计算机图形学基础 网格简化作业 算法说明

计算机系 计 15 刘晨

一、 提交文件目录：

--bin 可执行文件 测试 obj 文件 输出 obj 文件 用于 windows 批处理测试的文件。

--src Visual Studio 2010 项目文件

--doc 说明文档

二、 基本类的说明（详细说明见代码注释）

1. 点类 point 类

   用于表示三维空间中的一个点，包括坐标信息 x,y,z 和是否删除的标记 del。

2. 四阶矩阵 qmatrix 类

   重载了加减乘除运算符矩阵类，内有 4*4 的二维数组 elem。

3. 线段 line 类

   线段类作为之后 mesh 主类的十字链表的基本单元，包括上下左右四个指针，作为 mesh 类中维护小顶堆的元素对应单元，存有一个表示稍后指向小顶堆的整型索引变量 index。存有两个整型变量，分别表示两个端点的点的索引值，并且严格满足 first<=second。double 类型的 length 类表示线段的长度，point 类型的 combine 表示如果两点合并后的替代点的坐标。

   线段类本身有一定的计算功能，如果给定两个顶点的坐标以及它们对应的四阶矩阵的值，该函数可以算出替代点的坐标，返回值为 delta 的值。

4. 面片索引 tri_num 类

   作为索引类，tri_num 类包括三个顶点的编号值 p1,p2,p3。另有 a,b,c,d 四个变量表示面片的解析几何参数 ax+by+cz+d=0.函数 calc 通过传入三个顶点的坐标来计算出 a,b,c,d 的值。

5. 堆元素 heap_node 类

   堆中的元素由于需要大规模进行交换等操作，所以其结构需要尽量简洁。heap_node 类存有 delta 变量，用于表示两点合并的代价值，由于堆中的任何一个元素均对应一个线段，表示线段的 line 类中也存有其对应的堆中节点的信息，所以 heap_node 中也存有一个指向其对应的线段的 line 类型的指针。

6. mesh 主类

   mesh 类是进行网格简化的主类，main 函数主要就是调用该类中的方法对 obj 文件进行网格简化操作的。

   mesh 类的成员变量如下：

   vector<point> pt:用于存放图片中点的集合

   vector<tri_num> tri:用于存放面片索引的集合

   line **lx **ly:十字链表的坐标轴

   vector<int>* t_table:从点向面片索引的倒排列表，元素的值表示该面的索引在 tri 中的下标值，包含同一个顶点的所有面片的索引构成一个 vector。

   qmatrix *qpt *qface:所有面和点所对应的四阶矩阵。

   vector<heap_node>:优先级队列，由于表示符合要求（距离小于一定值）的线段按照 delta 代价组成的小顶堆。

   int m_point m_tri:总点数和总面数。

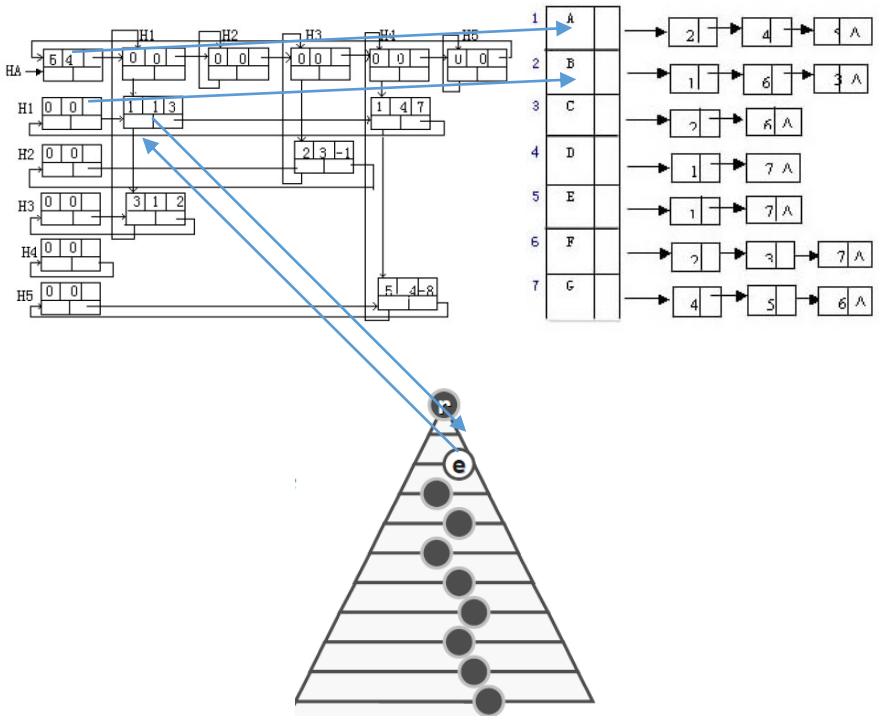
int del_point m_tri 已经删除的点数和面数。

double rate,maxdis 简化率和加入堆中的最大边的长度。

mesh 类对应的成员函数以及其主要功能如下表所示：

函数	功能	返回值
mesh(double r, double max)	构造函数,r 为简化率,max 为最长边长	
~mesh()	析构函数	
int input(string infile)	读入 obj 文件，初始化点列表和面索引列表	读入文件成功与否
void output(string outfile)	将简化后所剩的点和面的集合整理输出到 outfile 文件中	
void initialize()	初始化操作，建立关于边信息的十字链表，建立从点到面片索引的倒排列表，计算各点和面的 q 矩阵的值	
void find()	从边列表中选取长度小于 maxdis 的边，计算其 delta，组建成堆	
void heap_adjust(int num)	对堆进行局部调整，不符合堆中优先级限制条件要求的节点下标为 num	
int del_a_point()	合并图中的一组顶点	实际删除掉的面片数
void simplify()	按照一定的简化比对网格进行简化，如果出现堆中无元素的现象，则将 maxdis 翻一番，重新初始化合并顶点	
void show()	显示模块通过 openGL 显示化简网格	

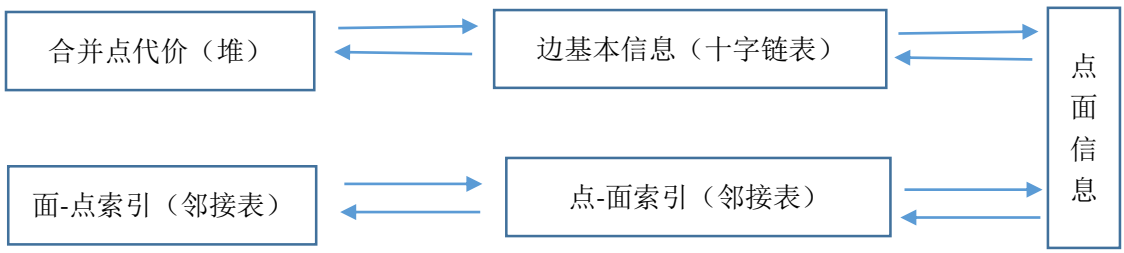
三、 主要数据结构和算法复杂度的分析：



如上图所示，在求解本题的过程中，我们主要用到了邻接表、优先级队列和十字链表三

种数据结构。采用一般的方法自上向下建堆，复杂度为 O(nlog(n))，n 为堆中元素数，对堆中的元素进行调整的复杂度为 O(log(n))。十字链表建立的复杂度与建立邻接矩阵的复杂度相同复杂度的上限为 O(m^2)，下限为 O(m^2/n)，其中 m 为十字链表或者邻接表中元素的个数，n 为它们的行数或列数，十字链表和邻接表的查找复杂度上限为 O(n)，下限为 O(m/n)，期望为 O(m/n)。

在下图中，我们可以形象地看出 mesh 类中各个存储结构之间的关系：



通过箭头连接的两端，我们都可以通过下标索引或者指针的形式进行访问，十字链表中的边信息通过下标索引访问堆中元素，而队中元素则通过指针的方式访问对应的边。除了 heap_node 之外所有的数据结构都存有 pt 列表中点下标或者 tri 中面下标的索引，通过它们借助点-面和面-点索引对照表可以访问到对应的点信息和面信息。所有的这些都方便合并顶点操作所造成的影响对各个列表参数的修改。

在 mesh 类的 del_a_point 函数中，合并两个顶点的顺序为：

1. 从堆顶取出一个节点，判断节点是否有意义，如无意义，则删去堆顶，调节堆，返回；否则转 2。
2. 记录堆顶对应的点 index1 和 index2，删去堆顶，调整堆。（复杂度 O(log(n))）
3. 懒惰删除 index2，将合并的顶点值赋给 index1。
4. 通过点-面索引删除 index1 和 index2 的 q 矩阵值周边面的影响，处理 index1 和 index2 的共同面（容斥原理）。（平均复杂度 O((m/n)^2)）
5. 更新点-面列表，将 index2 对应的部分与 index1 合并，删去 index1-面索引中重复的面编号，清空 index2-面列表，更新面信息。（复杂度 O(m/n)）
6. 对新点 index1 对应的新面，加上 index1 的 q 矩阵对它们的 q 矩阵的影响。（复杂度 O(m/n)）
7. 从十字链表中抽出所有含有 index2 的元素（根据坐标轴上的指针搜索，平均复杂度为 O(m/n)），将其中的 index2 改为 index1 重新插入十字链表，更新它们对应的堆中的元素的值。（平均复杂度 O((m/n)^2)）
8. 更新第 7 步中所涉及的十字链表中的边的长度，根据他们与 maxdis 的大小关系确定是否想堆中加入新元素或者将已有元素从堆中删除。（复杂度 O((m/n)*log(N))）

选用十字链表的原因在与它能够有效防止列表中出现重复边的现象，通过堆中节点和十字链表中的元素一一对应的方式能够防止堆中出现大量的重复的元素，防止堆规模的恶性膨胀，节约了内存资源，节省了运行时间。建立点面之间的双向链接关系，虽然在建立是成本比较高，但是在后续过程，特别是删除点的过程中，双向的索引提供的快速访问的接口能够使我们在后续过程中节省大量时间。

四、 实现的功能

支持 obj 文件的读入和写出，以命令行的形式读入化简的参数，才参数形式或者参数取值错误的情况下，提示错误并且输出。

采用 openGL 绘出化简后的效果图(三视图)。

五、 运行结果

以样例中 dragon 为例，下面三个图分别是简化比为 0.1、0.03 和 0.005 的效果图(左为用软件查看，右为 openGL 打印的主视图)：