Lightning State of the Art

Duarte Nuno Pereira Moreira

***Abstract*—This article presents a state of the art on the topic of Lightning in the field of computer graphics. Realistic ray lighting is a fundamental challenge in rendering virtual scenes. Through the analysis of previous works in this area, the most relevant techniques and methods are identified, as well as the approaches used to simulate and render rays convincingly. Furthermore, the article explores the latest innovations in the field, including advanced ray simulation methods and rendering techniques that enhance the appearance and realism of rays, as well as their processing time. Based on these approaches, recent advances and future trends in the area of ray lighting in computer graphics are discussed. Finally, conclusions and recommendations for future research in this complex and challenging domain are presented.**

***Index Terms*—lightning, thunder, real-time, computer graphics, rendering**

I. Introduction

The artificial or synthetic reproduction of natural landscapes has been one of the main focuses of the computer graphics community. Much attention given to this topic leans towards rendering of vegetation (such as fields, trees, and plants), mountainous terrains, clouds, fire, among others [1]. This increasingly realistic simulation capability of real-world elements has always been an important part of the field of computer graphics, from its use in games to films, as it enhances the sense of immersion for viewers.

Surprisingly, a natural phenomenon often overlooked and neglected in this field is lightning. Or at least, the works and research conducted in this area are scarce, as stated by **Reed** and **Wyvill** [1]. The complex and chaotic nature of the electrical effects of lightning makes them challenging to simulate and animate, which may explain the aforementioned problem.

Lightning or thunderbolts are essentially static electricity, as we are familiar with in our daily lives, but on a much larger scale. To human eyes, lightning appears to behave randomly, and each bolt is different from the others. Knowing this, how can we create a similar effect where no two lightning bolts are alike, yet they all have a realistic appearance and visual appeal?

In this work, we will explore the state of the art in real-time simulation and rendering of lightning in the field of Computer Graphics and how we can address these questions.

Organization: The remaining content of the article is organized as follows. Firstly, in Section II, a brief explanation of what lightning is and how it is formed is presented. Then, in Section III, some of the works conducted throughout history regarding lightning rendering are studied. In the subsequent phase, in Section IV, the state of the art related to the topic is presented. Finally, a brief conclusion is provided in Section V.

II. Lightning

A lightning bolt is nothing more than a giant spark of electricity in the atmosphere between clouds, the air, or the ground. In the initial phase, the air acts as an insulator between the positive and negative charges within the cloud and between the cloud and the ground. When the opposite charges accumulate sufficiently, the insulating capacity of the air breaks down, resulting in a rapid discharge of electricity that we know as lightning.

A lightning bolt can occur between opposite charges within a storm cloud (intra-cloud or cloud-to-cloud lightning) or between opposite charges in the cloud and on the ground (cloud-to-ground lightning) or the reverse (ground-to-cloud lightning). Sometimes, cloud-to-cloud lightning can branch out of the clouds but not reach the ground, and these are referred to as cloud-to-air lightning [2]. In Figure 1, we can observe all these different types of lightning bolts.

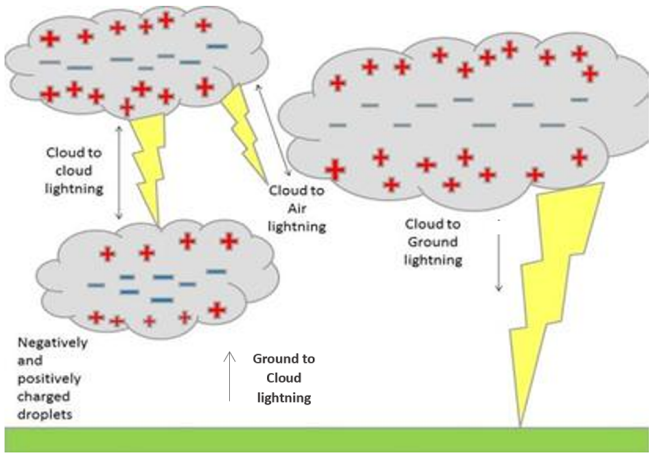

Fig. 1. Types of lightning. **Source**:Royal Meteorological Society

A lightning bolt exhibits certain properties in its structure that are necessary to understand in order to simulate it. Typically, it consists of three main parts: the first stroke, labeled as a stepped leader, the subsequent less branching strokes called dart leaders, and the secondary branches formed by the expanding leaders. In Figure 2, we can observe the general process behind a lightning stroke. The stepped leader itself is initiated by an unknown process termed the preliminary breakdown (a). The descending stepped leader forms a jagged

and branching channel, still invisible, that attracts an upward positive leader from the ground, thus starting the attachment process (b). The meeting of the downward and upward leaders triggers the first return stroke, which is responsible for illuminating the channel, and creating thunder (c). The return stroke begins at the point of contact between the down- ward and upward leader, usually close to the ground. If sufficient charge remains in the cloud, dart leaders may travel through the residual channel and create subsequent return strokes (d).

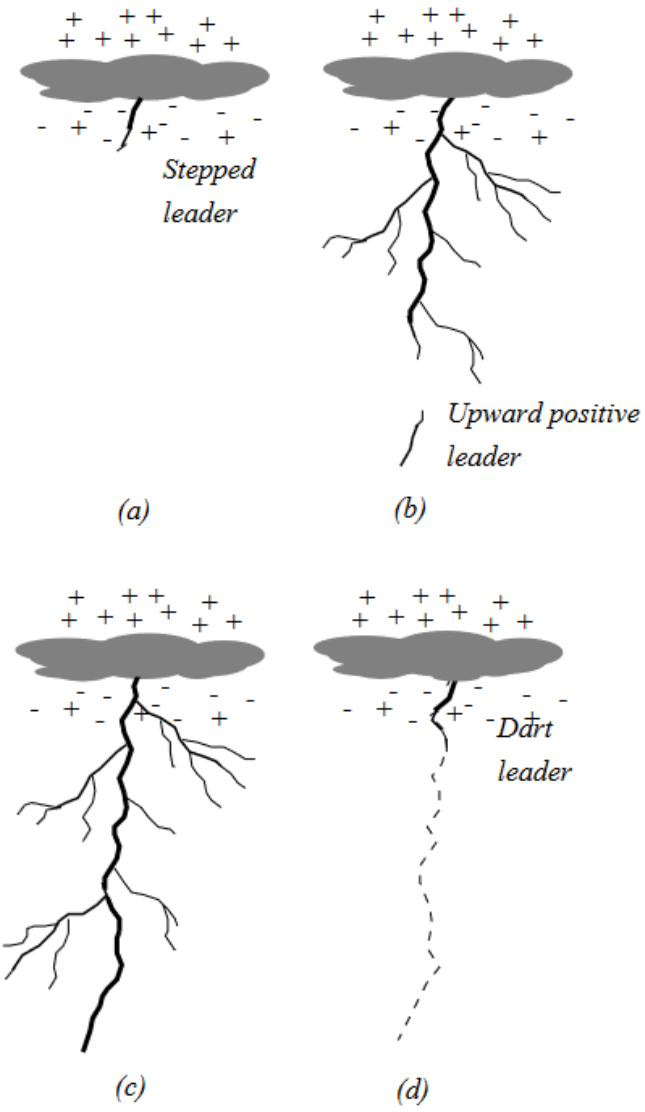

Fig. 2. Lightning flash. **Source**: [1]

III. Previous Work

As mentioned earlier, in this chapter III, we will delve into some of the works conducted in the rendering and simulation of lightning over the past years.

A. Visual Simulation of Lightning

[1] written by **Todd Reed** and **Brian Wyvill** in 1994, considered as the first published paper on this subject, the authors present a simple method for modeling and rendering lightning and objects struck by lightning using ray-tracing techniques.

To create the structure of the lightning bolt, the authors utilize a particle system that simulates the progression of the stepped leader to the ground zero, which is the point of contact of the lightning on the ground, for example. During this process, branches, i.e., the branching of the main channel of the lightning, are also generated recursively. To achieve this, a uniform distribution is used to allocate a number of segments to each branch. These branches will ”grow” until the respective number of segments runs out or the lightning reaches the ground zero. The level of branching is determined by a probability function, with branches occurring more frequently near the ground.

Regarding the rendering method, the authors decide to make some modifications to conventional ray-tracing techniques since a lightning bolt cannot be treated as a geometric object, as it is created using line segments. Therefore, they modify the shading function by adding a color contribution from the lightning bolt. To reproduce the glow effect around the main channel of the lightning, they also incorporate a secondary illumination function to the aforementioned modification.

The final result achieved by **Todd Reed** and **Brian Wyvill** in the presented work can be seen in Figure 3.

Fig. 3. Lightning over a plane of water. **Source**: [1]

B. A probabilistic technic for the syntetic imagery of lightning

[3] written by **Paul Kruszewski** in 1999, the author presents a probabilistic method for generating lightning. In contrast to [1], where a branching pattern is used, **Paul Kruszewski's** study is based on an algorithm derived from the random binary tree theory.

In addition, aiming to address one of the issues encountered in the work of **Todd Reed** and **Brian Wyvill**, "the erratic behavior of the pseudo-random number generator used made it difficult to consistently control branching" [1], where it was possible to control the forking process through probability parameters but not the randomness itself, **Paul Kruszewski** overcomes this with the split tree method, which allows control over both parameters. Figure 4 illustrates how the skeleton of a lightning bolt varies for different values of these parameters.

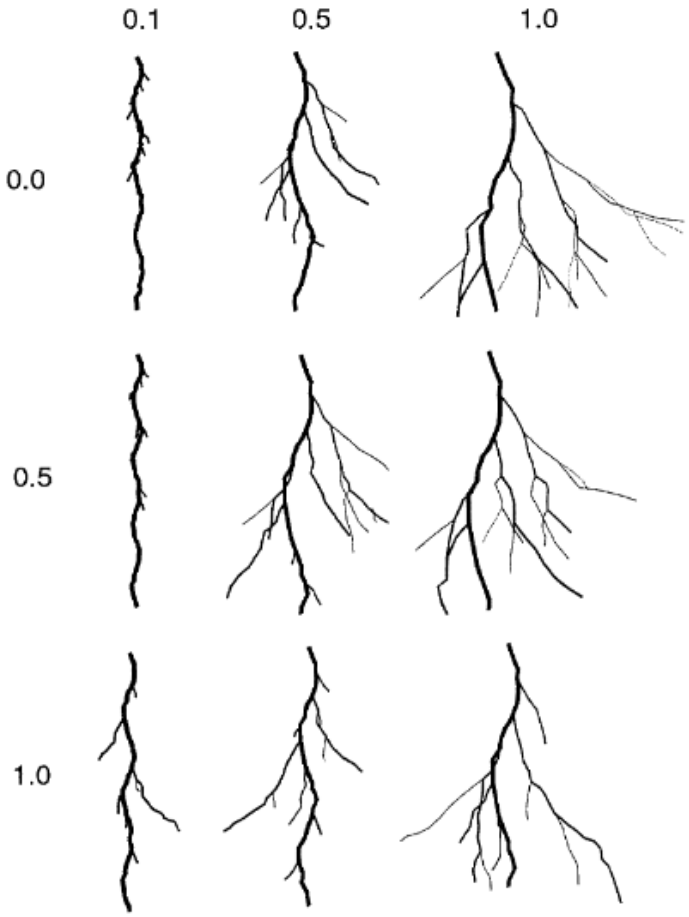

Fig. 4. Skeleton shape is a function of forking (horizontal) and randomness (vertical). **Source**: [3]

To finalize the modeling process, the segments created are transformed into three-dimensional objects by calculating their geometries. Each segment is shaped as a cylinder, thus modulating its form.

An interesting process used in this work is electrification, which involves deforming a segment to give it a more realistic and chaotic appearance. For this, let's consider a segment AB with a midpoint C. A random point is chosen within a circle centered at C with a predefined radius. Finally, the algorithm replaces the segment AB with the segments AC and CB. Figure 5 facilitates the visualization of this process.

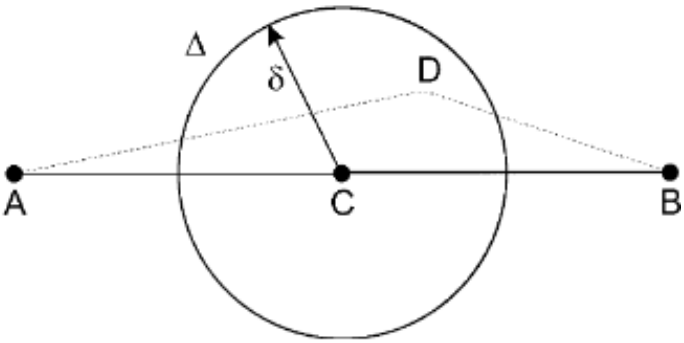

Fig. 5. Electrification breaks up skeleton segments into chaotic- looking subsegments. **Source**: [3]

Finally, regarding the rendering of the lightning bolt, it is represented as a three-dimensional mesh, and a simulated glow effect is added using default glow shaders. By applying a post-processing effect, a simple aura is added to the mesh. The resulting image can be seen in Figure 6.

C. Visual Simulation of Lightning Taking into account Cloud Growth

[4] written by **Batjargal SOSORBARAM, Tadahiro FUJIMOTO, Kazunobu MURAOKA** and **Norishige CHIBA** in 2001, these authors present a method for generating lightning patterns that takes into account the physical and natural process of their creation. They simulate an electric field within the clouds that controls the electrification process and gives rise to lightning bolts. This approach is something that had not been seen until then. Another problem encountered in previous works is the fact that they only consider one of the four common types of lightning, which is cloud-to-ground. In this study, a method is also developed to render intra-cloud lightning and ground-to-cloud lightning.

Thus, the method for generating lightning patterns involves the following steps: [4]:

1) Divide simulation space into cells.
2) Place charges on both the surface of the earth and the bottom part of the thundercloud. Calculate the potencial for every cell. Simulate the irregular propagation of the leader by apllying noise and then define potencial space.
3) Set up a leader point as a starting point of a discharge. The point is chosen from the region of negative net charge in the lower part of the cloud.
4) Choose M random points arround the leader point. Calculate the eletric potential between the leader point and each of the M points. Next, calculate the progression probability for each of the M points.
5) Choose N points that have the highest progression probability among the M points. These are called candidate points. Then, from the candidate points, choose the

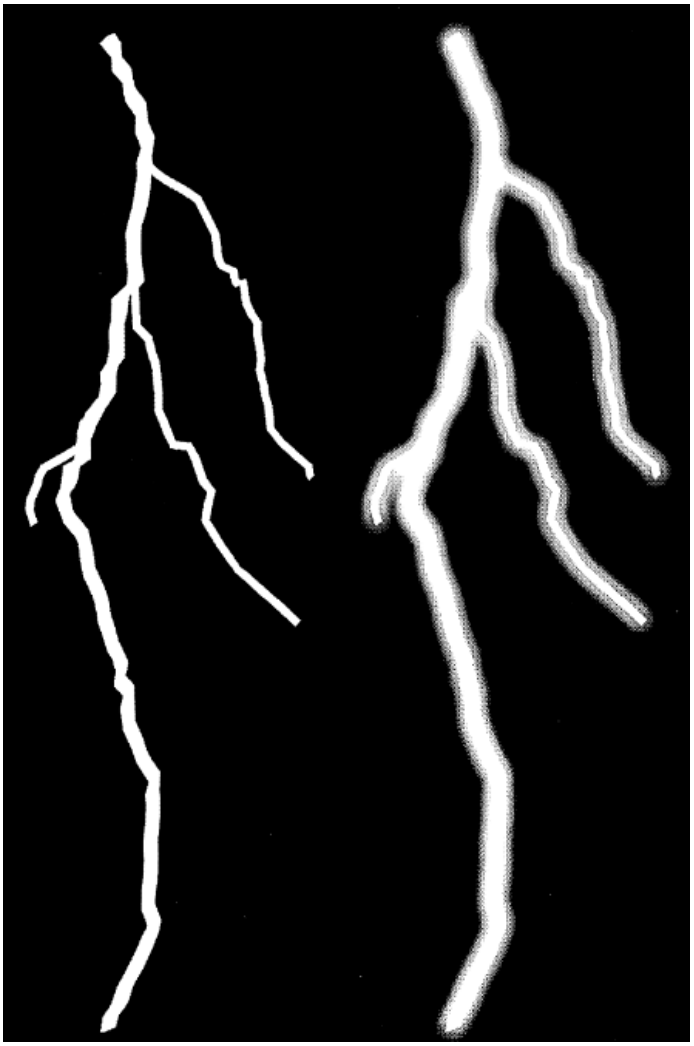

Fig. 6. Lightning before and after aura post-processing. **Source**: [3]

point that has the highest progression probability. If other points are located within a sphere of radius R centered on the chosen point, they are removed from the candidate points. This operation is repeated for each candidate point. The remaining candidate points become new leader points.

6) If no leader points reach the earth, return to Step 3. Otherwise, a lightning pattern is obtained by connecting leader points in the opposite direction; i.e. from ground to cloud.

The formulas have been omitted for readability purposes, and for more details, it is recommended to refer to the cited publication [4]. In Figure 7, some of the results obtained in the generation of typical lightning patterns can be observed. In these patterns, **Q** represents the total number of point charges, **R** is the radius of the sphere that encompasses the candidate points, and **N** is the total number of candidate points. The figure provides a visual representation of the generated lightning patterns and demonstrates the effectiveness of the proposed method in capturing the characteristic features of different types of lightning.

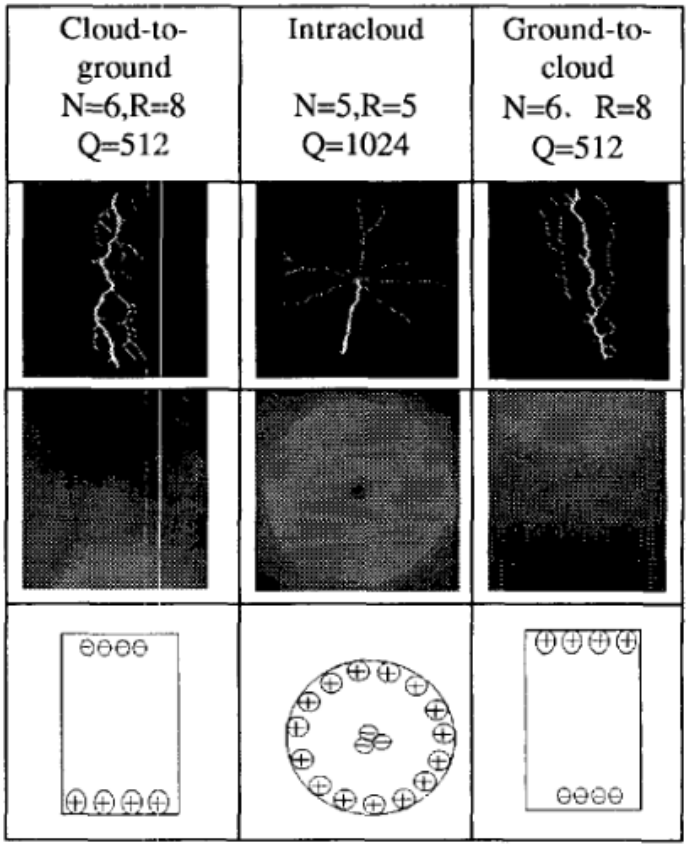

Fig. 7. Lightning patterns for different types of discharge. **Source**: [4]

In order to define the shape and render the clouds, the authors of this study follow the method proposed by **Kikuchi et al.** in [5]. They treat a cloud as a group of cloud masses, with each cloud mass represented by a single particle called a cloud particle. The modeling process involves moving each particle based on its buoyancy, the interaction force between particles, and the repulsive force between the particle and the atmosphere. By simulating these forces, the clouds can be dynamically shaped and animated, resulting in realistic and visually appealing cloud formations.

The rendering of cloud-to-ground lightning was performed by describing their patterns as combinations of textures of 3D cylindrical shapes with variable density values. The density is higher in the central area and gradually decreases in the surrounding areas. For this texture, a simple volume rendering method is applied. The resulting visual representation captures the dynamic and branching nature of cloud-to-ground lightning, as shown in Figure 8.

For intracloud lightning, as shown in Figure 9, a combination of two methods is used. The first method mentioned earlier is applied, along with another method based on single-scattering volume rendering. In this approach, a set of light sources is placed along the lightning segment. To reduce computational costs, the distance for obtaining light from

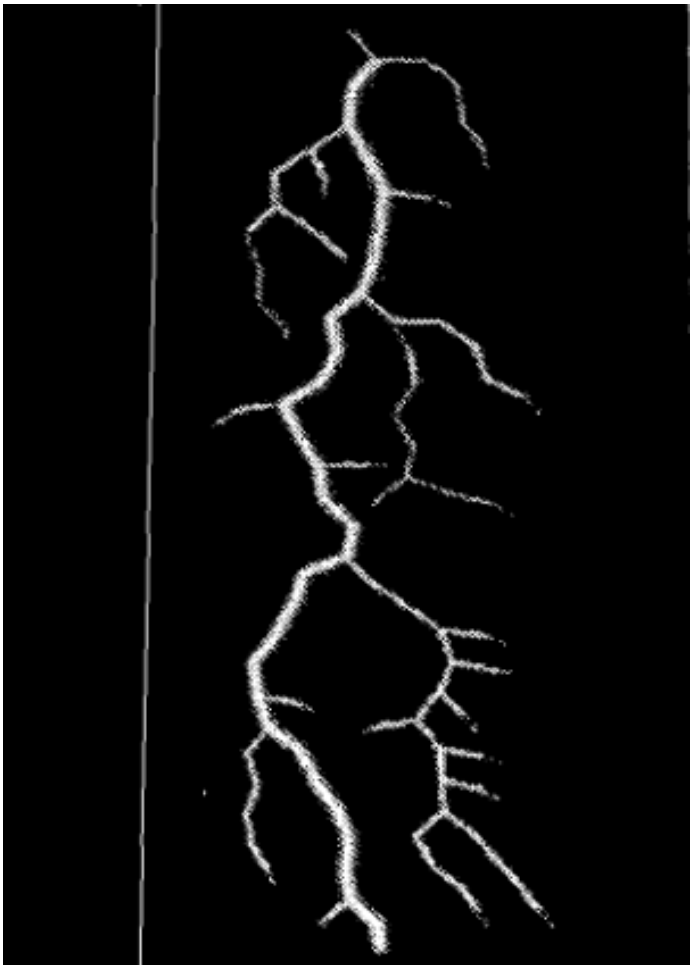

Fig. 8. Cloud-to-ground lightning. **Source**: [4]

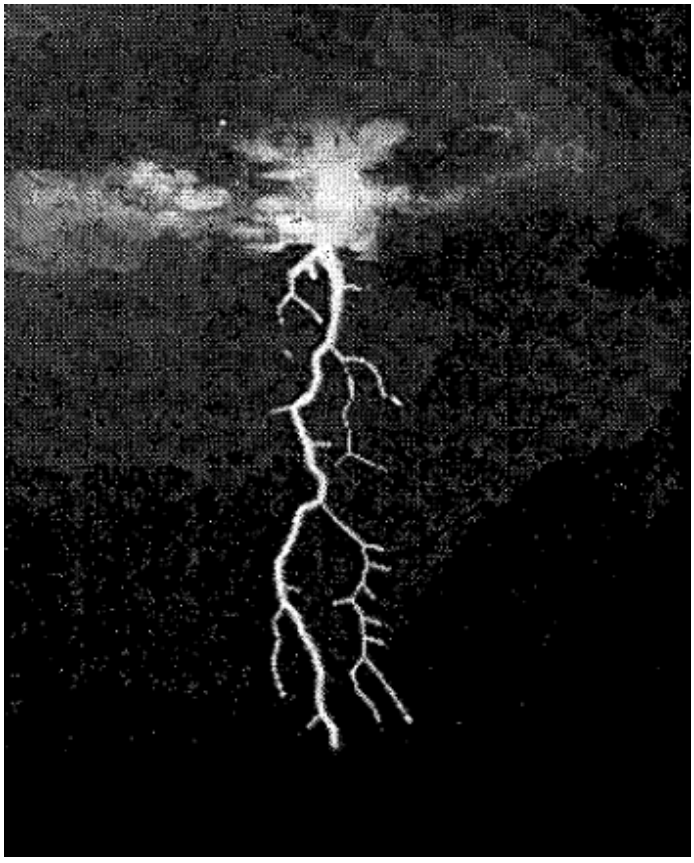

Fig. 9. Lightning with intracloud discharge. **Source**: [4]

these sources is reduced by choosing a nearby point. This combination of techniques allows for the realistic rendering of the intricate and branching patterns of intracloud lightning.

D. Efficient Rendering of Lightning Taking into Account Scattering Effects due to Clouds and Atmospheric Particles

[6] written by **Yoshinori Dobashi, Tsuyoshi Yamamoto** and **Tomoyulu Nishita** in 2001, this study presents a novel method capable of generating photo-realistic images of lightning, taking into account the presence of clouds and the surrounding atmosphere. Since a lightning bolt acts as a light source and illuminates its surroundings, the method considers the interaction of light with the atmosphere and objects. Additionally, the authors leverage graphics processing to accelerate the image generation process, enabling faster rendering times.

The generation of lightning bolts in this study relies on the method developed in [1], representing them as line segments. As for the intensity of the clouds and the corresponding scattering (the area surrounding the lightning), it is achieved by creating points along the line segments that function as light sources. This can be observed in Figure 10.

To alleviate the computational power required for rendering both the clouds and the scattering, which involves summing the contributions of each light source, the authors address this issue by treating the clouds as metaballs. Each metaball is associated with a density value and a radius. However, since satisfactory results were only achieved with a large number of metaballs, the authors decided to hierarchically group them to decrease the level of detail of distant metaballs. Thus, the density distribution of the clouds was represented by these structures called octrees (a group of eight metaballs), which allowed for efficient rendering of the clouds illuminated by the lightning by applying the concept of level of detail.

Finally, to further increase the efficiency of lightning generation, the authors introduce a method that implements the use of hierarchical impostors in creating images of clouds as seen from the viewport (camera). An impostor replaces an object with a textured, semi-transparent polygonal image of the same object, significantly reducing rendering times. One of the results obtained in this work is shown in Figure 11.

E. Physically Based Animation and Rendering of Lightning

[7] written by **Theodore Kim** and **Ming C. Lin** in 2004, the authors present a simulation technique based on convolutional methods. The main goal of the work is for this simulation technique to be able to compete with Monte Carlo ray-tracing. Additionally, they emphasize the manipulation of the final

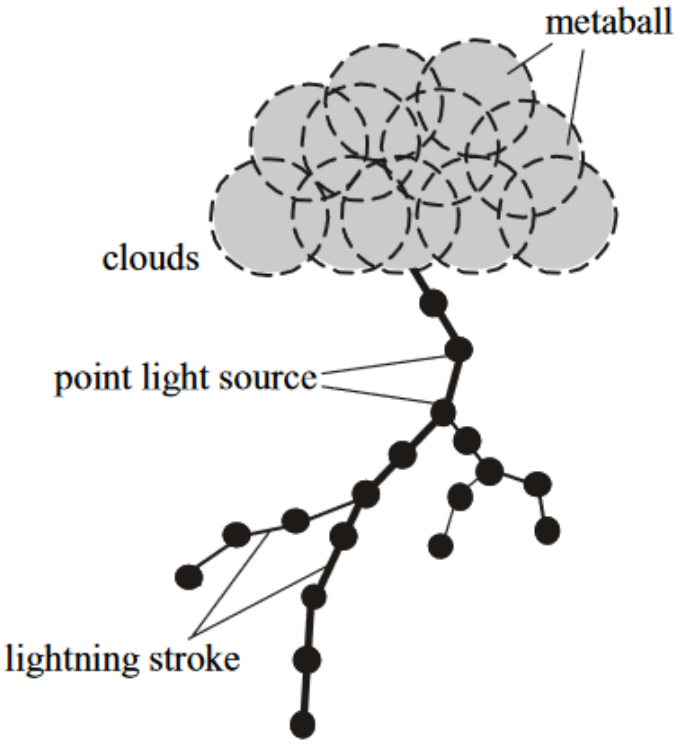

Fig. 10. Point light sources are placed on lightning strokes to compute the intensity of clouds illuminated by lightning and the glow of lighting due to atmospheric scattering. **Source**: [6]

Fig. 11. Multiple lightning strokes. **Source**: [6]

result through the user's control over various parameters. This includes controlling branching, the density field to guide the path of the lightning, a boundary condition to make certain objects repel the lightning, and the option to choose the initial and final points of the lightning.

The simulation is inspired by the Dart Leader Breakdown Model, as shown in Figure 12. In Figure 12b), the initial point of the lightning is represented in shades of gray, specified with negative charge values, while the ground is depicted in black, representing positive charge values. To make the result more realistic and physically accurate, the authors make some modifications to the model in order to simulate the subsequent lightning strokes known as dart leaders, which follow the initial stepped leader. For this purpose, they are based on the hypothesis that the stepped leader leaves behind residual charges that the dart leaders will follow.

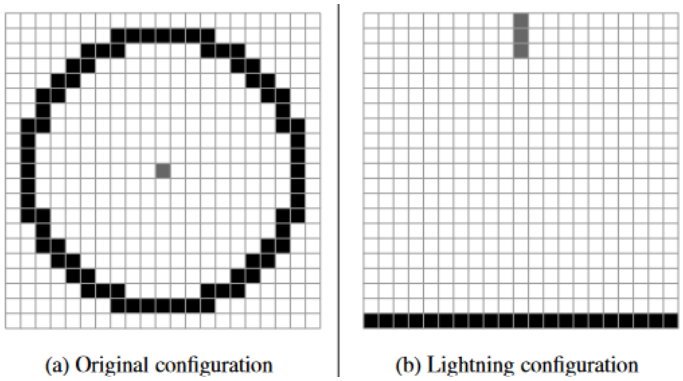

Fig. 12. (a) Configuração inicial. (b) Configuração ajustada para o raio. **Source**: [7]

In terms of rendering, the authors utilize a function called the "Atmospheric Point Spread Function," which, in general, represents a three-dimensional function that describes how light scatters around a specific light source in space. By determining how a point light source scatters in the image plane, this function can be used as a kernel to render light sources of arbitrary shapes. To simulate the thickness of the lightning plasma, Kim and Lin modeled a series of thin line segments and applied the mentioned kernel to create the glow effect.

The authors summarize the rendering process in three main steps:

1) create a graph based on the simulation, Figure 12.
2) assign different luminance values to each edge of the graph, as the main branch of the lightning, for example, is brighter than its branches.
3) render the edges as line segments and apply the Atmospheric Point Spread Function (APSF) to them.

The results of this study can be observed in Figure 13.

IV. State of the Art

A recent study worth mentioning and analyzing is [8], conducted by **Nuno Reis** and **António Fernandes** in 2022. This dissertation presents an alternative approach for the random and iterative generation of lightning based on the Space Colonization algorithm. This algorithm was originally designed for botanical purposes, specifically for modeling the growth of branching networks found in leaves or trees. However, the author identifies a similarity between the branching process of lightning and that of a tree, as depicted in Figure 14.

Thus, the modifications made to the algorithm for lightning simulation are divided into two phases:

1) reproduction of the main channel
2) natural and user-defined introduction of branches

Fig. 13. (left) Validation lightning. (right) Rendered lightning. **Source**: [7]

Fig. 14. Uncanny resemblance between a rare, upwards lighting and a tree. **Source**: [8]

Starting with the first phase, reproduction of the main channel, the author, as mentioned earlier, explores the similarities and differences between trees and lightning, and how they can be adapted for algorithmic purposes. Thus, the concept of the main channel of a lightning bolt is introduced, along with its search for positive charges to establish a connection. Inspired by the trunk modulation in tree-related studies, specifically in [9], a ascending vector is used to initiate the propagation of the lightning.

They then modify the algorithm to guide the main channel along a specific path, mimicking the behavior of a stepped leader that detects nearby attractors. Additionally, a constraint is introduced to allow only the last created node to be influenced by the attractors, resulting in a single channel. To control the degree of deviation from the central path, changes are made to variables such as the width and length of the column.

By implementing these modifications, a problem arises where the main channel keeps searching for charges infinitely, leading to undesirably long processes. To address this issue, a stopping condition is added where the algorithm stops when the ray reaches or surpasses a reference plane. Additionally, to resolve erratic behavior during the propagation of the stepped leader, weighted growth directions are introduced based on the ray's attraction towards the ground.

The author decides to expand this concept by introducing waypoints that the user can create to control and direct the path of the ray. Based on previous studies, the use of a reference plane or a single point is suggested as stopping measures, and dynamic calculation of weighted vectors is proposed based on the direction between the nodes and the waypoints. As a result, the algorithm is improved to allow users to specify sequences of waypoints, which serve as points to guide the path taken by the ray. The main channel propagates from one waypoint to another until the entire sequence is explored.

In regards to the second part, concerning the branching of the ray, the main objective was to allow secondary and tertiary rays to connect to the main channel in a natural way or with user input.

As the main channel is actively constructed, at certain points, based on a probability distribution and a threshold, branching can occur. This branching process is influenced by appropriate probability distributions that can simulate effects observed in real lightning, such as the fact that the probability of branching increases as the ray moves away from the starting point, as we have seen so far. To create a new branch, termination conditions are defined, such as the coordinates of an endpoint, using spherical coordinates. The direction of the branch is determined based on the distance of the point to the surface of the plane that contains the endpoint of the main channel. The direction and length of the branch are established to replicate the natural growth of lightning.

Once the branching is defined, the distribution of charges is applied, following the same process used for the main channel. The creation of envelopes and the distribution of charges within them simulate the propagation of electric charge along the branch. By defining parameters for the envelope, it is possible to create secondary and tertiary branches according to user-defined criteria, allowing for complete control over the branching structure. This enables users to customize and shape the lightning simulation according to their desired visual outcome.

Moving on to the rendering phase, the author divides this process into three distinct stages:

1) The necessary procedures for pre-processing the data to enable sequential rendering
2) The process of growing and animating a lightning bolt
3) The process of animating strobe effects and corona discharge

When it comes to the first topic, in order to render the lightning bolt in the correct order, it is necessary to sort the nodes created during its construction. Two approaches are discussed: transforming the lightning bolt into a tree-like data structure or using an index map. The second approach is considered more efficient and eliminates potential sorting errors. The result of this ordering is stored in two OpenGL index arrays. The rendering process involves populating the index arrays with information as time progresses, resulting in the growth of the lightning bolt. Two arrays are used: one for the indices of the main branch and another for the indices of the branches.

When it comes to the process of constructing the lightning bolt, the observed effect called the "stepped leader" is dis-

cussed. The stepped leader rapidly oscillates between various positions as it descends, resulting in the emergence of branches extending from the main lightning bolt. These branches exhibit very low luminosity during their formation. The goal in this phase is to realistically replicate this effect.

After constructing the index and vertex arrays, the focus shifts to rendering the lightning bolt on the screen. The program determines how many nodes should be rendered at a given moment based on time. This is achieved by comparing the current time with the start time of rendering. Using this comparison, the total rendering time interval is calculated and provided to the previously created index list. By utilizing a tag stored in each index, it is determined whether the segment corresponds to the main channel or a branch. This step is crucial for creating two separate textures, one for the main channel and another for the branches, enabling the application of distinct effects in a later stage.

Therefore, the first step is to sequentially construct the lightning bolt, simulating the descent of the stepped leader. In each frame, a fraction of the index array is passed to the renderer, resulting in the sequential images shown in Figure 46 of the paper [8]. The final state is considered when the luminosity of the lightning bolt reaches its maximum value. This effect is observed when the lightning bolt reaches the ground, and the connection between the ground and the air generates a very hot plasma that emits intense light. However, it is important to note that cameras typically exaggerate this effect due to the sensors' difficulty in rapidly adapting to changes in saturation.

Lastly, two important effects for creating realistic lightning images are addressed. These effects are interconnected and require special attention when rendering lightning. One of them is the brightness resulting from corona discharge, and the other is the stroboscopic effect.

To simulate the brightness of a corona discharge, a similar effect to bloom is used. Bloom is a post-processing technique used in computer graphics to enhance the way a bright light source is perceived by the viewer. This effect is characterized by the emergence of light streaks extending from a bright object in the image, creating a glow-like appearance around the light source. To implement bloom, the following steps are performed: capture the scene to a texture; separate the main channel from the branches; apply blur to these textures; and merge the original textures with the blurred results;

To simulate the stroboscopic effect, changes are made to the texture merging process. The process is divided into three distinct phases. In the first phase, the original textures are used without any post-processing, representing the rendering of the scene as it is. This phase captures the base appearance of the lightning bolt. In the second phase, post-processing effects, such as bloom or other visual enhancements, are applied with reduced intensity. This creates the weakening phase of the lightning's brightness. The goal is to decrease the overall luminosity and create a subdued effect. In the third phase, the post-processing effects are increased in intensity. This phase represents the intensification of the lightning's brightness. The visual enhancements are amplified, resulting in a more vibrant and powerful appearance. These three phases can be repeated in sequence to recreate the stroboscopic effect of the lightning. By alternating between weaker and stronger intensities of the post-processing effects, the lightning's brightness fluctuates, mimicking the rapid changes typically observed in stroboscopic phenomena.

Finally, one of the results obtained in this work is shown in Figure 15. These results were achieved using the *Nau3D* graphics engine.

(a) Edited to include waypoints. (b) Source: *flickr.com*.

Fig. 15. Comparing the simulation results with real lighting. **Source**: [8]

V. Conclusion

In conclusion, advances in ray simulation in computer graphics have enabled the creation of visually stunning and realistic representations. By adapting algorithms inspired by natural phenomena such as tree growth and lightning behavior, the scientific community has made significant progress in capturing the complex characteristics of lightning in virtual environments.

As seen in the last presented case study, section IV, by incorporating concepts such as attractors, waypoints, and weighted directions, models have been able to successfully replicate the unpredictable paths and branching patterns observed in real lightning cases. At this point, users have the ability to manipulate and direct the propagation of lightning, enhancing control and artistic expression in computer-generated images.

However, despite these achievements, there are still areas that require further exploration. One of these aspects is the accurate simulation of the electrical properties of lightning and its interactions with the surrounding environment. Despite the efforts made by the authors of the studies presented in sections III-C and III-D, this aspect still proves to be a challenge.

Furthermore, although the proposed algorithms have shown promising results, there is still room for the development of alternative approaches. For example, exploring machine learning techniques, such as neural networks, could open up new paths for simulating lightning in computer graphics.

Training models on a large set of real lightning images could enable the generation of even more realistic simulations.

In summary, the field of lightning rendering in computer graphics has made significant advancements in capturing the visual aspects of this phenomenon, despite the relatively limited research efforts. However, further advancements are needed to improve the physical accuracy, interactions, and computational efficiency of the simulations.

REFERENCES

[1] T. Reed and B. Wyvill, "Visual simulation of lightning," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: Association for Computing Machinery, 1994, p. 359–364. [Online]. Available: https://doi.org/10.1145/192161.192256

[2] N. NOAA National Severe Storms Laboratory, "Lightning basics." [Online]. Available: https://www.nssl.noaa.gov/education/svrwx101/lightning/

[3] P. Kruszewski, "A probabilistic technique for the synthetic imagery of lightning," *Computers Graphics*, vol. 23, no. 2, pp. 287–293, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849399000382

[4] B. Sosorbaram, T. Fujimoto, K. Muraoka, and N. Chiba, "Visual simulation of lightning taking into account cloud growth," in *Proceedings. Computer Graphics International 2001*, 2001, pp. 89–95.

[5] T. Kikuchi, "Visual simulation of cumulonimbus clouds," *The Journal of The Institute of Image Electronics and Electronics Engineers of Japan*, vol. 27, no. 4, pp. 317–326, 1998.

[6] Y. Dobashi, T. Yamamoto, and T. Nishita, "Efficient rendering of lightning taking into account scattering effects due to clouds and atmospheric particles," in *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*, 2001, pp. 390–399.

[7] T. Kim and M. Lin, "Physically based animation and rendering of lightning," in *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, 2004, pp. 267–275.

[8] N. Reis and A. R. Fernandes, "Using a space colonization algorithm for lightning simulation," in *2022 International Conference on Graphics and Interaction (ICGI)*. IEEE, 2022, pp. 1–8.

[9] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proceedings of the Third Eurographics Conference on Natural Phenomena*, ser. NPH'07. Goslar, DEU: Eurographics Association, 2007, p. 63–70.