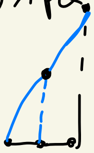
①

```cpp
void line(int x0, int y0, int x1, int y1, TGAImage &image, TGAColor color) {
    bool steep = false;
    if (std::abs(x0-x1)<std::abs(y0-y1)) { // if the line is steep, we transpose the image
        std::swap(x0, y0);
        std::swap(x1, y1);
        steep = true;
    }
    if (x0>x1) { // make it left-to-right
        std::swap(x0, x1);
        std::swap(y0, y1);
    }
    for (int x=x0; x<=x1; x++) {
        float t = (x-x0)/(float)(x1-x0);
        int y = y0*(1.-t) + y1*t;
        if (steep) {
            image.set(y, x, color); // if transposed, de-transpose
        } else {
            image.set(x, y, color);
        }
    }
}
```

$\rightarrow y = y_0 + t(y_1 - y_0)$

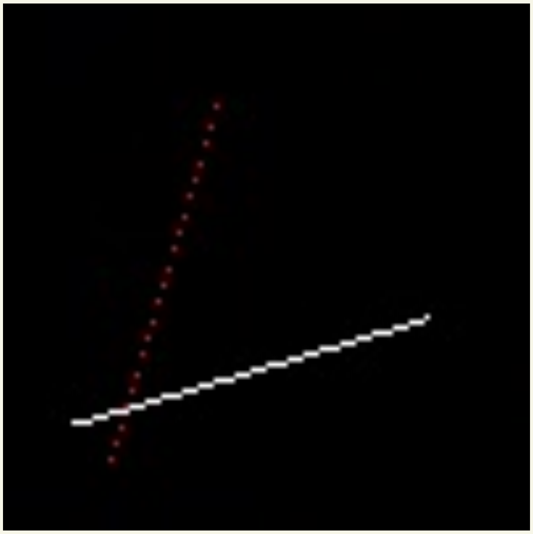
<1> 为什么要有 steep:

(x, y) 代表一个像素的位置. 如果 $|x_0 - x_1| < |y_0 - y_1|$:



$y_0 - y_1 > x_0 - x_1$

画出来的线是不连续的:



<2> 保证 $x_0 < x_1$

这样 for 循环从 x_0 开始

是正确的 for 循环.

优化：

<1>
```
for (int x=x0; x<=x1; x++) {
    float t = (x-x0)/(float)(x1-x0);
    int y = y0*(1.-t) + y1*t;
    if (steep) {
        image.set(y, x, color); // if transposed, de-transpose
    } else {
        image.set(x, y, color);
    }
}
```

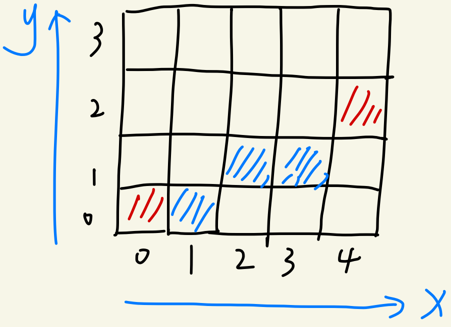
→ 因为 divisor (float)(x1-x0) 不变，所以可考虑将其提到 for 循环外部.

注意 int 的除法得到的还是 int. 所以如果用 float/double 接收结果，将分母或分子转换为 float/double.

进一步考虑：
能否将除法转换为乘/加法？

优化 ①

```
int dx = x1-x0;
int dy = y1-y0;
float derror = std::abs(dy/float(dx));
float error = 0;
int y = y0;
for (int x=x0; x<=x1; x++) {
    if (steep) {
        image.set(y, x, color);
    } else {
        image.set(x, y, color);
    }
    error += derror;
    if (error>.5) {
        y += (y1>y0?1:-1);
        error -= 1.;
    }
}
```



(0,0) → (4,2) .

$derror = \frac{dy}{dx} = \frac{2}{4} = \frac{1}{2}$

当 x 从 x_0 (0) 自增至 x=1 时, error += 0.5 = 0.5
此时因为还未 > 0.5, 填充 (1,0) 格 .

当 x 从 1 到 2 时, error = 1 > 0.5 .
填充 (2,1) 格 . 此时 error -= 1, error = 0 .

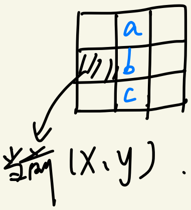
当 x 从 2 到 3 时, error = 0.5, 填充 (3,1)

x=4, 填充 (4,2) .

```
int dx = x1-x0;
int dy = y1-y0;
float derror = std::abs(dy/float(dx));
float error = 0;
int y = y0;
for (int x=x0; x<=x1; x++) {
    if (steep) {
        image.set(y, x, color);
    } else {
        image.set(x, y, color);
    }
    error += derror;
    if (error>.5) {
        y += (y1>y0?1:-1);
        error -= 1.;
    }
}
```

(1) 为什么 error>.5 ?

因为对每个像素点来说，x增加dx，那么y增加dy，x增加1，y增加$\frac{dy}{dx}$，也就是 error += $\frac{dy}{dx}$，为了保证曲线的连续性，



下一个像素，如果x往前增了1，即(x+1, 口)，那么像素一定是a, b, c中的一个.

a → (x+1, y+1)　b → (x+1, y)　c → (x+1, y-1).

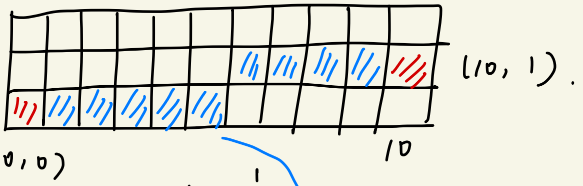
因为这样画出来的曲线才不会"断开".

如果 $y_1>y_0$，y+=1 → y在此前的y值上加1（往上走）

$y_1<y_0$，y-=1 → y在此前的y值上减1（往下走）.

error作为一个累计值，如果$\frac{dy}{dx}$非常小，只有累计的$\frac{dy}{dx}>0.5$即超过了一半时，才取 y+=1 或 y-=1，否则 y不变：

assume 一条非常平缓的曲线：



error += $\frac{1}{10}$

到此处 error>0.5（即超过了一半），y+=1.

优化(2):

```
int dx = x1-x0;
int dy = y1-y0;
int derror2 = std::abs(dy)*2;
int error2 = 0;
int y = y0;
for (int x=x0; x<=x1; x++) {
    if (steep) {
        image.set(y, x, color);
    } else {
        image.set(x, y, color);
    }
    error2 += derror2;
    if (error2 > dx) {
        y += (y1>y0?1:-1);
        error2 -= dx*2;
    }
}
```

```
int dx = x1-x0;
int dy = y1-y0;
float derror = std::abs(dy/float(dx));
float error = 0;
int y = y0;
for (int x=x0; x<=x1; x++) {
    if (steep) {
        image.set(y, x, color);
    } else {
        image.set(x, y, color);
    }
    error += derror;
    if (error>.5) {
        y += (y1>y0?1:-1);
        error -= 1.;
    }
}
```

优化：用 int 代替 float.

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

objective：消除小数

$$error \mathrel{+}= \frac{dy}{dx} \xrightarrow{① 消除 \frac{dy}{dx}}$$

$$error > 0.5$$

$$error \mathrel{-}= 1$$

$$derror \cdot dx = dy$$

$$error \cdot dx \mathrel{+}= dy$$

$$error \cdot dx > 0.5 \cdot dx$$

$$error \cdot dx \mathrel{-}= dx$$

② 消除 0.5

$$2 \cdot derror \cdot dx = 2 \cdot dy$$

$$2 \cdot error \cdot dx \mathrel{+}= 2 \cdot dy$$

$$2 \cdot error \cdot dx > dx$$

$$2 \cdot error \cdot dx \mathrel{-}= 2 \cdot dx$$

$$error2 = 2 \cdot error \cdot dx \ .$$

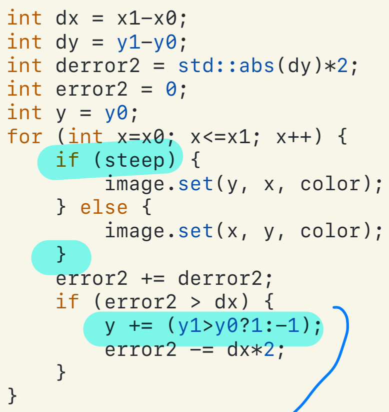
$$derror2 = 2 \cdot derror \cdot dx = 2 \cdot dy$$

$$derror2 = 2 \cdot dy$$

$$error2 \mathrel{+}= derror2$$

$$error2 > dx$$

$$error2 \mathrel{-}= 2 \cdot dx$$



```
int dx = x1-x0;
int dy = y1-y0;
int derror2 = std::abs(dy)*2;
int error2 = 0;
int y = y0;
for (int x=x0; x<=x1; x++) {
    if (steep) {
        image.set(y, x, color);
    } else {
        image.set(x, y, color);
    }
    error2 += derror2;
    if (error2 > dx) {
        y += (y1>y0?1:-1);
        error2 -= dx*2;
    }
}
```

↓优化：

for 循环里尽量不要有 branching 语句

branching 可以优化：

```
for (int x=x0; x<=x1; x++) {
    if (steep) {
        image.set(y, x, color);
    } else {
        image.set(x, y, color);
    }
    error2 += derror2;
    if (error2 > dx) {
        y += (y1>y0?1:-1);
        error2 -= dx*2;
    }
}
```

```
if(steep) {
    for(int x = x0; x<=x1; ++x) {
        img.set_pixel_color(y, x, color);
        error2 += derror2;
        if(error2 > dx) {
            y += (y1>y0? 1 : -1);
            error2 -= dx*2;
        }
    }
} else {
    for(int x = x0; x<=x1; ++x) {
        img.set_pixel_color(x, y, color);
        error2 += derror2;
        if(error2 > dx) {
            y += (y1>y0? 1 : -1);
            error2 -= dx*2;
        }
    }
}
```

You can also do the same for:
`(y1>y0? 1 : -1);`
By computing increment value at the start of the function:
`const int yincr = (y1>y0? 1 : -1);`
and then only doing a
`y += yincr;`
in the loop.