Kirkpatrick 平面点定位算法演示

尹航（024929）姚海龙（025422）袁国栋（025464）

问题背景

平面点定位算法广泛应用于地理信息系统（GIS）中的地图查询问题，其中 kirkpatrick[1] 提出的基于层次式数据结构的平面点定位算法可以达到理论上的最优解，其查询时间复杂度为 $O(\log n)$，存储复杂度为 $O(n)$，预处理时间复杂度为 $O(n)$。本文中所作的工作是研究该算法，并实现一个算法演示系统。

算法及原理

Kirkpatrick 算法的主要步骤如下所示：

1、通过如下步骤，对一个平面多边形进行转换，使得其平面细分是一个三角剖分，而且外边界是一个三角形。

1）首先计算多边形的凸包。

2）对凸包内的所有图形进行三角剖分。

3）用一个大的三角形包围所得的凸多边形。

4）在三角形顶点和凸多边形顶点间添加边。

2、对三角剖分中的每个三角形进行标号，每个标号都与三角剖分之前包含该三角形的多边形标号相同。

3、层次式的表示：

给定一个 n 个元素的平面细分 S，三角剖分序列$\{T_0, T_1, \cdots, T_k\}$称为 S 的一个层次式表示如果满足以下条件：

1）T_0是增广的 S 的三角剖分。

2）T_{i+1}中的每个三角形涵盖T_i中的常数个三角形$(0 \leqslant i < k)$。

3）T_k包含一个三角形，即最外围的三角形。

4）$k=O(\log n)$。

算法自底向上逐层生成一个层次式的数据结构，以备后续的自顶向下的点定位过程使用。

系统设计

整个系统分为接受用户输入，三角剖分，及基于层次式数据结构的点定位等部分。系统用 Java 编程语言实现，以 Applet 方式运行，由于文件的输入、输出操作受到 Applet 运行权限的限制，要想实现文件的读取和写入操作，需要将 Applet 转换为 Application 运行，该部部分代码已经完成，但考虑到算法演示的重点不在与此，为方便用户在网络上访问，提供 Applet 运行版本，另外，考虑到文件的读写操作可以方便用户使用，另附一个可读写文件的

Application 版本。

➢ 输入

提供文件和鼠标输入两种方式。鼠标输入的可以保存为文件，也可以 Load 文件。文件存储了点边信息，点通过坐标定义，边通过指定两端点 ID 定义。鼠标输入为用户提供了灵活的接口，可以表示任意的 subdivision。

用户输入限制在一个三角形范围内，只能在三角形内出现点，该三角形用作 Kirkpatrick 算法里的外围大三角形。找一个外围大三角形是很简单的操作，但考虑到演示时需要完全显示整个大三角形，如果允许用户在任意位置输入，外围的三角形可能会很大，显示也不方便，所以加了限制。这样可能有个疑问，即对于延伸到无限的 subdivision 怎样表示。这在 Kirkpatrick 算法里似乎不能处理，我们的解释是一来实际应用中需要用来点定位的区域肯定是有限的，二来真有无限的射线也可以单独处理，对顶点求凸包即可，不过这样复杂度上可能会打点折扣。

鼠标输入主要通过单击鼠标完成。先在大三角形内部点击一下加入一个点，这时鼠标再移动可以看到一条随鼠标移动的线，再单击一下即可确定一条边。也可以点击已经加入过的点，这样就是在已有的点上添加边。

注意画线操作不是拖动鼠标来完成的，而是在两个顶点处分别单击。画线时如果不直接单击而是按住拖动了一下则不做操作。因为用户想点一些特殊的位置时如果手一抖就可能点到别处，点击的话定位起来比较方便。如果画线时指定的起点和终点重合，认为是取消画线操作，只保留那一个顶点。

如果新输入的边与已有的边有交点的话会在交点处分成两条边，保持 subdivision 的纯洁性。待输入的边如果过一个已有的点，该点会显示为红色。如果鼠标移到已有的点边附近，点边会显示为绿色，并在右侧的文本框里显示点边的 ID。点的话直接显示数字，边则在前面加大写字母“L”标识。按文本框右的“Delete”按钮可以删除文本框指定的点或边。删除边时只删边，其顶点不删，而删点时会把其相连的所有边删除。

在一个已有的点上拖动鼠标是改变该点的位置，与之相关的边位置也相应改变。如果在允许范围以外释放鼠标键，则取消这步操作。如果拖动到已有的点上，则合并两个点。这个功能很有用，比如想画三条共点的边，但当时手一抖没有共线，反而交出了三个点，就可以通过这种合并使交点保持为一个。

提供了一步撤销功能。在运行其它的 demo 时，我们往往发现不小心点错一个点就必须从头来过，前面如果画过很复杂的结构也没有用了。为此我们提供了保存、删除、修改和撤销功能。撤销后 subdivision 恢复到上次修改以前，这个修改可以是画点、画线、删除、修改甚至 Reset。不过只保留了上一次操作的信息，不能多步撤销，因为那只是多占一些内存，对用户的方便、计算几何的理解、对编程能力的考验都没有多大意义。需要特殊说明的是如果加了一个新点，这时可以连成边，但如果连成边以后再撤销只是撤销“加入边”这一步最近的操作，如果这时对先前的点的位置不满意，要在加入边前直接按撤销。虽然已经提供了很多修改功能，但有时一次操作会引起 subdivision 很大的变化（如新加入了很多交点），这样要用上面的功能恢复这一误操作工作量就很大，而有了撤销功能就很方便了。

precision 框提供的是“就近靠点边”功能。用户往往希望测试一些规则的结构，如一点

关联很多边，或多边共点的情形（这两种区别在输入方式），或者希望从一个已有点连到某边上，如果没有辅助输入的工具，想精确定位是非常困难的。所以我们提供了一个 precision 输入，标识靠近点边多少象素以内则认为它们是同一点或该点可以把边分成两段。就近功能在刚加入点边时显示上保留了精确画图时的痕迹，重绘一下再显示干净的点边。

➢ 三角剖分

三角剖分问题是将多边形进行三角化，这是一个经典的问题；我们采用递归剖分的思路来对多边形进行剖分，算法的复杂度为 O(*n*log*n*)，下面是具体的一些细节。

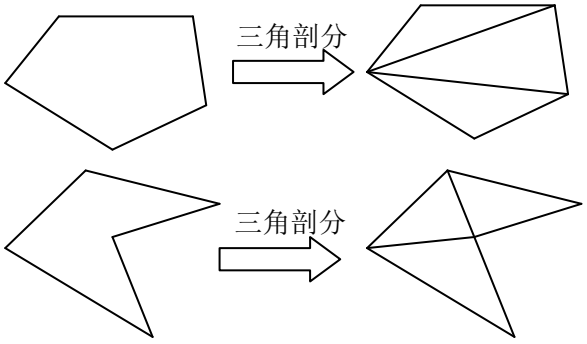

图 1 三角剖分

1 简单多边形的三角剖分

- **基本算法**

在进行简单多边形的三角剖分时，需要对凸多边形和凹多边形进行处理。如图 1 所示，我们在处理时，采用递归剖分的方法来进行处理，具体思路是任选多边形的一条内对角线将其剖分为两个子多边形，若子多边形已经是三角形，停止剖分，否则对子多边形进行递归剖分。递归正确性的证明可以参考有关资料。

- **多边形的凹凸性判断**

多边形凹凸性的判断，可以判断所有顶点的凹凸性来得到。这是一个简单的数学问题，这里不再说明。

- **凹多边形的内对角线计算**

就内对角线而言，凸多边形，任意一条对角线都是内对角线，而凹多边形则不尽然；我们采用如下处理：对凹多边形的一条对角线，判断该对角线与多边形的每一条边是否有交点（端点出外），如果有，则不是内对角线；否则，判断该对角线的任意一点（端点除外）是否在多边形内，如果在的话，这条对角线是内对角线，否则不是，如图 2 所示。

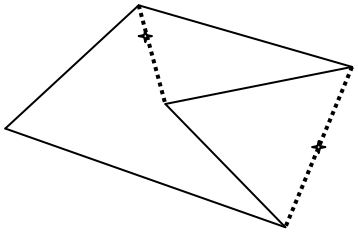

图 2 凹多边形的内对角线判断示意图

- **不同剖分结果的选择**

我们知道多边形的三角剖分是有二义性的，也就是同一个多边形它的剖分结果可能有多种形式，我们可以从中选择一种比较优化的剖分，这些原则有剖分后顶点的数目、剖分后多边形的面积等；

2 重三角化的问题

是在对三角形网格删除顶点后，需要对它进行重三角化，这个步骤是在建立层次型结构时所需要的。我们的处理方法是从需要删除的顶点出发，找到与该点有邻接关系的任意一个三角形，然后根据边邻接关系，构造删除顶点后的多边形。能否构造一个闭合的多边形，依赖于删除顶点是否是边界顶点，在建立层次结构时，删除顶点都在边界三角形的内部，生成

的多边形是闭合的。调用简单多边形的三角剖分算法，可以实现重三角化的问题。

3 加入边界三角形后的三角化问题

为了生成点定位算法的层次结构，需要求解多边形网格的包围三角形，包围三角形和多边形网格之间的区域需要进行三角化，这里涉及三个问题。

- 多边形网格的凸包求解

点集 Q 的凸包(convex hull)是指一个最小凸多边形，满足 Q 中的点或者在多边形边上或者在其内。求解多边形网格凸包的常用算法是 Graham 算法[2]和卷包裹算法[3]，我们采用了 Graham 算法，其具体步骤可以参考相关文献。

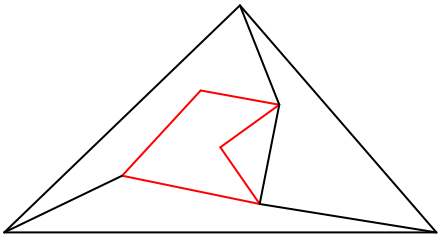

图 3 凸包和边界三角形

- 凸包和边界三角形之间区域的三角化

在此，我们采用如下方法，基于我们三角化实现的递归剖分方法，我们将边界三角形上三个顶点和从凸包上所选择的三个顶点进行连接，如图 3 所示，这样凸包和边界三角形之间的区域首先可以分刻为三个封闭的多边形，调用简单多边形递归剖分方法对其进行三角化。凸包上选择的三个顶点与边界三角形三个顶点相对应，是一个比较繁琐的过程，为了简化实现过程，我们把边界三角形的形状做了固定，是三角形的三个顶点分别和凸包的最左下点，最右下点和凸包的最右上点进行对应，以后如有时间，可以进行优化。

- “洞”的三角化

当多边形网格的边界是凹多边形时或者多边形网格是不连通的情况下，采用这种方法会产生空洞，如图 3 所示，这时我们需要做进一步的处理来将洞进行三角化后，加入新生成的三角网格中。当多边形网格是不连通的情况下的处理，有时间可以进一步进行处理。

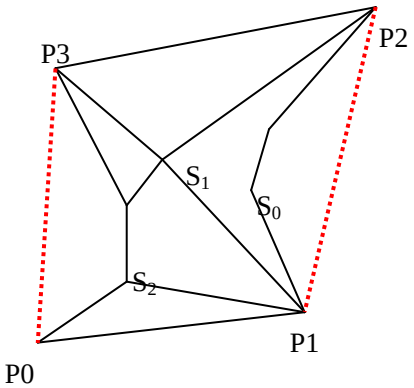

图 4 “洞”的三角化

“洞”是凹多边形的边界和凸包上的边构成的，我们首先从凸包的左下点出发，判断凸包上的边是否是多边形的边，如果不是，将边的起点作为“洞”多边形的起点，边的终点是“洞”多边形的终点，并且同时将这条边作为参考边；然后找与洞的起点相邻接的顶点，根据与参考边最临近的原则，依次找到“洞”上所有顶点。最后调用简单多边形的三角剖分算法将“洞”进行三角化，如图 4 所示。

➢ 层次式结构

层次式数据结构主要依据一个类的定义实现，该类的主要成员变量描述如下所示：

```
public class HierarStruct
{
    //该层次结构所处的级别
    private int level;
```

```
    //点定位到该级别时所处的三角面片的ID号
    private int faceID;
    //自底向上生成层次式数据结构时，在当前级别去除的两个顶点的ID
    private int firstID;
    private int secondID;
    //自顶向下进行点定位查询时，由当前级别查询下一级别对应三角面片ID的哈希表
    private Map downwardMap = new HashMap();
    //当前级别中所有面的集合
    private Vector faceVector;
    //当前级别中所有顶点的集合
    private Vector vertexVector;
}
```

在点定位过程的第一步是建立层次式的数据结构，对原始的多边形结构进行三角剖分后生成的三角面片进行处理，每次去除两个合适的不相邻顶点，然后将生成的多边形进行三角剖分，三角剖分结果由上述层次式数据结构保存，并同时经过三角形的相交测试记录本级新生成的三角形与下一级三角形的对应关系，存入downwardMap成员变量中。该层次式数据结构的生成过程重复进行直到最终生成顶级大三角形，此过程中生成的所有层次式结构HierarStruct的对象存入一个向量（Vector）中以备点定位查询使用。

原始数据结构建立完成之后，点的查询经过自顶向下的定位过程实现。首先测试定位点与顶层大三角形所关联的三个三角形之间的包含关系，由求得的三角形ID借助downwardMap向下一级一级查询，直到最底层的三角形面片为止。最后根据底层三角形三角形面片与多边形的对应关系求得三角形的位置。downwardMap是一个哈希表，记录三角形面片顶点ID间的对应关系。哈希表中的每个成员是一个索引键和之的对应关系对<key, value>，其中key和value都是对象。在实现中，考虑到问题的需要，key用的是一个Integer对象，其中包含了上一层三角面片的ID值；value是一个Vector对象，其中包含了key索引的三角面片对应的下一层三角面片集合的ID。由此可见，downwardMap完成了层次式结构中自顶向下的逐层索引关系，其中每个成员都是一个一对多或一对一的关系，点的定位借助downwardMap的引导，自顶向下逐层测试点在哪个三角形中，最终完成点的定位。该层次式结构的生成如图5 所示。

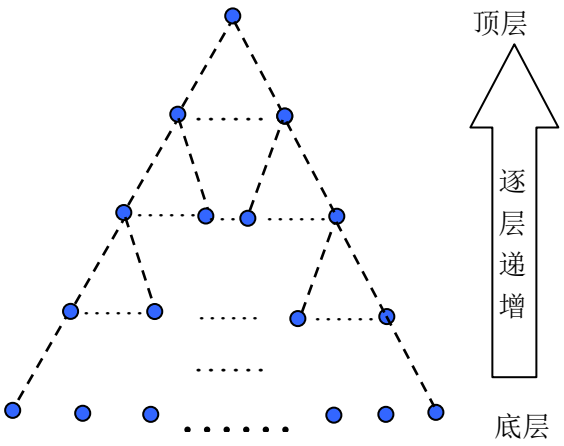

图 5 层次式结构的生成

难点及对策

➢ 用户输入

怎样表达 subdivision。这个开始就设计了很久，怎样既方便又可以涵盖用户需要的所有情况。

允许灵活的输入方式。这样有时需要大量的预处理。

边之间的关系有相离、相交、共线，每新加入一条边都要进行一系列的判断。还有一些特殊的输入都要处理。比如重合、包含等关系。

根据任意输入的点边生成 DCEL 结构。

用户的输入不会有面信息，而后面的处理又一定要用，所以要根据点和边把面及各种关系计算出来。如果二多边形内接，生成面表很麻烦，不退化的意义上说有非简单多边形。

就近靠近点线。DCEL 结构的及时更改。靠近的判断并要对判断负责。

删除与修改。见过的很多 demo 都是一个点点错就必须全清空重来，太痛苦了。所以提供了各种后悔药。对于一般的点定位算法，这种功能为演示 update 部分提供了方便。

➢ 界面实现

为防止图形显示中的抖动问题，程序中采用了 BufferedImage 来存取整个图形，首先将图形画到 BufferedImage 缓冲区中，然后将其全部画到画布上，这样便不会有闪烁产生。另外，使用画布（canvas）可以摆脱图形显示对 Applet 的依赖，使界面可以方便的转换为 Application 程序。

➢ 调试中遇到的问题

程序调试过程中，遇到了一些问题，其中主要是顶点、面存储访问使用的 ID 问题，在程序中，顶点和面都是用 Vector 存储的。最初采用顶点和面的 ID 进行索引，但是后来由于层次式数据结构需要新面 ID 在原有的基础上累加，便产生了索引问题。改问题耗费了将近一天的时间来调试，主要经验教训是在对象 ID 值和存储索引值可能不一致的情况下，不能为了减少数据存储量，将点、面的 ID 和数据存储的索引值混用。

任务分工

袁国栋 —— 基本数据结构与三角剖分算法
姚海龙 —— 层次式结构与总体设计
尹航 —— 接受用户输入并处理成 DCEL 结构

使用说明

先通过鼠标点击在画板上输入一个 subdivision，然后按“PreProcess”生成 DCEL 结构并做三角剖分、建立层次式数据结构，此时便完成了原始数据结构的创建。点击“startPL”按钮，可以开始点定位过程。这是在画板上任何区域单击鼠标，便可以得到相应点定位的结果。点所处位置的多边形以黄色高亮。这时就可以按“Up”和“Down”按钮或直接输入层号来显示各层的信息及相应点所处的面的信息，同时也会将当前层生成上一层次数据结构时所去除的点及生成的新多边形以不同颜色显示。点击“endPL”按钮或“reset”按钮可以终止当前 subdivision 的点定位过程，重新输入新的 subdivision 进行新的一轮点定位过程演示。系统界面如图 6 所示。

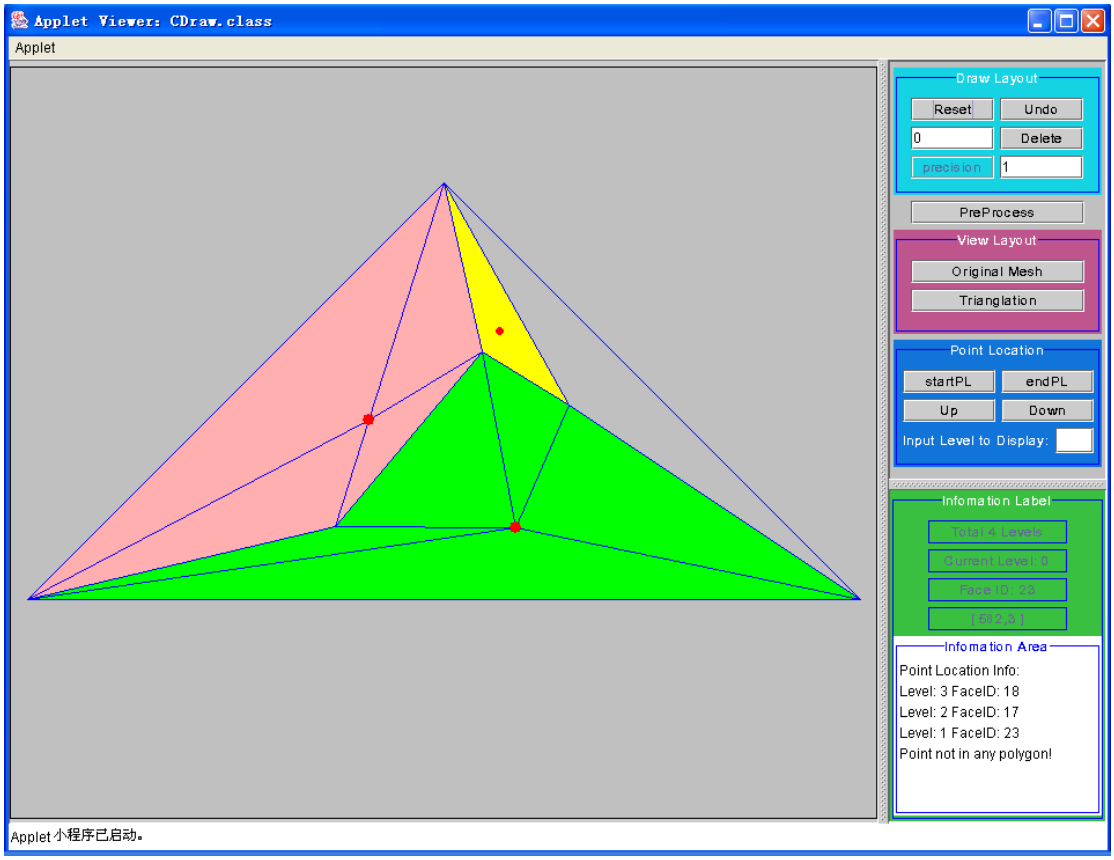

图 6 系统界面

展望

对于多边形网格不连通的情况，需要进一步进行处理，当然这是一个难点 。

如果能够采用三维图形来显示层次式数据结构，是一个非常不错的手段；近年来，Java 3D 的 API 虽然受到 Java 运行速度的影响，但是也取得的比较广泛的应用，以后若有时间，可以做一些近一步的工作。

参考文献

[1] D. G. Kirkpatrick. Optimal search in planar subdivisions, SIAM Journal on Computing, 12:28-35, 1983.

[2] R. L. Gramham, “An Efficient Algorithm for Determining the convex hull of a finite planar set.” Inform. Process. Lett., 1:132-133, 1972

[3] 卢开澄，计算机算法导引，清华大学出版社，2002 年 8 月

[4] http://www.gameres.com/Articles/Program/Abstract/Geometry.htm