A Virtual Reality Game Utilizing L-Systems for Dynamic Level Generation

Matthew Yaswinski *, Jeyaprakash Chelladurai, and Shivani Barot

Department of Computer Science, East Stroudsburg University of Pennsylvania, East Stroudsburg, Pennsylvania, USA
Email: myaswinski@live.esu.edu (M.Y.); jchelladur@esu.edu (J.C.); sbarot@live.esu.edu (S.B.)
*Corresponding author

***Abstract*—Virtual reality escape room games have gained popularity for their immersive and challenging nature. This paper presents an L-Systems enhanced virtual reality escape room game that leverages L-Systems to generate unique and intricate room environments. L-Systems are a formal language-based approach used to model and generate geometric forms. By incorporating L-Systems into the room design, the game creates realistic and distinct environments that vary with each gameplay. The paper provides a background introduction, outlines the problems addressed, describes the methods used for L-System integration, presents the results and analysis of player feedback, and proposes a credible algorithm for room generation over traditional techniques. This research showcases the potential of L-Systems as a powerful tool for enhancing the visual richness of virtual environments in VR game development.**

***Keywords*—virtual reality, L-Systems, android, escape rooms, unity, blender, graphics, modeling, GoogleVR**

I. Introduction

The virtual reality gaming market has witnessed rapid growth, driven by technological advancements, and increasing consumer demand for unique and immersive experiences. This paper aims to contribute to this growing industry by creating a virtual reality escape room game for Android smartphones, where players are trapped underground and must solve puzzles to escape the catacombs and subsequently explore an outdoor castle area and its indoor rooms. In addition to utilizing GoogleVR and Unity, our project incorporates L-Systems, a type of formal language and parallel rewriting system, to generate realistic and dynamic room environments. While the technical aspects of the game and plugin have been explored, this paper will also delve into the impact of the plugin on player experience.

To create this game, we employed Blender, Unity, and GoogleVR. Unity serves as the game engine for running code and implementing physics, using version 2019.4.40f1 and programming language C#. GoogleVR, integrated in Unity, enables the game to run on virtual reality headsets and tracks the player's head movement using the smartphone's gyroscope. Blender was utilized for creating custom models of the underground rooms and castle, as well as using L-Systems to generate the walls for each floor of the catacombs. The version of Blender used for this game is 3.3.1, and the programming language for L-Systems structure is Python.

In this paper, we will first provide an overview of the background and history of escape rooms as a form of entertainment. We will then delve into the details of L-Systems, including its definition and the various applications it can be used for. Following this, we will discuss the GoogleVR library for Unity, including its capabilities and how it is utilized in virtual reality gaming. Finally, we will describe how these concepts have been implemented in our own virtual reality game, which utilizes Blender, Unity, GoogleVR, and L-Systems to create a unique and immersive experience for players. The paper will conclude with a summary of our findings, including an analysis of the plugin's impact on player experience, and any suggestions for future research.

II. Related Works

Escape rooms are a form of game in which players are locked in a room and must solve puzzles using clues hidden within to escape and progress to the next area. These rooms often have a dungeon, prison, or other real-world setting and typically require multiple players to work together. Our game, featuring a castle setting, incorporates this concept by dividing the underground area into floors, each containing various puzzles that must be solved to move on to the next area, similar to traditional escape rooms.

L-Systems, also known as Lindenmayer systems, are a type of formal language and parallel rewriting system that utilizes four elements: an alphabet of symbols, production rules for transforming symbols into a string of symbols, an initial string known as the axiom, and a method for converting the generated strings into geometric forms [1]. Developed in 1968 by Aristid Lindenmayer, L-Systems were initially used for modeling the growth processes of plants and understanding the behavior of individual plant cells [2]. Today, L-Systems are also used to create various structures, including snowflakes and roads, beyond just plants. In our game, L-Systems are utilized to generate the walls of each floor in the underground dungeon area, enhancing the realism and complexity of the game design.

Manuscript received June 19, 2023; revised July 21, 2023; accepted September 12, 2023; published February 24, 2024.

An L-system is composed of an initial string, known as the axiom, along with a collection of production rules that determine the substitution of symbols within the string with new strings. Mathematically, an L-system is defined as a tuple (**V, ω, P**), where:

- **V** is a finite set of symbols, called the alphabet or vocabulary of the L-system.
- **ω** is an initial string, called the axiom, made up of symbols from the alphabet V.
- **P** is a set of production rules, also called rewriting rules, that describe how to transform symbols in the string **ω** according to certain conditions.

The production rules defined in set **P** are sequentially applied to the initial string **ω**, starting with the axiom. This iterative process generates a sequence of strings, where each string represents a specific generation of the L-system. The n^{th} string in this sequence is called the n^{th} generation of the L-system, The application of production rules to the string, also known as a derivation or development, results in the transformation and evolution of the L-System structure.

The GoogleVR library is a tool for creating virtual reality experiences in Unity [3]. Originally designed for the Google Daydream device, it enables developers to create VR games for Android devices using Google Cardboard. The library is distributed as a Unity Package file on GitHub [4]. When developing a game with GoogleVR, players can preview the experience by simulating head movement with the Alt key and mouse and tilting the head with the Control key and mouse movement. Additionally, the library features a pointer called GvrReticlePointer, which can be attached to the virtual reality camera and used to interact with objects based on the direction the player is looking. This, along with button inputs from a controller, is the main method of interaction in our game enhancing the immersive quality of the game.

The use of L-Systems in this game is unique as it is not a common approach in the development of virtual reality escape rooms. L-Systems, also known as Lindenmayer systems, is a formal language and parallel rewriting system [1]. It allows for the creation of complex geometric structures by using a set of symbols, production rules, an axiom, and a method for transforming the generated strings into geometric forms [2]. In this game, L-Systems is utilized to generate the walls of each floor of the underground dungeon area, adding an additional layer of complexity to the game design. This approach not only adds to the realism of the game but also provides a new and exciting challenge for players. While many materials and models were created by us, few other assets were gotten from the Unity asset store. These assets, listed at [5–13], were used to enhance the overall gaming experience.

III. Proposed Work

A. Custom Models and Terrain

For this project, we used Blender, a free and open-source 3D modeling program, to create a treasure chest model (Fig. 1), utilizing various tools such as basic transformation, extrude geometry, extrude vertices, and split edges. This chest model was then exported to Unity and animation was added to it as a puzzle for the players to solve. We then apply an animation for opening this chest in Unity for one of our puzzles.

Fig. 1. Treasure chest model.

The castle layout depicted in Fig. 2 was designed to emulate the look and feel of a traditional castle escape game. Utilizing the terrain tool in Unity, a variety of features were employed to create a unique and realistic landscape. Most of the terrain's texture was created using three brushes (raise and lower terrain, paint texture, and set height). Additionally, the Terrain Tools plugin was utilized to simplify the process of building the environment within Unity. To enhance the visual aesthetic, tree and grass elements from the Unity package were incorporated to create a natural exterior for the scene.

Fig. 2. Castle model.

B. L-Systems Model

L-systems are composed of an alphabet of symbols, production rules **P**, for transforming symbols into strings of symbols, an initial string known as the axiom **ω**, and a mechanism for converting the generated strings into geometric forms [1, 14]. Within the L-system framework, edge rewriting and node rewriting serve as two operational modes that draw upon concepts from graph grammars. These modes provide means for manipulating and transforming figures within the L-system structure. Node rewriting involves replacing or modifying individual nodes or vertices within a structure. Production rules are applied to specific nodes, thereby altering their properties or connections. On the other hand, edge rewriting entails substituting figures or shapes for polygon edges. This mode employs production rules to replace specific edges with new combinations of edges, resulting in the generation of intricate and complex structures. In the context of dungeon walls, edge rewriting involves substituting specific segments or sections of the walls with novel combinations

of segments. In our game, we utilized the Blender plugin to implement the following L-system model with the given axiom and production rules, as documented in [1].

$$\begin{aligned} \boldsymbol{\omega}: &\quad F_l \\ F_l \rightarrow &\quad F_l + F_r + + F_r - F_l - F_l F_r - F_r + \\ F_r \rightarrow &\quad -F_l + F_r F_r + + F_r + F_l - F_l - F_r \end{aligned}$$

In this model, the symbols "F_l" and "F_r" represent edges that are generated when the turtle executes the "move forward" command. These edges serve as fundamental building blocks for the structure. During the L-System execution, the production rules substitute instances of "F_l" or "F_r" with pairs of lines that form left and right turns. The symbols "–" and "+" indicate the direction of the turns, with "–" denoting a right turn and "+" denoting a left turn. Each turn is executed at a 90-degree angle. When our plugin executes the L-System with 2 iterations, it applies these rules to generate the corresponding lines and turns, resulting in the desired structure. To enhance the visual representation, we scale the model to increase the height of the walls. The final outcome is depicted in Fig. 3.

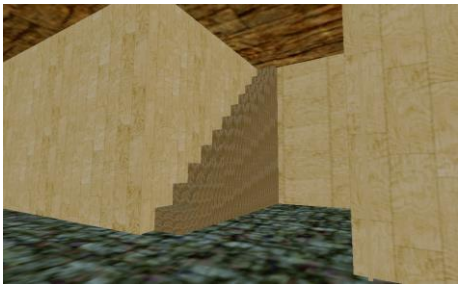

Fig. 3. Stairs model.

Once this is imported into Unity, cubes are placed around it to apply a wall-like texture that the player will see when they are navigating each floor. This allows for more complex materials to be used, whereas this model can only accept solid colors for textures. When it is inserted into Unity and all cubes have been added, the floor looks like what is shown in Fig. 4, but with a ceiling added on top of it.

Fig. 4. Outdoor area.

C. Puzzle Structure

Our virtual reality game has three separate scenes, with two of them being for each floor of the underground area. To play our virtual reality game, a player must have an Android phone and a virtual reality headset that can hold it. In addition, a Bluetooth controller must be connected to the phone in order to interact with objects in the game. The game consists of three scenes, two of which are for each floor of an underground area. The button used to interact with objects in the game is the A button on an XBOX Series X/S controller, though the specific button may vary depending on the controller being used.

The first floor of our virtual reality game features a rotating hint marker that can be selected by the player using a button on a paired Bluetooth controller. The rotation is achieved using Unity's Rotate function, with the angle changing by 90 degrees every second along the Y-axis. The player can select this hint marker by looking at the object and selecting it with a button on the controller. To allow a button press to select the object, the *GvrReticlePointer* (which is placed at the center of the player's eyes, though it is invisible) must be pointing directly to the collision of the object. To select the hint marker, the player must look at the object and use the "Submit" button on the controller, which is the A button on an XBOX Series X/S controller or the Enter key on a keyboard. This causes the object to turn blue and a message to appear on screen, informing the player that hints can be found by selecting these markers. Once this button has been pressed while the pointer was on this object, it will disappear by deleting the object and a message will pop up on screen telling the player that hints can be found by selecting these markers, which will cause a hint message to be shown.

The player can move to different locations in the game by selecting waypoints scattered throughout the scene. Similar to selecting a hint marker, the color of the waypoint changes when the player looks at it. Unlike hint markers, selecting a waypoint smoothly transports the player to a new location instead of destroying the object. The movement is accomplished by Unity's lerp function, which moves the player at a speed of 2.5 units per frame.

In the first puzzle room, the player is told to look into eyes to open the path by selecting the hint marker. The solution to this puzzle is to look at a specific part of a painting on the wall, behind the eyes of which there is an invisible cube. When the pointer is on this cube, it will open the door that leads to the next room and a message will appear indicating that the door has been opened.

In the next chamber, players are provided with a new hint marker that instructs them to proceed along the illuminated path. To solve this puzzle, players must follow the beams of light shining on waypoints. These lights are Unity spotlight objects that have a lower priority than the spotlight on the player's camera, allowing the player's view to remain unobstructed. The objective is to reach a particular waypoint with a light shining on it. Once the player selects this waypoint with their controller, it will be eliminated, and the next door will open. This is achieved by checking the name of the object, and if the waypoint with the specific name has been selected, the object will be deleted, and the door will open.

The next hint prompts the player to gaze at the door for a duration of five seconds, after which something beneficial will occur. Solving this puzzle entails the player directing their gaze at the door until it eventually opens. To achieve this, a timer is employed that increases if the pointer is

directed at the door. Once the timer surpasses five seconds, the door will open and a message will appear, instructing the player to select the steps leading to the next level. The steps are a custom-made Unity object constructed using cubes that are arranged alongside each other with increasing elevations, as depicted in Fig. 3. Each of these cubes possess a "stairs" tag, and if the controller button is pressed while the player's gaze is directed at an object with this tag, the next floor scene will load, enabling the player to solve additional puzzles.

On the following level, players will encounter a hint marker instructing them to unlock the treasure chests in a predetermined sequence. This sequence is determined by a random generation process once the scene is loaded. To generate this random order, the tag names of each of the treasure chest lids are placed into an array. Then, a separate array is created to hold random values. The first element of this new array is obtained by selecting a random integer between zero and two. The next element is acquired by generating a new random number, however, if it is the same number as the previous element, a new random number will be generated until it does not match the previous one. Finally, the last element is obtained by taking the sum of the numbers in the two previous elements. If the sum is equal to one, that means that the previous elements must be zero and one, so the next element's integer will be two. If the sum is equal to two, the previous numbers must have been zero and two, so the next element must be one. Finally, if the sum is three, the previous elements must have been one and two, so the next element must be zero. Once all of these elements have been found, a new array is created to hold values from the original string array, each element is set using the value from the integer array as a position value for the original array, then the old array of tags is set to the new array of tags.

The player will have to open the chests in a specific order, as determined by the randomly generated tag names in the array. If the player attempts to open a chest and the tag name matches the one in the array, it will rotate open using Unity's rotate function for a quarter of a second. If they try to open the next chest and it does not match the next tag in the array, a message will appear telling them that it was not the correct order, and they must try again. Furthermore, the previously opened chests will rotate closed with the opposite angle for the rotate function. Once all the chests are opened in the correct order, the door will be unlocked.

The player must locate and reach each of the corners in the subsequent room. Like the previous floor's lighting puzzle, the waypoints are identified by their names. However, this time the player must visit multiple corners. Once the player reaches a corner's waypoint, it will be eliminated. Once all the corners have been reached, the final door will open.

In the final chamber, the player is confronted with a set of stairs and a clue marker. When the clue marker is activated, it will reveal that the player has achieved their liberation and they are now able to roam the exterior of the castle. By selecting the stairs, the player will be transported to the outdoor castle environment.

In the concluding stage, players are presented with a castle and a number of markers located in various locations on the terrain, this is illustrated in Fig. 4.

The player can move through the outdoor castle terrain, visiting each of the designated locations (as indicated by the waypoints) in the same manner as before. These areas have been adorned with various assets from Unity's asset store and can be seen in Fig. 5.

Fig. 5. First castle room.

IV. Results and Analysis

In this section, we present the results of our L-Systems enhanced virtual reality escape room game and conduct a comparative analysis with existing technologies to thoroughly discuss the impact of our approach.

A. Player Feedback

During the evaluation phase of our game, participants from East Stroudsburg University provided valuable feedback on their experiences. The overall response was positive, with players praising the immersive nature of the game and the unique challenges presented by the L-Systems generated room environments. Participants appreciated the realistic and dynamic structures, which enhanced the sense of exploration and discovery.

B. Comparative Analysis

To provide a comprehensive analysis, we compare our L-Systems enhanced approach with existing technologies commonly used in virtual reality escape room games.

1) Procedural generation

Procedural generation is a popular technique employed in escape room game development [15]. It allows for the creation of randomized room structures, enhancing replayability. However, compared to procedural generation, our L-Systems approach offers greater intricacy and diversity in room designs. The application of L-Systems enables the generation of complex and organic structures that feel more natural and visually appealing.

2) Graph grammars and node rewriting

Some previous works have utilized graph grammars and node rewriting to manipulate and transform room structures [15–17]. While these techniques offer flexibility in modifying individual nodes and connections, our L-Systems approach expands on this concept by generating entire room layouts. L-Systems provide a systematic and rule-based approach to generating realistic and intricate room environments, resulting in a more immersive and challenging gameplay experience.

3) Traditional puzzle design

Traditional escape room games often rely on handcrafted puzzle designs, which can limit the variety and complexity of gameplay experiences [18–21]. Our L-Systems approach overcomes this limitation by dynamically generating puzzle layouts within the room structures. This allows for a greater variety of puzzles and challenges, making each playthrough unique and engaging.

Through our comparative analysis, it is evident that the incorporation of L-Systems in our virtual reality escape room game provides significant advantages over existing technologies. The generated room environments offer a level of intricacy, realism, and variety that enhance player immersion and enjoyment.

V. Conclusions and Future Work

This paper has presented a virtual reality escape room game that utilizes L-Systems to generate unique and engaging room environments. The incorporation of the L-Systems plugin enhances the realism and complexity of the game design, offering players an immersive and challenging experience.

While the technical operation of the L-Systems plugin has been explored, future research should focus on conducting user studies or surveys to gather more comprehensive data on player experiences. This would involve collecting feedback on player immersion, enjoyment, and overall engagement with the game. By analyzing player perceptions and reactions through qualitative and quantitative research methods, a more holistic understanding of the plugin's impact on player experience can be obtained. This valuable information will guide future iterations of the game and contribute to the ongoing development of virtual reality escape room experiences.

Additionally, we aim to improve the game by implementing several new features such as non-player characters, inventory system, a variety of weather conditions, and a day-night cycle, adding more depth to the game, multiplayer environment etc. One of our initial plans was to make this game compatible with Oculus Meta Quest 2, however, we were unable to do so due to time constraints. Therefore, we plan to port the game to this device in the future. Furthermore, we currently use Blender to create the L-Systems models, and then export them to Unity. We hope to eventually integrate this process within Unity, so that each playthrough of the game will have a unique room structure, increasing replay ability.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Matthew Yaswinski and Jeyaprakash Chelladurai were responsible for conducting the research, writing the code in Blender for L-Systems, and implementing the game in Unity. Shivani Barot was responsible for creating the castle and outdoor environment in Blender. All authors approved of the final version.

References

[1] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, New York: Springer, 1996.

[2] L-system. [Online]. Available: https://en.wikipedia.org/wiki/L-system

[3] Quickstart for Google VR SDK for unity with android | Google developers. [Online]. Available: https://developers.google.com/vr/develop/unity/get-started-android

[4] Googlevr. Googlevr/GVR-unity-SDK: Google VR SDK for Unity. [Online]. Available: https://github.com/googlevr/gvr-unity-sdk

[5] Asphalt materials: 2D roads. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/roads/asphalt-materials-141036

[6] Big furniture pack: 3D furniture. [Online]. Available: https://assetstore.unity.com/packages/3d/props/furniture/big-furniture-pack-7717

[7] Conifers [BOTD]: 3d trees. [Online]. Available: https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076

[8] Free Cartoon halloween pack-mobile/VR: 3D fantasy. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/fantasy/free-cartoon-halloween-pack-mobile-vr-45896

[9] Furniture free pack: 3D furniture. [Online]. Available: https://assetstore.unity.com/packages/3d/props/furniture/furniture-free-pack-192628

[10] Grass flowers pack free: 2D nature. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/nature/grass-flowers-pack-free-138810

[11] Outdoor ground textures: 2D floors. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/floors/outdoor-ground-textures-1255

[12] Paintings free: 3D interior. [Online]. Available: https://assetstore.unity.com/packages/3d/props/interior/paintings-free-44185

[13] Yughues free wooden floor materials: 2D wood. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/wood/yughues-free-wooden-floor-materials-13213

[14] P. Prusinkiewicz and J. Hanan, "L-systems: From formalism to programming languages," in *Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, 1992, pp. 193–211.

[15] G. S. Etchebehere and M. A. Eliseo, "L-systems and procedural generation of virtual game maze sceneries," in *Proc. SBGames*, 2017.

[16] T. Voncken, J. Rot, and H. Zantema, "Maze generation, an L-system based approach," Bachelor thesis, Radboud University, Netherlands, 2017.

[17] I. Antoniuk and P. Rokita, "Generation of complex underground systems for application in computer games with schematic maps and L-systems," in *Proc. International Conference on Computer Vision and Graphics, ICCVG 2016*, Springer International Publishing, 2016, pp. 3–16.

[18] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956. doi: 10.2307/2033241

[19] M. T. Trick and S. W. Director, "Lassie: Structure to layout for behavioral synthesis tools," in *Proc. the 26th ACM/IEEE Design Automation Conference*, 1989, pp. 104–109.

[20] M. Cook, "Universality in elementary cellular automata," *Complex Systems*, vol. 15, no. 1, pp. 1–40, 2004.

[21] P. H. Kim and R. Crawfis, "The quest for the perfect perfect-maze," in *Proc. 2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*, IEEE, 2015, pp. 65–72.