网格简化报告

计科20 2012012440 宋正阳

2015 年1 月23 日

1 算法综述

二次网格简化是一种边坍缩简化算法。每次执行将会选中一条边并将其删掉。具体来讲是首先删掉被选中的边，将它的两个顶点合为一个顶点。同时这条边两侧的面片会退化成一条边，也就是说这两个相邻面片也会被删掉。并且对于这被删掉的三角面片，其余的两条边合并为一条。对于顶点的其他相邻面片，只需将对应的顶点移动到相应的新点位置。综上，每次执行的结果是顶点数减一，面片数减二，边数减三。

对于一个给定的obj 文件，我们可以确定其所有的顶点、面片、边之间的互相关系。对于其每一条边，我们定义一个其坍缩后的cost。于是每次我们只要选出cost 最小的那一条，并执行该坍缩过程。

每条边的cost 是由其两个顶点决定的，而每个顶点又是由其相邻的所有面片决定的。具体来说是我们在每个面片、每个顶点、每条边处都存储了一个4 乘4 的矩阵。边上的矩阵为两个顶点上矩阵的和，顶点上矩阵为相邻所有面片矩阵的和。而面片上矩阵的计算则是假定其方程为$ax + by + cz + d = 0$,且$a^2+b^2+c^2 = 1$, $Q = (a, b, c, d)(a, b, c, d)^T$. 这样对于某一个点$v = (x, y, z, 1)$, vQv^T 则直接为v 到这个面距离的平方。于是对于某一条边的矩阵M 来说，某一个点$v = (x, y, z, 1)$, vMv^T 则直接为该点到该边两个顶点相邻所有面片的距离的平方和。于是我们指定边坍缩后的点为使得该平方和最小的点，而该条边的cost 即为此最小值。同时坍缩后我们不再重新计算新面片的矩阵及新点的矩阵，直接将该坍缩边的矩阵赋为该新点的矩阵，并据此计算出新边的矩阵，计算新边的candidate 及cost.

为了加快程序的执行速度，我们初始化数据结构之后对于所有的边计算cost, 并将其放入最小堆中。每次执行我们只需取出位于堆顶端的那条，而非遍历整个边结构。而坍缩执行完毕后，我们只需将需要改动的边标为已删除，并新生成cost 正确的边压入堆中。原因在于直接修改会打乱堆结构，而重建堆要耗费大量的时间。下次只需在执行坍缩前检查该边是否已被删除，如果是则直接抛弃，继续取出堆顶的下一条。

2 代码综述

本项实验的代码总共包含三个类一个主函数，各类的具体论述如下：

2.1 Vec3f 类

该类为助教所给辅助类，主要是三维向量的声明及各种运算。需要注意的一点是原代码中Normalize 函数是当$L2Norm_Sqr > 1e-6$ 才会调用，但是对于某些模型来说这个数值太过小（例如Kitten），这里我们把它改成$1e-12$.

2.2 SimpleObject 类

该类亦为助教所给辅助类，主要功能是输入输出obj 文件。输入即将obj 文件中的点、三角面片读入$m_pVertexList$, $m\_pTriangleList$ 两个向量，并获得点、面的个数$m_nVertices$, $m\_nTriangles$. 而输出即是将两个数组里所保存的数据结构输出为合适的obj 文件。

2.3 AuxMeshSimp

由于SimpleObject 类过于简单不适合完成网格简化的过程，所以我们重写了Vertex 类以及Triangle 类、Edge 类，使其包含更多地信息。

2.3.1 Triangle 类

Triangle 类最为简单，只是包含了三个顶点的指针vertex[3]，法向量normal，上文中提到的用于计算误差的矩阵Q，以及一个根据顶点坐标计算法向量的函数ComputeNormal。

2.3.2 Edge 类

该类包含其两个顶点的指针vertex[2]，用于计算误差的矩阵Q，计算出的误差cost，计算出的新点位置vbar ，并标明是否已被删除deleted。

成员函数包括：根据顶点矩阵计算该边误差矩阵ComputeMatrix，根据误差矩阵计算误差ComputeCost，计算新点位置ComputeVbar。

2.3.3 Vertex 类

该类包含最为丰富的信息。其保存了该顶点的index, 该顶点的位置position，该顶点是否被删除deleted，该顶点相邻的所有顶点的指针neighbors，该顶点为端点的所有边的指针pairs，该顶点所有相邻面片的指针faces，以及该顶点的误差矩阵Q。

同时成员函数ComputeMatrix 根据所有相邻面片矩阵计算该点矩阵（简单将其相加），以及CleanNeighbor(u) 检查u 是否仍与其相邻，若不是则将其从相邻顶点列表中删除。

2.4 mesh_simp

主函数main 所在的文件包含了处理网格所需的核心函数们, 其操作刘承俊具体如下：

2.4.1 main

首先确定命令行参数为4，第一第二个分别为输入输出的文件名，第三个为网格简化百分比。

开始计时。

将obj 文件读入SimpleObject 类所指明的两个vector 中，获得所有的顶点和三角形。将该obj 实例及第三个参数rate 传给SimplifyMesh 函数，意为将obj 的面数简化至rate。

简化完成后输出此时的点数和面数，并将结果保存至相应的文件。

停止计时。

2.4.2 MeshSimplify

该函数首先初始化数据结构，即根据obj 中内容初始化用于保存点，面片和边的vertices, triangles, edges 中。

计算出要删除多少个面片。

当未达到删除面片数时:

从堆中取出顶上的edge, 执行Collapse 函数。

坍缩结束后将表明已删除的点从vertices 中去除。

将结果写入obj.

2.4.3 Collapse

首先得到边的两个顶点u, v. 我们计划将v 移至相应位置，并将u 标为已删除。

首先将该边的新误差矩阵赋给v, 并将v 的位置改为相应新点的位置。

删掉相邻两个面。在这个过程中两顶点的faces 和pairs, neighbors. 注意到pairs 与neighbors 是平行变动的。

由于v 的位置已变动，故其相邻边均已失效，将v 的pairs 全部清空，将相应的边标为已删除，并遍历v 的neighbors, 添加新边至pairs, 并将新边压入堆中。

然后对于除已删除两面的其他面，均将顶点u 换为顶点v, 在这个过程中同样将变动位置的边变为已删除并添加新边进入。

将顶点u 标为已删除。

3 用法

只需在命令行中输入mesh_simp in.obj out.obj rate，即表示将in.obj 的面数简化至rate(百分比)，然后输出为out.obj.

4 编程体会

本次作业过程中的经验总结如下：
1. obj 文件中面的顶点保存顺序遵循右手法则，据此可知法向量的朝向。
2. visual studio 中win32 控制台程序下的stdafx.h 头文件要进行额外设置。
3.测试时obj 的输入及输出要与代码放在同一文件目录下。
4. 对于添加现有文件要修改项目属性中的附加包含目录。
5. visual studio 的报错scanf 不安全可直接取消掉。
6. obj 中点的下标是从1开始的，而数据结构中是从0 开始的。
7. WriteToSimpleObj 要引用传参，不然不会有任何变化。
8. vector 中循环下调用erase 时要格外小心。
9. 对于自定义的类，调用push_heap 时要传入自定义比较函数cmp.
10. delete u 是指将u 指针指向的元素删除，而u 指针本身不会变化。