第一次实验报告

一、 实验目标

我们的实验目标是对平面点集进行 Delaunay 三角剖分。

二、 主要算法

我们采用的算法是主要参考 Guibas 和 Stolfi 的基于分治策略的算法。

三、 系统设计

我们的程序分为前端和后端。

前端程序负责 GUI 界面，点集的读取和保存，三角剖分的绘制，动画演示等方面的功能。

后端程序负责实现三角化算法。我们将它封装在一个 dll 中。前端程序通过加载这个库来调用其中的函数。

1、 前端程序

前端程序基于 MFC 实现。其中动画演示部分采用多线程实现。

2、 后端程序

后端程序使用 C++语言实现。其中加入了修改后的 Geoff Leach 的 C 语言代码。

数据结构：这次实验中用来表达三角剖分结果的数据结构是 DCEL 结构。而原来的代码中所使用的是 QuadEdge 结构。DCEL 结构的好处是查询比较方便，它的每条边都被分成两条互逆的半边。

程序实现：原来的代码使用静态变量，不利于面向对象编程以及多线程实现；并且为了要演示过程，我们需要加入回调结构来捕获每一次 divide 操作的时刻。

前端程序与后端程序的交互：

后端程序向前端程序提供几种接口：

1、初始化顶点数组，前端程序调用接口

```
void *initialization(int n, double x[], double y[]);
```

来告诉后端程序初始化数据结构。

2、与之对应，前端程序调用接口

```
void destruction(void *mesh);
```

告诉后端程序释放数据结构和所分配的内存。

3、前端程序调用接口

```
void getTriangularMesh(void* mesh, void (*fn)(void*), void* arg);
```

来告诉后端程序进行三角化。

4、前端程序负责接受用户操作，并调用接口

```
void *getPointInFace(void* mesh, double x, double y, int& v0, int& v1, int& v2);
```

来查询点位于哪个面上，或者

```
void getIntersectEdge(void *mesh, double x0, double y0, double x1, double y1, int* edges, int len);
```

来查询和线段相交的有哪些边。

四、 实验结果

根据测试，这个程序能够完成基本的 Delaunay 三角化操作，并且能够对三角剖分的结果进行查询，例如查询点的位置，和线段的相交情况等等。

而且这个程序提供了用光标输入点、存取文件和随机生成点的

功能，方便进行演示和测试。

五、 遇到问题

实验过程中有很多简单的细节问题，这里不再赘述了。下面主要讲一下退化情况：

1、 三个或以上点共线的情况。这个主要在 merge 的时候考虑，如果这几个点能够和其他的点构成三角形的话就剖分这几个点，否则不考虑这些点，以避免产生面积为零的三角形。

2、 两个或以上点的 x 坐标相同的情况。因为在分治策略的 divide 过程中，我们是按照 x 坐标进行排序的，如果遇到 x 坐标相同的情况，就按照 y 坐标进行排序。这样可以解决 divide 过程中的问题。但是，在 merge 过程中，我们没有考虑这种情况，因此某些情况下可能会出现一些问题，以致使结果出错，这个问题有待进一步解决。

六、 参考文献

主要参考的文献为

Guibas, L. and Stolfi, J., "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", ACT TOG, 4(2), April, 1985.

关于 DCEL 数据结构，我们主要参考 Ryan Holmes 的描述，详见 http://www.holmes3d.net/graphics/dcel 。

我们还借鉴了一些 Geoff Leach 的 C 语言代码，见 dct.tar.gz。