석 사 학 위 논 문
Master's Thesis

물리적 특성을 고려한 대화형 번개 생성

Physically-Inspired, Interactive Lightning Generation

2017

윤 정 수 (尹 淨 洙 Yun, Jeongsu)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

물리적 특성을 고려한 대화형 번개 생성

2017

윤 정 수

한 국 과 학 기 술 원

전산학부

물리적 특성을 고려한 대화형 번개 생성

윤 정 수

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2016년 12월 20일

심사위원장 윤 성 의 (인)

심 사 위 원 김 민 혁 (인)

심 사 위 원 박 진 아 (인)

Physically-Inspired, Interactive Lightning Generation

Jeongsu Yun

Advisor: Sung-eui Yoon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Daejeon, Korea
December 20, 2016

Approved by

Sung-eui Yoon
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MCS
20153396

윤정수. 물리적 특성을 고려한 대화형 번개 생성. 전산학부 . 2017년. 18+ii 쪽. 지도교수: 윤성의. (영문 논문)
Jeongsu Yun. Physically-Inspired, Interactive Lightning Generation. School of Computing . 2017. 18+ii pages. Advisor: Sung-eui Yoon. (Text in English)

초 록

본 논문에서는 번개의 물리적 특성을 고려하여 상호작용이 가능한 사실적인 번개 생성 기법을 제시한다. 본 논문에서 제안된 기법은 절연 파괴 현상을 물리적으로 접근한 *Dielectric breakdown model*의 주요 특성을 포착하며, 서로 다른 전하 유형의 전위를 빠르게 계산하기 위해 거리 기반 근사 방법을 사용한다. 특히, 좀 더 나은 번개 패턴을 생성하기 위해 전위의 합을 계산하는 기존의 방법 대신 전위장의 특성을 잘 표현하는 분수식을 사용한다. 또한, 본 논문에서는 복잡한 장면에서 번개 생성 시 느린 번개 경로 생성 시간과 불필요한 잔가지 문제를 해결하기 위해 게임에서 자주 사용되는 중간지점 기법을 사용하여 번개 경로를 안내하는 방법을 제안한다. 제안된 기법을 적용하면 이전의 연구와 비슷한 번개 모양을 수십 배 빠르게 생성할 수 있으며, 상호작용이 가능한 사실적인 번개의 생성이 가능하다.

핵 심 낱 말 번개, 대화형, 절연파괴, 자연현상, 렌더링

Abstract

We present an interactive technique for generating realistic lightning. Our method captures the main characteristics of the *dielectric breakdown model*, a physical model for lightning formation. Our algorithm uses a distance-based approximation to quickly compute the electric potentials of different charge types. In particular, we use a rational function in lieu of summed potentials to better produce interesting lightning patterns. We also propose to use the waypoints commonly available in many game scenes to guide lightning shapes in complex scenes. We found that our algorithm is more than two orders of magnitudes faster than previous approaches and can generate realistic lightning shapes interactively.

Keywords Lightning, interactivity, dielectric breakdown, natural phenomena, rendering

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Realistic simulation and rendering of natural phenomena such as snow, rain, and lightning can improve immersion in movies and games. In movies, lightning is often used to set the mood and emphasize fear. In games, it is frequently used to portray realistic weather and represent the effects of magic.

Unfortunately, generating realistic lightning using physically-based techniques can be very time consuming [7]. While these times may be feasible for movie production, they are too slow for interactive applications like games. As a result, current games utilize pre-rendered images or approaches based on randomized trees. Immersion in the game can be hampered by the repetitive display of the same patterns, and the low quality of the results.

In this paper, we propose an approximate, physically-inspired method to interactively generate realistic lightning. Our algorithm aims to reproduce the main characteristics of a physically based technique, the dielectric breakdown model (DBM) [13, 7], but remove the dependence on numerical solvers such as the conjugate gradient method. We first propose to represent the potential as a rational function that combines different types of electric charges. Then, we introduce additional controls for the lightning shape by combining of two parameters, η and ρ, in our approximation method (Chap. 4.1). Finally, we suggest the use of waypoints to generate lighting shapes for complex scenes (Chap. 4.2).

We have compared our method against the DBM computed using conjugate gradient method [7], and observe speedups of over two orders of magnitude. We also compared our algorithm against the DBM approximation with a summation model of potentials [9]. Our method shows roughly twice as fast as this approach, and we found that ours show better approximation to the DBM approach and thus a wider applicability to different configurations.

Figure 1.1: A night scene with lightning generated by our method. It takes 38 ms on a 128 x 128 grid by using a single core and is about 20 times faster than using the conjugate gradient method from [7].

Chapter 2. Related Works

2.1 Lighting Shapes

Reed and Wyvill [14] propose an empirical method to generate lightning shapes based on the observation that lightning branches are randomly distributed at roughly 16 degrees. New segments are generated by rotating about the parent segment at an angle of about 16 degrees. In the similar approach, Glassner proposes a two-pass algorithm following the statistics of the lightning [4]. A large-scale structure of the lightning stroke is generated in the first step, and then zig-zag patterns are added as details to a long, straight stroke.

Niemeyer et al. presented a dielectric breakdown model (DBM) that physically explains dielectric breakdown phenomena (e.g. lighting and surface discharge) [13]. Thanks to its physical origins, DBM has been used widely for the lightning simulation. The model represents a scene with a regular grid and computes an electric potential for each grid cell according to a boundary condition. Given a cell that has already undergone dielectric breakdown, the next cell is selected randomly, using the electric potential as a selection probability. We will explain the algorithm in detail in chapter 3.

Sosorbaram et al. use the DBM to generate lightning shapes [16], but utilize a local approximation to the electric potential field instead of solving the exact Laplace equation from the DBM. Kim and Lin [7] propose a robust method to generate the lightning shape by solving the Laplace equation with the conjugate gradient method. However, creating the lightning shape can be time-consuming because the conjugate gradient method is an iterative algorithm. To improve the computation time, methods that utilize adaptive meshes like quadtrees or octrees were proposed [1, 8]. [9] propose a fast method that simulates lightning using a distance-based approach. It is similar to [16], which uses an electric potential equation, but uses spherical coordinates.

There are some approaches to generate lightning in real-time. Matsuyama et al. use the GPU to solve the Laplace equation [10]. By limiting the conjugate gradient method to two to four iterations, real-time rendering is achieved. Nvidia provides a real-time DirectX 10 example that uses geometry shader. Unfortunately, this approach is not physics-based.

In a different direction, Xu and Mould generate similar patterns using a path-planning based approach that finds the least-cost paths on a weighted graph within a randomly weighted regular lattice [17].

In this paper, we propose a physically-inspired method that generates lightning by approximating the characteristic value distributions of an electric potential field.

2.2 Lightning Rendering

How to render the atmospheric scattering of lightning is also a significant problem. Traditional rendering techniques that use polygon models are not suitable for rendering lightning.

Reed and Wyvill extended ray tracing to render the lightning stroke and glow [14]. Sosorbaram et al. propose a volume rendering technique using a 3D texture [16]. Dobashi et al. use hierarchically structured metaballs and precomputed lookup tables that store the integrated intensity of light scattering [3]. Kim and Lin [7] and Bickel et al. [1] utilize an atmospheric point spread function, which describes the scattering of light in participating media [12], as a convolution kernel to render the final lightning.

Chapter 3. Background

First, we will introduce the basic physics of the lightning and explain the dielectric breakdown model [13] for lightning simulation. Then, we will present the details of our method for quickly animating and rendering lightning.

3.1 The Physics of Lightning

Lightning occurs when a large charge difference exists between two areas such as a cloud and the ground. 90% of all cloud-to-ground lightning is *downward negative lightning* that occurs when negative charges from a cloud and spread out towards positive charges on the ground. Negative charges move through the lightning stream from the air to the earth until an equilibrium state is reached.

Lightning strikes the ground through several steps. The first stroke is called the *stepped leader* where negative charges spread out and hit the ground through points that have less residual resistance in the air. When the stepped leader reaches the ground, positive charges on the ground move up to the cloud along the path of the leader in a *return stroke*. After the return stroke, subsequent strokes, *dart leaders*, appear that follow the path of the previous leader.

Experimental observations have shown that lightning branches maintain an angle of about 16 degrees with their parent branch, and that the lightning has a fractal dimension of approximately 1.7 [13]

3.2 The Dielectric Breakdown Model (DBM)

The DBM [13] uses a regular grid representation and calculates an electric potential, ϕ, for each grid cell. Figure 3.1 shows a grid representation for the lightning simulation. Negative charges are placed at the top and their electric potentials are set to $\phi = 0$. Positive charges are placed at the bottom and are set to $\phi = 1$. The boundaries of the grid are also set to $\phi = 0$. These three type of electric potentials are fixed and treated as boundary conditions. For the remaining grid cells, the electric potentials are calculated by solving the Laplace equation:

$$\nabla^2 \phi = 0. \tag{3.1}$$

After computing ϕ over the grid, the lightning is grown by randomly selecting a "candidate" cell neighboring the existing lightning. The probability of selection is weighted according to the electric potential. In Figure 3.1, blue cells are the candidate cells for the current lightning. The probability of selection for each candidate cell i is computed using the following normalization equation:

$$P_i = \frac{(\phi_i)^\eta}{\sum_{j=1}^{n} (\phi_j)^\eta}, \tag{3.2}$$

where j is an index of each candidate cell and n is the total number of the candidate cells.

The electric potential for the chosen cell is then set to $\phi = 0$. The chosen cell becomes part of the lightning and is added into the boundary condition. This process is repeated until the lightning reaches the cells that have the positive charge on the ground. Figure 3.2 shows the overall process of generating lightning using DBM. η can control the number of branches. As η increases, the lightning shape has

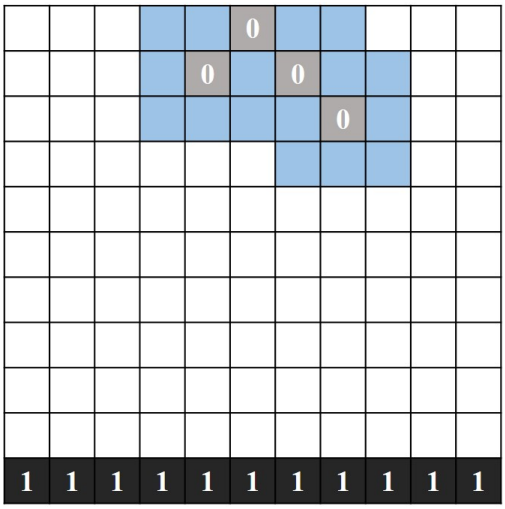


Figure 3.1: The grid used for DBM. Black and grey cells respectively represent the positive charges on the ground and the negative charges at the bottom of cloud. Blue cells are the next candidates that could be added to the lightning.

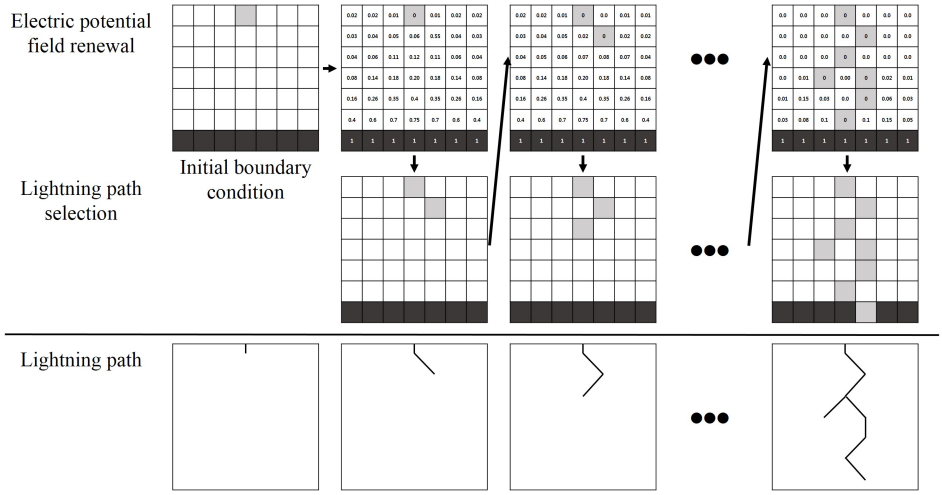


Figure 3.2: Generating the lightning shape using DBM.

fewer sub-branches. Therefore, DBM is often called the η model. A value of two or three is used for a realistic lightning.

Kim and Lin [7, 8] proposed practical techniques for using DBM to generate lightning. These techniques are based on DBM and solve the Laplace equation by using the conjugate gradient method with an incomplete Cholesky preconditioner. While this technique can generate realistic shapes, it can be computationally expensive, because the conjugate gradient method has a time complexity of $O(G^{1.5})$ [15], where G is the number of the grid cells. We found that it takes about 1.5 seconds to create a lightning shape using a 128 x 128 grid.

Chapter 4. Our Method

4.1 Rational Approximation of Electric Potentials

We propose a method that quickly approximates the dominant properties of electric potentials, and allows them to be computed at interactive rates. We do not solve the Laplace equation over a regular grid using the conjugate gradient method. Instead, we approximate the potential by considering its dominant characteristics. We found that there are two simple properties of a potential field that can be described according to its cell type:

- The potential increases towards the positive cells.
- The potential decreases towards the negative cells on the lightning path and boundary cells.

Figure 4.1 shows the electric potential on a 7 x 7 grid. The potential depends on the distance to the boundary cells, similar to other physical equations for electrostatics such as Coulomb's law. We are not the first one to observe such phenomenon. In physics, it is well-known that a potential at a point x that satisfies the Laplace equation is equal to the average potential computed on a virtual sphere located at the point x, each of which is governed by the electric potential equation [5]:

$$V = \frac{1}{4\pi\epsilon_0}\left(\frac{q}{r}\right), \tag{4.1}$$

where q and r are an electric point charge and the distance from the charge to the point x, respectively. This idea is also adopted in prior works [16, 9].

We have found that these techniques can generate less interesting branching patterns in the lightning shape depending on configurations. We conjecture that this phenomenon can occur because the potential can have very similar values between the candidate cells and other charges along the boundary condition. As a result, those candidate cells are likely to have similar probabilities of being chosen. Thus, the computed final lightning shape tends to spread out to all directions from the starting position.

Instead, we propose to approximate the $1/r$ relation of the electric potential that is controlled by ρ as follows:

$$V_{approx} = \sum_{i=1}^{n}\left(\frac{1}{r_i}\right)^{\rho}, \tag{4.2}$$

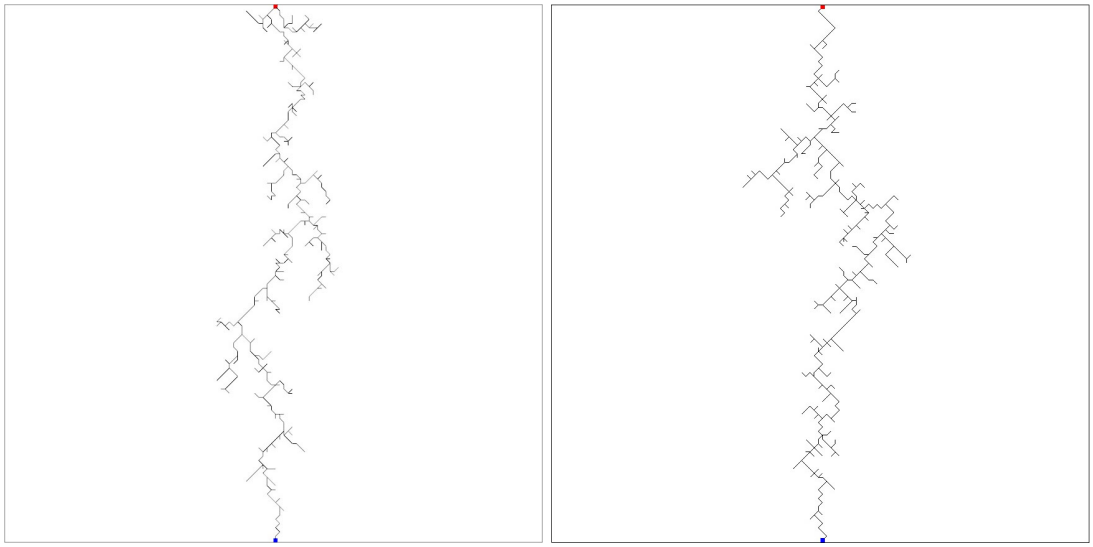
where $\rho > 1$ and i indicates an index of other charged cells; details of considering other charged cells are explained later. Once we have computed the electric potential between the candidate cells and other charged cells, we use Eqn. 3.2.

Before we compute the potentials, we divide the charges into three types: positive charges, negative charges along the lightning path, and boundary charges. We then calculate the electric potentials separately as P, N, and B,

Most prior approaches sum these three terms (P, N, B), to compute the final potential. However, we found that a simple sum does not create interesting branching patterns. For example, consider the 'tip effect', which indicates that a region surrounded by negative charges has a high probability of having negative charge [13]. The summation of the terms fail to produce this effect. The issue arises mainly because we compute potentials by assuming each charged point is in the center of a cell. Thus, the $1/r$ term cannot create extremely small values between even neighboring cells.

0.002	0.003	0.001	0	0.001	0.003	0.002
0.006	0.008	0.002	0	0.002	0.008	0.006
0.015	0.021	0	0.027	0	0.021	0.015
0.032	0.062	0.086	0.106	0.086	0.062	0.032
0.051	0.109	0.176	0.226	0.176	0.109	0.051
0.062	0.148	0.281	0.447	0.281	0.148	0.062
0.05	0.138	0.355	1	0.355	0.138	0.05

Figure 4.1: The values of the potential field after solving the Laplace equation using the conjugate gradient method on a 7 x 7 grid.



(a) CGM (b) Our method

Figure 4.2: Comparison of lightning shapes. Red and blue rectangles represent the start and goal position of the lightning. Our method shows a similar result to DBM, but is about 20 times faster.

To address this issue, and to express the properties of a potential field, we propose to use the following rational function:

$$\phi = \frac{P}{N \times B}. \tag{4.3}$$

Since we divide positive potentials with those of negative ones, we can generate stronger negative potentials among nearby negative charges.

Figure 4.2 compares lightning that was generated by the conjugate gradient method [7] to our method on a 128 x 128 grid. Figure 4.3 shows a potential starting from the top and increasing towards the bottom of a 128 x 128 grid. The summation model that sums P, N, and B does not match well to the result of DBM, while our rational model shows a similar value distribution to the reference.

Arguably, our parameter ρ has a similar effect as the existing parameter η used in the normalization equation. However, we have found that it has a subtle yet meaningful effect compared to that of η. To show the different behaviors of η and ρ, Figure 4.4 shows lightning with different values. As the value of ρ increases, the lightning has fewer branches and show stronger directionality to the target position.

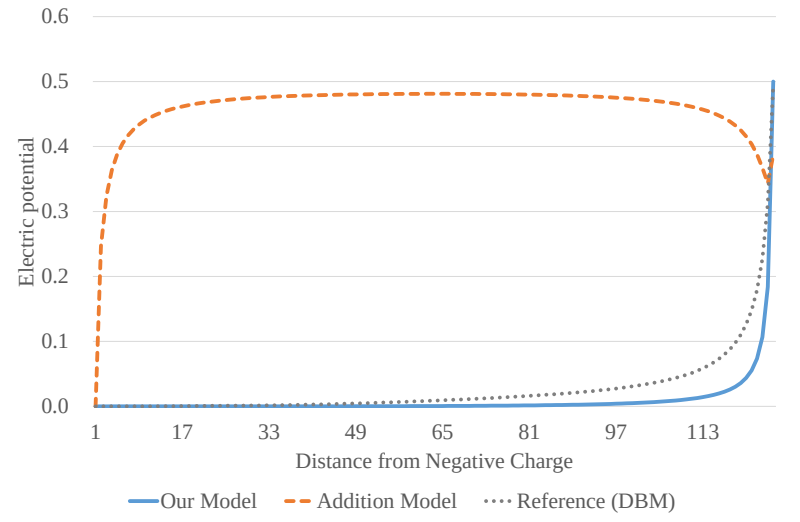


Figure 4.3: Distribution of the potential on the 128^2 grid scene shown in Fig. 4.2. The x and y axes represent distance from the initial negative charge and the computed electric potential, respectively. The summation model does not show a proper value distribution, while our rational model shows a similar pattern to the one computed by the DBM.

The η parameter of DBM also controls the number of branches and shows a similar effect to ρ. Unlike ρ, increasing η trims small branches, while maintaining the main stream. As a result, applying a proper ρ value before computing the probability with η can make more appealing branching patterns than those results only from the η term.

4.2 Avoiding Local Minima

There can be objects or obstacles that should not be hit by the lightning, such as buildings in a game scene. For such obstacles, we assign negative charges and treat them as part of the boundary condition.

When we have a complex scene, our method encounters local minima, similar to ones from other potential field methods [6]. Since our algorithm computes potentials based on distance, when a scene has obstacles that block the target position from the starting position, the lightning branches can spread out excessively (Figure 4.5).

To handle this problem, we utilize waypoints that are generated by a path planning methods such as the A* algorithm. Many games already use fast path planning algorithms to compute such waypoints for various purposes (e.g., computing navigable paths [2]). Waypoints consist of a sequence of points that define a path. To guide a lighting shape in a complex scene, we access the first waypoint cell, W, in the waypoint list and use it as a positive charged cell instead of considering the target positive charges. Once the lighting pattern reaches the cell W, we iterate the process by accessing the next waypoint, and continue until we reach the final waypoint in the list, which we set to the target positive cell. Figure 4.5 shows the lightning path guided by waypoints.

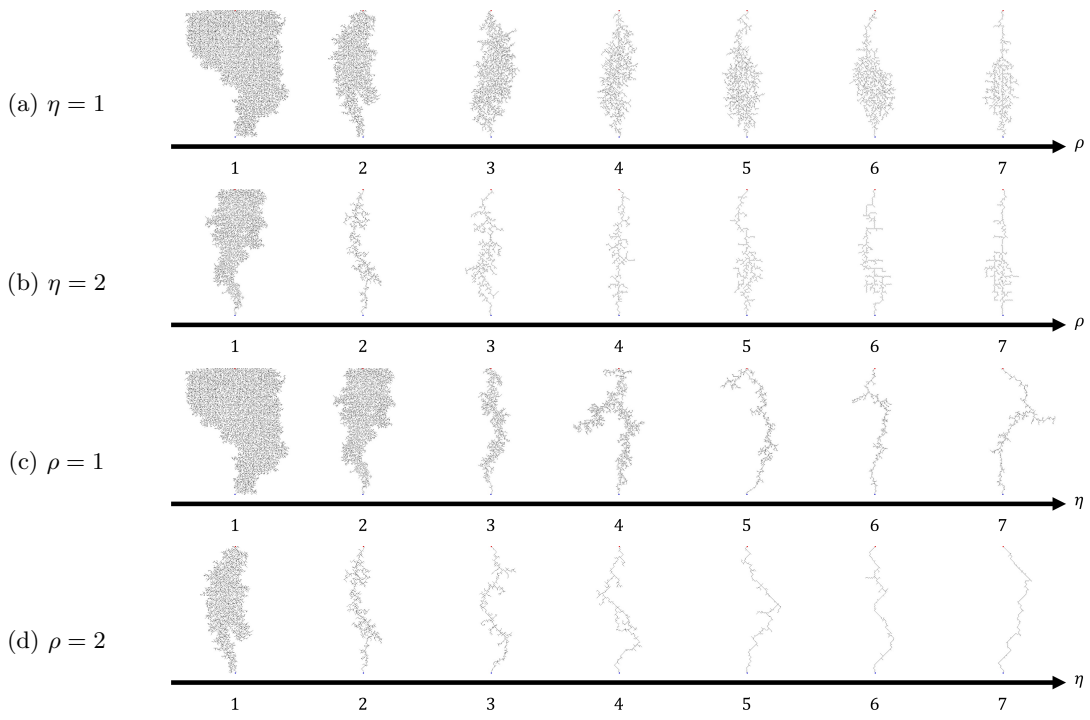
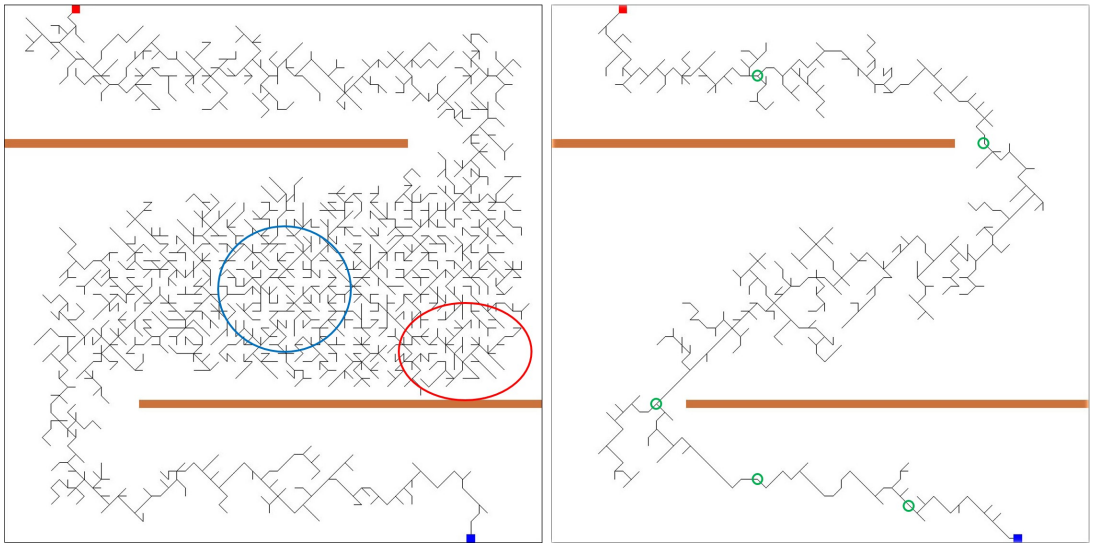


Figure 4.4: Lightning shapes as a function of η and ρ. By increasing ρ values, we can trim down branches with a stronger directionality to the target position. On the other hand, as we use bigger η values, we trim down small branches, while maintaining the main branches; see the pdf file for zoomed-in views.

4.3 Rendering

We utilize physical characteristics such as the thickness and brightness of the lightning stream to render the lighting [14, 7]. The lightning stream is commonly decomposed into the main and secondary channels. The main channel is a path that connects the starting and target points, while the secondary channels are sub-branches of the main channel. The main channel receives a bright and constant intensity, while the secondary channel receives a reduced intensity in proportion to the distance from the main channel. Also, the secondary channel has half of thickness of the main channel.

Our rendering algorithm uses deferred rendering and is implemented as in the OpenGL Shading Language (GLSL). We render a scene and its lightning to separate framebuffer textures. For rendering the lightning shape, we apply the thickness and brightness to each branch while considering the depth buffer. Lines of the lightning are rendered by using billboard techniques as rectangles. To represent the glow effect of the lightning, we use a fast two-pass Gaussian blur filter, which is a widely used technique in games [11]. First, we apply the Gaussian blur horizontally to the framebuffer texture of lightning and, then, use the filter vertically on the previous result. At the last stage, we combine two textures to get the final image. We also utilize jittering [8] to reduce an artifact of grid regularities that appear due to the lack of fine grid resolution.



(a) Result w/o waypoints (b) Result w/ waypoints

Figure 4.5: Lightning shapes computed with and without waypoints. Red and blue rectangles show the start and goal positions of the lightning. The light brown objects represent obstacles. (a) shows the local minima problem. Cells in the red circle are closer to the goal than cells in the blue circle. The lightning tries to grow in the red circle, even though it cannot reach the goal directly. (b) shows the result with waypoints, represented by green circles.

Chapter 5. Implementation and Results

We implemented our algorithm in C++ using data structures in STL. Figure 1.1 shows a night scene with the lightning that is generated by our method. All the experiments are run by using a single core on a 2.6 GHz Core i7 PC.

5.1 Acceleration Details

To accelerate our method, we used pre-computation and clustering. Our method only requires the electric potentials at candidate cells along the current lightning. While candidate cells vary depending on the growth shape of the lighting, the boundary charges and the target positions are fixed. As a result, we can precompute the potentials between any cells on the grid and these static cells.

As the lightning size increases, the number of negative cells and candidate cells increases. As a result, the computation time increases as a function of those cells. To reduce their computation time, we use a clustering technique. Negative charges that are far away from a candidate cell have less influence than those that are closeby. To utilize this, we cluster $c \times c$ cells on the original grid into a cell in a coarse version of the grid. To approximate the charge of the original grid, we calculate a representative charge on the coarse grid that has the average position and aggregated negative charges of the cells from the original grid. Figure 5.1 shows the clustered grid map with representative charges and shows how the potential is computed for the negative charges. If negative charges are in the same cluster as a candidate cell, we use Eqn. 4.2 with each of the negative charges directly. For other negative charges, we compute the electric potential using the coarse grid.

When a new lightning cell is selected, almost all of the old candidate cells remain, and some new candidates are added. For the the candidates that remain from the previous step, we can compute their new potential by incrementally adding the effect of the newly selected lightning cell [9]. For the candidate cells that are newly created, we compute the potential using Eqn. 4.2 and our clusters.

5.2 Comparisons

We ran experiments to compare the performance of our method with DBM [7] and its approximate method, the fast Laplacian growth method [9].

First, we compared the computation time of our method against DBM on a simple scene that has a negative charge at the top and a positive charge at the bottom. We used $\eta = 2$ and $\rho = 3$ in most cases, except for the 128 x 128 and 256 x 256 cases. We used $\eta = 3$ and $\rho = 3$ for those cases to produce similar branching to the result of DBM [7]. We performed ten trials of both algorithms and reported average values. Tables 5.1 and 5.2 show the average time to generate the lightning shape and the number of branches with different grid sizes for both two-and three-dimensional scenes. Our method is about 20 times faster for 2D scenes and 320 times faster for 3D scenes.

We also compared the computation times of different methods as a function of the number of branches for DBM, fast Laplacian growth [9], and our method. We used a 512 x 512 grid and same $\eta = 2$. Table 5.3 shows a computation time for each number of branches. While our method shows significantly faster performance than DBM, our method is slightly faster than the fast Laplacian growth.

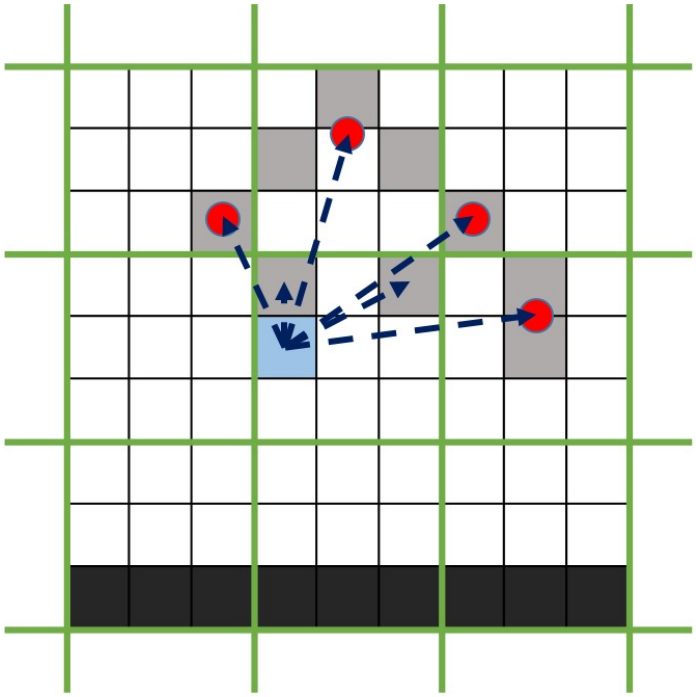


Figure 5.1: The green grid is a low resolution grid with representative negative charges shown in red circles. Grey and black cells respectively show the negative charges and positive charges. The blue cell is a candidate cell. When computing the potential for the candidate cell, we utilize representative cells when they are not in the same low-resolution cell as the candidate cell.

Nonetheless, we found that our method can generate a wide variety of lightning shapes thanks to our rational function and two different parameters, while also handling more complex scenes by utilizing waypoints.

DBM is not suitable for interactive applications, while our algorithm can provide interactive frame rates at 64^2 for 2D and 32^3 for 3D scenes. Furthermore, when we can allow multiple frames, e.g., two or three frames, to asynchronously generate a lighting shape, our method becomes practical with 128^2 and 64^3 grids.

Table 5.1: 2D timing comparisons: The table shows the average time (ms) to generate lightning for different grid sizes. The number enclosed in parenthesis is the average number of lightning branches. Our method is about 20 times faster than DBM.

Grid size (2D)	DBM (Kim and Lin)	Our method
32 x 32	26 (52)	2 (56)
64 x 64	199 (136)	13 (175)
128 x 128	1429 (395)	60 (371)
256 x 256	18555 (1262)	713 (1347)

Table 5.2: 3D timing comparisons: The table shows an average time (ms) to generate lightning. The number enclosed in parenthesis is the average number of lightning branches. The results of DBM with $\eta = 3$ and our method with η=2,ρ=3 have a similar number of branches, while our method is about 320 times faster than DBM.

Grid size (3D)	DBM (η=2)	DBM (η=3)	Our method (η=2, ρ=3)	Our method (η=3, ρ=3)
32 x 32 x 32	4017 (131)	3720 (76)	27 (75)	20 (50)
64 x 64 x 64	117014 (550)	104918 (280)	204 (313)	86 (160)

Table 5.3: Time comparison per branches: 512 x 512 grid and the same $\eta = 2$ value used. Our algorithm is about two times faster that fast Laplacian growth, while our method is significantly faster than DBM.

branches	DBM	Fast Laplacian Growth	Our method	Speedup over DBM
100	87157	10	8	10895x
200	91705	22	12	7642x
300	96864	38	19	5098x
400	102639	55	26	3948x
500	109009	72	35	3115x
1000	149833	181	94	1594x
2000	255792	496	288	888x
3000	343199	921	565	607x
4000	409383	1432	902	454x
5000	458477	2022	1319	348x

Chapter 6. Conclusion

We have presented a physically-inspired, interactive algorithm for generating realistic lightning. Our algorithm shows visually similar results to previous physically based methods at interactive speeds. Our algorithm can be used in games that require realistic lightning or magic effects.

While our algorithm can generate the lightning quickly, higher resolution of the grids consistently result in more realistic lightning. An obvious direction of accelerating our method to support these resolutions is to implement our algorithm on the GPU. This could drastically reduce the generation time and make it possible to use higher resolutions.

Finally, lightning emits light and generates a sound at the time of discharge. For representing a realistic lightning, the lightning should be considered as a light source in the scene. Furthermore, sounds generated by considering the distance between the user and the lightning could have a substantial impact on user immersion in games.

Bibliography

[1] B. Bickel, M. Wicke, and M. Gross. Adaptive simulation of electrical discharges. In *Vision, Modeling, and Visualization 2006: Proceedings, November 22-24, 2006, Aachen, Germany*, page 209. IOS Press, 2006.

[2] X. Cui and H. Shi. A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, 11(1):125–130, 2011.

[3] Y. Dobashi, T. Tamamoto, and T. Nishita. Efficient rendering of lightning taking into account scattering effects due to clouds and atmospheric particles. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pages 390–399. IEEE, 2001.

[4] A. Glassner. The digital ceraunoscope: Synthetic thunder and lightning, part 1. *IEEE computer graphics and applications*, 20(2):89–93, 2000.

[5] D. J. Griffiths and R. College. *Introduction to electrodynamics*, volume 3. prentice Hall Upper Saddle River, NJ, 1999.

[6] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

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

[8] 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.

[9] T. Kim, J. Sewall, A. Sud, and M. C. Lin. Fast simulation of laplacian growth. *IEEE computer graphics and applications*, 27(2):68–76, 2007.

[10] K. Matsuyama, T. Fujimoto, and N. Chiba. Real-time animation of spark discharge. *The Visual Computer*, 22(9-11):761–771, 2006.

[11] J. L. Mitchell, M. Y. Ansari, and E. Hart. Advanced image processing with directx® 9 pixel shaders. *ShaderX2*, pages 439–464, 2004.

[12] S. G. Narasimhan and S. K. Nayar. Shedding light on the weather. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–665. IEEE, 2003.

[13] L. Niemeyer, L. Pietronero, and H. J. Wiesmann. Fractal dimension of dielectric breakdown. *Physical Review Letters*, 52(12):1033–1036, 1984.

[14] T. Reed and B. Wyvill. Visual simulation of lightning. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 359–364. ACM, 1994.

[15] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[16] B. Sosorbaram, T. Fujimoto, K. Muraoka, and N. Chiba. Visual simulation of lightning taking into account cloud growth. In *Computer Graphics International 2001. Proceedings*, pages 89–95. IEEE, 2001.

[17] L. Xu and D. Mould. Constructive path planning for natural phenomena modeling. In *Artificial Intelligence Techniques for Computer Graphics*, pages 83–102. Springer, 2009.

Acknowledgments in Korean

석사과정 동안 부족한 저를 이끌어 주신 윤성의 교수님께 가장 큰 감사를 드립니다. 초반 연구 방향을 제대로 잡지 못하고 있을 때에도 많은 조언을 해 주시고, 열과 성의를 다해 연구지도를 해주신 덕에 이 논문을 완성할 수 있었습니다.

초기에 연구실에 잘 적응할 수 있도록 도와주며 유쾌한 성격으로 언제나 연구실 분위기를 화기애애하게 만들었던 재필이에게 감사합니다. 연구 뿐만 아니라 연구 외적으로도 많은 얘기를 나누며 도움을 주었던 웅직이에게도 감사의 말을 전합니다. 같은 렌더링 그룹으로 연구에 대해 날카로운 지적을 해준 병윤이, 연구와 구현에 많은 도움을 준 명배, 서투른 영어로 말해도 잘 알아듣는 Pio에게도 큰 도움을 받았습니다. 많은 얘기를 나누지는 못했지만 운동을 좋아하던 Peng, 밤낮이 바뀌어 있었지만 늘 밝은 성격의 창민이, 재치있는 성격으로 엠티 분위기를 잘 살리며 춤에 빠져있던 현철이, 연구가 잘 되지 않을 때마다 항상 친절하게 말동무가 되어주었던 용선이, 기숙사에서 편하게 지낼 수 있게 도와 준 민철이, 언제나 성실하게 연구하는 바른 생활 사나이 태영이, 술먹고 자주 힘들어하지만 운동과 연구를 다 잘하던 재형이, 약간 게을러 보이지만 자기 할 일은 모두 잘해내는 명석한 윤석이, 연구실 터줏대감으로 모든 일을 똑부러지게 잘하는 수민이, 엉뚱하지만 언제나 다른 사람들을 도와주려고 솔선수범하는 동혁이, 원두 커피를 나눠주며 신기한 물건을 많이 가지고 있는 인규, 연구 주제를 정하고 연구에 몰두하기 시작한 영기 모두에게 감사의 마음을 전합니다. 또한, 함께한지 얼마되지 않은 희찬이, 재원이, Wladimir, 하인우 연구원님께서도 모두 좋은 연구를 하시길 바랍니다.

마지막으로 항상 믿고 응원해 주시는 사랑하는 부모님과 장인, 장모님 그리고 주말부부로 떨어져 있으면서도 항상 걱정하며 챙겨준 아내 윤희와 사랑하는 두 딸 슬이, 결이에게도 감사의 말씀을 전합니다.

Curriculum Vitae in Korean

이 름: 윤 정 수

생 년 월 일: 1980년 8월 1일

주 소: 대전 유성구 대학로 291 한국과학기술원 전산학부 3443호

학 력

1996. 3. – 1999. 2. 성신고등학교

1999. 3. – 2007. 2. 중앙대학교 컴퓨터공학과 (학사)

경 력

2003. 3. – 2005. 12. 가마소프트 게임 개발팀 (병역특례)

2007. 1. – 현재 삼성전자 책임연구원