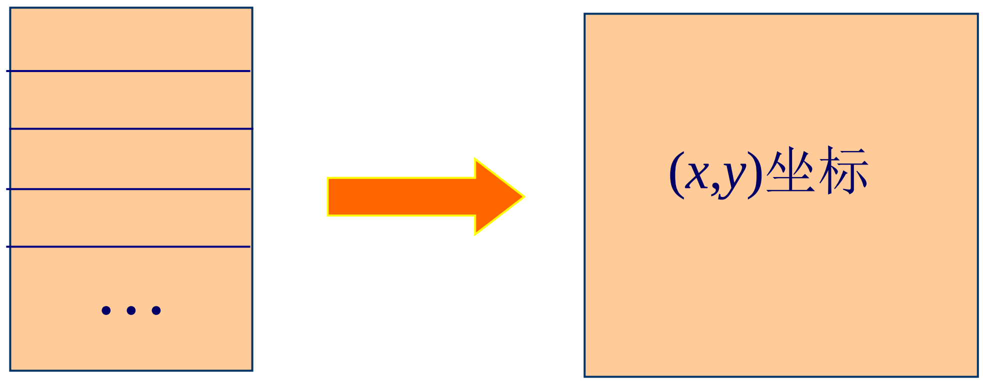
第三章 基本图形生成算法

图形的扫描转换

基本图形生成算法

- 图元扫描转换
  - 直线段扫描转换
  - 圆弧扫描转换
- 实区域填充

光栅图形中点的表示



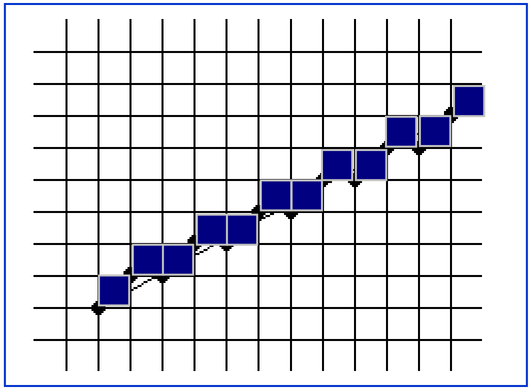
地址线性表

1D表示

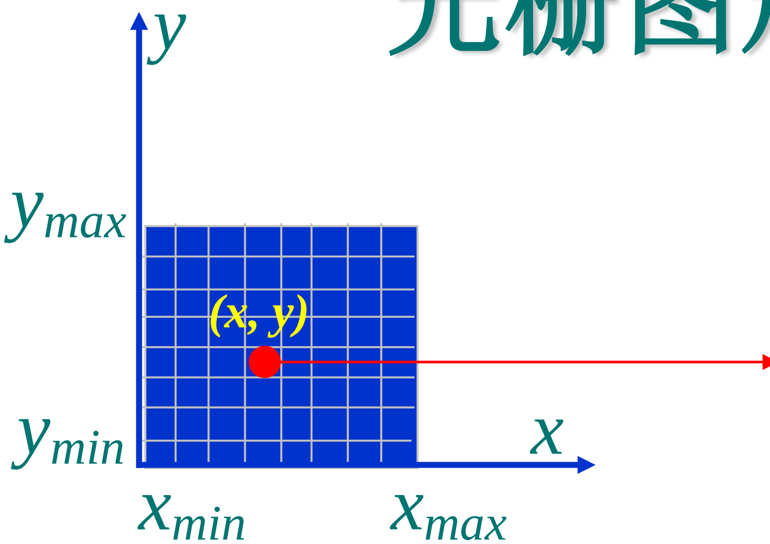
显示屏幕

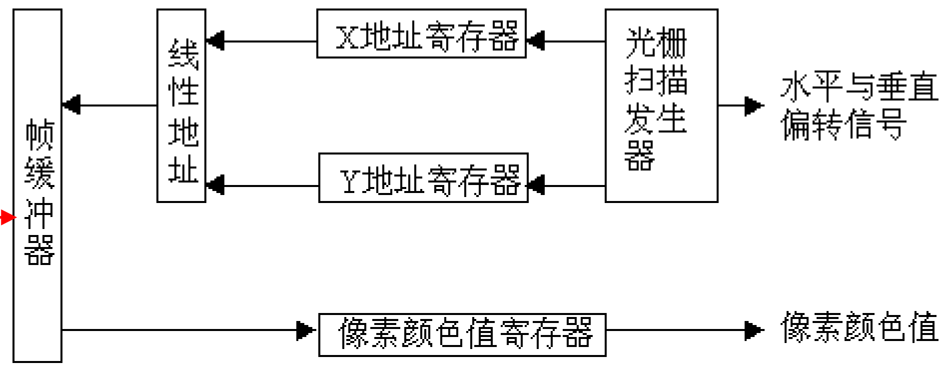
2D表示

像素由其左下角坐标表示



光栅图形中点的表示





$$\text{地址} = (x_{max}-x_{min}) * (y-y_{min}) + (x-x_{min}) + \text{基地址}$$

每行像素点数

行数

行中位置

光栅图形中点的表示

$$\text{Address}(x,y) = (x_{max}-x_{min}) * (y-y_{min}) + (x-x_{min}) + \text{基地址}$$

$$= k_1 + k_2 y + x$$

对像素连续寻址时，如何减少计算量？

$$\text{Address}(x\pm 1,y) = k_1 + k_2 y + (x\pm 1) = \text{Address}(x,y) \pm 1$$

$$\text{Address}(x,y\pm 1) = k_1 + k_2(y \pm 1) + x = \text{Address}(x,y) \pm k_2$$

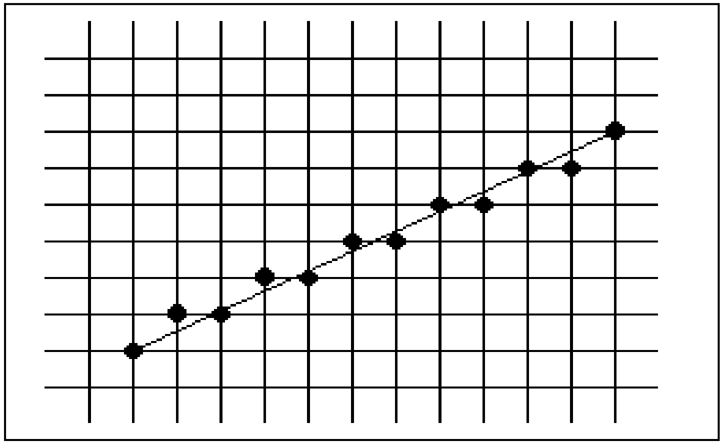
$$\text{Address}(x\pm 1,y\pm 1) = k_1 + k_2(y \pm 1) + (x\pm 1)$$

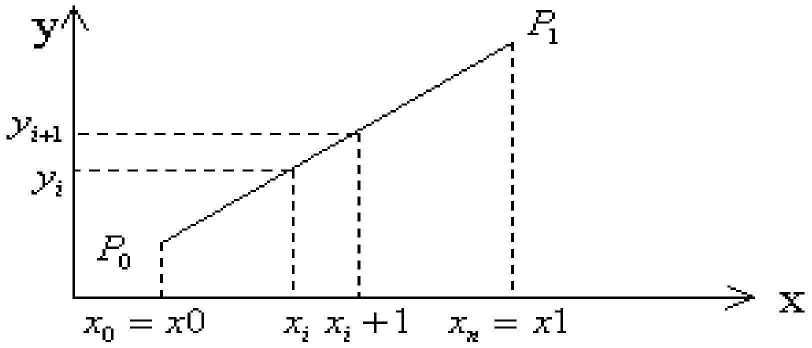
$$= \text{Address}(x,y) \pm k_2 \pm 1$$

增量法的优点？

直线段扫描转换

- 假设
  - 像素间均匀网格，整数型坐标系，直线段斜率0<m<1
    - X方向每次迭代都增1，y方向不一定
  - 对m＞1，x、y互换





直线段的扫描转换算法

- ***直线的扫描转换***
  - 确定最佳逼近于该直线的一组象素
  - 按扫描线顺序，对这些象素进行写操作
- ***三个常用算法*:**

  1数值微分法（DDA）

  2中点画线法

  3Bresenham算法。

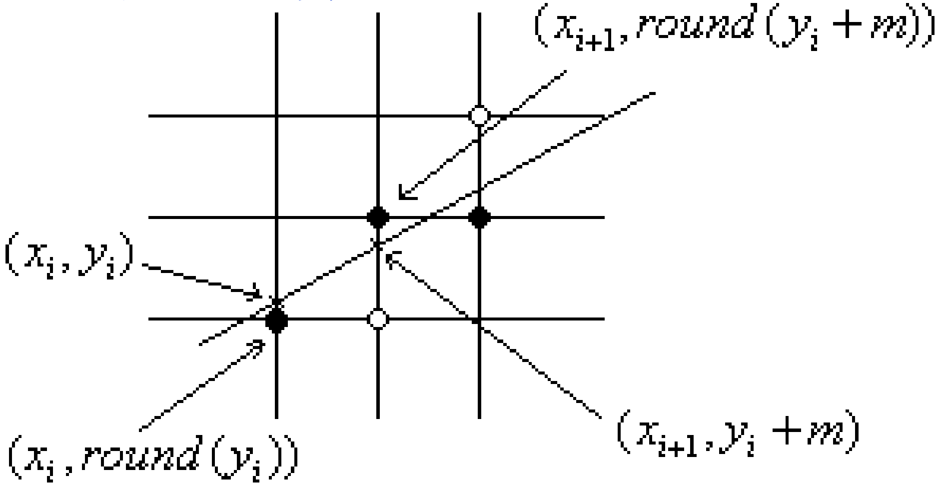
数值微分(DDA)法(1/5)

- 已知线段端点：$P_0(x_0,y_0)$, $P_1(x_1,y_1)$
- 直线方程

$$y=kx+b$$

$$\{(x_i, y_i)\},\ i=0,\ldots.n.$$

- 浮点数取整：$y_i=round(y_i)=(int)(y_i+0.5)$
  - 用到浮点数的乘法、加法和取整运算



数值微分(DDA)法(2/5)

- ***增量算法***
  - $y_{i+1}=kx_{i+1}+b=k(x_i+1)+b=y_i+k$
  - $(x_i,y_i)\to(x_i+1,y_i+k)$
- ***缺点*:**
  - 有浮点数取整运算
  - 不利于硬件实现
  - 效率低
  - 仅适用于$|k| \leqslant 1$的情形:x每增加1,y最多增加1。当 $|k|>1$时，必须把x，y互换。

数值微分(DDA)法(3/5)

- digital differential analyzer
- *基本思想*
  - 用数值方法解微分方程

$dx/dt = \Delta x$

$dy/dt = \Delta y$

$x_{n+1} = x_n + \epsilon\Delta x$

$y_{n+1} = y_n + \epsilon\Delta y$

如何选取ε？

选取ε的原则：使$0.5 \le |\epsilon\Delta x|, |\epsilon\Delta y| \le 1$

数值微分(DDA)法(4/5)

- *对称的DDA*
  - 取$\epsilon=2^{-n}$
  - 使 $2^{n-1}\le \max(|\Delta x|,|\Delta y|)\le 2^n$
- *简单的DDA*
  - 取$\epsilon= 1/\max(|\Delta x|,|\Delta y|)$
  - 使 $\epsilon|\Delta x|$, $\epsilon|\Delta y|$中必有一个是单位步长
  - Δx为最大时， $\epsilon\Delta x =1$, $\epsilon\Delta x =k$
  - Δy为最大时， $\epsilon\Delta y =1$, $\epsilon\Delta y =1/k$

数值微分(DDA)法(5/5)

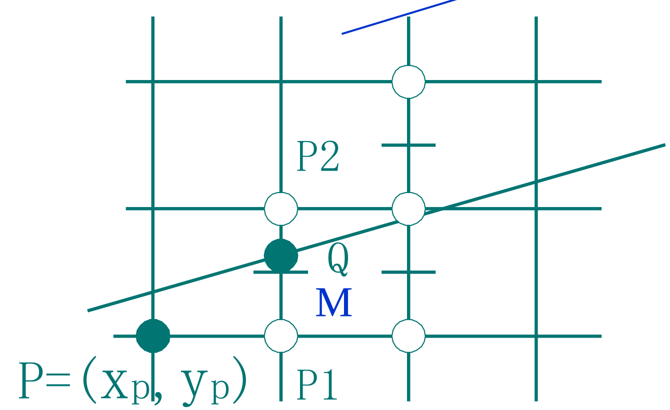
- 缺点：
  - 浮点数运算
  - 不易硬件实现

中点画线法（1/4）

- 问题：判断距离理想直线最近的下一个象素点
- 已知：线段两端点(x0,y0)，(x1,y1)
- 直线方程：F(x,y)=ax+by+c=0
  - a=y0-y1 （<=0）
  - b=x1-x0 （>-a）
  - c=x0y1-x1y0

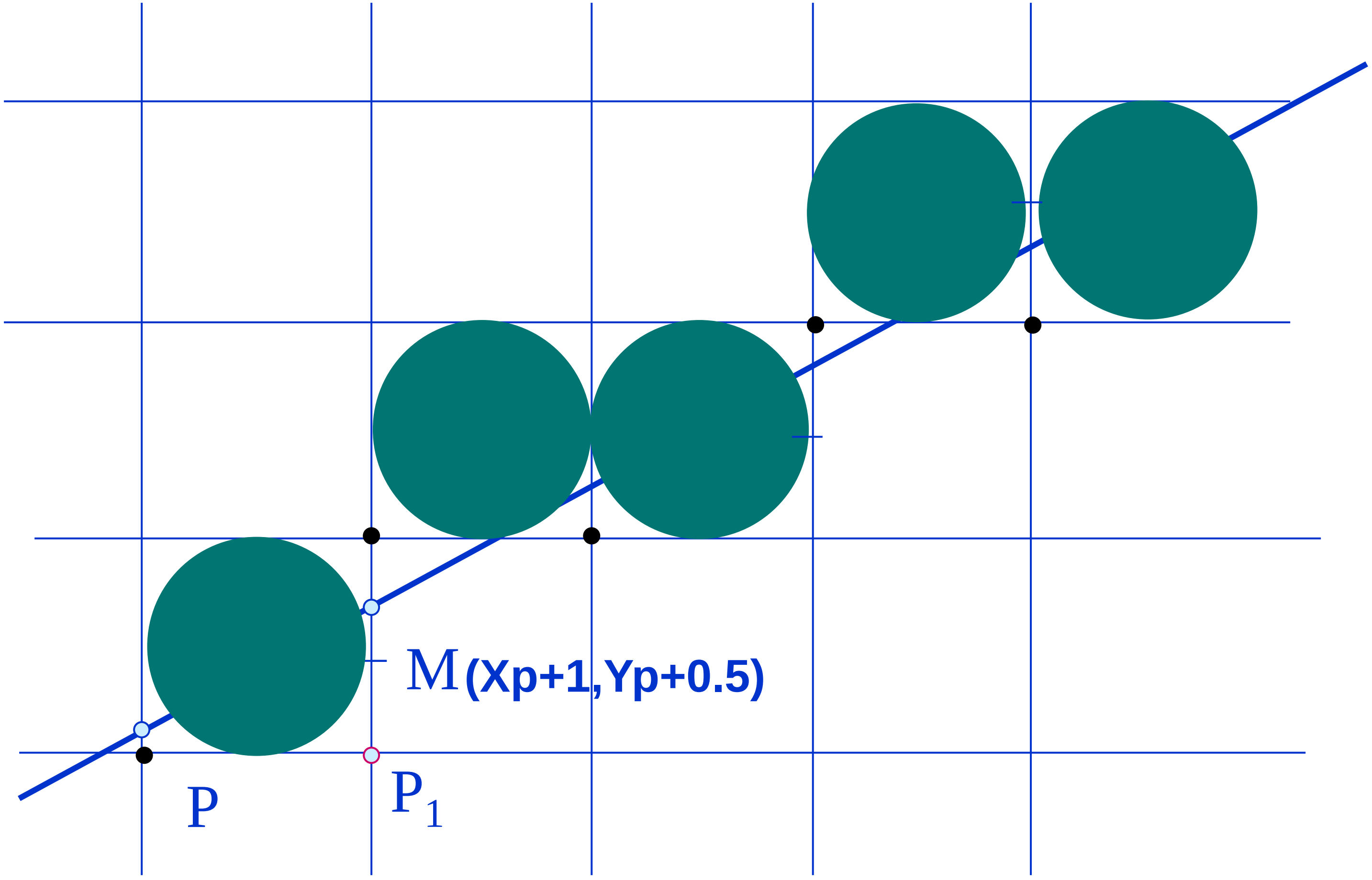
如何判断*M*点在*Q*点上方还是在*Q*点下方?

$$
\begin{cases}
F(x,y)=0 & \text{点在直线上面} \\
F(x,y)>0 & \text{点在直线上方} \\
F(x,y)<0 & \text{点在直线下方}
\end{cases}
$$

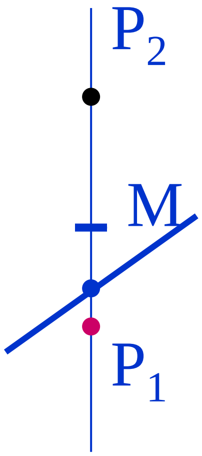


中点画线法（2/4）

- 直线上方点： **F(x,y)＞0**　　直线下方点： **F(x,y)＜0**
- 构造判别式： **d=F(M)=F(Xp+1,Yp+0.5)**
- 由**d＞0**，**d＜0**可判定下一个象素



- *分两种情形考虑再一下个象素的判定*：



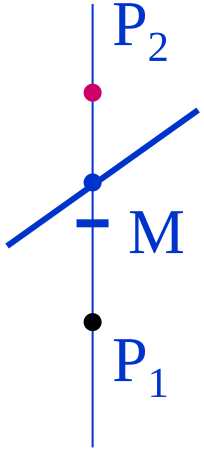
- 若d≥0，中点M在直线上方，取正右方象素P1 (Xp+1,Yp)
  - 再下一个象素的判别式为：

d1=F((Xp+1)+1,Yp+0.5)=a(Xp+2)+b(Yp+0.5)+c

= d+a

d的增量为a

- 若d＜0，中点M在直线下方，取右上方象素P2 (Xp+1,Yp+1)
  - 再下一个象素的判别式为：



d2=F((Xp+1)+1,(Yp+1)+0.5)= a(Xp+2)+b(Yp+1.5)+c

=d+a+b

d的增量为a+b

中点画线法（4/4）

- d的初始值
  - d0=F(X0+1,Y0+0.5)
    =F(X0,Y0)+a+0.5b
    =a+0.5b
  - 用2d代替d后，d0=2a+b
  - d的增量都是整数

因(X0,Y0)在直线上，
所以F(X0,Y0)=0

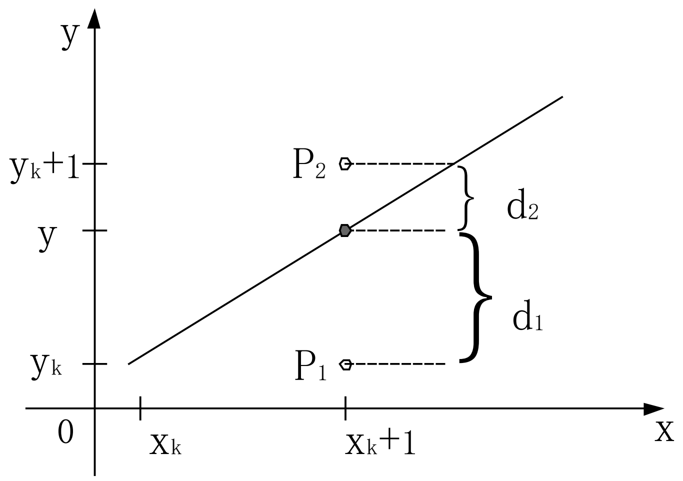
- 优点：
  - 只有整数运算，不含乘除法
  - 可用硬件实现

Bresenham画线算法（1/11）

- 使用最广泛
- 与中点画线法的思想类似
- 由误差项符号决定下一个象素取正右方像素还是右上方像素

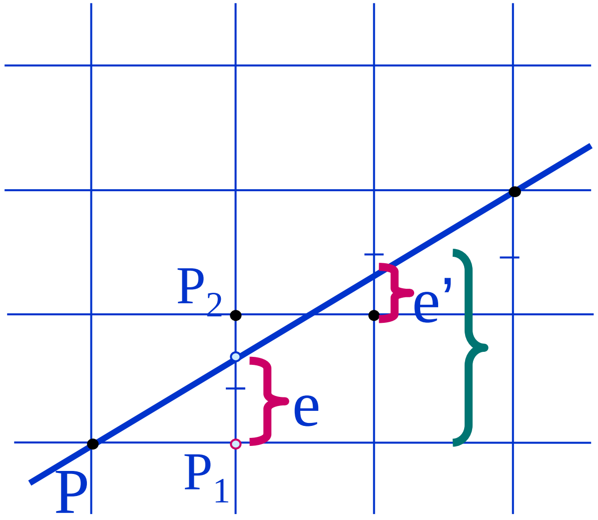
Bresenham画线算法（2/11）

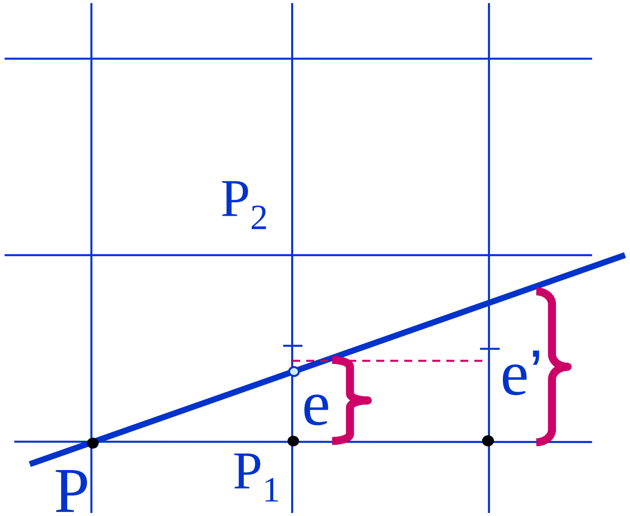
- *<u>基本思想</u>*
  - 比较从理想直线到位于直线上方的像素的距离d1和相邻的位于直线下方的像素的距离d2
  - 根据距离误差项的符号确定与理想直线最近的象素



Bresenham画线算法（3/11）

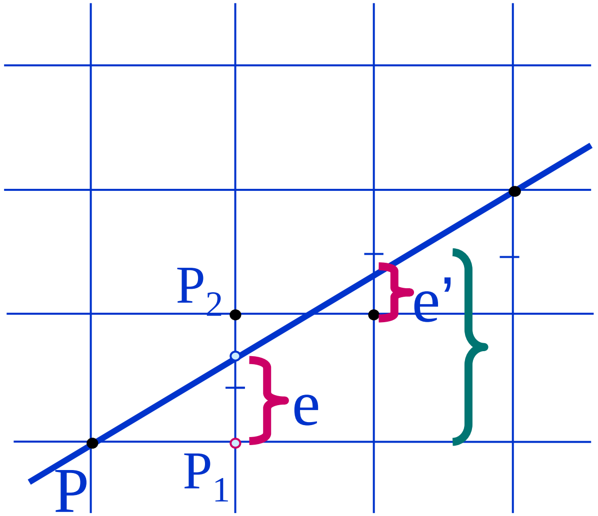
- *最大位移方向每次走一步*
  - k<1时，x为最大位移方向
- *y方向走步与否*
  - 取决于误差e值的大小
- *误差计算*
- 初值：e0= Δy/ Δx
- 当e≥0.5时，最接近P2(xi+1,yi+1)
  - y方向走一步
- 当e<0.5时，最接近P1(xi+1,yi)
  - y方向不走步

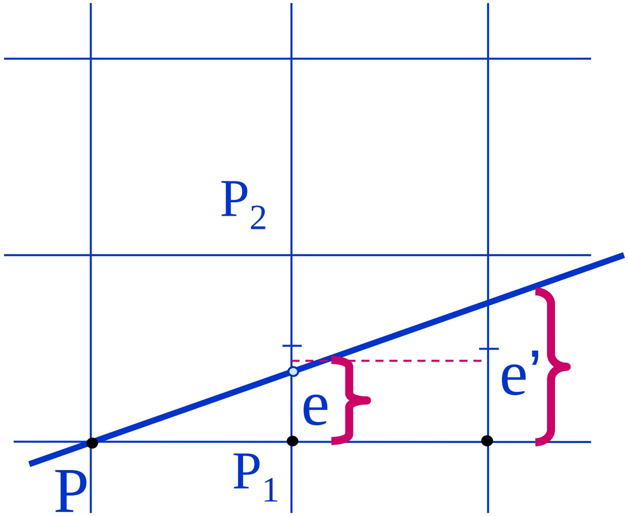




Bresenham画线算法（4/11）

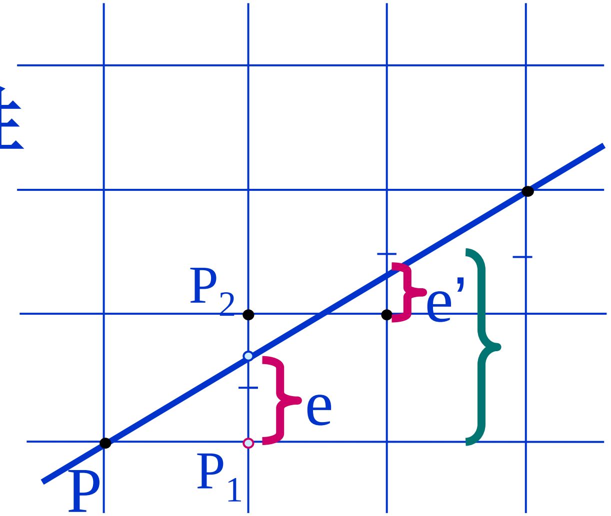
- 为方便与0比较，设e=e-0.5
- e0=Δy/ Δx-0.5
- 当e≥0时，最接近P2(xi+1,yi+1)
  - y方向走一步
- 当e<0时，最接近P1(xi+1,yi)
  - y方向不走步
- 有除法，不宜硬件实现

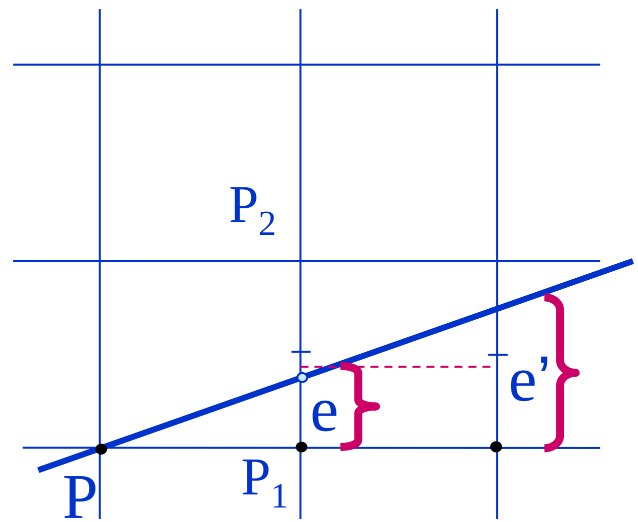




Bresenham画线算法（5/11）

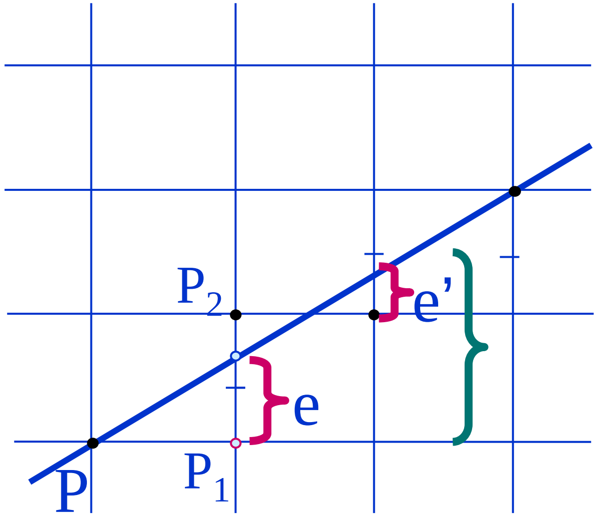
- 设e=e×2Δx，不影响判断的准确性
- e0=2Δy - Δx
- 当e≥0时，最接近P2(xi+1,yi+1)
  - y方向走一步
- 当e<0时，最接近P1(xi+1,yi)
  - y方向不走步



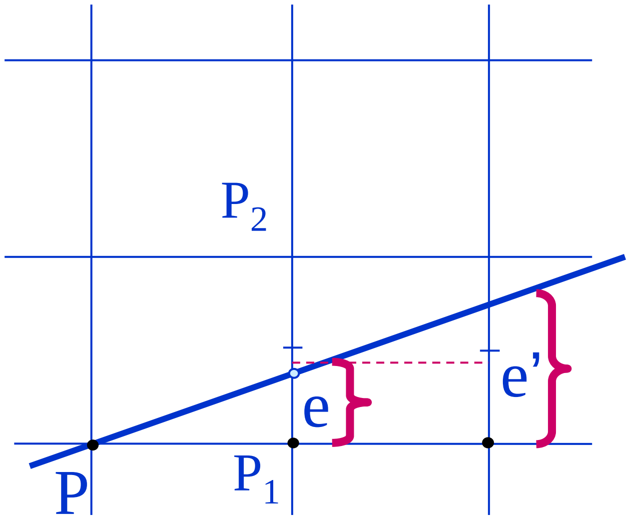


Bresenham画线算法（6/11）

- *下一步误差的计算*
- 当e≥0时，y方向走一步
  - $e' = 2\Delta y / \Delta x - 1 = e + \Delta y / \Delta x - 1$
  - $e' = e + 2\Delta y - 2\Delta x$



- 当e<0时，y方向不走步
  - $e' = 2\Delta y / \Delta x = e + \Delta y / \Delta x$
  - $e' = e + 2\Delta y$

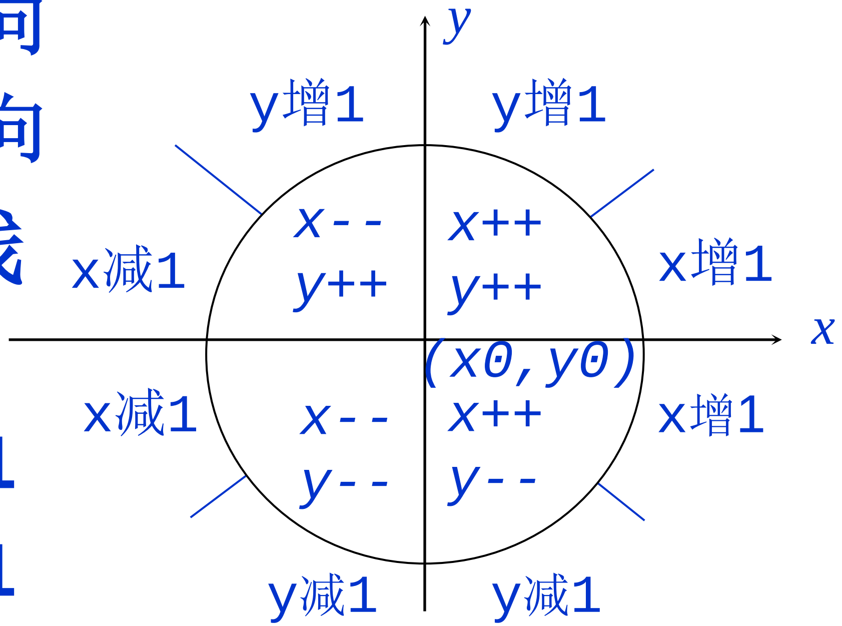


Bresenham画线算法（7/11）

- 先确定最大位移方向
- 确定误差e的计算方法，并根据e确定在非最大位移方向上如何走步

Bresenham画线算法（8/11）

- ***先确定最大位移方向***
  - **|k|<1时，x为最大位移方向**
  - **|k|>1时，y为最大位移方向**
- **增1还是减1，取决于直线所在象限**
  - **Δx≥0时，s1=1,否则s1=-1**
  - **Δy≥0时，s2=1,否则s2=-1**



Bresenham画线算法（9/11）

- ***<u>确定误差e的计算方法，并根据e确定在非最大位移方向上如何走步</u>***
- 误差初值的计算
  - **|k|<1时， e=2|Δy| - |Δx|**
  - **|k|>1时， e=2|Δx| - |Δy|**

Bresenham画线算法（10/11）

- ***确定误差e的计算方法，并根据e确定在非最大位移方向上如何走步***
  - **e<0,**不走步
    - **|k|<1**时，　**x=x+s1, e=e+2|Δy|**
    - **|k|>1**时，　**y=y+s2, e=e+2|Δx|**
  - **e≥0,**走步
    - **|k|<1**时，　**x=x+s1, y=y+s2, e=e+2|Δy|-2|Δx|**
    - **|k|>1**时，　**y=y+s2, x=x+s1, e=e+2|Δx|-2|Δy|**

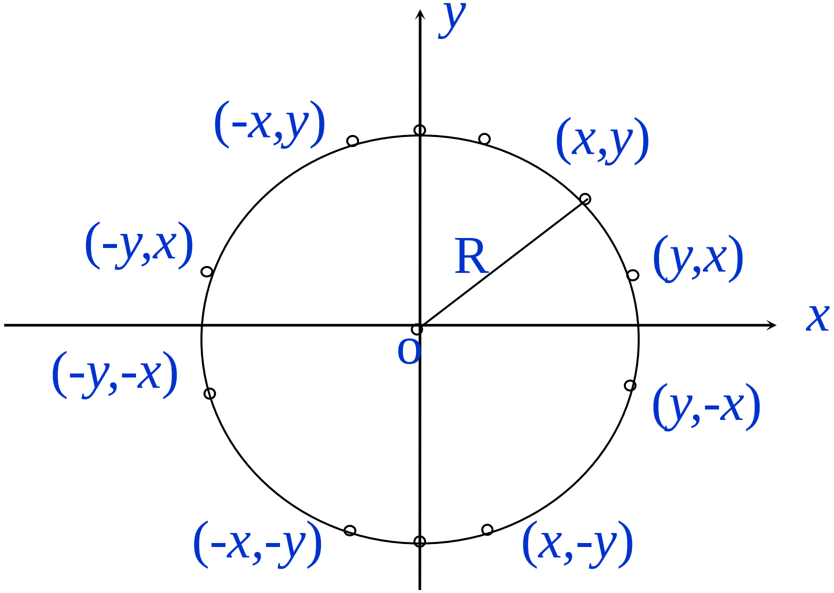
Bresenham画线算法（11/11）

- 优点
  - 整数运算，速度快
  - 精度高
  - 乘2运算可用移位实现，适于硬件实现

圆弧的扫描转换

- *圆的八对称性*
  - 只考虑第二个八分圆
- 假设圆心在原点

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



圆弧的扫描转换

- ***两种直接离散生成方法***
  - 离散点
    - 开方运算
  - 离散角度
    - 三角函数运算
- ***缺点*:**
  - 计算量大
  - 所画像素位置间的间距不一致

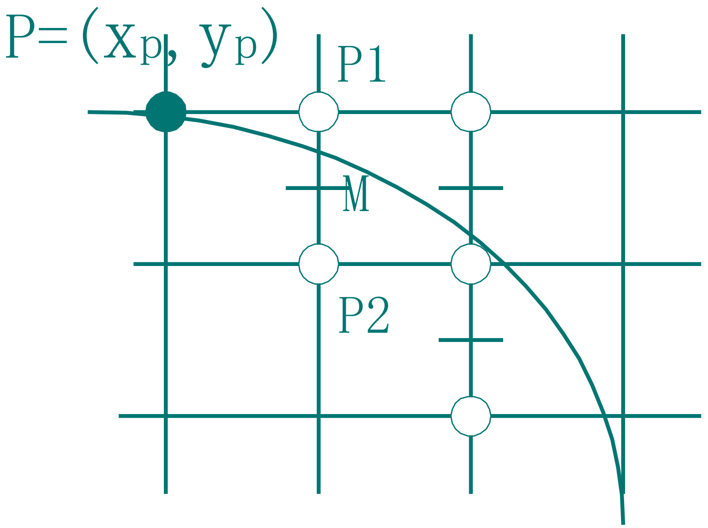
$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$

$$\begin{cases} x_i = x_c + r \cdot \cos\alpha_i \\ y_i = y_c + r \cdot \sin\alpha_i \end{cases}$$

中点画圆法（1/2）

- $F(X,Y)=X^2+Y^2-R^2=0$
- 中点 M=(Xp+1,Yp-0.5)

$$d = F(M) = F(x_p + 1, y_p - 0.5)$$
$$= (x_p + 1)^2 + (y_p - 0.5)^2 - R^2$$



- 当F(M)＜0时，M在圆内，P1距离圆弧近，取P1
- 当F(M)＞0时，M在圆外，P2距离圆弧近，取P2

中点画圆法（2/2）

若 **d<0，取P1**为下一象素，再下一象素的判别式为

$$d' = F(x_p + 2, y_p - 0.5) = (x_p + 2)^2 + (y_p - 0.5)^2 - R^2 = d + 2x_p + 3$$

若**d>=0, 取P2**为下一象素，再下一象素的判别式为

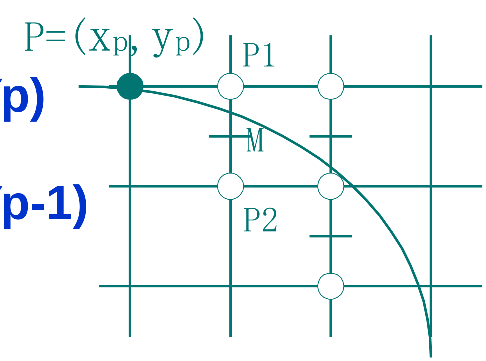
$$d' = F(x_p + 2, y_p - 1.5) = (x_p + 2)^2 + (y_p - 1.5)^2 - R^2 = d + 2(x_p - y_p) + 5$$

初始象素是（**0,R**），判别式**d**的初值为

$$d_0 = F(1, R - 0.5) = 1.25 - R$$

P1(Xp+1,Yp)

P2(Xp+1,Yp-1)



使用**e=d-0.25**代替**d**
e0=1-R

DDA画圆法（1/3）

- 圆的方程：$f(x,y)=x^2+y^2-R^2=0$
- 全微分：$df(x,y)=2xdx+2ydy=0$
- 微分方程：$dy/dx=-x/y$
- 递推方程:

$$(y_{n+1}-y_n)/(x_{n+1}-x_n)=-\varepsilon x_n/\varepsilon y_n$$

$$x_{n+1}-x_n=\varepsilon y_n$$

$$y_{n+1}-y_n=-\varepsilon x_n$$

实际画出的曲线不是圆，而是螺旋线，为什么？

DDA画圆法（2/3）

- 将递推公式写成矢量形式：

$$[x_{n+1} \quad y_{n+1}] = [x_n \quad y_n]\begin{bmatrix} 1 & -\varepsilon \\ \varepsilon & 1 \end{bmatrix}$$

- 构造一个行列式值为1的矩阵

$$\begin{bmatrix} 1 & -\varepsilon \\ \varepsilon & 1-\varepsilon^2 \end{bmatrix}$$

- 对应的圆方程递推关系为

$x_{n+1} = x_n + \epsilon y_n$

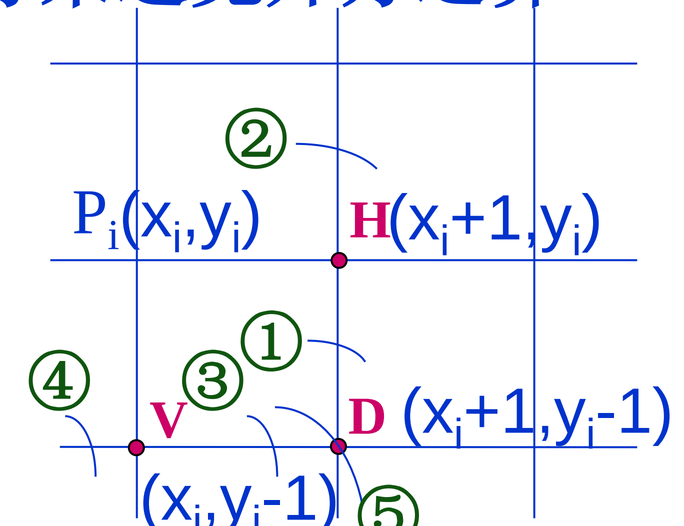
$y_{n+1} = -\epsilon x_n + (1-\epsilon^2) y_n = y_n - \epsilon x_{n+1}$

DDA画圆法（3/3）

- 针对不同象限及顺逆时针画圆，赋给ε适当的符号
- ε不同，圆形状不同， ε大近似椭圆

Bresenham画圆算法（1/7）

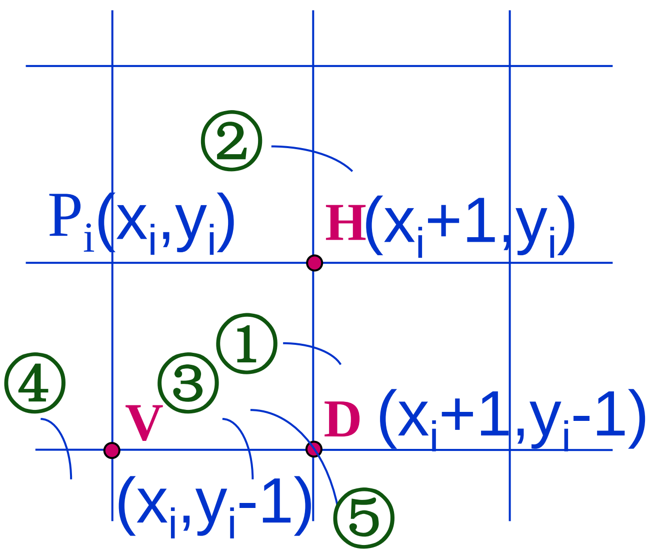
- 顺时针画第一四分圆，下一步选择哪个点？
- *基本思想*:
  - 通过比较像素与圆的距离平方来避免开方运算
- *下一像素有3种可能的选择*



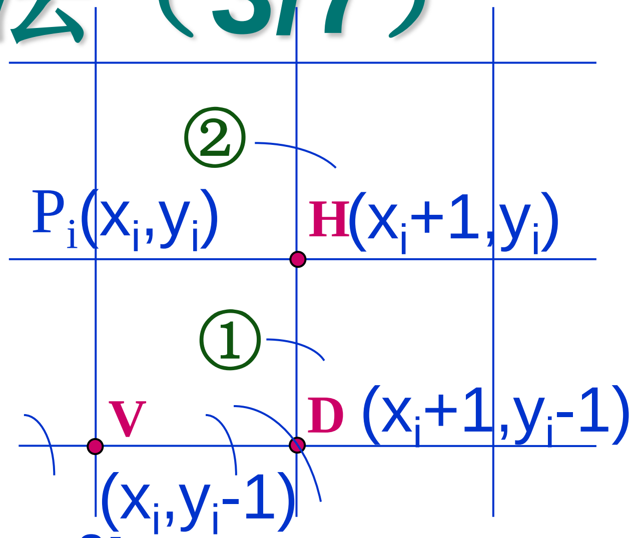
  - $m_H=|(x_i+1)^2+y_i^2-R^2|$
  - $m_D=|(x_i+1)^2+(y_i-1)^2-R^2|$
  - $m_V=|x_i^2+(y_i-1)^2-R^2|$
- *选择像素的原则*
  - 使其与实际圆弧的距离平方达到最小

Bresenham画圆算法（2/7）

- 圆弧与点(xi,yi)附近光栅网格的相交关系有5种
- *<u>右下角像素D (xi,yi)与实际圆弧的近似程度</u>*
  - $\Delta i=(x_i+1)^2+(y_i-1)^2-R^2$
  - 当Δi<0时，D在圆内，①②
  - 当Δi>0时，D在圆外，③④
  - 当Δi=0时，D在圆上，⑤



Bresenham画圆算法（3/7）



- 当$\Delta i<0$时，D在圆内，①②
- 情形①，选m_H，m_D 中最小者
- $d=m_H - m_D$

$$=|(x_i+1)^2+y_i^2-R^2| - |(x_i+1)^2+(y_i-1)^2-R^2|$$

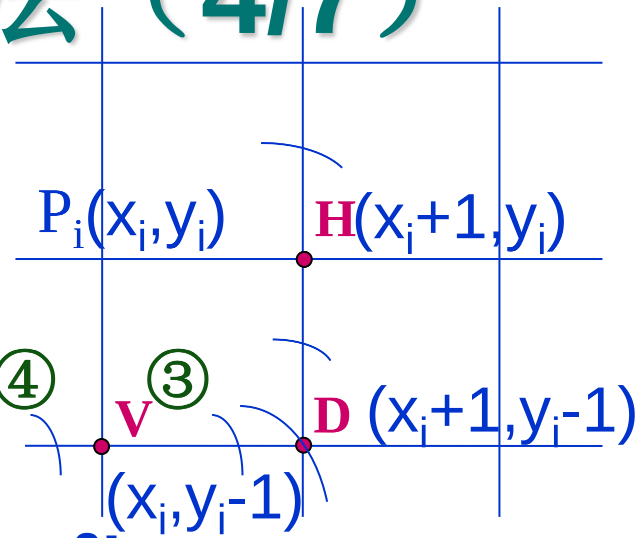
$$=(x_i+1)^2+y_i^2-R^2 + (x_i+1)^2+(y_i-1)^2-R^2$$

$$=2\ (\Delta i+y_i)-1$$

  - 若$d<0$，则选H
  - 若$d>0$，则选D
  - 若$d=0$，则选H

情形②也适用

Bresenham画圆算法（4/7）



- 当$\Delta i>0$时，D在圆外，③④
- 情形③，选m_V，m_D 中最小者
- $d'=m_D - m_V$

  $=|(x_i+1)^2+(y_i-1)^2-R^2| - |x_i^2+(y_i-1)^2-R^2|$

  $=(x_i+1)^2+(y_i-1)^2-R^2 + x_i^2+(y_i-1)^2-R^2$

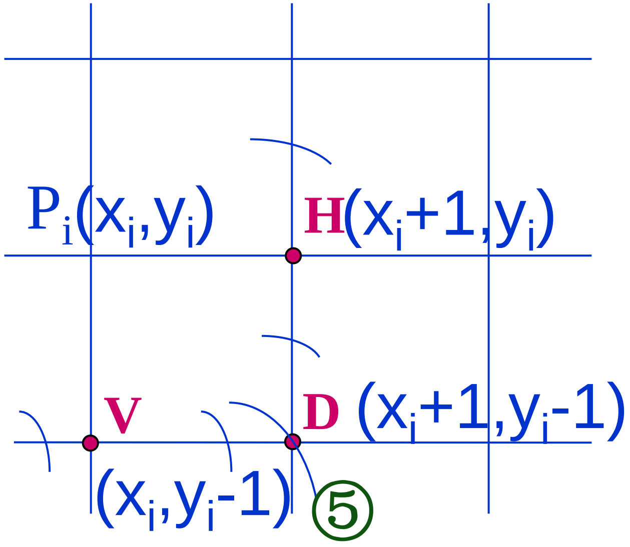
  $=2(\Delta i-x_i)-1$

  - 若$d'<0$，则选D
  - 若$d'>0$，则选V
  - 若$d'=0$，则选D

情形④也适用

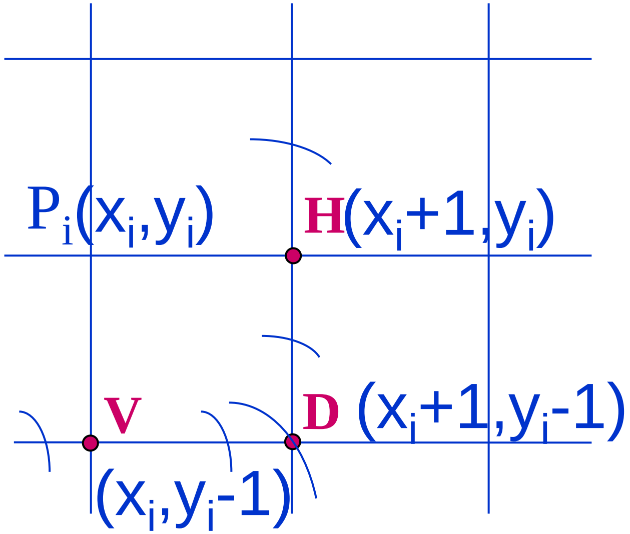
Bresenham画圆算法（5/7）

- 当$\Delta i=0$时，D在圆上，⑤
- 按d判别，有d>0，应选D
- 按d'判别，有d'<0，应选D

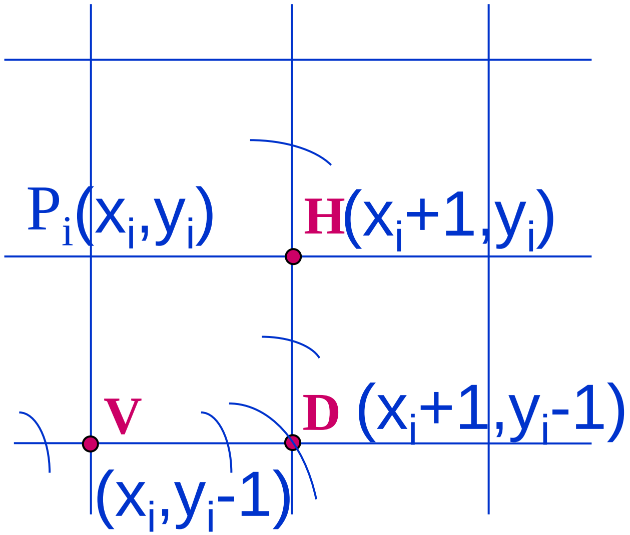


Bresenham画圆算法（6/7）

- 当Δi<0时,
  - 若d≤0，选H
  - 若d>0，选D
- 当Δi>0时,
  - 若d’ ≤0，选D
  - 若d’>0，选V
- 当Δi=0时，选D



Bresenham画圆算法（7/7）



- *判别式的递推关系*
- 当取H(xi+1,yi)时
  - $\Delta i+1=(x_i+1+1)^2+(y_i-1)^2-R^2=\Delta i+2(xi+1)+1$
- 当取V(xi,yi-1)时
  - $\Delta i+1=(x_i+1)^2+(y_i-1-1)^2-R^2=\Delta i-2(yi-1)+1$
- 当取D(xi+1,yi-1)时
  - $\Delta i+1=(x_i+1+1)^2+(y_i-1-1)^2-R^2=\Delta i+2(xi+1)-2(yi-1)+2$

多边形逼近法

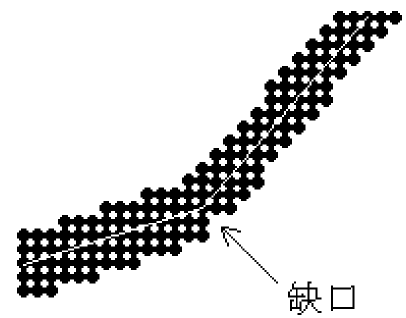
- 当圆的正内接多边形边数足够多时，可以用画该多边形近似代替画圆
- “以直代曲”的代表方法之一
- 内接正n边形顶点为$P_i(x_i, y_i)$
- 每条边对应的圆心角为 θ，则有

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

线画图元的属性控制（1/3）

- 线宽控制：刷子形状、朝向对线型的影响

1.用象素复制方法产生宽图元



优点：

线宽与线段的斜率有关

效率高，实现简单

缺点：

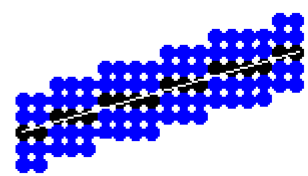
(1) 线宽较大时，不自然

(2)折线处有缺口

(3)宽度不符合要求

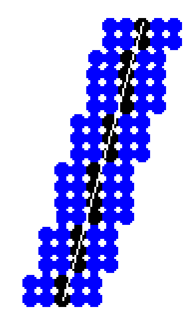
(4)对称问题：奇偶数像素，效果不同

竖直方向复制



$m \in (-1,1)$

水平方向复制



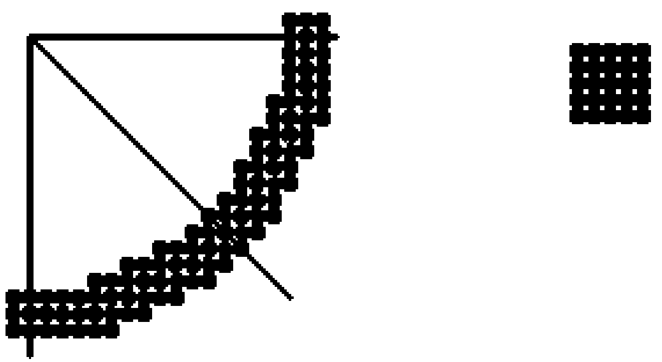
$m \notin (-1,1)$

线画图元的属性控制（2/3）

2.移动刷子产生宽图元

- 线宽变粗，刷子移动覆盖
- 线宽与线段的斜率有关

(p) (c)

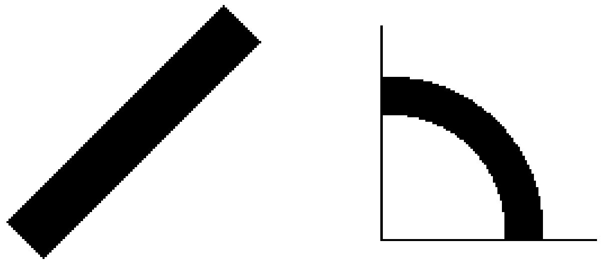


线画图元的属性控制（3/3）

3.用填充图形表示宽图元

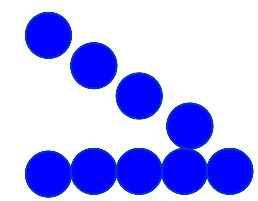
用等距线方法:

- 线宽均匀
- 端口处与边垂直
- 生成的图形质量高



线型控制

- 用位屏蔽器实现
  - 位屏蔽器中每一位对应的是一个像素，而不是单位长度，不能满足要求
  - 线型中的笔划长度与直线长度有关
    - 斜线笔划长度比水平或垂直线笔划长
    - 对工程图，这种变化是不允许的，它不符合国标规定



- 工程图，笔画作单独的扫描转换

1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0

● ● ● ● ○ ○ ● ● ● ● ○ ○ ● ● ● ● ○ ○