南开大学
Nankai University

南　开　大　学

计 算 机 学 院

计算机图形学大作业

网格简化算法实现

姜奕兵 1710218

周辰霏 1712991

专业：计算机科学与技术

2019 年 12 月 12 日

摘要

实现基于二次度量误差的边坍塌网格简化算法, 用户自选简化文件和简化比例, 最终结果由 OpenGL 显示, 界面可以通过键鼠操作进行平移、旋转、缩放, 以及不同的网格显示。

在此附上GitHub 仓库以及演示 Demo

关键字： 网格简化；边坍塌；OpenGL 显示；C++

目录

一、 作业要求

实现一个网格简化算法

- 实现边坍塌的网格简化方法
- 使用 OpenGL 进行网格显示
- 程序可以指定输入输出的.obj 文件并指定简化比例 (输出面数/输入面数)
- 除数学工具函数外不允许使用任何库提供的现成实现

升级功能

- 显示界面支持鼠标控制的旋转、平移、缩放

二、 开发环境

- 系统环境:MacOS 10.15,Windows 10 家庭中文版 (64 位)
- 编译环境:Visual Studio 2015,Xcode
- 编程语言:C++

三、 程序结构

(一) 代码实现流程

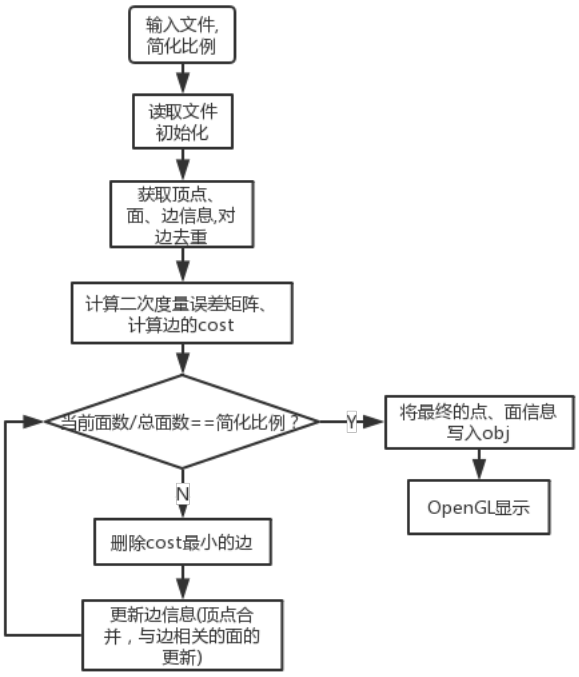

图 1: 代码实现流程图

(二) 代码结构综述

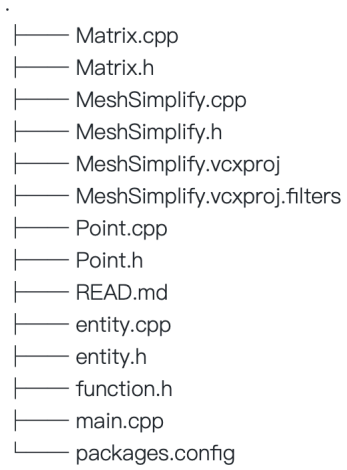

图 2: 代码结构

class	Specification
Point	空间点类, 包含点定义以及基本空间向量运算
Matrix	矩阵类, 二次度量误差矩阵的定义及矩阵线性运算
Entity	obj 信息类, 包含 obj 传递的点线面信息定义及其相应操作
Function.h	简单数学函数封装。向量内外积, 矩阵行列式, 向量标准化
MeshSimplify	主要网格简化功能类。读取 obj 信息并进行边坍塌简化并最终输出文件
main.cpp	借助上述类实现网格简化并通过 OpenGL 显示结果, 界面可以进行旋转平移等

表 1: 代码结构概述

四、 功能实现详细设计

(一) 初始化 obj 文件信息

通过 MeshSimplify 类中的 readfile 读取 obj 信息, 并将顶点 (v→Vertex) 和面 (f→Face) 信息存入对应列表, 再根据顶点和三角面片的信息计算出边 (Edge) 的信息。最后再根据极值性质计算每个面的二次度量误差矩阵, 这些矩阵最终由各顶点维护。

Algorithm 1 初始化 obj 文件信息——对应 MeshSimplify 类中 readfile 函数,Face 类 calMatrix 函数

Input: obj 文件, 顶点、边、面列表

Output: 是否读取成功

function calMatrix($Face$)
    $normal \leftarrow e1 \times e2$
    $normal \leftarrow normal/normal.length$
    $temp[] \leftarrow normal.x, normal.y, normal.z, normal \cdot Face.v1$
    $Matrix[i][j] = temp[i] * temp[j]$
    return $Matrix$
end function

8: 根据 obj 的 v 和 f 区分点面信息, 读取并加入列表
9: $scale \leftarrow$ 记录点坐标中距离原点最远的分量, 以便后续 OpenGL 进行显示
10: $ori \leftarrow$ 记录中心点, 便于 OpenGL 显示在中心位置, 避免有的 obj 偏移原点较多
11: 根据三角面片信息, 计算一个面的三条边
12: 计算每个面的矩阵 $\leftarrow calMatrix$
13: 将每个面的矩阵加到各点, 由点维护
14: **return** True

(二) 网格简化处理

1. 首先对每条边计算其边坍塌的误差值, 一条边的误差值就是两个顶点误差值的加和, 根据极值性质, 求解线性方程即可, 若有解, 则唯一解是最优解, 若无则取顶点之一或中点;

Algorithm 2 计算边坍塌误差值 cost——对应 Edge 类 calCost 函数

Input: 边两点维护矩阵 M
Output: 边坍塌 $cost$ 值
1: 解线性方程组 $Mv = (0, 0, 0, 1)^T$
2: **if** 方程组有解 sol **then**
3: $cost \leftarrow sol$ 各分量乘积之和
4: **else**
5: 取边中点作为 sol 并计算 $cost$
6: 同时根据 $cost$ 值选取最优合并点的位置
7: **end if**
8: **return** $cost$

2. 循环处理网格简化, 根据简化比例计算出总共需要退化掉的面数, 该值为 0 时跳出循环;

Algorithm 3 网格简化——对应 MeshSimplify 类 simplify 函数

Input: 还需删除的面数 $rFace$, 点、线、面列表
Output: 新的点、线、面列表
1: **while** rFace>0 **do**
2: 选取 $cost$ 最小的边进行删除 (坍塌)
3: 根据点矩阵选出坍塌后边两点退化到的同一点 v'
4: 将原先 $v1$ 和 $v2$ 构成的线坐标全部更新到 v', 并对 v' 边列表排序去重构成修正列表
5: 对面列表去重排序
6: 根据已经坍塌后的点边列表对面列表进行更新, 如对不存在的面的删除 $rFace--$
7: 对点进行重排编号, 并根据点编号重写面列表
8: **end while**

(三) 结果输出及显示

1. 输出 obj 文件

按照 obj 文件格式将简化后的顶点列表以及三角面片列表写入文件即可 (MeshSimplify 类 writefile 函数)

2. OpenGL 显示结果

Algorithm 4 OpenGL——对应 main.cpp 的 OpenGL 部分函数

Input: 简化后 obj 文件 *out*
Output: 图形界面

1: 读取文件 *out* 并获取点面列表
2: OpenGL 初始化
3: 加入鼠标键盘控制函数, 借助 OpenGL 库实现, 鼠标可以平移、旋转、缩放; 键盘输入 1、2、3 对应不同的显示模式
4: 使用 OpenGL 画图函数将三角面片以及面间连线画到 OpenGL 界面

五、 程序演示

或者可以直接查看我们的演示视频

- 基本操作界面

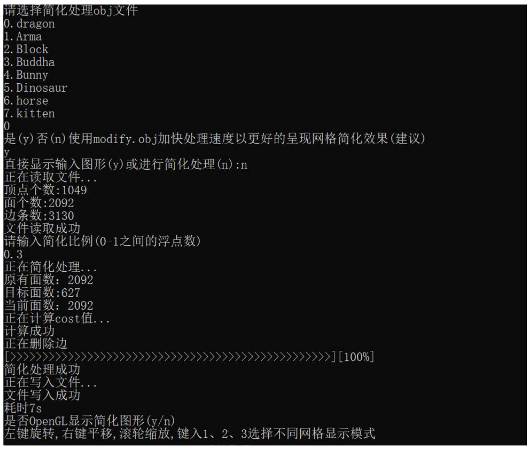

图 3: 基本操作界面

- 直接显示未经简化模型

图 4: 未简化显示

- 简化后模型显示 (简化比例 0.3) 以及左键拖动旋转模型显示角度

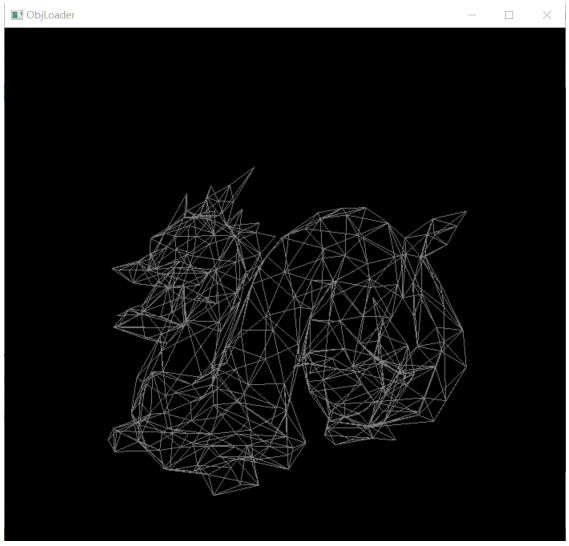

图 5: 简化后显示, 左键旋转

- 右键拖动平移模型

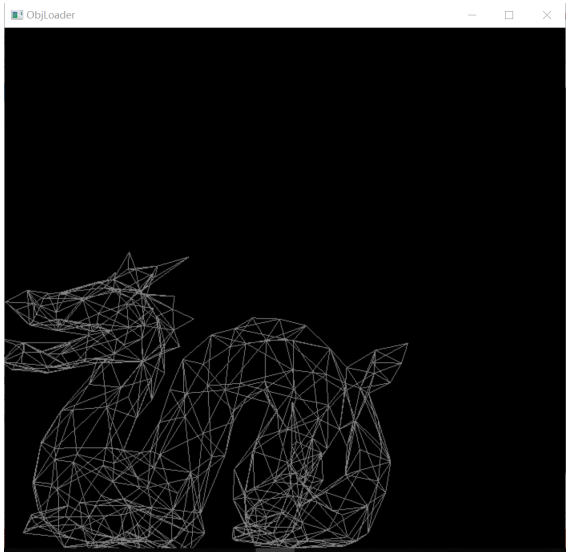

图 6: 右键平移

- 鼠标滚轮滚动缩放

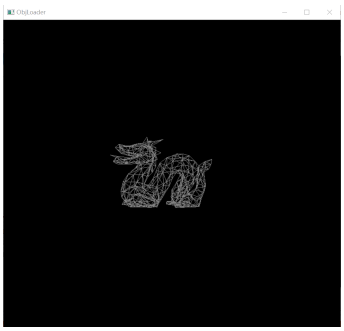

图 7: 鼠标滚轮缩放

- 三角面片着色显示 (type 1)

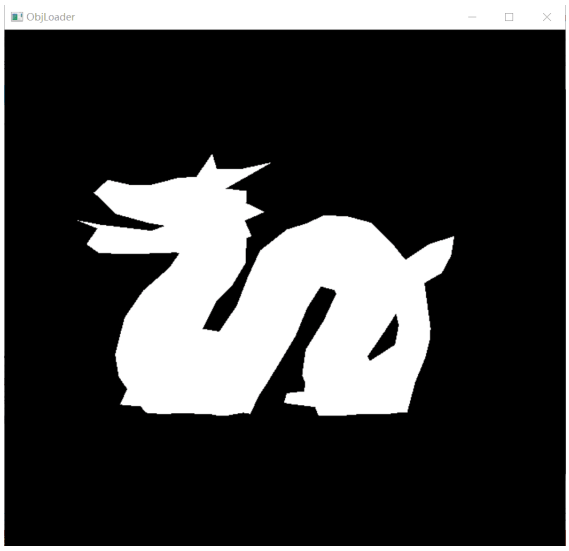

图 8: 面着色显示

- 三角面片着色显示并显示线条 (type 3)

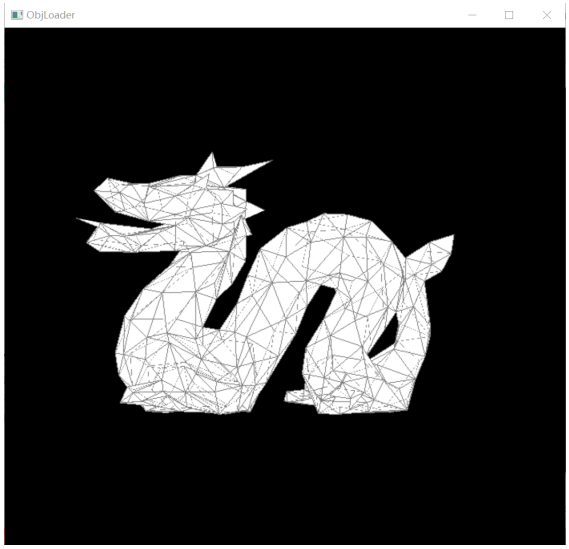

图 9: 面着色 + 线显示

六、 总结

网格简化算法是相对简单的一个大作业选项, 由于对其他两个方面知识的不完善, 我们选择用网格简化练手, 更多的熟悉不用 OpenGL 这种现成接口的纯 C++ 实现, 在网格简化的实现过程中就对图形学的世界观和一些基本概念有了更深层次的理解, 这为我们之后的实现 Phong 模型又提供了宝贵的经验。当然实现的过程中也遇到了不少坑, 多亏了老师的讲稿理解了网格简化边坍塌的算法, 又借助 Google 才最终完成了这次作业。由于这项大作业升级功能有限, 所以我们只实现了 OpenGL 显示界面的平移旋转缩放等, 并没有进一步加深也没有加速处理, 所以较多三角面片的 obj 文件的处理将会比较漫长。