第三次计算几何实验报告

一、 实验目标

根据输入的线段，生成相应的梯形图，并能够对给定的点进行查询，找出从该点发出的向上射线所相交的第一条线段。

二、 主要算法

我们采用了计算几何课程讲义中所阐述的梯形图构造算法。

三、 系统设计

本次程序分为两个部分：

1、 程序界面部分（GUI 部分）

2、 算法核心部分（Core 部分）

程序界面部分采用 OpenGL 绘制梯形图和相应的有向无环图。提供了输入线段、查询点、演示过程、文件输入输出、调整视图等功能。

四、 算法部分

Core 是这次试验的内核部分，包含的数据结构有 DCEL 半边图以及查找用的图，算法部分主要是采用 PPT 上的梯形图查找、构建算法。所有算法的操作用一个类 TrapMap 来集成。

数据结构

查找用的图中的节点定义了一个类 Node，Node 含有如下信息：

- 节点的左子节点和右子节点——pLeftChild、pRightChild
- 节点所有父节点，用一个 Array 保存——aParent
- 节点的类型，主要用来区分是点节点、边节点、还是面节点——mNodeType
- 节点的编号，依次给插入的点、插入的边、生成的面编号——mNodeID
- 用于界面绘制用的一些坐标、平移、选择信息

点节点（PointNode）、边节点（EdgeNode）、面节点（FaceNode）都继承了 Node 这个类，分别还保存了对应在 DCEL 图上相应的点、半边、面的指针，面节点还多记录了这个面上面的边的编号、左边界和右边界的 x 坐标。

DCEL 图中的数据结构中，在原有的基础上，点（DCELVertex）中保存了点的类型（用于判断合并），面（DCELFace）保存了由查找用的图对应的节点。

算法

初始化

在一开始需要给 TrapMap 类初始化，主要用于建立一个初始的矩形范围，以后要插入的线段都在这个区域中。初始化的时候在 DCEL 图中建立一个面，面有四个顶点以及四条半边，构成一个内环，此后 DCEL 图维护的时候，DCELFace 中保存的边都是指右上角往左的第一条边，如图 1 所示。对于初始化的一个面，查找图中建立一个面节点，将初始化的边界信息记录下来，并且编号为 0，向上相交的边编号为-1（表示没有插入的边）。

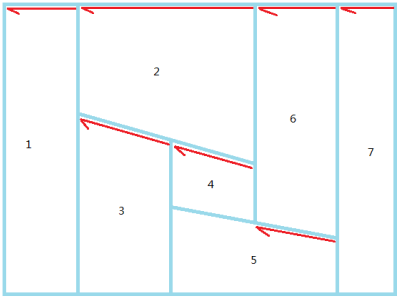


图 1：DCEL 图中面记录的边为右上角往左的第一条边，在示意图中红线表示了每一个面所记录的边

查找

由于构建的时候需要有查找算法，所以先说明查找算法。查找算法的实现是在 TrapMap 类中的方法 FaceNode* TrapMap::SearchArea(double x, double y)实现的，输入的 x、y 表示查找点的坐标值，方法返回的是查找点定位到的区域所对应的面节点。

对于任意一个给定坐标的点，从查找图中自顶向下一次查找，查找每个到每个节点的时候，如果是面节点表示已经定位出查找点所在的面，如果是另外两种节点则分情况往下查找，可以用表 1 说明查找策略。判断一个点在另一个点的左右是比较两者 x 坐标值的大小，根据插入线段是左闭右开的原则来判断查找点在点的左右；判断一个点在一条边的上下，由边的左端点指向右端点构建一条有向边 s1，以及边的左端点指向查找点构建出的第二条有向边 s2，如果 s2 在 s1 的左侧，则表示查找点在边的上面，判断左右可以通过 s1 与 s2 叉乘出来矢量的正负来判断。

节点类型	查找左子节点	查找右子节点
点节点	查找点在点的左边	查找点在点的右边
边节点	查找点在边的上面	查找点在边的下面

表 1：查找过程中中间节点的选择策略

构建查找图

查找图是通过每次插入一条边来生成、维护的。初始化的时候图中有一个面节点，DCEL 图中含有一个面。

插入一条边 s 的时候，首先查找左端点 sv（start vertex 的简称）和右端点 ev（end vertex 的简称）分别在图中所在的区域 ZoneX 和 ZoneY。分两种情况进行处理：

1. **两个端点落于同一个区域（ZoneX = ZoneY）**

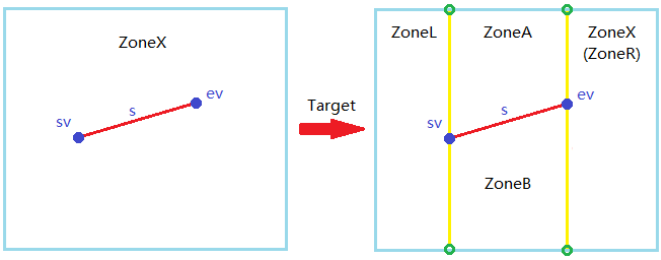


图 2：两个端点落于同一个区域的时候最终目标示意图

如果插入线段两个端点落在同一个区域，则会将区域分成四个子区域，左、右、中上、中下，如图 2 所示。为了实现这种分割，可以分成三步骤，如图 3.A 所示。第一步先用线段左端点的 x 坐标将区域 ZoneX 分割成左区域 ZoneL 和右区域 ZoneX；第二步用线段右端点的 x 坐标将区域 ZoneX 再分割成左区域 ZoneA 和右区域 ZoneX，此时 ZoneX 已经是最终目标的右区域 ZoneR 了；第三步用线段 s 将 ZoneA 分割成上区域 ZoneA 和下区域 ZoneB。注意到在分割过程中会在原来 ZoneX 的上边和下边各添加两个顶点（图 2 中四个绿圈所示），这个在程序里面将其标记成“虚顶点”，将在后面第二种情况中用上。

最终将查找图按照图 3.B 中所示取代替换后即可，注意到替换的时候，需要将 ZoneX 的所有父节点原来所指向它的指针都指向 sv 的节点。

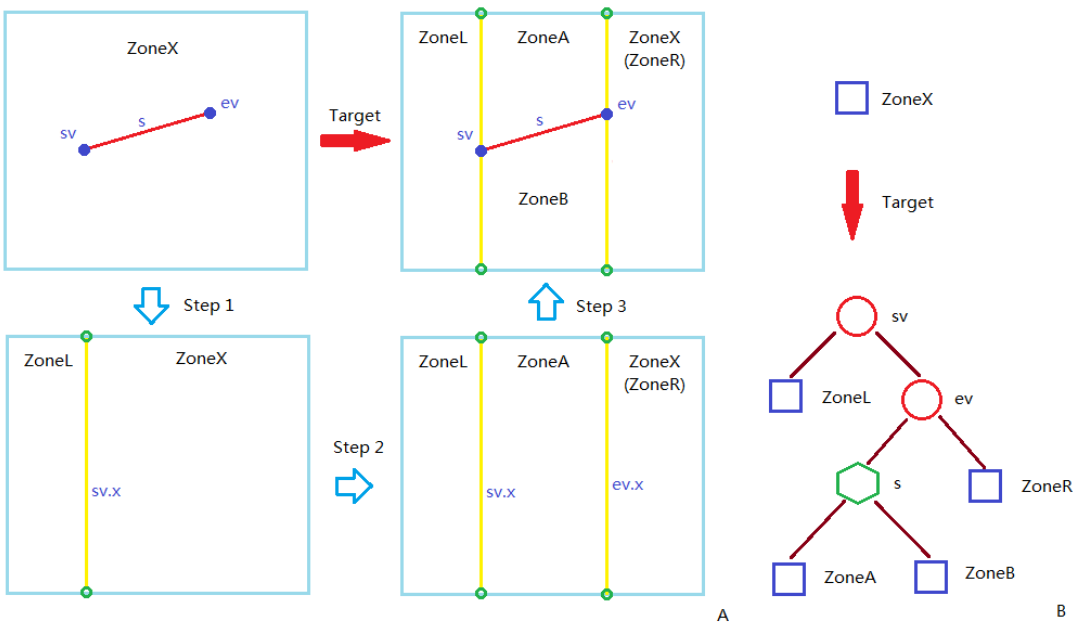


图 3：区域 ZoneX 的分解步骤、查找图替换示意图

2. 两个端点分别落于不同区域（ZoneX != ZoneY）

此时可以分割的时候可以分成三个部分：左端点部分、右端点部分、以及中间穿过的若干个区域的部分。

对于左端点（或右端点）部分的 DCEL 图的分割方法与第一种情况类似，只要注意插入边与区域边界边的那一个交点要标记成“虚顶点”即可，如图 4.A、图 5.A 所示。而查找图的修改参照图 4.B、图 5.B 所示修改即可。

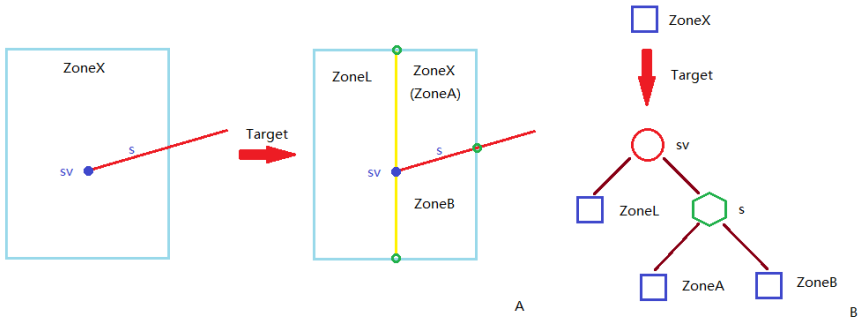


图 4：两个端点分别落于不同区域时，左端点所在区域的分解、查找图的修改示意图

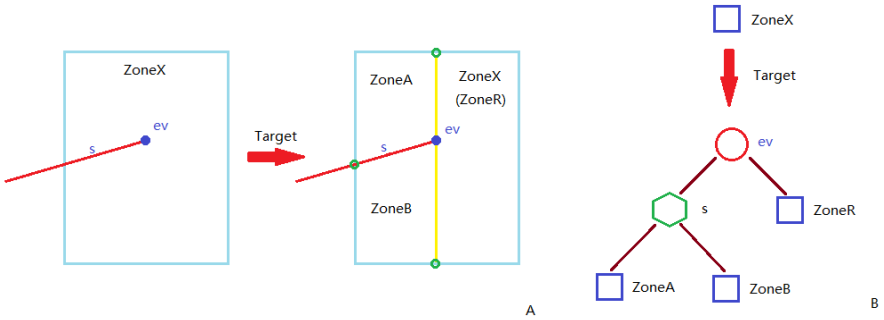


图 5：两个端点分别落于不同区域时，右端点所在区域的分解、查找图的修改示意图

对于中间穿过的部分需要合并相邻区域，如图 6.A 将 DCEL 图中 ZoneX 分割成上部 ZoneA 和下部 ZoneB，查找图的修改如图 6.B 所示。注意到图 6.A 中如果 p1 是虚顶点，则 ZoneX

关于线段 p1、p 的相邻区域需要合并，同理需要对其余 3 个点 p2、q1、q2 进行检查、合并。但是由于算法是从左端点出发，若干个中间区域，到达右端点，所以实际上只需要检查左边的部分并合并即可。合并的时候把冗余边从相关的边、面脱钩，并不直接删除，而是在最后通过释放 DCELMesh 的时候一并释放。

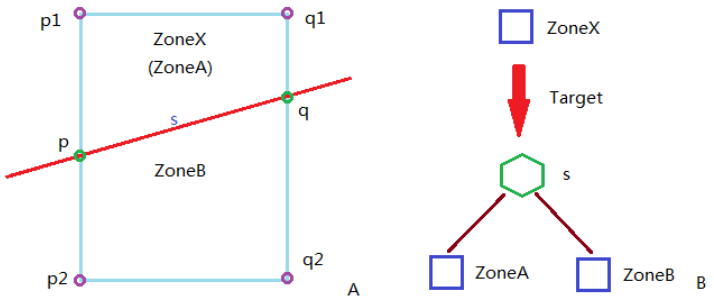
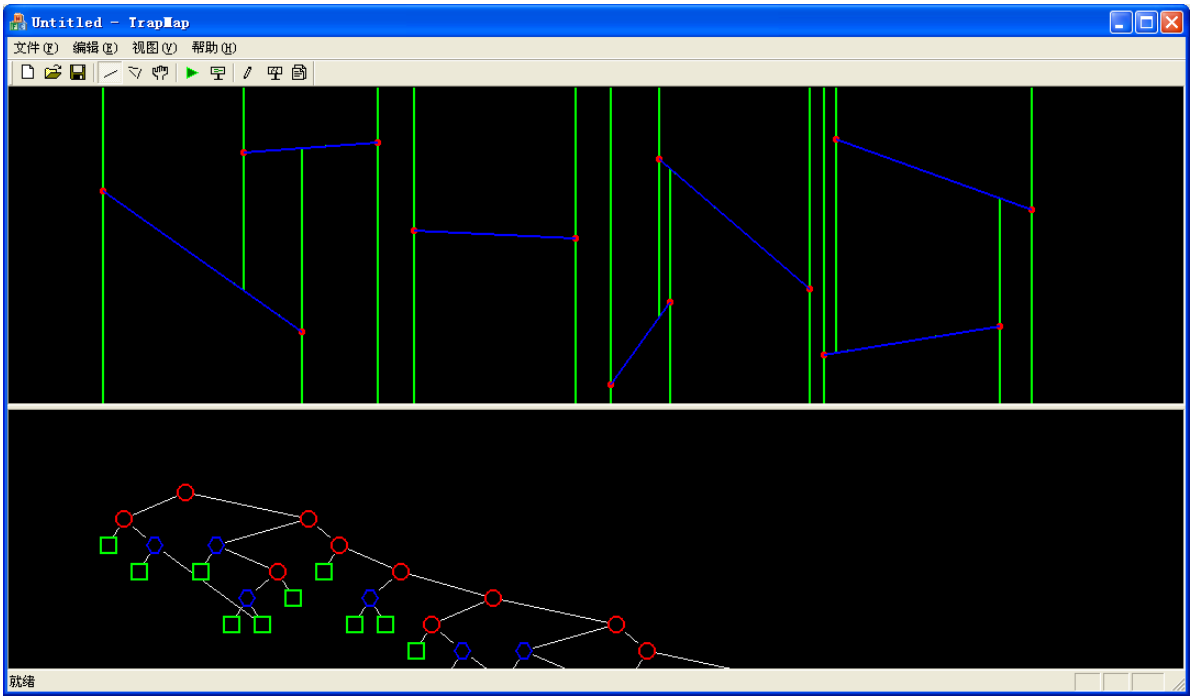


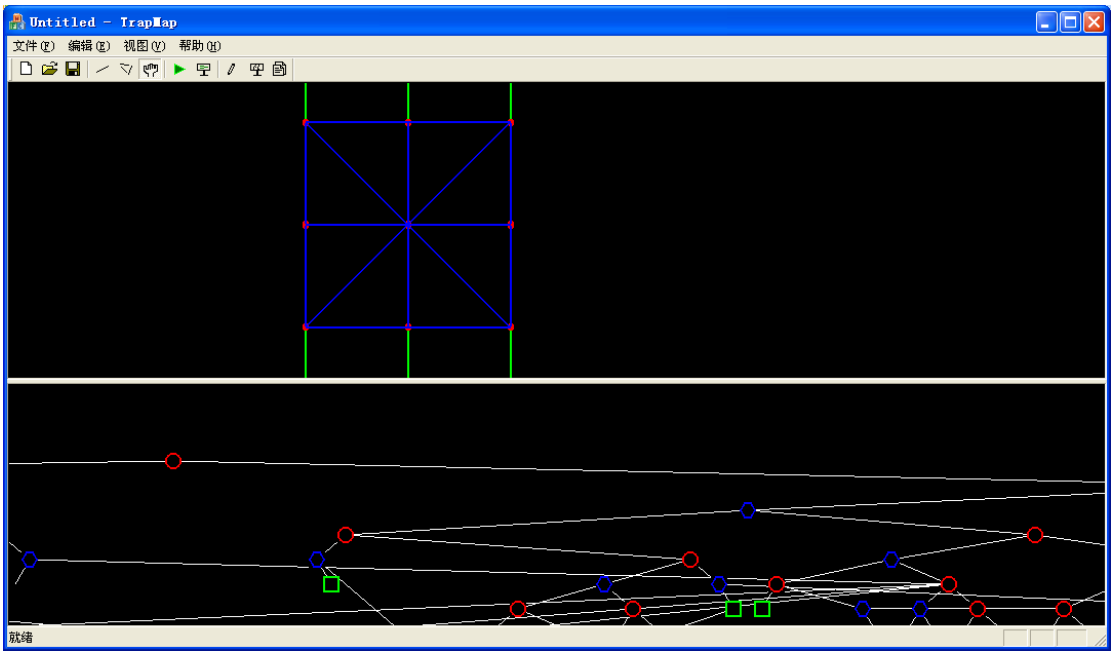
图 6：两个端点分别落于不同区域时，中间穿过多个区域的分解、查找图的修改示意图

五、 实验结果

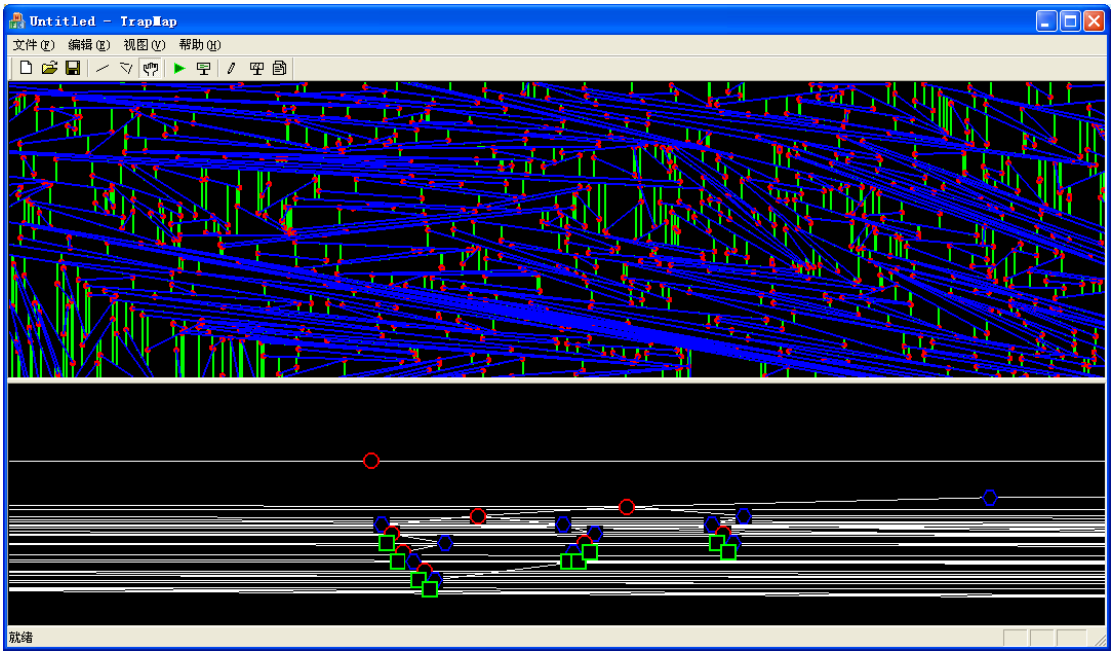
1、 主要界面



2、 退化情况



3、 随机生成线段



六、 主要问题

对于退化情况的处理仍然是本次实验的主要问题之一。本次实验我们采用了将出现退化情况的线段进行微小的变动处理的方法，例如对于端点连接的线段进行收缩处理，对于垂直线段进行旋转处理等等。这种方法依赖于计算的精度，只能用于查询精度比较低的时候。