采用增量思想的DDA算法，直观、易实现，每计算一个象素坐标，只需计算一个加法。

$$y_{i+1} = y_i + k$$

这个算法是否最优呢？若非最优，如何改进？

（1）改进效率。这个算法每步只做一个加法，能否再提高效率？

$$y_{i+1} = y_i + k$$

一般情况下k与y都是小数，而且每一步运算都要对y进行四舍五入后取整。

唯一改进的途径是把浮点运算变成整数加法！

（2）第二个思路是从直线方程类型做文章

$$y = kx + b$$

而直线的方程有许多类型，如两点式、一般式等。如用其它的直线方程来表示这条直线会不会有出人意料的效果？

直线绘制的三个著名的常用算法

1、数值微分法（DDA）

2、中点画线法

3、Bresenham算法

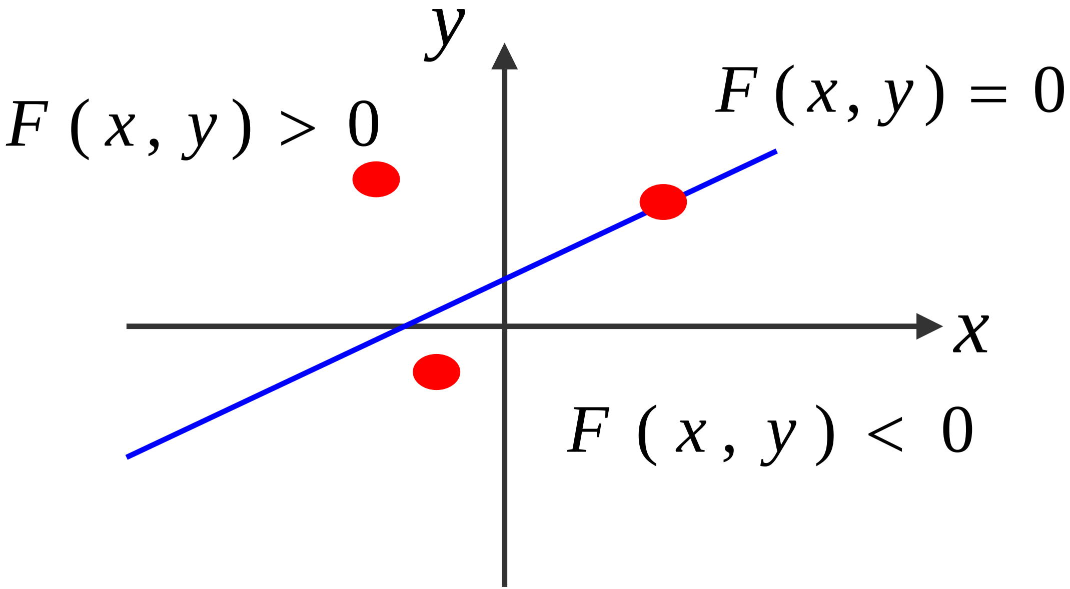
中点画线法

直线的一般式方程：

$$F(x, y) = 0$$

$$Ax + By + C = 0$$

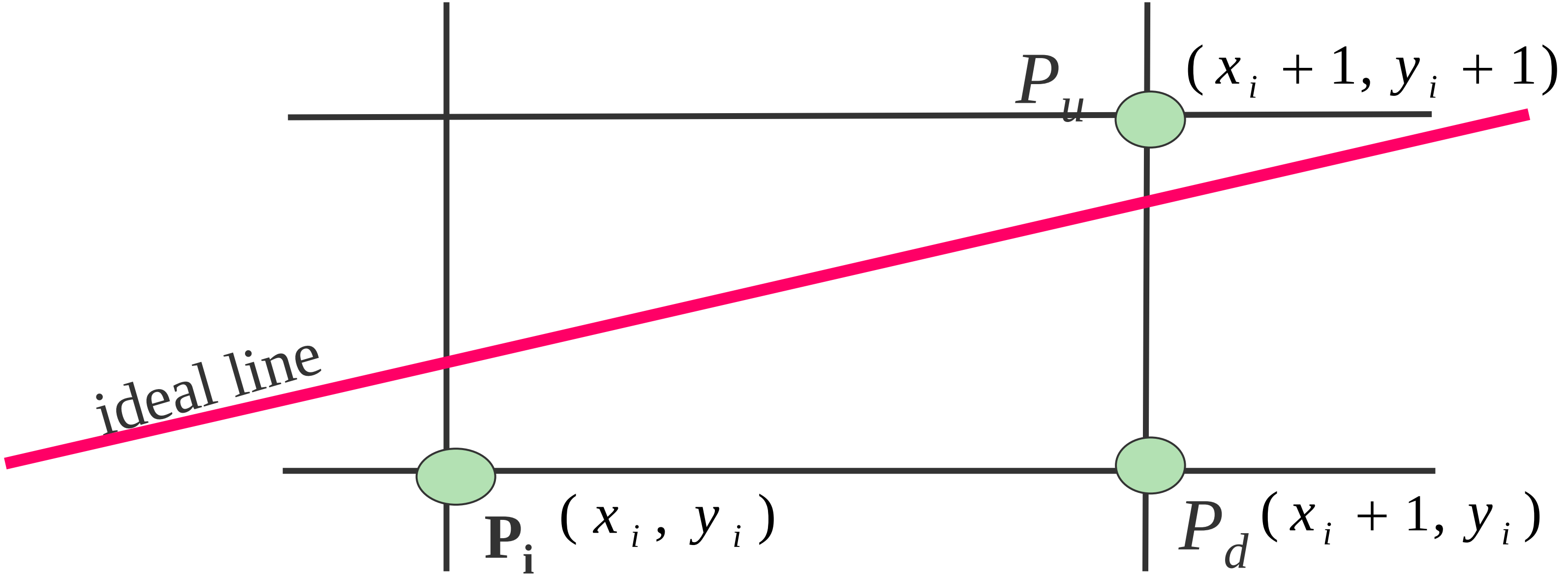
其中： $A = -(\Delta y)$; $B = (\Delta x)$; $C = -B(\Delta x)$

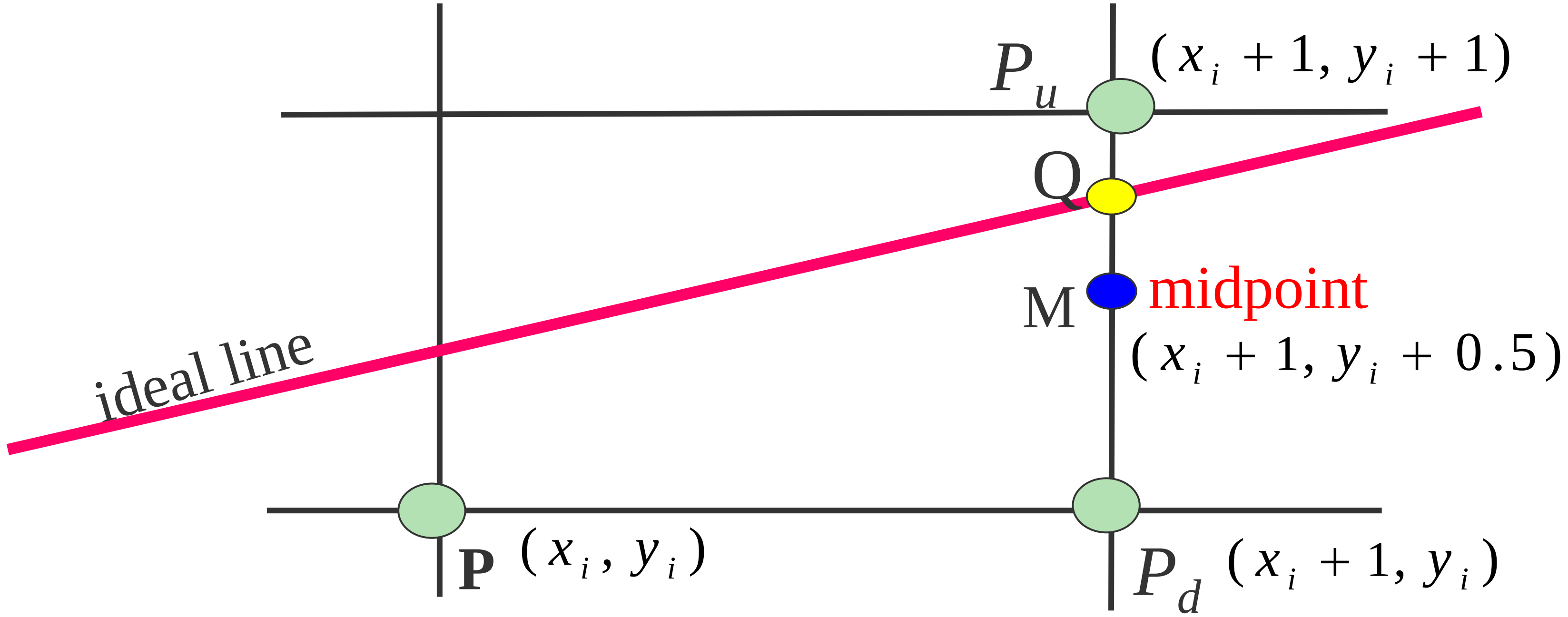


- 对于直线上的点：　$F(x,y) = 0$
- 对于直线上方的点：　$F(x,y) > 0$
- 对于直线下方的点：　$F(x,y) < 0$

每次在最大位移方向上走一步，而另一个方向是走步还是不走步要取决于中点误差项的判断。

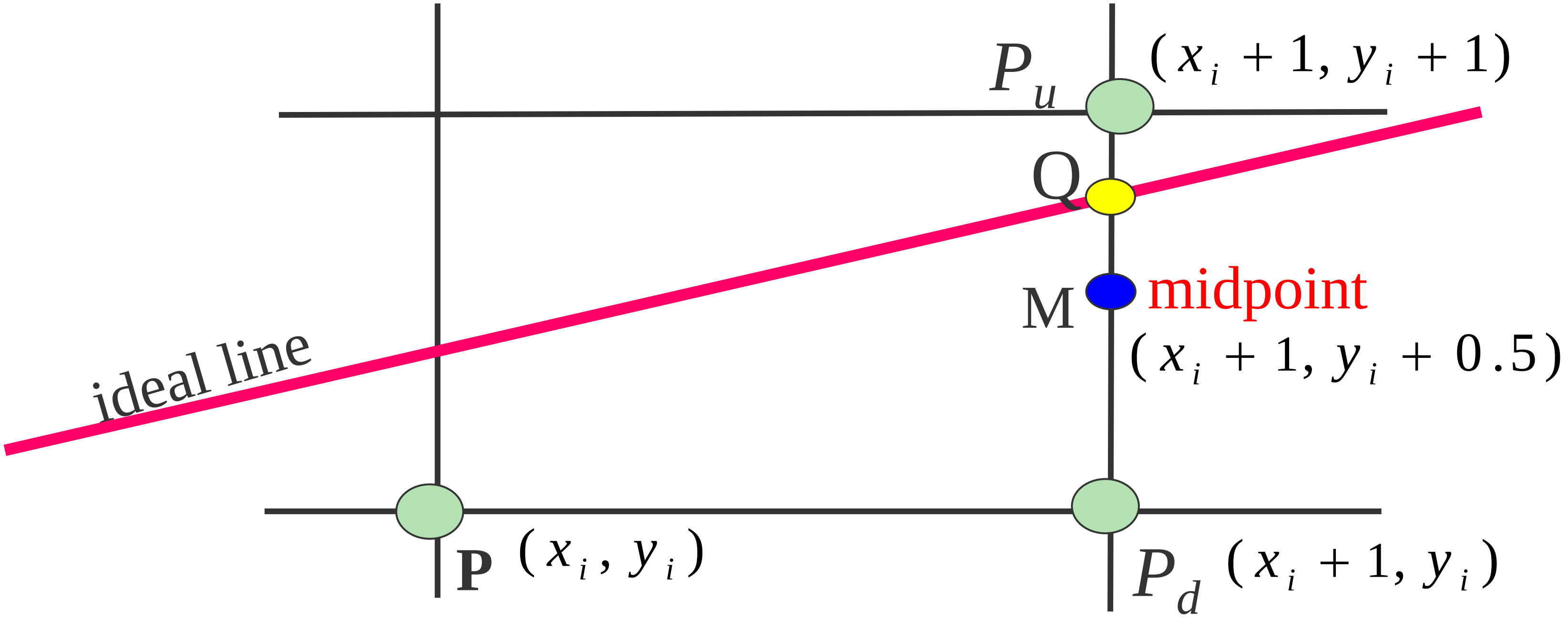
假定：$0 \leqslant |k| \leqslant 1$。因此，每次在x方向上加1，y方向上加1或不变需要判断。



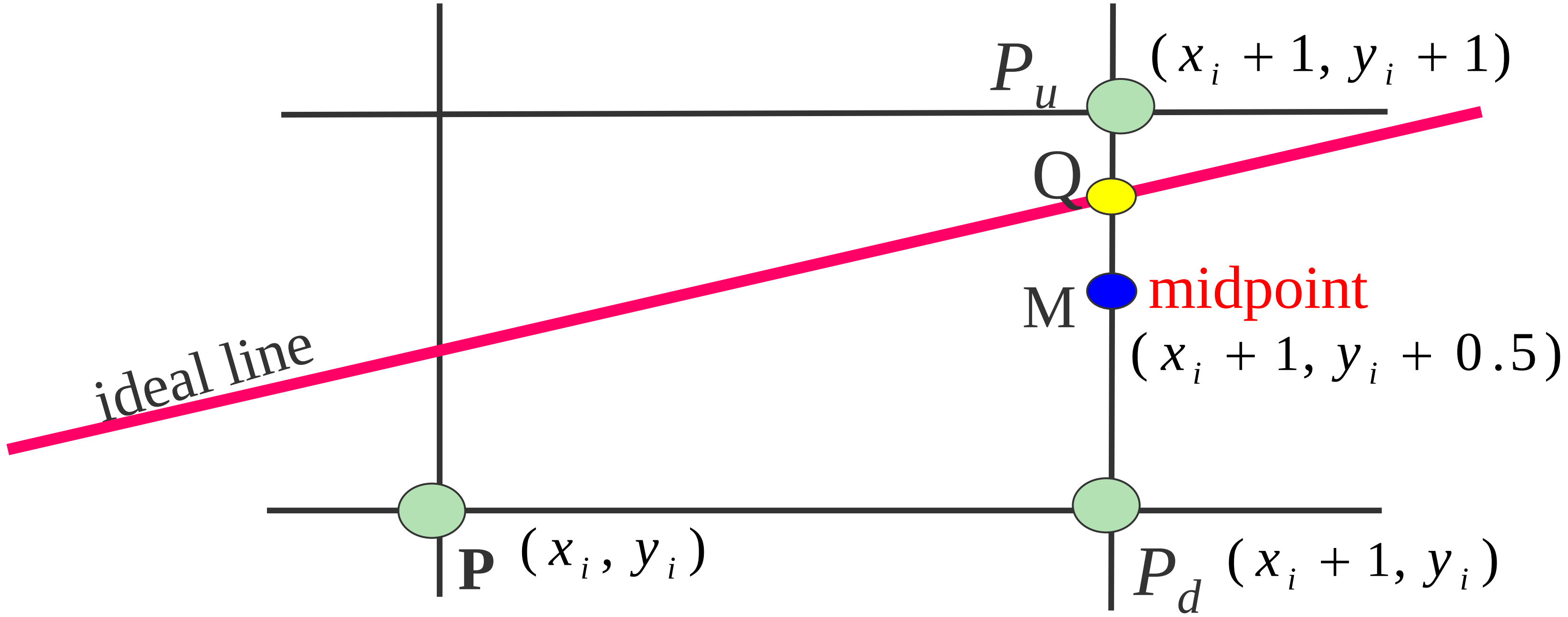


当M在Q的下方，则P_u离直线近，应为下一个象素点

当M在Q的上方，应取P_d为下一点。



如何判断Q在M的上方还是下方？



把M代入理想直线方程：

$$F(x_m, y_m) = Ax_m + By_m + C$$

$$d_i = F(x_m, y_m) = F(x_i + 1, y_i + 0.5)$$
$$= A(x_i + 1) + B(y_i + 0.5) + C$$

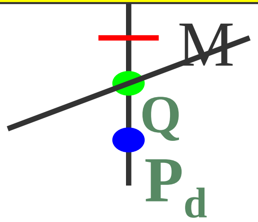
当$d <$

$$y = \begin{cases} y + 1 & (d < 0) \\ y & (d \geq 0) \end{cases}$$

这就是中点画线法的基本原理

P_u

当$d > 0$时： M在Q上方，应取P_d



当$d = 0$时： M在直线上，选p_d或p_u均可。

下面来分析一下中点画线算法的计算量？

$$y = \begin{cases} y+1 & (d<0) \\ y & (d \geq 0) \end{cases}$$

$$d_i = A(x_i + 1) + B(y_i + 0.5) + C$$

为了求出d值，需要两个乘法，四个加法

能否也采用增量计算，提高运算效率呢？

$$d_{i+1} = d_i + ?$$

增量

分析一下中点画线算法的计算量

$$y = \begin{cases} y+1 & (d < 0) \\ y & (d \geq 0) \end{cases}$$

$$d_i = A(x_i + 1) + B(y_i + 0.5) + C$$

能否也采用增量计算，提高运算效率呢？

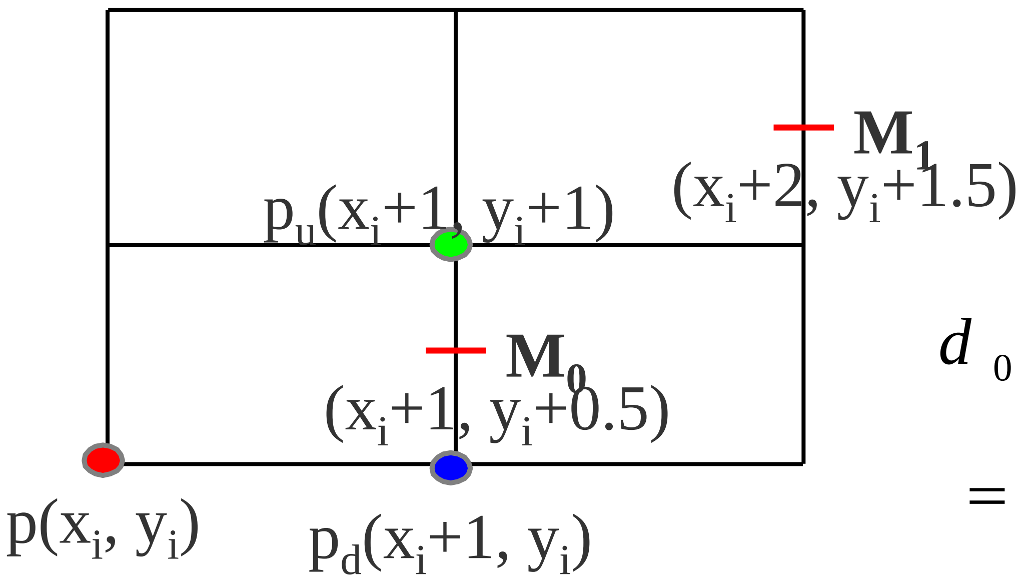
$$d_{i+1} = d_i + ?$$

增量

d是x, y的线性函数，采用增量计算是可行的

如何推导出d值的递推公式？

$$y = \begin{cases} y+1 & (d<0) \\ y & (d \ge 0) \end{cases}$$

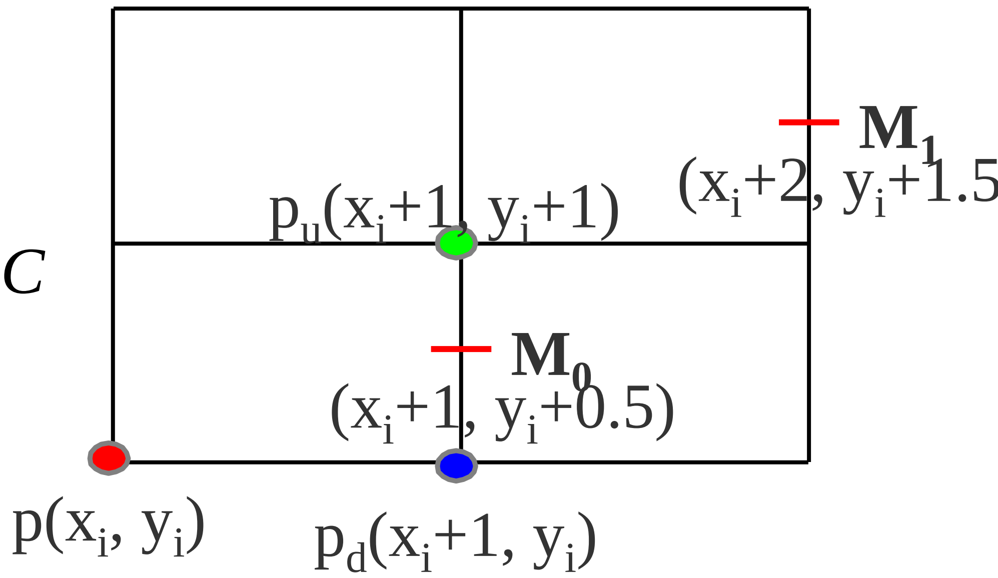


$$d_0 = F(x_{m0}, y_{m0})$$

$$= F(x_i+1, y_i+0.5)$$

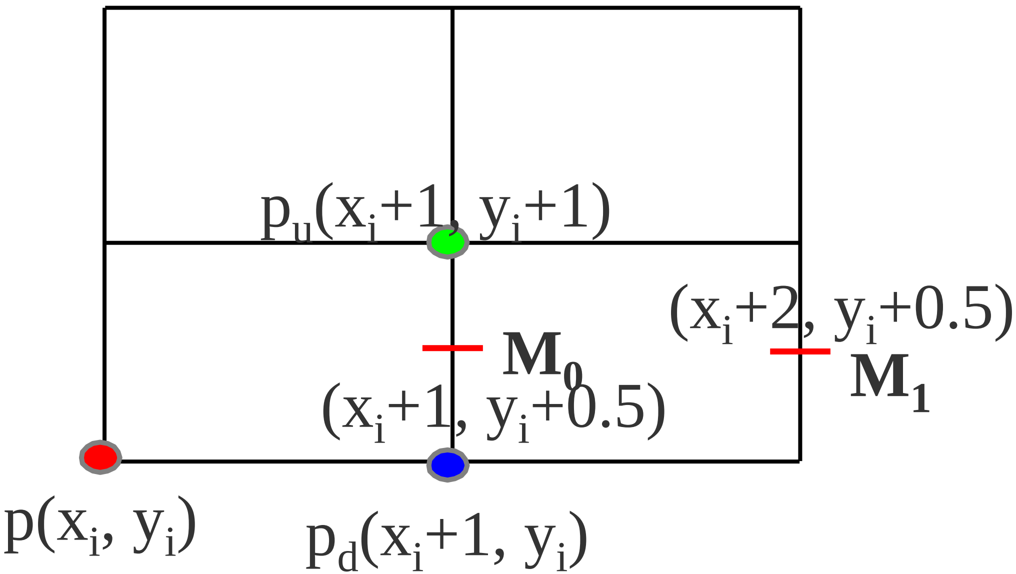
$$= A(x_i+1) + B(y_i+0.5) + C$$

$$d_0 = F(x_{m0}, y_{m0})$$
$$= F(x_i + 1, y_i + 0.5)$$
$$= A(x_i + 1) + B(y_i + 0.5) + C$$



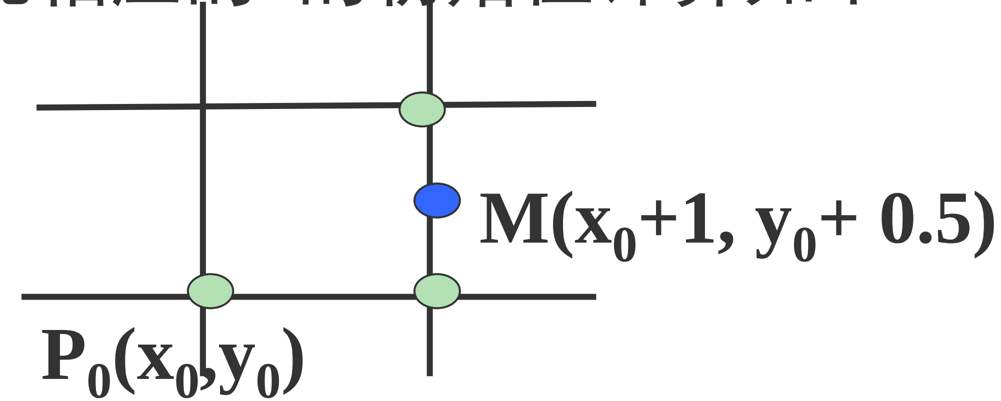
$$d_1 = F(x_{m1}, y_{m1})$$
$$= F(x_i + 2, y_i + 1.5)$$
$$= A(x_i + 2) + B(y_i + 1.5) + C$$
$$= A(x_i + 1) + B(y_i + 0.5) + C + A + B$$
$$= d_0 + A + B$$

$$y = \begin{cases} y+1 & (d < 0) \\ y & (d \geq 0) \end{cases}$$



$$\begin{aligned} d_1 &= F(x_i+2, y_i+0.5) \\ &= A(x_i+2) + B(y_i+0.5) + C \\ &= A(x_i+1) + B(y_i+0.5) + C + A \\ &= d_0 + A \end{aligned}$$

下面计算d的初始值d_0，直线的第一个像素$P_0(x_0, y_0)$在直线上，因此相应的d的初始值计算如下：



$$
\begin{aligned}
d_0 &= F(x_0+1, y_0+0.5) \\
&= A(x_0+1) + B(y_0+0.5) + C \\
&= Ax_0 + By_0 + C + A + 0.5B \\
&= A + 0.5B
\end{aligned}
$$

$$d_{new} = \begin{cases} d_{old} + A + B & d < 0 \\ d_{old} + A & d \geq 0 \end{cases} \qquad d_0 = A + 0.5B$$

至此，中点算法至少可以和DDA算法一样好！

可以用2d代替d来摆脱浮点运算，写出仅包含**整数运算**的算法。

这样，中点生成直线的算法提高到整数加法，优于DDA算法。

小 结

（1） $Ax + By + C = 0$

（2）通过判中点的符号，最终可以只进行整数加法

这个算法是否还有改进的余地？