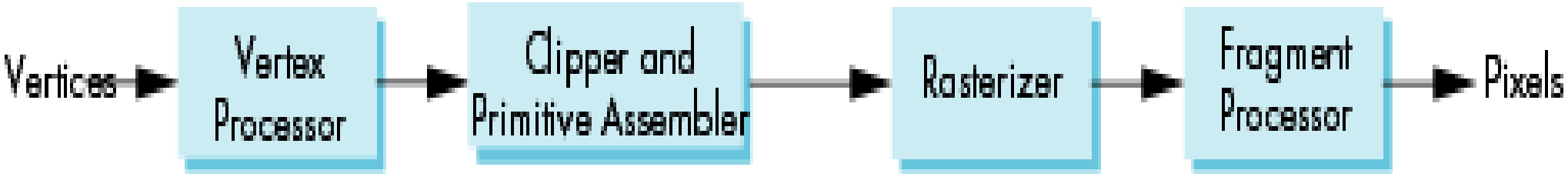
Rasterization



Primitive

- Point
- Line segments
- Polygons

fragments

- **Rasterizer produces a set of fragments for each object**
  - Fragments are "potential pixels"
    - Have a location in frame bufffer
    - Color and depth attributes

Scan conversion(扫描转换)/Rasterization（光栅化算法）

- **点的扫描转换**
- 直线的扫描转换算法
  - 直接计算
  - DDA
  - 中点Bresenham(中点算法)
  - 改进Bresenham(Bresenham算法)
- 圆等规则二次曲线的中点扫描转换算法

一、点的光栅化

OpenGL point functions

```
glBegin(GL_POINTS);
    glVertex3f(-78.05,   909.72,    14.60)
glEnd();
```

因为屏幕坐标只能为整数，实际屏幕窗口中位置为(-78, 910, 15）。

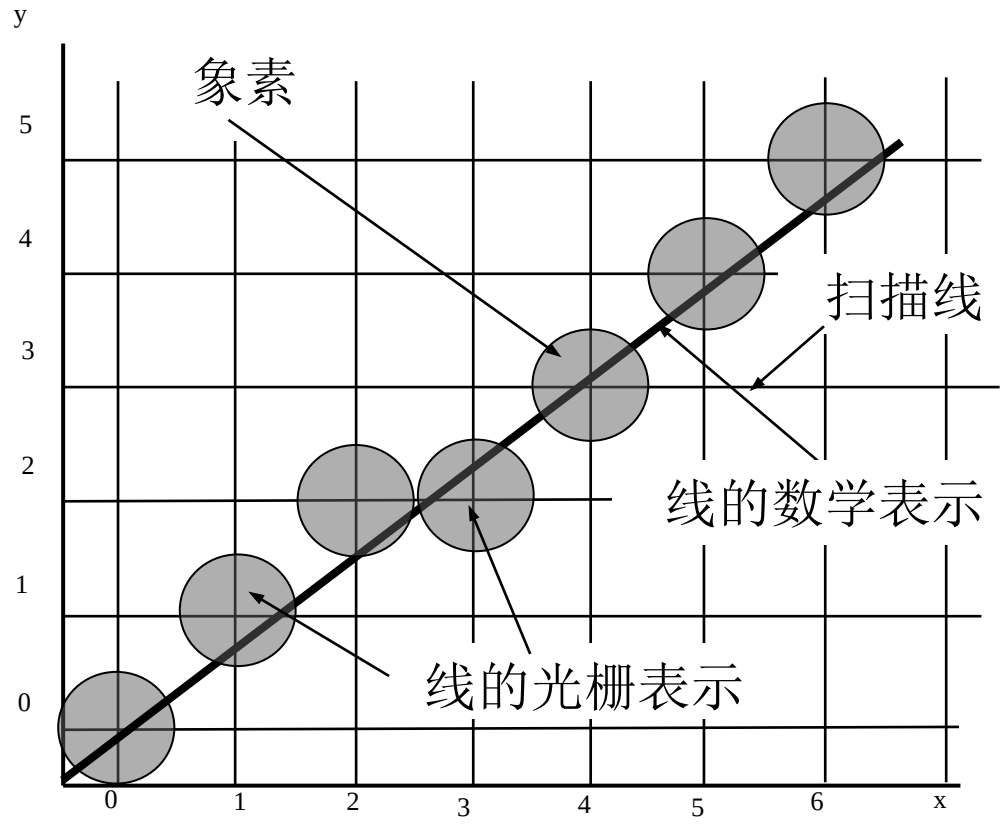
Scan conversion(扫描转换)/Rasterization（光栅化算法）

- **点的扫描转换**
- 直线的扫描转换算法
  - 直接计算
  - DDA
  - 中点Bresenham(中点算法)
  - 改进Bresenham(Bresenham算法)
- 圆等规则二次曲线的中点扫描转换算法

二、画线~线段光栅化

- **线图元生成**

在光栅显示器等数字设备上确定一个最佳逼近于图元象素集的过程。



OpenGL Line Functions

```
glBegin(GL_LINES)
    glVertext2iv(p1);
    glVertext2iv(p2);
glEnd();
```

思考：如何实现画线算法？

演示过程

1、线段光栅化-直接公式法

数学模型： $y=mx+b$

给定直线的两个端点(x_a,y_a)和(x_b,y_b)，

$m=(y_b-y_a)/(x_b-x_a)$ 和 $b=(y_ax_b-y_bx_a)/(x_b-x_a)$

算法思想：循环，x每次递增1，计算y

$X_{i+1}=X_i+1$；

$Y_{i+1}=int(m*X_{i+1}+b)$

```
/**直接带公式法--程序*、
x=xa;
m=(yb-ya)/(xb-xa);
b=ya-m*xa=(yaxb-ybxa)/(xb-xa);
While (x<xb)
   x=x+1; 单位递增
   y=mx+b; 计算y
  Setpixel(int(x), int(y), color);
End
```

优点：增量迭代计算X

缺点：计算量大（循环中有乘法操作）

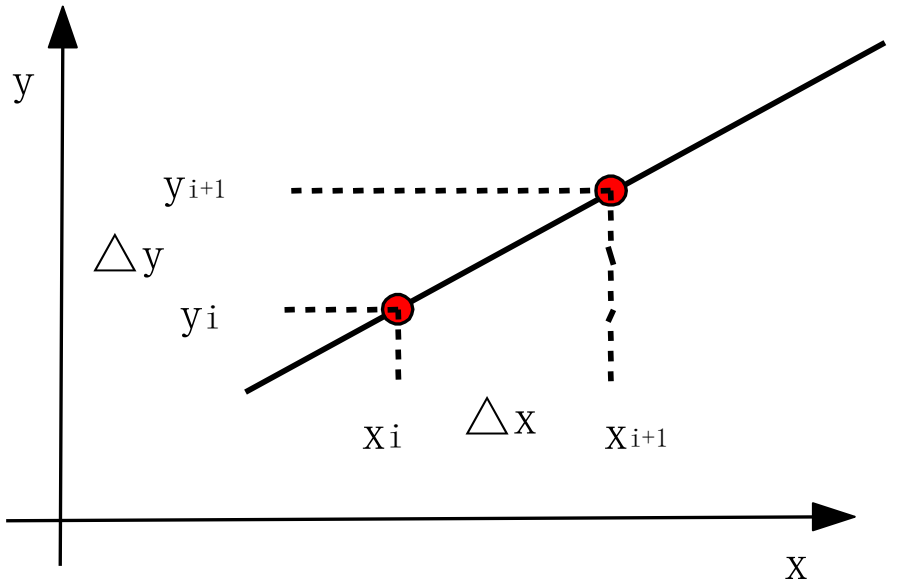
2.线段光栅化-DDA算法（Digital Differential Analyzer）

- **数学模型：　△y=m△x**

  即：$(Y_{i+1}-Y_i)= ((yb- ya)/(xb- xa))(X_{i+1}-X_i)$

  即：$(Y_{i+1}-Y_i)=m(X_{i+1}-X_i)=m\triangle x$

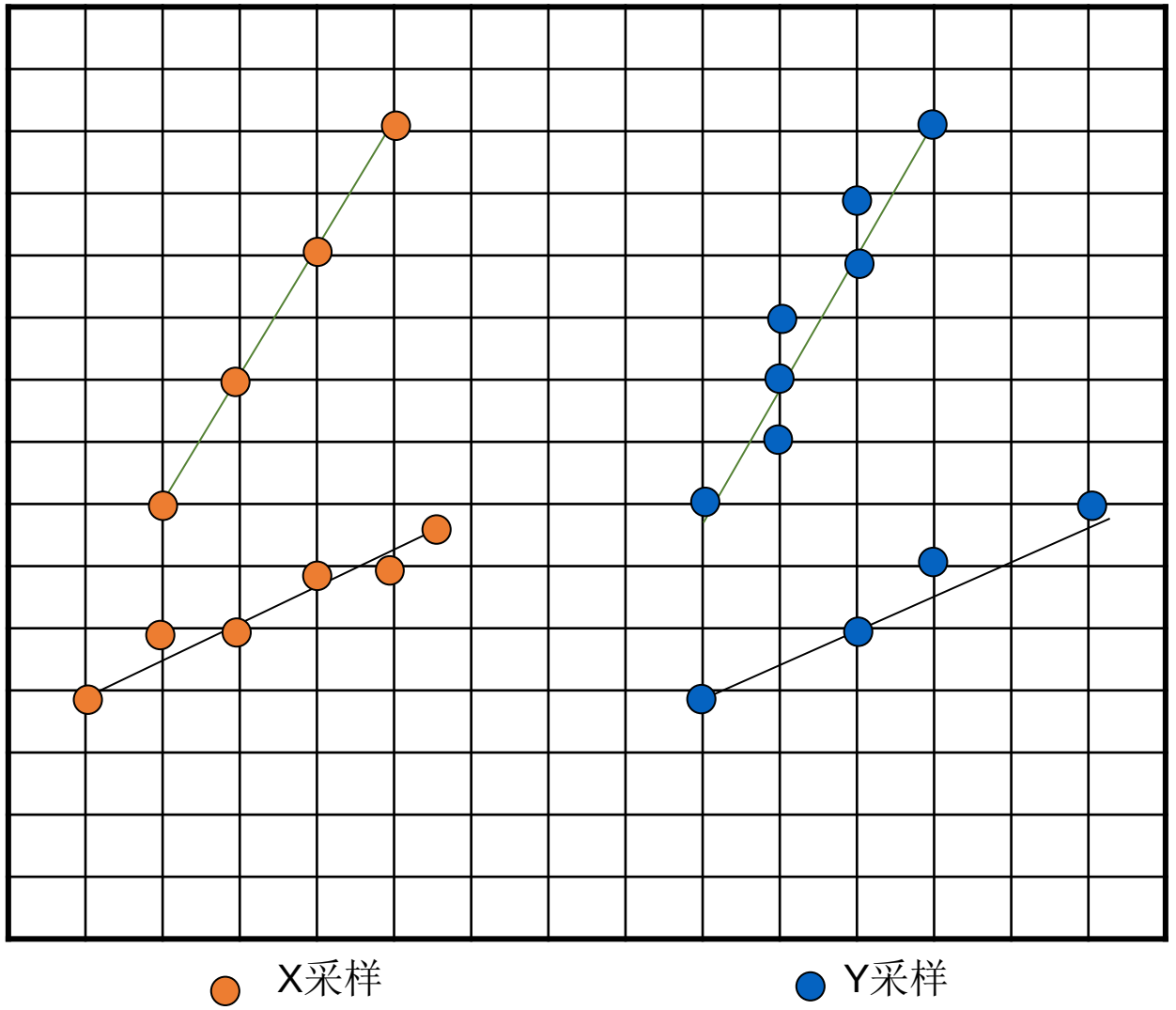
  $Y_{i+1}=Y_i+m\ \triangle x$



- ***算法思想：　当△x=1时，△y=y+m;***
  - $X_{i+1}=X_i+1$；
  - $Y_{i+1}=int(Y_i+m)$；y

优点：X,Y都是增量迭代计算。
缺点：　浮点加法（requires one floating point addition per step ）

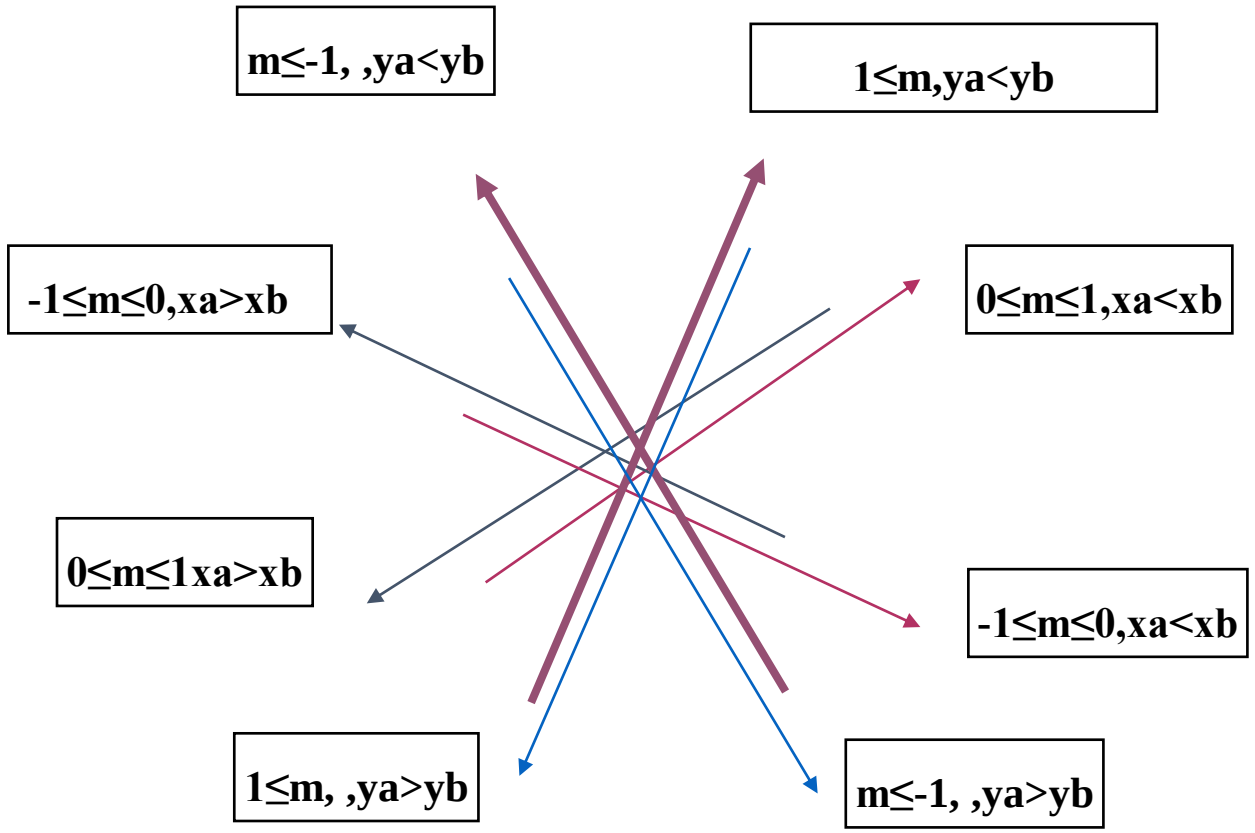
？选择x还是y单位步进1



|m|<1时，|Dx|>|Dy|,
X单位增长采样点更多

|m|>1时，|Dy|>|Dx|,
Y单位增长采样点更多

斜率（4种情况），坐标（2种情况），共8种情况讨论。



编程时，可合并简化斜率因素，分成四种情况

若|m|≤1，

$x_a<x_b$

$x_{k+1}=x_k+1, y_{k+1}=y_k+m$

$x_a>x_b$

$x_{k+1}=x_k-1, y_{k+1}=y_k-m$

若|m|>1，

$y_a<y_b$

$y_{k+1}=y_k+1, x_{k+1}=x_k+1/m$

$y_a>y_b$

$y_{k+1}=y_k-1, y_{k+1}=y_k-1/m$

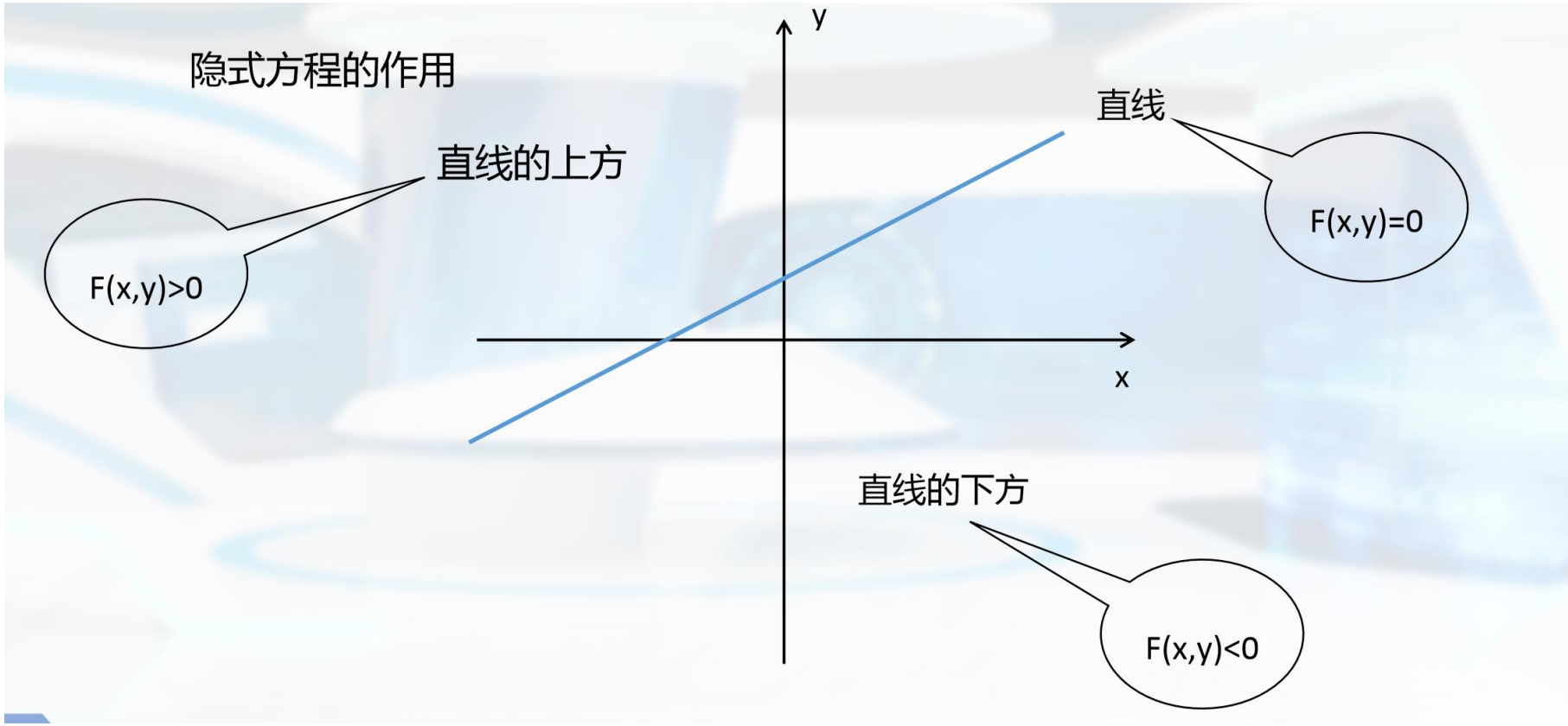
？是否还可以继续合并？

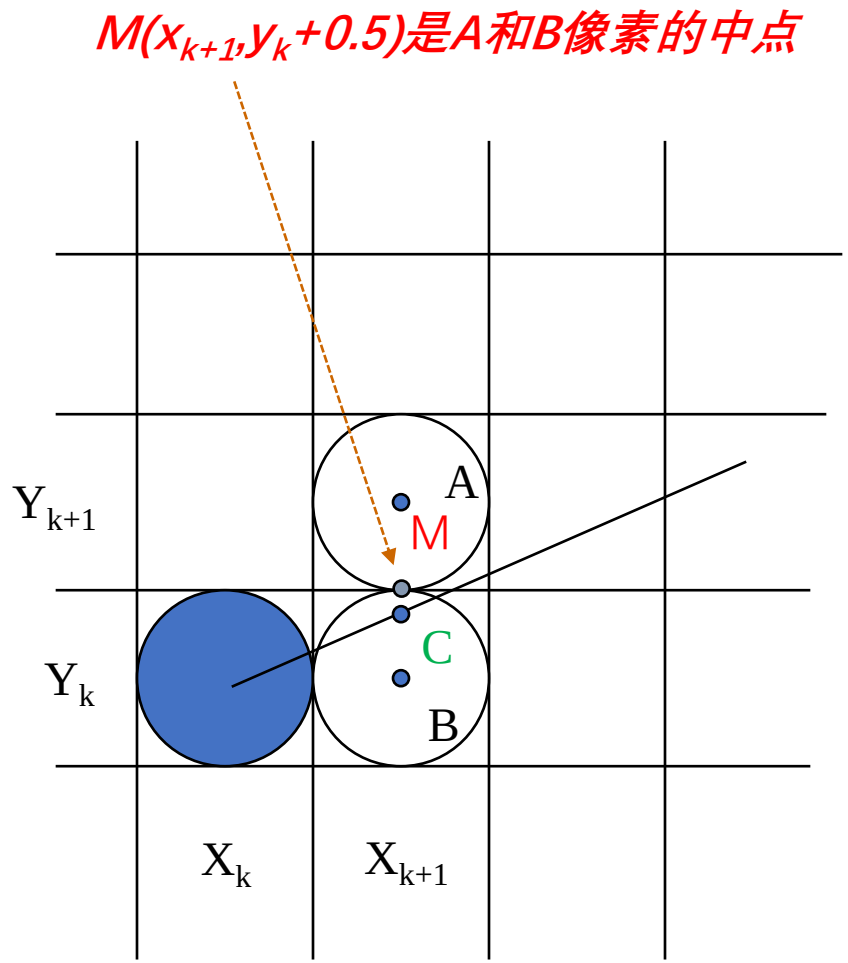
Scan conversion(扫描转换)/Rasterization（光栅化算法）

- 点的扫描转换
- 直线的扫描转换算法
  - 直接计算
  - DDA
  - **中点Bresenham(中点算法)**
  - 改进Bresenham(Bresenham算法)
- 圆等规则二次曲线的中点扫描转换算法

3、线段光栅化-中点算法

判定函数：F(x,y)=y-(mx+b)





C(x_{k+1},mx_{k+1}+b)是直线与$x=x_{k+1}$的交点。

考虑直线斜率0≤m≤1,且$x_a<x_b$

判别式：F(M)= $y_k+0.5$-(mx_{k+1} +b)

if (F(M)>0)

then 取点B(x_{k+1},y_k)

else 取A (x_{k+1},y_k+1).

1、构造判别式

$F(M_k)=F(M)= y_k+0.5-(mx_{k+1}+b)$

去掉小数0.5和负数，构造新判别式

$d_k=-2*F(M_k)$

$=-2*F(x_{k+1},y_k+0.5)$

$=-2*(y_k+0.5-(m*(x_k+1)+b))$

$=2m(x_k+1)+2b-2y_k-1$

$=2m(x_k+1)-2y_k+2b-1$

用d_k判定：

$d_k>0$， 取上面的点A, $(y_{k+1}=y_k+1)$

$d_k\leq 0$， 取下面的点B, $(y_{k+1}=y_k)$

2）判别式优化

$d_k=-2F(M_k)=2m(x_k+1)-2y_k+2b-1$

去掉m浮点计算采用新的判别式：

$p_k=\triangle x\, d_k=2\triangle yx_k-2\triangle xy_k+c$

($c=2\triangle y+\triangle x(2b-1)$为常数，　$\triangle y=yb-ya$)

$\triangle x=x_b-x_a>0$,

用p_k判定：

$p_k>0$， 取上面的点A$(y_{k+1}=y_k+1)$

$p_k\leq 0$，取下面的点B$(y_{k+1}=y_k)$

3）判别式的增量式推导

找出p_{k+1}与p_k的关系：

根据：$p_k=2\triangle yx_k-2\triangle xy_k+c$

所以：$p_{k+1}=2\triangle yx_{k+1}-2\triangle xy_{k+1}+c$

$=2\triangle yx_k+2\triangle y-2\triangle xy_{k+1}+c$

则：$p_{k+1}- p_k = 2\triangle y-2\triangle x(y_{k+1}-y_k)$

根据 $y_{k+1}-y_k$在p_k下得取值得到：

$p_{k+1}=p_k+2\triangle y-2\triangle x$　　($p_k> 0$)

$p_{k+1}=p_k+2\triangle y$　　($p_k< 0$)

4）求出判别式初始值

$p_k=2\triangle yx_k-2\triangle xy_k+c$

当k=0时，$b=y_a$,

$x_0=x_a$, $y_0=y_a-(\triangle y/\triangle x)*x_a$

$c=2\triangle y+\triangle x(2b-1)$为常数

$p_0 = 2\triangle yx_a-2\triangle xy_a+2\triangle y+\triangle x(2b-1)$

计算得到

$p_0 = 2\triangle y-\triangle x$

中点算法主要公式

- *适用于$0 \leqslant m \leqslant 1$且$x_a < x_b$情况：*

初始：

$\triangle y = y_b - y_a$； $\triangle x = x_b - x_a$；

$p_0 = 2\triangle y - \triangle x$； $x_0 = x_a$, $y_0 = y_b$

循环体中：

$x_{k+1} = x_k + 1$；

当$p_k \geq 0$：$y_{k+1} = y_k + 1$，$p_{k+1} = p_k + 2(\triangle y - \triangle x)$；

当$p_k < 0$：$y_{k+1} = y_k$，　$p_{k+1} = p_k + 2\triangle y$；

优点：

- *精确而有效的光栅线段生成算法，判定代替计算，增量迭代整数计算，效率高。*
- *不仅可用于直线、还可以用在圆和其它曲线的生成*

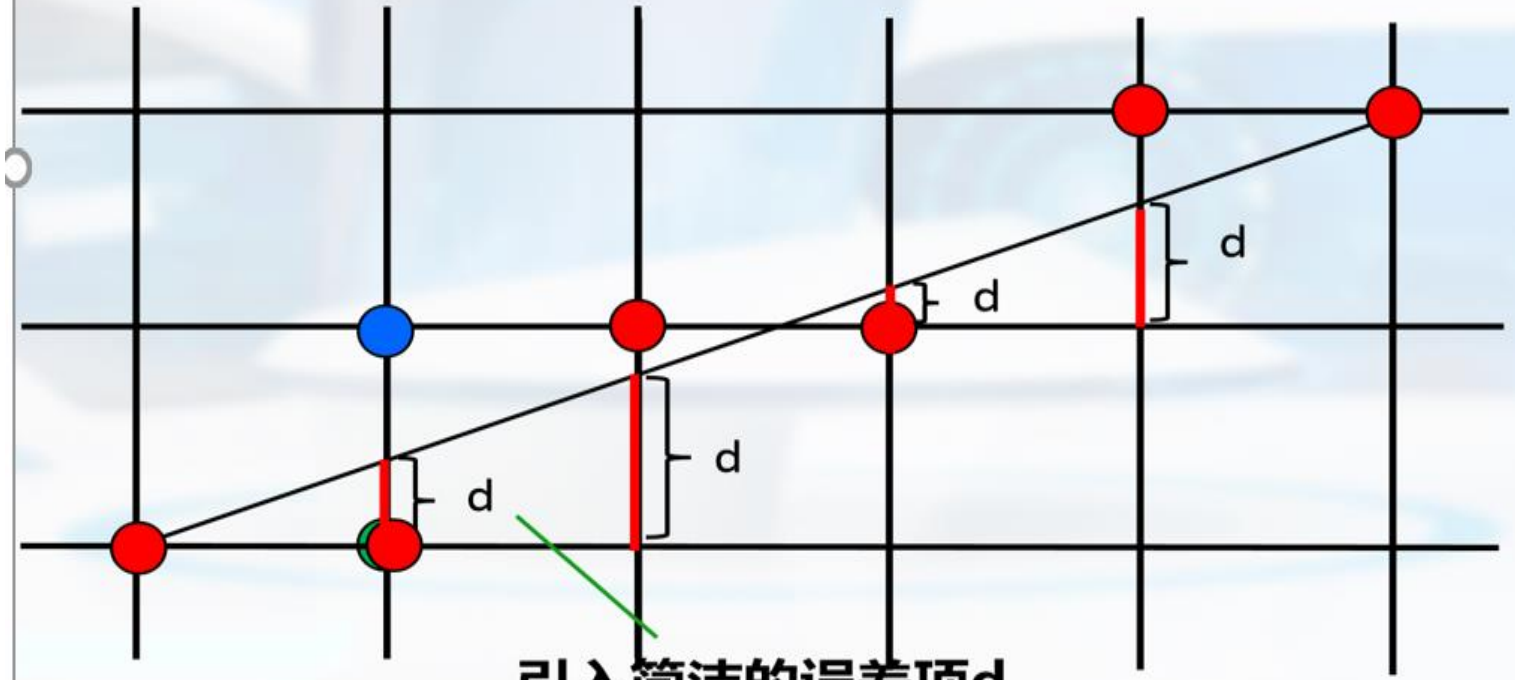
Scan conversion(扫描转换)/Rasterization（光栅化算法）

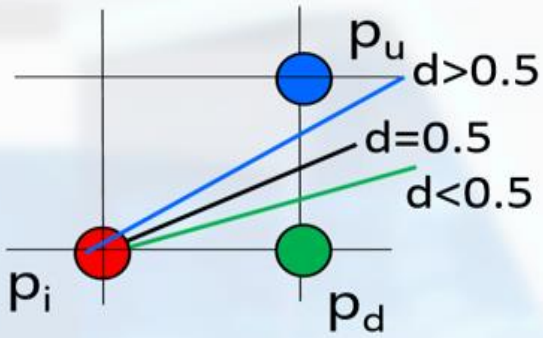
- 点的扫描转换
- 直线的扫描转换算法
  - 直接计算
  - DDA
  - **中点Bresenham(中点算法)**
  - 改进的Bresenham(Bresenham算法)
- 圆等规则二次曲线的中点扫描转换算法

更直观的想法

基本原理：

假定$0 \le k \le 1$，x是最大位移方向



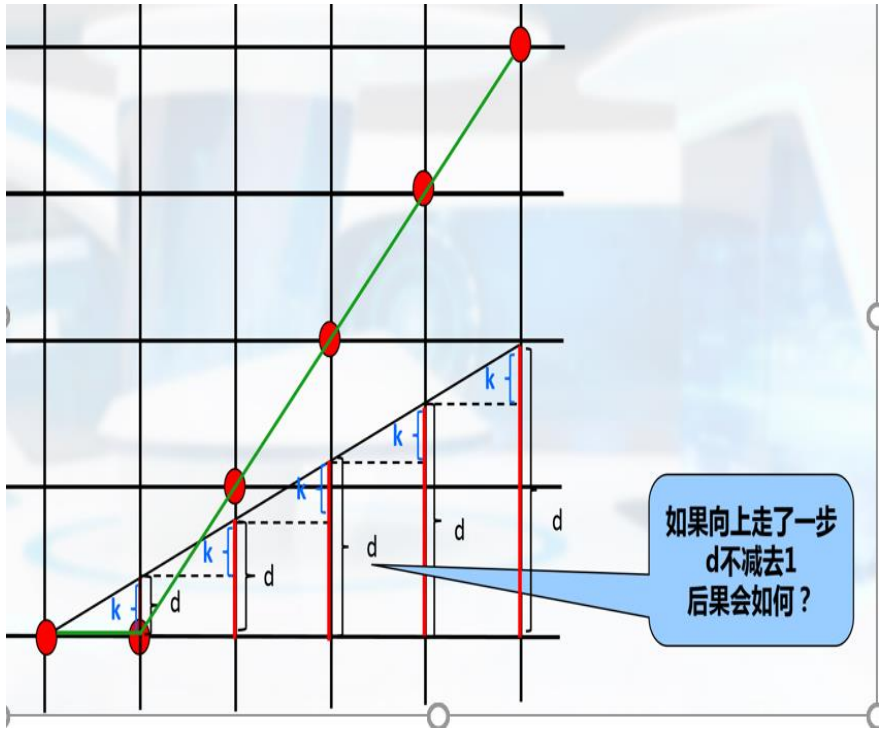


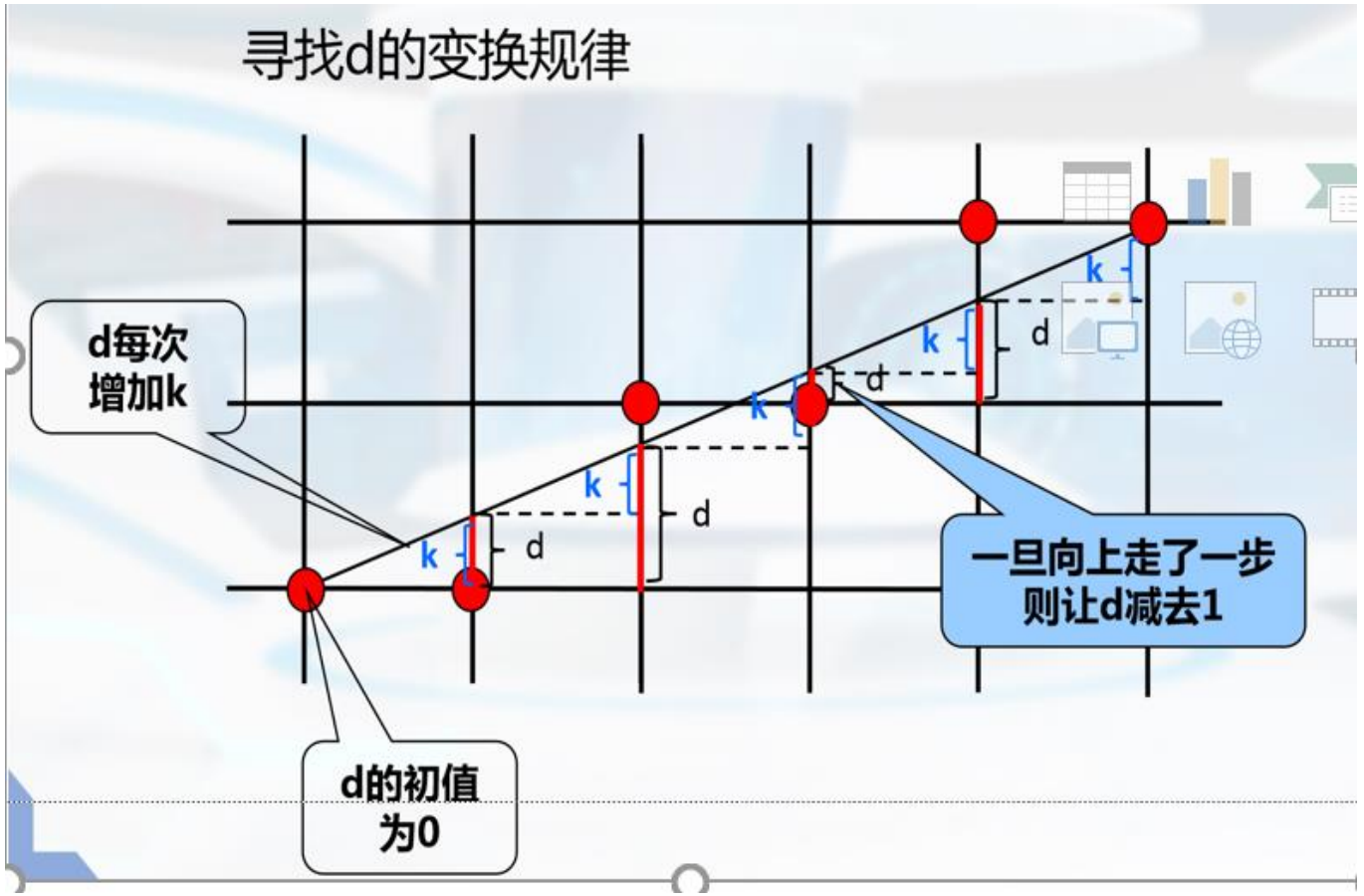
每次$x_{i+1}=x_i+1$

$$\begin{cases} d>0.5 & y_{i+1}=y_i+1 \\ d=0.5 & y_{i+1}=y_i \\ d<0.5 & y_{i+1}=y_i \end{cases}$$

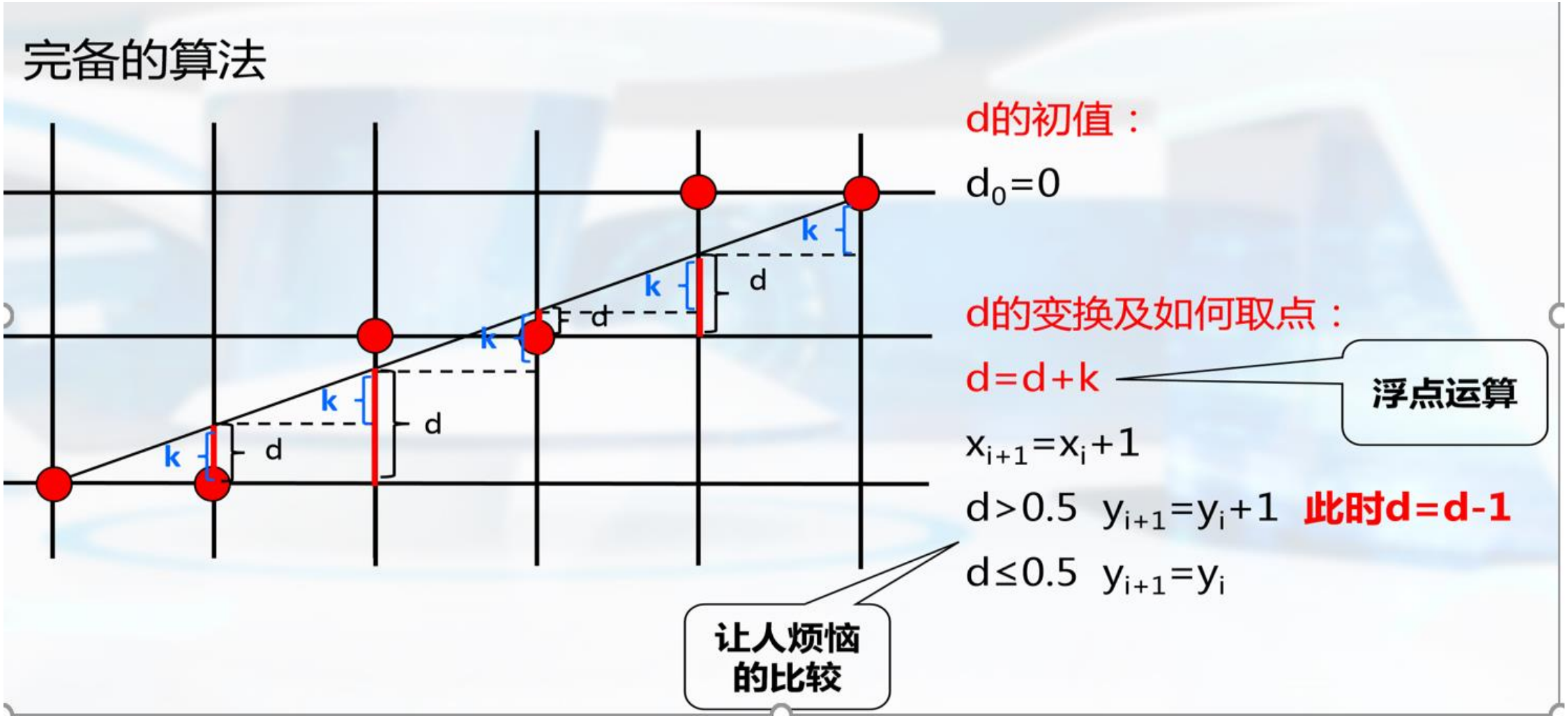
误差项d的变换规律

- 判别式d,斜率k,　判别式计算：$d_{i+1}=d_i+k$



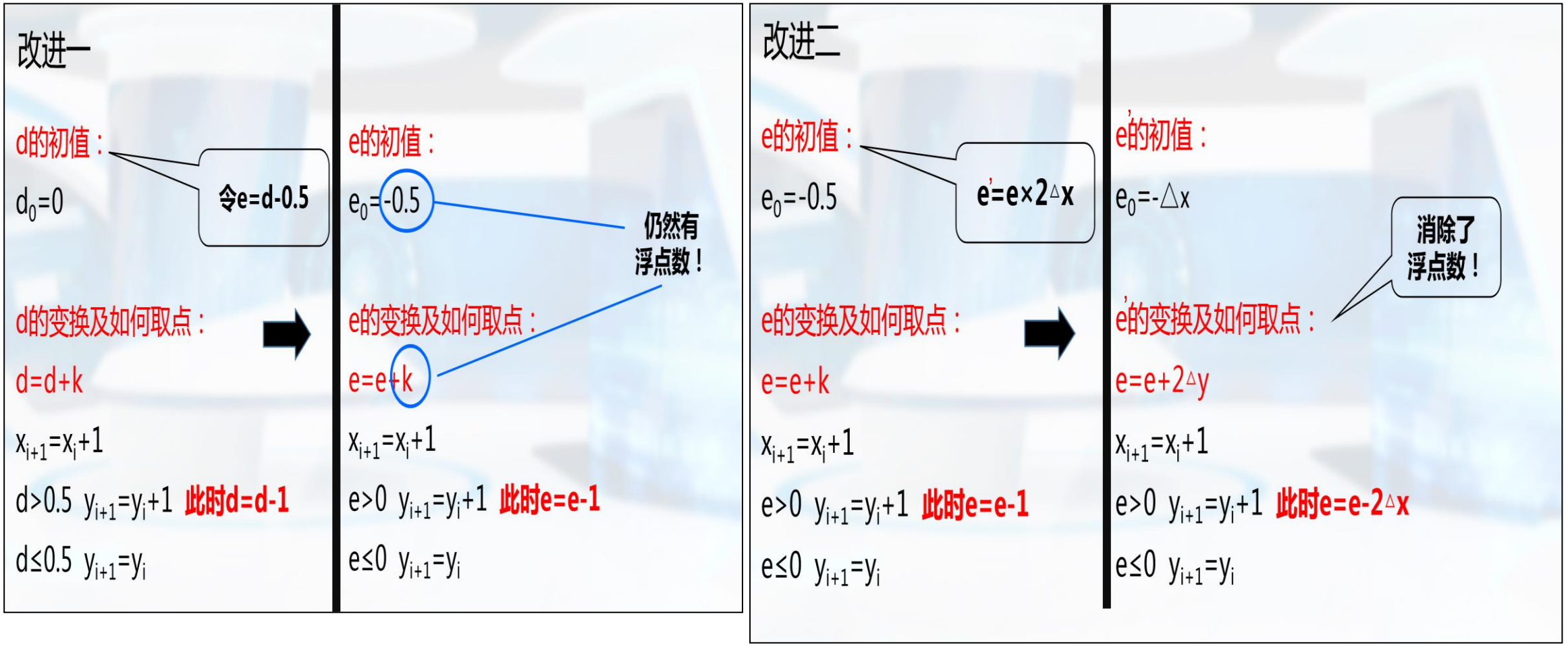


完备的Bresenham算法,但需要改进-浮点



算法改进：去掉浮点数计算

- e=d-0.5; e_0=-0.5
  - 因 d=e+0.5; 判别式增量式：e_{i+1}+0.5=e_i+0.5+k; 即 e_{i+1}=e_i+k.
- e'=e*2Δx; e_0'=-Δx;
  - 因 e=e'/2Δx; so 判别式增量式：e_{i+1}'/2Δx= e_i'/2Δx +k; 即e_{i+1}' = e_i' +2Δy



改进的bresenham算法

在0≤k≤1情况下改进的Bresenham算法：

（1）输入直线的两端点$P_0(x_0,y_0)$和$P_1(x_1,y_1)$。

（2）计算初始值$\triangle x$、$\triangle y$、$e=-\triangle x$、$x=x_0$、$y=y_0$。

（3）绘制点(x,y)。

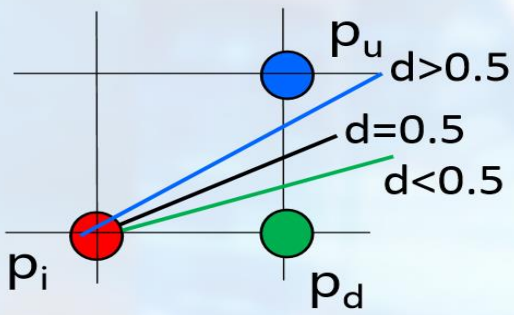
（4）e更新为$e+2\triangle y$

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$e-2\triangle x$；

否则(x,y)更新为(x+1,y)。

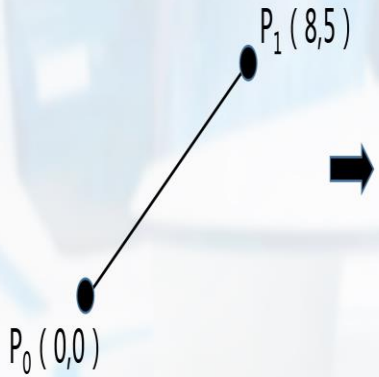
（5）当直线没有画完时，重复步骤3和4。否则结束。



Bresenham算法实例

输入：直线两个端点的坐标$P_0(0,0)$和$P_1(8,5)$

输出：最佳逼近这条直线的像素点集



$e_0=-\Delta x=-8$

e增量为$2\Delta y=2\times5=10$

如果e>0是y向上走一步

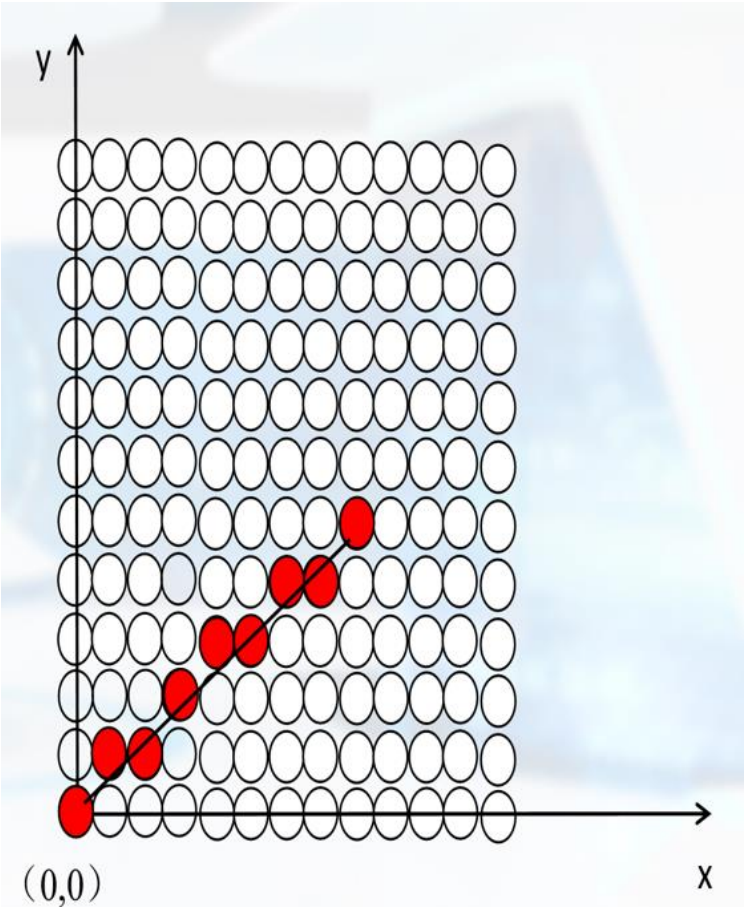
$e=e-2\Delta x$

其中$-2\Delta x=-16$

起点：$P_0(x_0,y_0)$为(0,0)

终点：$P_1(x_1,y_1)$为(8,5)

x	y	e	e+2△y
0	0	-8	2
1	1	-14	-4
2	1	-4	6
3	2	-10	0
4	2	0	10
5	3	-6	4
6	4	-12	-2
7	4	-2	8
8	5	-8	2



完整的直线扫描转换算法编写

- 1）可先处理特殊情况
  - 水平线（y2-y1=0)，直接转换
  - 垂直线（x2-x1=0)，直接转换
  - 对角线（y2-y1=x2-x1），直接转换

- 2)再写出其它一般情况的计算方法
  - 算法一般采用中点法或Bresenham算法进行推导
  - 同时还需要考虑斜率和线段方向，推导各种情况下的计算公式

Scan conversion(扫描转换)/Rasterization（光栅化算法）

- **点的扫描转换**
- 直线的扫描转换算法
  - 直接计算
  - DDA
  - **中点Bresenham(中点算法）**
  - **改进Bresenham(Bresenham算法）**

- 圆（规则二次曲线）的中点算法

5、圆的扫描转换算法 *圆的扫描转换演示*

圆心在任意点（a,b），

半径为整数R的圆

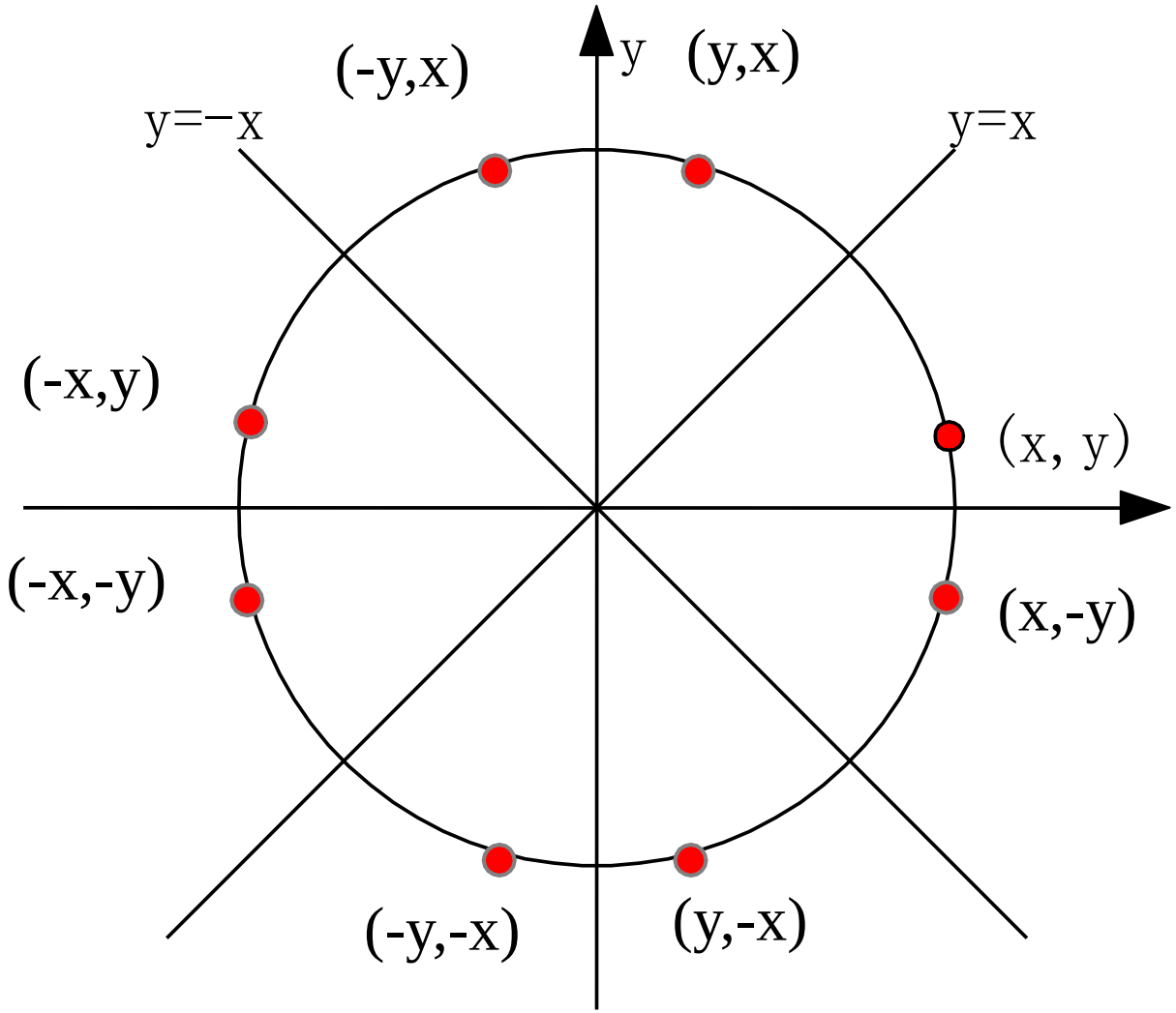
$(x-a)^2+(y-b)^2=R^2$

简化计算：

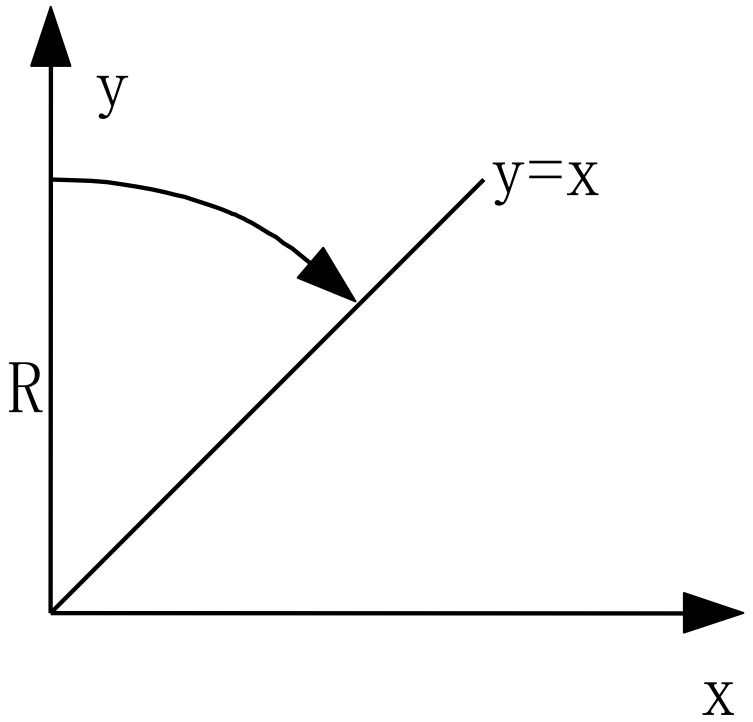
- 圆心在原点，半径为整数R的圆讨论：

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

- 利用圆的对称性，只推导八分之一圆的转换算法



问题：选择哪一个1/8圆进行讨论？



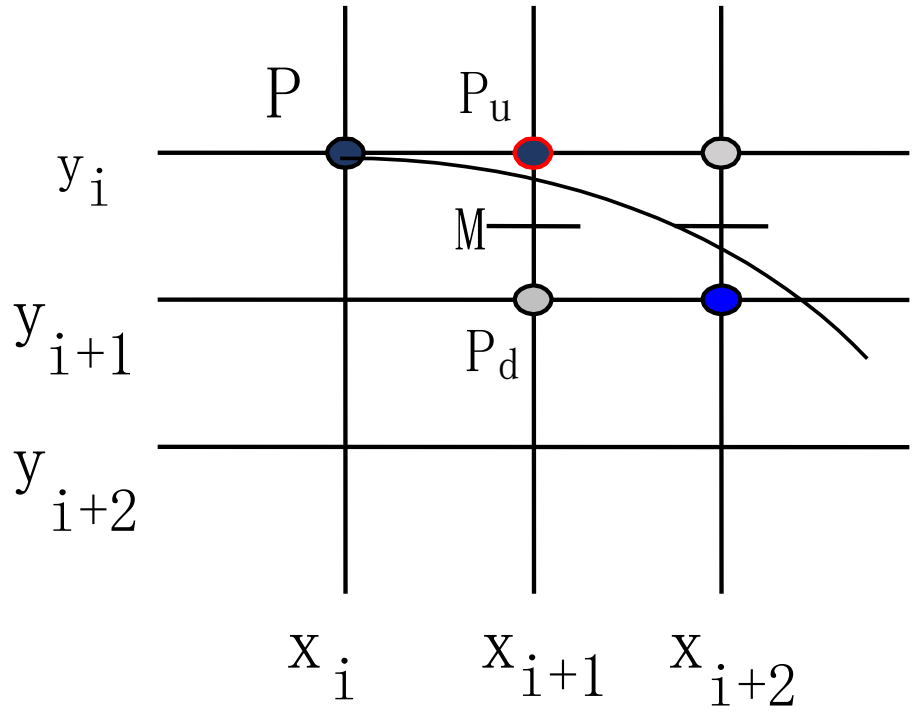
选择原因：Dx>Dy, x采样递减1，判定y的取值

中点画圆法

构造判别函数

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

- 对于圆上的点，　F(x,y)=0;
- 对于圆外的点，　F(x,y)>0;
- 而对于圆内的点，F(x,y)<0。



1.决策参数

$$di = F(x_M, y_M) = F(x_i+1, y_i - 0.5) = (x_i+1)^2 + (y_i - 0.5)^2 - R^2$$

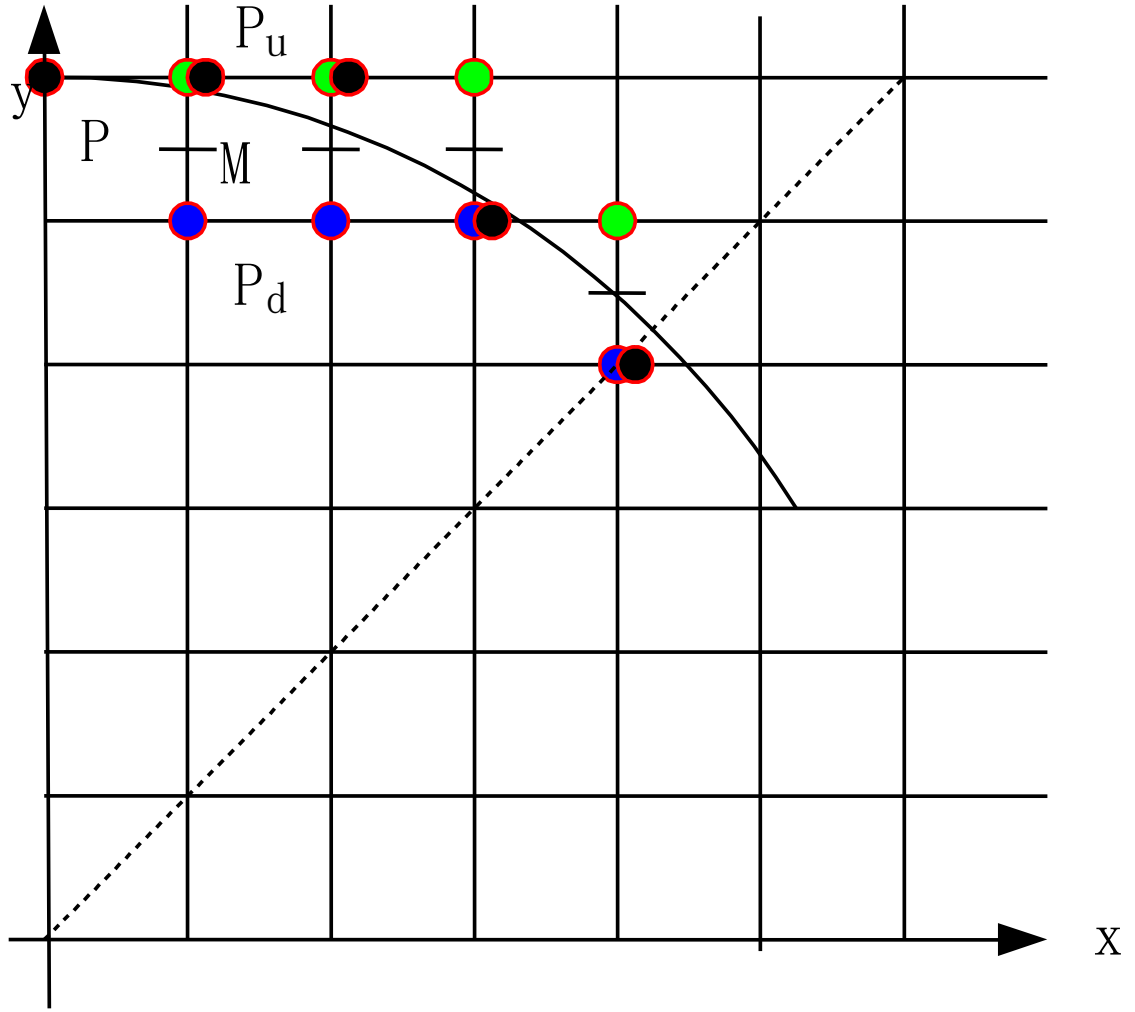
M的坐标为：$M(x_i+1, y_i-0.5)$

- 当$F(x_M,y_M)<0$时，

取$P_u(x_i+1, y_i)$，中点在圆内

- 当$F(x_M,y_M)>=0$时，

取$P_d(x_i+1, y_i-1)$，中点在圆外

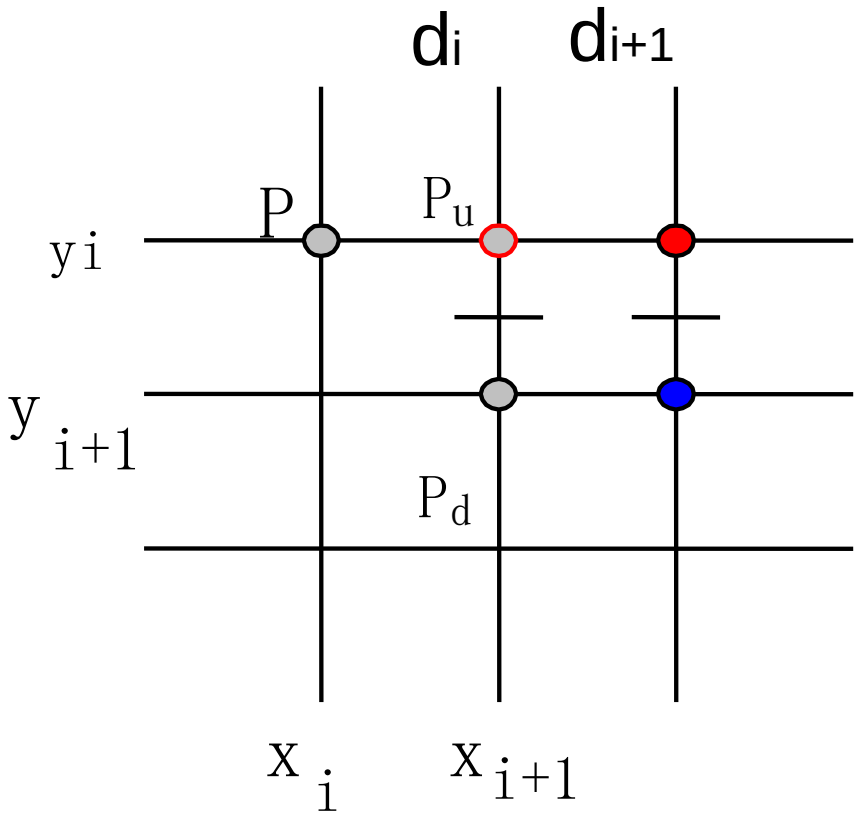


2、决策参数的增量计算

(a) di<=0时

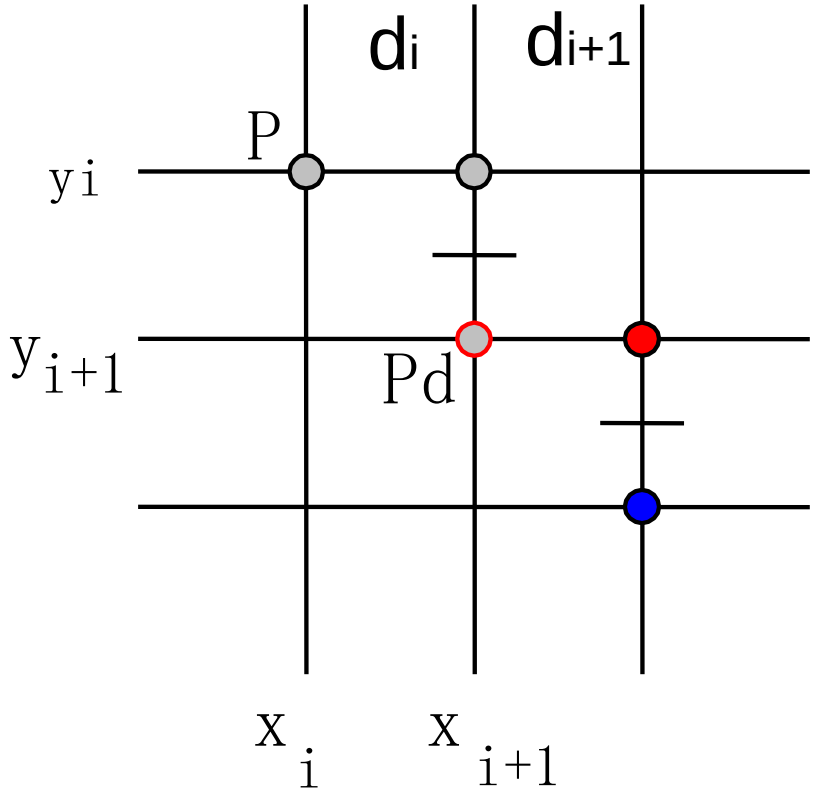
$y_{i+1}=y_i$; //即下一点取Pu(x_i+1, y_i)

$$d_{i+1} = F(x_i+2, y_i-0.5)$$
$$= (x_i+2)^2 + (y_i-0.5)^2 - R^2$$
$$= (x_i+1)^2 + (y_i-0.5)^2 - R^2 + 2x_i + 3$$
$$= di + 2x_i + 3$$



(b) di>0时

$y_{i+1}=y_i-1$;　//即取下面的点Pd



$$d_{i+1}=F(x_i+2,y_i-1.5)$$

$$=(x_i+2)^2+(y_i-1.5)^2-R^2$$

$$=(x_i+1)^2+(y_i-0.5)^2-R^2+(2x_i+3)+(-2y_i+2)$$

$$=d_i+2(x_i-y_i)+5$$

3、判别式的初始值

$$
\begin{aligned}
d_0 &= F(1, R-0.5) \\
&= 1+(R-0.5)^2 - R^2 \\
&= 1.25 - R
\end{aligned}
$$

圆的扫描转换算法

算法步骤:

1.输入圆的半径R。

2.计算初始值d=1.25-R、x=0、y=R。

3.绘制点(x,y)及其在八分圆中的另外七个对称点。

4.判断d的符号？若d≤0，则先将d更新为d+2x+3，再将(x,y)更新为(x+1,y)；否则，先将d更新为d+2(x-y)+5，再将(x,y)更新为(x+1,y-1)。

5.当x<y时，重复步骤3和4。否则结束。

//程序(略)

改进：用d-0.25代替d

算法步骤:

1.输入圆的半径R。

2.计算初始值**d=1-R**、x=0、y=R。

3.绘制点(x,y)及其在八分圆中的另外七个对称点。

4.判断d的符号。若**d≤0**，则先将d更新为d+2x+3，再将(x,y)更新为(x+1,y)；否则先将d更新为d+2(x-y)+5，再将(x,y)更新为(x+1,y-1)。

5.当x<y时，重复步骤3和4。否则结束。

//程序(略)