L-Systems and Procedural Generation of Virtual Game Maze Sceneries

Gustavo Santarsiere Etchebehere* Maria Amelia Eliseo

Presbiteriana Mackenzie, Faculdade de Computação e Informática, Brazil

ABSTRACT

Nowadays, the size of the sceneries of game worlds and the amount of content present in them raised their production costs. At this context, the procedural content generation presents itself as a partial solution to this issue. Among the techniques used to implement the procedural generation, there are the L-Systems. This work proposes a study of the use of L-Systems to generate maze sceneries, which are common in virtual games. Therefore, it was developed a maze game whose scenery is generated by a L-System, and this prototype was submitted to experimentation by the users. By means of this paper it was intended to verify the effectiveness of this technique at the generation of maze-type sceneries. As results, it was found out that the use of this technique produced effective and fittable results.

Keywords: Procedural Generation, Mazes, L-Systems.

1 INTRODUCTION

Game Content (GC) is a key aspect for public interest and approval for a given virtual game. With the constant evolution of electronic entertainment, the development of broad game sceneries such as large terrain and cities full of characters and objects has become possible, causing a fast-growing production costs of games [2].

Therefore, Procedural Content Generation (PCG), the creation of game content by computational procedures or algorithms, is presented as a way to assist in the solution of such problems. The benefits provided by this concept are, according to [6], [7]: 1) reduction of production costs of a game; 2) games with virtually unlimited duration, and 3) generation of content other than those developed by humans.

In order to implement such automatic generation techniques, several types of technologies can be employed. Among these, the L-Systems have aroused our interest for research.

L-Systems are a type of deductive system developed by the biologist Aristid Lindenmayer. L-Systems were designed to simulate cell growth of living organisms. This deductive system can be interpreted by the LOGO language, which provides a graphical representation of the results, making it possible to simulate the growth of more complex beings, such as trees [5]. This method also allows the easy automated description of other types of structures, such as complete cities [4], [10]. As a result, L-Systems have the potential to be used as a sceneries-generating tool for movies and games. Because of the ease of L-Systems in generating lines and branching, it is an opportunity to exploit its potential in other areas, such as maze generation.

The aim of this research is to study the potential of L-Systems as a PCG method for the automated generation of maze-type sceneries and their implementation in a virtual game, through the creation of a system called "maze generator". Such implementation should address game requirements, such as the balanced progression of difficulty in successions of stages, and the achievement of challenging and attractive sceneries for the user. From the implementation, tests were performed with users in order to evaluate the defined requirements and then perform an analysis of the results obtained to validate the potential of the method used and to delimit its limitations.

For this article, we have studied concepts such as PCG, grammars, and L-Systems. Later, applications of L-Systems as PCG method were studied. Then, a L-System was described to execute the procedural generation elaborated for the algorithm, being developed a virtual game. The implemented game was then tested and evaluated by users. Finally, analyzes and conclusions were made on the results obtained.

*e-mail: gustavo.etchebehere@mackenzista.com.br

2 THEORY

2.1 Procedural Content Generation (PCG)

Game content is almost the entirety of what is present in a virtual game. PCG refers to software components capable of creating game content on their own, or in conjunction with one or more players or developers [6], [7].

To implement PCG, there are techniques such as evolutionary algorithms, fractal algorithms [6], cellular automata, genetic algorithms, and generative grammars [9], each of which is appropriate for certain sets of applications. In addition to the generative grammars, there are the L-Systems, used in the procedural generation of vegetation in commercial games, and in the generation of entire sceneries in academic projects [4], [10].

2.2 L-Systems

L-Systems are deductive rewriting systems, where productions are applied in parallel, replacing all the letters of a word simultaneously, using certain substitution rules, in order to compose more complex objects. Deductive systems are composed of a set of axioms, the initial symbols, and the set of rules, which determine the sequence of symbols by which a given character is substituted in the iterations of the deductive system. The symbols that have no productions associated with it are considered the terminal symbols of the L-System [3].

2.2.1 Classes of L-systems

The simplest class of existing L-Systems are DOL-Systems, which substitution rules are static, pre-defined and independent of the elements around them. DOL-Systems, in a simplified way, can be described as an ordered triple G = V, ω, P, where V is the alphabet associated with this deductive system, ω is a non-empty word called axiom, and P is the finite set of productions of the deductive system.

There is a variant of L-Systems called the stochastic OL-Systems. This variant consists of a quadruplet Gπ = {V, ω, P, π}, where V, ω and P have the same definitions of traditional OL-Systems, and π is a function π: P $\rightarrow$ (0,1] , called of probabilistic distribution, and in which it is possible to have more than one production rule for the same symbol A. Such distribution function assigns to a given symbol A production rule the probability of it being chosen in an evolution, with the sum of the probabilities of all rules referring to A being equal to 1 (100%). This way, it is possible to

obtain different final results from each evolution of this deductive system.

In the parametric OL-Systems, each symbol is accompanied by one or more real numbers, denoted by parameters, forming structures with the form A (a1, a2, ..., an) for a symbol A accompanied by n parameters, called a module. Parameters may also consist of undefined symbols, on which arithmetic and logic operations are performed to define the value of the new parameter. A parametric OL-System is defined as a ordered quadruplet G = (V, Σ, ω, P), where V is the system alphabet, Σ is the set of parameters, ω is a nonempty word called axiom, and P is a finite set of productions [5].

2.2.2 Edge Rewriting and Node Rewriting

When using the LOGO interpretation machine, there are two possible interpretations for graphical growth: edge rewriting and node rewriting. Edge rewriting consists of replacing the edges (rows) of a word from the L-System with a set of new graphics. Node writing consists of replacing the vertices of the structure with a new group of graphs, that is, at the connection between two lines [5]. Figure 1 illustrates the differences between them in a graphical and simplified form.

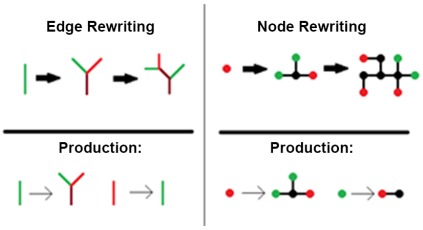

Figure 1: Differences between edge rewriting (left) and node rewriting (right) [5].

2.3 Related Works

The reference [4] have used L-Systems to generate the streets in their system of procedural generation of cities. In this approach, parametric L-Systems were used whose parameters values are not defined by the deductive system itself, but by external functions. These functions take into account global goals and constraints defined in the model to determine the final value of the parameters of the production rules.

In his research, [1] developed software capable of using L-Systems to generate virtual worlds for electronic games, equipped with rudimentary physical mechanics such as collision detection and gravitational action to allow for a certain degree of interaction. His work consists of a system that interprets information from a file containing the description of a L-System and generates its graphic interpretation within a possible field of play.

Based on [4], [8] and [1] this research developed and implemented L-Systems to generate sceneries of maze games. Stochastic, parametric OL-Systems and also stacks were used, which differentiates this proposal from this other studies.

3 A Maze generated by L-Systems

Mazes are puzzles whose challenge is to discover the path that must be taken to get from a starting point to an arrival point through an scenery with corridors, bifurcations and dead-ends. Such a game is commonly found in the print media, but they have also been used in some virtual games, like Pac-Man game field (Namco, 1980) [6].

To validate the use of the proposed procedural generation in a practical scenario, a maze game was developed that had the following requirements: 1. Sceneries with forked bifurcations and alleys, as observed in most mazes; 2. Galleries of varying sizes, to reduce the monotony of the scenery to provide a better use of the game field; 3. Sceneries with a single entry and a single exit, to simplify the gameplay and the tests to be performed; 4. Sceneries generated at random, the generation of each element is drawn, so that a new challenge is provided with each game; 5. Three game levels for test purposes of different sizes and complexities to test the natural growth of L-Systems by limiting the size of the game field; 6. Progression of the difficulty as the player finishes each scenery, that is, changes level, so that players can experience the difficulty provided by each level; 7. At the end of the last scenery, the game restarts so that players can experiment with different sceneries and better understand the characteristics of the project.

The generation of each scenery used a deductive system that was developed for the "maze generator". It was decided to use the node rewriting model, due to the way this model positions the new segments of the graph, allowing better control of how and where the corridors will be generated. With this, the productions are applied to nodes that are always created at the ends of the corridors, not the corridors themselves. These functionalities are the base of the construction of the application and will be described in the next section showing the generator algorithm that allows the formation of the scenery, from an empty field to the beginning of the interaction with the player.

3.1 Scenery Creation Algorithm

The algorithm starts with an empty game field. Then, a point on the game field is chosen randomly to position the axiom. The axiom of the deductive system constitutes a cross, with varying sized corridors, so that the corridors are propagated from the first iterations in all directions available. The corridors can be 3, 4 or 5 units length, with the new generator node formed at the end. Next, a production rule is chosen randomly and applied over each of the new node, resulting in new corridors and new nodes connected to them. The process repeats for all nodes generated until the number of predefined iterations is reached, or until there is no more space available. The number of iterations predefined varies with the complexity of the scenery, being 100, 250 and 500 for the 1st, 2nd and 3rd level of the game, respectively. Figure 2 shows a flowchart summarizing the process of maze creation.

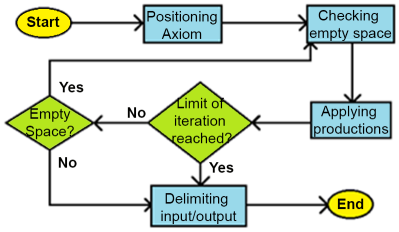

Figure 2: Flowchart of the maze generation process.

The size of the game field will be defined by the degree of difficulty, with ascending dimensions as the level of challenge increases, and will have the measures of the sides of the playing space determined by pre-defined values, these being 25X25, 45X45, and 65X65 in their respective three phases of the game. Each scenery is constructed based on the number of iterations for L-System graph generation, consisting of fewer iterations for the simpler sceneries and more iterations for the more complex sceneries. Iterations consist of symbol replacement cycles that result in the growth of the L-System. The illustration of the three game field sizes, with the same distribution logic of corridor in the final design, is shown in Figure 3.

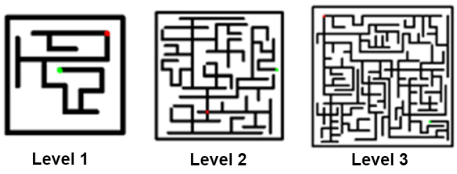

Figure 3: The three game field sizes present in the project.

Stochastic OL-System was used so that it was possible to obtain different results every time the scenery was generated, taking into account the need to obtain different mazes in each game.

Parametric OL-System was necessary in order to meet the requirement of creating variable-length corridors, since this model provides greater practicality and flexibility in controlling the size of generated segments. The parameter of each segment is obtained from an external source, as defined in the application [4]. In this article, the parameter of each segment will have its value defined by drawing among defined values as possible.

The stack was essential in the description of this L-System, to allow the correct formation of the bifurcations existing in the development of the maze.

Figure 4 shows the L-System deductive system developed for the creation of the maze game. The deductive system consists of only four types of symbols, F, G, "-" and "+". F are symbols that constitute the corridors already formed; G are the generating nodes, which create corridors and new nodes, in quantities and directions provided by the rule chosen at random. "-" indicates conversion to the right and "+" conversion to the left. The angle used in this system is 90°.

V : {F, G, +, -}

ω : [--F(rand[2-4])G][- F(rand[2-4])G][+F(rand[2-4])G]F(rand[2-4])G

p1 : $G \xrightarrow{.11}$ F(rand[2-4])G

p2 : $G \xrightarrow{.11}$ -F(rand[2-4])G

p3 : $G \xrightarrow{.11}$ +F(rand[2-4])G

p4 : $G \xrightarrow{.11}$ [-F(rand[2-4])G]F(rand[2-4])G

p5 : $G \xrightarrow{.11}$ [+F(rand[2-4])G]F(rand[2-4])G

p6 : $G \xrightarrow{.11}$ [-F(rand[2-4])G][+F(rand[2-4])G]

p7 : $G \xrightarrow{.34}$ [-F(rand[2-4])G][+F(rand[2-4])G]F(rand[2-4])G

Figure 4: L-System designed to describe the mazes.

The symbol that has associated productions in this deductive system is G and F has only the function of drawing the segments, according to the node rewrite model [5]. Since it is a stochastic L-Systems, there is the probability of occurrence of a given production in the upper part of the arrow of the production rule, shown as a decimal number (.11 means 11% probability of be chosen). The corridors generated are subdivided into 3 forms: one-way corridors, bifurcations and trifurcations, all three cases with the same possibility of occurring. Equal distribution of rules was desired to provide the greatest diversity and unpredictability of corridor distribution as possible.

The function "rand [2-4]", located within the parameters of the symbols F of the production rules, draws the value for the length of the generated corridor. The values can be 2, 3 or 4 units of measurement.

The symbols "[" and "]" provide the demarcation and recovery of the positions of the "maze generator", respectively, in both coordinates and direction.

The proposed system has sensitivity to the environment to avoid corridors overlapping, or even that the maze left the boundaries defined for the game field. This mechanism is applied during the generation of each new corridor, in which an analysis of the space necessary to generate the new corridor and generator node is made. This collision avoidance system also provides a mechanism for joining generator nodes, applied when there is a generator node at the exact location where a new node is to be created, and there is no other structure between them. If two corridors were disputing the same place of creation, priority will be given to the structure generated by the node created first in the process of generating the corridors. The symbols of the production that failed to create the structure are eliminated from the word.

The reading of the input and output nodes of the maze is done as follows: The input node will be, by convention, the central node of the axiom that generates the maze. The output node is assigned to the maze node that is generated last, since it is most likely to be the farthest from the input node.

The process of transcription of the maze, step by step, is shown in figure 5 and uses the L-System described in Figure 4, operating until the fourth iteration. The black frame of the pictures represents the limit of the space of the maze. To the left of each line, the respective iteration number, demarcated by the letter i. The graph shows in green the nodes on which the rule will be applied, in red those in which the rule has already been applied, and corridors in black. The blue lines are the empty space verification process. In frame 9, a loop is formed. Frame 8 shows a broker formation conflict in the upper right corner. In frame 12, the allocation of the input node (in yellow) and the exit point (in purple).

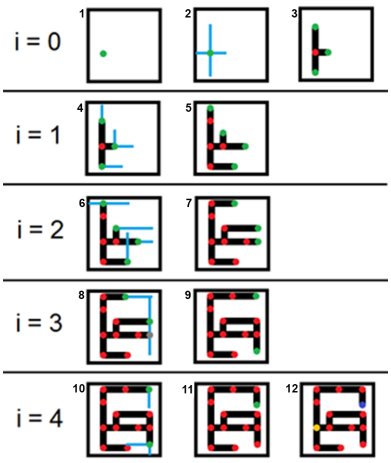

Figure 5: Simulation of the maze transcription process.

3.2 The Application

In order to verify the competence of the proposed generator method, a maze game has been implemented using the L-System to create the scenery.

For the implementation of the game we used the JavaScript language. The final product is a maze game with three phases, with increasing degree of difficulty. We chose to use three phases to simplify the tests. The player begins controlling a blue cube that is located in the spot with the house figure (entrance node), and should lead it to the spot with the arrival flag figure (exit node), after which the player moves on to the next phase. At the end of the last phase, the game returns to the first phase, giving the possibility of experi-

encing it again. Each of the phases generated has a zero-chance to repeat. Figure 6 shows an application game screen.

Figure 6: Image of game field on the first level.

4 Discussion and Results

The application proved capable of generating complex sceneries, which meet the requirements of a maze. The sceneries have variated and interesting shapes that reduce their monotony. The application also has a simple way to adjust the complexity of the challenge, by defining the number of cycles. These features were obtained from the use of L-Systems, through the description of a deductive system with the planned aspects. However, it has been realized that there is a possibility of generating a very small maze, or some mazes too easy to solve.

In order to validate the requirements of the game, it has been submitted to user tests. Twenty-two users participated in this test, who used the application individually, going through all three levels of the game. After the test, each participant answered an evaluative form. Users were previously informed of the purpose of the experiment, as well as basic instructions on how to play. The form consisted of assertions in a Likert scale of five degrees, two positive, two negative and one neutral, which evaluated the technical competence of the game (difficulty of the phases), the aesthetic competence of the game (scenery, motivation) and the gameplay of the application (completion of each level without external assistance, understanding of the objectives of the game), as well as a space for suggestions. Figure 7 shows the evaluation results. These results and their analyzes are shown below.

The statements about gameplay were: (P1) I was able to understand what I should do in the game. (P2) I was able to finish the first level of the game. (P3) I was able to lead the character through the maze. The statements about aesthetic competence were: (P4) I was motivated to continue playing. (P5) The appearance of the scenery was interesting when compared to others of the same nature. The statements on technical competence were: (P6) The degree of difficulty in each scenery was challenging. (P7) The progression of the sceneries was balanced and adequate.

The results show that, in the gameplay requirement, 95.45% of the participants understood how to play, 100% of them finished the first level of the game, and 90.91% of the respondents found the application easy to play. However, only 22.73% strongly agree that the scenery of the game was challenging, 36.36% found the scenery interesting and 31.82% considered that the scenery could be improved graphically. Regarding the motivation requirements for the continuity of the game 59.09% of respondents felt motivated. Also 59.09% strongly agreed that there is balance in the progression of the complexity of the challenge (Figure 7). This shows that the algorithm was efficient in providing growth with adequate difficulty at each level.

As suggestions, some participants pointed out the improvement of the following aspects: 1) A more creative placement of the maze corridors, 2) The presence of instructions in the game, 3) More complex game dynamics.

As shown by these results, L-Systems can be considered with a technology to be used in the development of PGC algorithms for the construction of maze-type games.

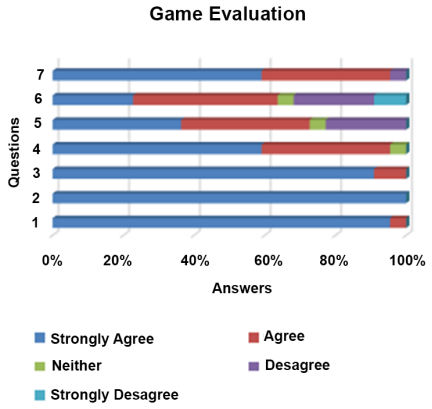

Figure 7: Game Evaluation by the users.

5 Conclusion

With this research, it was possible to construct, analyze and verify the efficiency of L-Systems in the generation of mazes for virtual game sceneries. The characteristics that L-Systems provided as a PCG technique for mazes were the variation of the sceneries obtained, the facility to set their standards, as well as to define the dimensions of each one. However, such a method has poor control over the adverse rules of its growth, such as regions where the presence of corridors is prohibited, which requires the L-System to be accompanied by additional algorithms to ensure the planned results. Through the practical study carried out, it was possible to perceive that the good gameplay generated by the sceneries in the three levels, although the game has presented little challenging.

References

[1] M. Fridenfalk. Application for real-time generation of virtual 3d worlds based on l-system. In *2015 International Conference on Cyberworlds (CW)*, pages 73–78, Oct 2015.

[2] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup. Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1):1:1–1:22, Feb. 2013.

[3] J. Mishra and S. Mishra. *L-System Fractals*. Elsevier Science, 2007.

[4] Y. I. H. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM.

[5] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[6] N. Shaker, J. Togelius, and M. J. Nelson. *Procedural Content Generation in Games*. Springer International Publishing, 2016.

[7] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley. Procedural Content Generation: Goals, Challenges and Actionable Steps. In S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, editors, *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 61–75. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013.

[8] M. A. G. Valverde. Geração de redes vasculares sintéticas tridimensionais utilizando sistemas de lindenmayer estocásticos e parametrizados. Master's thesis, Universidade de São Paulo, São Paulo, 2012.

[9] R. van der Linden, R. Lopes, and R. Bidarra. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):78–89, March 2014.

[10] A. B. Vila Nova. Modelagem procedural de cidades via algoritmo de colonização de espaço. Master's thesis, Univ. Federal de Pernambuco, Recife, 2010.