**开发技术文档**

**Sunsiy10**

**2014.12.25**

1. **程序的框架结构**

按照功能的实现流程划分，本程序分为布尔运算模块和合法性检查模块。

* 1. **布尔运算模块**

布尔运算模块按照流程划分可分为三部分，分别是将一般多边形转换成BspTree，BspTree的合并以及计算结果多边形的边。在实现时，BspTree的节点由代码中的CP\_BSPNode数据结构表示。

**1.1.1将一般多边形转换成BspTree**

这部分涉及到的函数主要为gb\_buildPolygonBSPTree，gb\_buildRegionBSPTree和gb\_buildLoopBSPTree。

gb\_buildPolygonBSPTree将CP\_Polygon转换为BspTree：

在实现时将CP\_Polygon内包含的CP\_Region依次转换为BspTree，再将这些BspTree用布尔运算“并”合并，最后返回得到的BspTree。其中合并操作在1.2小节介绍。

gb\_buildRegionBSPTree将CP\_Region转换为BspTree：

在实现时将CP\_Region内的所有CP\_Loop转换为BspTree，再用外环的BspTree依次与所有内环的BspTree做布尔运算“差”合并，最后返回得到的BspTree。

gb\_buildLoopBSPTree将CP\_Loop转换为BspTree。具体过程见原理文档5.1小节。

* + 1. **BspTree的合并**

这部分涉及到的函数主要为gb\_mergeBSPTree，gb\_partitionBspt和gb\_mergeTreeWithCell。

gb\_mergeBSPTree实现对输入的两个BspTree A和B的合并：

当A和B中有一个为叶子节点时，调用gb\_mergeTreeWithCell在指定布尔运算的条件下合并A和B，并返回合并后的结果；

否则用A根节点的partition将B分成两棵树B\_left和B\_right，这个过程通过调用gb\_partitionBspt实现。随后分别调用gb\_mergeBSPTree合并A的左孩子和B\_left，A的右孩子和B\_right。返回合并结果。

gb\_mergeTreeWithCell实现BspTree的叶子节点和BspTree的合并，通过指定布尔运算操作可以得到对应的合并结果。

gb\_partitionBspt实现用一个partition(等价于一条直线)将一个BspTree T分成两棵树T\_left和T\_right，这两个BspTree分别在partition的两侧。在实现时通过判断partition与T根节点的partition的位置关系的8种情况来决定是否需要继续用partition分割T的左、右孩子。具体介绍见原理文档第三大节。

* + 1. **计算结果多边形的边**

这部分涉及到的函数主要为gb\_generateCellPolygons和gb\_generateBSPTreeFaces。

gb\_generateCellPolygons得到每个叶子节点对应的区域的边界，并记录下从根节点到叶子节点之间的中间节点对区域边界的贡献。具体介绍见原理文档第四大节。

gb\_generateBSPTreeFaces得到结果多边形的边界。具体介绍见原理文档第四大节。

* 1. **合法性检查模块**

合法性检查模块用于判断一般多边形的合法性，这部分主要有如下几个函数：

gb\_checkPolygon判断一般多边形是否合法，在其内部通过调用gb\_checkRegion判断每个区域的合法性；通过调用gb\_checkRegionCrossRegion判断两个区域是否有边相交的不合法情况出现；然后将区域转换成BspTree，通过调用gb\_tree1OverlapWithTree2判断一个两个区域是否有重叠的部分。

gb\_checkRegion判断每个区域的合法性，在其内部通过调用gb\_checkLoopSelfIntersection判断环时候自交以及外环内环方向是否正确；通过调用gb\_checkLoopIntersection判断环之间是否有相交的不合法情况出现；然后将环转换成BspTree，通过调用gb\_tree1OverlapWithTree2判断内环之间是否有相互覆盖的情况出现；通过调用gb\_tree1InTree2判断内环是否在外环内部。

1. **编程技巧**

**2.1 BspTree合并的debug辅助**

在调试BspTree的合并操作时，由于树形结构的CP\_BSPNode节点在调试窗口中不便于查看其信息，在实现时将此结构输出到一个文档中，并标明每个节点的partition，这样在调试错误时会相对容易些。

**2.2 合法性检查**

在检查一般多边形的合法性时，利用了BspTree的结构特性。

通过将两个不同的区域转换成BspTree，并对其进行布尔运算“交”操作，得到的结果BspTree如果包含属性为“in”的叶子节点，则这两个区域之间有重叠的面积大于0的部分。利用这个特性可以辅助合法性检查。同理这个方法也可以辅助判断内环之间是否有面积大于0的重叠的部分。

通过将两个环L1和L2转换成BspTree，并对L1和L2的BspTree作布尔运算“差”(L1 – L2)，得到的结果BspTree若不包含属性为“in”的叶子节点，则L1在L2内部。这个策略可以用来辅助判断内环是否在外环内部。

**2.3 内存管理**

用全局vector容器记录动态申请的内存，在类内也有一个vector容器用于记录动态申请的内存。这样在一次布尔运算结束后，利用全部容器释放在类外部申请的内存，类内部函数动态申请的内存可在类析构时释放，即释放容器内申请的内存。

1. **程序的编译、运行和配置方法**

项目在VS2012环境下进行编写，基于MFC的Ribbon图形界面。在编译时需使用VS2012打开项目，按Ctrl + F5即可对程序进行编译和运行。