Universidade do Minho
Escola de Engenharia
Departamento de Informática

Nuno Filipe Maranhão dos Reis

Physically Plausible Lightning Simulation

February 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Nuno Filipe Maranhão dos Reis

Physically Plausible Lightning Simulation

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
António José Borba Ramires Fernandes

February 2021

ACKNOWLEDGEMENTS

I would like to formally thank every member of faculty and staff that has played any part, however small, in my academic journey to this point. No matter how minute their contribution, I would not be here were it not for them.

I also thank the scientific community and all those whose research, old and new, has contributed in any way, however small, to the development and formulation of this paper.

I especially extend my gratitude to professor António José Borba Ramires Fernandes for supervising my work and being an invaluable guide throughout the fourth and fifth years of this course.

On a more personal level, I'd like to thank my parents, Duarte Manuel Carvalho Gomes Grácio dos Reis and Sónia Maria Feiteira Maranhão dos Reis, as well as all of my family, especially my brother Ricardo Miguel Maranhão dos Reis, for their unquantifiable emotional and financial support both of which I'm extremely thankful for and forever indebted to.

I also extend my deepest gratitude to Dr. Mariana Pequeno and Dr. Isabel Martins for their help during a terribly difficult thunderstorm in my life. Without their help and my parents' watchful eyes and support, I would have never made it to this point.

To those who have been partners in work throughout the many projects in this course, to those who have been friends, to those who have come and went, to my peers and all those not mentioned by name in this brief note of gratitude, I thank you for having been at my side, however small the period of time.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

The goal of this dissertation is the development of an artistic, user-friendly software for a physically plausible lightning simulator. To this end, this dissertation will employ the usage of L-systems and Space Colonization to allow the user complete control over the path a given bolt of lightning can take. Due to their malleability as algorithms, along with their efficacy in the simulation of branches and other fractal phenomena, these two processes, previously used for the generation of plants, trees, leaves and other flora, prove to be incredible candidates for a truly user-controlled algorithm that allows its users complete control over the properties a given lightning bolt can and will exhibit.

KEYWORDS animation, lightning, physics, electric breakdown, patterns, rendering, real-time, computer graphics, physically plausible, l-systems, space colonization.

RESUMO

O objetivo desta dissertação é o desenvolvimento de um software artistico e fácil de utilizar para a simulação the relámpagos físicamente plausíveis. Deste modo, esta dissertação irá fazer uso de L-systems e algoritmos the colonização de espaço de forma a dar completo controlo ao utilizador no que toca ao desenvolvimento e caminho tomado pelo relámpago. Devido à sua maleabilidade como algoritmos, conjugada com a sua eficiência na simulação de ramos e outros fenómenos fractais, este dois processos, previamente usados na geração de árvores, plantas, folhas e outros tipos de flora, são candidatos tremendos para um algoritmo controlado pelo utilizador que dá a este controlo completo sobre as propriedades que um relámpago pode exibir.

PALAVRAS-CHAVE animação, trovoada, relampagos, física, modelo de quebra dielétrica, padrões, renderização, tempo real, computação gráfica, fisicamente plausível, l-systems, space colonization.

CONTENTS

Contents

LIST OF FIGURES

ACRONYMS

.exr OpenEXR Bitmap. 23, 25

.ppm Portable Pixmap Image. 23, 25

DBM Dieletric Breakdown Model. i, 6, 8, 11, 13

Part I

INTRODUCTORY MATERIAL

1

INTRODUCTION

Lightning is a naturally occurring phenomenon, one that has both terrified and fascinated mankind ever since its inception. Often deified, these seemingly mystical bolts of power were scarcely seen, yet due to their awesome effects, both visual and auditory, became deified by early humanity and classical civilizations. From Mesopotamia to the Roman Empire, lightning was often seen as the domain of the 'chief god' and, as such, became a staple in our culture for terrifying, inhuman power. Today, lightning is the object of study in many fields of academia for varied purposes though, despite all the work done in an attempt to understand it, some of its mechanisms still remain a mystery.

1.1 LIGHTNING

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 our 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 shockwave we have called thunder. These discharges from cloud to ground, labeled downward negative lightning, account for up to ninety percent of all naturally occurring lightning strikes. (NSSL, 2021a)

Lightning has a few outstanding properties to its structure, some of which are important for its simulation. A lightning strike is often composed of several bolts, also called strokes due to their resemblance to a painter's stroke of the brush, in rapid succession. It is composed of three main pieces, the first stroke labeled a stepped leader, the following, less branching strokes called dart leaders and the secondary branches formed by the expanding leaders. (NSSL, 2021b)

Generally speaking, stepped leaders are prone to forming the most branches due to their nature. It is well documented that the stepped leader, due to only being able to influence charges in a fifty meter radius, travels in segmented lines up to fifty meters in length, giving them the typical jagged and tortuous design that has now been associated with bolts. It is also due to this restraint that branches form within lightning strokes as the leader is unable to immediately identify the path of least resistance to ground zero. Instead, it progresses forwards blindly, extending its reach in an attempt to nullify its charge. Due to the properties of surrounding electrical

fields, the number of branches in a given segment of space follows a normal distribution with the most branch presence seen around the mid-point of the stroke's path. (US Department of Commerce, 2018b)

On the other hand, dart leaders follow a much smoother path and tend to barely branch. They are formed by additional negative charges being pulled by the conductive path formed by the stepped leader. As such, rather than path-finding their way to ground zero, they simply follow the already established path of least resistance, the main channel, and attain much higher speeds.

Most notably, leaders are imperceptible to the human eye. Instead, all we see are the following return strokes which discharge all their accumulated negative charge in the blink of one's eye, causing a bright flash. The existence of multiple return strokes and dart leaders explain the existence of flickering within particularly big bolts. (US Department of Commerce, 2018a)

1.2 MOTIVATIONS AND GOAL

One of the core tenants of computer graphics and simulation algorithms has been the translation of real life events, especially those too dangerous to properly study in a controlled environment, into the digital world. Constraints in physical hardware, paired with lack of understanding in both computer graphics and natural phenomena, have seen engineers struggle with various issues throughout the new millennium. As such, various techniques were developed to simulate qualitative accuracy, foregoing the need for physical laws.

From the simulation of light to the flowing of water, experts have seen themselves encounter wall after wall when translating physical phenomena into a digital setting. While the world seems to operate with infinite possibilities, computers are discrete machines and, as such, the concept of infinity becomes an impossibility even within a mathematical setting. In layman's terms, it is impossible to arrive at a finite result if we traverse infinite paths. However, the exponential increase in computational power has enabled the creation of innovative techniques which aim to raise the fidelity of our simulations, bar the possibility of convergence. While in the past qualitative accuracy may have been enough, the need for physically accurate or, at the very least, plausible simulations has placed a redoubled importance in the development of techniques that satisfy the need for accuracy. It is necessary, then to develop simulations accurate to the point where we can use these processes to better understand natural phenomena and its interactions with objects, without the need to put people at risk. However, it is important to consider how feasible these algorithms are for a real-time implementation.

The goal of this dissertation is the development of a simulation engine that will render and simulate lightning in a realistic manner. From a physical standpoint, lightning is nothing more than a stream of electrons between two areas exhibiting drastically different electric potential. However, the manner in which this phenomena forms and the way it determines its path is mysterious, to say the least, and utterly chaotic. From a mathematical lens, lightning is an incredibly complex phenomena and, while its origins and consequences appear relatively simple, it proves a challenge to computer scientists and engineers alike even to this day.

From the various forms a single dart may take, to the fractal patterns seen within its branches, the simple rendering of this deified phenomena has posed a challenge to members of the scientific community since 1994

(Reed and Wyvill, 1994). As such, it poses an interesting avenue for innovation, one that we wish to tackle by throwing our metaphorical hat into the ring.

1.3 STRUCTURE OF THE DISSERTATION

Following this brief introduction into the subject matter, this thesis presents the overall structure of this dissertation. Immediately following this chapter will be an analysis of historically significant work and state-of-the-art implementations that fall under the scope of this project. Furthermore, in chapter 3, a work flow for the project's future is delineated along with a brief description of what this dissertation sets out to accomplish. In chapter 4, the work developed thus far, such as tests and analysis of code and data models used by some of the state-of-the-art implementations highlighted below, is presented.

2

STATE OF THE ART

Throughout the years, Lightning Simulation has been an area of Computer Graphics with wavering interest. While there are many applications out there which study the behavior of lightning storms and their ensuing phenomena, most do so from an empirical standpoint, choosing to focus their lens on the damage such a destructive force may cause in human infrastructure. Of particular importance in this field, is the heuristic simulation of lightning for aerospatial engineers whose labour is at particular risk of being struck by these discharges. However, lightning itself can be much more than a destructive influence.

In 1994, Reed and Wyvill published their paper Reed and Wyvill (1994) and pioneered the simulation of lightning for its beauty and aesthetic, awesome impact rather than its destructive force. The presence of proper generated lighting, they would go on to say in Reed and Wyvill (1994), can dramatically change a scene adding layers to rather mundane storms and deeply immersing the user into the scenery and its unfolding storyboard. However, as niche and complex as it is, lightning simulation has become a field of computer graphics that, when compared to others, has gone woefully underappreciated.

2.1 VISUAL SIMULATION OF LIGHTNING, REED AND WYVILL, 1994

While there have been articles and other investigative works published in the webspace by computer scientists, none has been quite as influential as the first. Published in 1994, Reed and Wyvill (1994), this piece of computer science history has been cited many times. K. Todd Reed and Brian Wyvill, first published their article on January 7, 1994 and, since then, nearly every scientific work on the topic has referred to their original methodology in one way or another, even if only as an inspiration or to lay out historical groundwork.

In their original method, the authors described a simple particle system which simulates the stepped leader progression towards ground zero, that is, the physical location where the lightning bolt strikes. While not based on any physical conjecture or formulae, the authors took into account the qualitative research brought on by the thorough and tiresome examination of dozens of images, ensuring that the final product would replicate the sight of a bolt of lighting as accurately as possible.

Their system operates around the usage of a randomly generated number utilizing a seeded algorithm. By combining the output of this algorithm with rotations, the bolt is rendered stroke by stroke, with each segment depending on the ending of the last which, in the author's words, ensures congruity between all traces of the stroke and guarantees a quasi-linear shape. This rotation is done based on the observations and research

done not only by the authors but physicists which, at the time, could only conduct research through video and photographs. Through their combined effort, a mean of 16 degrees between the angles of each segment was discovered.

Notably, the method described uses a linear distribution to determine the length of each segment and, while they inaccurately describe the length of a segment as varying between one meter to one kilometer (US Department of Commerce, 2018b), they do correctly surmise that smaller segments produce the most visually realistic results.

Branching in this method is controlled by a simple probability function following a set of outlined properties. The authors mention that it makes use of the distance traveled to calculate the probability for branching occurrences, yet warn of erratic behavior and difficulty controlling the ensuing branches in a consistent fashion.

For rendering, a model is formed using a skeleton composed of elementary particles such as points, lines, polygons, circles and even splines tied together through a hierarchical link. This skeleton is then surrounded by a scalar field with decreasing values as distance to the skeleton increases. Alongside this, a ray-tracing model was built making use of ellipsoids to influence the scalar field. The nearest distance between each traced ray to each primitive ellipsoid is calculated and the shortest distance is passed to an equation, building a final shape following the zero contour, as shown in figure 1. Finally, passing a blending function in conjunction with the field, before the final drawing of the contour, allows the authors to shape the lightning accordingly and dictate how separate surfaces should blend together.

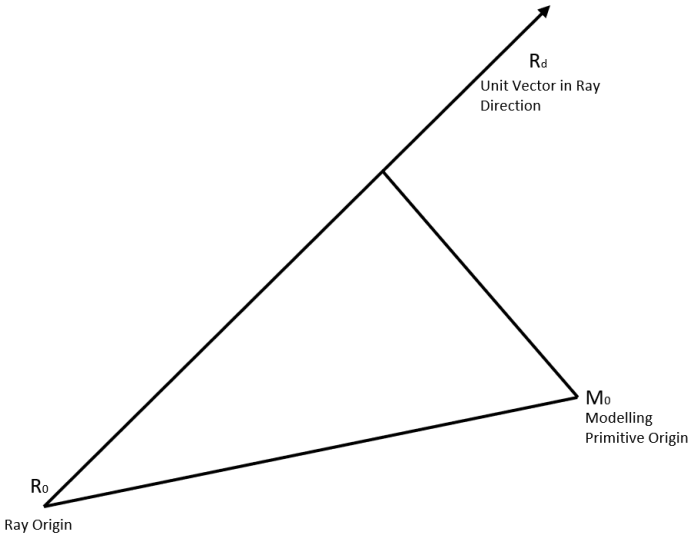


Figure 1: Finding the nearest distance from a ray to an implicit surface

The authors highlight the need for such a solution due to the limitations imposed by the technology available. The use of line-segments prevented the authors from treating strokes as geometric objects, rendering the available shading methods inappropriate due to the lack of a definable surface.

Since then, however, many others have iterated upon this work. Notably, Glassner (2000) performed a second pass upon the branches to add a degree of "tortuosity" to the represented bolts while Sosorbaram et al.

(2001) made use of a model which has then become a staple in the simulation of lightning surges: The Dieletric Breakdown Model, or DBM for short. (Niemeyer et al., 1984)

A dielectric breakdown is a formation of electrically conducting regions in an insulating material when exposed to a strong electric field. Most notably, the branching, fractal patterns that form as a result of this breakdown exhibit properties that allow modelling via the use of a Laplacian growth algorithm.

2.2 EFFICIENT RENDERING OF LIGHTNING TAKING INTO ACCOUNT SCATTERING, DOBASHI ET AL., 2001

This article proposes an efficient method for creating realistic images of scenes that include lightning. For that purpose, the authors designed an algorithm that can create photo-realistic images by taking into account atmospheric scattering when clouds and other phenomena are illuminated by lightning, something that was notably lacking in Reed and Wyvill (1994). Moreover, they fully utilized the graphics processing capabilities available at the time to accelerate image generation. Much like the previous article, it is important to note that the logic behind the proposed method and its implementation are of both historical significance and interest when creating 3D applications that include the realistic rendering of lightning, serving as the basis upon which modern methods have improved upon.

Notably, in Reed and Wyvill (1994), lightning and its glow were rendered by employing a heuristic approach. However, as previously stated, clouds were not taken into account and the modelled glow became entirely at odds with the physical representation of lightning as one of the primary mechanisms for its existence is its interaction with the surrounding atmosphere. Moreover, while there were a large number of research reports on the generation of images taking into account atmospheric scattering, Dobashi et al. (2001) stipulate that these algorithms were only fit for natural light sources such as sunlight, skylight and artificial spotlights, making them unsuitable for the rendering of lightning strokes which are characterized by large, irregular forms that lack a proper surface. The authors highlight the need for understanding that these methods often took dozens of minutes to create an image, prompting them to develop a technique that would render in a fraction of the time previously required.

In their article, Dobashi et al. (2001) use Reed and Wyvill (1994)'s technique for the generation of lightning strokes, representing them as a set of line segments. The generated bolt is then displayed by a drawing function with its color dictated entirely by the user. In tandem, the intensity of the clouds illuminated by lightning, along with their corresponding scattering, are computed by generating point light sources along the line segments, as seen in figure 2. The authors highlight an interesting problem: generating too many light sources proves incredibly costly on the algorithm while the usage of too little renders them moot for their intended purpose. Due to the fact both clouds and atmospheric scattering are rendered by summing the contributions from each light source, a balance needs to be struck between the quality of the render and the speed at which it is rendered.

To get around this fact, or at least to somewhat alleviate the problem's nature, the density distribution of clouds is expressed as a collective rather than a singular entity. These distributions are defined on three-dimensional voxels that the authors dubbed 'metaballs'. These metaballs are placed at the center of each voxel and given

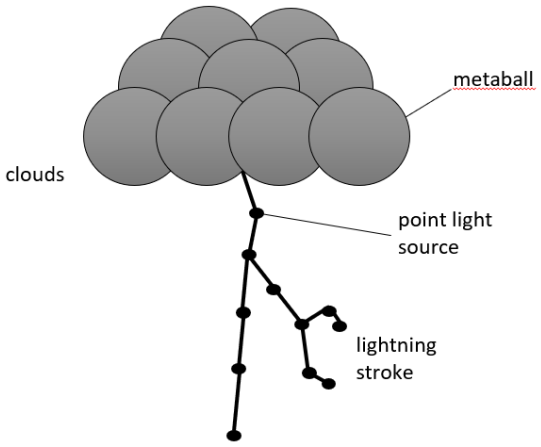


Figure 2: Basic idea of Dobashi's method

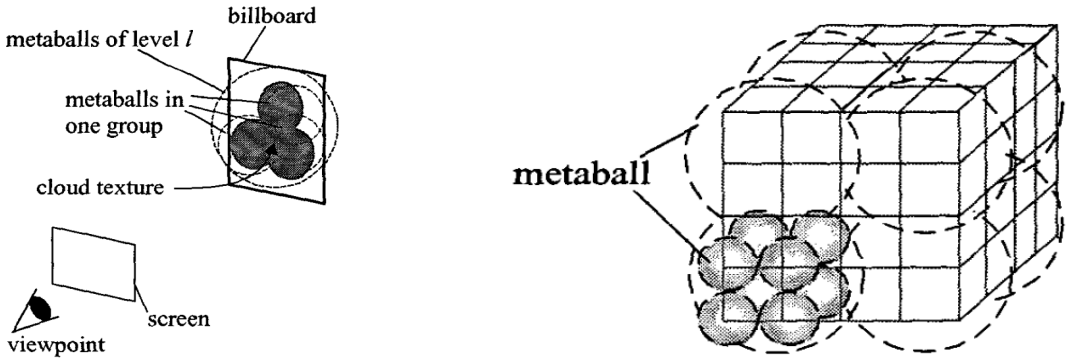
two values: the center density, calculated using the average density of each cloud present at that given voxel and effective radius as clouds are not perfectly spherical.

To achieve illumination, a billboard method is used to lower computational requirements and accelerate rendering times. The authors note, however, that despite their best efforts, this approach remained computationally expensive with a cost that is proportional to both the number of metaballs and their associated light sources. Furthermore, their experimentation only yielded satisfactory results when using thousands of metaballs and an equally vast amount of point sources. As such, the introduction of level of detail was necessary.

The intensity of light from a given point source decreases in proportion to the square of its distance to a metaball. Therefore, a straightforward approach is to simply ignore the metaballs too far from the point source as the intensity of light reaching these units is negligible. However, contributions from many point sources can become significant even if one alone is utterly insignificant. With this in mind, this proposed approach was labeled as entirely unfitting by the authors.

Instead, metaballs are grouped into a hierarchical structure. Larger metaballs are used to represent multiple smaller ones with equivalent intensities, something that becomes more prominent the farther a given set of metaballs is from a light source. To achieve this, clouds are firstly represented by a previously mentioned octree with its density distribution defined by voxels. Eight neighbouring voxels are then grouped and squashed, replaced by a larger voxel and, by repeating this process, the density distribution becomes easily represented within the octree itself as exemplified in figure 3.

By itself, this feature was able to reduce computational time and load by a factor of eight. Though the authors couple this approach with a few mathematical and physical shortcuts and simplifications as to further reduce the amount of computational weight the calculations for atmospheric light exert upon the algorithm. As the realism of the clouds themselves was not at stake, the authors proposed an efficient method to render them employing the use of imposters. An imposter replaces an object with a semi-transparent polygon texture-mapped image of said object, therefore severely cutting rendering times.



(a) Imposters.

(b) Hierarchical metaball representation.

Figure 3: Dobashi et al. (2001)'s method for cloud rendering.

As a final note, Dobashi et al. (2001) make it clear that fast image generation will be indispensable in the future, especially for the purpose of flight simulation and instruction under bad weather conditions.

2.3 PHYSICALLY BASED ANIMATION AND RENDERING OF LIGHTNING, KIM AND LIN, 2004

In their paper, Kim and Lin (2004) propose a simulation based on convolution techniques. Their goal was to rival Monte Carlo ray-tracing with a large emphasis in user parameters for the manipulation of the simulation and the usage of a physically inspired approach based on the DBM. They also provided a platform for the simulation of sustained electrical streams which are produced by solving a simplified version of the Helmholtz Equation.

To fulfill their goal, the authors proposed several modifications to existing formulae, most notably, modifying the DBM to simulate dart leaders and sustained arcs while making use of the Laplacian equation for electrical potential.

To initiate the simulation, a preconditioner is set. A vertical line of negative charges, negligible in size, is set perpendicular to the lightning's starting point. From thereon, after solving the ensuing system, one of the cells adjacent to a site of 0 charge is set as the site for the next dielectric breakdown and becomes part of the boundary condition.

Following the hypothesis in which dart leaders follow the same path as the stepped leader due to the residual charge left along the old channel, Kim and Lin (2004) designed a new method to introduce the aforementioned property. To this end, the authors noted that the Laplace equation could be viewed as a special case of the Helmholtz equation where the charge density is equal to zero and, due to lightning bolts presenting a linear velocity approaching the speed of light, both the angular and relativistic components are ignored. As such, by giving the subsequent leaders a positive charge density, dart leaders can be successfully simulated. Along with all these simplifications, the authors also advise the treatment of air as a homogeneous media as to avoid the usage of more rigorous and computationally expensive equations.

For rendering, the authors made use of an Atmospheric Point Spread Function, a method pioneered by Narasimhan and Nayar (2003), where an analytical model is used to reduce the rendering of certain media

to a 2D convolution. In layman's terms, the Atmospheric Point Spread Function is a three dimensional function that describes how much light reaches a point in space around a given light source. By determining how a single point light spreads out on the image plane, this function can then be used as a kernel to render any light source of arbitrary shape.

By using this to simulate the perceived thickness of lightning plasma, a series of thin line segments are modeled and the kernel is then applied to a 2D rendering of them in order to simulate glow. If the brightness is correctly set, then the final luminance values should exceed the range of the display device and create the expected thickness.

Finally, for the rendering portion, the authors devise three main stages: drawing a graph from the simulation, as seen in figure 4, assigning luminance values to each graph edge and, finally, rendering said edges as line segments as to apply the kernel, compositing the resulting image into a raytraced rendering.

With all these improvements, the authors boast a technique with competitive results with a render time of seconds rather than hours.

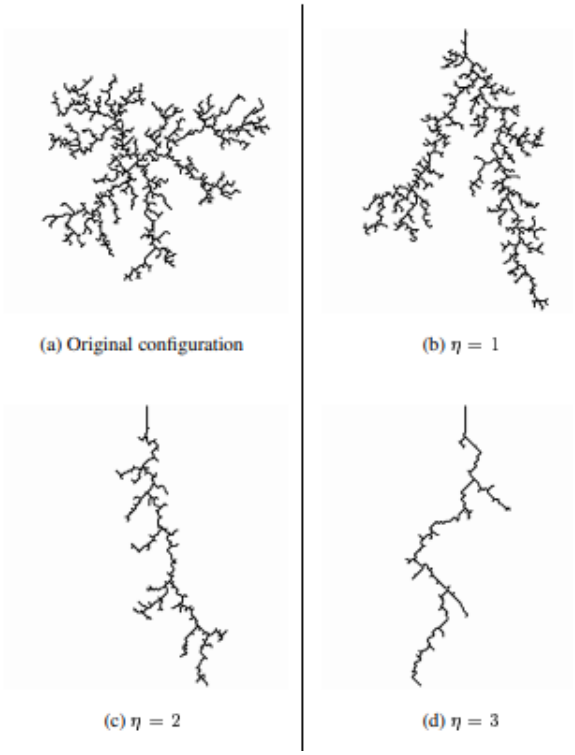


Figure 4: Simulation results from different charge configurations (Kim and Lin, 2004)

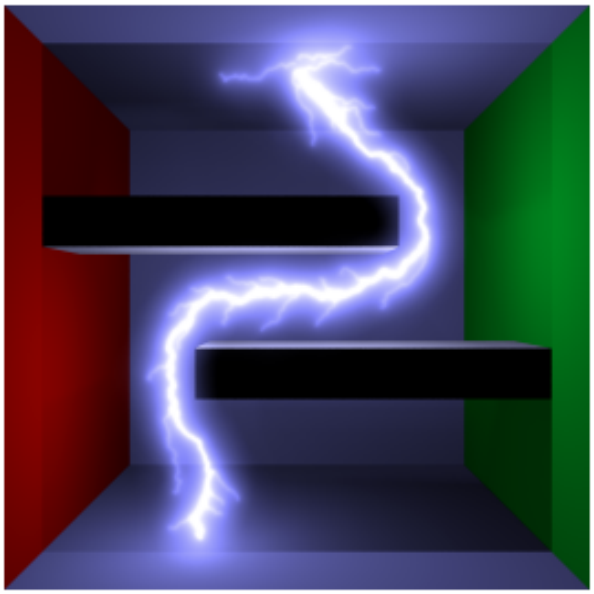


Figure 5: Results (Kim and Lin, 2004)

2.4 FAST SIMULATION OF LIGHTNING FOR 3D GAMES, BRYAN ET AL., 2005

Bryan et al. (2005) draws inspiration from Kim and Lin (2004) while coming to its own through a mixture of different techniques and an uniquely distinct goal. While following the same concept of main and secondary strokes, intrinsic to the previous rendering methods and to the real-life phenomenon, the methodology presented here makes use of cellular automata to generate volumetric data in an effort to produce a realistic stroke that could still be both generated and displayed in real-time.

While cellular automata were originally proposed by Von Neumann as a formal model for the representation of self-reproducing organisms, they are nothing more than dynamic systems that occupy a uniform, regular lattice and work in discrete steps across time. When one or two dimensional, they are defined by adjacent cells while, if three dimensional, they become defined by voxels or boxes.

The authors consider them suitable for the real-time simulation of iterative phenomena such as lightning, in large part due to their general behavior, their associated characteristics and reliance on neighbouring values, coupled with a simple yet powerful design.

By considering the atmosphere as a collection of voxels, the authors highlight, and allowing leaders to propagate by inspecting a local neighbourhood, the problem can be trivially translated into computer graphics while avoiding many of the downsides that come with local controls.

In order to allow the user to be able to control the lightning's destination, the model builds the bolt of lightning from the ground up. The generative process begins when a given area is selected, causing that cell to be added to a linked list and whirring the algorithm's metaphorical gears into motion. Every voxel, or cell, is attributed a randomly generated value representing atmospheric conditions and, by using this value, the algorithm is able

to calculate the electrical field for the neighbouring cells using Maxwell's Equations. The discrete progress of lightning is then modelled by the creation of a random number representing the length of a given segment. Only after the segment is fully generated does the next calculation take place.

By repeating this process, the main branch is concluded when the bolt reaches either an arbitrarily set altitude, the skybox, or a cloud, at which point a linked list is made with every point in the main stroke. Each member of this list is composed by its coordinates, randomly generated atmospheric value and a header for another list which can then be used for secondary leaders. To generate these branches, the author resorts to a uniform distribution.

Referring back to these secondary leaders, their branches are made by recursively running the probability function and are given arbitrary and random maximum lengths in order to force a mid-air termination, calculating the distance between its origin point and the ground as to never reach it. In general, this simple but powerful simulation combines the best of qualitative approaches and physically inspired simulations into a neat and friendly package. It also boasts the added benefit of directed impacts, making its implementation perfect for real-time renders where precise control over the strike's path is needed though offering less graphical fidelity and foregoing complex effects.

Figure 6: Iterative Simulation using Waypoints (Bryan et al., 2005)

2.5 FAST ANIMATION OF LIGHTNING USING AN ADAPTIVE MESH, KIM AND LIN, 2007

Kim and Lin (2007), is the authors' second take on the development of a fast renderer of lightning. As stated before, DBM is an expensive model to compute so the authors propose modifications and the usage of an adaptive mesh, exemplified in figure 7, to increase its computational speed. Notably, this approach is built upon a symmetric discretization of the Poisson Growth worked upon in their previous article.

While their previous algorithm produced visually acceptable results, the authors considered the possible running time and memory footprint prohibitive when protected over to 3D models. They cite that, in practice, they've found that in order to obtain results where artifacts are not distracting, a grid of 2563 cells must be used. Furthermore, in order to run a preconditioned conjugate gradient, seven arrays of this size are necessary and, even

if assuming single precision, the method already racked half a gigabyte of memory data. Increasing the resolution to 5123 bloated the footprint exponentially to 3.7GB. Even with an efficient implementation, the matrix could become so large that simulations began taking days.

The authors note that, until 2004, solving a Poisson problem over an adaptive grid had the caveat of producing a non-symmetric matrix, rendering its use an impossibility for this specific purpose. However, Losasso et al. (2004) overcame this limitation, allowing the use of a more robust gradient. Armed with this innovation, the authors coupled the usage of an adaptive grid with a balanced octree to greatly simplify the implementation of their solver.

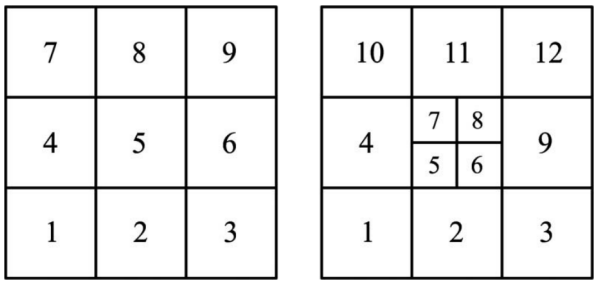


Figure 7: Differences between a regular and adaptive mesh, respectively.

This balanced octree is an octree where, given any leaf node, the size of each neighbouring node differs by, at most, a factor of two. The authors chose to enforce this property because it constrains the maximum number of neighbours of a leaf node to 24 (4 neighbors at each of the six cube faces). This is necessary to bound the complexity of performing incomplete factorization, as a leaf node and its neighbours define a row in the Poisson matrix. As a final step to ensure this property, a full scan of the octree is made and any outliers are remedied appropriately.

The authors make use of a modified incomplete Cholesky algorithm for the factorization process, reporting that, in modest quadtree simulations, the number of iterations required for convergence is identical to those required by the original Cholesky preconditioner and that, by using this discretization, they were able to report a speed up of over two orders of magnitude. This equated to a 22 times increase for a 1283 cell adaptive simulation and a 150 times increase when using a grid composed of 2563 cells.

As a last note, the authors advise caution to a noticeable rendering artifact appearing due to insufficient resolution in the simulations. Ideally, in their words, a grid cell within the simulation should correspond to a pixel on screen. However, such a resolution is impractical and a comparative result can be achieved by introducing jittering in the simulation results.

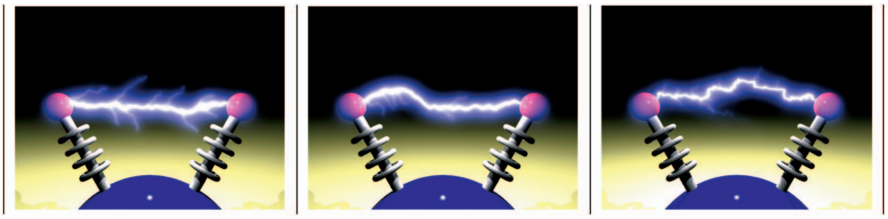


Figure 8: Results (Kim and Lin, 2007)

2.6 PHYSICALLY INSPIRED, INTERACTIVE LIGHTNING GENERATION, YUN ET AL, 2017

Yun et al. (2017) is what can be most defined as state-of-the-art and counter*poses itself against the models produced by Kim and Lin, explored earlier within this section(Kim and Lin (2007) & Kim and Lin (2004)) with the former also contributing to the methodology proposed herein. In contrast to said models, this one exhibits a reported increase in speed of about 20 times, taking an astonishing 38ms on a 128x128 grid while using only a singular core. When compared more fairly to the aforementioned DBM method that makes use of conjugate gradients, the reported speed ups are of over two orders of magnitude. When compared to a DBM approximation with a summation model of potentials, it showed twice as fast speeds while exhibiting better shape controls and being closer to the physical interpretation of a dielectric breakdown thus giving it wider applicability. An example of the DBM method is given in figure 9

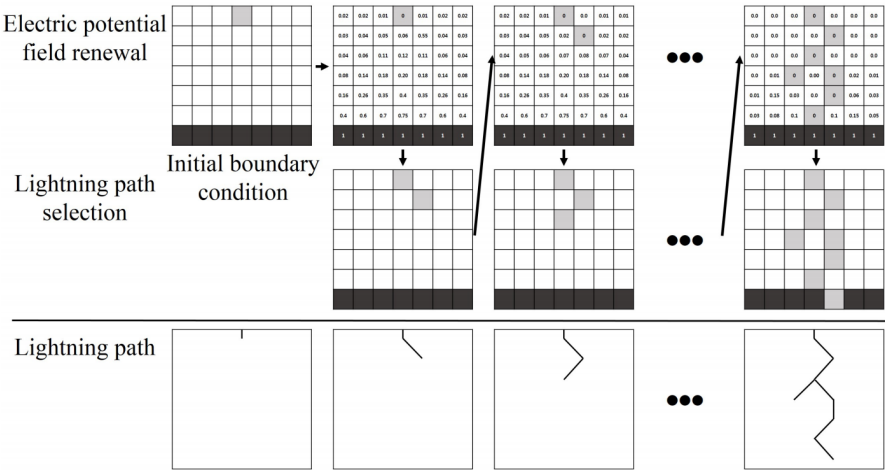


Figure 9: Generating a lightning shape using DBM (Yun et al., 2017)

When searching for a method to better the current state of lightning simulation, Yun et al. (2017) looked for sources of inspiration in many works. Most of them have been mentioned in this section of the thesis, though the authors delineate a few more of relative importance. First, an emphasis is given to the fact current games and real-time applications make use of pre-rendered images or approaches based entirely on randomized trees, both of which prove to be problematic. Lightning is a fairly expensive phenomenon to render but one that draws

the eye of the user and, as such, the usage of repetitive patterns and low quality sprites can severely hamper immersion.

It is noted that a previous work using the GPU to solve the Laplace equation in real time had already been done by Matsuyama et al. (2006) however, the cost was simply too high to be implemented outside a dedicated application. Similarly, while Nvidia provides a realm-time DirectX 10 example that uses a geometry shader, their approach is not physically based and suffers from the same downfalls as Reed and Wyvill (1994). In contrast to the aforementioned methods, Yun et al. (2017) propose a physically-inspired method that generates lightning by better approximating the value distributions within an electric potential field.

Many of the same difficulties were encountered, however, as rendering the atmospheric scattering of lightning has always posed a problem to researchers within the Computer Graphics sphere. While Reed and Wyvill (1994) extended ray tracing to render the lightning stroke and glow, Sosorbaram et al. (2001) proposed a volume rendering technique using a 3D texture. Furthermore, Dobashi et al. (2001) pioneered the use of metaballs and pre-computed lookup tables to store the integrated intensity of scattered light. Kim's previous work(Kim and Lin, 2004) made use of an atmospheric point spread function which described light scattering as a convolution kernel used to render the final stroke.

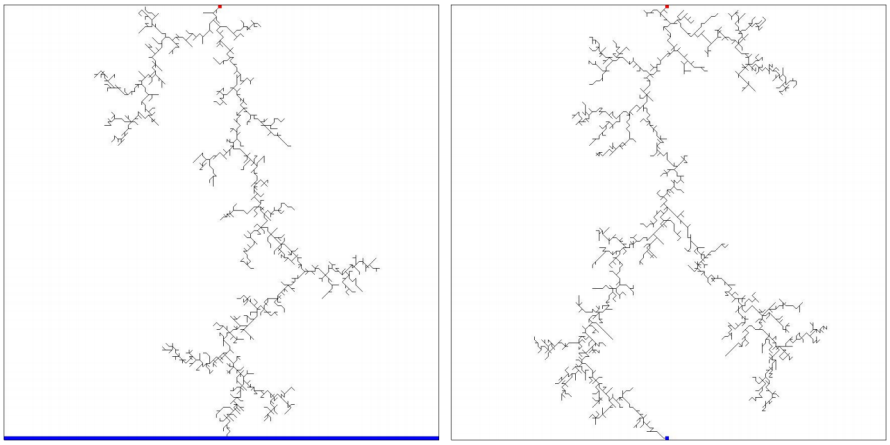


Figure 10: Laplacian Growth Model (Yun et al., 2017).

In this work, however, Yun et al. (2017) propose their own method based upon the basic properties of an electrical potential field. Using their gradient tendencies alone, the authors propose an approximation of the relation between electric potential and the distance between a given point and an electric charge. As seen in figure 10 and 11, where the leftmost simulation represents a typical cloud-to-ground bolt while the rightmost one represents lightning cascading from the red square to the blue one, the approximation is extremely faithful to the ones obtained via the Laplacian Growth model and, in some ways, qualitatively better.

Computing the electric potential between candidate cells and other charged cells and passing them through a normalizing equation, allows the authors to divide charges into positive, negative and boundary charges. This division is then used to calculate the electrical potential for each type separately which, when computed with certain values, actively alters the shape and depth of the branching patterns formed by the simulation. Notably,

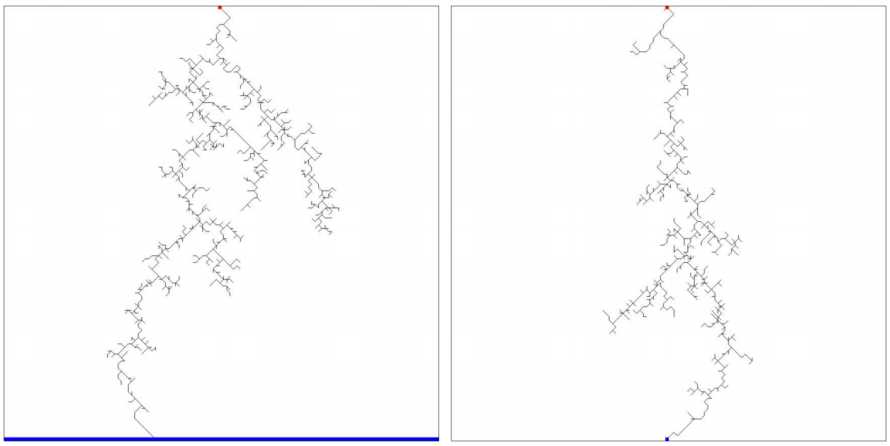


Figure 11: Yun's Growth Model (Yun et al., 2017)

Yun et al. (2017) provides an in-depth graphical representation of how changing the values in a given order can affect the shape of a lightning bolt.

Moving onto the rendering itself, there are a few important constraints to keep in mind. First, there can be objects or obstacles that should not be hit by the bolt, such as a wall in the game scene. For such obstacles, negative charges are assigned, allowing them to be treated as part of the boundary condition.

Moreover, in a complex scene, their method encountered local minima, similar to other potential field methods. Due to this and the fact their algorithm computed potentials based on distance, when a scene exhibits obstacles that block the target position from the starting position, the lightning branches could spread out excessively. To handle this problem, Yun et al. (2017) made use of waypoints that are generated by path planning methods such as an A* algorithm, as seen in figure 12

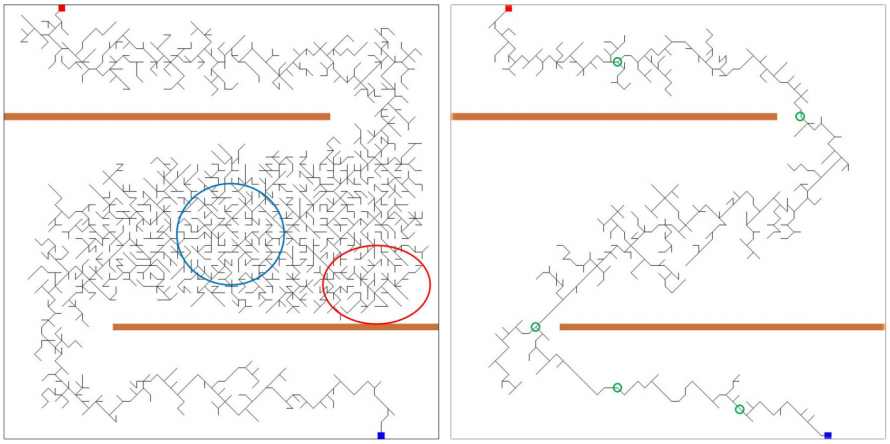


Figure 12: Local Minima vs. Waypoint Solution (Yun et al., 2017)

As for the physical characteristics, thickness and brightness are taken into account when rendering a stream which is further decomposed into main and secondary channels, the main channel being any path that connects

the starting and target points while secondary channels are sub-branches originating from the main channel, connecting to nowhere.

Lightning lines are rendered using billboard techniques such as rectangles while branches are represented by cylinders and spheres for three-dimensional scenes. Furthermore, the glow itself is created using a fast two-pass Gaussian Blur filter applied horizontally to the framebuffer texture and, then, used vertically upon the resulting image. Jittering is also extensively used to reduce grid regularities that appear due to the lack of fine grid resolution.

Taking all of this into consideration Yun et al. (2017), while also being the most modern, represents the most physically accurate representation of lightning able to be applied into real-time scenes, making it a stark candidate for further studying and reference material going forwards.

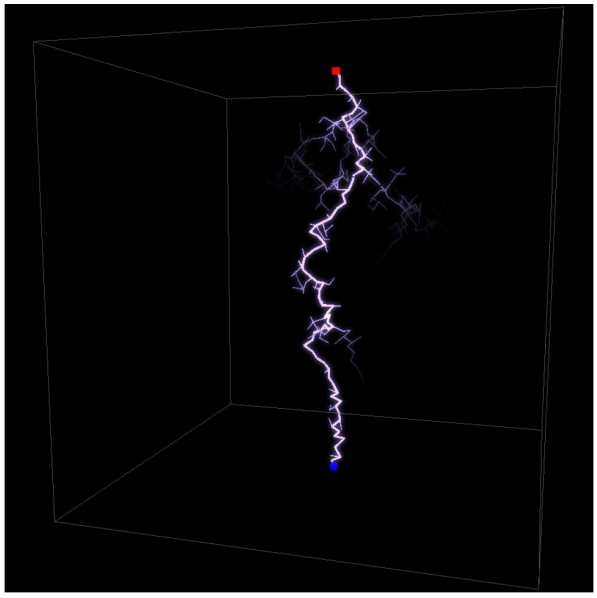


Figure 13: Results (Yun et al., 2017)

2.7 L-SYSTEMS

L-Systems are a grammar-based system developed by a Hungarian botanist in 1968 to model the growth patterns of plants. Originally, they were conceived as a mathematical theory of plant development, however, nowadays, they are often used to generate self-similar fractal patterns.

Fractals are a mathematical object whose structure remains the same even after a change of scale. Similarly, a central concept of L-systems is that of rewriting, a technique for defining complex objects by successively replacing parts of a simple, initial set using a number of rewriting rules. Also central to both is the concept of recursion, therefore making L-systems an efficient mechanism to achieve complex fractal patterns(Harzallah, 2020).

L-Systems involve three main components which can be seen in action in figure 14:

- An Alphabet: a set of valid characters that can be included in a “sentence”.
- An Axiom: The sentence that describes the initial state of the system.
- Rules: which are applied to the axiom and then recursively re-applied, generating new sentences consecutively. An L-system rule indicates two sentences, a predecessor and successor.

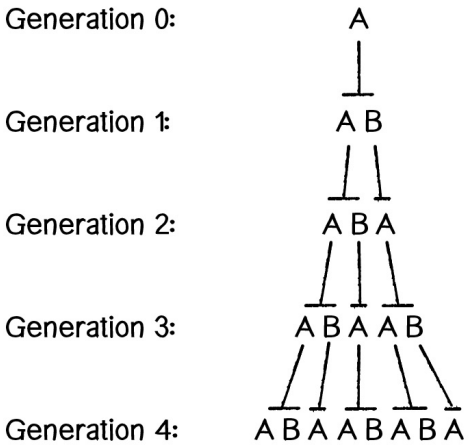


Figure 14: Example of an L-System grammar.

By laying this basic groundwork on the uses and inner workings of L-systems, one can easily see how applicable this mathematical algorithm could be to the generation of lightning bolts. As previously stated, these bolts are often fractal in design, especially when it comes to the branching they exhibit and, by using L-systems, one could easily model realistic branching with minimal mathematical overhead.

2.8 SPACE COLONIZATION

The space colonization algorithm treats competition for space as the key factor in determining the branching structure of a given object, trees, in this case. The algorithm relies on the usage of attraction points and one or several tree nodes. The tree is generated iteratively and, in each iteration, a given attraction point may influence the node closest to it.

This influence occurs if the distance between the point and the closest node is less than a given radius of influence. It is also worth noting that there may be several attractors at play at once. Once new nodes have been added, a check is performed to test which, if any of the attraction points should be removed due to the proximity to the tree branches that have grown towards these points. The criteria for this removal is the proximity of, at least, one tree node. Should a given node cross an established threshold, the attractor is removed as its purpose has been fulfilled as seen in figure 15.

Worth noting is the reliance of this algorithm in repetitively testing the set of attraction points for proximity to a given tree node. The author of Runions et al. (2007) article performs these calculations by constructing a three-dimensional Voronoi diagram and testing the resulting domains for attraction points.

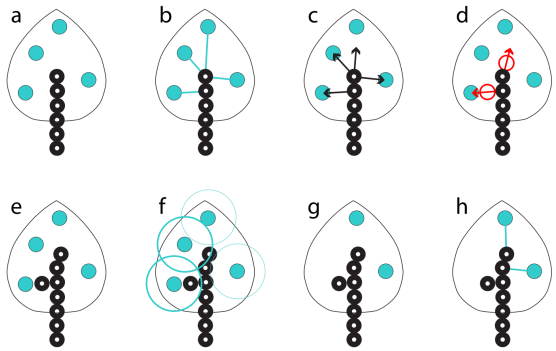


Figure 15: Example of a Space Colonization algorithm.

By extending this model to the usage of electrical potential fields as a density function to the number of positive charges within a given grid-space, one could feasibly use this algorithm as a tool to construct physically plausible images of lightning bolts.

2.9 CONCLUSION

The advances made in the span of twenty years when it comes to the rendering of lightning have been immense and the approaches through which scientists and members of the academic community have tackled the issue have been many and varied. Some sought only the representation of lightning with enough accuracy to be believable, others a more qualitative approach, wanting their own version to visually simulate the appearance of a bolt. Most recently, however, the domination of physically-based implementations have been on the rise, courtesy of developments in the Computer Graphics area and the exponential growth of hardware capabilities. Be that as it may, most implementations fall short of real-time efficacy and those who do not, take too many liberties in the simulation itself. However, Yun et al. (2017)'s model is an incredible showcase of what one can do given the right tools, proving to be a physically inspired approach able to run in real-time without any pre-generation.

3

FUTURE PROSPECTS

This dissertation will, from hereon, focus on the development of a real-time rendering solution for a physically plausible rendering and simulation of lightning bolts that allow for user input and control over their path and exhibited trait such as the number of branches and fractal depth. To this end, the dissertation will make use of L-systems and space colonization algorithms to develop two different software, compare them to one another and, possibly, join them together to make the best out of each of their strengths, compensating for the other's weaknesses.

To this end, the workload for this dissertation will be further divided as follows:

- During the months of February and March: Implementation of 2D Versions of an L-system and Space Colonization algorithm for the creation of a lightning bolt with emphasis on user-controls and malleability.
- In the following moths of April and May: Comparisons between the two developed algorithms, along with the development of a third piece of software that uses both L-systems and space colonization in tandem to better reproduce the effects of a lightning bolt, be it cloud-to-ground or simply a free-form path.
- June and July present the final steps for this dissertation and will, in large part, reflect the work done to this point. They will be promptly used for testing and polishing of our software demonstration.
- Throughout these six months, work will be done on writing the dissertation as well. With each milestone reached, notations will be kept and a part of the thesis proper will be crafted. As such, the final stretch will also serve as the opportunity to, via retrospection, improve previous segments of the dissertation and provide the final touches.

With this skeleton as a basis, the thesis will aim to distinguish itself from work done to this point by using innovative techniques and taking a more artistic approach to a physical phenomenon, allowing for a wider variety of applications and for extensive user input while remaining, to the best of its ability, physically plausible from a qualitative point of view. The aim of this dissertation is for one to be able to look at the resulting bolt and identify it as lightning, rather than for it to extensively portray all the physical properties that lightning encompasses.

BIBLIOGRAPHY

Jeremy L Bryan, Sudhanshu K Semwal, CH Edward Chow, and Tim Chamillard. *Fast Simulation of Lightning for 3D Games*. PhD thesis, University of Colorado at Colorado Springs, 2005.

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*, pages 390–399, 2001. doi: 10.1109/PCCGA.2001.962896.

Andrew Glassner. The digital ceraunoscope: Synthetic thunder and lightning, part 1. *IEEE Comput. Graph. Appl.*, 20(2):89–93, March 2000. ISSN 0272-1716. doi: 10.1109/38.824552. URL https://doi.org/10.1109/38.824552.

Hassen Harzallah. L-systems: draw your first fractals, Nov 2020. URL https://medium.com/@hhtun21/l-systems-draw-your-first-fractals-139ed0bfcac2.

T. Kim and M. C. Lin. Physically based animation and rendering of lightning. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 267–275, 2004. doi: 10.1109/PCCGA.2004.1348357.

T. Kim and M. C. Lin. Fast animation of lightning using an adaptive mesh. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):390–402, 2007. doi: 10.1109/TVCG.2007.38.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015745. URL https://doi.org/10.1145/1015706.1015745.

K. Matsuyama, T. Fujimoto, and N. Chiba. Real-time animation of spark discharge. *The Visual Computer*, 22: 761–771, 2006.

S. G. Narasimhan and S. K. Nayar. Shedding light on the weather. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I, 2003. doi: 10.1109/CVPR.2003.1211417.

L. Niemeyer, L. Pietronero, and H. J. Wiesmann. Fractal dimension of dielectric breakdown. *Phys. Rev. Lett.*, 52: 1033–1036, Mar 1984. doi: 10.1103/PhysRevLett.52.1033. URL https://link.aps.org/doi/10.1103/PhysRevLett.52.1033.

NOAA National Severe Storms Laboratory NSSL. Lightning basics, 2021a. URL https://www.nssl.noaa.gov/education/svrwx101/lightning/.

NOAA National Severe Storms Laboratory NSSL. Lightning types, 2021b. URL https://www.nssl.noaa.gov/education/svrwx101/lightning/types/.

Todd Reed and Brian Wyvill. Visual simulation of lightning. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 359–364, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916670. doi: 10.1145/192161.192256. URL https://doi.org/10.1145/192161.192256.

Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Proceedings of the Third Eurographics Conference on Natural Phenomena*, NPH'07, page 63–70, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673494.

B. Sosorbaram, T. Fujimoto, K. Muraoka, and N. Chiba. Visual simulation of lightning taking into account cloud growth. In *Proceedings. Computer Graphics International 2001*, pages 89–95, 2001. doi: 10.1109/CGI.2001.934662.

NOAA US Department of Commerce. Understanding lightning: Subsequent dart leaders and return strokes, Mar 2018a. URL https://www.weather.gov/safety/lightning-science-dart-leaders.

NOAA US Department of Commerce. Understanding lightning: Initiation of a stepped leader, Nov 2018b. URL https://www.weather.gov/safety/lightning-science-initiation-stepped-leader.

Jeongsu Yun, Myung-Bae Son, Byungyoon Choi, T. Kim, and S. Yoon. Physically inspired, interactive lightning generation. *Computer Animation and Virtual Worlds*, 28, 2017.

Part II

APPENDICES

A

DETAILS OF RESULTS

Work has been done in finding possible implementations of the methods outlined in the state of the art chapter. Two of finished implementations were found, one for Kim and Lin (2007) and another for Yun et al. (2017), both made by the authors themselves.

A.1 LUMOSQUAD

Based on the methods developed in Kim and Lin (2007), Theodore Kim and Ming Lin, authors of said article, developed the 2D Lightning Simulation program LumosQuad. The program works by using an image in a .ppm format with a black background as a free canvas, serving as a neutral zone. By painting different areas of the image in different colors, one can influence the output simulation in certain ways.

- Green - Prohibits the growing of lightning cells within the colored zones.
- White - Represents the end pixel. Should the Lightning branch hit this pixel, the simulation is stopped and rendered.
- Red - Represents the start pixel. It is the first negative charge created and the point from where all lightning branches.
- Blue - Represents a waypoint system. Every pixel colored blue is preferred when lightning generation occurs. If possible, the generated leader will always pass through it.

LumosQuad allows for many different tweaks within its render and simulation portions. The output of a given simulation results in two files, one .exr file and one .lightning file. The latter of the two represents the results of the simulation while the former contains the render itself. The program allows for one to pass the simulation results into it, bypassing the lengthy process of simulation and allowing one to simply tweak render settings as they please. The render setting, scale, is a third parameter in the programs input and change the render settings from large, blurry glows at smaller values to sharper, tighter ones at larger values.

A.2 LIGHTNING RENDERER

Yun et al. (2017)'s authors developed a Lightning Renderer to showcase the results of their research, using it to showcase the various improvements they made in relation to previous methods. Published on Github, this renderer lacks a formal name is it was made to be used only for the purpose of showcasing the algorithm's power and functionality.

It makes use of a simple premise: generate a bolt of lightning inside a cube, originating from a red cube and ending once it reaches a blue cube, without touching the walls of the encompassing polygon. To trigger the generation of a new bolt, one needs only to press the spacebar.

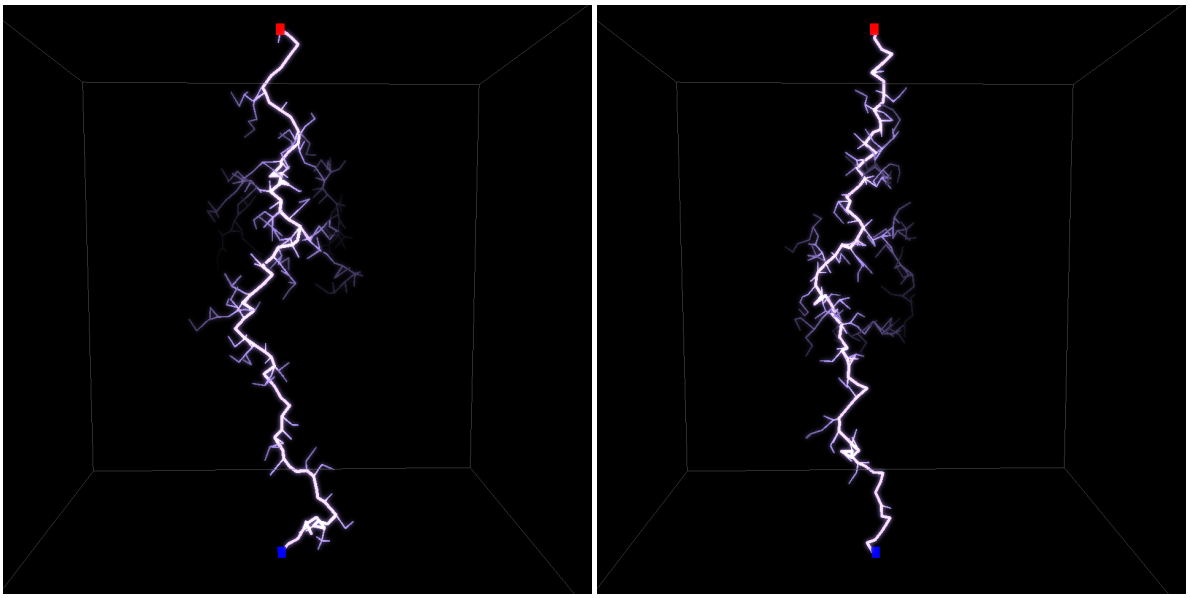
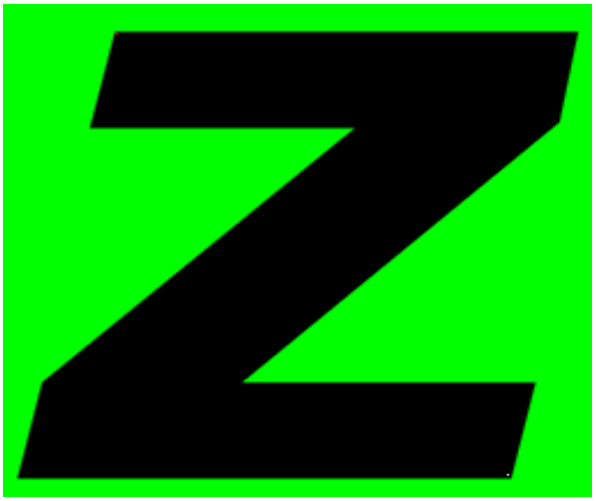
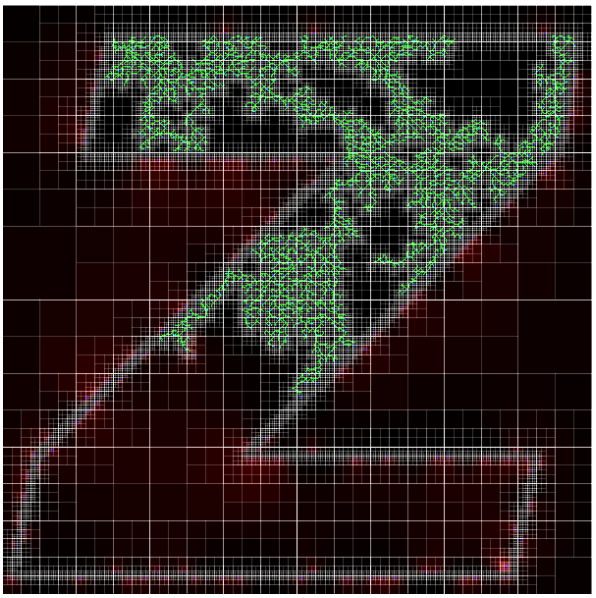


Figure 16: Examples from Yun's renderer

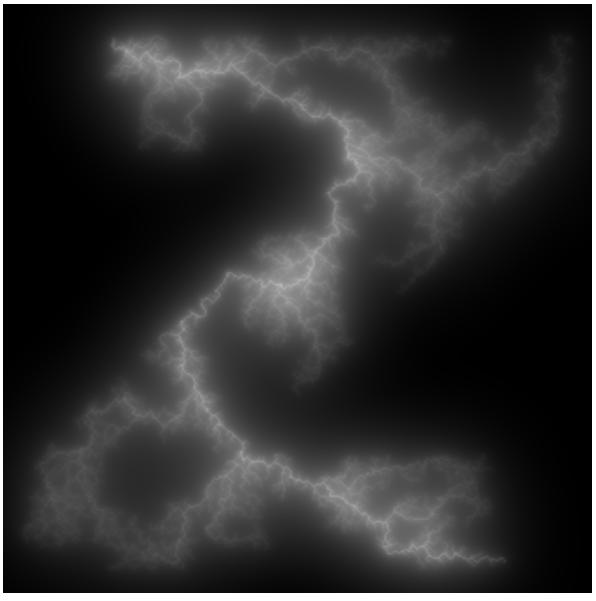
A.3 DETAILED RESULTS



(a) Input .ppm file.



(b) Real-time simulation progression.



(c) Output .exr file.

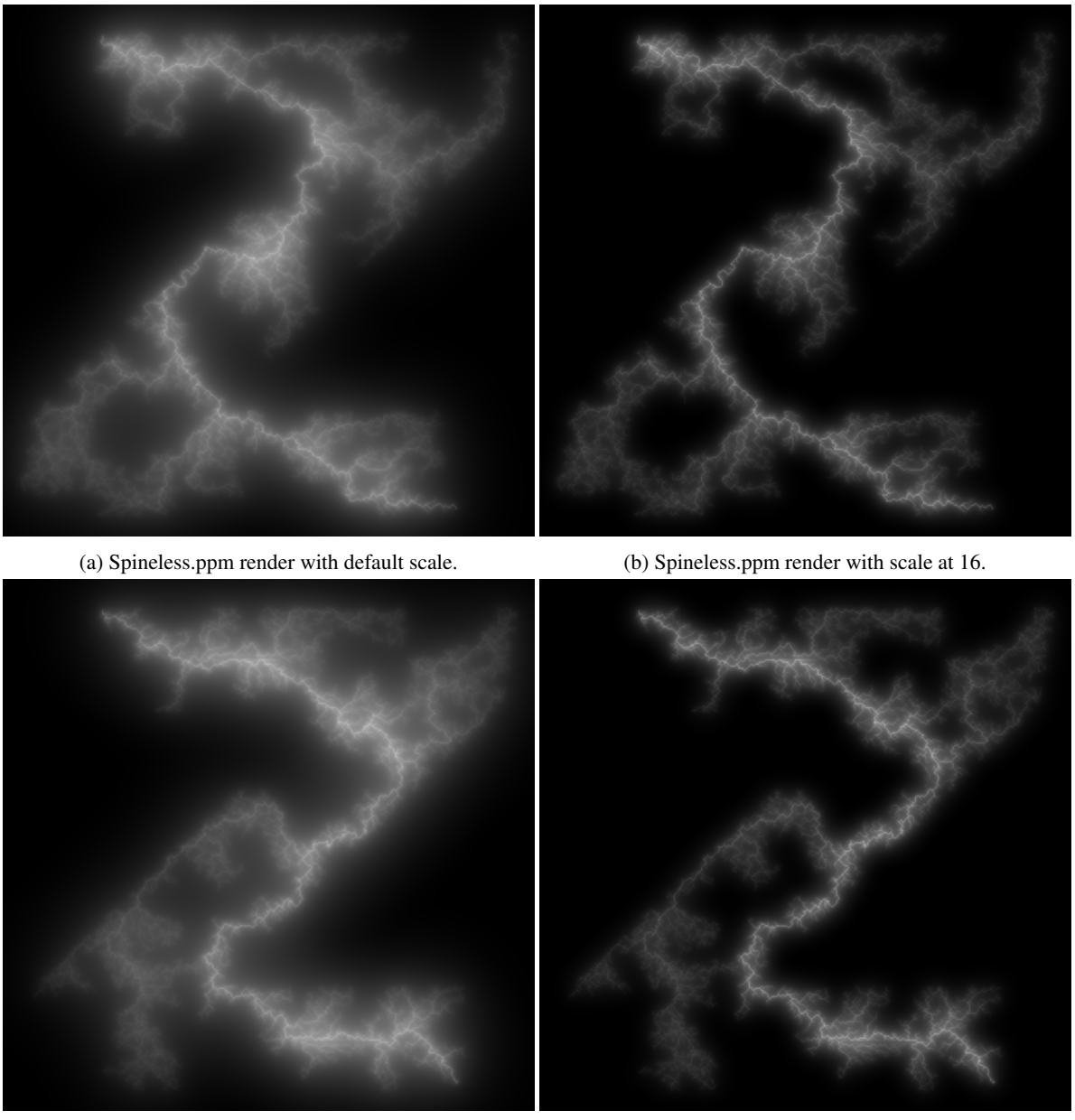
Figure 17: Examples of LumosQuad's functionality.

(a) Example input file: spine.ppm.

(b) Example input file: spineless.ppm.

(c) Example input file: hint.ppm.

Figure 18: Examples of possible input files for LumosQuad.



(a) Spineless.ppm render with default scale.

(b) Spineless.ppm render with scale at 16.

(c) Spine.ppm render with default scale.

(d) Spine.ppm render with scale at 16.

Figure 19: Examples of possible output files and scale parameters for LumosQuad.