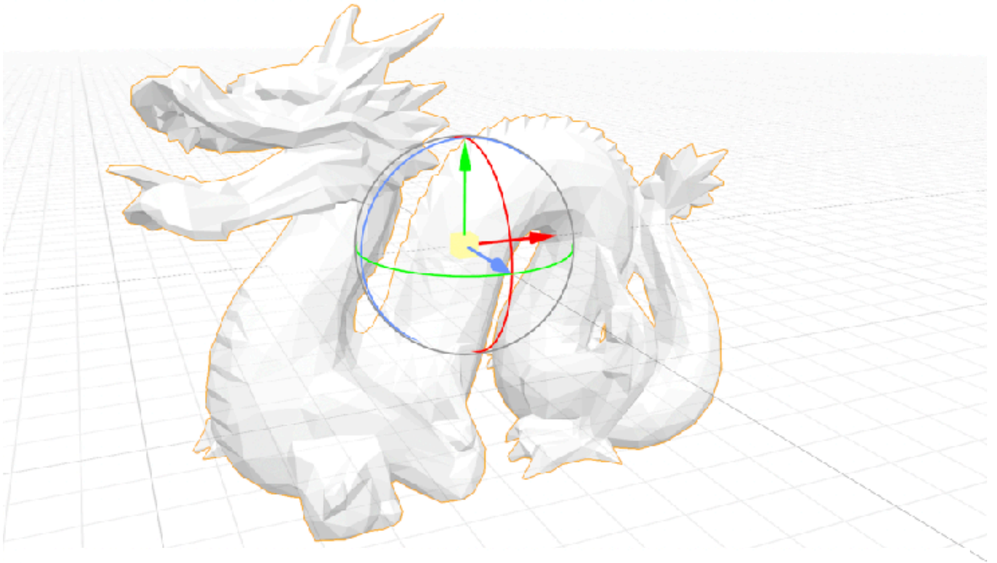
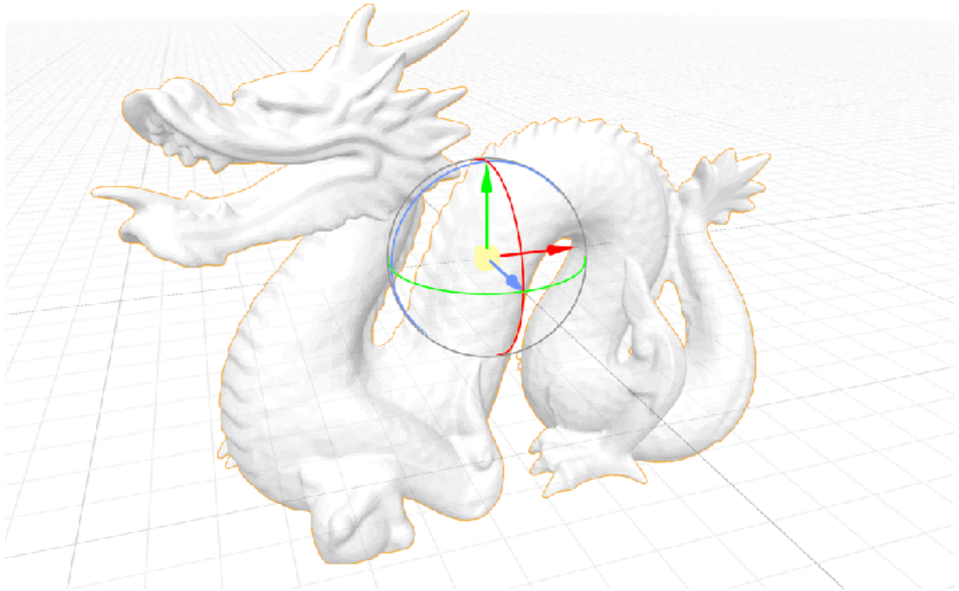
网格简化和重新网格化

冯灏

第一部分
网格简化

使用方法：基于QEM的边收缩方法





大致流程

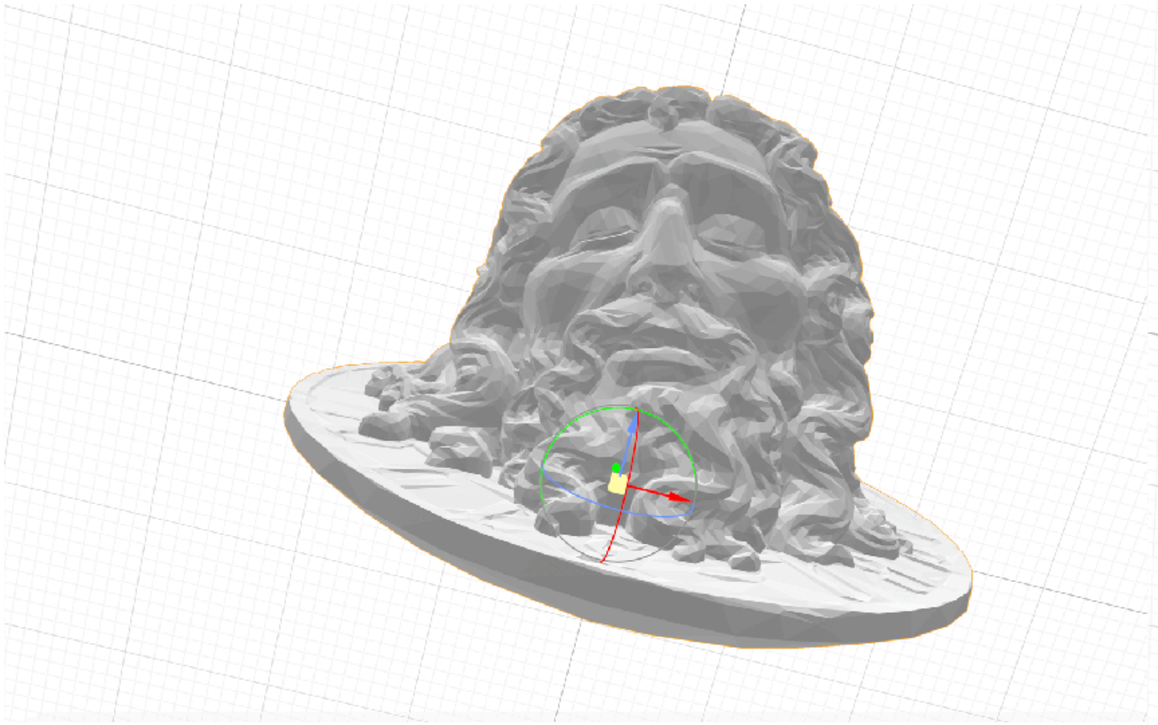
- 计算所有初始顶点的Q矩阵
- 选择所有的有效对
- 计算每个有效对的收缩目标及成本
- 将有效对放入按成本排序的堆中
- 反复删去最低成本的有效对

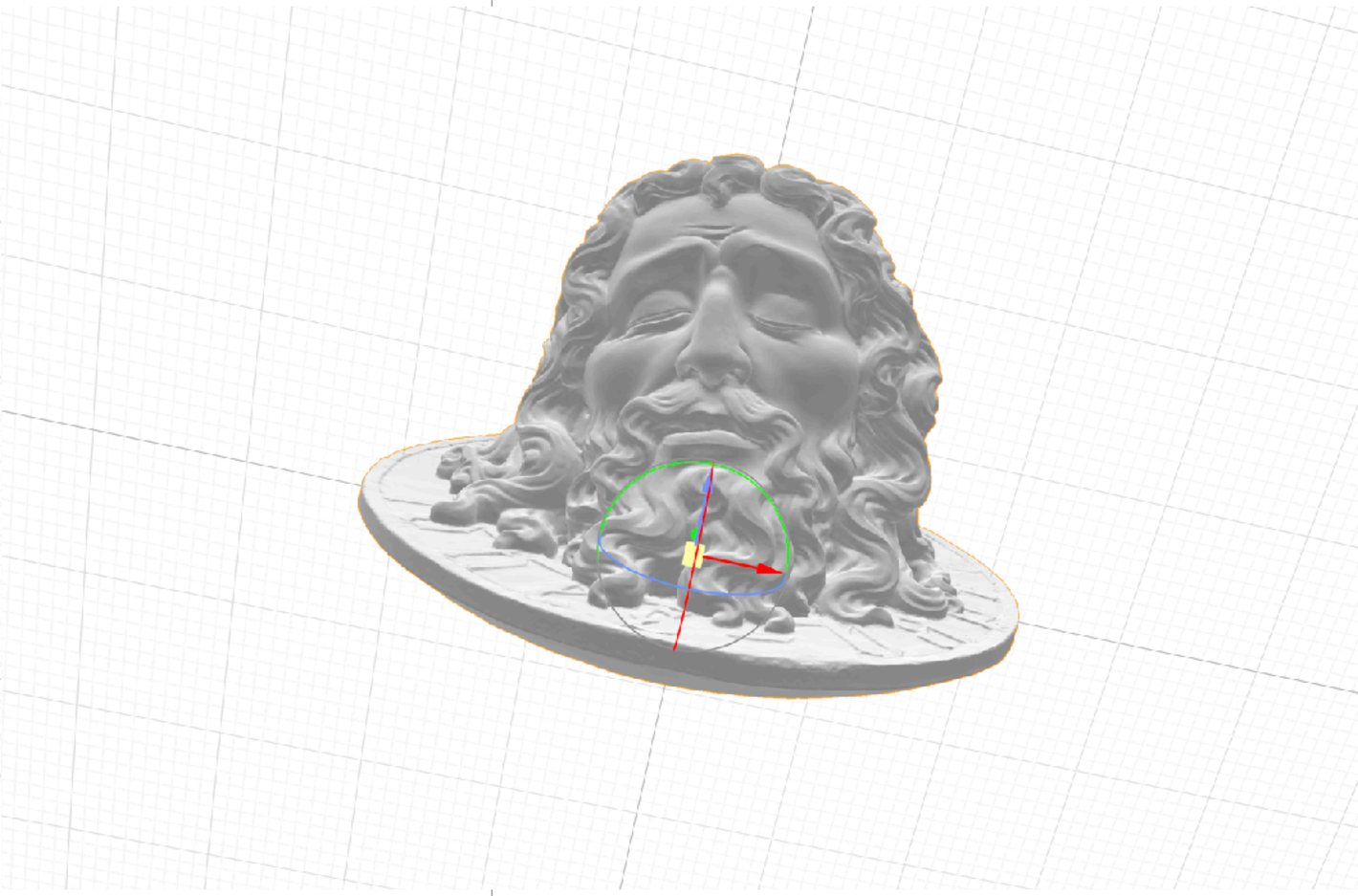
困难的地方

- 算法方面：　　收缩边时，（v1,v2）相连的点都需要更新他们的Q，有v1,v2的面也需要更新Kp，Q是Kp之和。

- 编程方面：　　与算法部分一致，在更新时非常繁琐。为了简洁，我没有去删去旧的而是使用tag与disable来进行标记。无论是点，边还是面均是如此。

- 优点方面：　　考虑了拓扑的特点，当收缩一条边会大幅改变周围的平面时，不去更改。flip函数，可以使在较大的简化情况下避免过度的错误。



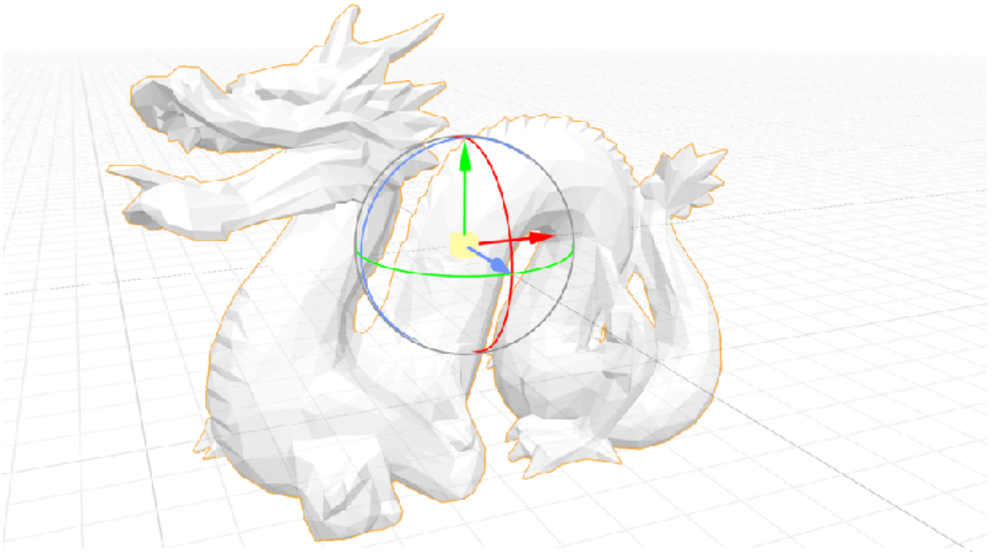


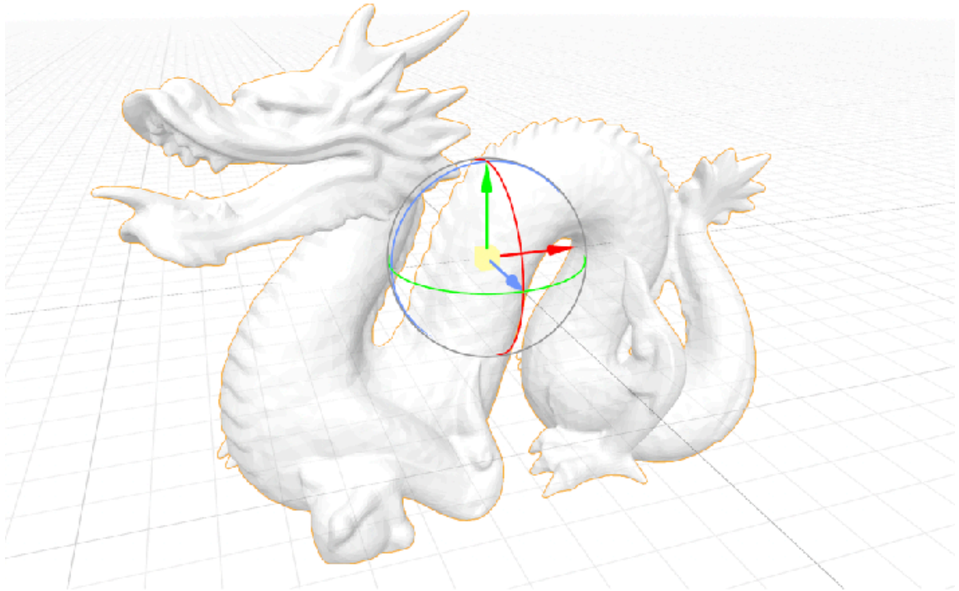
左图为点数为原来1.25%的网格

右图为原图

第二部分
重新网格化

使用方法：Delaunay 三角剖分



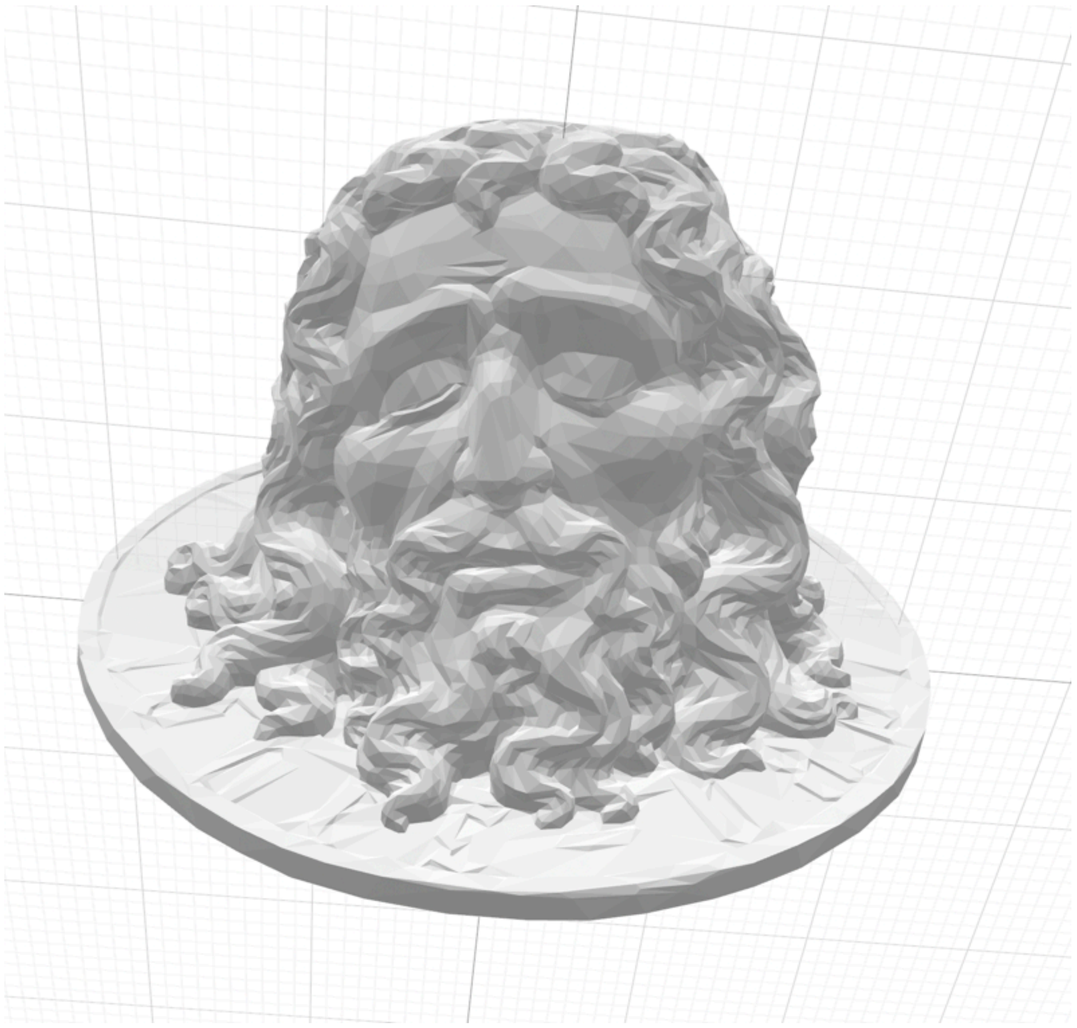


大致流程

- 构造最初的四面体
- 执行插入点的过程Lawson算法
- 部分四面体的外接球四面体单元不再符合Delaunay属性，则这些四面体单元被删除，形成Delaunay空洞，重新构造四面体
- 加入所有的点
- 删去不需要的四面体的面，如不可见的面

困难的地方

- 算法部分：　　当Delaunay空腔需要更新时，如何重新划分比较困难，从二维到三维的复杂度变高了
- 编程部分：　　与网格简化相比，操作的元单元从边与面变成了体，所以需要更好的抽象，使用了更多的类。
- 不足：这部分程序在移植后不能运行了……在读入点的时候，从使用vertex对象转换为点池，重构后跑不起来了。
- 在寻找新插入点在哪一个四面体内时，可以使用一个划分好的网格，或使用类似KD-tree的树形结构，能够加快速度。



重新网格化的结果，因为模型本身性质比较好，所以效果不太明显

- 参考文献：

- Surface Simplification Using Quadric Error Metrics——Michael Garland、Paul S. Heckbert

- http://www.cppblog.com/eryar/archive/2013/05/26/OpenCascade_Delaunay_Triangulation.html

- 任意形状三维物体的Delaunay网格生成算法——王建华 徐强勋 张锐