

# 期末实验报告

姓名：蒋卓洋  
学号：59119125

## 1. 效果展示：

### (1) 初始模型

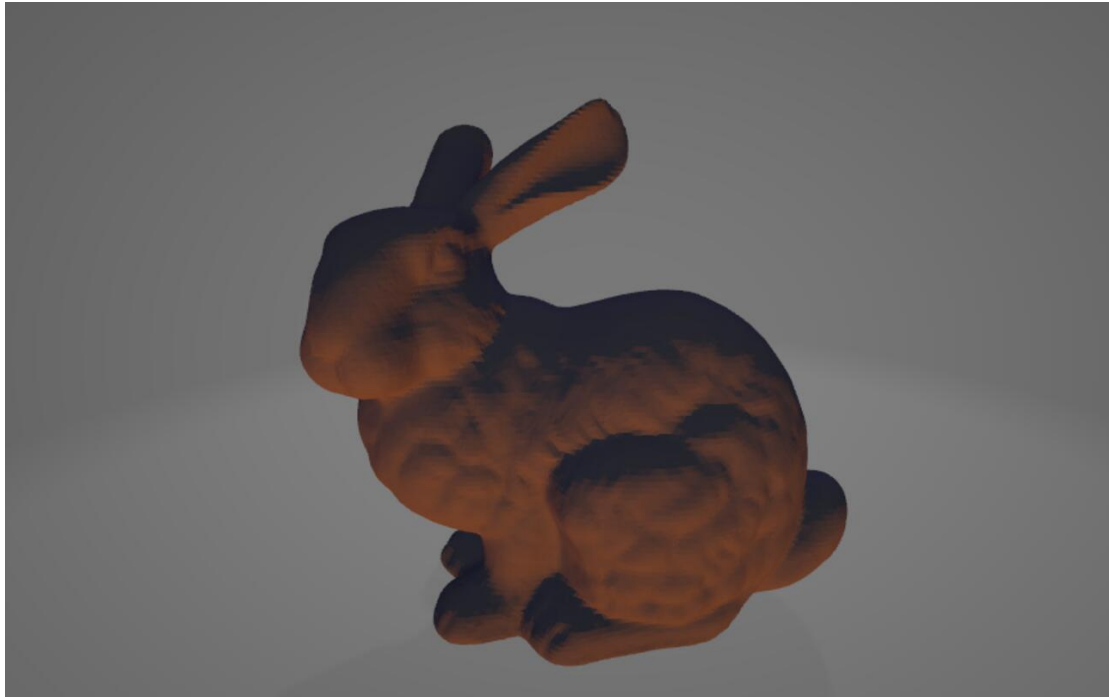


图 1. 初始模型

### (2) 单纯的网格简化



图 2. 简化率为 0.1



图 3. 简化率为 0.05



图 4. 简化率为 0.01

(3) 网格简化并重新网格化

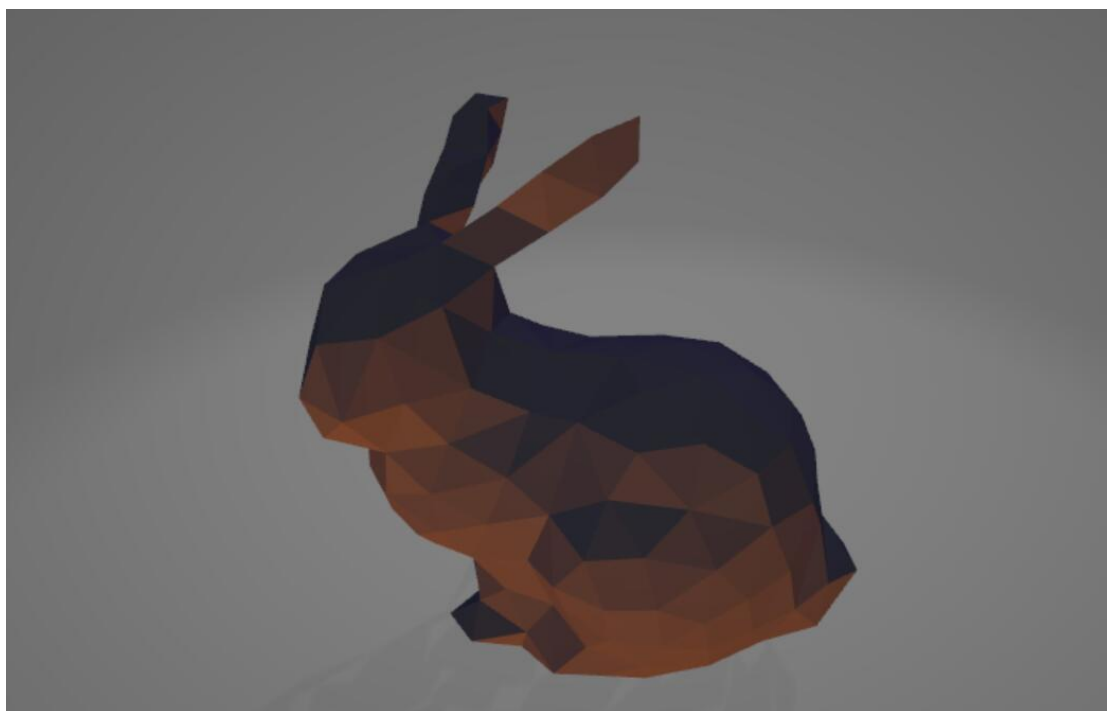


图 5. 顶点聚类数为 200 的重新网格化

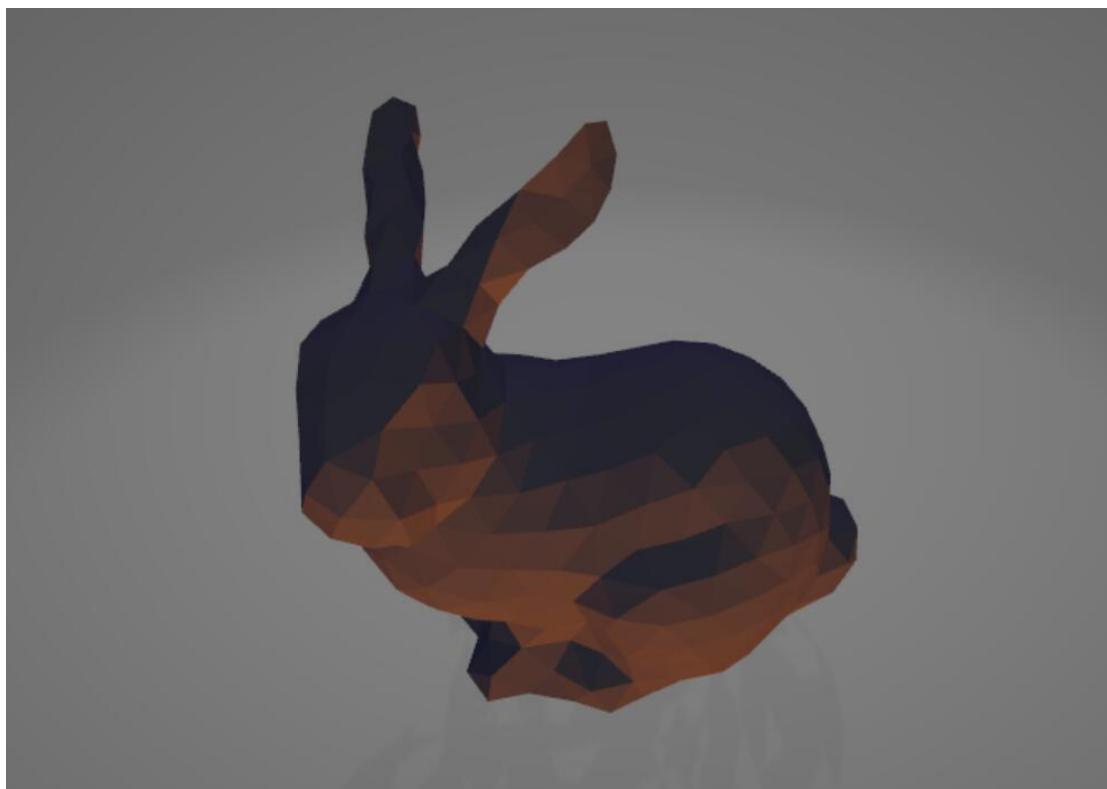


图 6. 顶点聚类数为 400 的重新网格化



图 7. 顶点聚类数为 800 的重新网格化

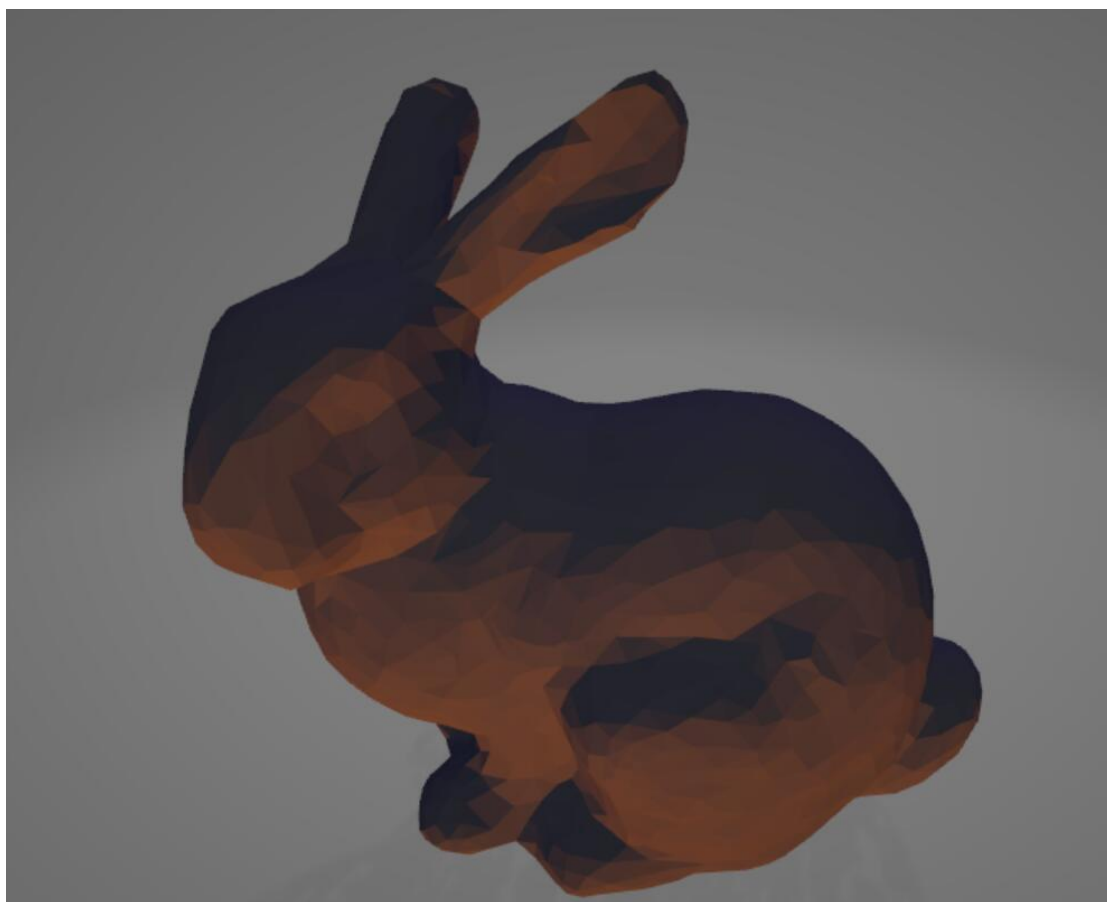


图 8. 顶点聚类数为 1600 的重新网格化

## 2. 任务介绍:

本主题主要有两大任务: 网格简化和重新网格化

### (1) 网格简化:

- ① 动机: 解决模型表示需求与计算资源消耗的矛盾
- ② 思路: 在删减 Mesh 的同时尽量维持原始模型的形状

### (2) 重新网格化:

- ① 动机: 优化模型表示效率与表达能力
- ② 思路: 引入结构优秀的图模型, 用以优化顶点信息和连接关系

## 3. 算法原理:

### (1) 网格简化: 主要采用 Vertex Clustering 算法

#### ① 原理:

- 1) 划分 Grid 和 Cell
- 2) 距离阈值范围内的点聚类为一点
- 3) 在一个 Cell 内进行边坍塌
- 4) 坍塌需要评估坍塌代价, 我们使用 Error quadrics 作为代价函数
- 5) 最小化代价坍塌

#### ② 算法流程:

1. 对所有的初始顶点计算  $Q$  矩阵.
2. 选择所有有效的边 (这里取的是联通的边, 也可以将距离小于一个阈值的边归为有效边)
3. 对每一条有效边  $(v_1, v_2)$ , 计算最优抽取目标  $\bar{v}$ . 误差  $\bar{v}^T (Q_1 + Q_2) \bar{v}$  是抽取这条边的代价 (cost)
4. 将所有的边按照 cost 的权值放到一个堆里
5. 每次移除代价 (cost) 最小的边, 并且更新包含着  $v_1$  的所有有效边的代价

图 9. Vertex Clustering 算法流程

### (2) 重新网格化: 主要采用 BowyerWatson 算法进行 Voronoi 构造

#### ① 原理:

##### 1) Delaunay:

对于任意给定的平面点集, 只存在着唯一的一种三角剖分方法, 满足所谓的“最大 — 最小角”优化准则, 即所有最小内角之和最大, 这就是 Delaunay 三角剖分。Delaunay 三角剖分还满足空圆性质。

##### 2) Voronoi:

给定一群平面 (或曲面) 的点, 其 Voronoi 图, 把平面 (或者曲面) 分隔成一块一块的区域, 每个区域包含一个点, 并且在所有点中, 这块区域所包含的点是这块区域的最近点。

Voronoi 图和 Delaunay 三角化的图, 互为对偶图。

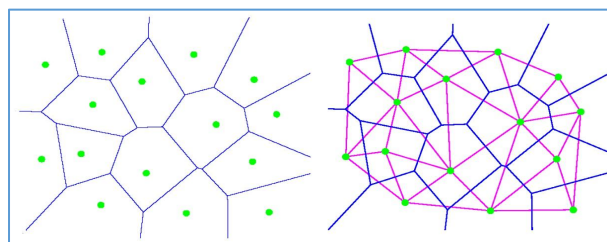


图 10. Delaunay 与 Voronoi 的对偶图关系

## ② 算法流程：

1. 假定已生成了连接若干个顶点的 Delaunay 三角网格
2. 加入一个新的节点，找出所有外接圆包含新加入节点的三角形，并将这些三角形删除，形成一个空腔
3. 空腔的节点与新加入的节点连接，形成新的 Delaunay 三角网格
4. 不断循环直到遍历完所有点

图 11. BowyerWatson 算法流程

其中对于初始化三角网络，我们采用窗口的四个顶点生成两个三角网格，完成 Remsh 后将其消去即可，如下图：

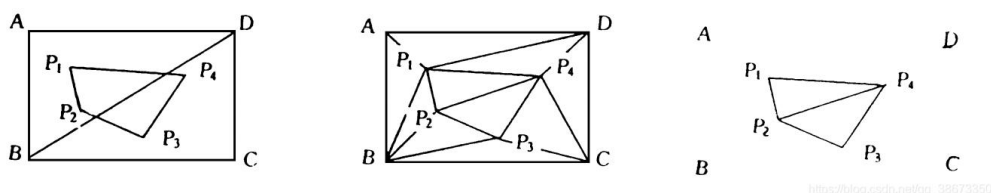


图 12. 初始化三角网格

新增点周围 Mesh 中，外接圆包含新增点的三角形称为影响三角形：影响三角形的查找是本算法的核心，使用暴力搜索来搜索影响三角形。

因为若一个三角形是影响三角形，那么它邻接的三角肯定有两个也是影响三角形，所以我们可以构建三角形邻接表，更快地找到影响三角形。

之后对影响三角形进行边的删除与新空腔生成，即可得到满足 Delaunay 三角形性质的 Mesh 网格。具体可以显示为以下步骤：

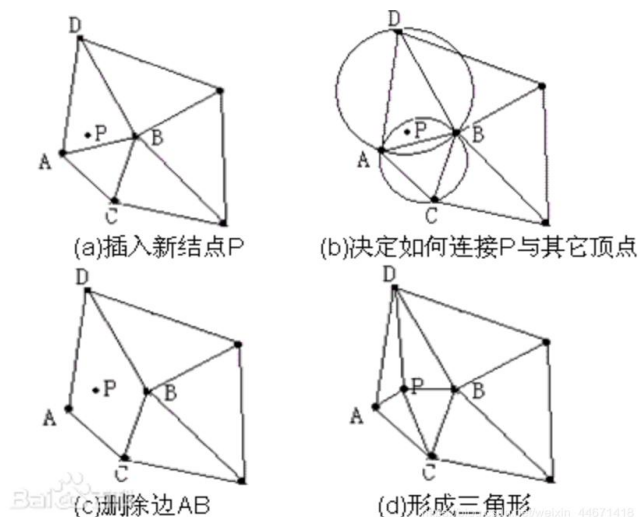


图 13. 每一次新增点的操作

## 4. 调试结果：

具体工具代码见源文件，调试情况如下：

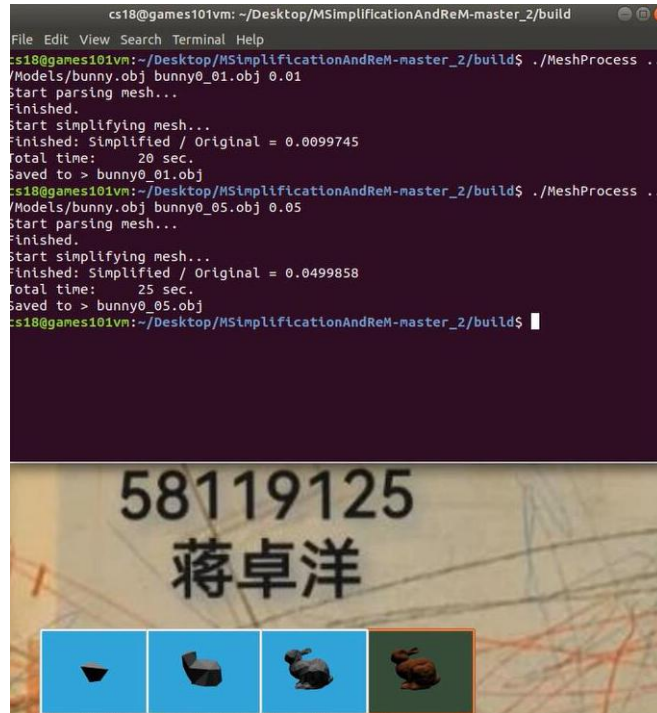


图 9. 网格简化的编译情况

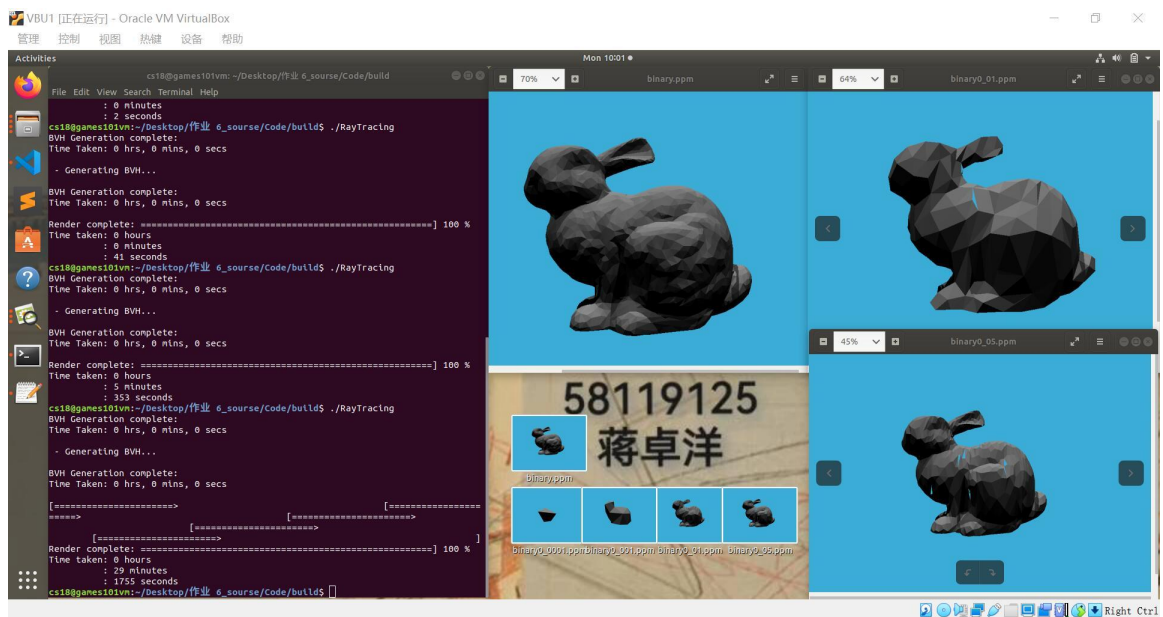
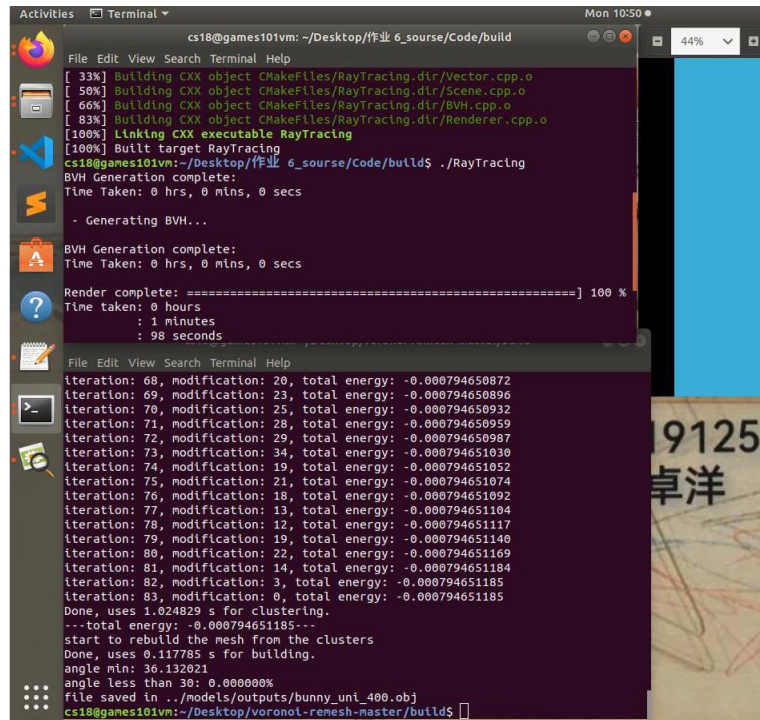


图 10. 网格简化模型的渲染输出





```
cs18@games101vm: ~/Desktop/作业_6_sourse/Code/build
[ 33%] Building CXX object CMakeFiles/RayTracing.dir/Vector.cpp.o
[ 58%] Building CXX object CMakeFiles/RayTracing.dir/Scene.cpp.o
[ 66%] Building CXX object CMakeFiles/RayTracing.dir/BVH.cpp.o
[ 83%] Building CXX object CMakeFiles/RayTracing.dir/Renderer.cpp.o
[100%] Linking CXX executable RayTracing
[100%] Built target RayTracing
cs18@games101vm:~/Desktop/作业_6_sourse/Code/build$ ./RayTracing
BVH Generation complete:
Time Taken: 0 hrs, 0 mins, 0 secs
- Generating BVH...
BVH Generation complete:
Time Taken: 0 hrs, 0 mins, 0 secs
Render complete: =====] 100 %
Time taken: 0 hours
          : 1 minutes
          : 98 seconds
iteration: 68, modification: 20, total energy: -0.000794650872
iteration: 69, modification: 23, total energy: -0.000794650896
iteration: 70, modification: 25, total energy: -0.000794650932
iteration: 71, modification: 28, total energy: -0.000794650959
iteration: 72, modification: 29, total energy: -0.000794650987
iteration: 73, modification: 34, total energy: -0.000794651030
iteration: 74, modification: 19, total energy: -0.000794651052
iteration: 75, modification: 21, total energy: -0.000794651074
iteration: 76, modification: 18, total energy: -0.000794651092
iteration: 77, modification: 13, total energy: -0.000794651104
iteration: 78, modification: 12, total energy: -0.000794651117
iteration: 79, modification: 19, total energy: -0.000794651140
iteration: 80, modification: 22, total energy: -0.000794651169
iteration: 81, modification: 14, total energy: -0.000794651184
iteration: 82, modification: 3, total energy: -0.000794651185
iteration: 83, modification: 0, total energy: -0.000794651185
Done, uses 1.024829 s for clustering.
---total energy: -0.000794651185---
start to rebuild the mesh from the clusters
Done, uses 0.117785 s for building.
angle min: 36.132021
angle less than 30: 0.000000%
file saved in ../models/outputs/bunny_uni_400.obj
cs18@games101vm:~/Desktop/veronoi-remesh-master/build$
```

图 11. 重新网格化的编译情况

## 5. 总结:

在这次实验中, 由于时间紧迫, 压力较大, 我在自己书写代码的过程中遇到了许多难以解决的问题, 比如空指针导致的 Core dump 问题, 我浪费了一些时间在这个问题的解决上, 最后实现的算法也不令人满意。之后我研究学习了 github 上的开源项目的代码结构, 并进行编译得到了较好的结果。

在学习过程中, 我深入了解了网格简化与重新网格化的原理与算法流程, 熟悉了对模型文件的处理过程, 算是有所收获的。