**作业内容：扫描线zbuffer&层次zbuffer算法的实现**

作者：鲁天逸 学号：22121190

编程环境：c++ visual studio 2017， win10 x86

硬件条件 intel i7 10700，Nvidia GTX TITAN X， DDR4 16G

依赖库：c++标准库、glm（仅使用矩阵和向量的基本运算方法）、GLFW&glut（仅使用window创建、键盘控制、像素点的显示等方法）

使用方法：用cmd运行zb.exe，语法为zbuffer [a] [b]

[a] 0 = 扫描线zbuffer

1 = 层次zbuffer，不使用八叉树的简单模式

2 = 层次zbuffer，使用八叉树的完整模式

[b] obj模型的文件名

使用例：zbuffer 2 miku.obj

模型读取成功后，会显示Model load success!以及片面数等信息。

在界面中可以使用[up] [down] [left] [right] [PageUp] [PageDown]来旋转模型，

[Esc]退出界面

源文件说明：为了便于管理，本作业采用了c++面向对象的特性，将方法封装成类。

Mesh类中主要包含从obj文件中读取顶点和面的信息的方法，以及存储面的数据结构。

ScanlineZbuffer类中主要包含扫描线zbuffer算法，分类边表、分类多边形表、以及活化多边形链表和边链表的数据结构。

HieZbuffer类中主要包含层次zbuffer算法，层次四叉树的数据结构以及建立、插入、搜索的方法。

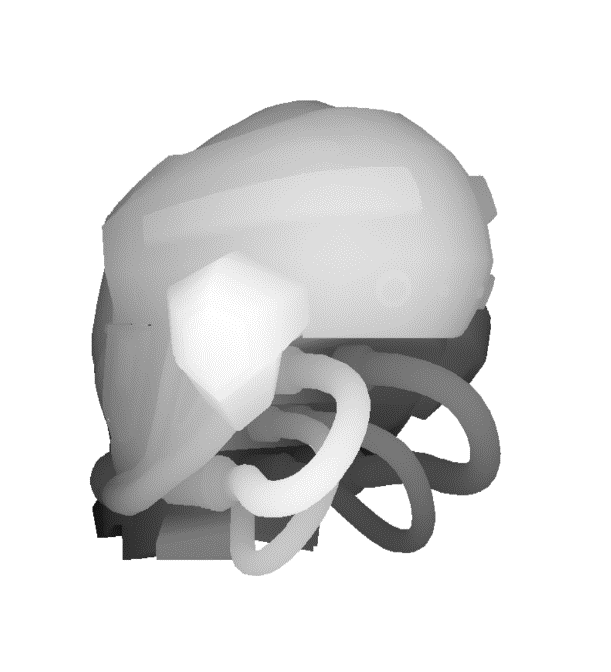
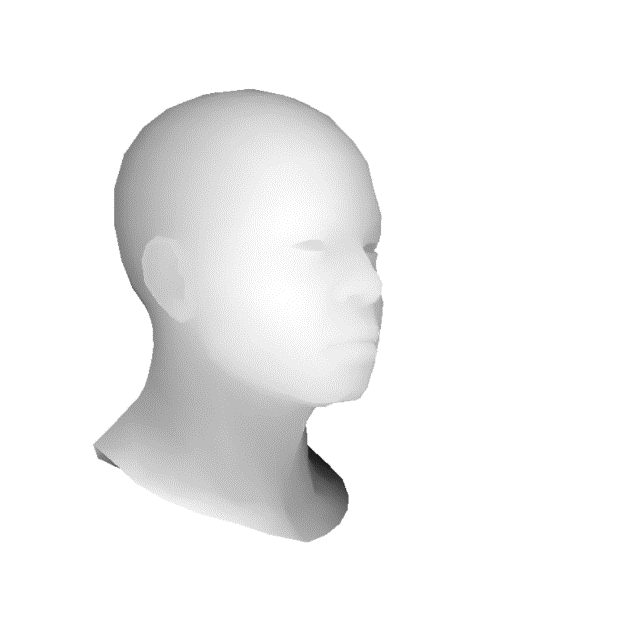
Octree中主要包含场景八叉树的结构以及构造、插入、搜索、析构八叉树的方法。

Camera类中主要包含缩放、旋转、透视等视角转换的方法。

本作业还根据自己理解采用了一些加速方法，比如跳过一些与摄像机几乎平行的面、利用包围盒进行快速剪枝等，还利用了openMP对大量可以并行计算的for循环进行优化，使得运算速度有大幅提升。

实验结果：

1、使用层次zbuffer得到的head.obj/helmet.obj/miku.obj模型结果分别如下，视频可见doc/rotatation.mp4：



2、各方法效率对比

模型的平均渲染时间（ms），水平绕模型一圈取平均值

|  |  |  |  |
| --- | --- | --- | --- |
| 片面数 | 扫描线 | 层次 | 层次+八叉树 |
| head（2492） | 10.19 | 127.7 | 128.8 |
| helmet（15452） | 59.05 | 154.0 | 154.6 |
| miku（174514） | 561.6 | 211.6 | 228.6 |

分析：面数较少时，扫描线算法更具有优势；面数较多时，层次算法更快。

但是我的作业中八叉树没有发挥明显的作用，我分析的原因如下：

1. 八叉树深度较深时（只能在面数较少的模型work，否则内存不够），构建树所需的时间占比较大，最后结果上不理想。
2. 八叉树深度较浅时，拒绝的面就会变少，加速效果也变差，这是为什么在100k面数以上的模型中效果也不好的原因。
3. 没有用八叉树的简单版本，我使用boundingbox来对四叉树进行剪枝，效果上似乎和八叉树差别不大