**网格简化实验报告**

计24 黄晨 2012011337

目录

[1 程序描述 1](#_Toc392532229)

[2 程序类介绍 1](#_Toc392532230)

[2.1 Vec3f类 1](#_Toc392532231)

[2.2 SimpleObject类 1](#_Toc392532232)

[2.3 Matrix类 1](#_Toc392532233)

[2.4 EdgeCollapse类 1](#_Toc392532234)

[3 程序算法描述 2](#_Toc392532235)

[3.1 初始化 2](#_Toc392532236)

[3.2 网格简化 2](#_Toc392532237)

[3.3 结果转换 3](#_Toc392532238)

[4 程序亮点 3](#_Toc392532239)

[5 程序运行结果 3](#_Toc392532240)

[6 总结与收获 3](#_Toc392532241)

[7 参考资料 3](#_Toc392532242)

1. 程序描述

本程序实现了网格简化，采用边坍塌的方式，基于二次误差的度量实现了网格简化。采用堆、红黑树（set）和map进行了优化，使初始化网格与简化更加迅速。

使用说明：通过命令行传递参数，第一个参数为输入文件名，第二个参数为输出文件名，第三个参数为简化率，第四个为可选参数，为t值，默认为0.01

1. 程序类介绍
   1. Vec3f类
      1. 表示坐标、颜色的类
      2. 提供了计算2阶距的函数和单位化的函数
      3. 重载了[]运算符以方便使用
   2. SimpleObject类
      1. 简单obj类，进行obj文件的读取（仅读取顶点v和面片f）
      2. 顶点坐标以Vec3f类存储，面片信息用三维数组表示
   3. Matrix类
      1. 矩阵类，用以存储矩阵
      2. 重载了加、减和乘运算符以实现矩阵运算
      3. 提供了transposition()方法实现矩阵的转置，方便运算
      4. 重载了[]操作符以方便使用
   4. EdgeCollapse类
      1. 边坍塌算法的实现类，继承自SimpleObject，方便读取obj文件
      2. 成员
         1. heap：堆，用以存储各pair的二次误差值和pair的点的信息
         2. order：map，记录各pair在堆中的位置，方便修改和删除
         3. v\_Q：各点的Q方阵数组
         4. pairs：与各点组成pair的点的列表，其中对每个点使用set结构进行存储
         5. faces：各点所属面片的列表，对每个点使用set结构存储面片编号
         6. deletedPoints, deletedFaces：算法过程中被删除的点和面片的列表
         7. projection：点编号的映射，用于最后将结果进行转换时使用
         8. initialize()：初始化方法，主要功能为获取pair信息，计算各点的Q方阵，建立最小堆，更新pairs、faces和order的信息
         9. belong(…)：判断某点是否属于某个面
         10. simplify()：网格简化的外部接口，其中会调用\_simplify()进行简化的步骤
         11. 其余方法属于辅助函数，在此不在赘述其功能
2. 程序算法描述
   1. 初始化
      1. 定义2各set：v\_edge\_pairs, v\_noedge\_pairs，分别表示有边相连的pair集合和无边相连（距离小于t）的pair的集合
      2. 遍历所有的面，对每个面f，计算f的K方阵，将属于f的每个顶点的Q方阵加上K。同时，将该面中由边连接产生的pair加入v\_edge\_pairs，加入时注意使第一个顶点的编号小于第二个顶点以方便后续计算，更新faces列表
      3. 将所有顶点按照其到原点的距离从小到大排序，距离相等按点编号从小到大排序，注意在排序时记录顶点编号，得到顶点列表v\_sort
      4. 依下标从小到大遍历v\_sort中的所有顶点，对每个顶点vi，依下标从小到大遍历从i+1开始的所有顶点，对满足|vk-vi|<t的顶点，若(vi,vk)不在v\_edge\_pairs中，则将其加入v\_noedge\_pairs。当遍历结束，或|vk|-|vi|>t时停止遍历。
      5. 遍历v\_edge\_pairs和v\_noedge\_pairs中的每一个点对，计算点对的二次误差值，将(delta, v1, v2)加入堆中，更新v1和v2在pairs中的信息
      6. 调用stl的标准函数将heap建成最小堆
      7. 遍历heap中的每一个元素，更新map中各点对在heap中的位置
   2. 网格简化
      1. 若已删除面片数达到目标值，返回
      2. 取出堆顶元素(delta, v1, v2)
      3. 将v2标记为已删除点，v1将在之后更新为坍塌后的点。本程序为简化计算，直接去中点为坍塌后的点。
      4. 将v1的Q矩阵清零
      5. 建立set：changePoints，用以存储此次简化中矩阵发生变化的点
      6. 遍历所有与v1相关联的面
         1. 若其不包含v2，则将其包含的不同于v1的点加入changePoints，更新其Q矩阵。这里采用先剪去原来的面的K矩阵，再加上新的面的K矩阵的方式更新。同时更新v1的Q矩阵。
         2. 若其包含v2，则将该面加入deletedFaces，更新另一不同于v1和v2的点的Q矩阵（减去原来该面的K矩阵即可），将该面从v2的faces中去除
      7. 遍历所有与v2相关联的面，类似3.2.5中的更新步骤更新各关联顶点，其后将面列表中v2的标号替换为v1的标号，并更新v1的Q矩阵
      8. 将所有与v2关联的面加入到与v1相关面的列表中
      9. 在pairs中删除点对(v1, v2)（这里注意需要更新两个点的信息）
      10. 根据order和pairs的信息在堆中更新changePoints中各点相关联的pair的二次误差，并对堆进行维护，使其保持堆序性。注意这里需要将与v2相关的pair替换为与v1相关，若堆中本身就有该点与v1构成的pair，则直接删除该点与v2的pair即可，否则需要替换
      11. 将pair中与v2相关的点更新为与v1相关
      12. 将与v2相关的点全部加入v1的pairs中
      13. 更新v1坐标为v1和v2中点
      14. 将点v2映射至v1（修改projection中的值为v1的编号）
      15. 回3.2.1
   3. 结果转换
      1. 将所有未删除的顶点加入新的顶点列表，并记录顶点在新顶点列表的编号，更新projection
      2. 对每个未被删除的面片
         1. 对其每个顶点v
            1. 若v在被删除顶点列表中，则将v置为其在projection中的值，回3.3.2.1
            2. 否则，将v置为其在projection中的值
3. 程序亮点
   1. 初始化过程优化
      1. 先对顶点排序，然后再进行遍历
      2. 在点不是非常密集的情况下，可以有效地加快初始化效率
   2. 简化过程优化，通过牺牲空间提高简化效率
      1. 一个obj所产生的pair非常多，若每次依次遍历将非常耗时，特别在t值比较大的情况下
      2. 通过faces和pairs两个列表减少每次遍历的次数，有效减少了每次简化的时间，提高了简化效率
      3. 实际测试发现，在不优化的情况下，简化速率约为6-12片/s，优化后速率可打500-1000片/s，即使在t值大于物体尺寸1/15时简化效率仍有100片/s
4. 程序运行结果
   1. 见附件打包的obj，每个obj的明明方式为name\_p.obj，其中p表示简化率
5. 总结与收获
   1. 在空间允许的条件下，可以通过牺牲空间以提高程序性能
   2. 对于简化结果的现实，可以使用光线跟踪的程序对其进行显示，结合键盘操作即可实现视角变换
6. 参考资料
   1. 课件
   2. C++ Reference