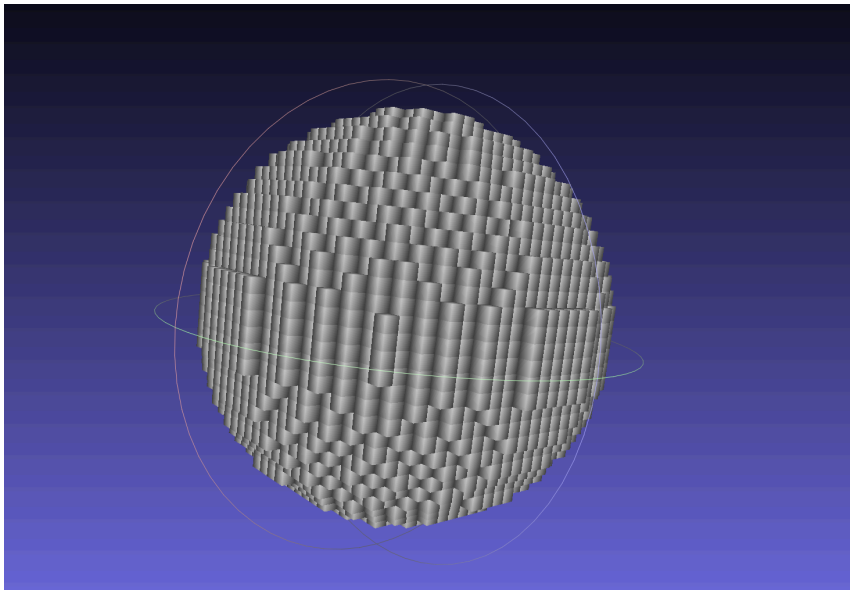
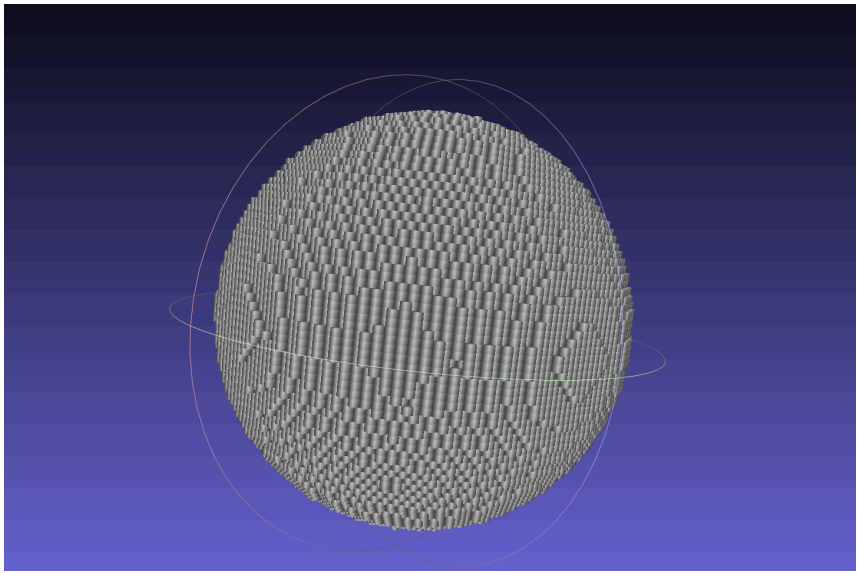
CSC 2521 Assignment 1
Ziheng Liang - 1000393059

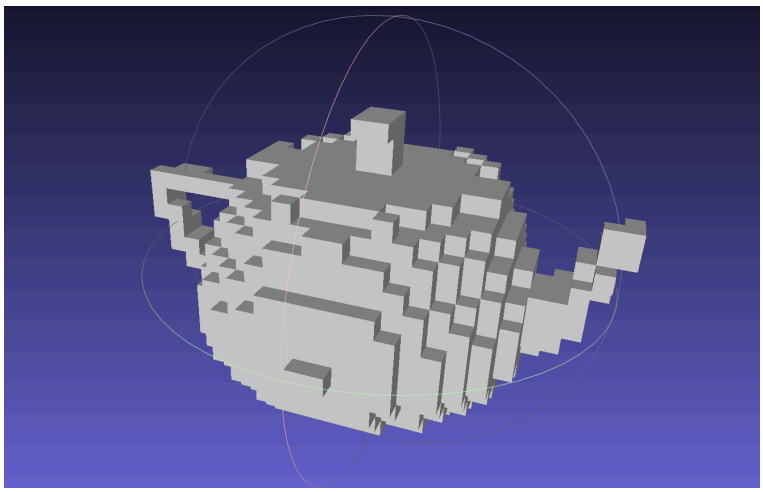
Result:



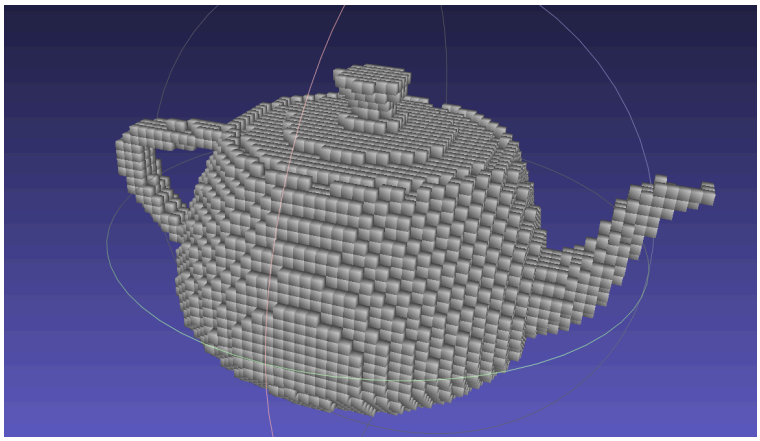
32 * 32 * 32 Sphere



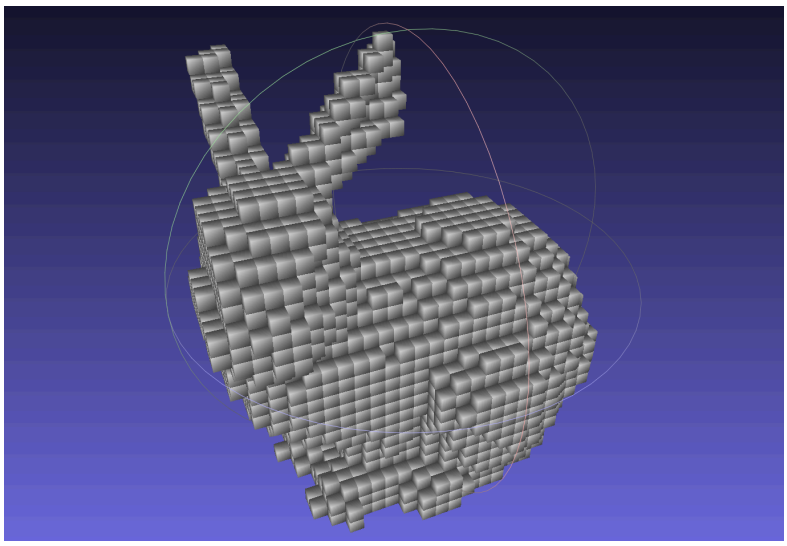
64 * 64 * 64 Sphere



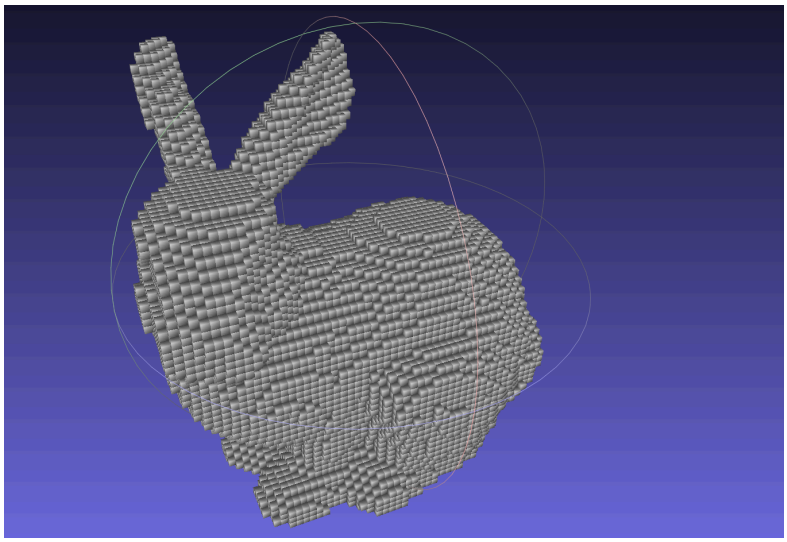
32 * 32 * 32 Teapot



64 * 64 * 64 Teapot



32 * 32 *32 Bunny



64 * 64 * 64 Bunny

Reference:

There are three sources that I used for this assignment:

https://blog.frogslayer.com/kd-trees-for-faster-ray-tracing-with-triangles/
This is the website where I learn about bounding-box and tree structure of a kd-trees.

https://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracer-rendering-simple-shapes/ray-box-intersection
This is the website where I learn about ray box intersection.

Mathematics for 3D Game Programming and Computer Graphics, Third Edition by Eric Lengyel
I used this book to review about triangle ray intersection.

Problem:

When constructing a kd-tree, each mesh is assigned to node base on its mid-point. So, some part of the mesh might lie beyond the edges of the bounding box. I've tried a few possible fixes, but they all have some problems.

Firstly, I've tried to create a new bounding box after all the mesh assignment is done. The approach fails when a mesh is so large that smaller meshes are cover by it. What I meant is that after subdividing the bounding box, the new bounding box after assigning all the mesh might be the same size of the original bounding box before subdividing. This may cause infinite recursion. Even adding more constrain to force the recursion to end, this approach might cause some of the node contain too many meshes.

My second approach is to assign the mesh to both node if it's crossing the cutting edge. Sometimes this approach fails when a bounding box contains only edge cutting mesh. Then the node keeps replicating itself and assign them as children. This can be avoided by adding more terminating constrain, but it also might end up with nodes with large amount of node.

These two approaches above can be surely improve given more time. But the approximation algorithm I am currently using can be very accurate combining with multiple ray casting. Since only a small portion of meshes are cut by boundary and even the cut meshes still have over 50% lie within the bounding box.

Additional Features:

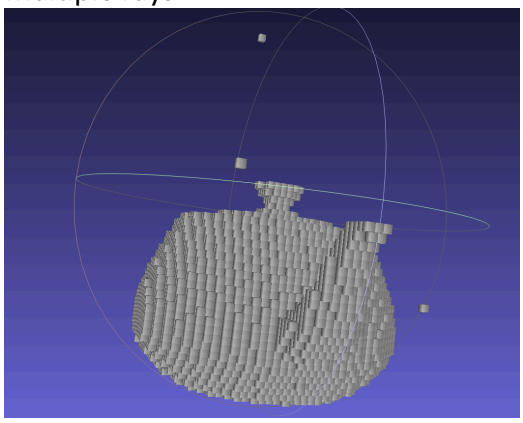
I've include both features in my compiled code. These features are controlled by two variables useTree, multipleRay which are set to true in the executable. To change setting, change the variables on line 229 and 230 in file main.cpp. Both source folder and include folder are required to compile.

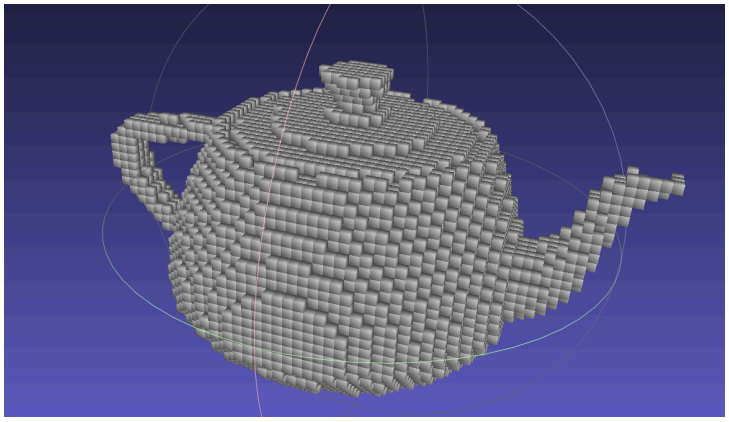
Speed up structure KD-Tree

9 rays at each voxel using tree structure			Not using tree structure
Runtime in sec	32 * 32 * 32	64 * 64 * 64	32 * 32 * 32
Sphere	4.31s	34.24s	66.62s
Teapot	2.37s	19.31s	169.05s
Bunny	8.65s	72.37s	> 20 mins

Speed is significantly fastened. I did not border to run 64 dimensions without tree structure. It's reasonable to assume they are much slower.

Multiple rays





There are random voxel floating around using one single ray. This error is significantly reduced using multiple rays.