武汉大学国际软件学院

实验报告

课程名称 计算机图形学

专业年级 **2016**

姓　　名 刘瑞康

学　　号 **2016302580242**

协 作 者 无

实验学期 **2017-2018** 学年 下 学期

课堂时数 **6** 课外时数 **6**

填写时间 **2018** 年 **5** 月 **20** 日

实验概述

【实验项目名称】：**The final project of computer graphics**

【实验目的】： Get familiar with basic concepts of Computer Graphics

【实验环境】Microsoft Visual Studio 2015 + EasyX(Graphics library for C++)

【参考资料】：Teaching slides and CSDN

实验内容

【实验方案设计】： 全部实验均使用 **C++**实现

Ⅰ. 二维题目 - 1

1. 功能描述

利用 EasyX 库的 initgraph 函数生成一个 400×300 分辨率的位图，位图背景为黑色，同时随机生成 XX 个点并连线，连线为白色。利用种子点填充算法，在位图任意位置用鼠标单击后设置种子点，以白色连线为边缘实现空白位置的填充。填充计算中使用 Sleep 函数，使得循环每隔一小段时间进行，从而实现动画效果

2. 算法说明

采用 4 连通非递归的种子填充算法。

从多边形区域的一个内点（即鼠标点击设置的种子点）开始，将其加入待填充队列，对于带填充队列的每一个像素点，逐一判断该点是否已被填充（或是到达连线边缘），若未被填充，则将该点置为填充（白色）并将该像素上、下、左、右四个方向的像素点加入待填充队列，从而由内向外用给定的颜色画点填充直到边界内的像素点均被填充为止。

3. 关键技术实现

```
//非递归的种子填充算法
void seedsFilling(int x, int y)
{
    //若种子点已被填充或是边界则不做处理
    if (getpixel(x, y) == WHITE)
        return;

    pair<int, int> currentPoint;
    queue<pair<int, int>> pointsQueue;
    pointsQueue.push(pair<int, int>(x, y)); //将种子点推入队列
    //对队列中的每一点进行处理
```

```
    while (!pointsQueue.empty())
    {
        Sleep(1);    //为每次循环设置间隔，实现动画效果
        currentPoint = pointsQueue.front();
        pointsQueue.pop();
        //若该点已被填充或是边界则不做处理
        if (getpixel(currentPoint.first, currentPoint.second) == WHITE)
            continue;
        else
        {   //否则将其填充并将四连通方向上的像素点加入队列
            putpixel(currentPoint.first, currentPoint.second, WHITE);
            pointsQueue.push(pair<int, int>(currentPoint.first - 1, currentPoint.second));
            pointsQueue.push(pair<int, int>(currentPoint.first + 1, currentPoint.second));
            pointsQueue.push(pair<int, int>(currentPoint.first, currentPoint.second - 1));
            pointsQueue.push(pair<int, int>(currentPoint.first, currentPoint.second + 1));
        }
    }
}
//单击填充响应
void clickToFill()
{
    MOUSEMSG  m = GetMouseMsg();
    //等待鼠标事件，直到获得单击事件，
    while (m.uMsg != WM_LBUTTONDOWN)
        m = GetMouseMsg();
    seedsFilling(m.x, m.y);    //以点击位置为种子点进行填充
}
//随机生成点的个数，随机生成相应位置并连线
void drawPolygon()
{
    srand((unsigned)time(NULL));
    rectangle(0, 0, WIDTH - 1, HEIGHT - 1);

    int numOfPoints = rand() % 8 + 3;
    int *points = new int(numOfPoints * 2);
    for (int i = 0; i < numOfPoints; i++)
    {
        points[2 * i] = rand() % WIDTH;
        points[2 * i + 1] = rand() % HEIGHT;
    }
    drawpoly(numOfPoints, points);
}
```

Ⅱ. 二维题目 - 2

1. 功能描述

利用 EasyX 库的 initgraph 函数生成一个 600×400 的位图,在该像素点范围内以随机的方式生成直线条数以及每条直线两个端点的坐标，然后利用 Bresenham 画线算法，实现直线段的绘制。每次绘制的直线条数不少于 10 条；绘制计算中使用 Sleep 函数，使得循环每隔一小段时间进行，从而实现动画效果。

2. 算法说明

采用递推步进的办法，令每次最大变化方向的坐标步进一个象素，同时另一个方向的坐标依据误差判别式的符号来决定是否也要步进一个象素。

其中，满足 0≤k≤1 且 x1<x2 时

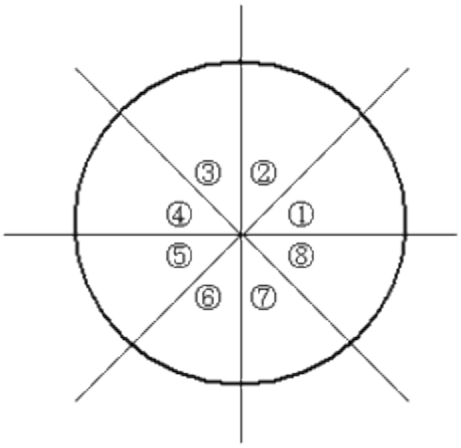
1、输入线段的两个端点坐标：x1，y1，x2，y2；

2、设置象素坐标初值：x=x1，y=y1；

3、设置初始误差判别值：p=2·Δy-Δx；

4、分别计算：Δx=x2-x1、Δy=y2-y1；

5、根据结果选择下一像素点，并进行下一轮循环，直到到达另一端点实现直线的生成。



①④：$x_{i+1}=x_i+1$、x_i-1
$y_{i+1}=y_i$ 或 y_i+1

⑧⑤：$x_{i+1}=x_i+1$、x_i-1
$y_{i+1}=y_i$ 或 y_i-1

②⑦：$y_{i+1}=y_i+1$、y_i-1
$x_{i+1}=x_i$ 或 x_i+1

③⑥：$y_{i+1}=y_i+1$、y_i-1
$x_{i+1}=x_i$ 或 x_i-1

以第一个端点为原点，以第二个端点为方向，则线段方向可分为八种，从原点出发射向八个区。由线段的方向可决定 xi+1 和 yi+1 的变换规律。当线段处于①、④、⑧、⑤区时，以|△x|和|△y|代替前面公式中的△x 和△y，当线段处于②、③、⑥、⑦区时，将公式中的|△x|和|△y|对换。

3. 关键技术（或步骤）实现

```
//使用Bresenham画线算法根据两端点画线
void drawLine(int x0, int y0, int x1, int y1)
{
    int x = x0, y = y0;
```

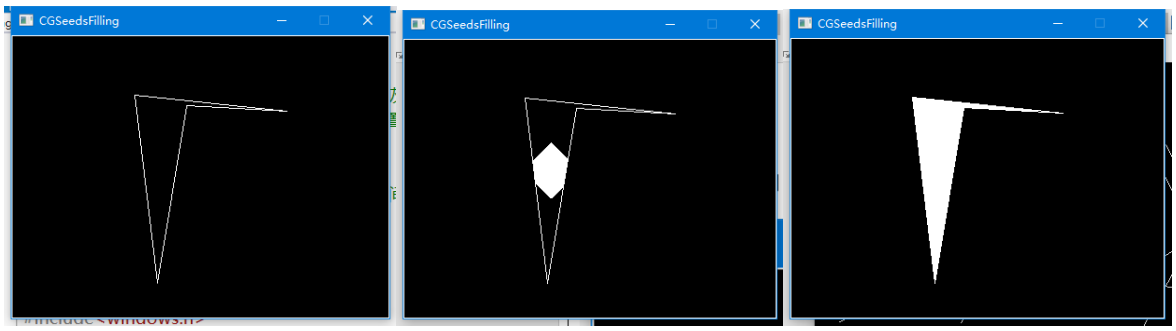
```
int dx = abs(x1 - x0);
int dy = abs(y1 - y0);
int incrementX, incrementY;    //XY增量
bool interchange; //是否交换步长方向
int p;   //2·Δy-Δx

//根据线段方向确定增量正负
if (x1>x0)
    incrementX = 1;
else
    incrementX = -1;

if (y1>y0)
    incrementY = 1;
else
    incrementY = -1;

//根据Δy、Δx大小确定步长方向
if (dy>dx)
{
    int temp = dx;
    dx = dy;
    dy = temp;
    interchange = true;
}
else
    interchange = false;

//循环实现直线的生成
p = 2 * dy - dx;
for (int i = 1; i <= dx; i++)
{
    Sleep(1);
    putpixel(x, y, WHITE);
    if (p >= 0)
    {
        if (interchange == false)
            y = y + incrementY;
        else
            x = x + incrementX;
        p = p - 2 * dx;
    }
    if (interchange == false)
        x = x + incrementX;
```

```
        else
            y = y + incrementY;
        p = p + 2 * dy;
    }
}
```

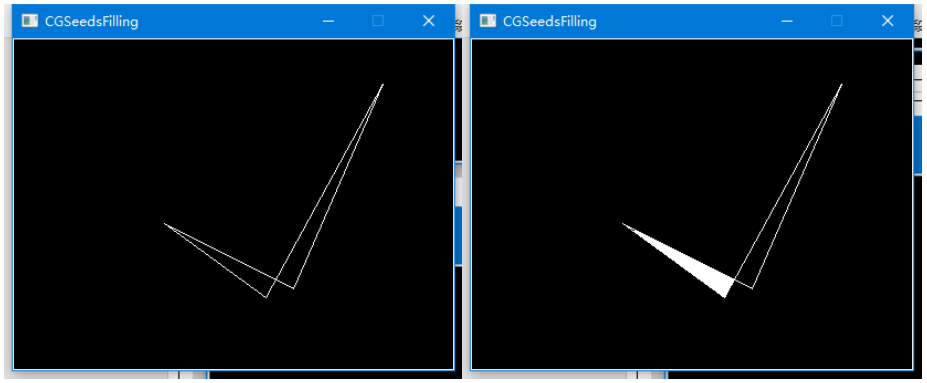
【结论】（结果）：运行结果

Ⅰ. 二维题目 - 1

① 随机生成-单击填充　　填充中　　填充完毕

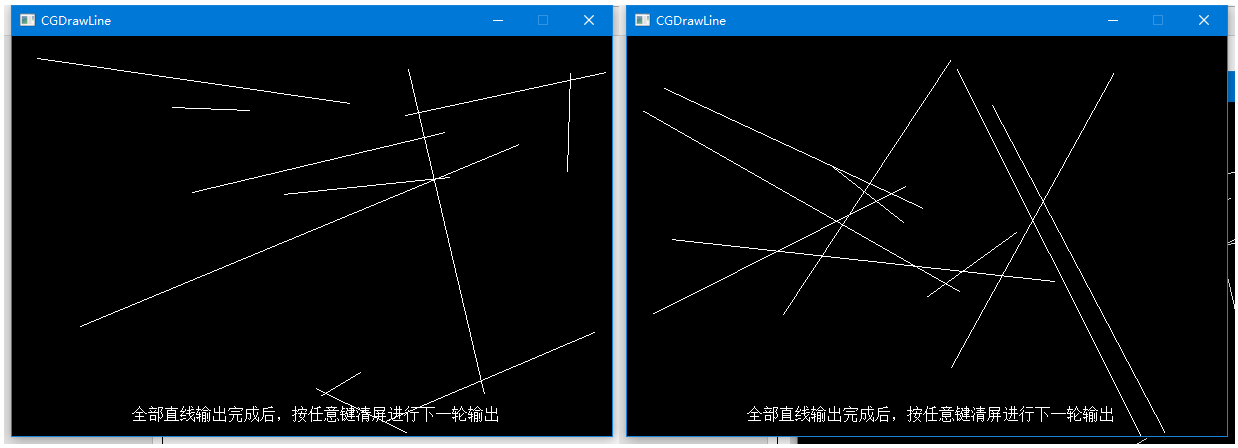


② 随机生成-单击填充　　填充完毕



Ⅱ. 二维题目 - 2

运行后自动开始第一轮动画形式输出，输出完成后按任意键清除已绘制直线重新输出



【小结】：

① 通过两个实验项目，巩固了本学期学习的计算机图形学的基本知识，对部分计算机图形学的原理有了更深入的理解，同时结合实践有效的锻炼了实际编程能力和技巧。

② 在二维题目 1 中，起初，出于实现简单的考虑，使用递归的种子填充算法。但实际调试时，由于像素点过多，递归的调用层级数量级很大，多次导致栈溢出。因此，舍弃递归的方式，通过待填充队列的方式，实现非递归的种子填充算法，大大提高了填充效率。

③ 在二维题目 2 中，开始时参考教学课件的内容进行实现，而忽略了课件算法的前提—0<k<1，导致输出了许多横线（因为无法向负方向增加），其他不为横线的线段也均为同一方向。而后通过对算法的修改，添加了判断步长和增量正负的变量，通过对输入的两端点的判断进行初始化，从而实现了各个方向线段的绘制。

指导教师评语及成绩

【评语】：

成绩：　　指导教师签名：

批阅日期：