Single-Scattering and Multi-Scattering in Real-time Volumetric Rendering of Clouds

Mikołaj Bajkowski[1] and Dominik Szajerman[1][0000−0002−4316−5310]

Institute of Information Technology, Lodz University of Technology, Łódź, Poland
dominik.szajerman@p.lodz.pl

Abstract. The aim of this work was to design an algorithm for rendering volumetric clouds in real time using a voxel representation. The results were verified using reference renders created with the *Blender* program using the *Principled Volume* shader. The important properties of the algorithm that were tried to be achieved are the ability to display clouds with different characteristics (thin and dense clouds) and the speed of operation enabling interactivity. We proposed a method consisting of two parameterizable image display algorithms with various performance and properties. The starting point was the single-scattering algorithm, which was extended with precalculation, and a simplified form of multi-scattering. Individual methods were compared with reference images. Methods performing similar tasks, depending on the purpose, generate single image frames at a rate ranging from several dozen hours to a few seconds. Using the described mechanisms, the proposed method allowed to achieve times between 1 and 200 milliseconds, depending on the method variant and quality settings.

Keywords: Single-scattering · Multi-scattering · Ray marching · Cloud rendering.

1 Introduction

Being an inherent element of the Earth's landscape, clouds are an important element of the field of special effects, three-dimensional animation, and computer games that try to transfer at least a small part of the real world to virtual space. Unfortunately, the architecture of graphics accelerators is optimized for displaying graphics based on triangle meshes, which cannot properly reflect the complex structure of such a phenomenon.

For this reason, rendering realistic volumetric structures remains one of the most difficult challenges in high-performance computer graphics. This difficulty is caused by the great complexity of light phenomena that influence the appearance of such a structure. One of the frequently cited features of clouds is their high albedo. This causes the light to be dispersed multiple times in the cloud, which multiplies the number of physical interactions that must be simulated in order to generate a realistic image.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

2 Related Work

Methods for realistic cloud rendering have been known for a long time[7]. One of them is synthesis using ray tracing, among others with support for Monte-Carlo[3] methods. In many scientific works, it serves as a reference technique to which the developed or optimized method must be as close as possible. Unfortunately, it is characterized by very high computational complexity. This precluded the method from being used in interactive applications for many years. Only recently, with the introduction of graphics cards supporting this technique in hardware, it has become possible to hybridly use ray tracing to enhance the effects obtained with traditional polygon rendering techniques in real time.

In the case of interactive solutions, where the observation point is only on the ground, it may still be a reasonable solution to implement a dome with a static or dynamically generated two-dimensional texture of clouds distributed in the sky[11]. This is a simple solution, and the appropriate lighting effects are synthesized by pre-processing the clouds to extract the data needed for appropriate shading operations. This method pays off with efficient rendering. This works particularly well when the lack of depth will not be perceived negatively, e.g. when the interactive experience effectively focuses the observer's attention on other objects in the virtual space.

If clouds are an object that takes up most of the rendered image or is a significant part of *gameplay*, then a different approach is needed. In games such as *War Thunder* or the *Flight Simulator* series, the player takes their feet off the ground, having the opportunity to observe clouds from different angles, lighting conditions, and the possibility of being inside such a phenomenon. Creators have at their disposal hybrid techniques based to some extent on *billboarding*[6]. A more advanced and increasingly popular solution is volumetric representation. The clouds in the above-mentioned titles look similar to those in Egor Yusov work[13]. A particle system was used here. The similarity of the spheres that make up clouds allowed, among other things, to optimize the display by pre-calculating the light scattering characteristics within such a volumetric domain. The problem in this case, however, is the relatively small variety of textures and shapes of clouds that can be obtained in this way.

There are techniques that compete with ray tracing. These are methods for rendering volumetric structures using voxels, but most of the work focuses on offline rendering solutions, often using methods simulating multi-scattering. D. Koerner et al. [10] used the *flux-limited diffusion* method, the effects of which were very similar to reference images created using the ray tracing method, and rendering reduced to below a second.

Another example is the work of Simon Kallweit et al. [8], which introduces a realistic synthesis of cloud images using a multilayer perceptron predicting the radiance distribution. There, it was also possible to achieve effects indistinguishable from the reference image generated by ray tracing, and the full-quality rendering time was reduced from several dozen hours to just a few seconds. Additionally, the results are compared with the *flux-limited diffusion* method, which

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

in this case performs unfavorably when rendering volumetric structures with high albedo, such as clouds.

There are no works describing single- and multi-scattering methods using voxel-based raymarching of atmospheric clouds. This work aims to investigate this scenario and explore the possibility of extending the basic single-scattering algorithm to achieve render times that allow the technique to be used in interactive applications. At the same time, it will try to synthesize lighting effects characteristic of techniques with higher computational complexity.

3 Method

Later in the work, it is described how the following solutions were implemented in the rendering pipeline:

- Single-Scattering brute-force,
- Single-Scattering with cached photon map,
- Simplified Multi-Scattering with cached photon map based on Single-Scattering.

3.1 Dataset

The OpenVDB[5] format was used as the cloud density data source. It is a fixed-depth hierarchical structure that allows the use of voxels of a specific resolution and optimizes memory usage by excluding areas where the value can be considered equal to the background. In the presented case, these are all areas where the cloud density is zero. Standard OpenVDB structure settings were used: the root has a maximum of $(2^5)^3$ branches, each branch has a maximum of $(2^4)^3$ leaves and each leaf has a maximum of $(2^3)^3$ voxels. The data in the appropriate format was obtained from the 3D Art[2] website.

3.2 Single-Scattering

Single-scattering (SS) is understood as a situation in which light on its way from the source to the observer's eye is subject to a single scattering event. In the context of volumetric cloud rendering, this problem is solved by creating a ray originating from the point of view that penetrates the cloud mass, undergoing a single scattering event towards the light source, repeatedly along an axis that intersects the point of view.

In the SS implementation used, two parts can be basically distinguished: the primary ray and the shadow ray. The path of the primary ray begins at the point of observation and its direction is determined to pass through a given pixel on the screen. The step length is predetermined and should be a compromise between the algorithm's efficiency and the desired accuracy of mapping the cloud structure.

At each step of the primary ray, information about the cloud density is collected. The density of the sample determines the background occlusion factor

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

and the amount of light reflected towards the camera. The shadow ray is taken into account at each step of the primary ray for which the cloud sample has a non-zero density (Fig. 1). It begins in the last position of the primary ray and is directed towards the light source. On the section from the beginning of this ray to the boundary of the cloud, the integral of the cloud mass is calculated. As in the case of the primary ray, a constant step length is defined here. The integration result is important to determine the amount of light absorbed by the cloud.

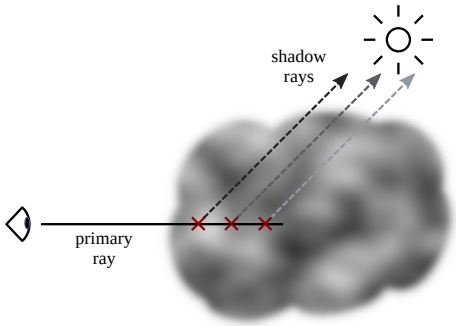

Fig. 1. Visualization of the rays used in the single-scattering algorithm.

The Monte-Carlo [9] method was used to stabilize the integration results in the case of multiple sampling. It is important that the stochastic process in this case guarantees a uniform distribution of values in the considered interval over subsequent samples. This is because it is assumed that a light scattering event can occur with equal probability at any point along the main ray. In this case it was implemented by randomizing the length of the first iteration of the step for each ray

The noise source is a previously generated blue noise texture whose offset is randomized for each call to the rendering pipeline. Blue noise has a high amount of high-frequency noise, which favors the integration of results and the use of noise removal techniques[4].

The background occlusion factor (final alpha of a given fragment) is calculated by the formula (1):

$$\alpha_x = \prod_{i=1}^{x} e^{-d_x l_g} \tag{1}$$

where x is the next step number, d_x is the cloud density at a given point in space resulting from the step number, and l_g is the length of the primary ray step.

The color of the pixel is calculated by the formula (2):

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

$$c_x = \sum_{i=1}^{x} \alpha_x c_{ss} i_{xss} (1 - e^{-d_x l_g}) \tag{2}$$

where c_{ss} is the base color of the cloud, and i_{xss} (3) is the intensity of the incoming light calculated from the shadow ray (4).

$$i_{xss} = L(d_v, \frac{10}{t_{xss}}) + l_b(t_{xss}) \tag{3}$$

$$t_{xss} = \sum_{j=1}^{\inf} d_j \cdot l_p \tag{4}$$

The function $L(d_v, d_p)$ is a modified approximate Lorenz-Mie phase function that depends on the dot product of the primary and shadow ray direction vectors. Nishita et al.[12] proposed two empirically derived approximations for nebulous atmospheres of different densities. The modification consists in modulating the shape and strength of the function depending on the integral value in order to imitate changes in the light propagation characteristics depending on the mass of the cloud integrated by the ray. The solution is sufficient for the presented application, as it is used to obtain the effect of *silverlining*[1] which occurs in areas with a low value of the shadow ray integral. The formula (5) contains a generalized form of the function proposed by Nishit et al.[12], modified so that there is no significant light intensity decay when the coefficient $d_p \approx 0$, and the primary and shadow ray vectors have opposite directions ($d_v \approx -1$).

$$L(d_v, d_p) = \frac{1}{4\pi} d_p \left(\frac{1 + d_v}{2} \right)^{d_p} \tag{5}$$

An additional isotropic factor (6) showing a lower degree of absorption imitates light scattering, assuming the existence of irradiance coming from the surrounding cloud mass. The coefficients modifying the actual weights of the input parameters were also selected empirically.

$$l_b(t_{xss}) = e^{-t_{xss}} + s e^{\frac{t_{xss}}{10}} + \frac{1}{5} 2 s e^{-\frac{t_{xss}}{50}} \tag{6}$$

3.3 Single-Scattering Cached

In the case of a stationary light source, the value returned by the shadow ray depends only on the starting position of this ray. In addition to the density value, the structure can store the previously calculated integral value discretized in the

[1] A bright outline of the cloud observed "against the sun", characteristic of dense clouds. A light ray, passing through a small mass of cloud, is largely scattered in the original direction, which causes a large amount of light to reach the observer's eye, being perceived as intense [1].

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

space of a single voxel. This should largely reduce the computational complexity of the algorithm at the expense of limited resolution.

The theoretical increase in memory requirement according to the previous solution is twofold due to the need to store information about the sum of the integral in parallel with the cloud density data. In practice it is higher due to the unfavorable *padding* of the leaf cell array of the hierarchical structure. The final highest resolution cloud data size was 239 Megabytes compared to the original 62.

3.4 Multi-Scattering

The multi-scattering (MS) algorithm was created on the basis of the Single-Scattering Cached algorithm and the photon map used in it. In order to achieve performance that will allow real-time interaction, a number of simplifications have been made to make this possible.

The implemented scenario is based on the following assumptions:

- The scene has one directional light source (sun).
- A single volumetric cloud is rendered.
- The properties of the cloud located only at a short distance from the point under consideration are important

In relation to the SS algorithm, in addition to a single light scatter, a point cloud is introduced at a short distance from the branching point, which simulates the irradiance resulting from the scattering of light coming from the immediate vicinity of the sampled point. In addition to the value of the integral taken from a given cloud point, the integral between the initial sampling point and the cloud point is approximated by taking the density value at both points (7). Indeed, unlike a single scattering in the basic form of the algorithm, two such events occur, which are additionally repeated for different variants of the point arrangement at which the second scattering occurs.

$$t_{xms} = \sum_{j=1}^{n} |\boldsymbol{x} - \boldsymbol{x_j}| \frac{d_x + d_{xj}}{2} + t_{xjss} \tag{7}$$

The set of points that is part of the sampling cloud has the following properties:

- The scattering direction vector is uniformly random.
- In the case of a cloud consisting of N points, the distance of subsequent points from the local center of the cloud coordinate system is $r_x = \frac{Rx}{N}$. Where R is the maximum radius of the sphere of influence.

The MS factor undergoes a different transformation of the obtained intensity into the color of the cloud. In this case, the phase function used is isotropic. Additionally, the underlying SS algorithm is still in use, but its absorption coefficient is higher to effectively illuminate only directly illuminated parts of the cloud. This action increases the dynamic range deep in the cloud, where the MS algorithm plays a greater role.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

4 Experiments and results

In order to compare the results, images generated for different camera angles and parameters affecting the performance and quality of the render were compared separately for each method. Additionally, images of differences between selected scenarios have been processed to show in which parts of the images they are the largest. The difference in speed and accuracy of the single-scattering algorithm depending on the presence of the precalculation mechanism is shown. Based on a precalculated integral, single-scattering and multi-scattering are compared to reference method. Reference images are obtained from Blender using *Principled Volume* shader.

The generated images have a resolution of 1024×1024 pixels. Renders of the same shots are compared (identical camera position, cloud position, light direction). Frame render time was also documented. For each scenario, there are settings that change from the default ones and affect performance and image quality (Tab. 1).

Table 1. Rendering settings. Columns "hq" show hight quality settings and columns "hp" show high performance settings.

Parameter	SSBF		SSC		MS	
	hq	hp	hq	hp	hq	hp
Primary ray lenght	0.20	0.60	0.05	0.60	0.05	0.60
Shadow ray lenght	0.11	0.60	0.60	0.60	0.60	0.60
Integral multiplier	22.00				5.00	
Intensity multiplier	3.573				6.414	
Lorenz-Mie function multiplier	1.00				0.00	
Influence factor exponent	n/a				2.00	
Radius of the sphere of influence	n/a				0.97	
Points	n/a				11	8
Rand on	n/a				1	0

This part presents the results of the implemented algorithms compared with reference images. Each scene shows an identical cloud and light source setting. Different scenarios arise due to changes in camera position and cloud density, which should introduce sufficient variability in the algorithm's input data.

The following terminology is used hereinafter:

- SS – Single Scattering – Single scattering algorithms in a general sense.
- SSBF – Single Scattering Brute Force – An algorithm implementing single scattering without precalculation support.
- SSC – Single Scattering Cached – Single scattering algorithm supported by a precalculated photon map.
- MS/MSC – Multi Scattering Cached – Multiple scattering algorithm supported by the same photon map.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

– Reference – Render using *Principled Volume* shader.

Full image quality (shown in all subsequent renders) is achieved after generating 100 frames of a still image. All time measurements show the time it takes to generate a single frame.

4.1 Single Scattering Brute Force

This section contains visual (Fig. 2) and performance (Fig. 3) results for the SSBF method with high quality and high performance settings settings. Cloud rendering with a lower density multiplier has a longer single frame render time. There is a visible reduction in the detail of the generated image and an increase in noise for high performance settings.

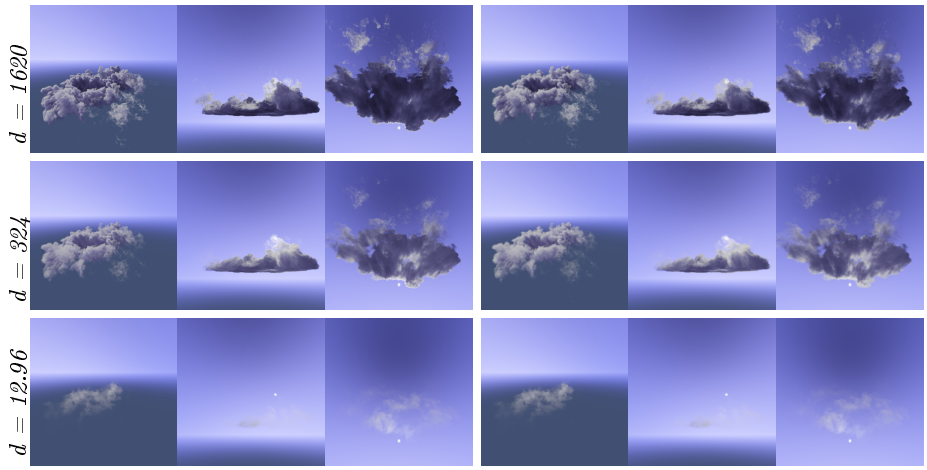

Fig. 2. SSBF, high quality (3 columns on the left) vs. high performance (3 columns on the right), various points of view (one per column), various density multipliers d.

4.2 Single Scattering Cached

Table 1 shows the settings for this algorithm. Unlike the SSBF algorithm, no shadow ray is used because its results have been precalculated and discretized within a single voxel.

Figure 4 shows visual results for the method. For high quality settings the images obtained are very similar to those generated by the SSBF method.

The high performance setting is characterized by an increased primary ray length. As in the case of SSBF, there is a visible reduction in the level of detail in similar areas. Moreover, in the case of dense areas visible from the observation point in the direction of incidence of light, a significant darkening of directly

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

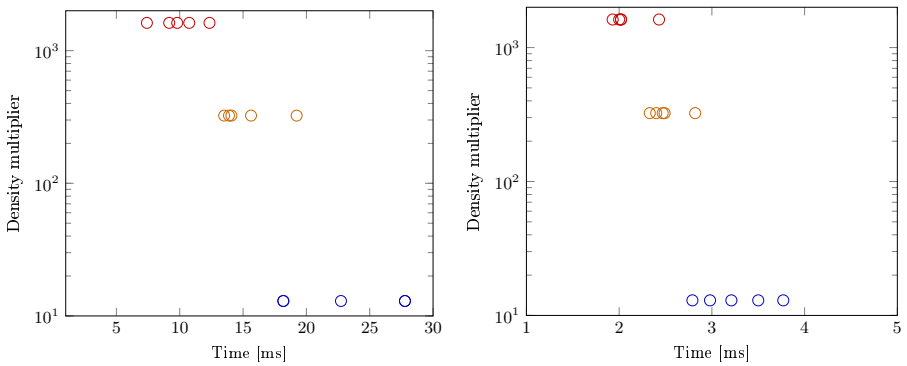

Fig. 3. SSBF, high quality (left) and high performance (right). Frame generation time vs. cloud density multiplier d.

illuminated areas is visible. The frame generation time is inversely proportional to the cloud density multiplier (Fig. 5).

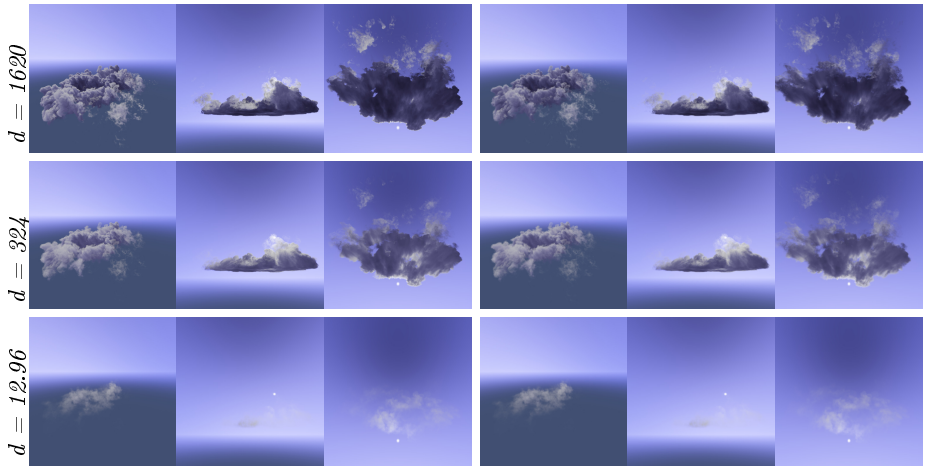

Fig. 4. SSC, high quality (3 columns on the left) vs. high performance (3 columns on the right), various points of view (one per column), various density multipliers d.

4.3 Multi Scattering Cached

Table 1 contains the settings for the algorithm. The length of the primary ray and the number of sampled points have an influence here. In each frame of the animation, the position of the sampling points is randomized. For high quality

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

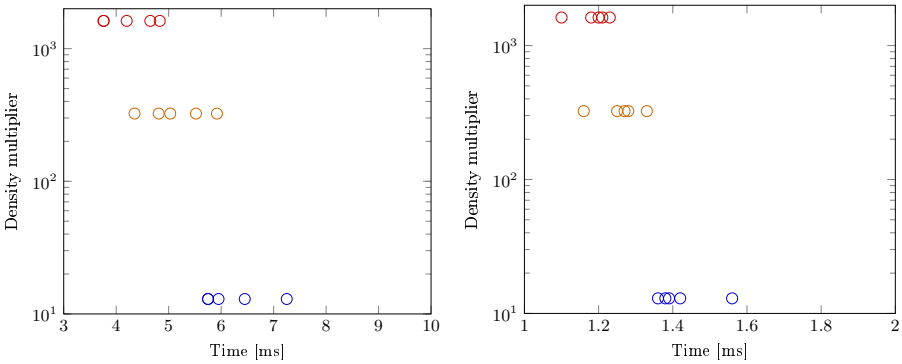

Fig. 5. SSC, high quality (left) and high performance (right). Frame generation time vs. cloud density multiplier d.

settings, unlike SS renders, details and increased brightness of cloud fragments that are not directly lit are visible (Fig. 6).

For high performance settings, the length of the primary ray has been increased and the number of sampled points has been reduced. Randomization of points per frame is disabled to speed up image stabilization.

High-performance render is characterized by a reduction in the detail of the cloud mass representation, similar to the rest of the algorithms. The details created by MS operation do not differ significantly. However, it can be suggested that a larger number of points and their randomization in each frame (the results are stabilized by the TAA algorithm) are not necessary in the case of high performance settings in order to obtain a detailed representation of the cloud.

The frame generation time compared to SS algorithms has been significantly increased (Fig. 7). It is inversely proportional to the density multiplier of the generated volumetric structure.

4.4 Comparison with the reference method

In Figure 8 (top) there is a big difference in contrast due to the different color management system. In addition, in the case of the SS algorithm, the cloud is perceptually denser. In dark parts of the cloud, the SS algorithm highlights less details than the reference image. The texture of the low-density cloud (Fig. 8 bottom) is very similar for both algorithms. The backlight view (Fig. 9 top) of a dense cloud highlights the differences in the *silverlining* effect of both methods, which is more intense in the case of reference. The same shot for a thin cloud (Fig. 9 bottom) shows a similar effect, with the difference that it is visible throughout the entire volume of the cloud.

The use of the MS algorithm (Fig. 10 top) shows a similar nature of details in the dark parts of the cloud to those present in the reference image, but their intensity is lower. In the case of thin clouds (Fig. 10 bottom), the effect is similar to the SS algorithm (Fig. 8 bottom). The application of the MS algorithm

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

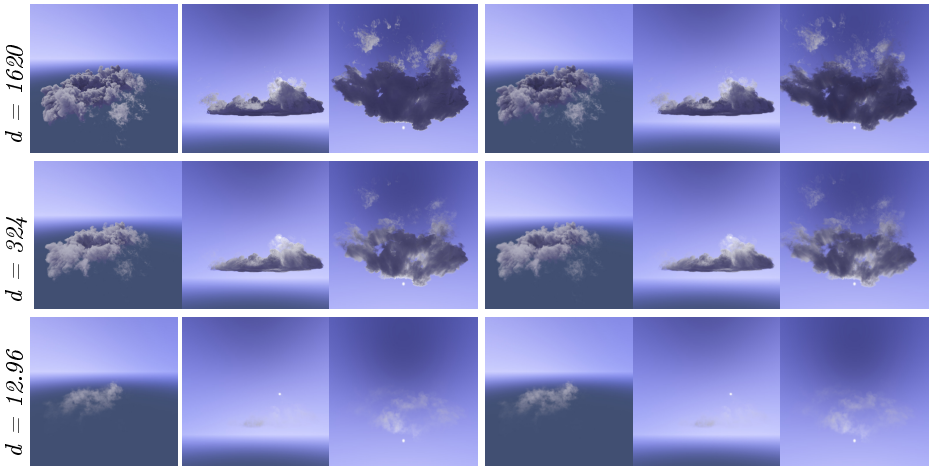

Fig. 6. MS, high quality (3 columns on the left) vs. high performance (3 columns on the right), various points of view (one per column), various density multipliers d.

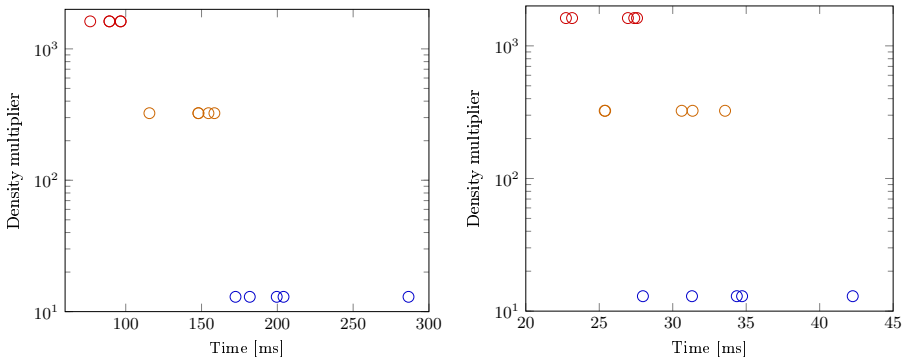

Fig. 7. MS, high quality (left) and high performance (right). Frame generation time vs. cloud density multiplier d.

(Fig. 11 top) shows a similar difference to that in Fig. 10 (top). The effect of *silverlining* does not change because it comes from the underlying SS algorithm. In the case of thin clouds (Fig. 11 bottom) the situation is similar to Fig. 9 (bottom).

4.5 Summary

The algorithm based solely on traditional single scattering effectively reflects the appearance of low-density clouds and fragments of dense clouds that are directly illuminated by a light source. Areas hidden in their own shadow are characterized by a lack of contrast and a low level of detail. Using our multi

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

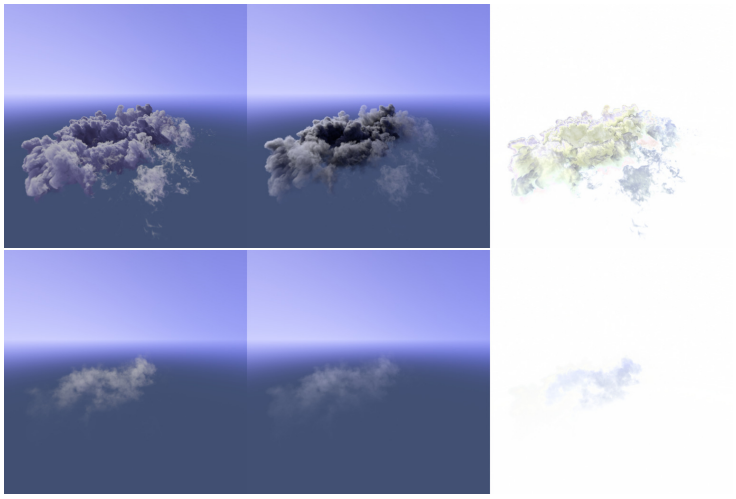

Fig. 8. Comparison of SS with the reference, and difference, *d=1620* (top) *d = 12.96* (bottom), the first point of view.

Fig. 9. Comparison of SS with the reference, and difference, *d=1620* (top) *d = 12.96* (bottom), the second point of view.

scattering approximation, details are obtained in these parts of the structure. It also produces a visual effect that is less different from the reference images than the initial SS algorithm.

The lengthening of the primary ray reduces the effectiveness of reflecting the cloud density, and the longer shadow ray in the case of the SSBF algorithm reduces the detail of determining the brightness of a given cloud fragment.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

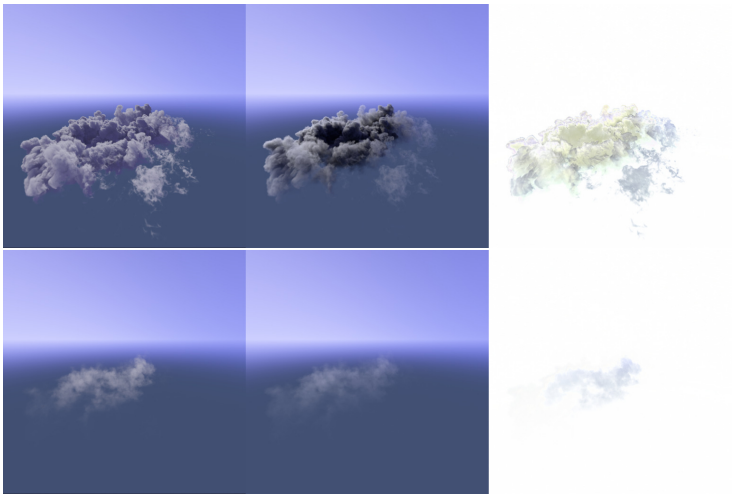

Fig. 10. Comparison of MS with the reference, and difference, *d=1620* (top) *d = 12.96* (bottom), the first point of view.

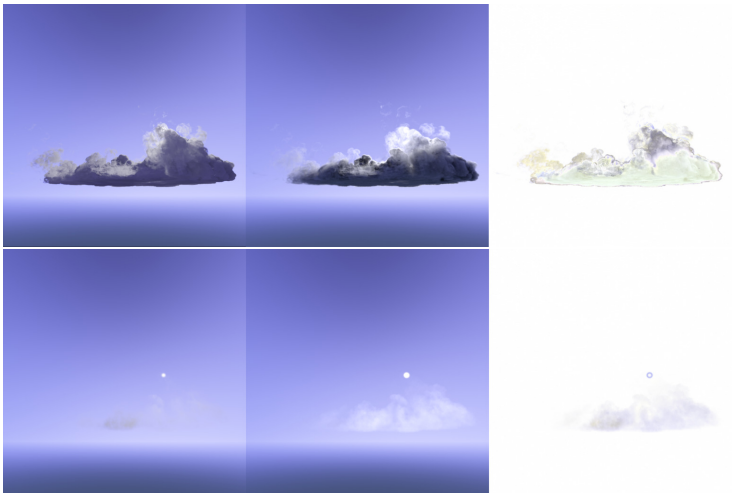

Fig. 11. Comparison of MS with the reference, and difference, *d=1620* (top) *d = 12.96* (bottom), the second point of view.

For each implemented algorithm, a performance degradation is observed in proportion to the reduction of the cloud density multiplier (Tab. 2). In thin clouds, the difference between settings with different performance is also reduced. What is important from the point of view of the MS algorithm is the number of points and randomization in each frame for the final effect. There are visible differences, although they are small.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

Table 2. Average frame render times (in milliseconds) for various density multipliers.

	SSBF		SSC		MS	
density	hq	hp	hq	hp	hq	hp
1620	9.90	2.08	4.24	1.18	89.66	25.56
324	15.27	2.50	5.13	1.26	144.90	29.26
12.96	22.93	3.25	6.23	1.42	208.89	34.12

5 Discussion

Slow performance was observed for each algorithm when rendering thinner clouds. This results from the use of optimization, which samples the denser structure with the primary ray only to the point where it is profitable. during the operation *alpha* is accumulated. Exceeding the limit value (here it is 95%) stops sampling because the influence of further samples on the final color is negligible. Thinner clouds are less sensitive to step length in terms of final render quality, so increasing the step length should allow to recover the performance lost due to the above-mentioned reason.

The use of multi-scattering is profitable only in the case of dense clouds. Below a certain density, the capabilities of the single-scattering algorithm do not differ much from those of reference renderers.

The generated images were created as a result of a different *tone mapping* process, therefore the reference images are more contrasting than those generated using the presented methods. Nevertheless, when comparing the performance of the MS and SS algorithms, it can be concluded that the characteristics of the details are more similar to the reference images in the case of the MS implementation. MS algorithm, while generating an image slower than any SS method, it is still significantly faster than ray tracing used in *Principled Volume* shader.

The use of precalculation results in a significant increase in performance at the expense of increased demand for GPU memory. To avoid this problem, a different data arrangement would be needed in the SSBO structure to avoid unfavorable *padding*.

6 Conclusions

The assumption of constant lighting conditions allowed the use of precalculation of SS calculations, while the analysis of the characteristics of phenomena occurring in high-albedo clouds and other algorithms based on MS allowed the creation of a simplified method that can be used in interactive applications.

The resulting solution renders the volumetric structure in a way closer to the reference method than the initial algorithm (SS). However, it was not possible to achieve effects identical to the images generated in *Blender*. The main problem is the use of a different color transformation at the post-processing stage in both programs, which is the reason for differences in contrast and color tone.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12

Differences in the nature of clouds with different densities were also noticed. In the case of thin clouds, the SS algorithm is sufficient to generate images similar to the reference methods. Thin clouds also have lower step length requirements for *ray marching*. Dense clouds, however, benefit from optimization based on their properties. On the other hand, they require the use of an MS-based algorithm to generate more realistic results.

Rendering times allows the use of the proposed methods for real-time rendering, especially for the Single Scattering Cached algorithm where the times are single milliseconds per frame. For Multi Scattering Cached, the high performance setting reaches times of approximately 30 milliseconds, which also allows for attempts to be used in real-time rendering. The high quality setting in MSC gives times above 100 milliseconds. Future works could consider improving these times.

References

1. Silver lining and cloud iridescence, `http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/opt/wtr/ir.rxml`, accessed: 3 Feb 2024
2. 3D Art: Free VDB clouds by VFX assets, `https://www.3dart.it/en/free-vdbclouds/`, accessed: 19 Dec 2022
3. Deng, H., Wang, B., Wang, R., Holzschuch, N.: A practical path guiding method for participating media. Computational Visual Media **6**(1), 37–51 (Mar 2020). https://doi.org/10.1007/s41095-020-0160-1
4. Eric Heitz, L.B.: Distributing monte carloerrors as a blue noise in screen space by permutingpixel seeds between frames (2019), `https://hal.archives-ouvertes.fr/hal-02158423/le/blueNoiseTemporal2019\slides.pdf`, accessed: 2 Jan 2023
5. Foundation, A.S.: Openvdb, `https://www.openvdb.org/`, accessed: 19 Dec 2022
6. Harris, M.J., Lastra, A.: Real-time cloud rendering. Computer Graphics Forum **20**(3), 76–85 (Sep 2001). https://doi.org/10.1111/1467-8659.00500
7. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. ACM SIGGRAPH Computer Graphics **18**(3), 165–174 (Jan 1984). https://doi.org/10.1145/964965.808594
8. Kallweit, S., Müller, T., Mcwilliams, B., Gross, M., Novák, J.: Deep scattering: rendering atmospheric clouds with radiance-predicting neural networks. ACM Transactions on Graphics **36**(6), 1–11 (Nov 2017). https://doi.org/10.1145/3130800.3130880
9. Keller, A.: Quasi-monte carlo methods in computer graphics: Theglobal illumination problem, `https://www.semanticscholar.org/paper/Quasi-Monte-Carlo/579bca8c938f1e0f25474a02a493f5b6ea8886`, accessed: 23 Jan 2024
10. Koerner, D., Portsmouth, J., Sadlo, F., Ertl, T., Eberhardt, B.: Flux-limited diffusion for multiple scattering in participating media. Computer Graphics Forum **33**(6), 178–189 (Mar 2014). https://doi.org/10.1111/cgf.12342
11. Mukhina, K., Bezgodov, A.: The method for real-time cloud rendering. Procedia Computer Science **66**, 697–704 (2015). https://doi.org/10.1016/j.procs.2015.11.079
12. Nishita, T., Miyawaki, Y., Nakamae, E.: A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In: Proceedings of 14th SIGGRAPH '87. ACM (Aug 1987). https://doi.org/10.1145/37401.37437
13. Yusov, E.: High-performance rendering of realistic cumulus clouds using precomputed lighting (2014). https://doi.org/10.2312/HPG.20141101

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_12