Volumetric Ray Tracer in Homogeneous Media

Zhen Tong
The Chinese University of Hong Kong, Shenzhen
120090694@link.cuhk.edu.cn

Abstract

The volumetric ray tracer is designed for scatter media scene rendering like fog and clouds. The paper introduces the fog renderer model, using the exponential fog particle sampling, light attenuation through fog, fog scattering, and the global illumination pipeline. The implementation details are also provided, which involve reading the scene, performing BVH segmentation, ray casting, collision detection, and sampling reflections and refractions based on the BSDF. The experiments section compares the performance of one-bounce ray tracing and global illumination. It also explores the effects of different phase parameters and density parameters on the final rendered images.

1 Introduction

In the field of computer graphics, rendering is the process of generating realistic images from 3D scenes. Traditionally, we use ray tracing as a rendering technique that simulates the path of light rays as they interact with the objects in the scene to produce highly realistic images[8].To achieve an effective and realistic rendering performance, people design algorithms for global illumination[7]. When dealing with scenes containing scatter media, such as fog, smoke, or clouds, traditional ray tracing methods may not be sufficient to capture the complex light interactions that occur within these media[4].

In this project, we focus on the challenging task of scatter media ray tracing, which involves accurately simulating the interactions of light rays with participating media like clouds and fog. The main goal is to achieve physically accurate renderings of scenes containing these scattering effects.

2 Related Work

2.1 Bounding Volume Hierarchy

Bounding Volume Hierarchy (BVH) is a data structure commonly used in ray tracing to accelerate the intersection tests between rays and complex 3D objects[1]. BVH is particularly effective for scenes containing numerous geometric primitives or complex meshes.

The main idea behind BVH is to organize the objects in a scene hierarchically into a tree-like structure, where each internal node represents a bounding volume that encloses its child nodes (subtrees) and each leaf node contains a single object or a small group of objects. BVH construction typically involves a recursive process that subdivides the scene into smaller regions, creating the hierarchy. In this project, the BVH uses a heuristic to decide which axis is used to split along:

$$H_i = N_l V_l + N_r V_r \tag{1}$$

$$i \in \{x, y, z\} \quad \text{is the axis split along} \tag{2}$$

Link for codes: https://github.com/58191554/FogRenderer
Link for webpage: https://58191554.github.io/fog_renderer.html

N is the number of primitives of the left(l) or right(r) node, and V is the volume of the sub bounding box. Therefore, we can construct the bounding volume binary tree recursively, until the primitive number in one node is less than a threshold.

2.2 Bidirectional Scattering Distribution Function

Bidirectional Scattering Distribution Function (BSDF) is a fundamental concept in computer graphics and rendering that describes the way light scatters at a surface point[9]. The BSDF defines how light is reflected, transmitted, or absorbed at the surface, taking into account both incoming and outgoing directions of light.

When a light ray interacts with a surface, it can be either reflected or transmitted (refracted) depending on the properties of the material. The BSDF quantifies how much light is scattered in different directions for a given incident light direction. It is usually denoted as $f(\omega_i, \omega_o)$, where ω_i is the incoming light direction, ω_o is the outgoing light direction, and f represents the probability of light being scattered from ω_i to ω_o. This project implements 4 types of BSDF: diffuse BSDF, mirror BSDF (total reflection), refraction BSDF, and glass BSDF, where the glass BSDF exhibits both reflection and refraction properties.

2.3 Monte Carlo Sampling

Monte Carlo sampling is a probabilistic method used to estimate the radiance (light intensity) of reflected rays in ray tracing for computer graphics[10]. In ray tracing, when a ray intersects a surface, it needs to determine how the surface reflects light and compute the resulting radiance, which determines the color and brightness of the pixel. It provides a powerful sampling approach to multiple bounces of light.

$$L_r(x, \omega_o) = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot \cos\theta_i}{p(\omega_i)} \tag{3}$$

$L_r(x, \omega_o)$ represents the outgoing radiance from point x in the direction ω_o.N is the number of samples used in the Monte Carlo estimation.$f_r(x, \omega_i, \omega_o)$ is the Bidirectional Reflectance Distribution Function (BRDF), which represents the proportion of light from direction ω_i that reflects to direction ω_o at the surface point x.$L_i(x, \omega_i)$ is the incoming radiance from direction ω_i at point x.$\cos\theta_i$ is the cosine of the angle between the incident direction vector ω_i and the surface normal. $p(\omega_i)$ is the probability density function (PDF) of the sampling directions ω_i.

3 Fog Renderer Model

3.1 Exponential Fog Particle Sampling

In scatter media, rays may encounter fog particles in the air before hitting the actual primitives in the scene. Since the objective of this project is to simulate homogeneous fog effects, we assume that fog particles are uniformly distributed. As a result, the process of a ray traveling from its origin to its first collision with a fog particle can be considered as a Poisson process. Consequently, the subsequent collision events with fog particles follow an exponential distribution.[5]. According to the definition of the Poisson process, the time interval between the k^{th} random event and the $(k+1)^{th}$ random event follows an exponential distribution [Eric P. Lafortune and Yves D. Willems]. And the probability of zero random events occurring in a time period of length t is equal to

$$P(t|hit = 0) = \frac{e^{-\lambda t}(\lambda t)^0}{0!} = e^{-\lambda t} \tag{4}$$

$$P(t|hit > 0) = 1 - e^{-\lambda t} \tag{5}$$

CDF of exponential distribution:

$$F(t) = \begin{cases} 1 - e^{-\lambda t} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{6}$$

Then using a uniform random variable $z \sim U[0,1]$, we can sample particle intersection $t = -\frac{\log(1-z)}{\lambda}$. The physical interpretation of λ is the number of events per unit time; therefore, a larger λ will result in shorter time t.

3.2 Light Attenuation Through Fog

Jensen and Buhler propose a practical model for simulating subsurface light transport, which includes the behavior of light as it passes through scattering media such as fog[6]. When a ray of light travels through the fog medium. Specifically, the radiance of the light will experience the following processes:

* Absorption: Some of the light's energy is absorbed by the fog particles as it passes through the medium.
* Emission: The fog particles may also emit light of their own, contributing to the radiance of the transmitted light.
* Out-scattering: The fog particles scatter the incident light in various directions, causing it to be redirected as it travels through the medium.
* In-scattering: Light from other sources in the scene may also scatter and enter the fog medium, contributing to the radiance of the transmitted light.

When simplifying the model to focus only on the increase or decrease of the emitted ray radiance, there are two processes that contribute to the decay of radiance in the simplified model:

$$dL = -\sigma_t(t)Ldt \tag{7}$$

where dL is the light intensity derivative, and dt is the time derivative along the ray. $\sigma_t = \sigma_a + \sigma_s$ represents the sum of the absorption coefficient σ_a and the out-scattering coefficient σ_s. This equation describes how the intensity L of light decays as it passes through the absorbing and out-scattering media located at position $\vec{o} + \vec{dt}$ along the ray.

Solving the equation, we obtain the integrated formula

$$L = L_0 e^{-\int_0^{t_{end}} \sigma_t(x)dx} = L_0 e^{-\sigma_t t_{end}} \tag{8}$$

where L is the final intensity of light observed after decay, L_0 is the initial intensity of the light, and t_{end} is the intersection time along the ray. This integrated formula allows us to determine how much light remains after decay at the intersection point t_{end} due to the absorbing and out-scattering effects along the ray's path.

3.3 Fog Scattering

According to the four processes above, different kind of fog transmit the light differently. The Henyey-Greenstein phase function is commonly used to model the scattering behavior of light as it interacts with particles the medium[3].In a volumetric renderer, the Henyey-Greenstein phase function is applied to approximate the probability distribution of light scattering in different directions. The phase function takes the form:

$$P_{HG}(\theta) = \frac{1}{4\pi} \frac{1-g^2}{(1+g^2-2g\cos\theta)^{\frac{3}{2}}} \tag{9}$$

Monte Carlo simulations are based on the generalization of random samples that are distributed according to a certain distribution. Samples from nonuniform distributions can be generated by transforming sequences of uniform random numbers on (0, 1). Using Monte Carlo sampling, a continuous random variable μ can be generated by taking the transform of a uniform random variable ξ on (0, 1):

$$\mu = F^{-1}(\xi) \quad \text{and} \quad F(\mu) = \int_{-\infty}^{\mu} p(\mu_0)d\mu_0 \tag{10}$$

where μ is a random variable with the desired distribution, $p(\mu)$ denotes the probability density (in this setting, the scattering phase function), and $F(\mu)$ denotes the cumulative distribution function

(CDF).The Henyey-Greenstein phase function can be sampled with the inverse CDF[13]:

$$\mu = \frac{1}{2g}\left(1 + g^2 - \left(\frac{1 - g^2}{1 - g + 2g\xi_1}\right)^2\right) \tag{11}$$

$$\phi = 2\pi\xi_2 \tag{12}$$

where $\mu = \cos\theta$, and ξ_1 and ξ_2 are random numbers uniformly distributed on (0, 1).

3.4 Global Illumination Pipeline

In this project, ray bounces more than one time according to the pipeline. The global illumination can be implemented by recursively sampling new rays, hit on objects, and sampling reflection by the Monte Carlo method. The fog particles act as interactable object in the recursive sampling. The process can be interpreted as[8]:

$$L_i = f_r \cdot g(w_i) \cdot cos\theta / pdf \tag{13}$$

In the equation, L_i is the radiance for a sampled ray. f_r is the sampled reflection following BSDF when the ray intersects with an object. pdf is the probability assigned by the sample reflection ray w_i. Function g represents the recursive global illumination function. θ is the angle between hitting ray and normal vector on the intersction.

Algorithm 1 Global Illumination

function GLOBALILLUMINATION(ray, intersection)
  if ray depth > 0 **and** coin flip(p) **then**
    BVH intersection
    set ray time $t = \min\{t_i, t_{\text{fog}}\}$
    if $t_{\text{fog}} < t_i$ **then**
      L_i add fog lighting sphere estimation
    end if
    if $t_i < t_{\text{fog}}$ **then**
      L_i add fog lighting hemisphere estimation
    end if
    Set a new ray with random direction.
    Decrease the ray depth
    set weight of reflectance with ray direction
    if hit fog particle **then**
      sample new_ray following inverse HG
    end if
    assign HG_pdf
    if hit object **then**
      sample new_ray following object bsdf
      assign bsdf_pdf
      calculate intersection
      return L_i += $f \cdot$ GlobalIllumination$(new_ray, intersection) \cdot \cos\theta / p /$pdf
      ray.depth decrease by 1
    end if
  end if
  return $L_i \cdot f \cdot \cos\theta \cdot$ BSDF_pdf
end function

Each time sampling a new intersection, the fog particle intersection time t_{fog} sample by exponential fog particle sampling mentioned in 3.1 will compare with the object intersection time t_i. Furthermore, Russian Roulette[7] is a technique used to improve the efficiency of the Monte Carlo path tracing algorithm because it can be computationally expensive due to the need for many ray bounces and sampling of light sources. Overall, the global illumination pipeline can be described as pseudocode.

4 Experiments

4.1 Dataset

The dataset used in this study is The Stanford 3D Scanning Repository, which contains a collection of 3D scan models from Stanford University[12]. Additionally, to enhance the lighting effects in the scenes, spot lights were manually added using Blender, a 3D modeling and rendering software[2]

4.2 Implementation Details

This project implements all the algorithms in C++. It is based on the third assignment starter code provided by the University of California, Berkeley's CS184 course.The first step involves reading the DAE (Digital Asset Exchange) file, which contains the 3D scene data.After reading the object from the file, the next step is to perform BVH (Bounding Volume Hierarchy) segmentation on the object. BVH is a spatial data structure used for efficient ray-object intersection tests.Once the BVH segmentation is done, rays are cast into the scene. The number of rays emitted is determined by the sample rate and sample light size settings. If thin lens parameters are used, lens sampling is performed during ray generation to achieve Depth of Field (DOF) effect.During ray casting, collision detection with scene objects is performed. The material parameters and the Bidirectional Scattering Distribution Function (BSDF) of the colliding surface are accessed. The rays are then sampled for reflection or refraction based on the BSDF. Schlick's Approximation[11] is employed for estimating glass material.If importance sampling is enabled, the algorithm iterates through all light sources and samples rays accordingly.

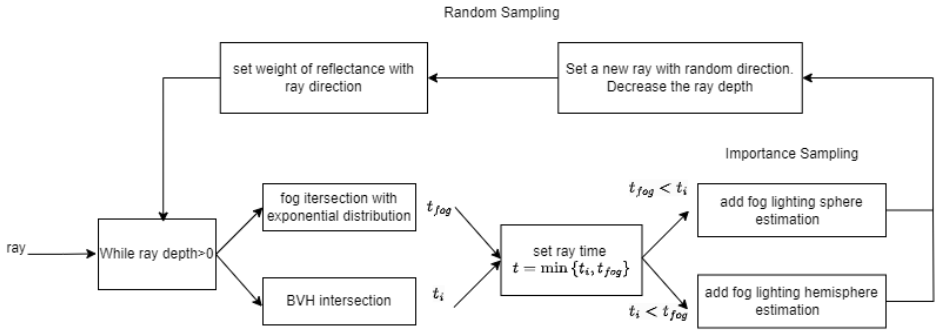

Figure 1: The overall logic of our project

4.3 Performance and Image Result

One-Bounce & Global Illumination We have achieved global illumination in our rendering, allowing spotlights to illuminate walls outside their direct range, and the shadowed areas now have increased brightness (they are not completely black anymore). By employing light direction importance sampling, we have significantly improved the quality of the rendered images. As shown in the images, without importance sampling, the rendering appears completely dark with low sample counts and exhibits significant noise at high sample counts.

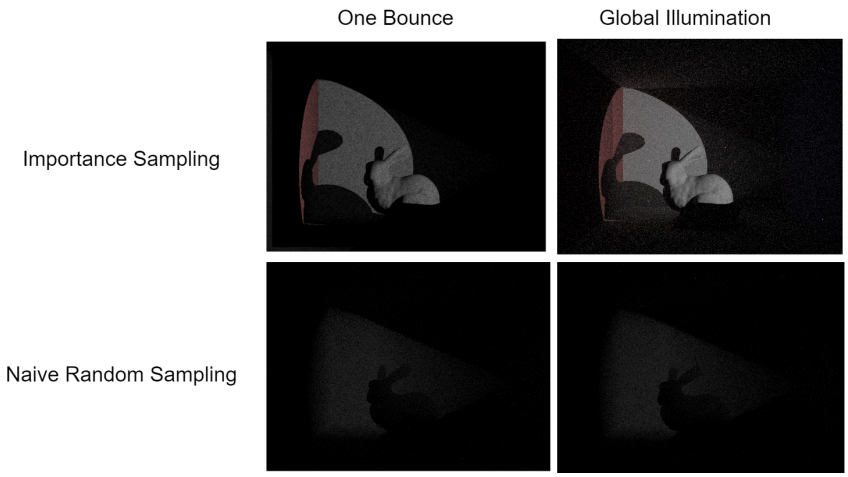

Figure 2: One-Bounce Global Illumination

Different Density Parameter & Phase Parameter After continuously adjusting the value of lambda λ, it was found that lambda between 0.02 and 0.2 produces a fog effect that closely resembles reality.

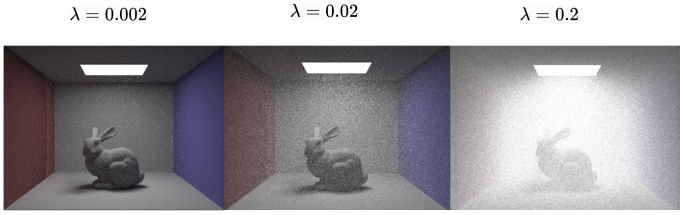

Figure 3: comparison of different λ in equation (6)

As shown in the figure, different values of the parameter "g" ranging from -1 to 1 result in varying effects of fog on light in different directions. The image displays distinct results for selected values of "g." For values of "g" less than 0, the differences are not significant, as the scene appears relatively dark overall. In the range of "g" values greater than 0 and less than 0.6, the differences are not substantial, and the scene appears to have a milky white appearance. However, when "g" is set to 0.9, a phenomenon of a "halo" around the light source becomes evident, where the area surrounding the light source appears brighter than the rest of the scene.

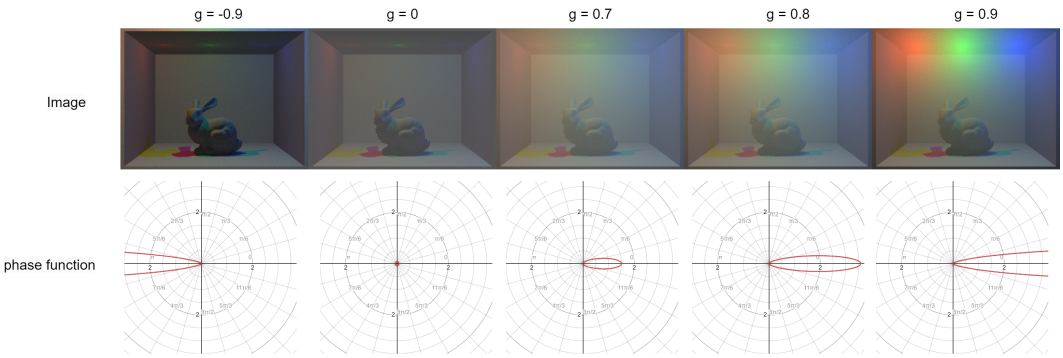

Figure 4: comparison of different phase function parameter

5 Discussions & Future Work

* According to Zhang [13], importance sampling can improve the sampling efficiency in volumetric rendering.
* The parameter in 3.1 and 3.2 are independent, however, they can be related. With a more accurate value of σ in equation(8), λ in equation(6), and g in equation (9), the fog effect will be closer to reality.
* This project is implemented based on CPU programming, without real-time rendering. Future GPU acceleration may reach a real-time rendering performance.
* The sharp edges in importance-sampled shadows of spot lights occur because the light rays are directly sampled towards a spot during importance sampling. This results in deterministic sampling instead of probabilistic sampling, leading to the appearance of sharp edges in the shadows. To address this issue, one solution is to treat the spot light source as a small circle with an area or a small sphere with volume, rather than just a single point. By considering the spot light as having an area or volume, the sampling becomes probabilistic, which helps to alleviate the sharp edge artifacts in the shadows.

References

[1] J. Arvo and D. Kirk. Fast ray tracing by ray classification. *ACM Siggraph Computer Graphics*, 21(4):55–64, 1987.

[2] B. Foundation. Blender - a 3d modeling and rendering software. https://www.blender.org/.

[3] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Astrophysical Journal*, 93:70–83, 1941.

[4] W. Jarosz. *Efficient Monte Carlo methods for light transport in scattering media*. PhD thesis, University of California, San Diego, 2008.

[5] W. Jarosz, H. W. Jensen, H. Moreton, and J. Stam. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (TOG)*, 30(1):1–19, 2011.

[6] H. W. Jensen and J. Buhler. A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001.

[7] H. W. Jensen, P. H. Christensen, and E. Chan. A practical guide to global illumination using ray tracing and photon mapping. *SIGGRAPH Course Notes*, 2001.

[8] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, 1986.

[9] F. E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–775, 1965.

[10] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2010.

[11] C. Schlick. An inexpensive brdf model for physically-based rendering. *Computer graphics forum*, 13(3), 1994.

[12] S. University. The stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep/.

[13] J. Zhang. On sampling of scattering phase functions. *Astronomy and Computing*, 29:100329, 2019.