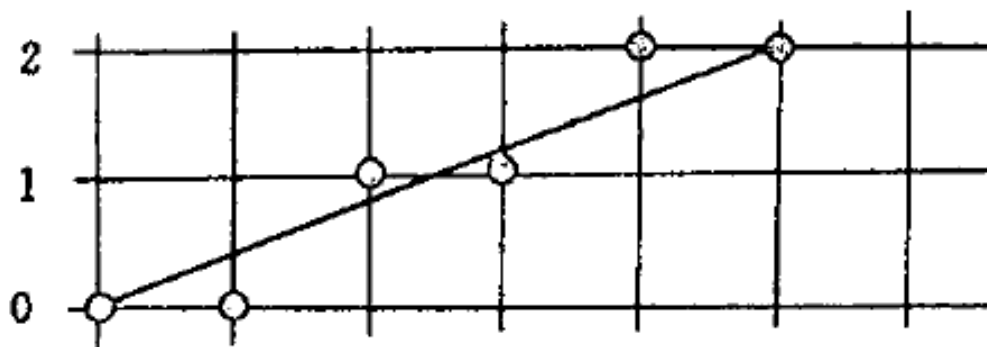
因为只根据d的正负判断取点，所以取2d代替d摆脱小数

2、直线的扫描转换

(2) 中点画线法

示例：

- 从点 (0, 0) 到 (5, 2) 的直线
- 斜率为 $2/5$ ，可以应用中点画线算法 (<1)
- 第一个像素为 (0, 0)。
- 判别式 d 的初始值为： $d_0 = 2*a + b = 2*(-2) + 5 = 1 > 0$
- d 往正右方向的 $2d$ 增量为： $\text{delta}1 = 2*a = -4$
- d 往右上方向的 $2d$ 增量为： $\text{delta}2 = 2*(a+b) = 6$



2、直线的扫描转换

(2) 中点画线法

示例：迭代计算过程

$$a=y_0-y_1; b=x_1-x_0; c=x_0y_1-x_1y_0$$

$$a=-2$$

$$b=5$$

$$c=0$$

$$2d_0=2(a+0.5b)=1$$

$$d_1=2a=-4$$

$$d_2=2(a+b)=6$$

x	y	d	
0	0	1	(初值)
1	0	-3	
2	1	3	
3	1	-1	
4	2	5	
5	2	1	

$$d_{i+1}=d_i+\begin{cases} \Delta 1 (d \geq 0) \\ \Delta 2 (d < 0) \end{cases}$$

2、直线的扫描转换

(2) 中点画线法

程序:

midpointline(x0,y0,x1,y1,color)

int x0,y0,x1,y1,color;

{

int a,b,d, delta1,delta2,x,y;

a=y0-y1;

b=x1-x0;

d=2*a+b;

delta1=2*a;

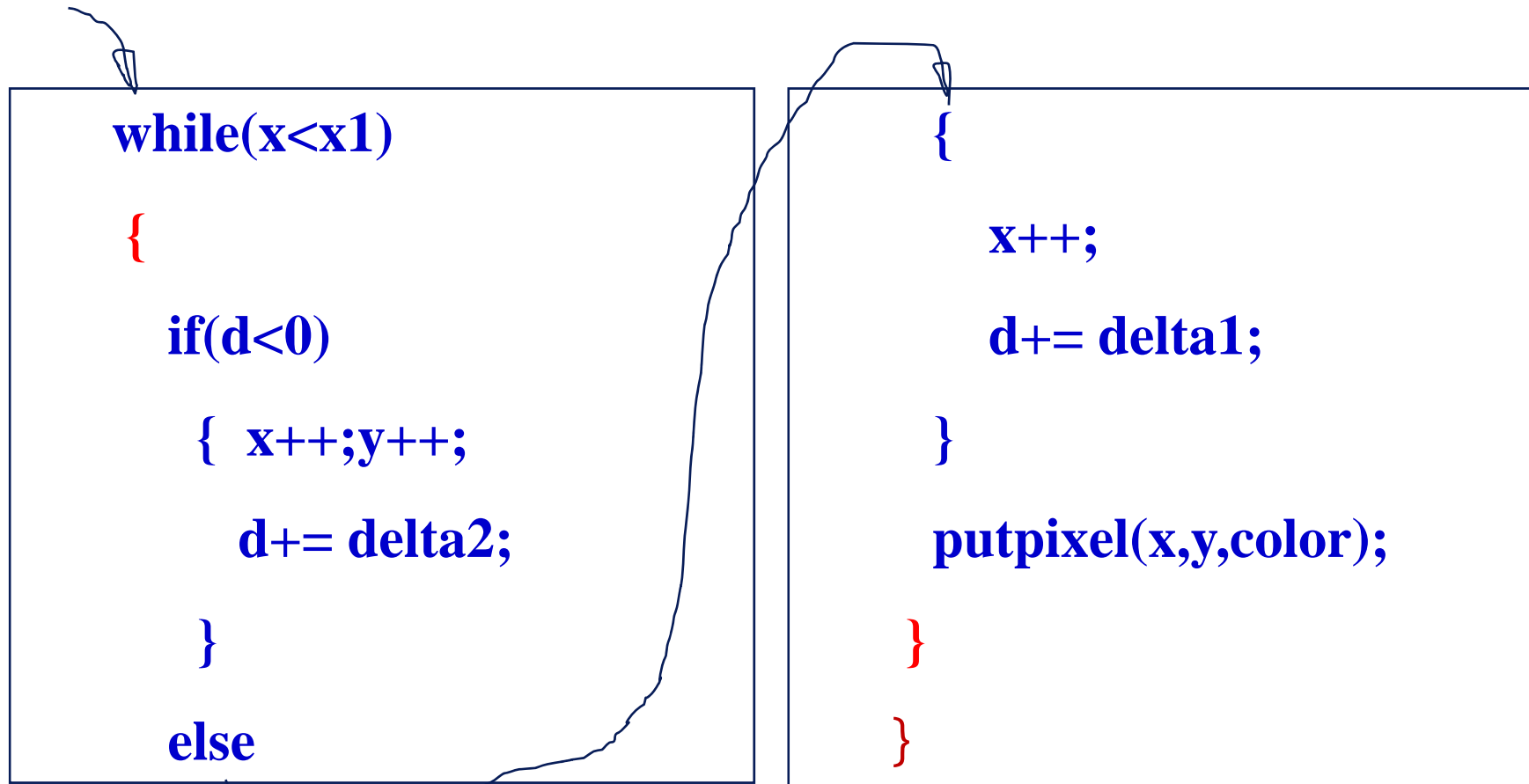
delta2=2* (a+b) ;

x=x0;

y=y0;

putpixel(x,y,color);

2、直线的扫描转换



2、直线的扫描转换

(3) Bresenham画线算法

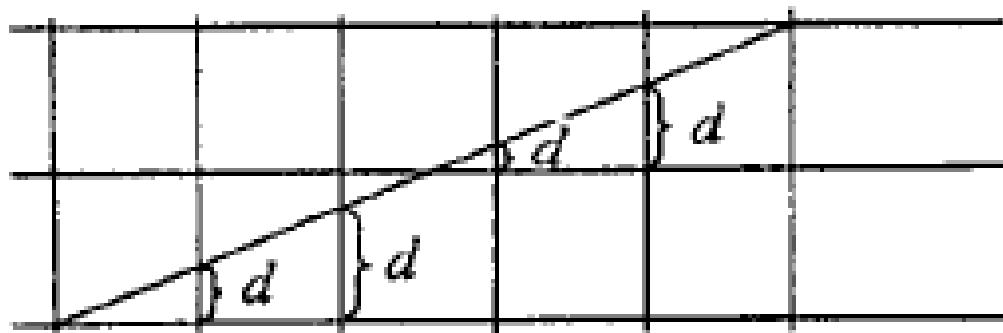
- 计算机图形学领域中使用最广泛的直线扫描转换算法。
- 最初用于数字绘图仪
- 也使用于光栅图形显示器，后来被广泛应用于直线的扫描转换和其他一些应用之中。
- 为了讨论方便，假定直线的斜率在0，1 之间。

2、直线的扫描转换

(3) Bresenham画线算法

➤与中点画线法类似，Bresenham算法通过确定每列像素中与理想直线最近的像素来进行直线的扫描转换。

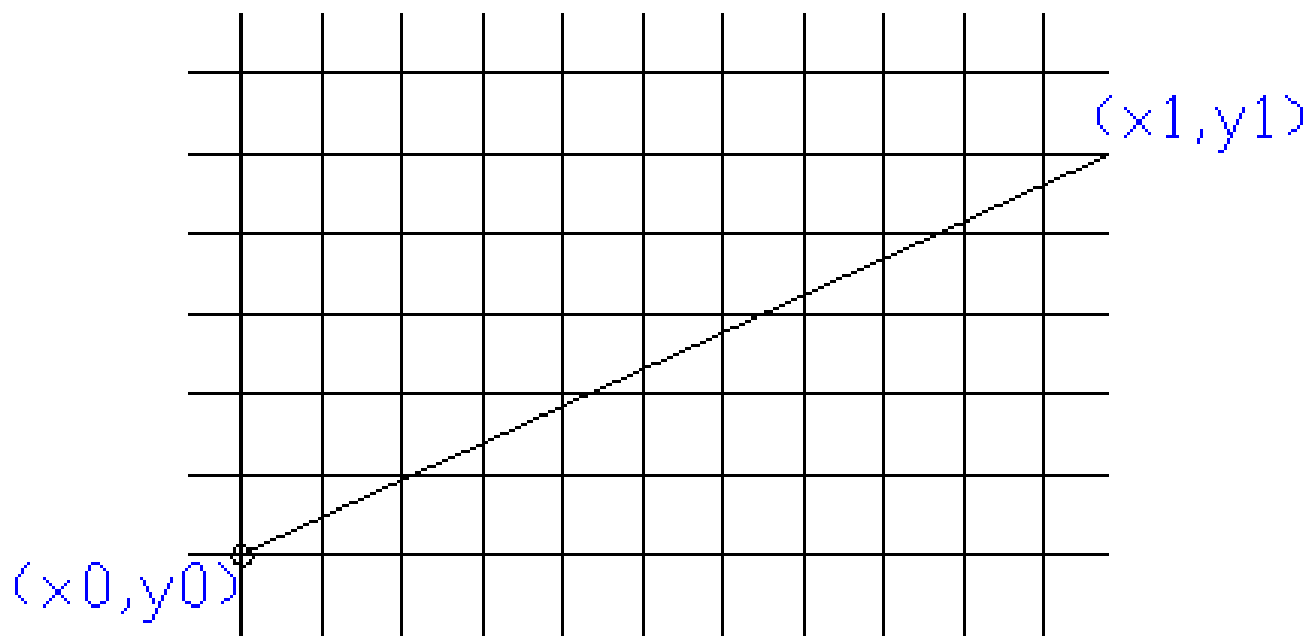
➤原理：过各行、各列像素中心作虚拟的水平、垂直线，构成网格。按直线从起点到终点的顺序，计算直线与各垂直网格线的交点，采用增量计算的方法，通过检查一个误差项的符号，来确定该列像素中与交点最近的像素。



2、直线的扫描转换

(3) Bresenham画线算法

➤转换步骤：设从点 (x_0, y_0) 到点 (x_1, y_1) 画直线。下一个像素的X坐标为 $x+1$ ，而Y坐标要么不变，要么增1（个像素），取决于误差项 d 的大小。



2、直线的扫描转换

(3) Bresenham画线算法

➤转换步骤:

1. 确定判别式（增量式）：

$$d_{i+1} = d_i + k \quad (k \text{ 为斜率, } k = \Delta y / \Delta x)$$

2. 确定选择像素点的方法

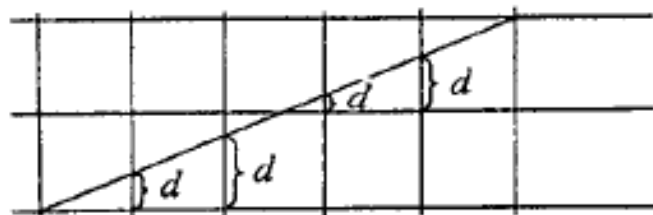
当 $d \geq 0.5$ 个像素时，取点 (x_0+1, y_0+1)

当 $d < 0.5$ 个像素时，取点 (x_0+1, y_0)

当 $d \geq 1$ 个像素时， $d = d - 1$ 【取就近像素点】

x 始终+1

y 视情况+1或+0



2、直线的扫描转换

(3) Bresenham画线算法

➤转换步骤:

3. 简化判别式

为便于判断, 用 $e=d-0.5$ 作为判别式 (这样增量起始值可以直接和0比较)。

当 $d>0.5$ 个像素时, $e>0$

当 $d<0.5$ 个像素时, $e<0$

当 $d=0.5$ 个像素时, $e=0$

在点 (x_0, y_0) 处, $d=0$, e 的初值为 $e=-0.5$

4. 计算 e , 根据 e 的符号选择像素点。不断循环, 直到终点。

2、直线的扫描转换

(3) Bresenham画线算法

➤程序:

```
bresenham_line(x0, y0, x1, y1, color)
```

```
int x0, y0, x1, y1, color;
```

```
{
```

```
    int x, y, dx, dy, i ;
```

```
    float k, e;
```

```
    dx=x1-x0;
```

```
    dy=y1-y0;
```

2、直线的扫描转换

(3) Bresenham画线算法

➤程序:

```
k=dy/dx;
```

```
e=-0.5;
```

```
x=x0;y=y0;
```

```
for(i=0;i<=dx; i++)
```

```
{
```

```
    putpixel(x, y, color);
```

```
    x=x+1;
```

```
e=e+k; /*可改为 e=e+2*dy*/
```

```
if(e>=0)
```

```
{
```

```
    y=y+1;
```

```
    e=e-1; /*可改为e=e-2*dx*/
```

```
}
```

```
}
```

```
}
```

2、直线的扫描转换

(3) Bresenham画线算法

➤为了避免小数和除法，对判别式进一步简化：

1. $e=e+k$; 改为 $e=e+2*dy$

推导过程：

两边乘以 $2*dx$ ， 得 $2*dx* e=2*dx* (e+k)$ ；

令 $e'= 2*dx* e$ 则 $e=e+k$ 变成

$$e'=2*dx*e=2*dx*(e+k)=2*dx*(e+dy/dx)$$

$$=2*dx*e+2*dy=e'+2* dy$$

即 $e=e+2*dy$

2、直线的扫描转换

(3) Bresenham画线算法

2. $e=e-1$; 改为 $e=e-2*dy$

推导过程:

$e=e-1$ 的两边乘以 $2*dx$, 得

$$2*dx*e=2*dx*e-1)$$

$$=2*dx*e-2*dx$$

所以 $e'=e'-2*dx$

即 $e=e-2*dx$

$$e=e+2*dy$$

当 $e>0$ 时

$$e=e-2*dx$$

内容

- 1、综述
- 2、直线的扫描转换
- 3、一般曲线的扫描转换**



3、一般曲线的扫描转换

3.1 逐点比较法基本概念

一般曲线的扫描转换可采用逐点比较法：

在绘图过程中，根据笔位与理想线的相对位置，决定笔的走向，然后移动一步，反复进行，直到终点为止。

逐点比较法就是在输出直线或圆弧的过程中，每走完一步就与理论的直线或圆弧进行比较，确定当前点是在线或弧上，还是在线或弧的一侧，然后再决定下一步的走向，这样一步一步地逼近所画直线或圆弧。

3、一般曲线的扫描转换

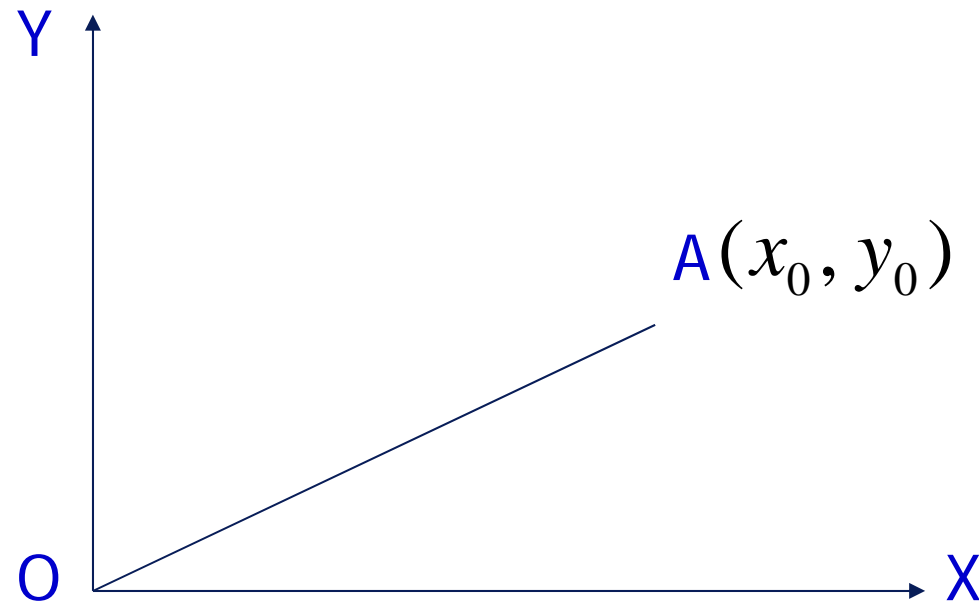
扫描步骤

- 1) 建立偏差函数
- 2) 规定走向（进向选择）
- 3) 偏差判别
- 4) 终点判别：以步数大者为判别方向，否则以两者之和为判别要素

3、一般曲线的扫描转换

3.2 逐点比较法进行直线扫描(适合单调函数)

问题：由原点扫描至点A (x_0, y_0)



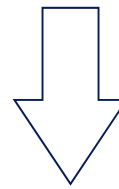
3、一般曲线的扫描转换

直线扫描—建立偏差函数

1、根据直线方程建立偏差函数

直线方程

$$\frac{y}{x} = \frac{y_0}{x_0}$$



偏差函数

$$F(x, y) = x_0 y - y_0 x$$

3、一般曲线的扫描转换

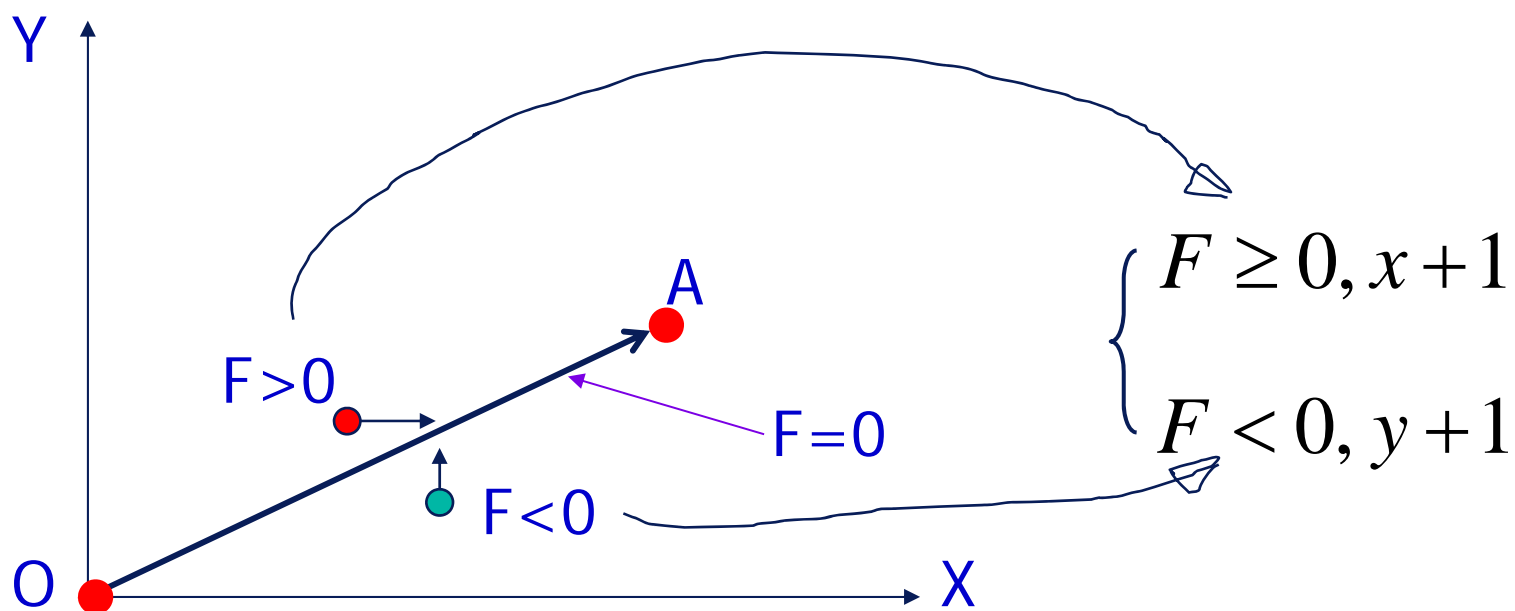
问：同一问题中，偏差函数有几个？

答：有无穷多个。但总的分成两类
($\geq 0 / < 0$)。

3、一般曲线的扫描转换

直线扫描—规定走向

2、根据偏差函数决定各区域偏差符号，然后决定走向



3、一般曲线的扫描转换

直线扫描—偏差判别

3、利用偏差函数计算偏差值，亦可进行简化偏差计算

$$F(x+1, y) = x_0 y - y_0(x+1) = F(x, y) - y_0$$

$$F(x, y+1) = x_0(y+1) - y_0 x = F(x, y) + x_0$$

注：1指的是一个步长

$$F(x, y) = x_0 y - y_0 x$$

3、一般曲线的扫描转换

直线扫描—终点判别

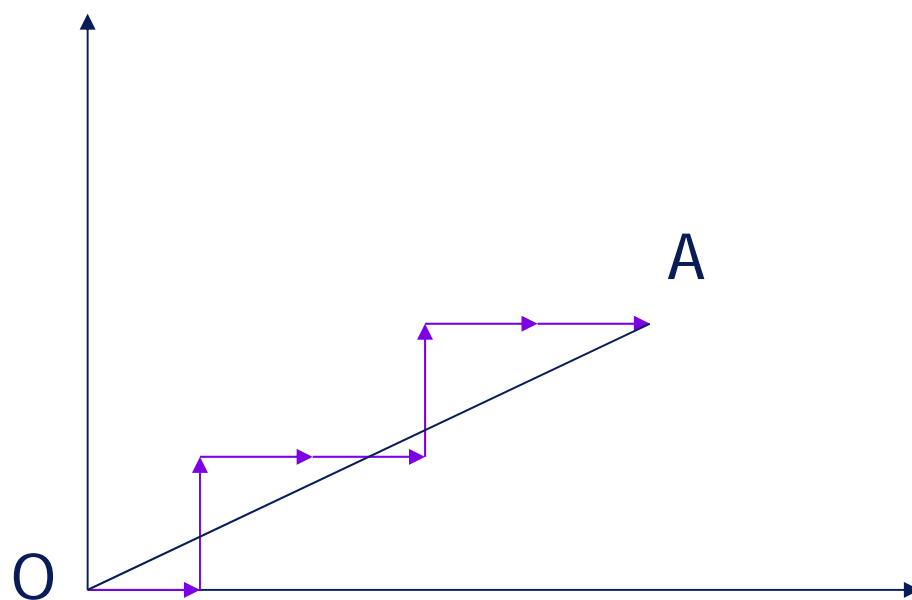
4、利用X和Y方向的步长数进行终点判别以步数大者为判别方向，否则以两者之和为判别要素

$$\mathbf{x}\text{向步数} \quad |(x_0 - 0) / h|$$

$$\mathbf{y}\text{向步数} \quad |(y_0 - 0) / h| \quad h\text{为步长}$$

3、一般曲线的扫描转换

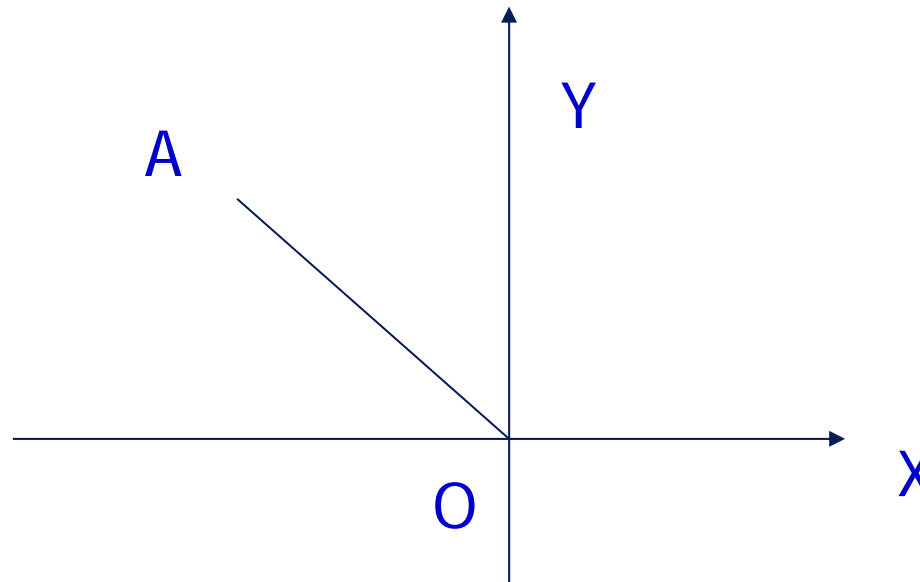
直线扫描-过程示意



3、一般曲线的扫描转换

直线扫描示例

例：由原点到点A $(-0.4, 0.3)$ 进行直线扫描。
(步长为0.1)



3、一般曲线的扫描转换

直线扫描示例—要点

由于步长为0.1，故须进行变换。

题目变成：



由原点到点A (-4, 3) 进行直线扫描。
(步长为1)

3、一般曲线的扫描转换

直线扫描示例—解

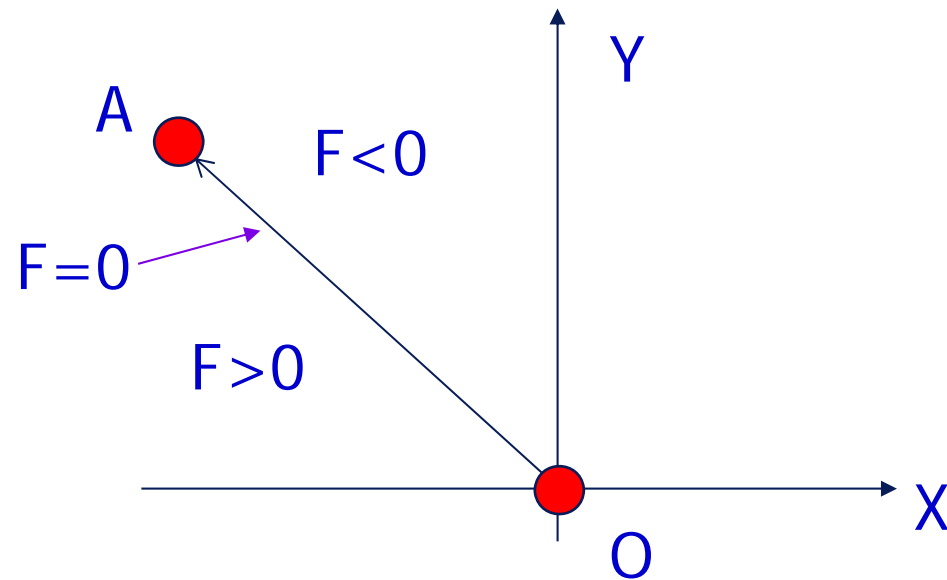
$$F(x, y) = x_0 y - y_0 x$$

1、建立偏差函数 $F(x, y) = -4y - 3x$

2、规定走向

$$F \geq 0, y+1$$

$$F < 0, x-1$$



3、一般曲线的扫描转换

直线扫描示例—解

3、偏差判别

$$F(x-1, y) = F(x, y) + 3$$

$$F(x, y+1) = F(x, y) - 4$$

4、终点判别

x向步数为**4**， **y**向步数为**3**，故选择**y**方向为终点判别方向。亦可取**x**和**y**方向的步数之和**7**为终点判别标志。

3、一般曲线的扫描转换

直线扫描示例—运算过程

$$F \geq 0, y+1$$

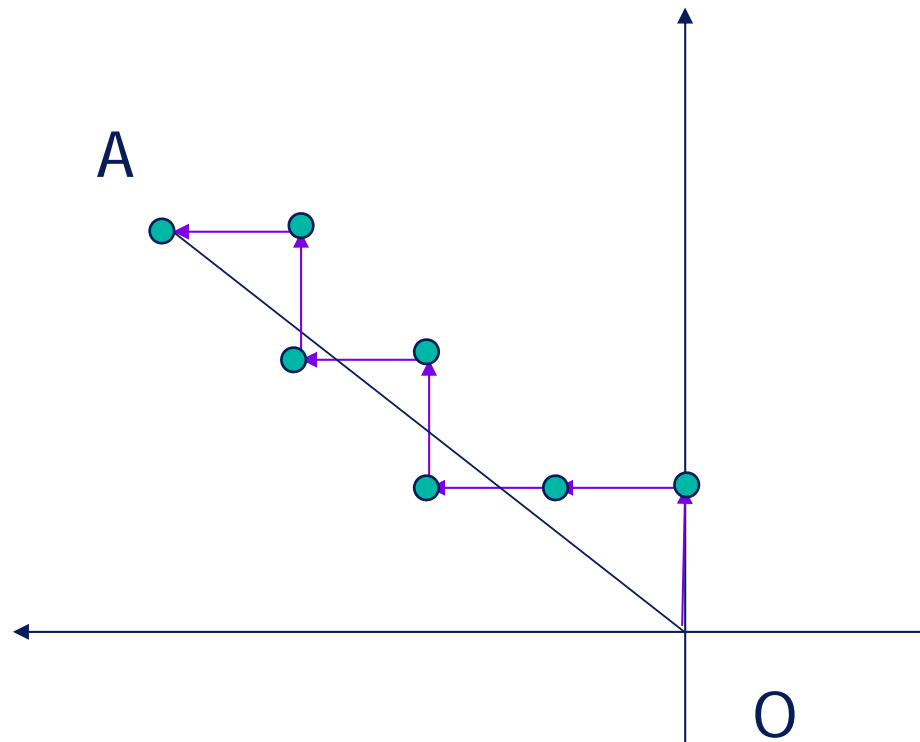
$$F < 0, x-1$$

步序	偏差判别	进向选择	到达点坐标	终点步数判别
1	0	Y+1	(0,1)	4
2	-4<0	X-1	(-1,1)	4-1=3
3	-1<0	X-1	(-2,1)	3-1=2
4	2>0	Y+1	(-2,2)	2
5	-2<0	X-1	(-3,2)	2-1=1
6	1>0	Y+1	(-3,3)	1
7	-3<0	X-1	(-4,3)	1-1=0

取x方向的步数

3、一般曲线的扫描转换

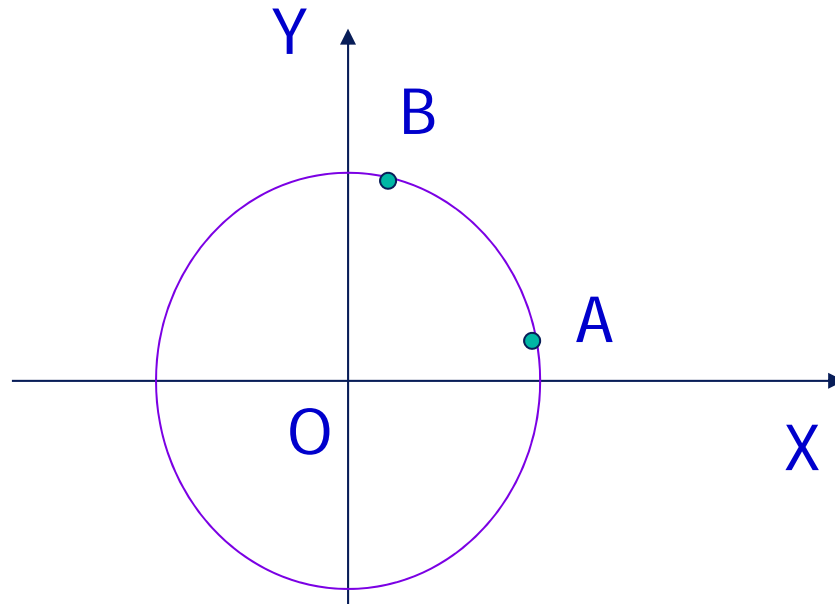
直线扫描示例-过程示意



3、一般曲线的扫描转换

3.3 逐点比较法进行圆弧扫描

问题：圆心在原点的圆上从A点扫描至B点



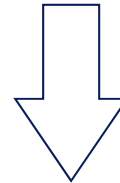
3、一般曲线的扫描转换

圆弧扫描—建立偏差函数

1、根据圆弧方程建立偏差函数

圆弧方程

$$x^2 + y^2 = R^2$$

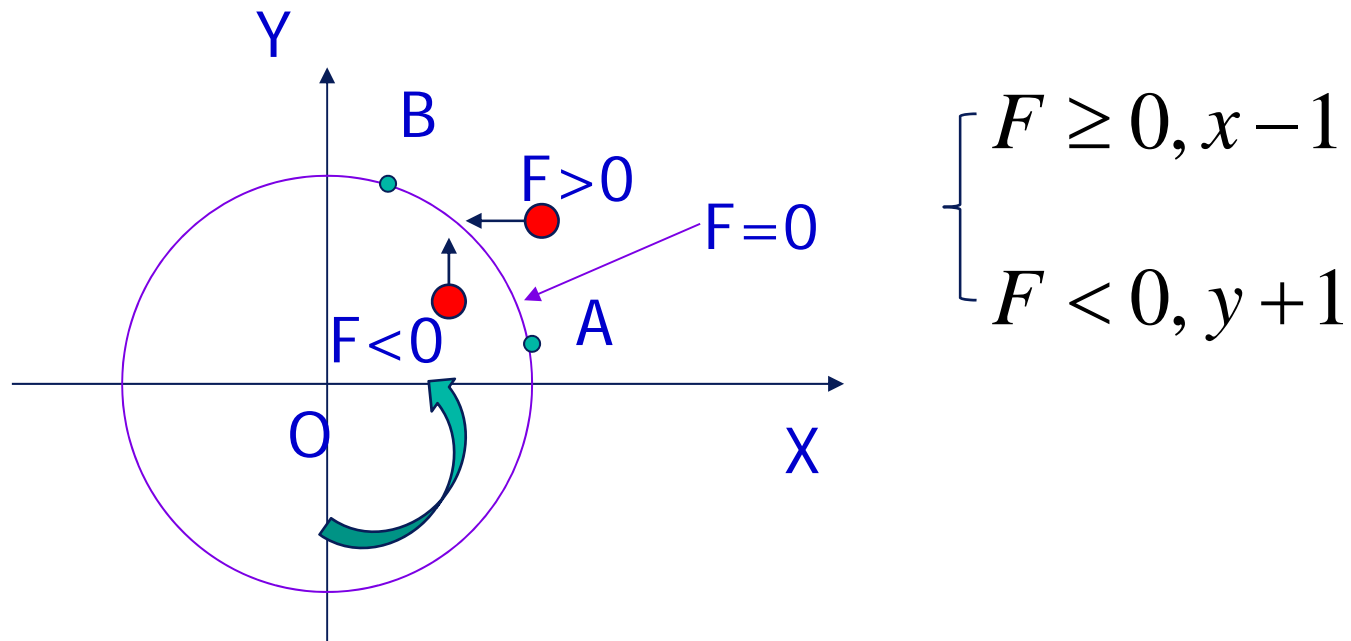


偏差函数 $F(x, y) = x^2 + y^2 - R^2$

3、一般曲线的扫描转换

圆弧扫描—规定走向

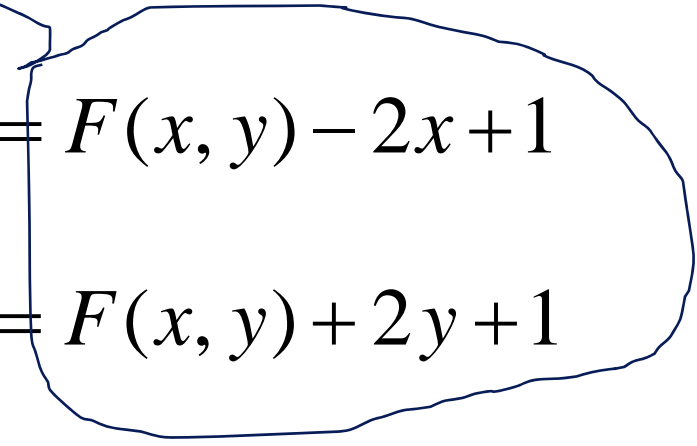
- 2、根据偏差函数决定各区域偏差符号，
然后决定走向（假定逆时针）



3、一般曲线的扫描转换

圆弧扫描—偏差判别

3、利用偏差函数计算偏差值，亦可进行简化偏差计算



$F(x-1, y) = (x-1)^2 + y^2 - R^2 = F(x, y) - 2x + 1$

$F(x, y+1) = x^2 + (y+1)^2 - R^2 = F(x, y) + 2y + 1$

3、一般曲线的扫描转换

圆弧扫描—终点判别

- 4、利用X和Y方向的步长数进行终点判别
以步数大者为判别方向，否则以两者之和为判别要素

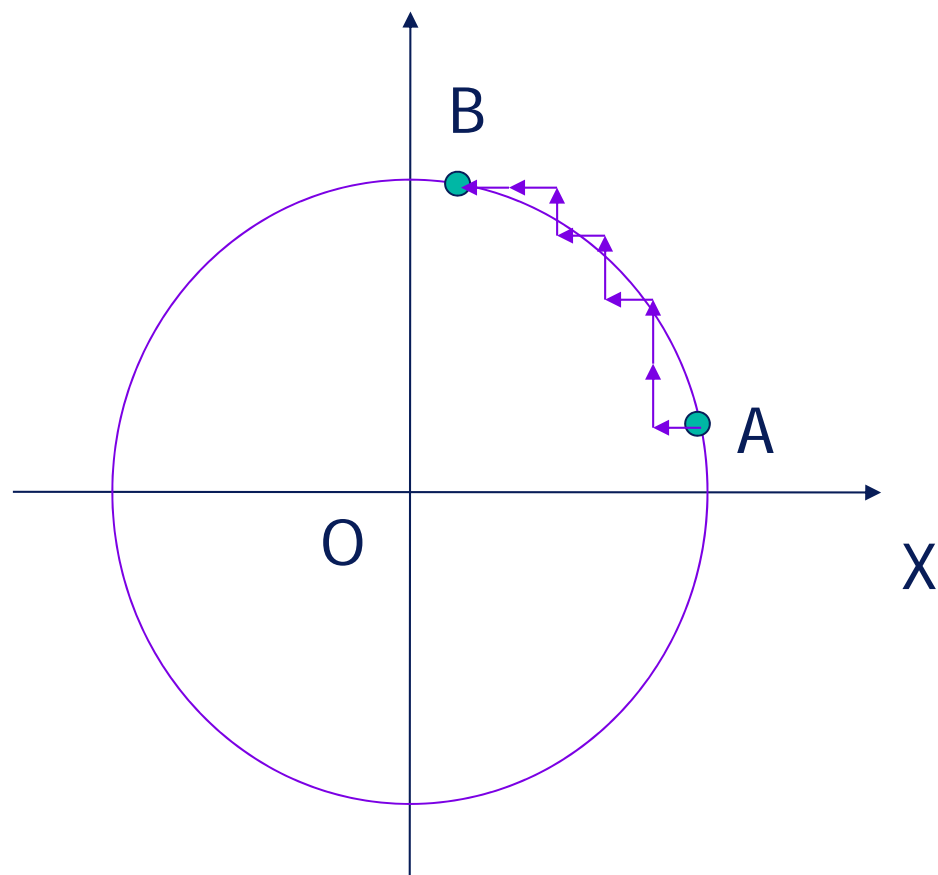
$$\text{X向步数} \quad |(x_b - x_a) / h|$$

$$\text{y向步数} \quad |(y_b - y_a) / h|$$

h为步长

3、一般曲线的扫描转换

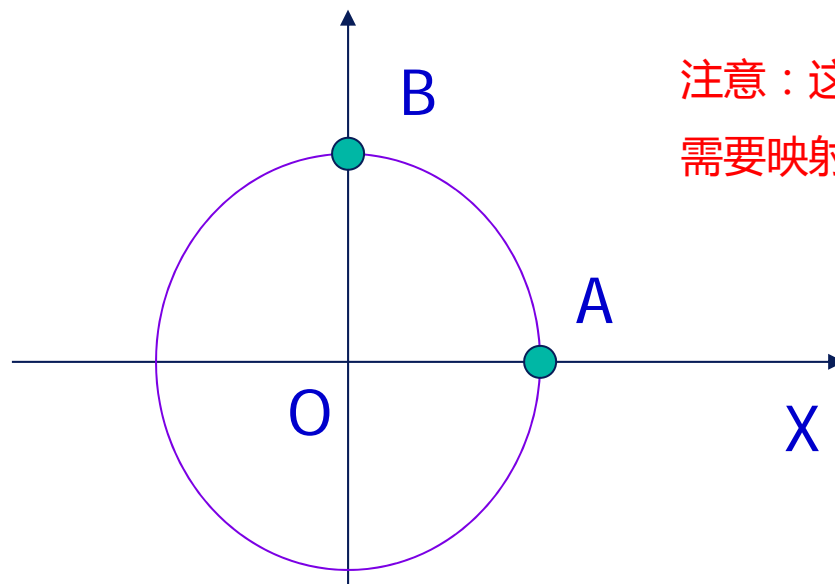
圆弧扫描-过程示意



3、一般曲线的扫描转换

圆弧扫描示例

例：以原点为圆心，0.5为半径,由点 $(0.5,0)$ 到点 $(0,0.5)$ 逆时针扫描的过程。步长为0.1。



注意：这里的坐标不代表像素数量
需要映射转换到像素“坐标系”

3、一般曲线的扫描转换

圆弧扫描示例—要点

由于步长为0.1，故须进行变换。

题目变成：



以原点为圆心，5为半径,由点(5,0)到点(0,5) **逆时针扫描**的过程。步长为1。

3、一般曲线的扫描转换

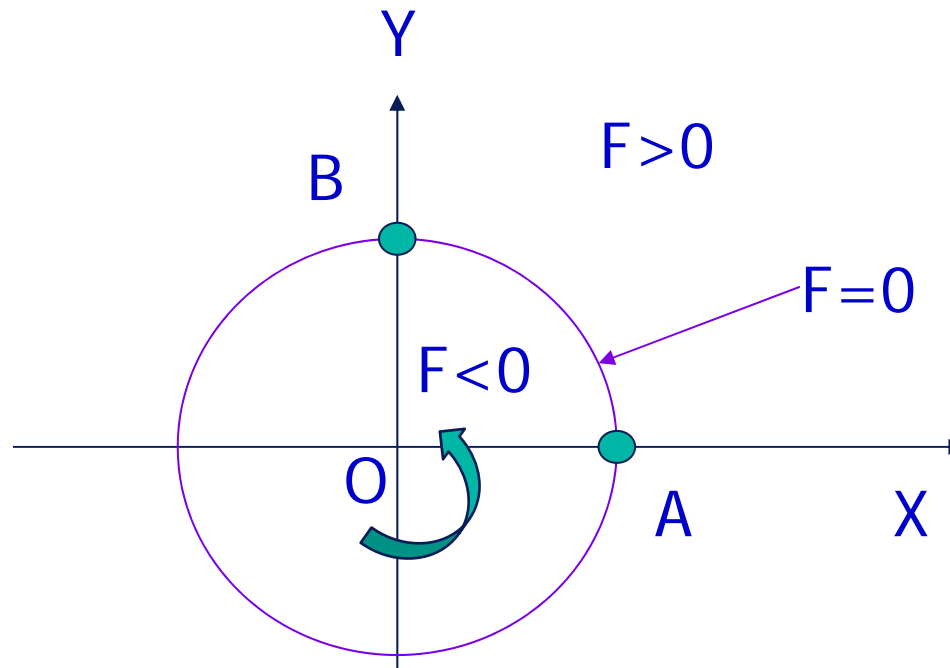
圆弧扫描示例—解

1、建立偏差函数 $F(x, y) = x^2 + y^2 - 25$

2、规定走向

$$F \geq 0, x-1$$

$$F < 0, y+1$$



3、一般曲线的扫描转换

圆弧扫描示例——解

3、偏差判别

$$F(x_{i+1}, y_{i+1}) = F(x_i, y_i) - 2x_i + 1$$

$$F(x-1, y) = F(x, y) - 2x + 1$$

$$F(x, y+1) = F(x, y) + 2y + 1$$

4、终点判别

x向步数为**5**， **y**向步数为**5**，故选择**x**和**y**方向的步数之和**10**为终点判别标志。

3、一般曲线的扫描转换

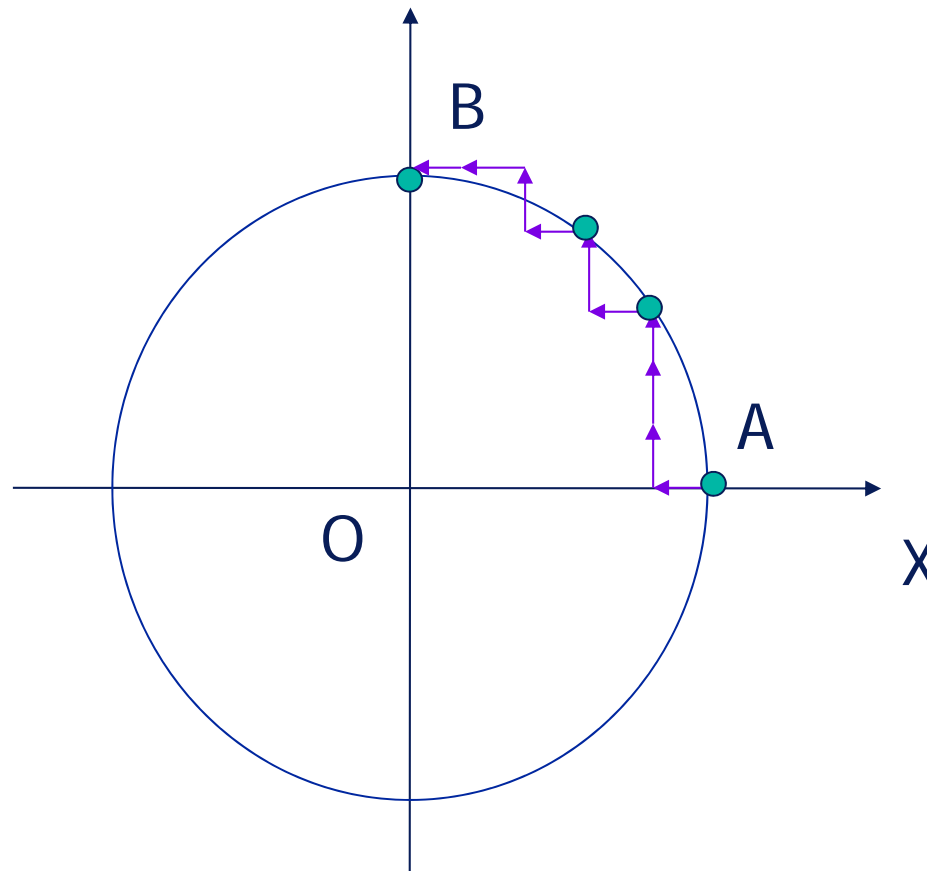
圆弧扫描示例—运算过程

步序	偏差判别	进向选择	到达点坐标	终点判别
1	0 $F(x_0, y_0)$	X-1	(4,0)	10-1=9
2	-9<0	Y+1	(4,1)	9-1=8
3	-8<0	Y+1	(4,2)	8-1=7
4	-5>0	Y+1	(4,3)	7-1=6
5	0	X-1	(3,3)	6-1=5
6	-7>0	Y+1	(3,4)	5-1=4
7	0	X-1	(2,4)	4-1=3
8	-8<0	Y+1	(2,5)	3-1=2
9	4>0	X-1	(1,5)	2-1=1
10	1>0	X-1	(0,5)	1-1=0

$F(x_1, y_1)$

3、一般曲线的扫描转换

圆弧扫描示例-过程示意



总结

- 1、综述
- 2、直线的扫描转换
- 3、一般曲线的扫描转换

作业

P248 -1



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



问题？
谢谢！

