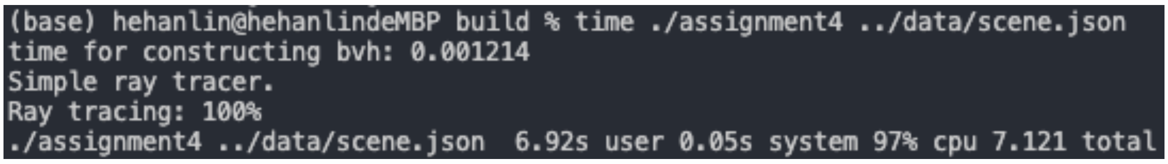
Assignment 4

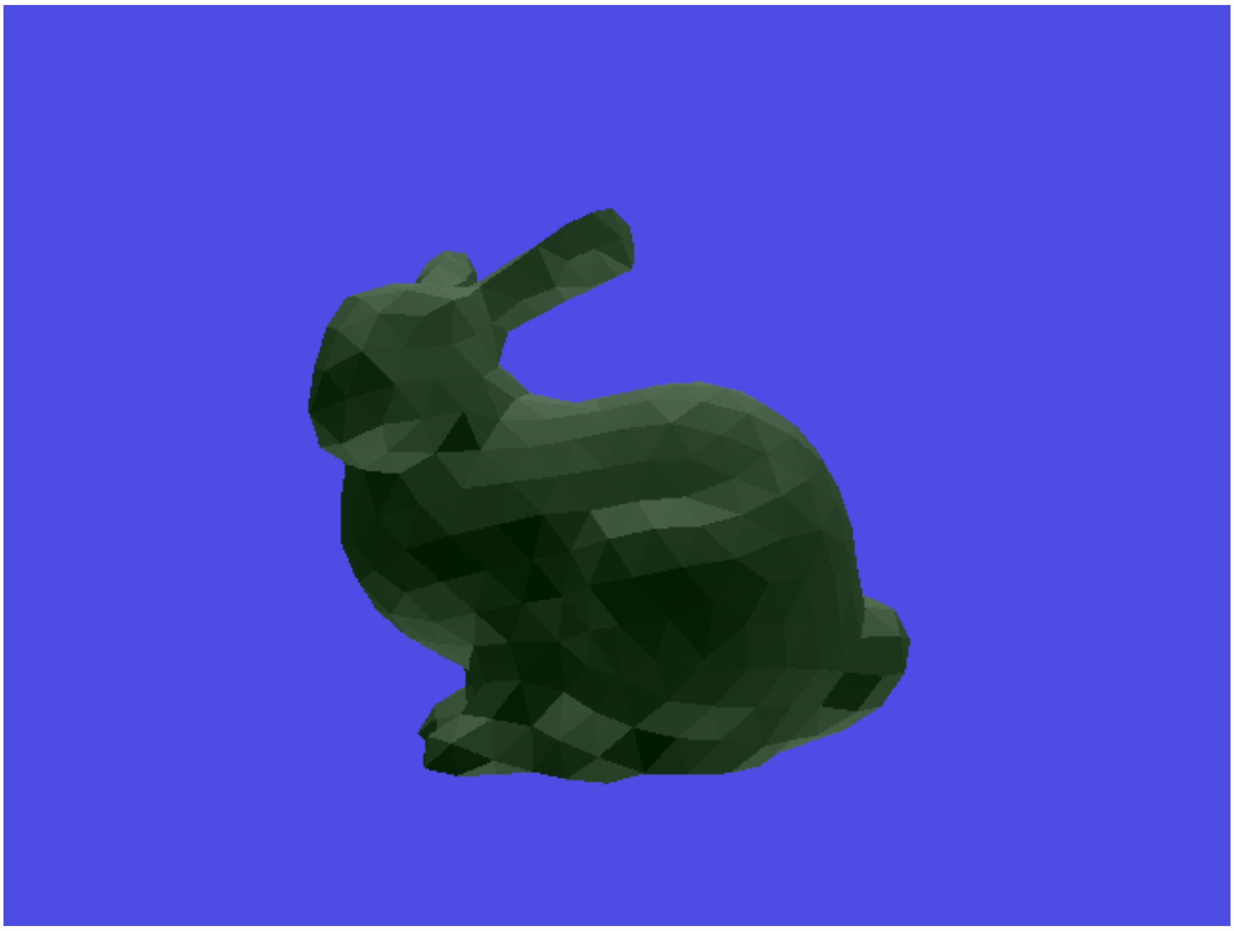
Ex.1

I implemented triangle-ray intersection by solving the system $f(u, v) = p(t)$. As well as the ray_color function, render_scene function and all other related function used in assignment3. However, since we don't need reflection, refraction and depth of field in assignment4, I use a struct "config" to control the enablement of these features.

In the exercise one, I haven't implement a bah. It takes several seconds to produce "bunny", and seems impossible to render "dragon".

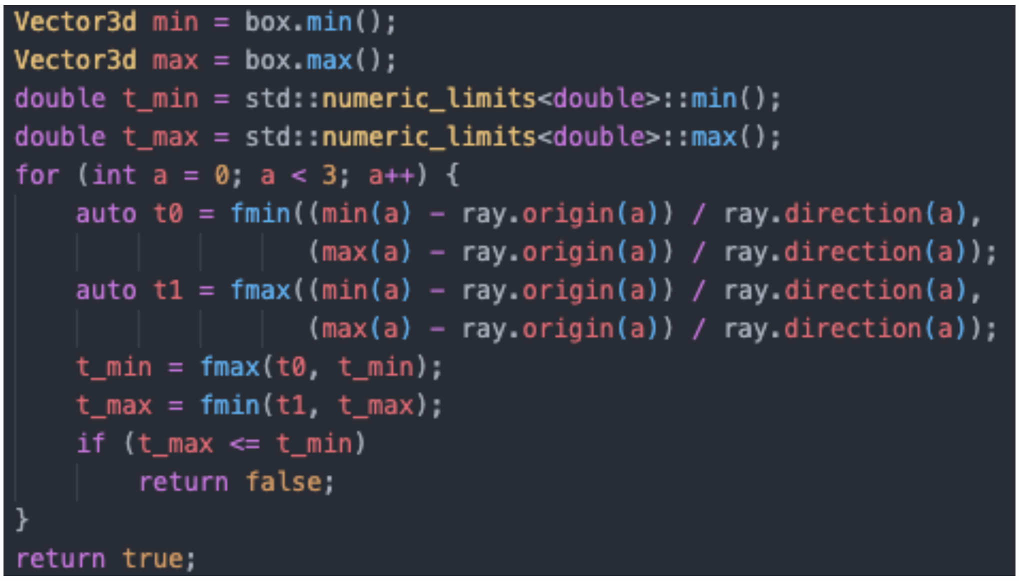


```
(base) hehanlin@hehanlindeMBP build % time ./assignment4 ../data/scene.json
time for constructing bvh: 0.001214
Simple ray tracer.
Ray tracing: 100%
./assignment4 ../data/scene.json  6.92s user 0.05s system 97% cpu 7.121 total
```



Ex.2

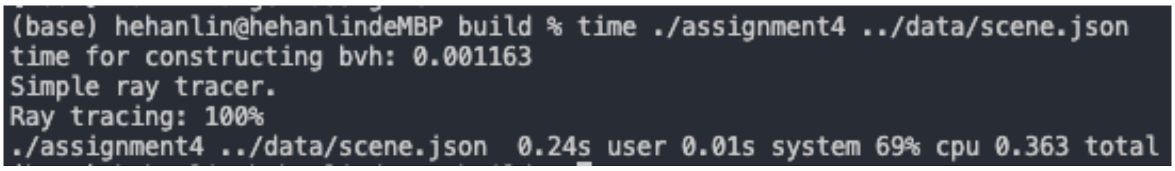
In this exercise, I implement a bvh tree by recursively split them in top-down order. And in each recursion, I order the nodes passed in along the longest axis. For intersect_box function, I implemented a linear solution to find if a ray intersect with the axis-aligned-box, shown as follow:



```
Vector3d min = box.min();
Vector3d max = box.max();
double t_min = std::numeric_limits<double>::min();
double t_max = std::numeric_limits<double>::max();
for (int a = 0; a < 3; a++) {
    auto t0 = fmin((min(a) - ray.origin(a)) / ray.direction(a),
                   (max(a) - ray.origin(a)) / ray.direction(a));
    auto t1 = fmax((min(a) - ray.origin(a)) / ray.direction(a),
                   (max(a) - ray.origin(a)) / ray.direction(a));
    t_min = fmax(t0, t_min);
    t_max = fmin(t1, t_max);
    if (t_max <= t_min)
        return false;
}
return true;
```

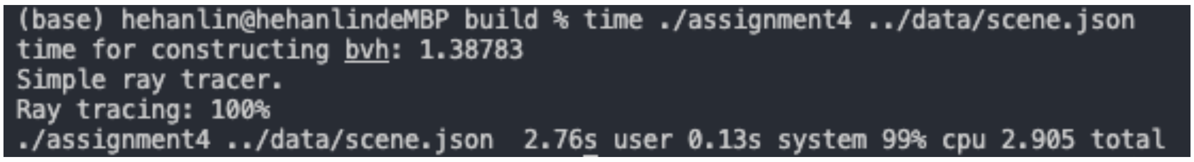
After implementing the bah, I render "bunny" and "dragon" in seconds.

Time cost of "bunny", from 7.12s to 0.363s:



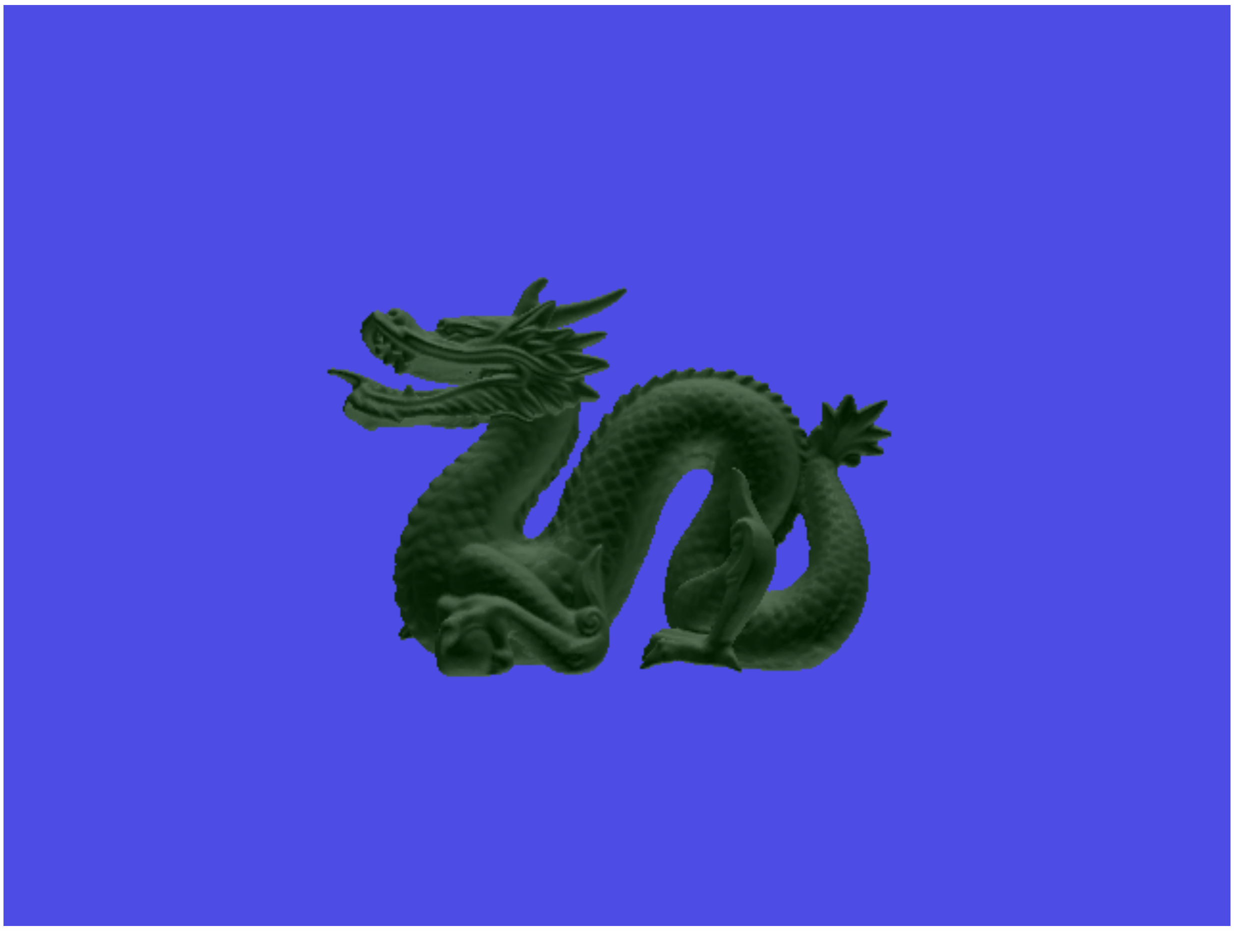
```
(base) hehanlin@hehanlindeMBP build % time ./assignment4 ../data/scene.json
time for constructing bvh: 0.001163
Simple ray tracer.
Ray tracing: 100%
./assignment4 ../data/scene.json  0.24s user 0.01s system 69% cpu 0.363 total
```

Time cost of "dragno":

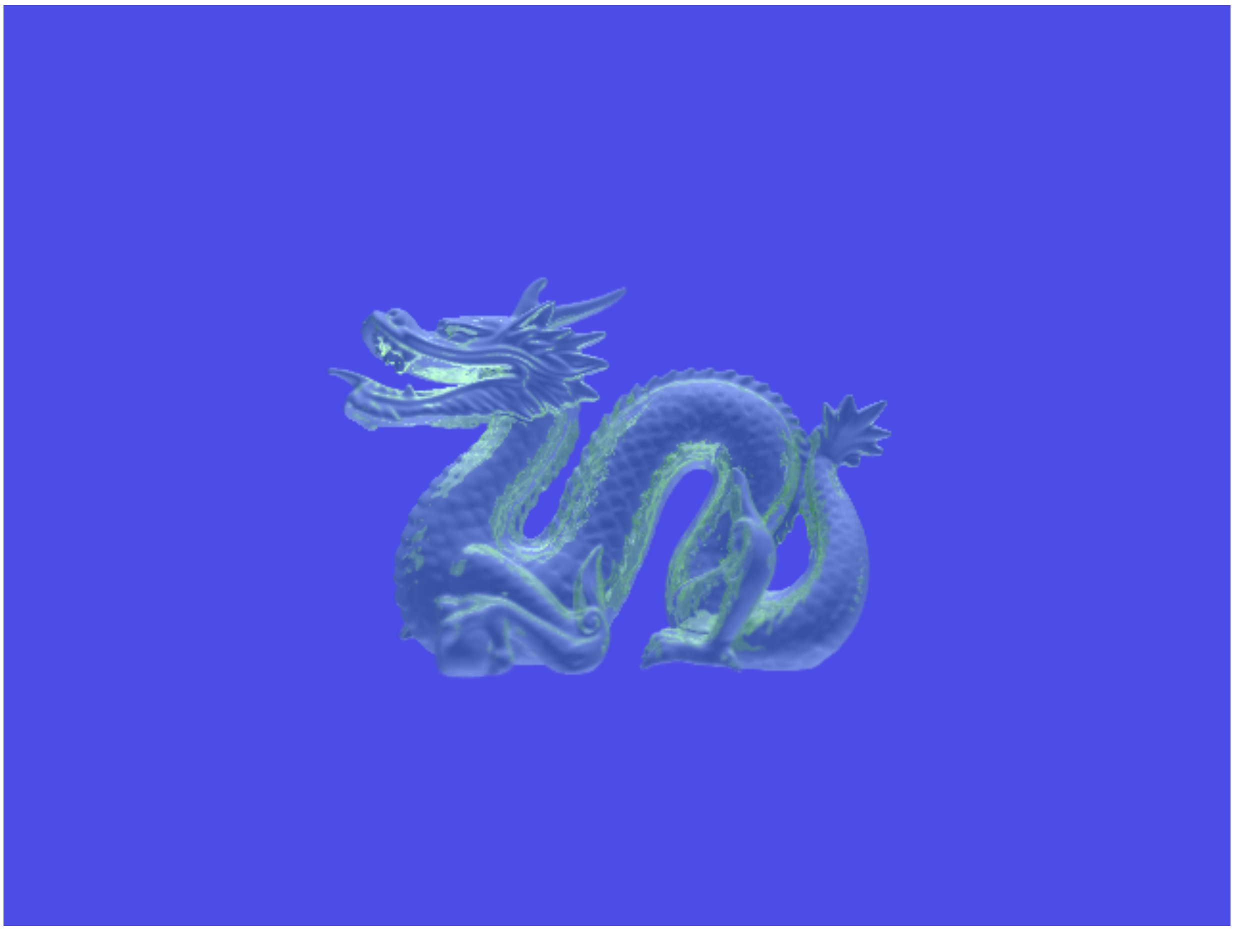


```
(base) hehanlin@hehanlindeMBP build % time ./assignment4 ../data/scene.json
time for constructing bvh: 1.38783
Simple ray tracer.
Ray tracing: 100%
./assignment4 ../data/scene.json  2.76s user 0.13s system 99% cpu 2.905 total
```

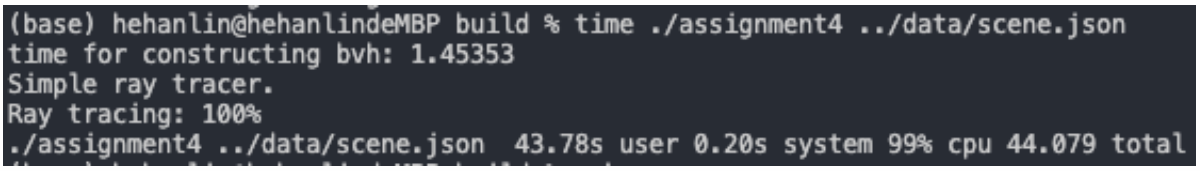
Final result:



If we enable reflection, refraction, and depth of filed, then we get:



Plus shadow, all features enabled. It takes about 40s:



```
(base) hehanlin@hehanlindeMBP build % time ./assignment4 ../data/scene.json
time for constructing bvh: 1.45353
Simple ray tracer.
Ray tracing: 100%
./assignment4 ../data/scene.json  43.78s user 0.20s system 99% cpu 44.079 total
```