Visual Computing:
Volumetric Lighting in 3D environments

***Abstract*—Volumetric lighting, commonly referred to as "God rays," is a pivotal technique in 3D computer graphics that simulates the interaction of light with particulate matter, such as fog, dust, or smoke, to create realistic lighting effects. This paper delves into the fundamental principles of volumetric lighting, its significance in visual computing, and the different rendering techniques.**

***Index Terms*- Volumetric lighting, God rays, Visual computing, Ray marching, Fog volumes, Voxel grid, Screen-Space Volumetric Lighting, Scattering, Particle-Based God Rays**

I. Introduction

Volumetric lighting, often referred to as "God Rays" or "Crepuscular Rays," is a fundamental technique in 3D computer graphics that simulates the interaction of light with participating media such as fog, dust, or smoke to create realistic lighting effects. This phenomenon is common in nature; for instance, when sunlight penetrates through clouds or when light traverses a dusty room, making light beams visible. Figure 1 illustrates crepuscular rays occurring in nature [1].

Figure 1 – Crepuscular rays on Nature.

In visual computing, accurately replicating these interactions is crucial for creating immersive and realistic virtual environments. This type of lighting is extensively used in various games, such as "Red Dead Redemption 2" and "The Witcher 3" to guide players toward specific locations, serving as visual cues. Additionally, it is employed to enhance the atmosphere of scenes, making them appear more tense or, conversely, indicating a peaceful area, thereby conveying emotions to players.

Moreover, in scientific and medical simulations, volumetric lighting is essential for visualizing complex volumetric data, such as computed tomography (CT) scans, allowing for a better understanding of internal structures. This technology has been studied and demonstrated in several successful applications, including modalities with a low signal-to-noise ratio, such as 3D ultrasound and magnetic resonance tomography (MRT) [2].

The evolution of volumetric lighting techniques reflects a significant challenge with computational efficiency, highlighting the continuous pursuit of a balance between visual fidelity and performance.

This article begins with a brief explanation of this phenomenon as it occurs in nature. Following that, various methods of implementing this model will be explored, along with examples showcasing how each approach has been utilized in different games. Finally, a comparison of the different techniques will be presented, highlighting which method is best suited for specific purposes

II. Visual phenomenon in the real word

In the natural environment, light interacts with participating media - substances composed of particles that occupy space, such as fog, dust, or smoke. When sunlight penetrates these media, it scatters in various directions due to interactions with the particles, a process known as multiple scattering. This scattering causes the light to diffuse, creating visible beams or shafts of light, commonly referred to as "God rays" or crepuscular rays [3].

Unlike interactions with solid surfaces, where light primarily reflects, in participating media, light enters the medium and undergoes scattering and absorption. Scattering refers to the deflection of light in multiple directions upon encountering particles, while absorption involves the attenuation of light intensity as it passes through the medium, contributing to the perception of depth and shadow.

Volumetric lighting in computer graphics aims to replicate these natural phenomena by calculating light paths through volumes rather than merely on surfaces. This approach considers how light scatters and absorbs within a medium, enriching visuals by providing a sense of dimension and atmosphere.

In visual computing, rendering each individual particle and its interactions to achieve a "perfect" simulation of volumetric lighting is computationally impractical. Consequently, developers employ various approximation techniques to replicate this phenomenon efficiently, balancing visual fidelity with performance constraints.

III. Implementations

Over the years, various implementations have been developed, ranging from those focused on minimizing computational costs to those prioritizing the realism of the scene.

A. Polyhedron Light Sources

A commonly employed technique, especially in earlier video games, involves the use of geometric transparent polygons to simulate volumetric lighting effects. This method places polygons within the scene, assigning them transparency levels corresponding to the desired light intensity. Often, these polygons are textured or animated to enhance visual realism.

This technique was described in 1987 by Tomoyuki Nishita, Yasuhiro Miyawaki, and Eihachiro Nakamae in their paper "A Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources." [4]

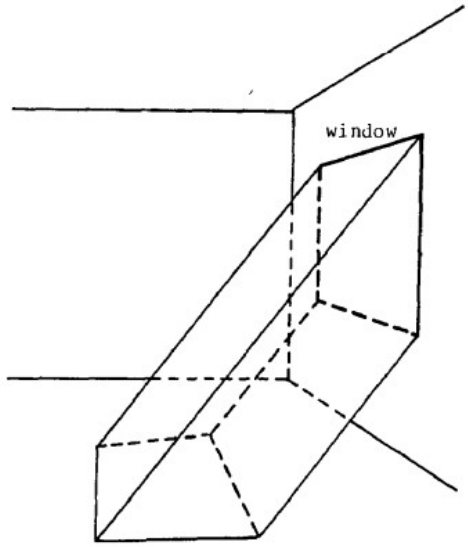

Figure 2 – Illumination volume for a parallel source.

A good example of this technic applied in early games is Mario 64 (1996) in the Piano Room.

Figure 3 – Entering on Piano Room Mario 64

While effective for its time, this approach has significant limitations. The polygons are static and do not adapt to changes in the physical environment, reducing realism in scenes with dynamic lighting.

In practice, when moving the piano, these issues become clear. The static polygon keeps the shadow area fixed, regardless of the piano's position. Figure 4 shows a square on the floor where the piano should block light, and the white mark representing reflected light on the piano remains unchanged even after it moves.

Figure 4 – Moved Piano on Piano Room Mario 64

This lack of realism stems from the static nature of the technique. While the polygon's simplistic appearance (in this case a semi-transparent yellow plane) could be improved with 3D animations, the inability of static lighting to interact dynamically with the environment remains a fundamental flaw, limiting its use in real-time scenarios.

Despite these limitations, the use of polygons for volumetric lighting remains prevalent in modern games due to their computational efficiency. Developers have refined this approach by adding features such as animations, dynamic light variations that respond to player movement, and mechanisms to fade or remove these effects at close proximity, thereby improving realism and immersion. A notable example of this technique can be seen in Battlefield V, where it is used to create atmospheric lighting effects that significantly enhance the visual experience [5].

Figure 5 – Polyhedron Light on Battlefield V.

B. Particle-Based

Particle-Based God Rays is a technique widely used in game engines such as Unity to simulate volumetric light beams. This method relies on a specialized particle system, often referred to as a volume particle system, to define the spatial extent and location of the light rays. Developers configure key properties in the particle system's main module, while the engine assigns attributes such as size, opacity, and emission rate to each particle, creating a visually compelling effect. You can find a tutorial on implementing this technique in Unity published on Styly Magazine [6].

To enhance realism, developers can introduce texture variations, which add depth and detail to the light beams. Shaders are also crucial in refining the effect by dynamically adjusting the particles' appearance based on their distance from the light source or the camera angle, further enhancing the realism of the light rays.

One of the key strengths of this technique is its dynamic adaptability. Particles can respond to changes in the scene, including the movement of light sources, shifts in camera perspective, and environmental conditions such as fog, smoke, or obstacles. This responsiveness makes particle-based god rays particularly effective in creating immersive visual experiences.

However, a significant drawback of this approach is its high performance cost. Rendering a large number of particles can be computationally expensive, particularly in scenes with multiple light sources or complex interactions. This limitation makes careful optimization crucial for ensuring the technique is viable for real-time applications.

Figure 6 – Particle-Based God Rays in Unity.

C. Ray Marching

Ray marching has been employed in several games to create stunning visual effects, a notable example is DOOM (2016).

Unlike traditional ray tracing, which computes precise intersections, ray marching incrementally advances rays through a scene, sampling the environment at each step to determine the presence of geometry or effects [7] [8]. This results in dynamic lighting that interacts naturally with the environment, closely resembling authentic god rays. By calculating light propagation incrementally, it aligns with mathematical models, enhancing the realism of the effect [9].

Figure 7 – Ray Marching effect with maximum resolution

An example of ray marching applied in a practical context is described in NVIDIA's GPU Gems 2, Chapter 8 [10]. The authors showcase how ray marching can enhance visual fidelity in real-time rendering by simulating small-scale geometric details. This technique uses distance functions to achieve per-pixel displacement mapping, effectively adding fine-grained details to surfaces with minimal computational overhead.

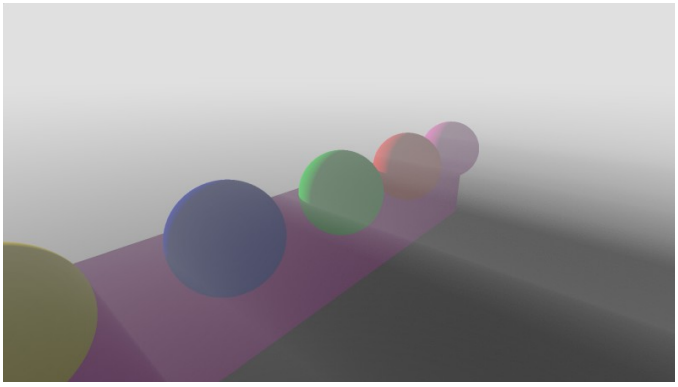
Figure 8 - Ray Marching Fog With Blue Noise

However, despite its visual advantages, ray marching remains computationally expensive. To address this, various optimizations have been developed, such as adaptive stepping and precomputed distance fields, which reduce processing requirements. As shown in Figure 4, the rays appear unnatural due to their overly sharp edges and poorly blended transitions between the circular segments of the light shafts. Additionally, there is a distinct, harsh line separating the illuminated areas from those in shadow, which breaks the realism of the effect. While the intention behind the implementation is clear, and this technique has been widely adopted in games since the use of ray tracing, its execution depends on so many optimizations that it leaves much to be expected in terms of realism.

D. Voxel Grid

In 2014, B. Wronski presented a comprehensive approach to real-time volumetric fog rendering in his work "Volumetric Fog: Unified compute shader based solution to atmospheric scattering" [11]. This concept was developed and used in Assassin's Creed 4 and was later presented at SIGGRAPH 2014.

The Voxel Grid is a technique that divides three-dimensional space into a grid of small cubes called voxels (volumetric pixels) [12]. Each voxel represents a specific portion of the environment and stores information such as density, color, or light intensity. This structure enables the efficient and realistic simulation of complex volumetric effects like fog, smoke, and atmospheric lighting.

During rendering, light rays pass through the voxel grid, collecting data from the voxels they intersect. This allows for mathematical calculations of how light interacts with particles in the air, producing effects like light scattering and absorption.

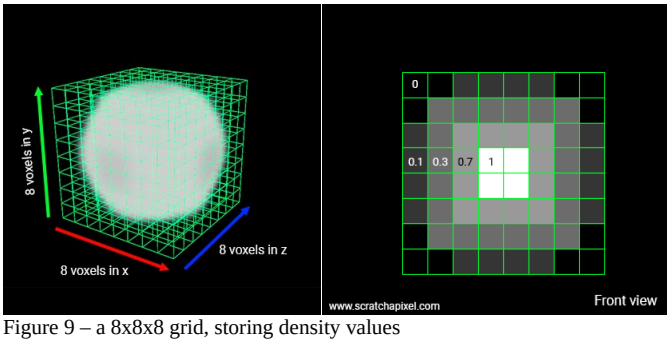

Figure 9 – a 8x8x8 grid, storing density values

The use of a voxel grid is a significant advantage as it offers an excellent balance between performance and visual quality, allowing complex scenes to be rendered in real-time without a massive computational cost [13].


Figure 10 – Example of illumination done with Voxel Grid

This method was widely used and still has its place in several games such as Horizon Zero Dawn and Control.

E. Screen-space

The main difference between the screen-space technique and the previously discussed methods is that the latter rely on 3D models to achieve a more realistic appearance, while the screen-space technique uses 2D models, processing only what is visible in the rendered image. This approach sacrifices some realism to achieve significantly faster processing times, while still maintaining good visual fidelity. As a result, the screen-space technique is highly effective for real-time applications where performance is critical.

The process involves rendering the scene from the light source's perspective to identify occluding objects. Then, a radial blur is applied from the light's position, creating the illusion of light scattering through the environment. This approach allows for dynamic lighting effects that respond to changes in the scene, such as moving objects or varying light conditions, without the heavy computational load of full volumetric calculations [14].

As this technique offers an excellent balance between realism and low computational cost, it is widely adopted in many games. Some examples include Shadow of the Tomb Raider [15] and The Witcher 3.


Figure 11 – God Rays in The Witcher 3

IV. Comparison of the different techniques

The various implementations for representing volumetric light showcase several interesting methods, each tailored to specific needs and objectives. In the table below is a summary of how these techniques can be utilized depending on the desired outcome. The ranking indicates their effectiveness, with 1 being the best and 5 the worst for both realism and computational cost. A ranking of 1 in realism corresponds to a highly realistic technique, while 1 in computational cost indicates a low computational overhead.

Rating	Realism	Computational cost
1	Ray Marching	Screen-Space
2	Voxel Grid	Polyhedron Light
3	Polyhedron Light	Particle-Based
4	Particle-Based	Voxel Grid
5	Screen-Space	Ray Marching

Table 1 – Ranked Technics for Realism and Computational cost

With this table we can get a good idea of which technique we can use depending on the balance between realism and computational cost that we want to achieve:

- Screen-Space techniques are best suited for real-time applications like fast-paced games as performance is critical.
- Polyhedron Light Sources work well for scenarios requiring basic volumetric lighting with limited resources, such as stylized games.
- Particle-Based God Rays are ideal for dynamic effects without requiring physical accuracy, making them suitable for artistic or cinematic moments.
- Voxel Grid Volumetric Lighting is a great choice for games or applications that prioritize realism in atmospheric effects, such as dense fog or smoke, while maintaining a good performance.
- Ray Marching is the best choice to achieve the highest level of realism in volumetric effects, making it perfect for high-fidelity rendering, cinematics, or visual demonstrations, special in cases where time or computational resources are less of a concern.

V. Conclusions

In this article, different ways of representing volumetric light were presented, providing an overview of the fundamentals of each technique. Furthermore, a comparison was made between the techniques, highlighting the realism - computational cost balance of each one. Therefore, this work offers a practical tool for computer graphics developers who wish to implement volumetric light effects, but are not yet clear about which technique is best suited to their specific needs. By providing an accessible and targeted analysis, this article contributes to facilitating decision-making in developing realistic and efficient scenes.

References

[1] D. L. Wang, S. Li, L. P. Yang and A. M. Hao, "Real-time volumetric lighting using polygonal light volume," 2014 International Conference on Information Science, Electronics and Electrical Engineering, Sapporo, Japan, 2014, pp. 619-625, doi: 10.1109/InfoSEEE.2014.6948188.

[2] Britannica, The Editors of Encyclopaedia. "crepuscular rays". Encyclopedia Britannica, 4 Jan. 2019, [Online] https://www.britannica.com/science/crepuscular-ray. Accessed 25 November 2024.

[3] T. Ropinski, C. Döring and C. Rezk-Salama, "Interactive volumetric lighting simulating scattering and shadowing," 2010 IEEE Pacific Visualization Symposium (PacificVis), Taipei, Taiwan, 2010, pp. 169-176, doi: 10.1109/PACIFICVIS.2010.5429594.

[4] T. Nishita, Y. Miyawaki, and E. Nakamae, "A shading model for atmospheric scattering considering luminous intensity distribution of light sources, " In Proc SIGGRAPH87. 1987, pp. 303-310.

[5] Tim Schiesser, “Battlefield V DXR Real-Time Ray Tracing Performance Tested”, November 15, 2018, [Online] https://www.techspot.com/review/1749-battlefield-ray-tracing-benchmarks/

[6] StylyMagazine, [Unity] Using the particle system to express a God Ray, March 2, 2021, [Online] https://styly.cc/tips/godray-particle/

[7] The blog at the bottom of the sea, “Ray Marching Fog With Blue Noise”, May 10, 2020, [Online] https://blog.demofox.org/2020/05/10/ray-marching-fog-with-blue-noise/

[8] C. Wyman and S. Ramsey, "Interactive volumetric shadows in participating media with single-scattering," 2008 IEEE Symposium on Interactive Ray Tracing, Los Angeles, CA, USA, 2008, pp. 87-92, doi: 10.1109/RT.2008.4634627.

[9] Benjamin Glatzel, “Volumetric Lighting for Many Lights in Lords of the Fallen”, Digital Dragons, 2014 [Online] https://www.slideshare.net/slideshow/volumetric-lighting-for-many-lights-in-lords-of-the-fallen/34507223

[10] William Donnelly, “Per-Pixel Displacement Mapping with Distance Functions”, NVDIA – GPU GEMS 2, chapter 8, [Online] https://developer.nvidia.com/gpugems/gpugems2/part-i-geometric-complexity/chapter-8-pixel-displacement-mapping-distance-functions

[11] B. Wronski, “Volumetric Fog: Unified compute shader based solution to atmospheric scattering”, SIGGRAPH, 2014 [Online] https://www.advances.realtimerendering.com/s2014/wronski/bwronski_volumetric_fog_siggraph2014.pdf

[12] Scratchapixel, “Volume Rendering Based On 3D Voxel Grids”, [Online] https://www.scratchapixel.com/lessons/3d-basic-rendering/volume-rendering-for-developers/volume-rendering-voxel-grids.html

[13] Artem Kovalovs. 2020. Volumetric Effects of The Last of Us: Part Two. In Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH ’20 Talks), August 17, 2020. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3388767.3407349

[14] Kenny Mitchell, “Volumetric Light Scattering as a Post-Process”, NVDIA – GPU GEMS 3 chapter 13, [Online] https://developer.nvidia.com/gpugems/gpugems3/part-ii-light-and-shadows/chapter-13-volumetric-light-scattering-post-process

[15] Andrew Burnes , “Shadow of the Tomb Raider Graphics and Performance Guide”, NVDIA GameWorks, 2018 [Online] https://www.nvidia.com/en-us/geforce/news/shadow-of-the-tomb-raider-graphics-and-performance-guide/

ChatGPT was used to correct grammar and rephrase sentences.