计算机图形学小白入门

————从0开始实现OpenGL

Brensenham直线绘制算法

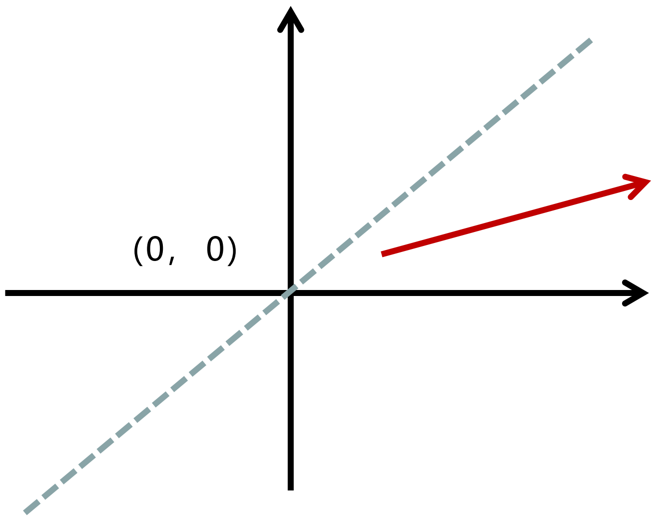
授课：赵新政
资深三维工程师

专注3D图形学技术
教育品牌

如何在屏幕上绘制直线

- 问题定义：给到两个点 $p_1(x_1,y_1)$，$p_2(x_2,y_2)$;请在屏幕像素空间绘制一条直线

- 最简单的问题模型满足如下条件：
- —— $x_1 < x_2$
- ——二者构成直线斜率：$0 < k < 1$



最简单的方案

- 根据两点坐标 $p_1(x_1,y_1)$,$p_2(x_2,y_2)$ 可求出如下直线方程：

$$y = kx + b$$
$$k = (y_2 - y_1)/(x_2 - x_1)$$

- 根据直线方程，可以得到直线绘制算法如下:

```
x=x1;
WHILE(x < x2){
    y = (int)(kx + b);
    drawPoint(x,y);
    x++;
}
```

缺点：

在本算法中，有一次k * x这样的浮点数乘法运算以及一次+b这样的浮点数加法运算。

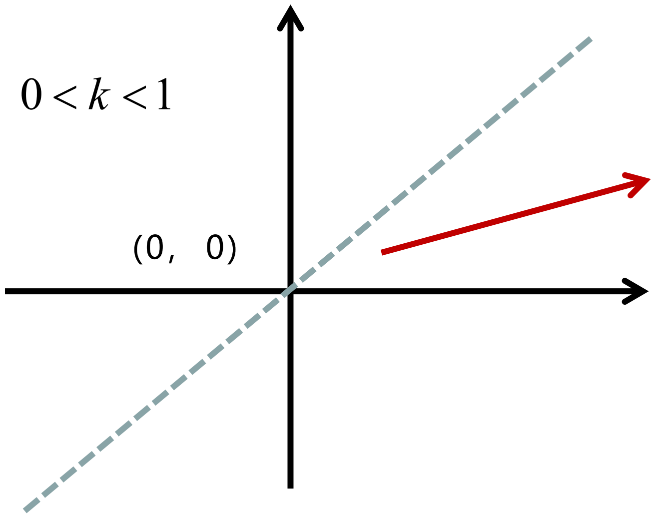
浮点数运算效率较低，可否找到一种算法，只运用整数的四则运算呢？

Brensenham直线绘制算法

- 本算法整体的计算过程中，只使用了整数型数据进行运算与决策，省略了浮点数运算，从而提高了绘制效率

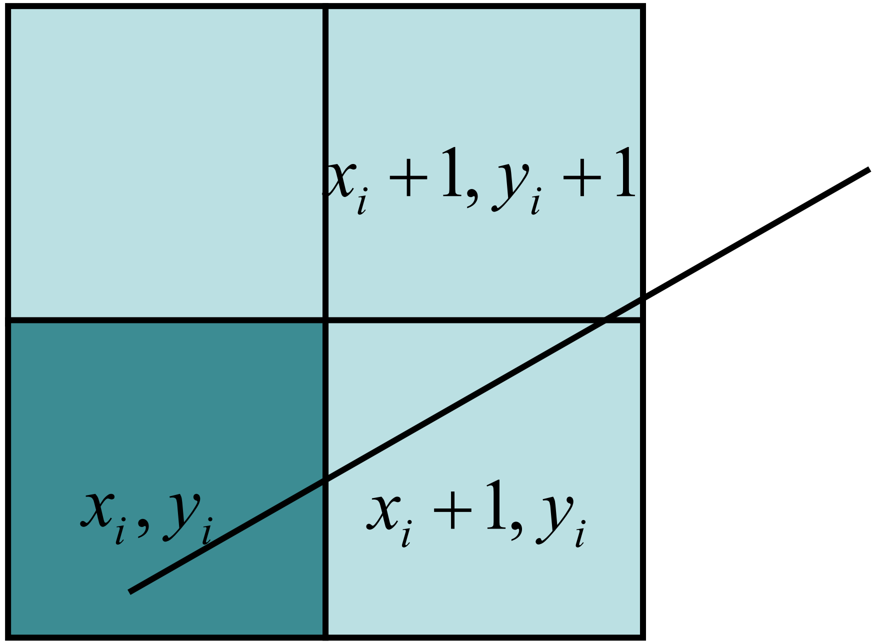
- 问题定义：给到两个点 $p_1(x_1,y_1)$，$p_2(x_2,y_2)$;请在屏幕像素空间绘制一条直线

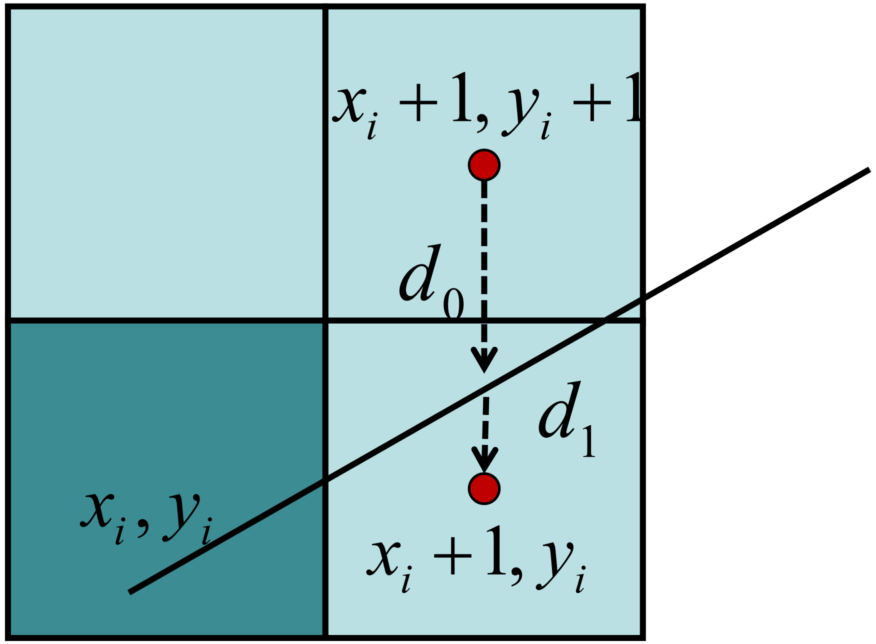
- **研究最简单的情况：**
- —— $x_1 < x_2$
- ——二者构成直线斜率：$0 < k < 1$



$$y = kx + b$$

$$k = (y_2 - y_1)/(x_2 - x_1)$$





假设当前直线已经通过点 x_i, y_i ，

那么当x方向向前步进+1的时候，

如何能够决定y的选项呢？

$$d_0 = y_i + 1 - k.(x_i + 1) - b$$

$$d_1 = k.(x_i + 1) + b - y_i$$

$$d_0 = y_i + 1 - k.(x_i + 1) - b$$

$$d_1 = k.(x_i + 1) + b - y_i$$

$$d_1 - d_0 = k.(x_i + 1) + b - y_i - y_i - 1 + k.(x_i + 1) + b$$
$$= 2k.(x_i + 1) - 2y_i - 1 + 2b$$

由已知条件:

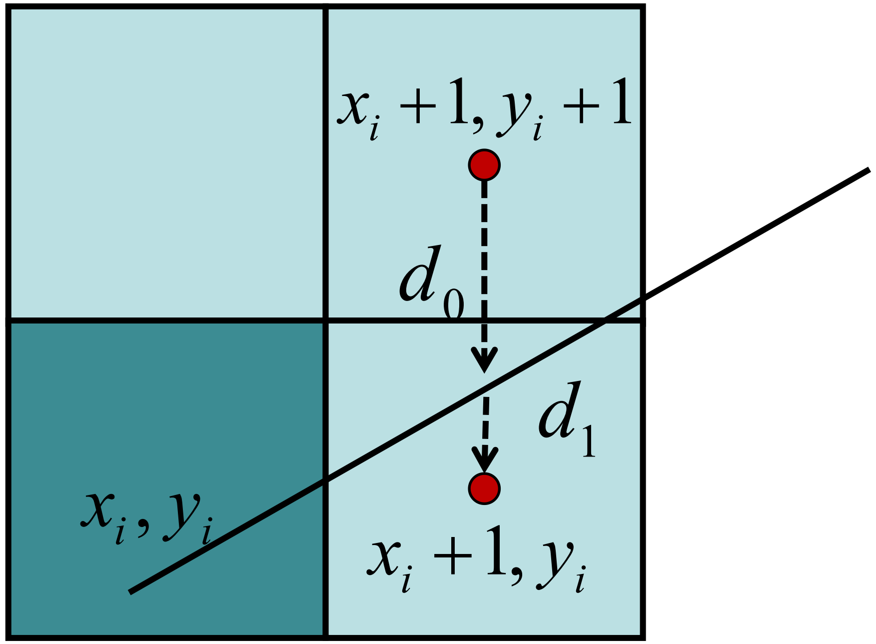
$$k = (y_2 - y_1)/(x_2 - x_1)$$
$$\Delta x = (x_2 - x_1) > 0$$

等式两边同乘以 Δx 表达式的正负号不会发生变化

$$p_i = \Delta x(d_1 - d_0) = 2\Delta y.(x_i + 1) - (2y_i - 1 + 2b).\Delta x$$
$$= 2\Delta y.x_i - 2\Delta x.y_i + (2\Delta y + 2b.\Delta x - \Delta x)$$

使用 p_i 的正负即可判断选择哪个栅格

$$p_i = 2\Delta y.x_i - 2\Delta x.y_i + (2\Delta y + 2b.\Delta x - \Delta x)$$



$$\begin{cases} p_i > 0 \quad d_1 > d_0 \quad y_{i+1} = y_i + 1 \\ p_i < 0 \quad d_1 < d_0 \quad y_{i+1} = y_i \end{cases}$$

迭代模型

- 我们已经知道了任何一个p的表达式，那么是否可以利用迭代法来计算每个接下来的p呢?

$$p_i = 2\Delta y.x_i - 2\Delta x.y_i + (2\Delta y + 2b.\Delta x - \Delta x)$$

$$p_{i+1} = 2\Delta y.(x_i + 1) - 2\Delta x.y_{i+1} + (2\Delta y + 2b.\Delta x - \Delta x)$$

$$p_{i+1} - p_i = 2\Delta y - 2\Delta x.(y_{i+1} - y_i)$$

- 考察第一个p值，带入 $x_1, y_1 = k.x_1 + b$

$$p_1 = 2\Delta y.x_1 - 2\Delta x.(\frac{\Delta y}{\Delta x}.x_1 + b) + (2\Delta y + 2b.\Delta x - \Delta x)$$

$$p_1 = 2\Delta y - \Delta x$$

迭代模型

- 整理上述推导，可以得到如下算法：

$$p_{i+1}-p_i=2\Delta y-2\Delta x.(y_{i+1}-y_i)$$

```
x=x1;y=y1;
p = 2Δy − Δx
WHILE(x < x2){
        drawPoint(x,y);
        x++;
        if(p>=0){
            y=y+1;
            p=p-2.Δx
        }
        p=p+2.Δy
}
```

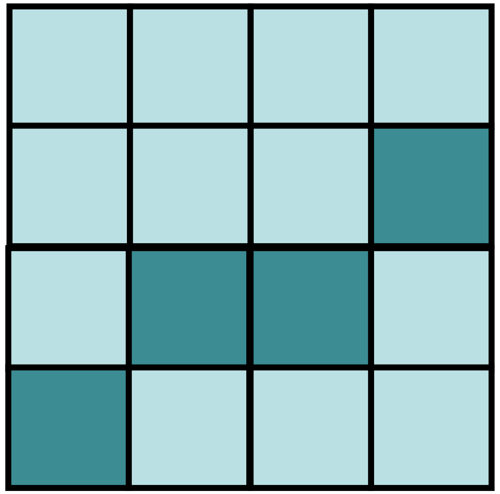
注意：本方法只对满足前置条件的直线生效

- **最简单的情况：**
- —— $x_1 < x_2$
- ——二者构成直线斜率：$0 < k < 1$

验证模型

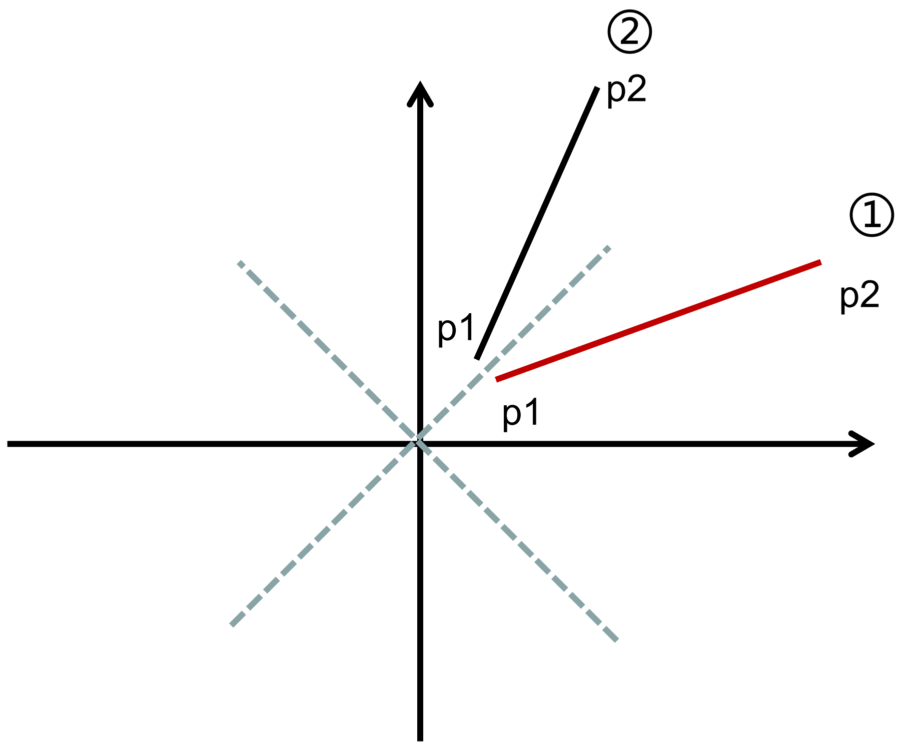
$$p_1(0,0) \quad p_2(100,60) \quad \Delta y = 60; \Delta x = 100 \quad p = 2\Delta y - \Delta x$$

轮次	p值	x当前值	y当前值	y下一轮值
第一轮	p=20	x=0	y=0	y=1
第二轮	p=-60	x=1	y=1	y=1
第三轮	p=60	x=2	y=1	y=2
第四轮	p=-20	x=3	y=2	y=2



拓展更多情况(一)

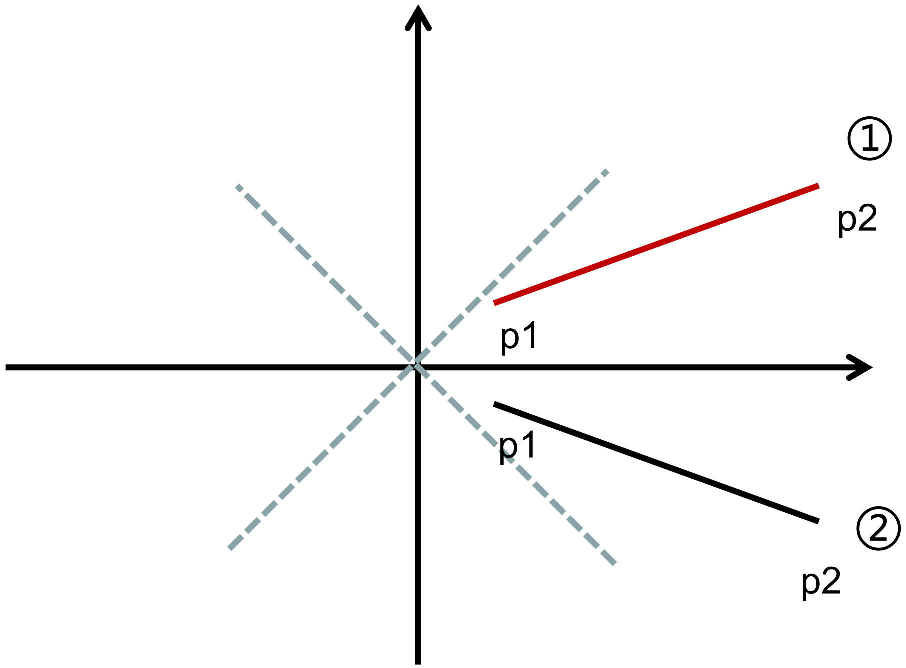
- 考虑只满足 $x_1 < x_2$ $y_1 < y_2$ ，但是 $k > 1$ 的情况



- 对于2的情况，只需要把**两个点的xy值各自互换**就可以，在绘制过程中，把**每个点的xy互换**

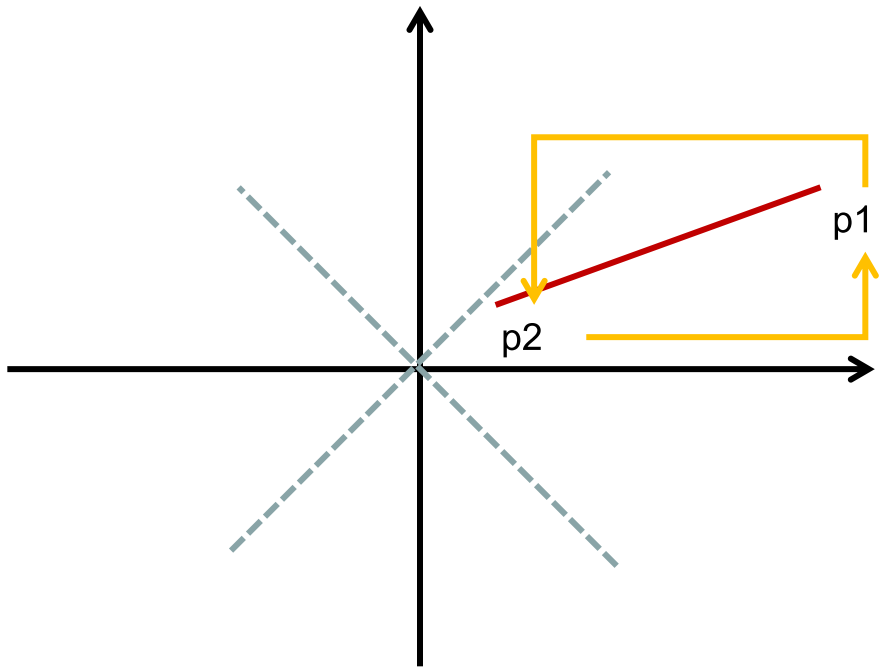
拓展更多情况(二)

- 考虑只满足 $x_1 < x_2 \quad y_1 > y_2$ 的情况



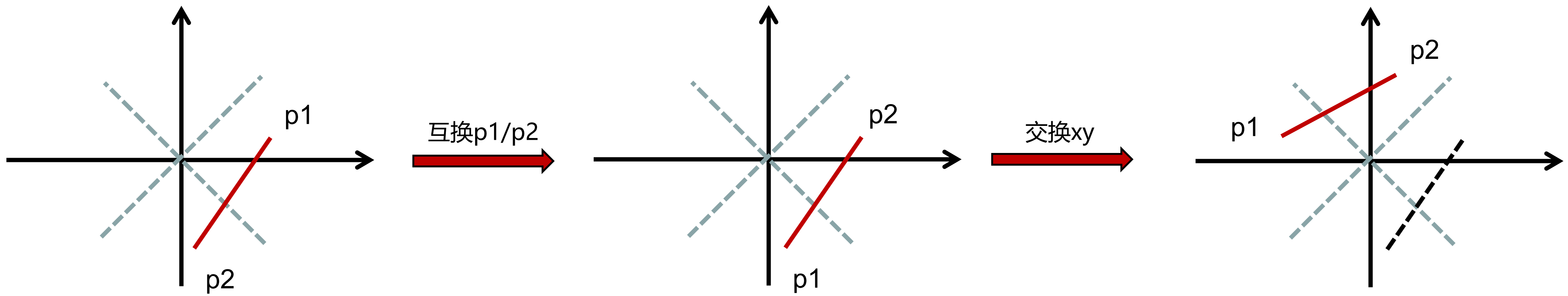
- 对于2情况，只需要把**两个点的y值变符号**即可到达第一象限，在绘制的过程中，把**每个点的y值符号变回去**

拓展更多情况(三)



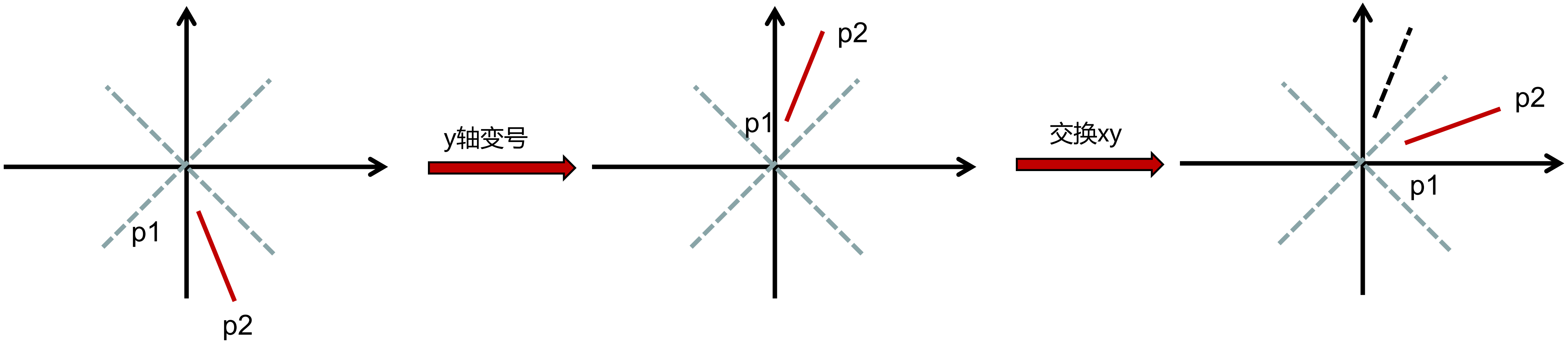
- 只需要把绘制**起始点**与**中止点**互换即可

拓展情况-案例（一）



绘制的时候，每一个点坐标得到后，要交换xy坐标，然后再绘制

拓展情况-案例（二）



绘制的时候，每一个点坐标得到后，要交换xy坐标，再改变y坐标符号，然后绘制