GIS专业主干课 ：21905001

计算机图形学
Computer Graphics

林伟华

中国地质大学（武汉）信息工程学院

lwhcug@163.com

目录

- 线段绘制方法
- DDA算法
- 中点画线算法
- Bresenham算法

线段绘制方法

- VC++
  - CDC::MoveTo(int x, int y);
  - CDC::LineTo(int x, int y);
- OpenGL
  - glBegin(GL_LINES);
  - glVertex2iv (point1);
  - glVertex2iv (point2);
  - glEnd();
- MapGIS
  - _Appendline(short ai, D_DOT *pxy);
- 绘点方式
  - CDC::SetPixel(POINT point,COLORREFcrColor); **//vc++**
  - setPixel(int x, int y) ; **//OpenGL**

直线扫描转换

•光栅扫描显示下画直线

- **显示速度：**

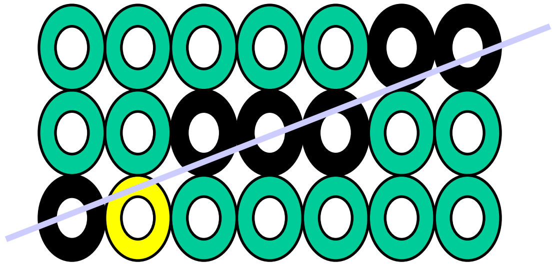
例：　分辨率：1024×768， 24Bit 彩色，

帧存容量：1024×768×3 = 2,359,296 Byte

刷新率 85Hz: 85× 2,359,296 = 200,540,160 (Byte / S)

存储器读出时间：~5nS

- **显示质量：**
  - 阶梯状
  - 线的粗细不一
  - 线的亮度差异



解决的问题：

给定直线两端点$P_0(x_0,y_0)$和$P_1(x_1,y_1)$，画出该直线。

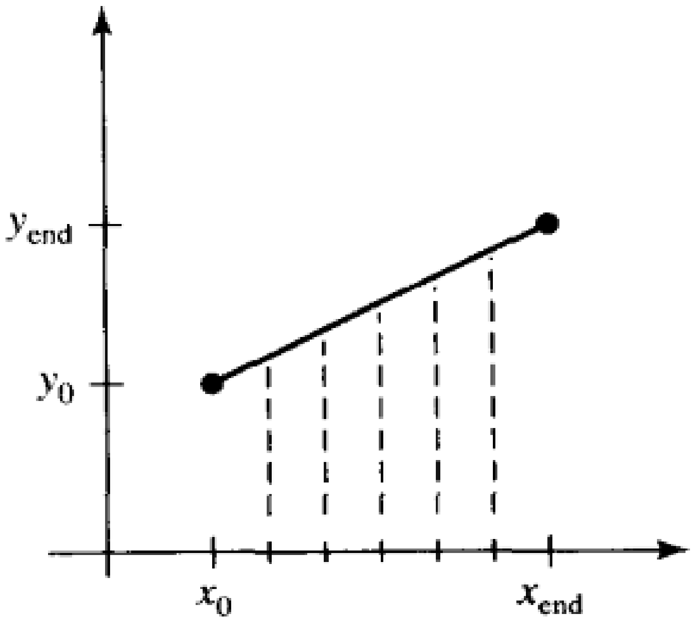
直线方程

- **斜率截距方程**
  - y = mx + b
  - 直线用两点坐标(x_0, y_0) 和 (x_{end}, y_{end})表示，则：

$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

  - x 增量dx，y 增量dy 计算方式：
    dy = m (dx)
  - 同样可得到：

$$dx = \frac{dy}{m}$$



数值微分法（DDA）

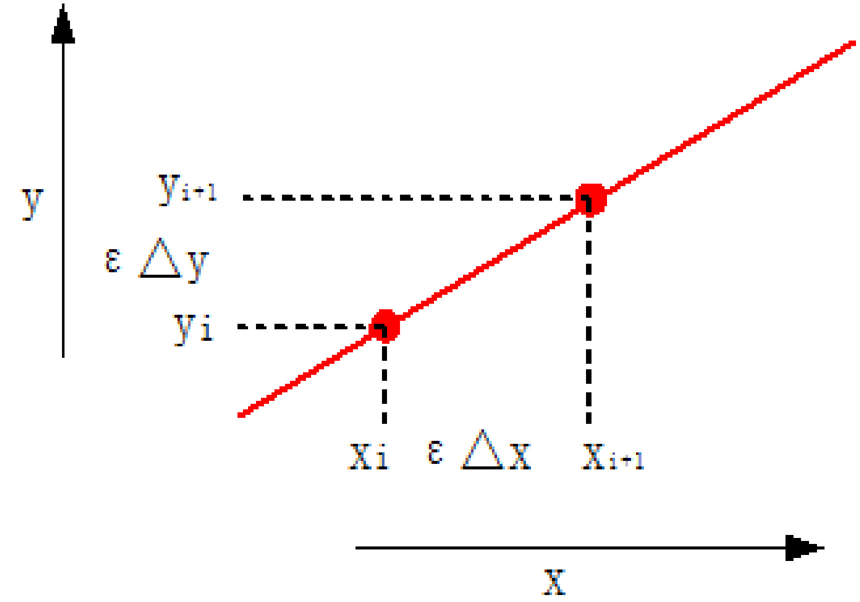
- **DDA算法原理**

直线的微分方程：

$$x_{i+1} = x_i + \varepsilon \cdot \Delta x$$

$$y_{i+1} = y_i + \varepsilon \cdot \Delta y$$

ε=1/max(|△x|,|△y|)



数值微分法（DDA）

max(|△x|,|△y|)=|△x|，即|k|≤1的情况：

$$x_{i+1} = x_i + \varepsilon \cdot \Delta x = x_i + \frac{1}{|\Delta x|} \cdot \Delta x = x_i \pm 1$$

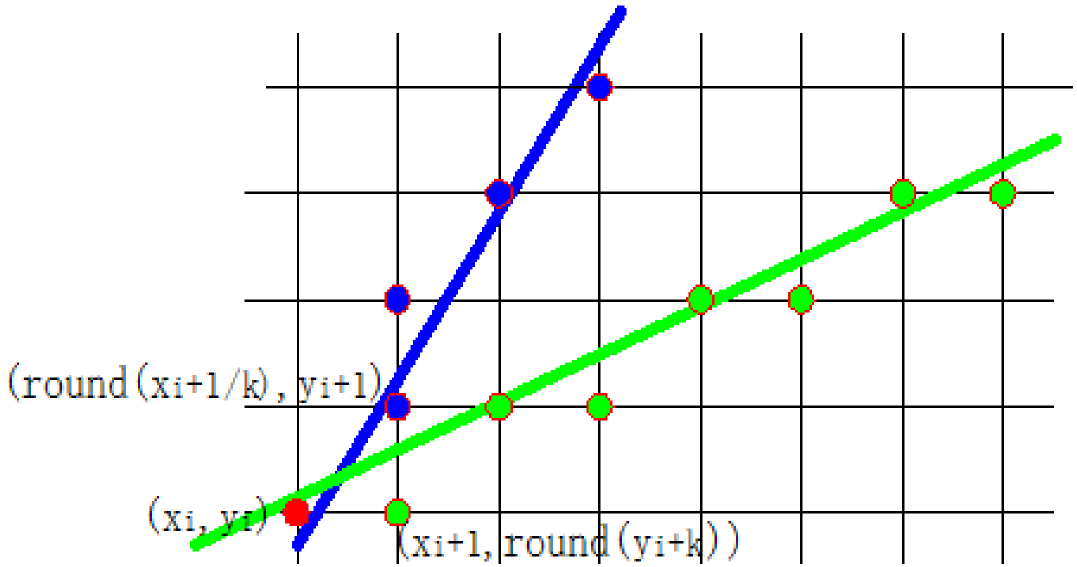
$$y_{i+1} = y_i + \varepsilon \cdot \Delta y = y_i + \frac{1}{|\Delta x|} \cdot \Delta y = y_i \pm k$$

max(|△x|,|△y|)=|△y|，此时|k|≥1：

$$x_{i+1} = x_i + \varepsilon \cdot \Delta x = x_i + \frac{1}{|\Delta y|} \cdot \Delta x = x_i \pm \frac{1}{k}$$

$$y_{i+1} = y_i + \varepsilon \cdot \Delta y = y_i + \frac{1}{|\Delta y|} \cdot \Delta y = y_i \pm 1$$

数值微分法（DDA）



- **注意：** round(x)=(int)(x+0.5)

- **特点：** 增量算法，直观、易实现
- **缺点：**
  - 浮点运算、取整---废时，且不利于硬件实现。
  - 不利于用硬件实现。

数值微分法（DDA）

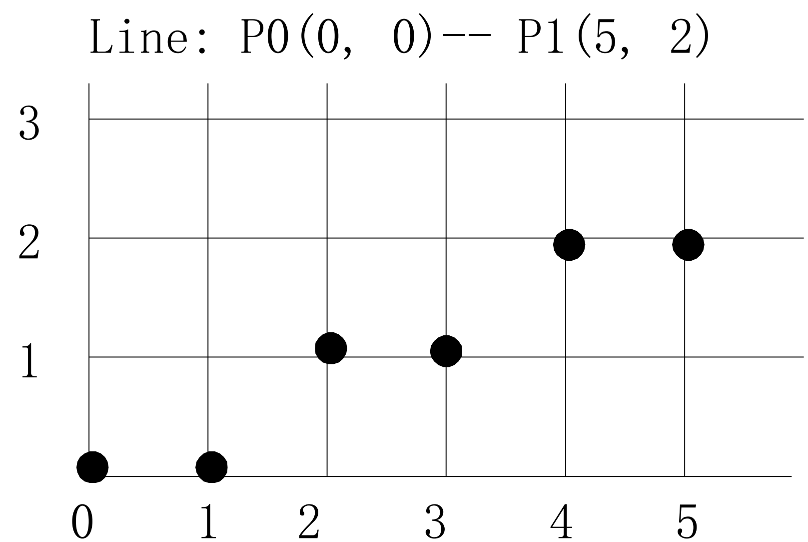
- void DDAline(int x0,int y0,int x1,int y1)
- {
-   int   dx,dy,eps1,k;
-   float x,y,xIncre,yIncre;
-   dx=x1-x0; dy=y1-y0;
-   x=x0;   y=y0;
-   If (abs(dx)>abs(dy)) eps1=abs(dx);
-   else eps1=abs(dy);
-   xIncre=(float)dy/(float)eps1;
-   yIncre=(float)dy/(float)eps1;
-   for (k=0;k<=eps1;k++) {
-    SetPixel((int)(x+0.5),(int)(y+0.5));
-    x+=xIncre;
-    y+=yIncre;
-    }
- }

数值微分法（DDA）

$$P_0(0,0)-P_1(5,2)$$

例：画直线段

x	int(y+0.5)	y + 0.5
0	0	0
1	0	0.4+0.5
2	1	0.8+0.5
3	1	1.2+0.5
4	2	1.6+0.5
5	2	2.0+0.5



注：网格点表示象素

中点画线算法

- **原理：**
- 假定直线斜率K<1，且已确定点亮象素点P（X_p ，Y_p ），M为中点，Q为交，现需确定下一个点亮的象素。

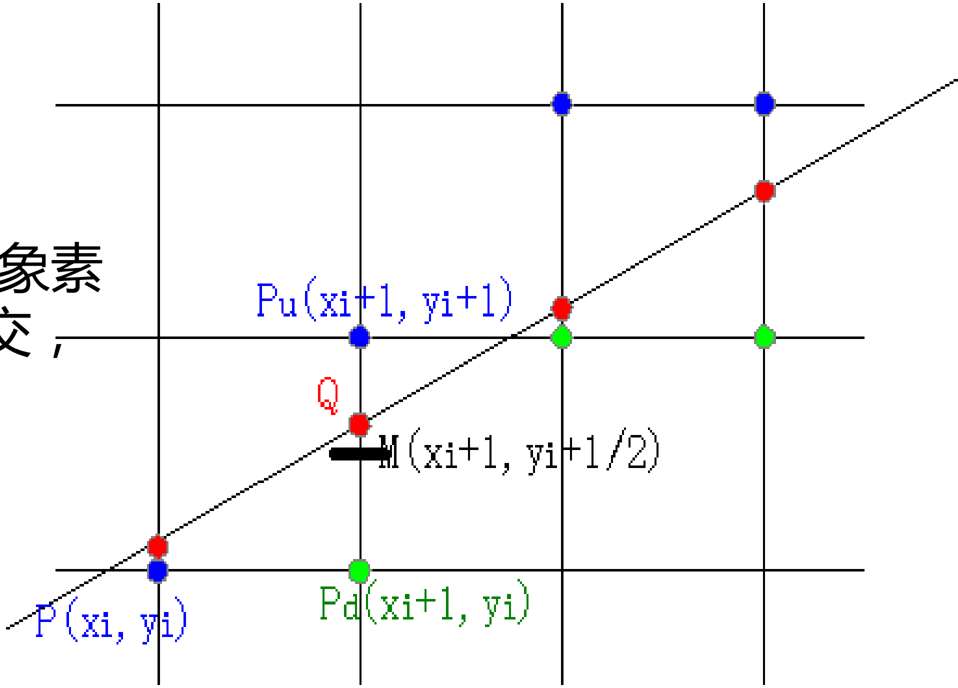


图5-5 Brensemham算法生成直线的原理

显然可得出如下结论：若M 在Q的下方，选Pu，否则选Pd

中点画线算法

- **算法实现：**
- 假设直线的起点、终点分别为：（X0,Y0),（X1,Y1)
- 该直线方程可表示为：

F(x,y)=a*x+b*y+c

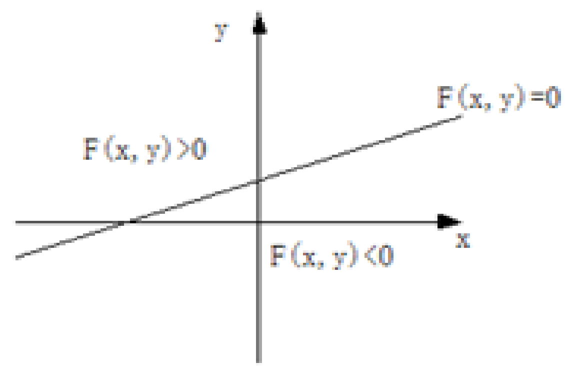
其中： a=Y0-Y1, b=X1-X0,

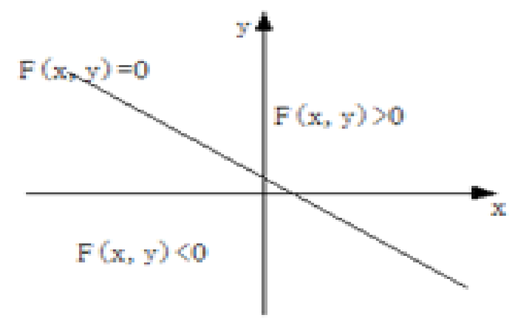
c=X0*Y1-X1*Y0

当：F(Xt,Yt) = 0 → (Xt,Yt) 在直线上

F(Xt,Yt) < 0 → (Xt,Yt) 在直线下方

F(Xt,Yt) > 0 → (Xt,Yt) 在直线上方





中点画线算法

- 设x_k处中点为P_k，则：

$P_k=F(x_k+1,y_k+1/2)$

$\quad =a(x_k+1)+b(y_k+1/2)+c$

$P_{k+1}=F(x_{k+1}+1,y_{k+1}+1/2)$

$\quad =a(x_{k+1}+1)+b(y_{k+1}+1/2)+c$

$P_{k+1}-P_k= a(x_{k+1}+1)+b(y_{k+1}+1/2)+c$

$\quad - a(x_k+1)-b(y_k+1/2)-c$

$\mathbf{P_{k+1}=P_k+a+b(y_{k+1}-y_k)}$ $\mathbf{P_0 = F(x_0+1,y_0+1/2) = F(x_0,y_0)+a+1/2*b}$

($\mathbf{P_0=2*a+b}$)

($\mathbf{P_{k+1}=P_k+2a+2b(y_{k+1}-y_k)}$)

因只用P 的符号作判断，为了只包含整数运算, 可取2P代替 P。

$x_{k+1} = x_k+1$

若$P_{k+1}\geq 0$ 取$y_{k+1} = y_k$， $P_{k+1}=P_k+2*a$

若$P_k<0$ 取$y_{k+1} = y_k+1$, $P_{k+1}=P_k+2*(a+b)$

中点画线算法

```
void MPLine(int x1, int y1, int x2, int y2, int color)
{  int x, y, a, b, d, d1, d2;    /*  0≤m≤1  */
     a=y1-y2;  b=x2-x1;
     y=y1;
     p=2*a+b;  d1=2*a;  d2=2*(a+b);
     setpixel(x,y,color);
     for(x=x1; x<=x2; x++)
     {  if(p<0)
          {  y++;  p+=d2;  }
          else  p+=d1;
          setpixel(x, y, color);
      }
  }
```

中点画线算法

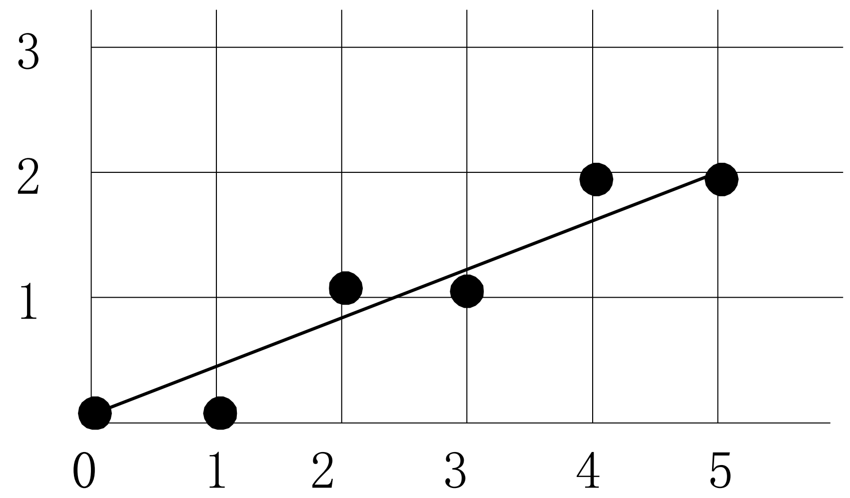
$P_0(0,0) - P_1(5,2)$

例：用中点画线法

$a = y_0 - y_1 = -2,\ b = x_1 - x_0 = 5$

$p_0 = 2a + b = 1,\ d_1 = 2a = -4,\ d_2 = 2(a + b) = 6$

I	xi	yi	pi
1	0	0	1
2	1	0	-3 (p+d1)
3	2	1	3 (p+d2)
4	3	1	-1
5	4	2	5



Bresenham算法

- **基本原理：(假定直线段的0≤k≤1)**

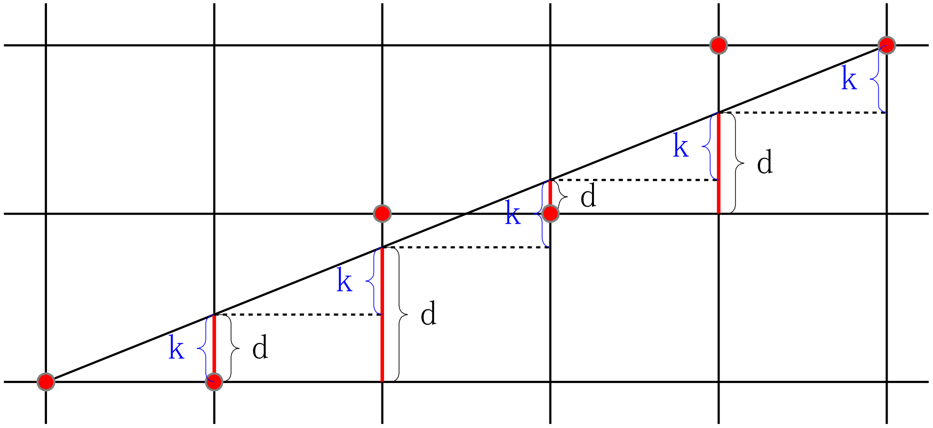


图5-8 改进的Brensemham算法绘制直线的原理

- 假定直线斜率,0<k<1, 起点坐标为(x,y);
- 下一个点亮象素可能是：(x+1,y) 或(x+1,y+1)。这可通过d值的大小来确定:
- d = d+k, 当d>1 时 d=d-1；
- 当d<0.5取 (x+1,y)，否则取(x+1,y+1)。

$$\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = \begin{cases} y_i + 1 & (\mathrm{d} > 0.5) \\ y_i & (\mathrm{d} \le 0.5) \end{cases} \end{cases}$$

Bresenham算法

- **算法步骤：**
- 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△x、△y、d=0、x=x_0、y=y_0。
- 3.绘制点(x,y)。
- 4.d更新为d+k，判断d的符号。若d>0.5，则(x,y)更新为(x+1,y+1)，同时将d更新为d-1；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。

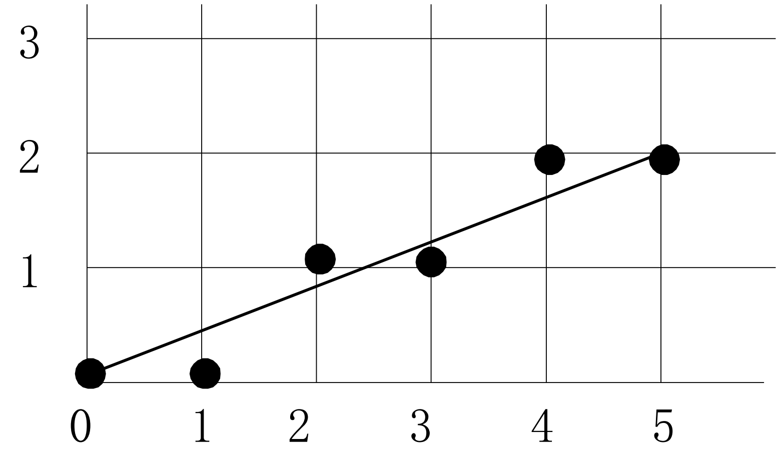
Bresenham算法

示例

例：Line: $P_0(0, 0)$, $P_1(5,2)$ k=dy/dx=0.4

x	y	e
0	0	-0.5
1	0	-0.1
2	1	0.3
3	1	-0.3
4	2	0.1
5	2	-0.5

$$e = e + k$$



大于零，y加1，小于零，不变

Bresenham算法

- **优化1**：令e=d-0.5
  - e初= -0.5，
  - 每走一步有e=e+k。
  - if (e>0) then e=e-1

$$\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = \begin{cases} y_i + 1 & (\mathrm{e} > 0) \\ y_i & (\mathrm{e} \le 0) \end{cases} \end{cases}$$

- **优化2**：用2e△x来替换e
  - e初= -△x，
  - 每走一步有e=e+2△y。
  - if (e>0) then e=e-2△x

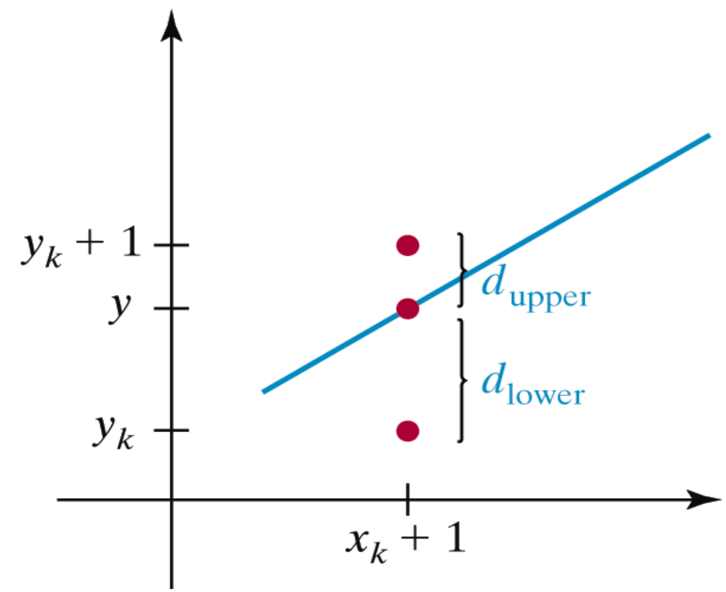
Bresenham算法

- **算法步骤：**
- 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Δx、Δy、$e=-\Delta x$、$x=x_0$、$y=y_0$。
- 3. 绘制点(x,y)。
- 4. e更新为$e+2\Delta y$，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$e-2\Delta x$；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

- **决策参数推导（优化2）：**

- $y=m(x_k+1)+b$，令$c = 2\Delta y + (2b-1)$
- $d_{lower}=y-y_k=m(x_k+1)+b-y_k$
- $d_{upper}=(y_k+1)-y=y_k+1-m(x_k+1)-b$
- $P_k= \Delta x(d_{lower}-d_{upper})$　（$\Delta x>0$）
- $\quad =2\Delta y \cdot x_k-2\Delta x \cdot y_k+c$
- $\mathbf{P_{k+1} = P_k+2\Delta y-2\Delta x(y_{k+1}-y_k)}$
- $\mathbf{P_0=2\Delta y-\Delta x}$
- $x_{k+1}=x_k+1$
- 若$P_k\geq 0$ 则 $y_{k+1}=y_k+1$　$P_{k+1}=P_k+2(\Delta y-\Delta x)$
- 若$P_k<0$ 则 $y_{k+1}=y_k$　$P_{k+1}=P_k+2\Delta y$



思考

1、试用Bresenhan算法（优化2）画线Line：P0(0，0), P1(5,2) 的像素点过程。

谢　谢！