Using a Space Colonization Algorithm for Lightning Simulation

Conference Paper · November 2022

DOI: 10.1109/ICGI57174.2022.9990427

2 authors:

Nuno Reis
University of Porto
3 PUBLICATIONS **2** CITATIONS
SEE PROFILE

António Ramires Fernandes
University of Minho
38 PUBLICATIONS **130** CITATIONS
SEE PROFILE

Using a Space Colonization Algorithm for Lightning Simulation

Nuno Reis
Department of Informatics
Universidade do Minho
Braga, Portugal
a77310@alunos.uminho.pt

António Ramires Fernandes
Algoritmi Centre/Department of Informatics
Universidade of Minho
Braga, Portugal
arf@di.uminho.pt

***Abstract*—We present a method for the procedural generation and rendering of lightning bolts. The geometric definition of the bolt is based on space colonization, exploring the similarity between natural trees and lightning bolts. Regarding rendering we aimed to show the life cycle of a lightning bolt, encompassing the three phases of its formation: stepped leader, return stroke, and dart leader. The strobing effect is also detailed in our proposal. With this work, we create an alternative to the current state of the art modeling, a physically-inspired method, using an heuristic iterative algorithm. Notably, the goal was to create a solution of rivaling quality, while being capable of real-time applications, and fully customizable, both in terms of the geometric definition as well as the final appearance and life cycle animation.**

***Index Terms*—lightning, rendering, space colonization, procedural modeling**

I. Introduction

Lightning occurs when electrically charged regions in the atmosphere, generally negatively charged, and the ground, generally positively charged, temporarily equalize themselves, causing an instantaneous release of energy. The amount of energy carried by this stream of electrons is so incomprehensibly vast that it is able to directly effect the molecular structure of the atmosphere it propagates through, turning the gaseous vastness of the skies into a brightly lit bolt of plasma. In fact, the increase in pressure from this sudden transition is so tremendous that it results in the shock wave we call thunder. These discharges from cloud to ground, labeled downward negative lightning, account for up to ninety percent of all naturally occurring lightning strikes [1].

From the inception of Computer Graphics, the simulation of physical phenomena has been a pursuit of choice for academics within the field. However, the reproduction of lightning, rather than the effect of its strikes, has been woefully underappreciated. While one of the most recent models [2] focuses on approximating physical reality as accurately as possible, the overhead that came with the associated calculations was obvious.

Parallel to the academic pursuit were the reproduction of lightning in games and movies which rely primarily on specially crafted renditions. In the former, specifically, it is not uncommon for lightning to be represented by a sprite randomly selected from a predetermined pool which leads to potential repetition.

Fig. 1. An example of a bolt constructed and rendered using our methods at the moment of maximum brightness.

With this in mind, we sought a manner to approximate the appearance of a lightning bolt while doing away with the expensive computation associated with a physical simulation. Something which might be appealing for real-time rendering while still maintaining both a degree of unpredictability and

approximation to the real life phenomena.

To this end, we explored the similarity between natural trees and lightning bolts, repurposing an existing algorithm for procedural tree generation: the Space Colonization (SC) algorithm [3]. As it stood, this algorithm had its inception in botany, first used to emulate the venation patterns of leaves [4] and later to simulate the growth of several different types of flora. The goal was to use SC as a basis through which to develop an adaptation capable of producing adequate results, as seen in Fig. 1. Besides the geometric definition, we also present the steps that allow for the rendering of the bolt full life cycle, including the strobing effect. Finally, being an heuristic method, it is fully customizable allowing for a large degree of freedom when creating and rendering bolts.

II. Literature Review

This section serves as a review of relevant literature which inspired or otherwise contributed to our method.

A. Simulating Lightning

When it comes to the manner in which lightning is simulated and its shape drawn, the observed literature can be rather cleanly split into two categories: Physically-Inspired Methods and Qualitative Methods, the latter of which mainly makes use of iterative or procedural algorithms to achieve a desired result. The former, on the other hand, makes use of mathematical and computational manipulation to generate algorithms used to simulate the phenomenon that births lighting as accurately as possible.

The first method to ever research and construct a lightning bolt in the realm of Computer Graphics was conceived by Reed and Wyvill in 1994 [5]. Through the careful observation of countless images, they were able to discern that each segment and, subsequently, each branch incurs on average a 16° rotation from the segment that preceded it. By using this value, a repeated randomization of the angle between segments results in a pattern resembling that of the observed lightning bolts.

Sosorbaram et al. [6] use the, at the time, newly discovered dielectric breakdown model (DBM), to generate lightning shapes. Rather than solving the exact equations, they made use of an approximation based on local values. This quickly became the main point of focus in physically-inspired approaches: to better solve or approximate the Laplace equations found in the DBM while sacrificing as little performance as possible.

Kim and Lin [7] propose a different method based on a conjugate gradient. However, despite their best efforts, the iterative nature of this solution still presented a considerable toll on the computational requirements. As such, they revised their work [2], and proposed a distance-based method. This approach makes use of adaptive meshes such as quadtrees and octrees, in conjunction with electrical potential equations, to produce a faster alternative.

The DBM also inspired the method developed by Bryan et al. [8] which combines a qualitative approach with a physically-based approach. Employing Cellular Automata in conjunction with potential differential equations. This was the first method able to resolve lightning in real time.

Yun et al. [9] presents a method for physically-inspired simulation. The authors discovered two integral properties present in electric potentials which can be easily described via cell-types in a grid. Combining this approximation with the DBM, allowed them to solve a gradient between each neighboring voxel and a given point, dramatically reducing the computational requirements. However, there were still times in which their method encountered local minima, a problem they solved by generating waypoints via path-finding algorithms.

Fig. 2 shows the results obtained by the works presented in this section. The result by Kim and Lin [2] is the one which better resembles real life lightning.

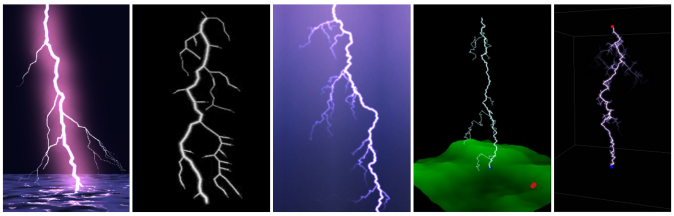

Fig. 2. Results from previous works. From left to right: [5], [6], [2], [8], [9]

B. Rendering Effects

The rendering of lightning has been a topic of contention amid those involved in its simulation. Most resort to ray-tracing to simulate the dispersion of its light across a scene while others focus on the effects of atmospheric scattering observed when it strikes, including the manner in which it interacts with clouds.

Reed and Wyvill [5] pioneered a method in which a blur is applied over the lightning's geometry, simulating the glowing body of hot plasma that is easily observed whenever lightning strikes. A technique that later evolved into bloom, however, at the time they used ray-tracing to simulate the resulting light.

Dobashi et al. [10] focused on these ray-traced effects, looking for an alternate method with which to simulate atmospheric scattering and cloud simulation. By using metaballs, precomputation and, most importantly, billboard techniques to render 3D spaces into 2D scenes before applying lighting effects, they were able to severely reduce the computational load that had been previously observed. For the lightning itself, however, a similar process to Reed and Wyvill was used.

Kim and Lin [2], [7] made use of a similar post-processing technique to simulate glow. Additionally, an atmospheric point spread function was employed to illuminate the surrounding scene. This function, when combined with a convolution kernel, can be used to emulate the scattering of light in a surrounding medium.

Yun et al [9] alter the brightness and thickness of each branch, extending billboard techniques seen in Dobashi et al. Much like the other methods previously described, a blur filter

is used to simulate glow. Due to the specific nature of their method, a soft jittering is introduced to mask any artifacts.

Notably, none of the methods seem to focus on any effects other than the light emanated by the lightning itself at its maximum brightness point.

III. Space Colonization

Space Colonization (SC) is an algorithm developed by Runions et. Al [3], [4]. It was first conceptualized as a method for simulating the venation patterns [4] in tree leaves, and latter was expanded to simulate the growth of various types of flora from trees to algae [3]. The method, outlined in Fig. 3 works by treating competition for space as a key factor in determining branch growth, relying on the concept of attractors and nodes, the former competing with each other to influence the latter's growth direction.

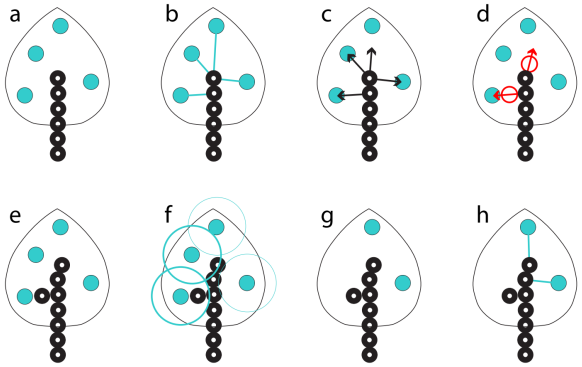

Fig. 3. The mechanism through which the space colonization algorithm expands. **A** presents the initial state of the system. Should the distance between a given node and an attractor become smaller than its radius of influence, **B** follows, representing the process after said relationship is formed. Shown in **C**, is the fact each node can have multiple vectors as it can be influenced by multiple attractors. These vectors are then added and the result is normalized, as seen in **D**, representing the growth direction, resulting in **E**. **F** show the attractor removal process, in which the minimum deletion distance is used to identify if any nodes exist within it, resulting in the removal of affected attractors. With the attractors removed, **G**, the algorithm begins anew, as seen in **H**. Source [4]

A. Properties and Traits

One of the core advantages of space colonization as a model for lightning, is that it was built to simulate natural growth. As it stands, in nature, all that exists must compete for space with those that share similar traits, a space which is necessary for something to further grow, be it flora or fauna. It stands then that structures resulting from a method employing this algorithm as a basis exhibit very natural shapes, making SC well-suited for our goal. Secondly, the manner in which it was conceptualized allows for extensive parametrization, not only that, it facilitates modification of the algorithm down to its very core behavior.

What follows from that mechanism of action are a number of entangled parameters and variables which can be used to achieve an emulation of a given piece of flora. The number of attractors and the distance at which their deletion is triggered have a tremendous influence on the shape of the tree and its branches as well as their density and own offspring. Concretely, the amount of attractors available directly influences the frequency and length of branches.

As for the deletion radius, larger numbers result in sparser structures. This is due to the decreased lifespan of each individual attractor resulting in less chances for a new node to be influenced by an attractor that had been previously reached, shortening the overall width and length of the structure.

Lastly, envelope shape, the volume in which these attractors are distributed, is directly tied to the shape of each structure. This degree of influence is directly tied to the attraction and deletion radii chosen at a given time as the algorithm is programmed to iterate until no new attractor is found within effective range.

IV. Methodology

The formation of a Lightning Bolt can be separated into three distinct phases, as seen in Fig. 4: stepped leader, return stroke and dart leaders. The stepped leader phase encompasses the formation of the bolt itself in which an imperceptible stream of negatively charged particles descends from the clouds, extending tendrils which propagate outwards. These 'feelers', as they are called, search for the path of least resistance, allowing the bolt to expand into several branches though only one results in a finished stroke. Once one of these feelers touches the ground, a return stroke manifests towards the cloud itself, what we associate with the striking of lightning and subsequently, the iconic thunder. From hereon, it is often that dart leaders travel downwards, discharging any excess negative charge to achieve equilibrium, resulting in a strobing effect.

A complete overview of this process is essential in understanding why lightning exists, how it forms and, crucially, how to best emulate them. Albeit normally imperceptible, there are times in which the stepped leader and its feelers are visible to the naked eye. Similarly, there are times in which the strobing behavior is easily observed, something that we wished to capture with the manner in which we rendered lightning. Furthermore, understanding the process behind a stepped leader's descent granted us some insight into how the space colonization algorithm could be molded to better fit the desired outcome.

The more natural interpretation for the propagation of feelers during the stepped leader phase is that of a path-finding algorithm. This is seen in Yun's [9] work and the manner in which the DBM operates, however there are other ways in which this process could be interpreted: particle density, competition for space and interference. These three concepts are emulated in SC, with attractors serving as positive charges competing for space and aggregating with a given density, interfering with one another via competing attraction vectors.

A. Reproducing the Main Channel

Our first focus was solely the reproduction of the main channel, the stepped leader, without worrying about emulating any of the surrounding branches.

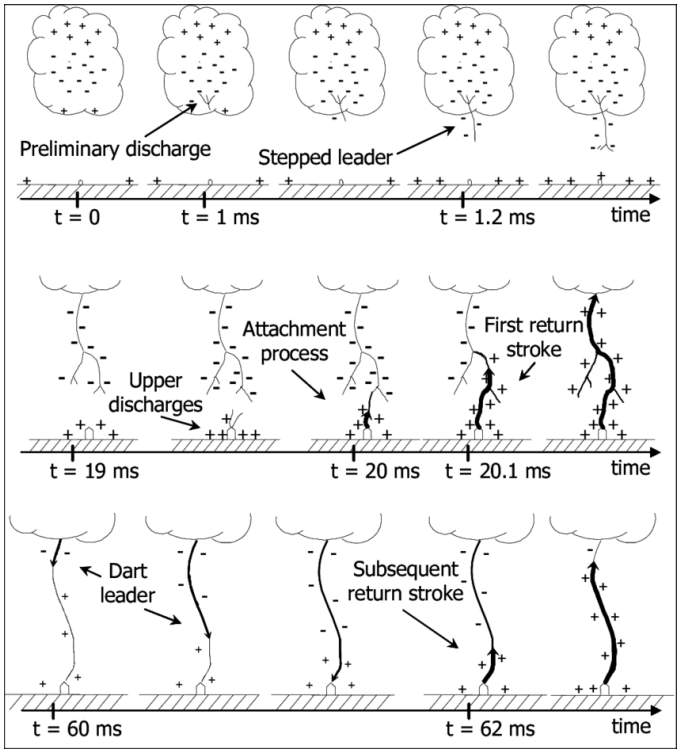

Fig. 4. The process by which a lightning bolt forms [11].

The main channel is built using a modified version of the standard SC algorithm. Firstly, we turned the ability to grow into a trait available only for the most recent node, effectively removing the algorithm's natural ability to produce branches.

Considering a beginning and an end point, the goal is to propagate from a starting point and build the main channel iteratively until the end point is reached. This is another departure from the original SC algorithm. Rather than waiting for all available attractors to be consumed, the algorithm checks for the existence of a specific attractor representing the endpoint. Once this attractor is reached, that is to say, once that specific attractor is consumed, that given iteration of the algorithm ceases.

Attractors are randomly placed inside an envelope volume containing these two points. Arbitrary shapes can be considered for the envelope. For illustrative purposes consider an oriented cylinder where the center of each cap is one such point. The manner in which nodes and attractors interact remains largely unchanged.

Due to the way the algorithm was conceived, any structures made with the original algorithm tends to resemble some form of flora. In order to restrict any behavior that would be considered erratic when considering lightning simulation, such as looping directions and reversing strokes, a method briefly mentioned in Runions et al.'s 2007 article is used [3]. A predetermined weight in order to simulate the elasticity of bark or weight of the tree's boughs. This direction, pointing from the very first node towards an end goal position, served as a way to direct the bolt's propagation. Furthermore, by assigning it a given weight, it was possible to tune the influence of this pre-calculated direction to a fine degree, affecting the overall straightness of the resulting bolt, see in Fig. 5.

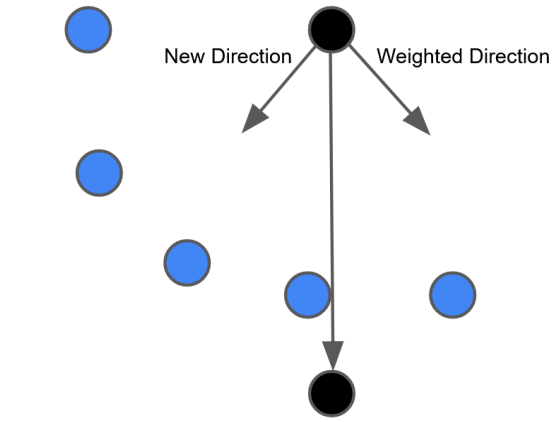

Fig. 5. Weighted growth can be used to influence the manner in which nodes propagate. In this case, half of the growth direction is dictated by normal calculation while the other half is dictated by the weighted direction.

Borrowing from the concept of waypoints explored by Yun et al. [9], a chain of end points can be defined, allowing further path customization. In such a scenario, the algorithm is applied sequentially to each segment, i.e., the algorithm will take the last grown node as the first node in a new iteration and begin propagating towards the next endpoint.

This approach allowed us to introduce the concept of finality to the algorithm, of reaching a goal such as the floor or a lightning arrester, while also providing the flexibility of a waypoint system which lets the user construct a path for their lightning to traverse, an example can be seen in Fig. 6.

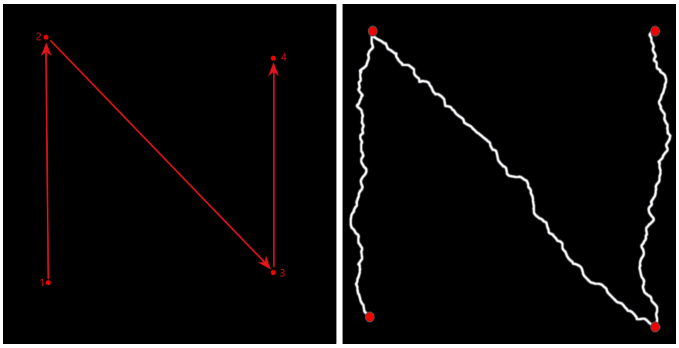

Fig. 6. Waypoints can be used to dictate the path which a lightning bolt takes, drawing out shapes or directing it towards elements.

The algorithm, as described above, can face a situation where there are no attractors available to influence a node, hence prematurely stopping the algorithm. To circumvent this issue a fallback approach was implemented, based in Reed and Wyvill's [5], where the direction of growth is the line linking the last grown node and the end point, rotated by an average of

16° at each step. Employing this method allows the algorithm to propagate until either the new node is within the influence of an attractor or the end attractor is reached.

B. Generating Branches

Through observation of images and videos of lightning, bolts primarily form with three types of channels: primary, secondary and tertiary. The primary branch, resulting from the stepped leader which reaches the ground, was already covered, leaving this section to focus on the secondary and tertiary channels, the branches of the lightning bolt.

In order to finely control the amount of branches produced and how they manifested, we maintained the inability for them to be produced and progressed naturally during the algorithm's execution. The solution found was instead conjured sampling from a probability distribution, employed each time a new node is created. This method allows us to flag nodes as capable or incapable of producing branches, serving as the root node for said structure, while also promoting more granular manipulation of branch density when compared to its SC counterpart. After a root node is identified, the next step in branch creation lies with settling a suitable endpoint.

To allow for an easier tweaking of parameters, spherical coordinates are employed, a decision which also simplified the randomization process tremendously. The azimuthal angle is determined via sampling from a uniform probability distribution function without any restraints. The inclination is decided as a function of the distance from the root node to the main channel's endpoint.

Based on observation of real footage and imagery, it was found that, in general, the angle between the parent and child channel is relatively wide initially, sometimes even growing upwards, becoming narrower the closer the main channel is to the ground. As such, by default, we settled the lower limit for the inclination at 30° while the upper limit may vary up to 135°. Nevertheless, these are just parameters that can be tweaked by the user.

The final parameter, length, is chosen as a random percentage of a user defined length. The usage of probability distribution functions allows for an increase or decrease in disparity between branch lengths.

With an ending selected and a root node already available, the ensuing branch is formed by employing the same method outlined for the main stroke with tertiary branches being the result of a further reduced probability function. These are then also created in the same manner, though their ability to produce further branches is disabled. An example of a result with the three types of branches can be seen in Fig. 7.

Finally, in the event a user defines a secondary branch, parameters such as the root node, the endpoint, the envelope type and width, charge density, the ability to sprout tertiary branches and their density can all be manually defined separately from the natural branches.

Fig. 7. A lightning bolt generated via our method, exhibiting all three types of channel.

C. Rendering

When it came to rendering, we focused solely on the lightning bolt and the effects which surrounded it. This includes the animation of the lightning bolt life cycle.

Our first effort began with the iterative rendering of the structure that would become our bolt. To build it line by line, as if a stepped leader or feeler propagating through the air. The first step to achieve the desired result was found in properly sorting the generated nodes, necessary due to the way branches are built during the algorithm's lifetime. Once they are sorted by hierarchical order, in a manner similar to a data tree, the program can calculate the amount of nodes to render at any given point.

To achieve this, we had to determine how long, at default speed, this section of the rendering should take. With that number, we can then use the percentage of time elapsed to gather an equal percentage of the data structure which contains every point to render in that given frame. This results in a slowly growing bolt, as in Fig. 8 which emulates many of the quirks found in slow motion footage.

The next effect to tackle was the bolt's luminosity. The manner in which it glows once it strikes and illuminates all

Fig. 8. Snapshots of the animation of the iterative rendering of geometry.

around it. To emulate the former, a dilation shader is applied to the main channel, used to emulate the thickness seen once the bolt hits the ground. Afterwards, a separable Gaussian blur with a small kernel size is applied to the entirety of the bolt, including branches. This helps in concealing artifacts formed by the geometry as well as giving a smoother appearance to the bolt itself.

Lastly, two different blurs are applied. The one applied to the branches is given a slightly larger kernel than the previous one. Combining these two allows for a large contrast in luminosity as it nears the branch structure itself, more accurately reflecting the real life phenomenon. The main stroke, however, is given a much larger kernel to reflect its overwhelming presence.

This however, did not fully represent the luminosity seen in the presence of a lightning bolt. To further enhance the rendering, a mipmap of the final blurred texture was created. Fetching one of the lower levels of this map and overlaying the blurred texture on top of it allows for a soft distribution of color that can be easily interpreted as an overall glow.

The last effect we wished to emulate was the strobing of the lightning bolt, caused by subsequent dart leaders and return strokes as well as the effect seen whenever a bolt entirely dissipates, vanishing from sight. To do this, we altered the blending process by defining it as a function of time.

By dividing the strobing effect into separate waxing and waning phases, we can make the blended image increase or decrease in intensity as time elapses within these phases, making use of the same concept explored when iteratively rendering the geometry. Then, by chaining these phases in pairs, the bolt can be made to strobe a number of times, Fig. 9, until it finally fades in its entirety.

V. Results of Implementation

The results achieved from our implementation reflected our efforts to promote user-freedom and parameterize as much of our method as we possibly could. From the listing of waypoints to the definition of individual branches and their associated parameters, the user is allowed the freedom to design their own lightning bolt but also to change its appearance as they see fit. While not all of the possible user parameters were implemented, such as the individual selection of probability functions, most of them and, most importantly, the ones which would most effect the generated bolt are present and as follows:

- Toggle creation of secondary branches.
- Toggle creation of tertiary branches.
- Adjust the probability of secondary branch creation.
- Adjust the probability of tertiary branch creation.
- Adjust the average and maximum length of secondary branches.
- Adjust the average and maximum length of tertiary branches.
- Select envelope type for charge distribution.
- Adjust maximum envelope radius for charge distribution.
- Adjust distance between nodes.
- Adjust the influence of the weighted direction.
- Adjust the color of the lightning bolt.

Most of all, however, we wished to see just how well the results of our method would stack up when directly compared to real lightning bolts we attempted to emulate. To achieve this, we made full use of the tools at our disposal, including the definition of waypoints and branches and the fine-tuning of individual parameters until the resulting simulation was sufficiently close to the real phenomenon. Figures 10-14 show the results we achieved by attempting to emulate different real life lightning. See the acknowledgments section for the source of the real imagery used in this comparison.

From a graphical point of view we believe to have achieved results which are similar in quality to Kim and Lin [2] (see Fig. 2 for a result in [2]), and closely emulate real lightning strikes. Regarding customization, our method offers users a wide range of control, allowing, for instance, for separate configuration for primary and secondary branches.

VI. Conclusion

Throughout this article we detailed a procedural algorithm that can be used for the generation and rendering of lightning.

Fig. 9. Snapshots of the animation of a generated lightning bolt waning.

The geometric procedure is based on the Space Colonization algorithm, and it is fully heuristic. Rendering goes one step further than previous attempts by providing a visualization of the bolt's life cycle.

We believe that this algorithm could already be used in real-time media to great effect, allowing for a game engine, or the users themselves, to fully control and design their lightning as they see fit. Furthermore, the amount of user parameters that were made available makes this method fit even for artistic purposes.

The way the light emanating from a bolt interacts with a 3D scene, including clouds, is a promising avenue for future work, complementing this work, and allowing for a complete solution for lightning simulation in 3D real-time engines.

Further experimentation could be done in an effort to achieve plausible simulation of rare and esoteric events related to lightning such as jets, elves .

A final avenue that might be worth pursuing in the future is an attempt to integrate this manner of procedural generation with the DBM. The DBM could be used to dictate the distribution of the attractors, amongst other settings, causing the creation and definition of a bolt to be based on meteorological conditions, while keeping the SC based approach as defined.

Acknowledgements

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the RD Units Project Scope: UIDB/00319/2020.

Lighting images from real scenarios taken from:

1) https://www.flickr.com/photos/eclipsx/5751977702
2) https://www.flickr.com/photos/veggiefrog/2573076436
3) https://www.flickr.com/photos/hunty66/390350345
4) https://www.flickr.com/photos/snowpeak/3761397491
5) https://www.flickr.com/photos/snowpeak/3761397565

References

[1] N. NSSL, "Lightning basics," 2021. [Online]. Available: https://www.nssl.noaa.gov/education/svrwx101/lightning/

[2] T. Kim and M. C. Lin, "Fast animation of lightning using an adaptive mesh," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 390–402, March 2007.

[3] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm." *NPH*, vol. 7, no. 63-70, p. 6, 2007.

[4] A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, and P. Prusinkiewicz, "Modeling and visualization of leaf venation patterns," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 702–711. [Online]. Available: https://doi.org/10.1145/1186822.1073251

[5] 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

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

[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.*, Oct 2004, pp. 267–275.

[8] J. L. Bryan, S. K. Semwal, C. E. Chow, and T. Chamillard, "Fast simulation of lightning for 3d games," Ph.D. dissertation, University of Colorado at Colorado Springs, 2005.

[9] J. Yun, M. Son, B. Choi, T. Kim, and S.-E. Yoon, "Physically inspired, interactive lightning generation," *Computer Animation and Virtual Worlds*, vol. 28, no. 3-4, p. e1760, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1760

[10] 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*, Oct 2001, pp. 390–399.

[11] J. Bermudez, F. Rachidi, M. Rubinstein, W. Janischewskyj, V. Shostak, D. Pavanello, J. S. Chang, A. Hussein, C. Nucci, and M. Paolone, "Far-field-current relationship based on the tl model for lightning return strokes to elevated strike objects," *IEEE Transactions on Electromagnetic Compatibility*, vol. 47, no. 1, pp. 146–159, Feb 2005.

Fig. 10. Our lightning vs. Real Phenomena (1).

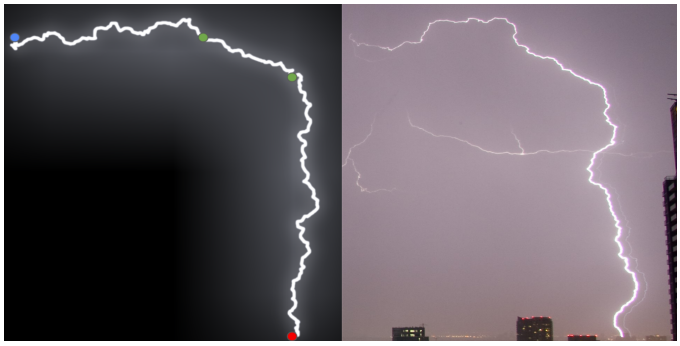

Fig. 11. Our lightning vs. Real Phenomena (2).

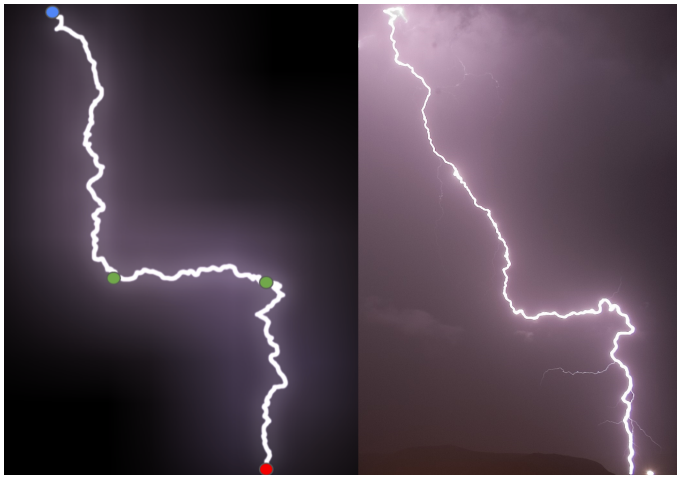

Fig. 12. Our lightning vs. Real Phenomena (3).

Fig. 13. Our lightning vs. Real Phenomena (4).

Fig. 14. Our lightning vs. Real Phenomena (5).