TNM084 - Procedural Frost using L-Systems

Michaela Rabenius

September 2019

1 Introduction

Frost is a very interesting pattern found in nature. It is interesting due to its fractal-like properties which can potentially be procedurally generated. This report describes a project where the goal was to create a procedural frost pattern using computer graphics.

2 Aim

The aim of this project was to create a procedurally generated pattern similar to frost in nature. The main method for doing this was to use l-systems.

3 Frost and l-systems

There are several different types of frost. One example is window frost, sometimes referred to as fern frost, which is the type of frost that can spread over a glass pane due to the difference in temperature between the outside and inside, see Figure 1. This type of frost pattern has several qualities that makes it similar to some type of plants (for example fern, hence the name fern frost), which means that it could potentially be procedurally generated using an l-system.

L-systems (also known as Lindenmayer systems) is a popular tool for creating procedural geometry of various natural shapes. L-systems were created by biologist Aristid Lindenmayer to describe the growth patterns of plants[1] and has since then found usage in the field of computer graphics[2]. An l-system is a grammar-based rewriting system that can be used for drawing shapes. An L-system typically consists

Figure 1: Natural window frost pattern.

of three parts: an **alphabet**, **axiom** and a set of **production rules**[3].

The **alphabet** is a set of characters where each character can be used to define some sort of action, such as *"move one step to right"* or *"move one step to the left"*. The l-system can use the available characters in the alphabet to form a set of characters called a **sentence**. A sentence can be used as a set of instructions for drawing a pattern, where each character defines some sort of drawing action.

The **axiom** is a set of characters that describes the initial state of the system, i.e. the first basic shape drawn. The **production rules** in an l-system are rules that can be used to generate longer sentences from the initial axiom. Typically, this means that a specific character in the sentence can be replaced by a different set of characters to form a new sentence. The sentence grows recursively each time the rules are applied. By iteratively increasing the size of the sentence using the rules available, an increasingly more detailed pattern can be ob-

tained from the starting axiom and with more complex rules more interesting shapes can be created.

For example, take the following system:

Alphabet: AB
Axiom: A
Rules: (A → AB), (B → A)

Over n iterations the sentences develops the following way:

$n = 0$**: A**
$n = 1$**: AB**
$n = 2$**: ABA**
$n = 3$**: ABAAB**
$n = 4$**: ABAABABA**
$n = 5$**: ABAABABAABAAB**
...

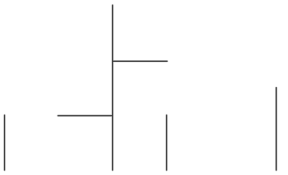
The structure of the l-system can be compared to a tree structure, where the number of n is the structure's *depth*.

This type of pattern generation can be found in several places in nature, for example plants. An l-system can be used to draw a plant by letting the axiom be the base stem and let the rules draw branches originating from the base stem. On the next iteration, new branches will be drawn originating from the previous branches and so on. This creates a self-similar pattern consisting of one shape repeated over and over.

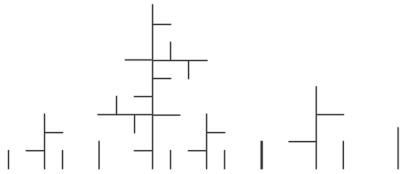
4 Implementation

The project was implemented using OpenGL and the shader language GLSL. The actual pattern generation was done exclusively in a GLSL fragment shader. As described in the previous section, L-system are consists of a set of characters that are recursively generated. This was a problem as GLSL does not support characters nor allow recursion. The consequence of this was that the type of pattern that could be created became limited and therefore it was decided to create a simpler pattern with some likeness to frost, as opposed to creating a complex realistic pattern.

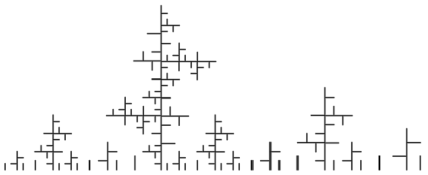
(a) Depth-level 0. The axiom.



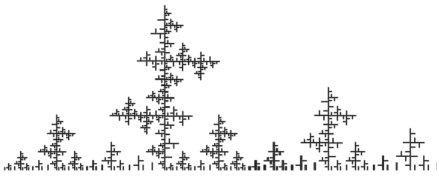
(b) Depth-level 1. The production rules has drawn 5 new lines based on the position and length of the axiom.



(c) Depth-level 2.



(d) Depth-level 3.



(e) Depth-level 4

Figure 2: Different depth-levels of the l-system.

The final implementation takes inspiration from an implementation of l-systems that models plants, created by Max Sills[1]. This implementation uses an iterative loop to draw a number of branches depending on a predetermined depth-level. The larger the depth, the more de-

[1]Max Sills' shader implementation of l-systems `https://www.shadertoy.com/view/XtyGzh`

tail the pattern will have.

To create the frost pattern, a single straight line was created and used as the axiom of the l-system, see Figure 2a. To define the drawing rules of the l-system, a number of different transformation matrices were created. These transformation matrices (rotation and translation) were used to determine where and how to draw new lines based on the position and length of the axiom line.

After some experimenting with the pattern generation, the final l-system has only one drawing rule: for each line five new lines are created, see Figure 2b. Two of the lines originate from a point on the initial line but with a rotation of $\pm$ 90 degrees. The three remaining lines are drawn to the left and to the right of the initial lines and their length are set to either 1/2 or 1/3 the length of the initial line.

The final pattern was created by applying this rule to all lines generated from a predetermined depth-level. By increasing the depth of the l-system, several more and progressively smaller lines are added to the pattern. The lines generated to the sides of the initial lines causes the "frost" to spread to the sides, while the lines generated on the initial line gives a more interesting structure. The result from this is that more detail is added to the pattern with each depth level, see Figure 2c-2e.

5 Results

The final results can be seen in Figure 3. In Figure 3, the pattern created by the l-system was repeated four times with different starting positions to place frost along the edges of the window. The l-system was drawn with depth-level 4. To make the pattern more reminiscent of frost, it was coloured white on a dark blue background.

6 Discussion and future work

The final result is not very realistic but it still has qualities that makes it somewhat reminiscent of frost. The pattern is very regular: all rotations in the pattern are limited to 90 degree angles, while the formation of real frost would be more random. The pattern could possibly be improved by adding some degree of randomness to it, such as randomizing where lines are drawn or the length of the lines. Another thing that could result in a better pattern is to introduce more productions rules to the l-system. By using more rules it could be possible to add more properties to the pattern and make it somewhat more irregular.

The choice to use GLSL exclusively to perform the pattern generation was a big limitation as it may not be the best tool for creating l-systems. The implementation in GLSL also makes it a bit difficult to define new rules as many if-statements are used instead of characters. Because the implementation uses nestled loops and many transformation matrices are created on each frame, it also becomes very computationally expensive very quickly. Rendering the pattern with a depth of 4 takes up a lot of computational resources and significantly lowers the frame-rate. To improve the pattern, a better implementation is needed, either through different coding or by using some other coding language or software.

A disadvantage of this pattern is that it is not based on any actual physics behind frost formation. In any future work it would therefore be interesting to see if it is possible to create a pattern more based on the actual physics of frost and its propagation.

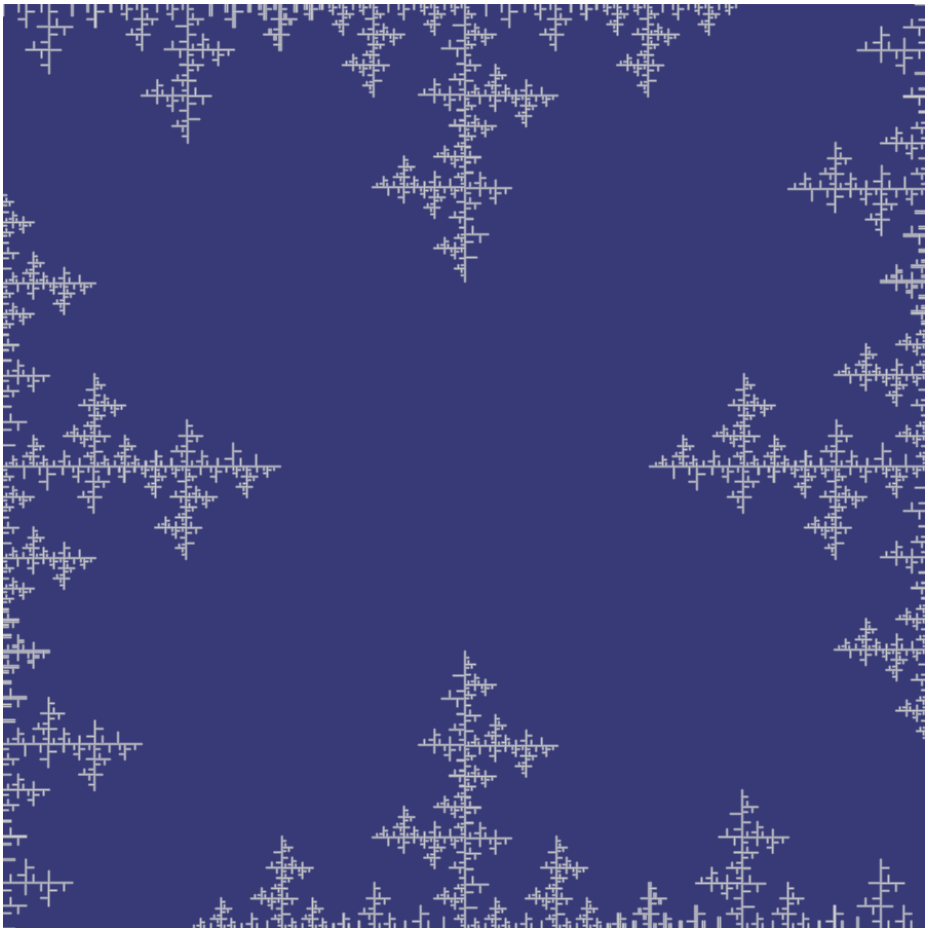


Figure 3: The final generated frost pattern.

References

[1] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1996

[2] D. S. Ebert, F. Musgrave Kenton, D. Peachey, K. Perlin and S. Worley, *Texturing and Modeling: A Procedural Approach*, 3rd edition, Morgan Kaufmann Publishers Inc., 2002

[3] D. Shiffman, *The Nature of Code*, 2012