Go2 系 统 设 计 说 明 书

作品名:Go2

赛题组类:A 类赛题

赛题名称:矢量图形（line 和 circle）在非自交多边形边界中的裁剪显示

作者:于留传 ***

指导老师:江 涛

Email:*****

From:山东科技大学测绘学院遥感系

2015 年 4 月 13 日

目录

1. 需求分析

1) 读取 xml 文件，正确解析图形，支持读取大文件。

2) 正确显示图形。

3) 判断矢量图形 line 与非自交多边形边界的位置关系。

4) 判断矢量图形 circle 与非自交多边形边界的位置关系。

5) 计算与非自交多边形边界相交的矢量图形的交点。

6) 判断与非自交多边形边界相交的矢量图形落在多边形内部的部分。

7) 尽可能减少时间和内存开销。

2. 设计理念

在满足功能需求的基础上，尽可能满足非功能性需求，同时提高算法的普适性和软件的稳定性。

3.设计思路

1) 小 xml 文件(TestData1.xml)的处理,可以采用普通的方式,例如 FILE *fp, fstearm...; (C/C++)。然而处理大 xml 文件（TestData2.xml）时，再使用这种普通方式处理，I/O 的频繁操作将耗费 CPU 利用率与内存。所以统一采用内存映射文件的方式处理。

2) 图形的显示，因为屏幕的物理坐标原点在左上角，而需要显示的逻辑坐标原点在左下角，所以需要进行一个变换，即 X = Xo，Y = 窗口高度 - Yo。

3) 判断 line 与多边形边界的位置关系，即判断 line 在多边形内部、外部、还是相交，采用 line 与多边形每条边求交点的方式，根据“交点检验法”判断线与多边形的位置关系。

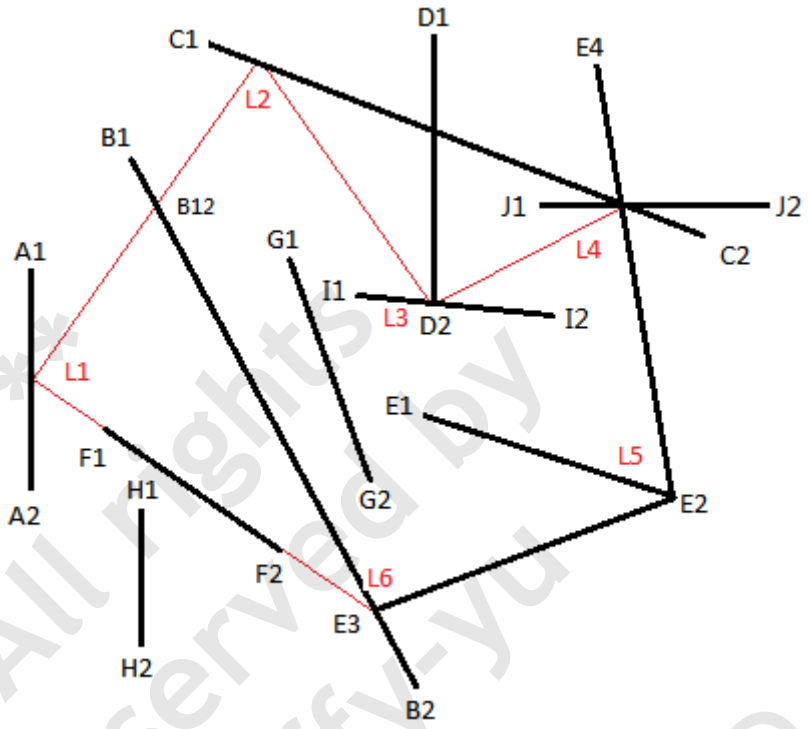


图 1

如上图所示：L1、L2、L3、L4、L5、L6 为多边形顶点。剩下的是线段。下面对每条线段进行分析。

a) A1A2

与多边形有多个交点，其中在线段上的交点有 1 个，该交点亦是多边形的顶点。线段并未穿过交点，即线段的一部分不存在 L6L1L2 组成的多边形的内角的范围内。因为线段 A1A2 上的点在多边形外部，所以 A1A2 在多边形外部。

b) B1B2

与多边形有多个交点，其中在线段上的交点有 2 个，其中一个为多边

形的顶点 L6，并且线段穿过该交点。在线段两端点之间有 1 个交点，1 个穿过顶点的交点。所以 B1B2 与多边形相交。

c) C1C2

与多边形有多个交点，其中在线段上的交点有 2 个，并且都是多边形的顶点，同时线段并未穿过交点。所以同 A1A2 一样，C1C2 在多边形外部。

d) D1D2

与多边形有多个交点，其中在线段上的交点 1 个，该交点既是多边形的顶点，又是线段的端点，但是线段并不穿过顶点。又因为线段上的点在多边形外部，所以 D1D2 在多边形外部。

e) E1E2

与多边形有多个交点，其中在线段上的交点为 1 个，该交点既是多边形的顶点，又是线段的端点，但是线段并不穿过顶点。因为线段上的点在多边形内部，所以 E1E2 在多边形内部。

f) E2E3

与多边形有多个交点，其中在线段上的交点为 2 个，并且该线段是多边形的一条边。交点并不穿过顶点。过线段上一点 E23 做 Y 轴的平行线，与 L3L4、L5L6 边的交点，组成一条线段。该线段上有两个与多边形相交的点，按照“交点检验法”，E23 的一个方向上有 1 个交点，另一个方向上无交点。即 E23 与多边形的交点个数是“奇数偶数”的组合。所以 E23 这个点在多边形的边上，为了便于处理，把这种情况归为线段在多边形外部的情况。所以 E2E3 在多边形的外部。

g) E2E4

与多边形有多个交点，其中线段上的交点有 2 个，该线段的一部分与多边形的一条边重合。所以 E2E4 的部分点在多边形的边上，一部分在多边形的外部。在多边形边上的部分处理同 E2E3，不在多边形边上的部分，即在多边形外部的部分，可得出线段在多边形外部。为了方便处理，把该线段划分到多边形的外部。所以 E2E4 在多边形外部。

h) F1F2

与多边形有多个交点，其中线段上无交点，该线段与多边形的一条边部分重合。按照 E2E3 的处理方式，F1F2 在多边形的外部。

这样 E2E3、E2E4、F1F2 就都属于在多边形的外部。

i) G1G2

与多边形有多个交点，其中线段上无交点。线段上的任意一点在多边形内部，所以该线段在多边形内部。

j) H1H2

与多边形有多个交点，其中线段上无交点。线段上的任意一点在多边形外部，所以该线段在多边形外部。

k) I1I2

与多边形有多个交点，其中线段上有 1 个交点，该交点亦是多边形的顶点。线段并不穿过该顶点。由于线段上的点在多边形内部，所以该线段在多边形内部。

l) J1J2

与多边形有多个交点，其中线段上有一个交点，该交点亦是多边形的

顶点。线段并不穿过该顶点。由于线段上的点在多边形外部，所以该线段在多边形外部。

综上所述：在多边形内部的线段为 E1E2、G1G2、I1I2，共 3 条。在多边形外部的线段为 A1A2、C1C2、D1D2、E2E3、E2E4、F1F2、H1H2、J1J2，共 8 条。与多边形相交的线段为 B1B2，共 1 条。

所以判别线段与多边形的位置关系的思路是：线段分别与多边形的每条边计算交点，去掉不是在线段上的点。

（1）如果线段端点之间没有交点，或者只存在不穿过顶点的交点，那么该线段在多边形内部或者外部。否则线段与多边形相交。

（2）对于不相交的情况，选择线段上一个不是多边形顶点也不是 line 端点的点，判断该点是否落在多边形的内部。

（3）如果该点两个方向上各有奇数个交点（若交点是多边形的顶点，穿过的计数 1 个，不穿过的计数 0），那么该点在多边形内部，所以线段在多边形内部，否则线段在多边形内部。

4）判断 circle 与多边形边界的位置关系，即判断 circle 在多边形内部、外部、还是相交，采用 line 与多边形每条边求交点的方式，根据“交点检验法”判断线与多边形的位置关系。

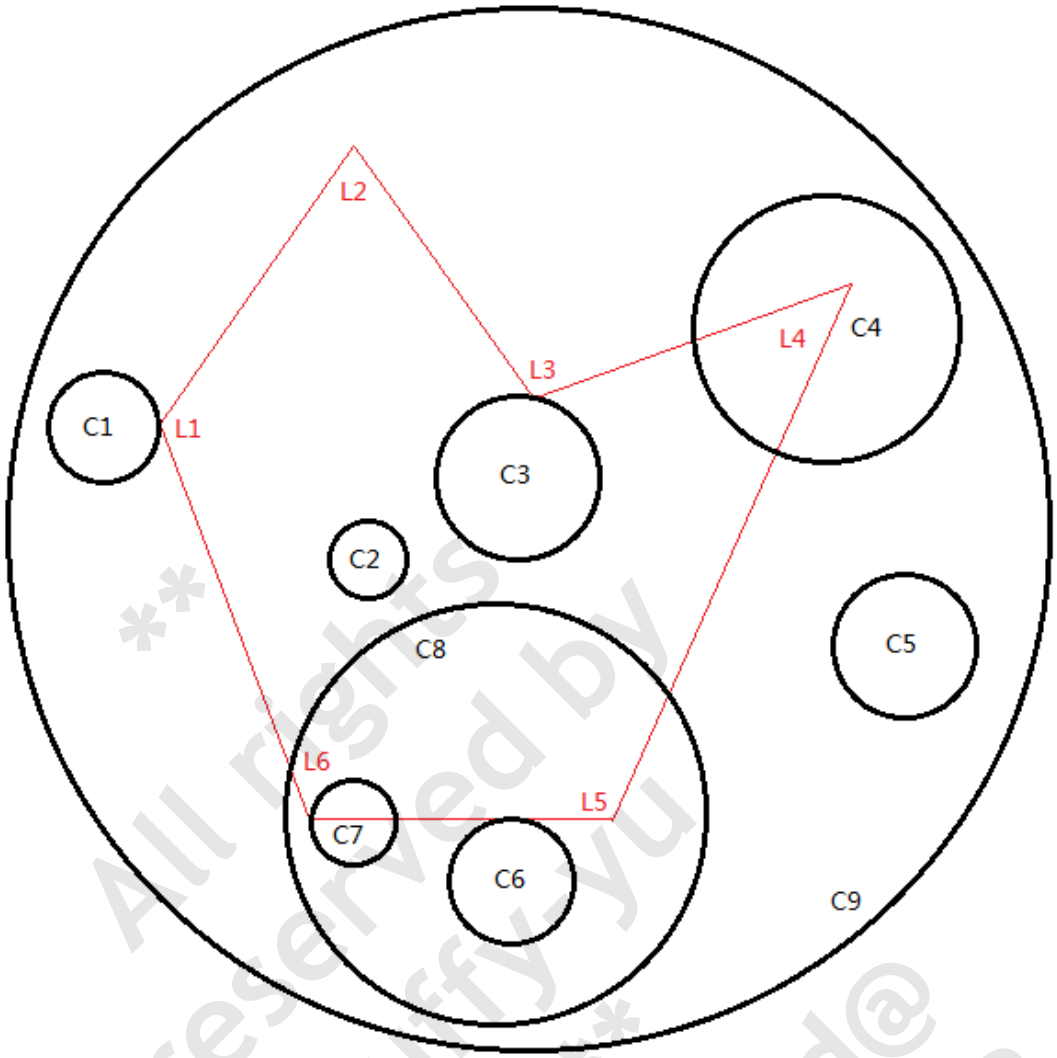


图 2

如上图所示：L1、L2、L3、L4、L5、L6 为多边形顶点。剩下的是圆。下面对每个圆进行分析。

a) C1

与多边形有一个交点，并且该交点为多边形的一个顶点，圆弧并不穿过顶点（圆在该点的切线不经过 L6L1L2 组成的内角区域）。因为圆心在多边形外部，所以圆在多边形外部。

b) C2

与多边形没有交点。因为圆心在多边形的内部，所以圆在多边形的内

部。

c) C3

与多边形有一个交点，并且该交点为多边形的一个顶点，圆弧并不穿过顶点。

因为圆心在多边形的内部，所以圆在多边形内部。

d) C4

与多边形有两个交点，并且都是穿过边的交点。所以圆与多边形相交。

e) C5

与多边形没有交点。因为圆心在多边形的外部，所以圆在多边形的外部。

f) C6

与多边形有一个交点，交点在多边形的一条边上，并且圆弧并不穿过该边。因为圆心在多边形的外部，所以圆在多边形的外部。

g) C7

与多边形有两个交点，其中一个交点是多边形的顶点，另一个交点在边上。并且圆弧穿过顶点(过顶点的切线穿过L5L6L1组成的内角区域)，同时穿过边。所以圆与多边形相交。

h) C8

同 C4，与多边形有两个交点，并且都是穿过边的交点。所以圆与多边形相交。

i) C9

与多边形没有交点，并且圆心到各边的距离都不大于半径。所以圆在

多边形外部。

综上所述：在多边形内部的圆有 C2、C3，共两个。在多边形外部的圆有 C1、C5、C6、C8，共 4 个。与多边形相交的圆有 C4、C7、C8 共两个。

所以判别圆与多边形的位置关系的思路是：首先假设圆包围多边形，如果圆心到直线的距离大于半径，则假设不成立。然后圆分别与多边形的边求交点，如果交点落在边上（不包括顶点），那么假设亦不成立。

（4）当计算出圆与多边形的所有的交点之后，如果假设不成立，则圆包括多边形，即圆在多边形的外部。

（5）当无交点时，圆在多边形的内部或者外部，此时根据圆心的位置，判断圆与多边形的位置关系。如果圆心在多边形的外部，则圆在多边形的外部；如果圆心在多边形的内部，则圆在多边形的内部。

（6）当有交点时，若交点中包含穿过边的交点或者穿过顶点的交点，那么圆与多边形相交。否则，即交点中不包含穿过边的交点或者穿过顶点的交点，那么圆与多边形不相交。再根据圆心的位置判断圆与多边形的位置关系，如果圆心在多边形的外部，则圆在多边形的外部；如果圆心在多边形的内部，则圆在多边形的内部。

5）计算 line 与多边形裁剪后的结果。

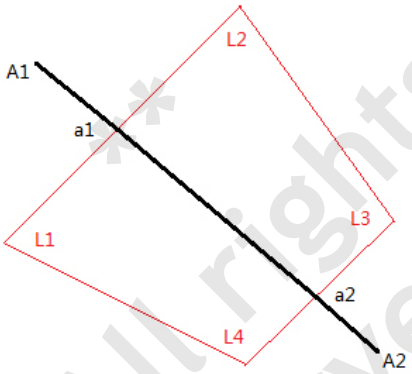
（1）裁剪后的结果不包括在多边形外部的线段。

（2）裁剪后的结果包括在多边形内部的线段。

（3）对于与多边形相交的线段，对线段的端点及所有交点进行排序，并且去除重复的点。然后把线段分成整数小段，判断小段是否在多边形的内部，思路是取小段 AB 上的一点 P，然后过该点做 Y 轴的平行

线（可以是任意线，平行于 Y 轴可方便计算），求出该直线与多边形的所有交点，然后利用交点检验法判断 P 点是否在多边形的内部。如果 P 点在多边形的内部，则 AB 在多边形内部，即裁剪后的结果包含 AB 小段。如果 P 点在多边形的外部，则 AB 在多边形内部，即裁剪后的结果不包含 AB 小段。

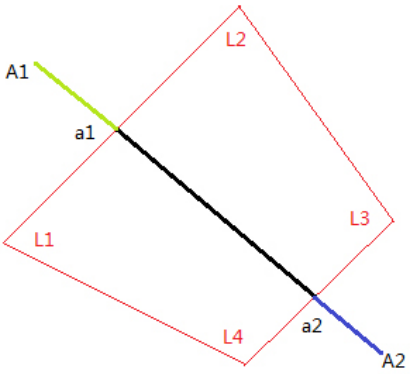
（4）下面演示（3）中的过程。



第一步：

线段端点与多边形的交点排序。

A1->a1->a2->A2



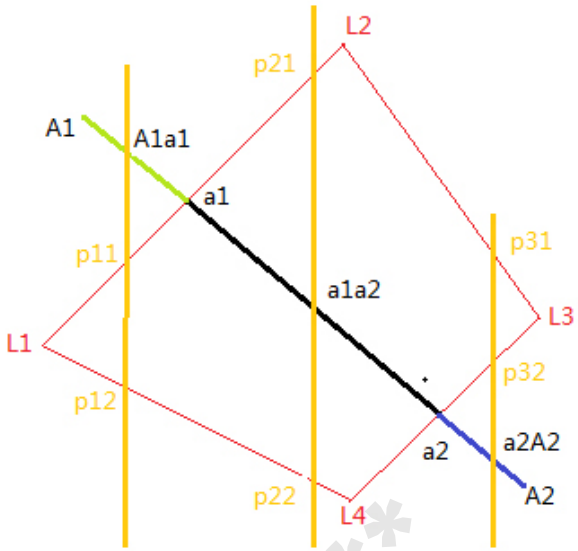
第二步：

把线段分成整数小段。

A1-A2分成：

A1-a1、a1-a2、a2-A2

图 3



第三步：

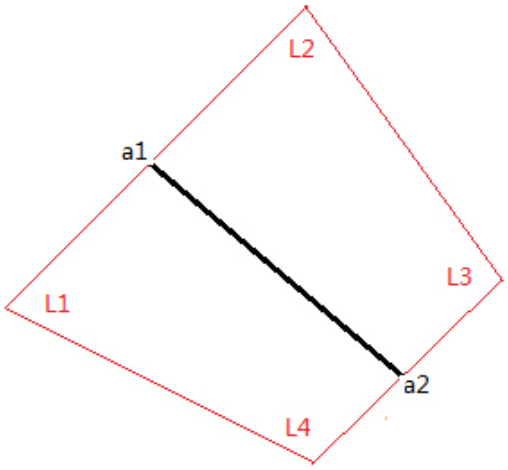
判断小线段是否在多边形内部。

(1)过A1-a1线段中点A1a1作Y轴的平行线，与多边形相交于p11、p12。根据交点检验法，A1a1向下方向有2(偶数)个交点，A1a1向上有0(偶数)个交点，所以A1a1在多边形的外部，所以A1-a1在多边形的外部。

(2)过a1-a2线段中点a1a2作Y轴的平行线，与多边形相交于p21、p22。根据交点检验法，a1a2向下方向有1(奇数)个交点，a1a2向上有1(奇数)个交点，所以a1a2在多边形的内部，所以a1-a2在多边形的内部。

(3)过a2-A2线段中点a2A2作Y轴的平行线，与多边形相交于p31、p32。根据交点检验法，a2A2向下方向有0(偶数)个交点，a1A2向上有2(偶数)个交点，所以a2A2在多边形的外部，所以a2-A2在多边形的外部。

综上所述：a1-a2在多边形内部。



第四步：

在多边形内部的小线段的两端点作为裁剪后的结果输出。

输出a1、a2。

裁剪完成。

图 4

6）计算 line 与多边形裁剪后的结果。

（1）裁剪后的结果不包括在多边形外部的圆。

（2）裁剪后的结果包括在多边形内部的圆。

（3）对于与多边形相交的圆，对交点进行顺时针排序，因为多边形的顶点是按照顺时针的方式存放的，所以圆与多边形的交点也是按照顺时针的顺序存放，不需要再次排序，只需要去除重复（重复即相邻

的两个交点相同)。然后判断相邻的两个交点间的圆弧是否在多边形内部。思路是判断圆弧的中点是否在多边形的内部。

a) 关于圆弧中点的计算。两交点两线为圆的一条弦，圆弧中点与圆心的连线与该弦垂直，满足这样条件的交点有两个。交点的选择依据弦的前进方向判断，因为交点是顺时针存放的，所以交点在行进方向的左手边。按照下面图中的方法选择圆弧中点。

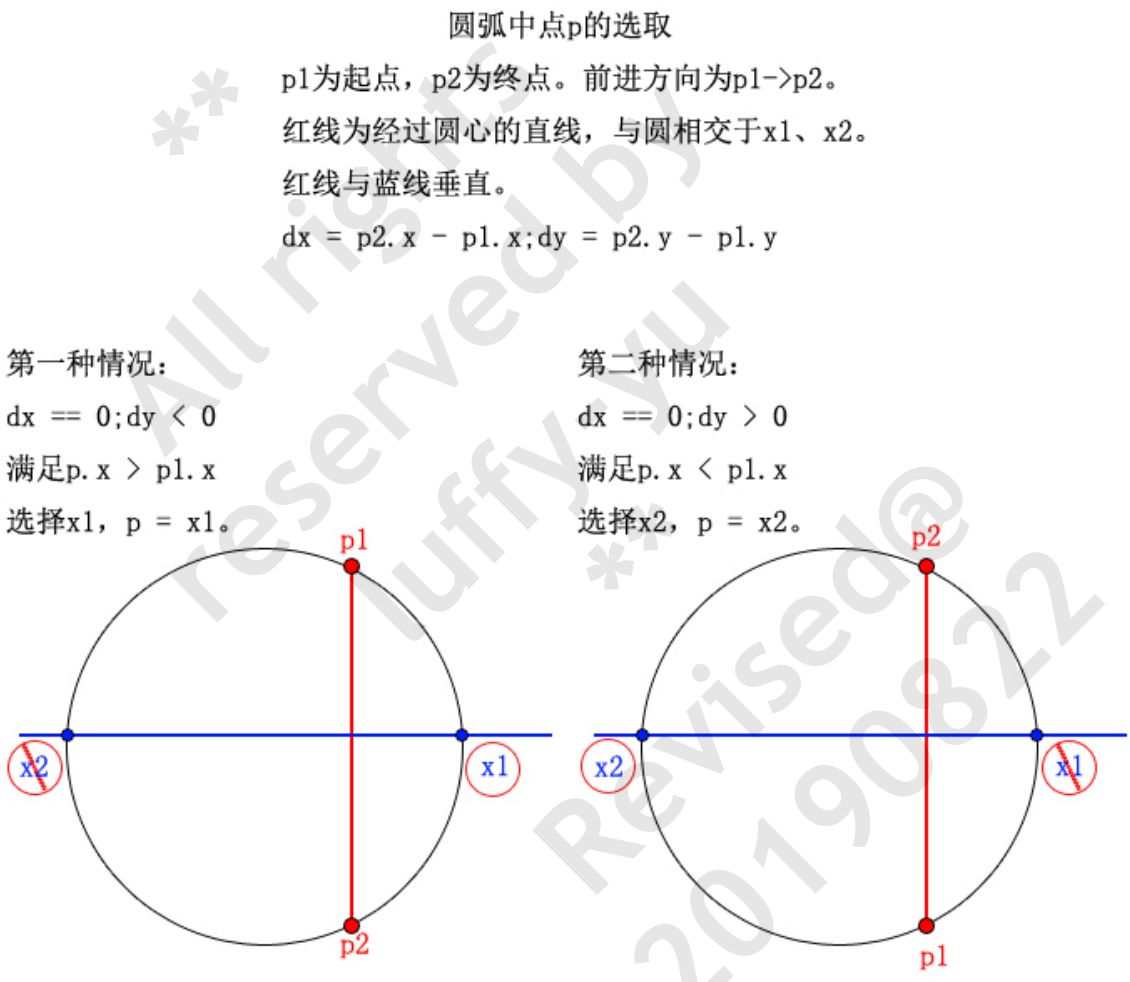


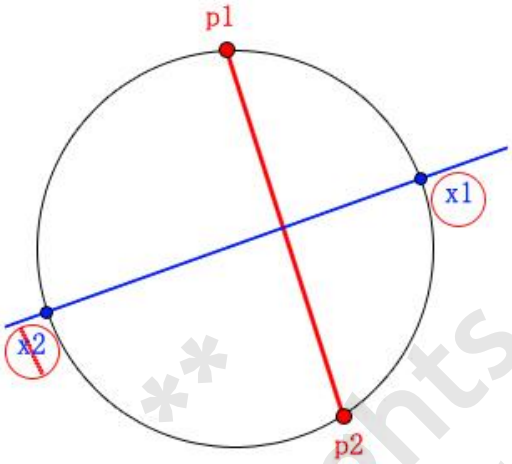
图 5

第三种情况：

dx > 0;dy <= 0

满足p.x > p1.x

选择x1，p = x1。



第四种情况：

dx > 0;dy > 0

满足p.x < p2.x

选择x2，p = x2。

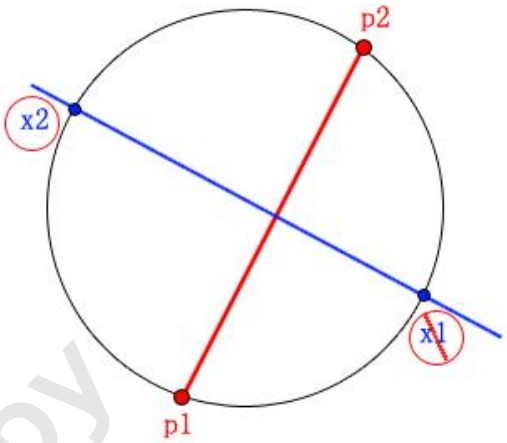


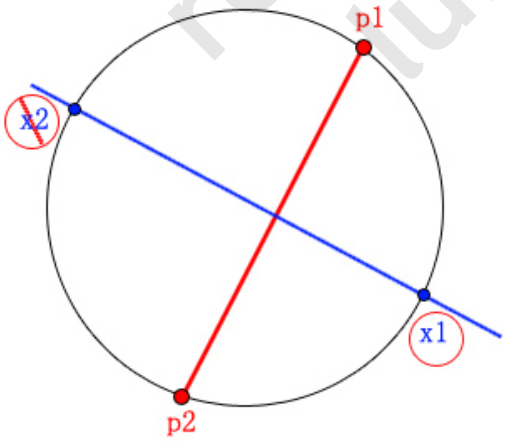
图 6

第五种情况：

dx < 0;dy <= 0

满足p.x > p2.x

选择x1，p = x1。



第六种情况：

dx < 0;dy > 0

满足p.x < p1.x

选择x2，p = x2

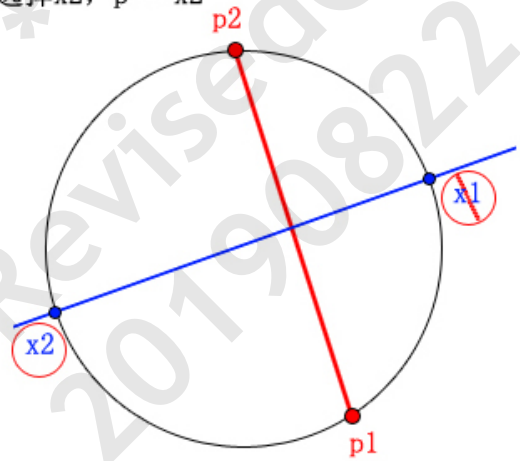


图 7

b) 关于圆弧中点与多边形的位置关系的判断。方法是过该点作 Y 轴的平行线。接下来的步骤就和判断小线段是否在多边形内部的步骤一样。

4.算法设计

4.1 line 在非自交多边形边界中的裁剪

(一) 从所有的 line 中取出一条 line，若无 line 可取，转到(九)。

(二) 计算 line 与多边形的交点，如果交点是多边形顶点，判断 line 是否穿过该顶点，若穿过，则标记该交点为穿过，否则标记该交点为经过。

(三) 对交点进行排序，去除重复点。

(四) 若 line 两端点间存在不是经过端点的交点，那么 line 与多边形相交，转到(五)。否则转到(四)。

(五) 在 line 上取一个不是多边形顶点也不是 line 端点的点，判断该点是否在多边形的内部。若是，则 line 在多边形的内部，转到(六)。否则，line 在多边形的外部，转到(七)。

(六) 将 line 标记为相交，并按照顺序把 line 端点和所有交点存入数组 V 中，最后插入一个分割点。

(七) 将 line 标记为在内部。

(八) 将 line 标记为在外部。

(九) 依次将 V 数组的相邻的两个非分割点的点组成线段，若当前点的下一个点为间隔点，则取下下一个点和下下下一个点组成线段。取该线段中点，判断该中点是否在多边形内部，若在，将该线段的两个端点存入裁剪后的数组 C 中。

(十) 被标记为在内部的 line 和 C 数组中第 2 * i（0 <= i < C.size / 2）个点和第 2 * i + 1（0 <= i < C.size / 2）个点组成线段即为裁剪后的

结果。

4.2circle 在非自交多边形边界中的裁剪

(一) 从所有的 circle 中取出一条 circle，若无 circle 可取，转到（十)。

(二) 假设 circle 包含多边形。

(三) (二)中的假设成立。计算 circle 圆心到多边形各条边的距离 d，若 d＞R 则(二)中的假设不成立。计算 circle 与多边形的交点，如果交点是多边形的顶点，判断 circle 是否穿过该顶点，则标记该交点为穿过，否则标记该交点为经过。如果交点不是多边形的顶点，则(二)中的假设不成立。

(四) 若(二)的假设成立，则转到(八)；若不成立，则转到(四)。

(五) 除去相邻的重复的交点。若所有的交点中存在穿过边的交点或者穿过顶点的交点，转到(八)；否则转到(五)。

(六) 判断圆心是否在多边形的内部，若在，转到(六)，否则转到(七)。

(七) 将 circle 标记为在内部。

(八) 将 circle 标记为在外部。

(九) 将 cricle 标记为相交。将所有的交点和圆的信息存入 V 数组中。

(十) 将 V 数组中的元素 E 依次取出。取 E 中的任意两个相邻交点（最后一个和第一个相邻)，判断两个交点与 E 中的圆所组成的圆弧的中点是否在多边形的内部。若在，将该两个交点以及圆信息裁剪后的数组 C 中。

(十一) 被标记为在内部的 circle 和 C 数组中的元素 F 中的圆与第 2 * i（0 <= i < F 中点的个数 / 2）个点和第 2 * i + 1（0 <= i < F 中点的个数 / 2）围城的圆弧即为裁剪后的结果。

5.开发环境

（一） 电脑硬件：联想 G450L、CPU 1.8GHz 双核、内存 3GB

（二） 操作系统：Windows 8.1 专业版 64 位

（三） IDE：Qt Creator 2.8.0、Qt 4.8.5、编译器 GCC 4.4.0

6.代码设计说明

（一） 全局变量说明

a) vector<Point> lineEntityPoints; //线段

保存 line，第 2 * i 个和第 2 * i + 1 个表示一条 line。

b) vector<Point> boundaryPoints; //裁剪窗口(边界)

保存边界，第 i 个点是多边形的顶点，第一个点同最后一个点。

c) vector<Circle> circleEntitys; //圆

保存 circle。

d) LineClipStatistic lineClipStatistic;//线段的统计数据

int total;//总数

int inCount;//在内部的数量

int outCount;//在外部的数量

int crossCount;//相交的数量

int crossPointCount;//交点的数量

e) CirCleClipStatistic circleClipStatistic;//圆的统计数据

int total;//总数

int inCount;//在内部的数量

int outCount;//在外部的数量

int crossCount;//相交的数量

int crossPointCount;//交点的数量

f) vector<LineFlag> linesFlags;//标记线段

大小等于 line 的总数，所以第 i 个的值就表示 lineEntityPoints 中第 2 * i 个和第 2 * i + 1 个点表示的 line 与多边形边界的位置关系。

g) vector<Point> clipLinePoints;//裁剪线段的点

保存排序后的不重复的与多边形相交的 line 的端点及交点，line 与 line 之间用一个分割点区分。

h) vector<Point> clipedLinePoints;//裁剪后的线段

保存与多边形相交的 line 裁剪后的线段。

i) vector<CircleFlag> circleFlags;//标记圆

大小等于 circle 的总数，所以第 i 个的值就表示 circleEntitys 中第 i 个 circle 与多边形边界的位置关系。

j) vector<CirclePoint> clipCirclePoints;//裁剪圆的点

保存与多边形相交的圆和该圆的有序的不重复的交点。

k) vector<CirclePoint> clipedCircleArcs;//裁剪后的圆弧

保存与多边形相交的圆裁剪后的圆弧。

(二) 关键代码详解

a) 判断 line 与多边形的交点是顶点的交点的类型

i. 原理

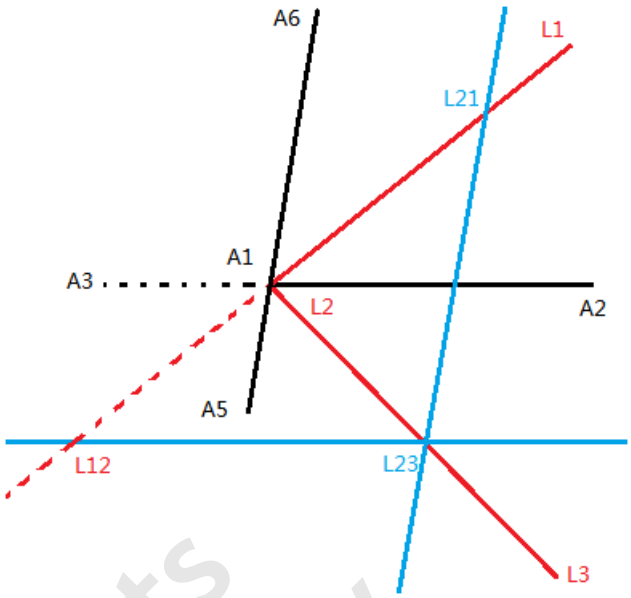


图 8

如上图：L1、L2、L3 为多边形的三个顶点。A2A1A3 共线。L23 是边 L2-L3 的中点。过 L23 作 A1A2 的平行线，与 L1L2 的延长线相交与 L12。A5A1A6 三点共线。过 L23 作 A5A6 的平行线，与 L2L1 相交于 L21。A1 与 L2 表示同一个点。

(1) 若判断 A1A2 与多边形的交点 L2 的类型，因为 L2 同 A1，即交点是线段的一个端点，所以 A1A2 不穿过顶点 L2。

(2) 若判断 A2A3 与多边形的交点 L2 的类型。过 L23 作 A2A3 的平行线与 L1L2 的交点 L12 在 L1L2 的延长线上(L1 - > L2 方向上)。即向量 L1L2 与向量 L2L12 同向，所以 A2A3 穿过顶点 L2。

(3) 若判断 A5A6 与多边形的交点 L2 的类型，过 L23 作 A5A6 的平行线与 L1L2 的交点 L21 在 L2L1 上(L2 - > L1 方向上)。即向量 L1L2 与向量 L2L21 反向，所以 A5A6 不穿过顶点 L2。

(4) 用向量的方式判断，假设向量 A=(x1,y1)，向量 B=(x2,y2)，两向量夹角为 a，那么 cos a = A · B / (|A| × |B|) = (x1x2 + y1y2)/(sqrt(x1x1 + y1y1)×sqrt(x2x2 + y2y2))。因为 a = 0° 或者 a = 180° ，即 cos

a = 1 或者 cos a = -1，即 cos a > 0 或者 cos a < 0，因为 sqrt(x1x1 + y1y1)×sqrt(x2x2 + y2y2)为正数，所以问题可转换为判断 x1x2 + y1y2（即向量的内积）大于零还是小于零的问题。

ii. 代码解释

```
PointType linePassByVertexPointType(Point *vertex, Point *vertexR, Point *vertexL, bool centerPoint, Point *start, Point *end)
{
    //centerPoint 为 true 的意思是判断经过线段上的点或者圆心做 Y 轴的平行线与多边形的交点的类型。
    //centerPoint 为 false 的意思是判断线段的交点。
    //因为判断交点时 line 的长度是固定的，即不能延长。
    //而 Y 轴的平行线是无线长的，即可以任意延长。

    //如果判断的是交点，并且待判断点是线段的起点或者终点，
     //那么一定是经过顶点，即 PassByVertexPoint

     if((!centerPoint)    &&    (compare2Points(vertex,start)    || compare2Points(vertex,end))){
         return PassByVertexPoint;
     }
     //如果是中点、或者是不是起点或者终点的交点
     Point *PA = vertexR;//多边形下一个顶点
```

```
Point *PB = vertexL;//多边形上一个顶点
Point *PO = vertex;//当前点，即交点
Point *PC = start;//线段的另一个点，取起点

//如果交点与起点重合，则取终点的坐标
if(compare2Points(PO,PC)){
    PC->x = end->x;
    PC->y = end->y;
    //PC->type = end->type;
}
//计算线段的一般式方程 Ax + By + C = 0;
//因为过多边形的一条边的中点作线段的平行线，所以该线一般式方程中 A、B 同线段的 A、B，只是 C 不同，所以代入该边中点可以计算出 C。
float A = PC->y - PO->y;
float B = PO->x - PC->x;
float C = (-1) * (A * (PO->x + PB->x) / 2.0 + B * (PO->y + PB->y) / 2.0);
//计算多边形的另一条边的直线方程的一般式
float A1 = PA->y - PO->y;
float B1 = PO->x - PA->x;
float C1 = (PA->x - PO->x) * PO->y - PO->x * (PA->y - PO->y);
//计算交点
```

```
    //利用两个一般式方程计算交点。
    float x = (B * C1 - B1 * C) / (A * B1 - A1 * B);
    float y = (A * C1 - A1 * C) / (A1 * B - A * B1);

    //生成两个向量
    float vectorAOX = PO->x - PA->x;
    float vectorAOY = PO->y - PA->y;
    float vectorOPX = x - PO->x;
    float vectorOPY = y - PO->y;
    //计算两个向量的内积
    float value = vectorAOX * vectorOPX + vectorAOY * vectorOPY;

    if(value > 0){
    //内积大于零，是穿过。
        return CrossVertexPoint;
    }else{
    //内积小于零，不穿过。
        return PassByVertexPoint;
    }
}
```

b) 判断 cricle 与多边形的交点是顶点的交点的类型

i. 原理：过交点（也就是顶点）做圆的切线，在切点两边各取一个点，

用这两个点组成一条线段，该问题就转换为判断 line 与多边形的交点是顶点的交点的类型。

ii. 代码解释

```
PointType circlePassByVertexPointType(Point *vertex, vector<Point> *bounds, Circle *cl)
{
    Point *vertexL = NULL;//多边形的上一个顶点
    Point *vertexR = NULL;//多边形的下一个顶点
    //计算当前点在边界中顶点的索引
    int index = calculateVertexIndex(vertex->x,vertex->y,bounds);

    int total = bounds->size();

    if(index == 0){
    //如果是第一个点，则上一个点是倒数第二个点，因为最后一个点同第一个点。
        vertexL = &(bounds->at(total - 2));
        vertexR = &(bounds->at(1));
    }else{
        vertexL = &(bounds->at(index - 1));
        vertexR = &(bounds->at(index + 1));
    }
```

```
// 圆心-顶点 直线方程
float A1 = vertex->y - cl->y;
float B1 = cl->x - vertex->x;
// 切线的直线方程，因为与上面的直线垂直，所以 A2 = -B1，B2 = -A1
float A2 = -B1;
float B2 = A1;
float C2 = (-1) * (A2 * vertex->x + B2 * vertex->y);

//生成切线上的点
Point p1 = createPoint(0,0);
Point p2 = createPoint(0,0);

if(B2 == 0){
    //与 y 轴平行，x 相同
    p1.x = vertex->x;
    p2.x = vertex->x;
    p1.y = vertex->y - 10;
    p2.y = vertex->y + 10;
}else{
//在切点两侧各取一个 x，然后代入切线方程计算 y
    p1.x = vertex->x - 10;
```

```
            p2.x = vertex->x + 10;
            p1.y = (A2 * p1.x + C2) / (-B2);
            p2.y = (A2 * p2.x + C2) / (-B2);
        }
        // 用判断 line 与多边形的交点是顶点的交点的类型的方式处理。
        return
linePassByVertexPointType(vertex,vertexR,vertexL,true,&p1,&p2);
    }
```

7.用户界面设计说明

1：标题栏 2：菜单栏 3：绘图区域

4：裁剪显示(完整显示)按钮 5：统计结果显示区域

图 9

8.测试示例

8.1 测试环境 1：本机测试

（一） 电脑硬件：Lenovo G450L CPU：赛扬 双核 1.8GHZ 内存：3GB

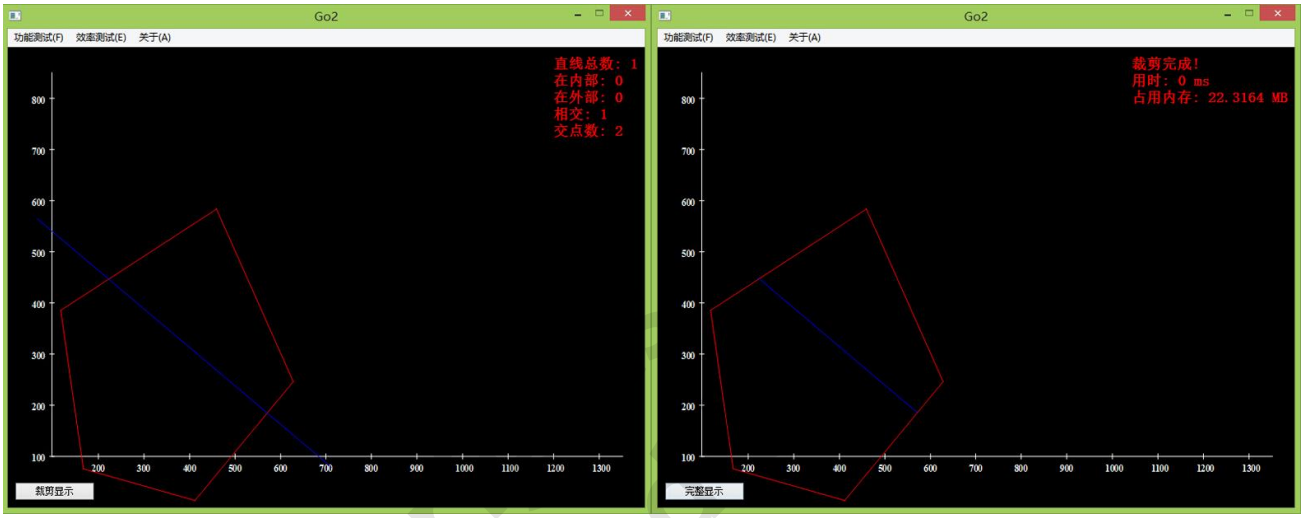
（二） 操作系统：Windows 8.1 64 位 专业版

（三） 测试文件：原文件：TestData1.xml、TestData2.xml

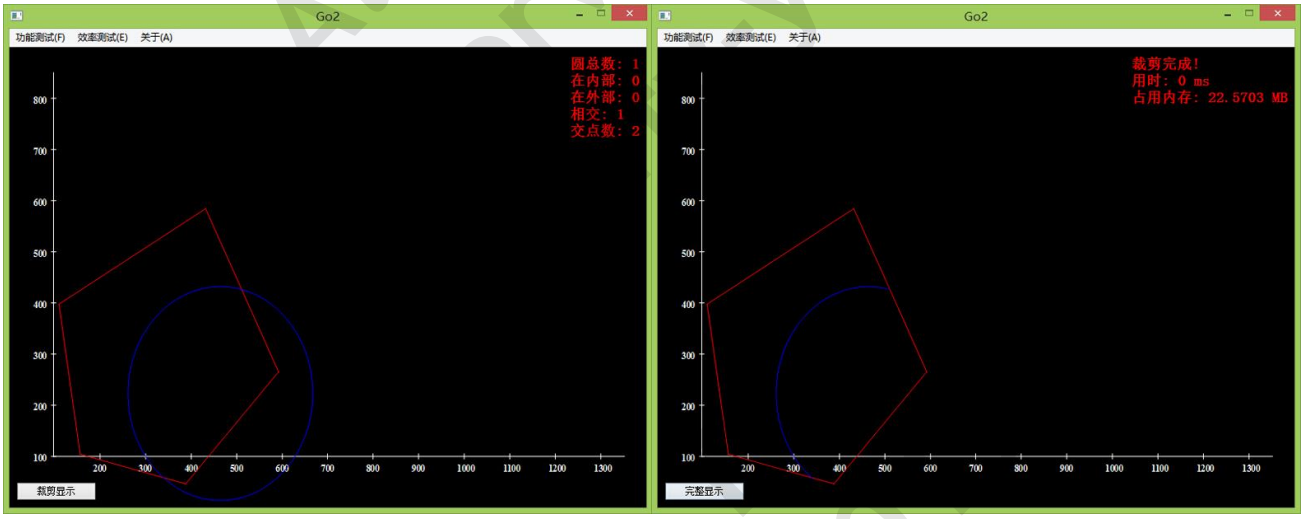
（四） 附加测试文件：./附加测试/ TestData1.xml。

（五） 测试结果：

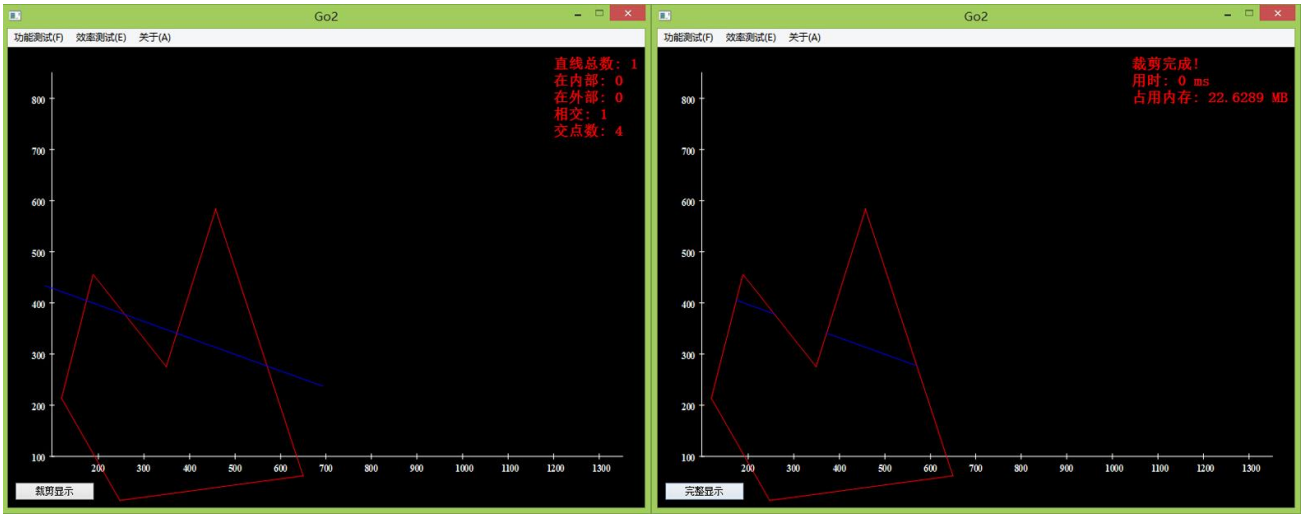
a） 功能测试 1



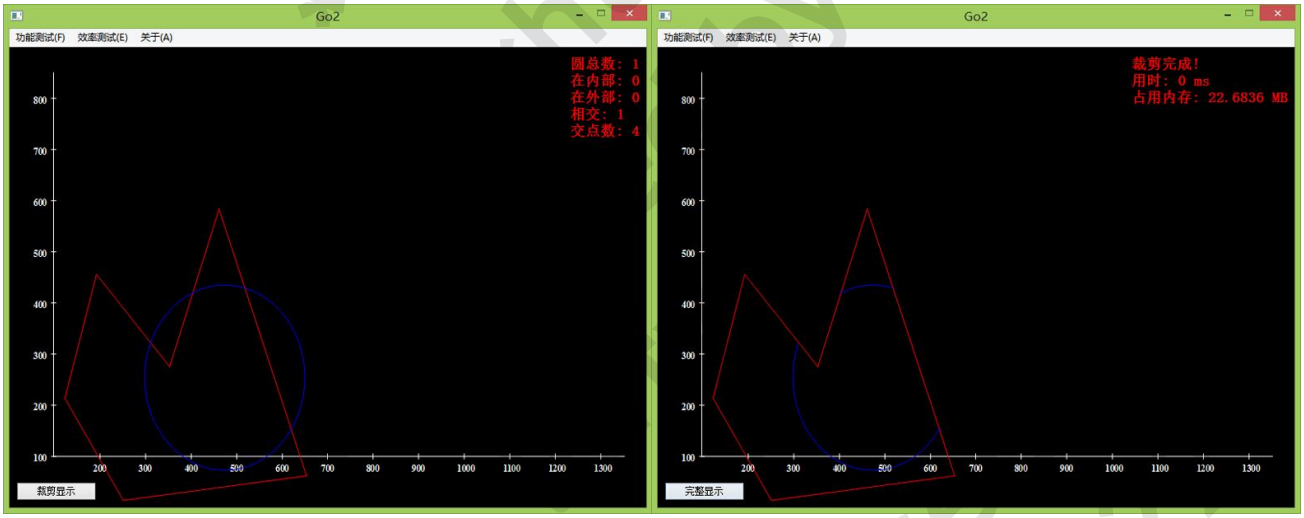
b） 功能测试 2



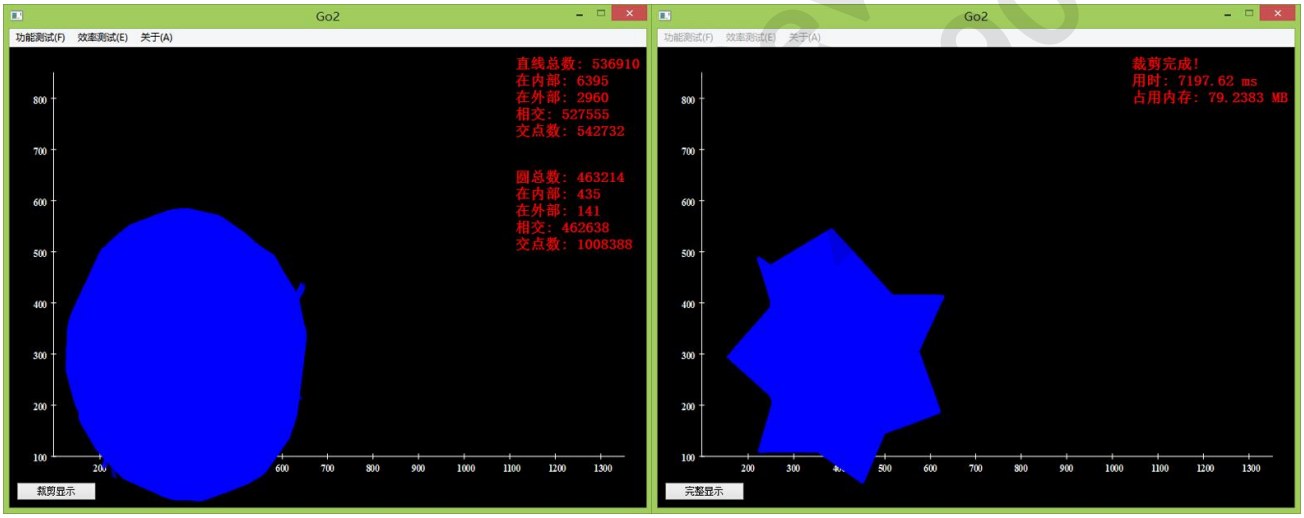
c） 功能测试 3



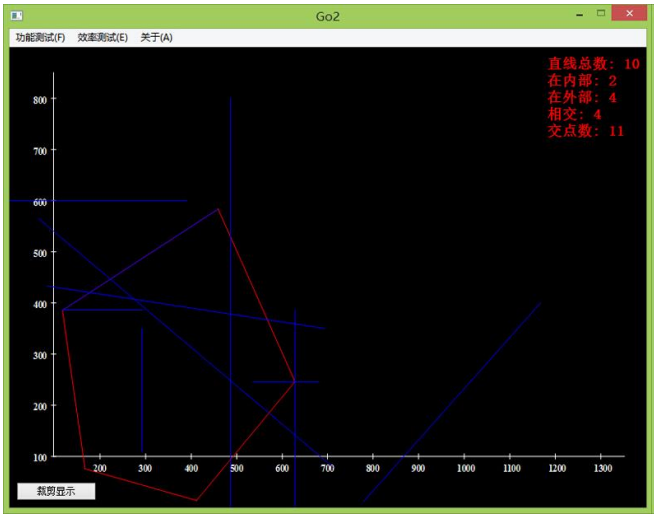
d）功能测试 4

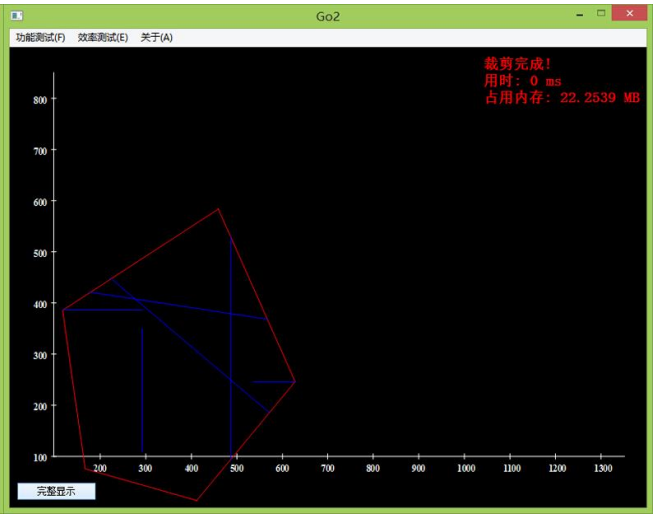


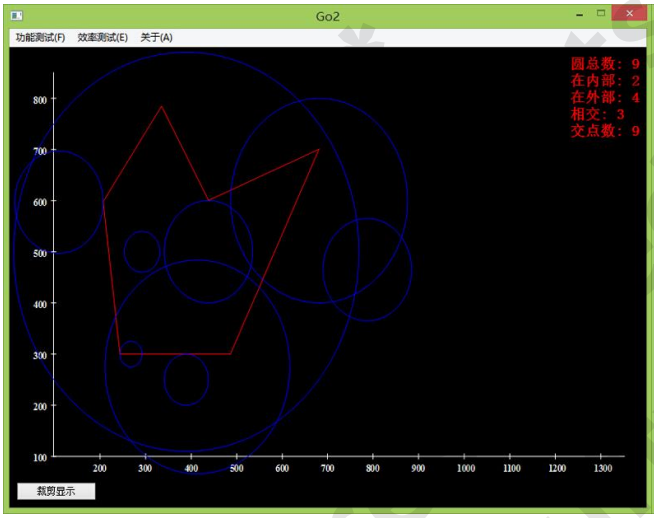
e）效率测试 5



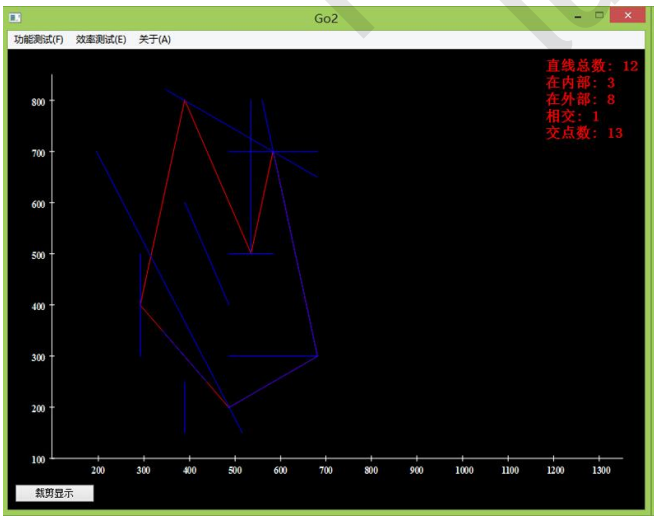
f）附加测试：

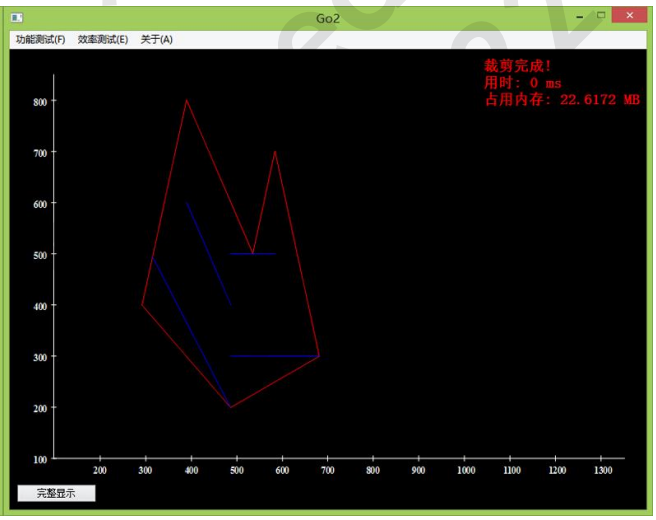

Go2
功能测试(F) 效率测试(E) 关于(A)
直线总数：10
在内部：2
在外部：4
相交：4
交点数：11
裁剪显示

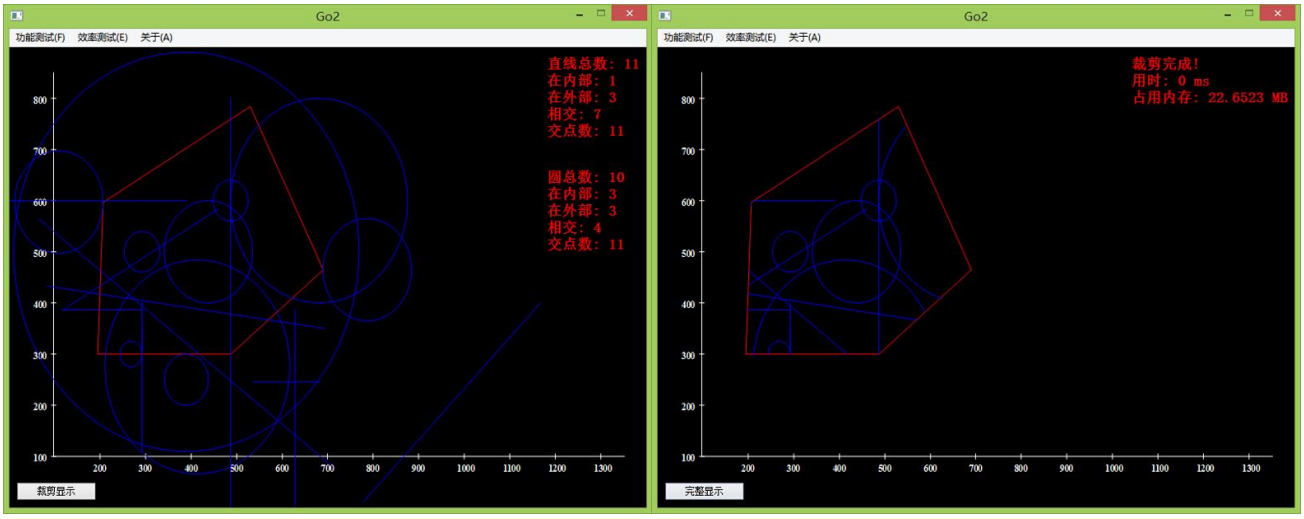

Go2
功能测试(F) 效率测试(E) 关于(A)
裁剪完成!
用时：0 ms
占用内存：22.2539 MB
完整显示


Go2
功能测试(F) 效率测试(E) 关于(A)
圆总数：9
在内部：2
在外部：4
相交：3
交点数：9
裁剪显示


Go2
功能测试(F) 效率测试(E) 关于(A)
裁剪完成!
用时：0 ms
占用内存：22.457 MB
完整显示


Go2
功能测试(F) 效率测试(E) 关于(A)
直线总数：12
在内部：3
在外部：8
相交：1
交点数：13
裁剪显示


Go2
功能测试(F) 效率测试(E) 关于(A)
裁剪完成!
用时：0 ms
占用内存：22.6172 MB
完整显示



9. 问题及不足

(1) 交点的计算存在重复点。

由于交点个数的统计是每个 line 或者 circle 与多边形求交点，去掉重复后的交点的个数，所以存在重复点。对于小数据量可以通过排序去除重复，但是如果是大数据量的话，再通过排序去除重复将增加 CPU 耗用。

(2) 图形的绘制过慢。

使用 Qt 的 paintEvent 事件绘制图形时，当窗口发生拖动、按钮的点击或者窗口焦点的得失，都会引发窗口重绘。所以这增加了图形绘制的时间。另一种方案是使用图形视图框架。但是如果使用 Qt 的图形视图框架绘制图形（使用 QGraphicsItem 的子类展示 line 和 circle）将大幅增加程序的内存消耗。