# Participating Media Nori-extended Renderer Using Single Scattering Approximation

Saeed Hadadan[1]

***Abstract*— This project extends Nori to support participating media.**

## I. INTRODUCTION

Handling participating media in renderers is of huge importance. Every volume consisting of at least light-absorbing particles can affect any traversing ray through it. These effects include absorption, out-scattering which together account for extinction, meaning that it decreases the radiance. On the other hand, in-scattering along with volume emission increases the passing-through radiance. Summing up all these effects are calculated in the volume rendering equation.

## II. FEATURES SUPPORTED

The following features are supported in this project:

1) Introducing participating medium in the scene desription(.xml file). The corresponding absorption $\sigma_a$ and scattering $\sigma_s$ coefficients and the volume lemit $L_{ve}$ should be passed in that file.(Fig. 1)
2) Homogeneous and hacky cases of heterogeneous media implemented.
3) Henyey-Greenstein is used as the phase function.
4) A new path tracer *volumePathTracer* is developed from the scratch based on pbrt. My previous path tracers had problems, yet this new one is tested at least on Cornell box and it works.
5) Ray marching and single scattering approximation.
6) In addition to a our famous Cornell box, another nice scene is also rendered in different settings of the participating media. This scene is another Cornell box, yet with a different setting.

## III. VOLUMETRIC PATH TRACER STRUCTURE

A very rough highlevel of what the path tracer does is in Fig. 2. Actually, what I implemented here is ray marching and single scattering approximations. Lines 7-8 in Fig. 2 show this. First I spawn a shadow ray to a sampled point on a sampled lighter with certain PDFs. Only area lights are implemented. After we get the proper $L_s$ here, we should divide the current ray into steps and calculate transmittance for it.

This means that we assume that each ray before hitting an actual surface, encounters a certain number of interactions with the participating medium(Fig. 3). (Please suppose that we implemented a simple case of it where the media is present all over the place.) This number of steps are considered to be 10 in my project. We calculate the transmittance through the media in each step via Beer's law and multiply it by the attenuation coefficients we have in our loop. We also include $L_{ve}$ if applicable in this step.

```
1
2  <medium type="homo">
3          <!-- By default, the medium occupies the
           region [0,0,0]-[1,1,1]. Scale and rotate it:
           -->
4          <transform name="toWorld">
5              <scale value="5,5,5"/>
6              <rotate axis="0,1,0" angle="45"/>
7          </transform>
8
9          <!-- Absorption and scattering coefficients
           -->
10         <color name="sigmaA" value="0.01,0.01,0.01"/>
11         <color name="sigmaS" value="0.4, 0.4, 0.4"/>
12         <color name="Lve" value="0.2, 0.5, 0.6"/>
13
14
15         <!-- Instantiate a Henyey-Greenstein phase
           function -->
16             <!-- Configure as slightly
               forward-scattering -->
17             <float name="g" value="-0.3"/>
18     </medium>
19
```

Fig. 1. Medium tag structure in .xml file.

Another point in line 7 is that inside the sampler function, not only do we calculate the phase function and multiply it by $L_s$, but also we use another possible ray marching for the shadow ray to the light source. This step may be unnecessary for homogeneous media, but for the heterogeneous it is useful to march the shadow ray to the source as well (Fig. 4).

### A. Ray Marching

Ray marching is implemented for a general case, but it can be used for marching both original rays and shadow rays. The steps in ray marching are considered to be 10 in this project. However, as in Fig. 4, when rayMarch() is used for shadow rays in a homogeneous medium, there is no need to have multiple steps(Fig. 5).

### B. Homogeneous vs. Heterogeneous

There are two main implementation differences for homogeneous and heterogeneous.

1) $\sigma_s$ and $\sigma_t$ are constant for homo and change by a density function for hetero(see Fig. 6 for an example).

*This work is a project for CMSC740: Advanced Computer Graphics

[1] S. Hadadan is a student at the University of Maryland, saeedhd@terpmail.umd.edu

```
initialize some variables;
for (k = 0; ; k++) {
    scene->rayIntersect(ray, its);
    if (!foundIntersection || maxDepth) break;

    if (handleSingleScattering) {
        add sampleLighterAndReturnL(scene,medium);
        find Tr in rayMarch(its,medium,n);
    }
    if ( specularBounce || k ==0){
        add Possible Path Vertex emmission;
    }
    add sampleLighterAndReturnL(scene);

    if(sampleBSDFOut(w_i, ray)) break;

    if(russianRoulette(k, q)) break;
}
```

Fig. 2. Volume path tracer: Li() function high level pseudocode.

```
if (handleSingleScattering) {
    Color3f L_s =
    sampleLighterAndReturnL(...);

    float dt = (its.t - newRay.mint)/n;
    Color3f L_raymarch =
    rayMarch(...);
    L_raymarch*=dt;
    L += L_raymarch+alpha* L_s;
}
```

Fig. 3. Lines 6-10 in Fig. 2. Handling interaction with medium.

```
Color3f sampleLighterAndReturnL
(..., medium) const {

    sampler an emmiter;
    sample a point on it;
    spawn a shadow ray;
    if(hits) L = Le;
    if (inMedium) {
        if (homogeneous medium)
            steps=1;
        rayMarch(steps);
        phase = medium->p(wi, wo);
        L*= medium->sigma_s* Tr * phase;
        L/= pdfs;
```

Fig. 4. sampleLighterandReturnL() highlevel implementation for in medium interactions. This function has another not-shown part. The trivial operations that we did for surface intersections using bsdf values and pdfs are not shown, but they are as else to this if.

2) Shadow ray marching is not done with multiple steps for homo media.

## IV. Results

Here are the results. For the first case, I tested volPath tracer to render our famous Cornell box, see Fig. 7.

```
Color3f rayMarch(n, &L_ve, &Tr) {
    L = 0.f;
    float dt = (its.t - ray.mint)/n;
    for (int i = 0; i<n ; i++){
        L+=Tr*L_ve;
        Vector3f d = ray(dt*i);
        Color3f cS = sigma_t*(density(d));
        Color3f cT=1-cS*dt;
        Tr*=cT;
    }
    return L;
}
```

Fig. 5. Ray march(). For shadow ray marching, the passed $L_{ve}$ is zero. The d is the actual 3D coordinate of the point on the ray corresponding to the current step.

```
float density(p, isHomo) {
    if (isHomo)
        return 1.f;
    Vector3f range = scene->getBoundingBox().max
    -  scene->getBoundingBox().min;
    Vector3f dis = (p - scene->getBoundingBox().min);
    float d = sqrt(dis.dot(dis)/range.dot(range));
    float f = abs(sin(15*d*M_PI));
    return f;
}
```

Fig. 6. density(). In a homogeneous media, the density function always return 1. For a heterogeneous medium it calculates a density value. This function used is a sin function with $7.5\pi$ period. This is because I wanted to model water waves.

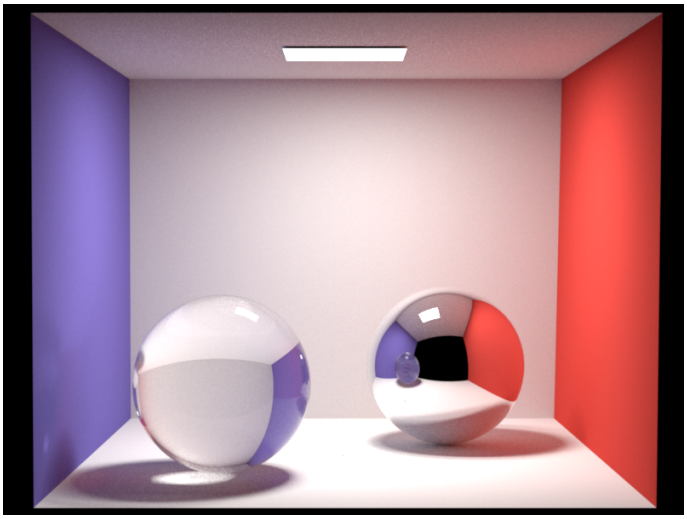

Fig. 7. Cornell box. No participating media. correctL = 0.2

Then, cbox with a homo medium in Fig. 9.

I created a scene in Blender and used that as my tester. It is another instance of cbox, but this the front and right wall, the ceiling, and the floor are mirrors. The two boxes are opaque, but the icosphere is dielectric 10. The scene is a little vague, due to not having any illumination or object on the camera side and the mirrors make it hard to tell walls apart. Let's

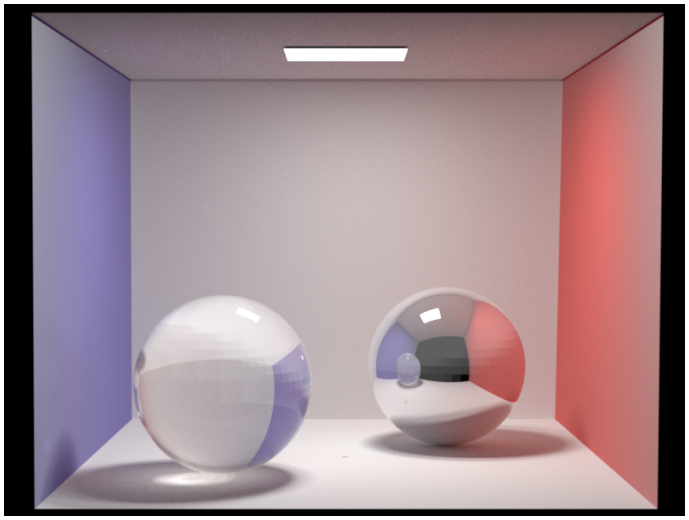

Fig. 8. An intermediate result for Cornell box. Homo medium with arbitrary parameters. Please do not depend on this, since it is a little old, I changed the code furthur.

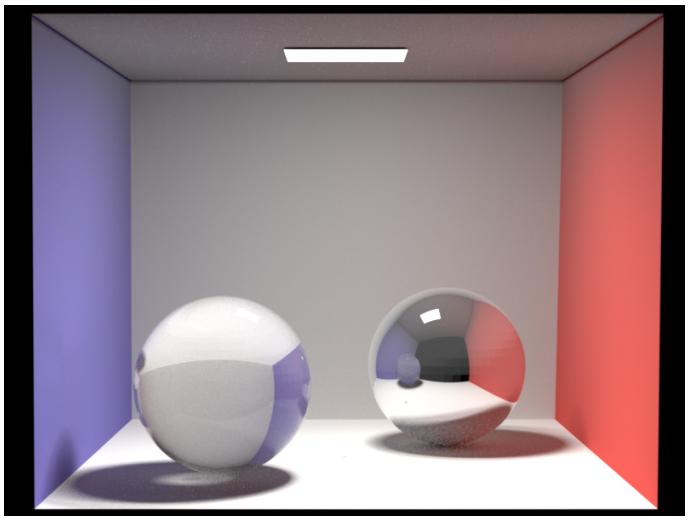

Fig. 9. Cornell box. Homo medium with $\sigma_s = 0.4$, $\sigma_a = 0.01$ and $g = -0.3$. correctL = 0.2.

add some atmosphere to it, so we can see something in the following Figs. 11, 12, 13, 14.

## V. Challenges

The first challenge which took most of my time was developing the path tracer again. My previous path tracers were problematic. I noticed my sampleLighterAndReturnL was using wrong PDFs the whole time.

Another major problem I had was distance dependant measure in the integrator. For example, geoTerm was a correction term we used to account for change in the integral variables. However, the definition has a division by length and it is totally length dependent, so in these two cboxes due to huge difference in dimensions, it was problecmatic to accomodate poth of them in the tracer. I assumed x-y is normal, so it was 1 and the problem was solved. However,

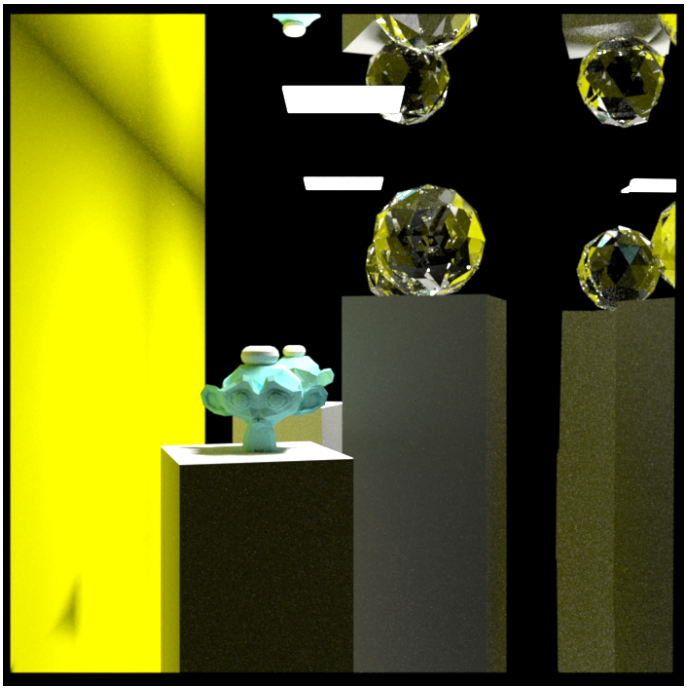

Fig. 10. My scene. No participating media. correctL = 0.2

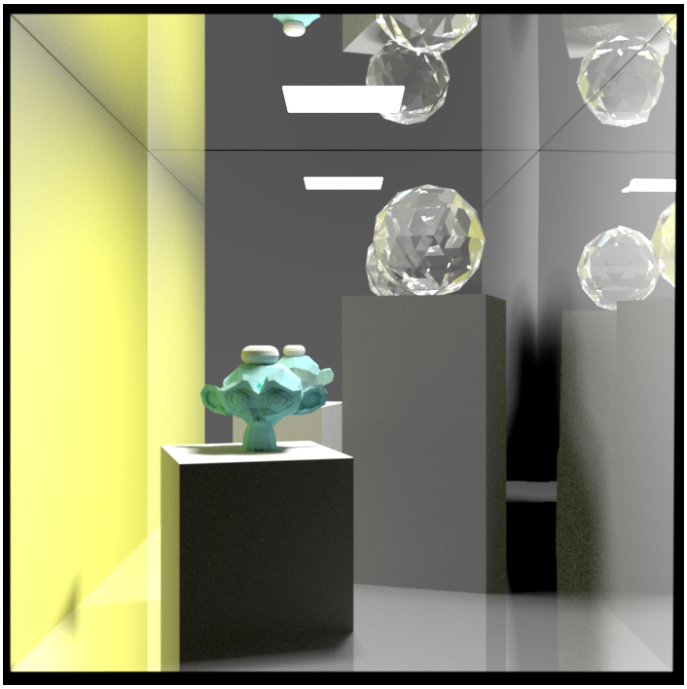

Fig. 11. My scene. Homo medium with $\sigma_s = 0.4$, $\sigma_a = 0.01$ and $g = +0.3$. correction = 700, correctL = 0.2

I still need a correctL coefficient to cut the result L down, which is around 0.2.

But, there was another problem of the same class. In ray marching we use $ds$ to find transmittance and this is not normal. This skews the transmittance values for different dimension scales. I used correction factor to account for this, but I assume there must be a normalization procedure inside

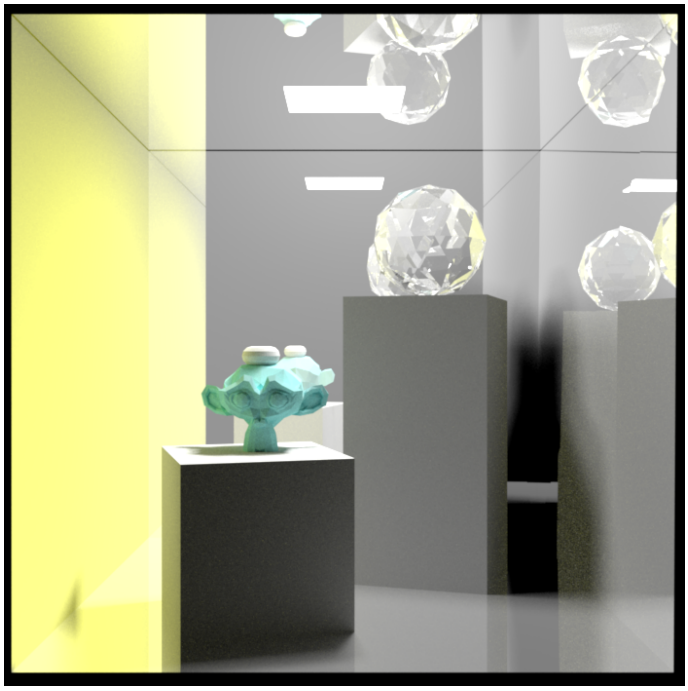

Fig. 12. My scene. $g = -0.3$

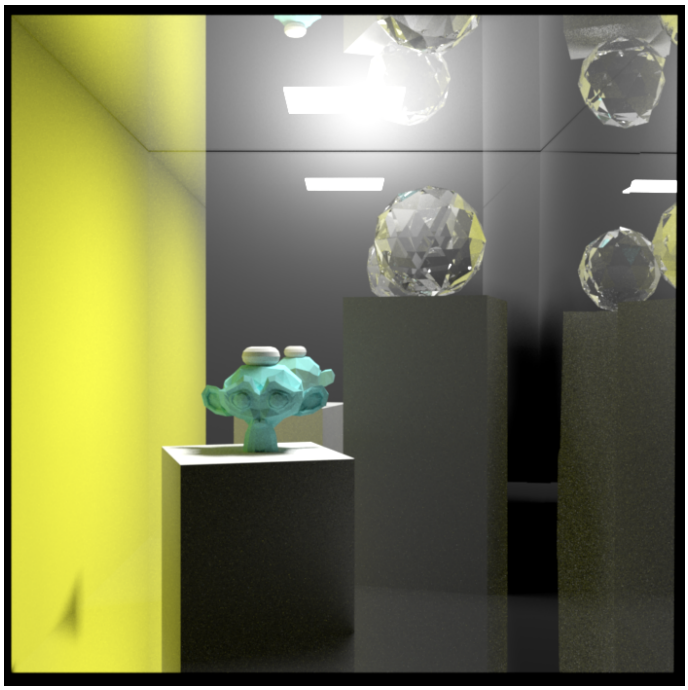

Fig. 13. My scene. $g = -0.9$

Nori that handles such dimension discrepancies and I am unaware of.

## REFERENCES

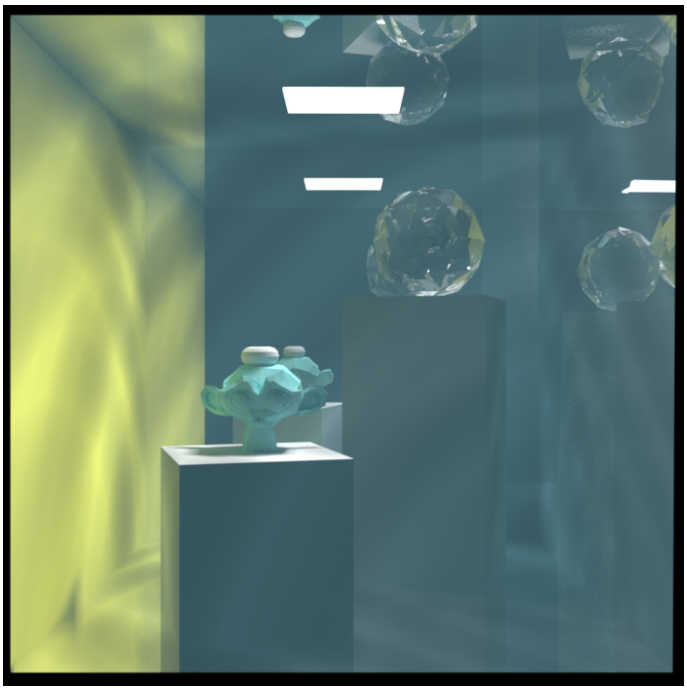

Fig. 14. My scene. Hetero medium with density as shown in Fig. 6. Tried to model water in the cbox. $\sigma_s = 0.4$, $\sigma_a = 0.01$ and $g = +0.9$. correction = 700, correctL = 0.25