Abstract

This research introduces a method of using Lindenmayer Systems to model the spreading and behavior of fire inside a factory building. The research investigates the use of L-System propagated fires for determining factors such as where the fire is most likely to spread first and how fast. It also looks at an alternative way of storing the Lindenmayer System, not in the form of a string but rather as a graph. A variation on the building and traversal process is also investigated, in which the L-System is traversed depth first instead of breadth first. Results of fire propagation are presented and we conclude that L-Systems are a suitable tool for fire propagation.

Keywords: Lindenmayer Systems, Fire simulation, Java 3D

1. Introduction

The goal of this research is to develop a fire simulation system based on L-Systems, which will enable the simulation of fire within an enclosed building such as a factory. The model is to be realistic both geometrically and in terms of behavior. The simulation will enable one to determine where the fire is most likely to spread first and how fast. Most fire simulations appear in open areas such as valleys or forests. In this research this is going to be done in an enclosed building, the benefits of this are practical in nature. It will enable one to determine criteria like what part of the building is least likely to be affected by a fire and so, which is the safest part of the building.

A variety of fire simulation models exist. These either focus on the correct behavior of the fire or on the correct propagation thereof. In this research, a model of fire propagation is developed that aims to be accurate in both cases. The method of propagation chosen is Lindenmayer Systems.

L-Systems are well suited for simulation of phenomena which require a lot of interaction with the environment. This fits in well to the problem of fire modeling because fire is usually propagated by examining the conditions in the surrounding environment and then deciding on the manner in which the fire should spread. L-Systems also have a large database amplification factor which means that they can grow from a primitive structure to a very complex one in a short space of time. This is very much like fire, which can start at a single point and spread out to engulf an entire forest or building.

1.1 Overview of the document

The remaining part of section 1 will give a brief overview of Lindenmayer Systems, paying attention only to those aspects that are relevant to this research. Section 2 is used to introduce related work in the field of fire modeling and illustrates how the concept of L-Systems can be applied to fire modeling. Section 3 identifies the main components of the required fire simulation system and goes on to explain how they are constructed. Section 4 and 5 present the results that have been obtained so far and conclusions drawn.

1.2. Lindenmayer Systems

Lindenmayer systems were introduced in 1968 by Aристид Lindenmayer [Prusinkiewicz et al. 1996]. Originally, they were used as a mathematical formalism to model the growth of primitive multicellular organisms as mentioned in the work of Prusinkiewicz et al [Prusinkiewicz et al. 1996]. In essence an L - System is a string rewriting system that consists of three parts:

- The axiom
- The alphabet
- Rewriting rules.

The axiom is the initial state of the system, the alphabet is a set of symbols that may be used to define the system and the rewriting rules are a set of productions that define how the symbols or modules in the L - System are to be replaced with each iteration of the system. The L - System is 'grown' by applying the rewriting rules in parallel to the axiom a prescribed number of times. An example of a basic L - System follows:

axiom = acca
alphabet = {a,b,c,+,-,[,) (where [ and ] mean push and pop the state of the turtle of the stack)

rules =  a-> cb
        b->c[+a]c[-a]c

An example of how this L-System will evolve after a number of iterations is shown below:

0 : acca
1 : cbcccb
...

The graphical representation of an L-System can be obtained by means of using a logo style turtle and a stack. As the final string is interpreted by the turtle, various symbols in the string cause it to perform certain actions. For example, the symbol ‘f’ might cause it to move forward in the direction it is currently pointing in, while the symbol ‘+’ will cause it to turn left by a predefined number of degrees. Figure 1 illustrates how the string ‘a[+ac][a-ac]af’ might be interpreted.

![Figure 1. Possible interpretation of a L-System string](image)

In this interpretation, the turtle puts down more complex pieces of geometry instead of drawing primitive lines and shapes. Figure 1 also illustrates how square brackets are used to allow branching structures in L-Systems. For a more detailed explanation of this approach, the reader is referred to the work of Prusinkiewicz et al [Prusinkiewicz et al. 1996].

Many extensions have been developed for L-Systems in the past years. All of these extend the capabilities of L-Systems and make it possible to model real world objects such as plants more accurately than before. Two of these extensions are relevant to this work and they will be briefly discussed here.

Parametric L-Systems: these allow more control over the L-System as it is possible to associate a set of parameters with each module in the system. A module ‘f’ can now be denoted as f(s,w) where s and w are two parameters which could be used to specify the length and width of a branch considering that f is used to denote a branch. A production can make use of this by specifying what needs to be done to the parameters as the production is applied.

Environmentally Sensitive L-Systems: These L-Systems allow for the system to obtain information from the environment it is growing in and also to write information back. A common case where this would be used is a plant growing beside a building. Before a new branch is grown, the L-System would first query the environment to make sure that the branch is not going to be grown into a wall. Only if this is not the case, does further growing of the branch commence, otherwise the growth is terminated. Communication between the environment and the L-System takes place by using special symbols in the productions which indicate that information is needed from the environment and/or information is to be written to the environment.

2. Related work

The fire simulation model developed in this research makes extensive use of the concept of environment interaction. Especially through environmentally sensitive L-Systems. Research on plants interacting with their environment has been done in the work of Mech et al [Mech and Prusinkiewicz. 1996]. In their research, use is made of Environmentally sensitive L-Systems to model competition for space by branch tiers, spread of a hypothetical plant based on the amount of light available in the environment and development of the root system of a plant. In the last two cases, as the L-System representing the plant grows, it is given a set of symbols that it can use to query the environment for information it needs at a particular spot. Based on what information is received, the L-System controls how the plant or root should grow.

Another approach at environmentally sensitive plant development is described in the work undertaken by Arvo et al [Arvo and Kirk 1988]. In this approach use is made of environmentally sensitive cellular automata to grow the plants. The automata can gather information about their surroundings by casting rays into the environment and determining what they intersect. Based on this information the automata decide whether the conditions are suitable for further growth of the plant.

In [Veach et al. 1994], Veach et al develop system that uses cellular automata to simulate fire propagation. Each automaton has a set of variables accompanying it, these variables represent factors which are important in fire simulations. The automaton examines its own variables, the ones of its neighbors, and subsequently uses Rothermel’s equations to determine the resulting fire at the place where it is located in the environment.

Research presented by Lee et al [Lee et al. 2001] introduces another method of fire simulation. Here emphasis is more on propagating the fire correctly in terms of geometry rather than accuracy. They are more concerned with the fire sticking to the surface it is burning on, rather than how accurate the conditions in the fire are.
In [Beaudoin et al 2001], Beaudoin et al use similar approaches. They develop methods to propagate, animate and render the fire.

L-Systems have been largely used to model individual plants and ecosystems [Prusinkiewicz (1996, 1995, 1993), Hammel 1995, Smith 1984]. The following paragraph aims to justify the use of L-Systems to model fire. The example described below is of a plant (grown as an L-System) interacting with its environment. The experiment is described in more detail in the work of Mech et al [Mech and Prusinkiewicz. 1996].

Figure 2: L-System plant interacting with its environment

Figure 2 shows the plant through a number of stages of growth. The plant thrives in parts of the environment where there is plenty of incoming light, these areas are denoted by the light colored patches. The dark gray areas receive far less light. As the L-System plant grows, it is supplied with three symbols through which it can communicate with the environment. These are ?E(0,x), ?E(1,0) and ?E(2,x). The number in these symbols denotes the type of information exchange between the L-System and the environment. A 0 indicates that the L-System is querying the environment for the amount of incoming light at that spot, 1 serves to notify the environment that a leaf of radius x has been created at that spot and 2 serves to notify the environment that a leaf of radius x has died at that spot.

Using this, the L-System can determine if a certain place in the environment has favorable conditions for the creation of a new leaf and, if so, it can proceed to do so, notifying the environment of this. The environment can in turn keep track of how much light is getting to a particular place because it knows how many leaves have been created there and so it can supply the L-System with accurate information.

This model of plant growth is similar to the approach Veach et al [Veach et al. 1994] use to simulate fire. In their approach, cellular automata were used to move around the environment, evaluate conditions found therein and create the resulting fire. Just like in this example, a L-System is used to move around the environment, determine if conditions are favorable for plant growth and if so, proceed with development of the plant. This indicates that L-Systems can also be used to model fire.

The L-System will grow in the environment, as it does so it examines conditions at the location it is at and grows the fire in a corresponding manner.

3. System Overview

Figure 3.1 illustrates the basic structure of the fire simulation system. The main three components of this system are: the environment, the L-System and the rendering system. Each of these is described in turn

3.1 The Environment

As with other ecosystem and fire simulation models, an environment is needed in which the fire will burn.

The environment is also the platform in which the Lindenmayer System will grow. The environment also stores all the information needed for realistic fire simulation. This information includes quantities like fuel content, geometry, temperature, moisture and pre-ignition temperature.

Because this simulation model is concerned with realism of the fire not only in geometric terms, a polygon mesh is insufficient as an environment. The environment is therefore represented as a three dimensional array of cubes. Each of the cubes contains the above-mentioned information. Each cube also has a set of functions for retrieving and setting this information. The cube representation allows for each room in the building to be divided into discrete spaces, each space having unique conditions of its own, different from its neighbors. This type of accuracy is needed for realistic fire simulation. The data is loaded into the environment through the use of data maps.

The data maps are ASCII files containing numeric values for each of the cubes making up the environment. The geometry of the building is obtained in this manner as well.

Figure 3.2 shows a rendering of the fuel map inside a small building. The fuel map is used to denote how much combustible material is found at different locations in the factory.
3.2 The L-System

The extended version of Lindenmayer systems used in this research is the heart of the fire simulation system. Interaction between the Lindenmayer system and the environment takes place via a feedback loop. The Lindenmayer system may read information from the environment regarding any of the variables stored therein, it then processes this information according to formulas within it and writes the resulting information back. This section of the paper will explain in more detail the structure of the extended L-System and the communication mechanism with the environment.

3.2.1 The extended L-System

The definition of the system is read in from a text file (box labeled ‘L-System definition’ in figure 3.1). The file contains the axiom of the L-System as well as the productions that contain information on how the data in the environment is to be processed and how the L-System is to grow. The file is parsed and the appropriate structures are created.

With conventional L-Systems, the L-System evolves as a string to which the productions are applied. Because fire is an occurrence that spreads quickly, the underlying L-System will grow exponentially as well. In earlier implementations of L-Systems undertaken in this research, the L-System grew to several mega bytes in size after just 11 iterations. This resulted in severe impact on performance and memory usage. This is due to the fact that conventionally, the L-System is built in a breadth first manner and a complete epoch of the L-System must be ready before the next iteration can begin. Also, the environment in which the L-System is to evolve can be quite a large data structure. For example, consider a factory consisting of 1000 by 1500 by 15 cubes, each cube containing 20 bytes of information. The total amount of memory required for this factory will be 450Mb.

To get around this problem a method has been devised to allow the L-System to be traversed in a depth first manner. Using this approach, it is only necessary to build that part of the L-System, which is currently being traversed. To make this possible, the L-System is no longer stored as a string but rather as a connected graph.

The productions and axiom are also stored as graphs. Figure 3.3 illustrates a graph for the production:

\[ t(f,s) \rightarrow m(f,s)[k(f,s)][t(f,s)]p(s) \]

The graph consists of nodes, circles under the RHS (Right Hand Side) label in figure 3.3. The nodes contain information on the parameters that the node carries \((f, s)\) as well as the expressions that need to be applied to them. The letter representing the symbol denoted by each node is also contained within the node. Using this method, the axiom for the L-System may be represented as follows:

\[ m(4,6)[k(4,7)][t(4,9)]p(s) \]

Figure 3.4 illustrates the visual representation of the graph representing this axiom.

Too see how the L-System is built, consider the production in figure 3.3 applied to the axiom in figure 3.4. The resulting L-System is illustrated in figure 3.5.

Because of the nature of the way in which fire spreads, it is necessary to accommodate for branching structures in the L-System. Symbols written in a set of square brackets ‘[‘ and ‘]’
are interpreted as branches and children of their parent node. To distinguish between branching nodes such as k and t in figure 3.3 and nodes that simply follow a node like node p follows node m in figure 3.3, each node has two types of links: a follower link and a set of child links. These are stored as vector data structures within each node.

Figure 3.5 Application of a production

Traversal of the L-System is done using a recursive algorithm. The pseudo code for this algorithm is outlined below:

```
Traverse(graphNode, depth, processor)
Process parameters of graphNode
applicable_rules = find rules applicable to graphNode:
if(depth = 0)
    process graphnode
else
    for every rule r in applicable_rules
        copy parameters from graphNode to RHS of r
        traverse(RHS of r, depth - 1, processor)
    end for
end if
for every child c of graphNode
    traverse(c, depth, processor)
end for
if graphNode has a follower f
    traverse(f, depth, processor)
end if
end Traverse
```

3.2.2 Communication

When simulating fire or ecosystems, some way of communicating with the environment is necessary. This is needed because information about conditions in the environment is necessary in order to make the right decision as to where and how the fire should spread next. The L-System used to simulate fire will therefore have to be environmentally sensitive. The approach used in this implementation is similar to the one used by Mech et al [Mech and Prusinkiewicz. 1996].

Communication occurs through dedicated symbols that may be used in productions to indicate that information is needed from the environment or that information needs to be written to the environment. The symbols used in the system are ‘E’ which indicates that information is needed from the environment and ‘W’ which indicates that information is to be written to the environment. Along with specifying the symbol to use, the type of information that is needed is also specified. This refers to the information that can be found in each of the cubes making up the environment. An Example of the use of these symbols follows.

```
E(x,y,z,getWall) : This call would cause the environment to return whether there is a wall at the location indicated by the three co ordinates x, y and z.
```

```
W(x,y,z, setFlameStage,8): This call would cause the value 8 to be written to the flameStage variable of the cube located at co ordinates x, y, z in the environment.
```

The fact that the system is implemented in Java, allows for the use of the Java Reflection API. This makes it possible to pass a variable like “getWall” as a string and the API will check if the cube class does indeed have such a method and invoke it if that is the case.

These communication functions are the only link between the L-System and the environment. The L-System cannot influence the environment in any other way and vice versa. This approach is the same as the one used by Mech et al [Mech and Prusinkiewicz. 1996] to model the growth of plants and plant roots.

3.3 Fire propagation mechanism

Propagation of the fire is achieved through the growth of the underlying L-System. Conventionally, the symbols constituting the L-System are used to drive a turtle. For the purpose of propagating fire, the symbols are used to tell the L-System which cube in the environment to visit next. An example of a production follows.

```
n(x,y,z):E(x,y,z,getWall)=0.0~W(x,y,z,setTemp,5)n(x,y,z)
```

If this production is applied to a symbol n, it will firstly test if there is a wall at the cube located at x, y and z. If no wall exists at that location, the right hand side of the production will be applied (the part after the ‘~’ character). When this rule is applied, it proceeds to set the temperature variable of the cube located at x, y and z to 5. Following the ‘W’ module is a ‘n’ module which will cause the L-System to move one cube north from the cube located at x, y and z. If this production is applied to a depth of 5, the cumulative result will be that the fire propagates 5 cubes to the north of x, y and z.

Because fire spreads in all directions from the point it originates at, similar rules are defined that allow it to spread to all cubes surrounding the current one.
As the L-System propagates throughout the environment, it extracts information from it, processes this information and writes the information back.

The processing of the symbols is done through an object known as a processor. The processor is a user-defined class that is able to deal with every symbol encountered in the L-System. For example, in the current implementation, if the processor receives the symbol ‘n’, it knows that it must decrement the z parameter by one. Since the processor is responsible for dealing with the various symbols, different implementations of the processor will interpret the L-System in different ways. This makes the system easily extendable.

### 3.4 The Fire model

A very common fire model has been developed by Rothermel [Veach et al. 1994]. This fire model consists of a series of equations that are applied to certain input factors such as temperature, moisture, fuel content, etc. The equations then calculate the conditions, based on the inputs, at the location concerned.

The model used in the current simulations is not as accurate as Rothermel’s model. This is because the system is in early stages of testing. The model is greatly simplified to assist in finding any flaws in the system. Currently the model used is based on intuition. For example, the temperature at a certain spot is set to the average of the temperatures of the blocks surrounding it. If temperature at a certain spot exceeds the ignition temperature of the fuel contained there, the fire will start. The fuel is decremented by a constant amount in the cubes that are burning.

The equations that are applied to the data are incorporated into the rules constituting the L-System. This means that the actual code for the system does not have to be modified in order to accommodate an alternative fire model. This will minimize the effort needed to incorporate Rothermel’s fire model into the system.

### 3.5 Visualizing the simulation

Rendering of the environment is done through the Java 3D API. A voxel approach is used to render the environment. The rendering algorithm runs through every cube in the environment and examines the state of the variables in each cube. Based on the conditions found in each cube, the appropriate geometry is rendered. The algorithm is outlined below:

```
For every cube w along the width
For every cube l along the length
For every cube h along the height
Examine cube at (w,h,l)
Create appropriate geometry
Transform geometry
Insert appropriate geometry into J3D
branchGraph
```

Figure 3.6 illustrates the environment with fuel present in each of the cubes rendered. Several approaches are being experimented with. In figure 3.6 a + c, the fuel is rendered as a bar, the height of the bar serves to indicate how much fuel there is at a particular spot.

In figure 3.6(a) the bars are opaque while in figure 3.6(b) they are transparent. The transparent bars lend themselves much better to the visualization because they do not obstruct visibility of the fire. In figure 3.6(c) the fuel is rendered as spheres with the color indicating how much fuel is present within a particular cube. The brighter the color the more fuel there is.

In a typical fire simulation, a substantial amount of variables are necessary to simulate the fire realistically. These include among others: fuel, moisture, temperature, ignition temperature and rate of burn. It would make the visualization of the fire very cluttered if the state of all these variables was to be rendered simultaneously. The user of the system is therefore allowed to pick the variable that needs to be rendered. The system is forced to refresh the geometry and only the variable the user selects is rendered.

Whereas the variables are rendered through the use of geometric primitives available through Java3D, the fire is rendered by using a primitive implemented for the purpose. The geometric primitive used for fire takes the shape of a pyramid with fire texture mapped onto it. To add to the realistic appearance of the fire, the texture is randomly selected from a set of pre-loaded textures. Figure 3.7 shows the effect of using random textures.

![Figure 3.6 Fuel map rendered inside a factory (a) using solid bars, (b) using transparent bars and (c) using spheres.](image-url)
The height of the pyramid represents the strength of the fire. Although this seems like a very primitive approach, the results of using this compact fire primitive are convincing, this can be seen in figure 3.8.

The rendering of the environment is kept separate to the fire simulation process itself. The function of the rendering system is simply to reflect the state of the environment. While a fire simulation is run, the L-System must finish an evolution step and then call the rendering system to render the environment in order to display any changes that may have been made.

Figure 3.7 The fire primitive (a) using same texture, (b) using random textures

Figure 3.8 Fire and fuel rendered together. The L-System for the fire is described in the next section.

4. Results

Figure 4.1 shows propagation of a fire over several iterations of the L-System.

The L-System used in this fire simulation is shown below. The fire is started as a line in the middle of the room and there is one production for propagating the fire to the left. The L-System looks as follows:

Axiom:
T(44,0,24) T(44,0,25) T(44,0,26) T(44,0,27) T(44,0,28)
T(44,0,29) T(44,0,30) T(44,0,31) T(44,0,32) T(44,0,33)
T(44,0,34) T(44,0,35)

Production:
T(x,y,z)\cdot E(x,y,z,\text{getWall})=0 \Rightarrow W(x,y,z,\text{setFuel},\text{getFuel}(x,y,z)-2)W(x,y,z,\text{setTemp},\text{getFuel}(x,y,z) \ast 1.5)T(x,y,z)

Where T indicates that the fire is to spread to the left. The L-System does reflect the model correctly. The fire is stronger where there is more fuel and it also burns in that spot for a longer period of time. As the fire consumes the fuel, the fire decreases in strength. The part of the fuel map that initially had more fuel, burns for a longer period of time and breaks off from the fire front. The breaking away of fire from the propagation front is something the fire propagation methods in [Lee et al. 2001] and [Beaudoin et al 2001] neglect.

Figure 4.1 Evolution of the fire over several iterations

<table>
<thead>
<tr>
<th>Iterations</th>
<th>L-System time (ms)</th>
<th>Rendering Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>1710</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>1540</td>
</tr>
<tr>
<td>10</td>
<td>2370</td>
<td>1530</td>
</tr>
<tr>
<td>15</td>
<td>3350</td>
<td>1320</td>
</tr>
<tr>
<td>20</td>
<td>5930</td>
<td>1260</td>
</tr>
</tbody>
</table>

Table 4.1 Timing results

Table 4.1 shows some timing results for the system. As can be seen, the time taken for traversal of the L-System steadily
increases as the number of iterations increases. This is expected since the L-System has more locations in the environment to visit and so more calculations to perform. The timing results of the rendering time are somewhat strange. The deeper the simulation goes, the shorter the time gets. This can be ascribed to the time taken by Java 3D to build the necessary geometry graph. As the simulation progresses, more fuel is used up, thus there is less fuel and fire primitives to add to the geometry graph. This in turn results in shorter rendering times.

The results also show how easy it is to change the fire model used by the system. For example, figure 4.2 shows the fire propagation after several steps using the L-System described below.

Axiom:
\[ T(44,0,26)A(44,0,26)T(44,0,27)A(44,0,27)T(44,0,28)A(44,0,28) \]
\[ T(44,0,29)A(44,0,29)T(44,0,30)A(44,0,30)T(44,0,31)A(44,0,31) \]
\[ T(44,0,32)A(44,0,32)T(44,0,33)A(44,0,33) \]

Productions:
\[ T(x,y,z):E(x,y,z,\text{getWall})=0 \Rightarrow W(x,y,z,\text{setTemp}, \text{getFuel}(x,y,z))T(x,y,z) \]
\[ A(x,y,z):E(x,y,z,\text{getWall})=0 \Rightarrow W(x,y,z,\text{setTemp}, \text{getFuel}(x,y,z))A(x,y,z) \]

Where A indicates that the fire is to spread to the right. This system sets the temperature of the cells it visits to the fuel found at that spot. By simply changing the rules, the behaviour of the system is changed.

Figure 4.2 The second L-System

To illustrate that the L-System propagated fire is capable of exploring the environment in a full three dimensional fashion, productions can be added to the system that make it possible for the system to climb up a wall. The resulting fire is shown in figure 4.4. The L-System used for this purpose is shown below:

Productions:
\[ n(x,y,z) : (\text{getWall}(x,y,z-1)=0.0) : n(x,y,z) \]
\[ w(x,y,z,\text{getFuel}(x,y,z), \text{setTemp}(x,y,z,f)) \]
\[ n(x,y,z) : (\text{getWall}(x,y,z-1)=1.0) : v(x,y,z) \]
\[ w(x,y,z,\text{getFuel}(x,y,z), \text{setTemp}(x,y,z,f)) \]

Axiom:
\[ n(30,0,20)n(31,0,20)n(32,0,20)n(33,0,20)n(34,0,20)n(35,0,20) \]
\[ n(36,0,20)n(37,0,20) \]

In the second production, the condition, getWall(x,y,z-1)=1.0, will evaluate to true if there is a wall at the cube the L-System would visit next, if so the rest of the production is applied, which is responsible for propagating the fire up the wall. The symbol v causes the fire to proceed vertically up from where it currently is.

Figure 4.3 The L-System does not spread across environmental boundaries

Figure 4.3 indicates the ability of the L-System to sense the environment. The L-System does not cross the boundaries of the building it is expanding in.
5. Conclusion

This research outlines a method of simulating fire through the use of extended Lindenmayer Systems. Details of the extended Lindenmayer Systems are presented. The system architecture is discussed and the structure and function of each component is explained.

From the results it is evident the Lindenmayer systems are a suitable approach to fire modeling. The fire adheres to the model and results are as expected. This method of modeling fire is also easily extensible which makes it useful if one wants to experiment with different fire models. The research also explains how the fire and the environment are visualized and a number of alternative visualization options are tested.

The model can easily be extended to incorporate other fire models available. Rothermel’s fire equations may be used to make the model more accurate.

6. References


