Modeling Lightning through Recursion

Alex Young
younga6@oregonstate.edu

Original Proposal

I will use OpenGL to model 3D lightning bolt shapes. The pathways of lightning will be recursively generated using the L-system. The lightning itself will be a source of light that shines on other objects. There could be other simple objects that will shine in the lightning and cast shadows. These objects can use textures to make the scene more interesting.

Requirements

- The lightning pattern to be in 3D - generated by OpenGL
- The lightning is a light source
- Lightning being generated recursively using L-system
- The ability to be able to change the lightning's color (using a menu and keyboard press system)
- The lightning pattern to be randomized - each strike will be different
- The ability to step through the recursive generation of lightning (again using menu and keyboard)
- The ability to have the lightning "autoplay" with lightning flashing on and off (menu/keyboard)
- The speed and brightness of lightning will not be entirely realistic in order to get a more clear view of the generated patterns
- The ability to move the eye position around to get a better view of the patterns

Extra Ideas

- Other 3D objects
- Textures on these objects
- Have the object's cast shadows from the lightning light sources
- Clouds

My Project

In my project I use OpenGL to model 3D lightning bolt shapes using recursion. Each iteration of the recursion is saved into an array to be drawn to the screen in the Display() function. There are various keyboard and menu options that can be used to generate lightning bolts.

The generation of lightning bolts is based on a L-system where we start with a line segment and each line segment is divided into 2 or 3 line segments at its midpoint.

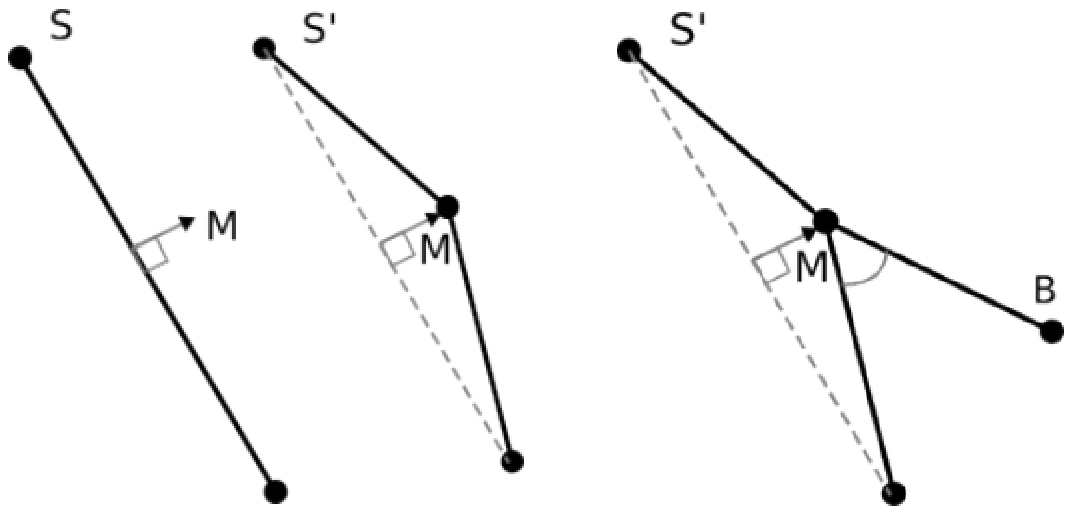
Here the L-System would look like:
Alphabet: SBS`
Axiom: S
Rules: (S -> S` + B) (S` -> S + S) (B -> S)

Here S is a line segment and S` is created by finding the midpoint of S and shifting it perpendicular by a random distance M. The two new line segments together create S`.

The ‘+’ sign represents moving onto the new point generated by shifting over M and drawing a new line segment with this as the origin.

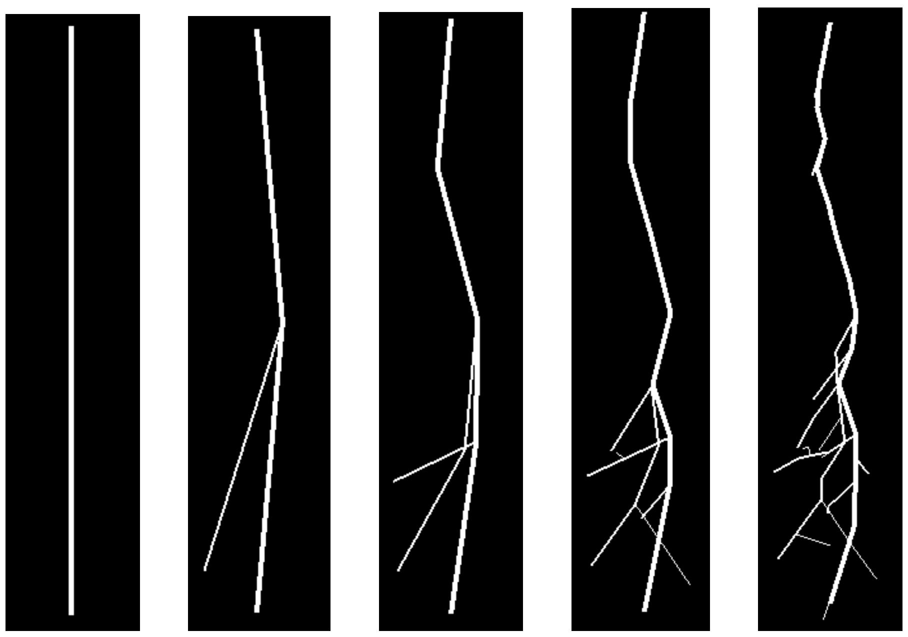
Here the new line segment B can be treated as another line segment with different properties.

This algorithm is better visualized using the following images:



Images from:
https://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S09/final_projects/lapointe_stiert.pdf

This algorithm was implemented into my project which can be seen below:

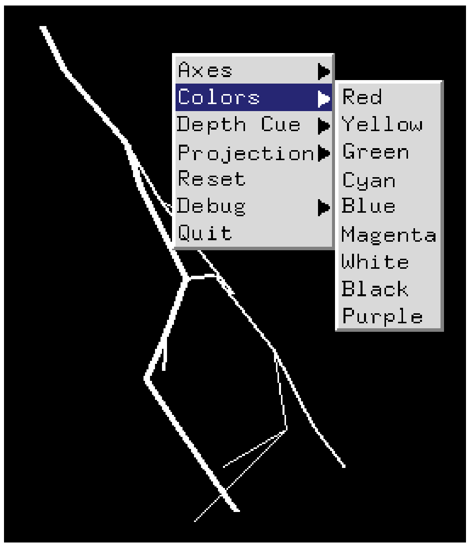


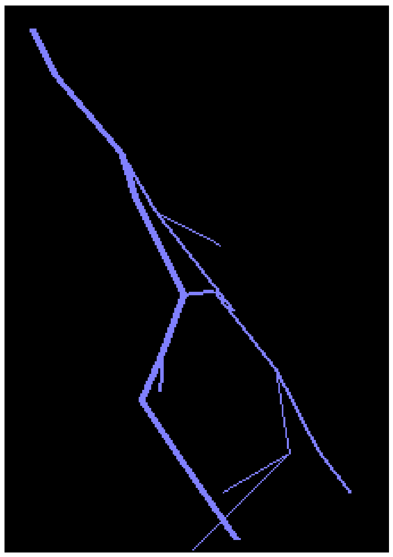
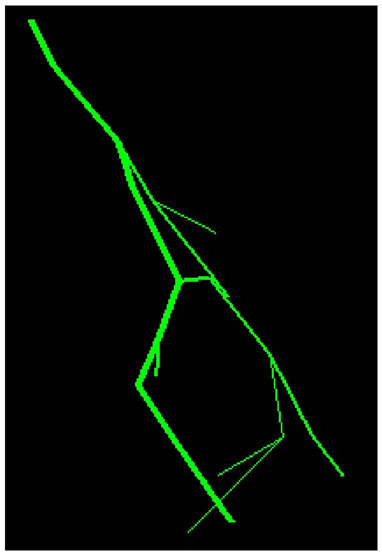
Each image corresponds to a different level of recursion where each line segment is split into two line segments at the midpoint. In addition there is a 50% chance of a bolt being generated at the midpoint (which can be seen by the random length and smaller width). Each bolt is generated at a random angle from the midpoint of S\` which is supposed to emulate a lightning bolt's randomness. In an attempt to make my pattern look more realistic I changed up the bolt's possible lengths and angles, also making it so that it would always spawn in the -y direction.

While my program has capabilities to recurse into a depth level of 7, I had trouble determining lengths and angles to make the image actually look like lightning at higher levels of recursion.

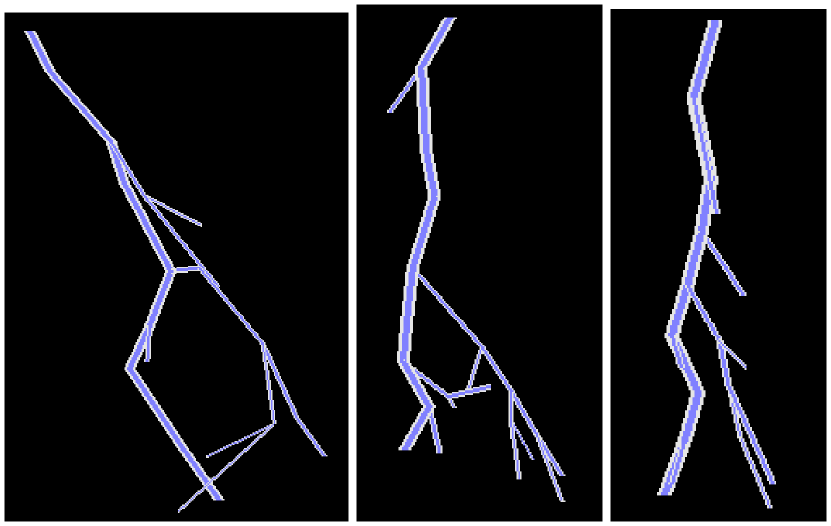
Another difficulty here was keeping track of every single line segment that I was creating in order to both call the recursive function on it as well as draw them out later with lighting.

With this in mind, pressing 'A' to generate a new lightning bolt will only draw up to a depth level of 3 recursions. The color and light from these lightning bolts can be changed in the main menu, as can be seen in the screenshots below:

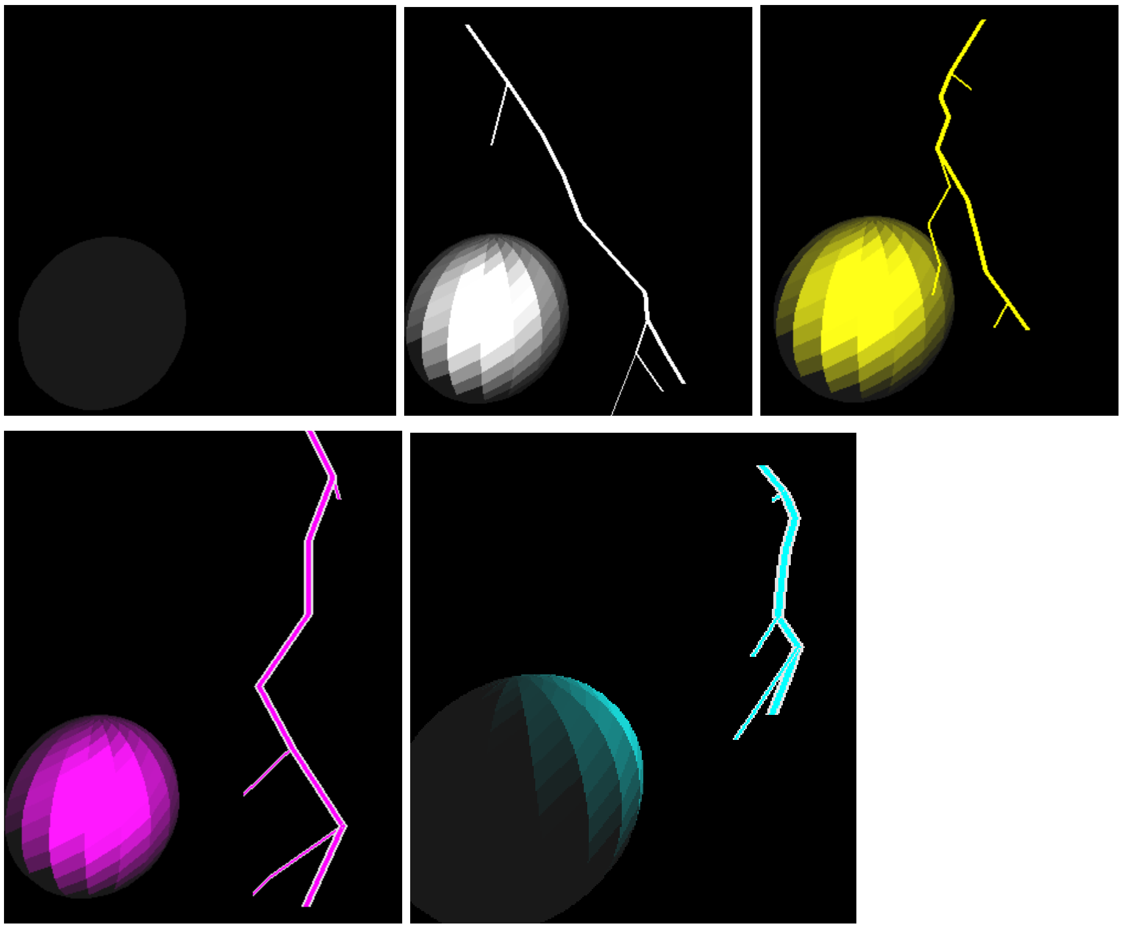


In addition, I have added the ability to press 'G' to add a "glow". However the glow is quite unrealistic and only outlines the lightning in white, without adding any intensity. I ended up including this option in the screenshot below because I was not able to find a more realistic way to emphasize the brightness of a lightning strike. Screenshots of this are shown below:



Finally, by pressing ‘D’, the program will draw a sphere, which is there to show that the lightning is a light source which works with different colors.



What I Learned

The first thing that I learned is that it is very difficult to generate a lightning shaped model to look realistic. Despite following the L-System algorithm as best as I could, at higher levels of recursion it looks quite unrealistic. This is exaggerated by the fact that I had trouble finding a way to make the bolt glow brightly.

The second thing I learned is that with drawing lines in 3-D geometry, understanding how to manipulate coordinates in 3-D is very important. While this may seem obvious at first, I ended up spending a large portion of my time on this project figuring out how to generate the S->S` line segments. In order to do this I found that I had to take the cross product between the input vector and an arbitrary vector to find a perpendicular vector. With the perpendicular vector I could calculate the points on a circle perpendicular to the original vector.

Finally, the most important thing I learned was how to draw objects recursively in OpenGL. While I prepared the algorithm with an L-System, I substituted each grammar in the L-system with certain functions in my code. In addition, I found that it was easier to draw out the recursion if I saved each line segment into an array.

Video explaining my project

Link: https://media.oregonstate.edu/media/t/1_v3cnktqi

In this video I go over the various functionalities of my program and how I meet the various requirements I set for myself. I also describe any difficulties I had and look over the more interesting parts of my program.

References

Lapointe, C., & Stiert, D. (2009). Volume Lightning Rendering and Generation Using L-Systems.