华东师范大学计算机科学与技术学院实验报告

实验课程：计算机图形学	**年级：**2018 级	**实验成绩：**
实验名称：直线算法	**姓名：**	
实验编号：2	**学号：**	**实验日期：**2021-3-16
指导教师：王长波、李洋	**组号：**	**实验时间：**13:00-14:30

一、实验目的

利用操作系统 API 实现基本直线绘制功能。

二、实验内容与实验步骤

实验内容与实验步骤：

完善上一次实验内容设置的基本绘制逻辑

存储每次鼠标移动的信息

恢复上一次鼠标移动所画的区域

实现 DDA 直线算法

实现 Bresenham 直线算法

三、实验环境

Windows 10

Visual studio 2019

四、实验过程与分析

1.首先我导入了助教写好的操作系统 APi 的代码

2. 打开 line.cpp 文件，里面有需要我补充的两个画直线的算法 DAA 和 Bresenham，

两个算法均需要计算出每一个要画的点的坐标，然后利用已经创建好的结构体 Point，存入坐标，再把当前 Point 存入最后要返回的 Vector 中。

关键代码如下：

DAA

```cpp
std::vector<Point> draw_line_DDA(Point p_src, Point p_dst) {

    std::vector<Point> result;
    float x0 = p_src.x, y0 = p_src.y;
    float x1 = p_dst.x, y1 = p_dst.y;
    float dx = x1 - x0;
    float dy = y1 - y0;
    int steps = abs(y1 - y0);
    // 这里是取绘制点数最多的值
    if (fabs(dx)>fabs(dy))
        steps = abs(x1 - x0);
    // 先初始化两个坐标为起点
    float x = x0;
    float y = y0;
    // 然后定义两个x和y的增量变量
    // 分别用两点的x、y的差除与需要绘制的点数来获得每绘制结束一个点后需要前进多少
    float xinc = dx / steps;
    float yinc = dy / steps;
    Point po;
    po.x = round(x); po.y = round(y);
    result.push_back(po);
    for (int i = 0; i<steps; ++i)
    {
        // 注意这里需要放到本次循环开始
        // 因为第一个点已经绘制了
        x += xinc;
        y += yinc;

        po.x = round(x); po.y = round(y);
        result.push_back(po);
    }
    return result;
}
```

Bresenham

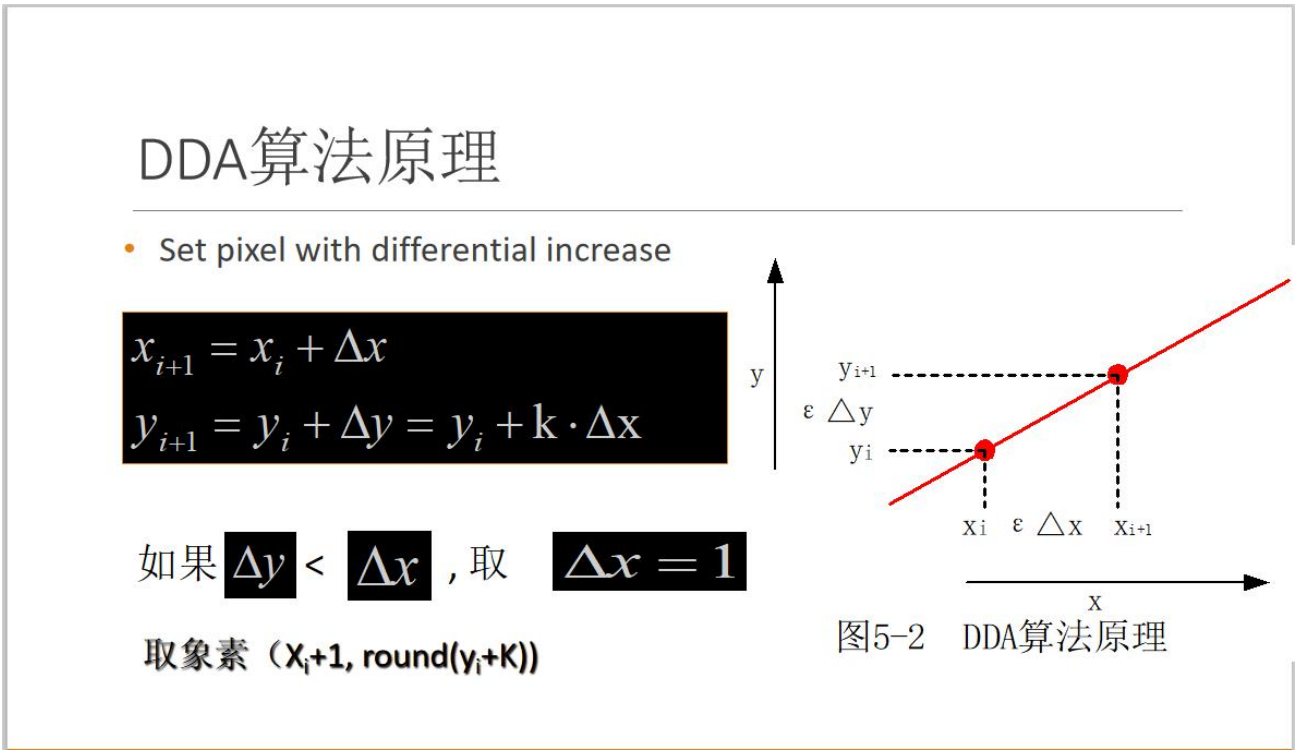
```cpp
std::vector<Point> draw_line_bresenham(Point p_src, Point p_dst) {


    std::vector<Point> result;
    int x0 = p_src.x, y0 = p_src.y, x1 = p_dst.x, y1 = p_dst.y;
    int dx = abs(x1 - x0), sx = x0 < x1 ? 1 : -1;
    int dy = abs(y1 - y0), sy = y0 < y1 ? 1 : -1;
    int err = (dx > dy ? dx : -dy) / 2, e2;
    for (;;) {
        Point P;
        P.x = x0;
        P.y = y0;
        if (x0 == x1 && y0 == y1) break;
        e2 = err;
        if (e2 > -dx) { err -= dy; x0 += sx; }
        if (e2 < dy) { err += dx; y0 += sy; }
    }
```

五、实验结果总结

1.首先我在导入代码方面遇到了问题，VS 这个平台我不熟悉，导致创建好空项目后不知道应该怎么导入，后面询问老师找到解决方法。

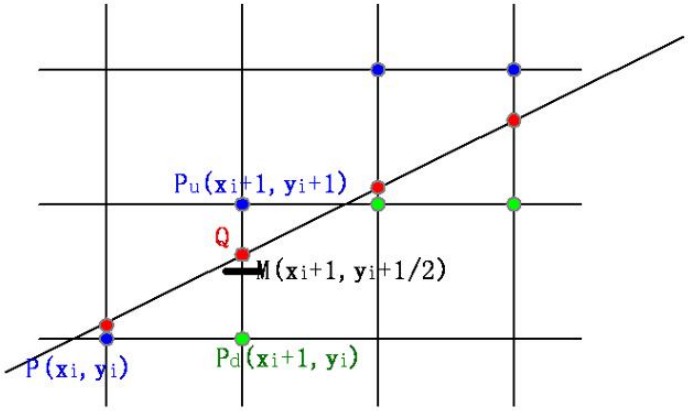
在这里可以调出解决方案资源管理器视图，然后就可以导入相应的文件了。

2. 两个算法的原理



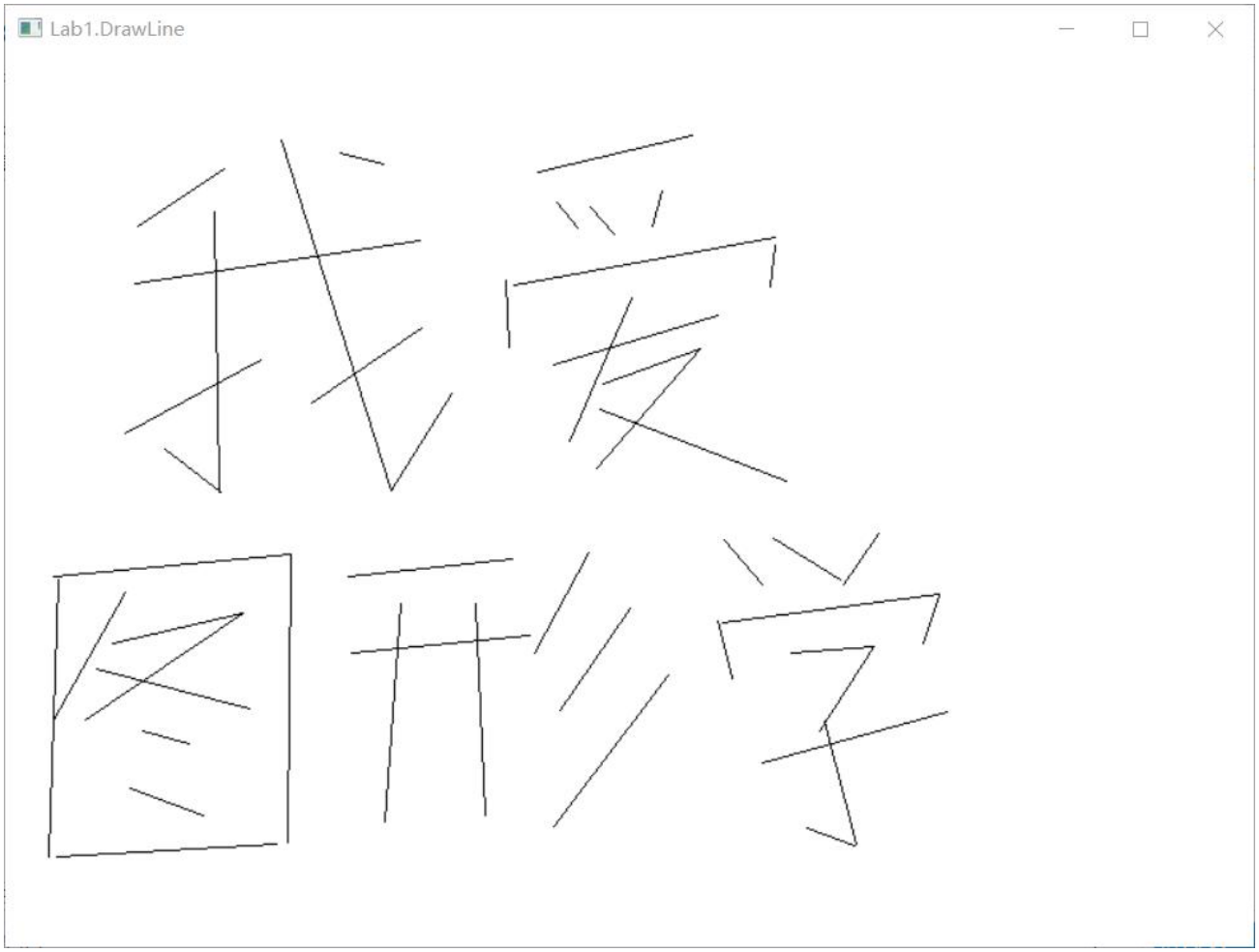
中点Bresenham算法

- 基本原理：
- 假定0≤k≤1，x是最大位移方向
- 要判断下一步画P_u还是P_d，只需要判断M点在线上还是线下.



原理如上，我在编程的时候先是按照自己的理解，写了一个麻烦的代码。后面在网络上查询资料进行了改进。

3. 实验结果



4.能发现 DAA 算法在实现后，即在画直线的时候反应会慢一些

六、附录（参考资料）

https://blog.csdn.net/cjw_soledad/article/details/78886117