

**How Can L-Systems Be Used for Games Development?**

Calum Lindsay – 21010093

2025

Submitted in partial fulfilment of the requirements for the BSc (Hons) Computing (Software Development) degree.

Supervisor: Magnus Tullock

College: UHI Shetland

Department: Computer Science

Submitted: 31/07/2025

**Declaration**

I certify that unless otherwise noted in the text, this is all my own work.

Signed: [Your signature here]

Full name: Calum Lindsay

Student number: 21010093

Date: 31/07/2025

**Abstract**

Creating games requires many assets to be created, and mechanics to be designed which costs both time and money. A procedural generation technique called Lindenmayer Systems has seen some use in games development although the majority of this is in the generation of plants. This dissertation highlights other use cases seen in recent literature and develops two use cases of its own. The first is a terrain generation algorithm which works by having a turtle move nodes on a flat plane up or down as it traverses the output of the L-System producing comparable results to other simple generation techniques. The second is a 3D maze generation algorithm that regenerates and renders the maze at a configurable fixed time step which could be used to create an extremely challenging maze that changes shape as you solve it. The algorithm allows multiple L-Systems to be stacked so that the generation can be split into multiple logical steps. Overall, the dissertation accomplishes its goal of showing that L-Systems could be used to generate significant portions of games in ways that are not commonly seen in the industry.

**Acknowledgements**

Thank you to Magnus Tullock, despite my inability to be present and my lack of communication this year you have been prompt, available, and insightful whenever I have reached out for you which I greatly appreciate.

Thank you to Euan Robertson, you have been a brilliant support throughout the last five years and show a tremendous willingness to help and support every student you can. Additionally, I believe your voracious passion to learn, and share deserves significant praise.

Thank you to Jason Spooner, your small act of kindness many years ago helped direct my energies in the right direction allowing me to develop my abilities earlier than I otherwise would have. Today I find your energy, knowledge, and sense of humour to be exquisite company.

Thank you to my oldest friend, Fergus Tonner, …!!!...!!!...!!!

Thank you to my grandmother, …!!!...!!!...!!!

Thank you to my grandfather for beginning my journey all those years ago.

Thank you to Dannii, Chelsea, and Ray for putting up with me and being my reason.

**Table of Contents**

[1 Introduction 9](#_Toc31925259)

[1.1 Background 9](#_Toc31925260)

[1.2 Outline of Dissertation by chapters 9](#_Toc31925261)

[2 Literature review 10](#_Toc31925262)

[1.1 Section heading 11](#_Toc31925263)

[2.1.1 Subsection heading! 11](#_Toc31925264)

[2.1.2 Another subsection 11](#_Toc31925265)

[2.2 Another section heading 11](#_Toc31925266)

[3 Method 12](#_Toc31925267)

[4 Results 13](#_Toc31925268)

[5 Analysis 14](#_Toc31925269)

[6 Conclusions & Future Work 15](#_Toc31925270)

[6.1 Conclusions 15](#_Toc31925271)

[6.2 Summary of Aims/Objectives and or Research Questions 15](#_Toc31925272)

[6.3 Future Work 15](#_Toc31925273)

[7 Critical Reflection & PDP 16](#_Toc31925274)

[8 Bibliography 17](#_Toc31925275)

[Appendices 18](#_Toc31925276)

**Table of Figures**

[Figure. 1. Figure caption below the figure. Inverted pyramid model for the Literature Review 10](#_Toc31925253)

**Table of Tables**

[Table. 1 Table caption goes here 11](#_Toc31925238)

[Table 2.Table caption above. Not totally necessary but you could use it if you wish. It’s here as an example of how to caption a table. 11](#_Toc31925239)

# Introduction

Creating computer games is an expensive endeavour and in fact over time “the most popular commercial games get larger, prettier, more atmospheric, and more detailed” (Hendrikx, Meijer, Veldon, & Iosup, 2013, p. 2) thus, requiring increasingly higher quality content which is a challenge when “manual content production is already expensive and unscalable” (Hendrikx, Meijer, Veldon, & Iosup, 2013, p. 2).

Procedural generation is fast, cheap and can produce many variations of an asset with similar characteristics which allows the fast creation of large amounts of content with smaller development teams than if the content was manually generated.

Lindenmayer Systems are a specific type of procedural generation, mostly being used “for botanic modelling and generation” (Fornander, 2013, p. 8) with some famous examples of games that use them through a tool called SpeedTree being: Battlefield 3, The Witcher 2, Gears of War 3 (Fornander, 2013, p. 8), Elden Ring and Ghostwire: Tokyo (SpeedTree - Wikipedia, 2024), however, their usage has been fairly limited in other areas such as game mechanics, terrain generation and text generation.

This dissertation builds a picture of the literature both recent and historic and explores the problems and solutions presented within. An emphasis is put on use cases that are not botanical in nature as this area has been well researched. Hypothesises for both further development of discovered solutions as well as entirely original solutions are developed. These Hypothesises are tested by attempting to develop them in to prototype algorithms. The final selection of successful prototypes are compiled into an application for demonstration purposes, tested, and compared to existing solutions.

## Background

### Procedural Generation

Procedural generation in games development is a technique which uses an algorithm to generate content. This could be, for example, using a combination of random noise and mathematics to generate the height of the ground at each point in a game world rather than having an artist do this manually one point at a time. Procedural generation is extensively used due to its wide array of benefits including:

* Reducing development time as designers need to spend less time creating assets which can be created either completely or partially by an algorithm.
* Allowing for almost limitless variations to be generated without consuming any more physical space.
* They can create unexpected and unique gameplay experiences, commonly called “emergent gameplay” when systems interact in unanticipated ways.

### Lindenmayer Systems

Lindenmayer Systems are parallel self-rewriting string generation algorithms at heart and were first proposed as a way of modelling the development of plant like structures by Aristid Lindenmayer in 1968 with later work expanding on this and importantly introducing the concept of a turtle renderer to produce rendered images, culminating in a fascinating book: The Algorithmic Beauty of Plants (Prusinkiewicz & Lindenmayer, 1990).

Lindenmayer Systems consist of two main components:

* An axiom – a string used as the initial state for the generator.
* A set of re-write rules – each rule consists of a symbol to find and a set of symbols to replace it with.

The generator is initialized with the axiom and then on each iteration of the algorithm the rewrite rules are applied.

A white background with black text

AI-generated content may be incorrect.

### Rendering of L-Systems

Rendering is achieved by using a turtle renderer which is an algorithm where a virtual entity named a turtle reads the output of an L-System as a set of instructions to traverse and interact with a virtual environment. The mapping of symbols to actions performed by the turtle is implementation specific although a common configuration for a turtle that can draw edges in a 2D environment and an example output is shown:

A computer screen shot of a tree

AI-generated content may be incorrect.

### Game Assets

The term game asset typically refers to a data file or discrete element that can be used on its own or composed with other assets in order to create logical objects within a game world that can be sensed by the player. This includes but is not limited to:

* Environmental Models – Terrain, boulders, lakes, rivers and plant life.
* Architectural Models – Buildings, roads and bridges.
* Images – Textures, user interface components and maps.
* Text – Dialogues, descriptions and objectives.
* Audio – Voice acting, music and sound effects.

### Game Mechanics

Game mechanics are the behaviours of objects and the rules or systems within the virtual world rather than the objects themselves. Examples of these are:

* Day/ Night Cycles
* Quest Systems
* Patrolling Behaviours
* Skill Systems
* Level Scaling

## Purpose

The purpose of this dissertation is to develop new solutions using L-Systems to existing problems as well as further develop existing solutions that are uncovered through research.

### Hypothesis

That there exists solutions using L-Systems to problems that are traditionally solved using other methods, that there are further developments that could be made to existing solutions using L-Systems, and that exploration of these solutions would prove beneficial for games developers.

### Aims

* Creation and implementation of original solutions in domains with little to no prior work.
* Expansion of existing solutions.
* Combining of elements from multiple solutions to produce new solutions.

### Objectives

* Identify existing solutions and how developed they are.
* Compare developed solutions to existing solutions.
* An application for demonstrating created solutions.
* An interface for tweaking the parameters of the created solutions.
* Reasonable performance of developed solutions.
* Modularity of final solution to allow composition of complex generation systems.

## Structure

**Chapter 2** will present a comprehensive review of the currently known literature.

**Chapter 3** will present the methodology used to create and test the project.

**Chapter 4** will present the results produced during the project.

**Chapter 5** will critically evaluate the project, and the results obtained.

**Chapter 6** will present conclusions drawn and future work.

**Chapter 7** will discuss challenges faced, lessons learned and future goals.

# Literature review

!!!!! This dissertation builds a picture of the literature both recent and historic and explores the problems and solutions presented within. An emphasis is put on use cases that are not botanical in nature as this area has been well researched. Hypothesises for both further development of discovered solutions as well as entirely original solutions are developed. These Hypothesises are tested by attempting to develop them in to prototype algorithms. The final selection of successful prototypes are compiled into an application for demonstration purposes, tested, and compared to existing solutions.

This is your lit review. Introduce and outline what is being discussed here. Start with an inverted pyramid model

**Referencing examples. Please use the different types of layout for in text citations as it makes the finished dissertation look more professional. Use RefWorks rather than Word for referencing.**

**Quoting directly:**

Aitken (2006: 48) argues that land fill sites are “not cost efficient”….

**Paraphrasing:**

Aitken (2006: 48) argues that land fill sites are expensive and inefficient

**Summarising an entire book or article:**

A recent study reveals new information about child poverty in Scotland (Weir 2007).

**Summarising a point made on two consecutive pages of a book or article:**

The book provides examples of how the eating habits of parents directly influence children (Taylor 2006: 19-20).

A close up of a map

Description automatically generated

Figure. 1. Figure caption below the figure. Inverted pyramid model for the Literature Review

If you use a Table then this caption goes above… To create a caption you use menu Insert> Caption then choose the type. If you do this for each table and figure then you can create a Table of Tables and a Table of Figures in the front matter of this document.

Table. 1 Table caption goes here

|  |  |  |
| --- | --- | --- |
| **Subject** | **Author** | **Year** |
|  |  |  |
|  |  |  |

## The Problem

What is the problem that we are trying to solve? Basically time and cost of artists and game designers.

You may choose to divide this into sections, perhaps looking ahead to the method chapter.

Table 2.Table caption above. Not totally necessary but you could use it if you wish. It’s here as an example of how to caption a table.

|  |  |
| --- | --- |
| **Type of Literature** | **Authors** |
| Machine Learning | Bleh et al. 2019 |
| Deep Learning | Mushroom et al. 2014 |

### Subsection heading!

### Another subsection

#### Another subsection

Only go down to four levels of sub-sections. If you need another level just use:

**Another section formatting**

More text.

## L-Systems

The Algorithmic Beauty of Plants (Prusinkiewicz & Lindenmayer, 1990) introduces the concept of L-Systems as a parallel string rewriting algorithm and turtles which interpret the strings produced by L-Systems and render 2D images of plants.

### Stochastic L-Systems

Stochastic L-Systems allow for the creation of generators that provide non-deterministic outputs. This is achieved by allowing multiple rules to be declared that have the same predecessor but different successors and giving each rule a probability where the sum of probabilities of all rules that have the same predecessor must equal 1. When a predecessor is to be replaced during an iteration of the generator a random number will be generated and used to select the successor that is used.

A black and white rectangular object with black text

AI-generated content may be incorrect.

In the example 50% of the time F becomes FF, 25% of the time it becomes +FF, and the remainder of the time it becomes -FF.

### Context-Sensitive L-Systems

Context sensitive L-Systems contain rules that take into account the symbols surrounding the predecessor being evaluated so that rules can be created that will only take effect when the predecessor exists within a certain context (Prusinkiewicz & Lindenmayer, The Algorithmic Beauty of Plants, 1990, pp. 30-32). These rules can make L-Systems significantly more complicated to process where the final product is in 2 or more dimensions as the algorithm needs to examine the proximity of symbols in multiple dimensions while the L-System is a one dimensional string.

A screenshot of a computer program

AI-generated content may be incorrect.

If the output from the example was interpreted by a turtle that interpreted the b as a ball and the other symbols as blank spaces in a 1-D grid we would get a ball that moves back and forth from left to right.

### Pseudo L-Systems

Pseudo L-Systems allow predecessors to be created that are not a single symbol but instead a set of symbols, commonly called a string (Prusinkiewicz, Graphical Applications of L-systems, 1986, p. 250). This means that rules can replace whole strings instead of being limited to single symbols. This is similar to context-sensitive L-Systems in that a pattern of symbols is required for a rule to be applied but differs in that the entire pattern is replaced rather than a single symbol.

A screenshot of a computer program

AI-generated content may be incorrect.

This example gives some hope to the idea that L-Systems could be used to generate textual content although multiple types of system would need to be combined to produce anything useful.

### Parametric L-Systems

Parametric L-Systems allow parameters or variables to be associated with symbols in the string being operated on. This allows arithmetic operations to be performed upon these values in rewriting rules and a turtle can then interpret the parameters depending on what is desired (Prusinkiewicz & Lindenmayer, The Algorithmic Beauty of Plants, 1990, pp. 40-42). A simple example being the thickness of a line that is drawn allowing simulation of plants becoming thicker at their base as they grow by increasing this parameter on each iteration.

### Extended L-Systems

An Extended L-systems is where a parametric L-System produces a generic template with parameters unassigned, a second step is performed where the parameters of the L-System are set by a function to bring the L-System within specific global goals and, finally, a third function modifies the parameters to keep them within local constraints. This allows the output of the L-System to be modified without changing its rule set (Parish & Müller, 2001, p. 303). Examples of global goals might be that there can only be a certain number of a specific symbol or that all instances of a specific type of generated object should have a specific scale defined by a parameter of the L-System. Examples of local constraints might be that 2 generated objects must be a minimum distance apart or that an object must fit within a certain space.

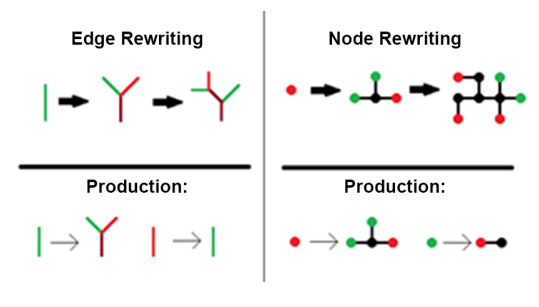
A diagram of a goal

AI-generated content may be incorrect.

(Parish & Müller, 2001). P 303. “Functions applied to a successor” !!!TODO!!!

### Turtle Rendering

Turtles can interpret the output of an L-System in any way a programmer desires, depending on how they implement it, however, two common conventions are edge rewriting and node rewriting.

p603 L-Systems and Procedural Generation of Virtual Game Maze Sceneries

Turtles are not limited to 2 dimensions but can in fact support any number of dimensions as long as every rotation operation required to navigate the space is implemented which for the case of 3 dimensions are yaw, pitch and roll.

Timed DOL-Systems can be used to produce animations of plant development. This works by assigning each symbol an age parameter that increases over time and a terminal age where any production rules that apply are executed. The L-System can then be queried for it’s state at a specific time which will provide a set of symbols all with ages that can be used to render a plant at any stage of development (Prusinkiewicz & Lindenmayer, The Algorithmic Beauty of Plants, 1990, pp. 133-139).

## Botanical Modelling

The Algorithmic Beauty of Plants (Prusinkiewicz & Lindenmayer, 1990) presents many examples of how many differing forms and versions of L-Systems can be used with the combination of a turtle renderer to produce beautiful visuals of:

* Grasses, flowers, ferns and bushes.
* Trees, fruits, seeds and nuts.
* Specific arrangements of lateral organs (leaves on a stem, composite flowers, etc).
* 3D cellular structures and microbiological phenomena.

### Combining L-Systems with genetic algorithms

Genetic algorithms use a virtual approximation of the process of evolution to select candidates using a fitness function and combine them similarly to how genes from both parents are combined in the child in nature. The initial parameters for each L-System are randomly generated and then the L-Systems are allowed to develop with a user defined fitness function being used to determine which plants will pass their genome on to the next generation. This continues until a user-defined number of generations are simulated at which point the user will have a population of plants to choose from (Fitch, Parslow, & Lundqvist , 2018). This is intended to produce more realistic plants by emulating the processes by which they are created in nature, however, as noted by the author while the initial results are promising significant work is still required to include evaluation criteria to provide a more realistic incubation environment.

### Semantic Death of Plants

The semantic death of plants can be emulated by using a parametric L-System. A parameter representing vitality and another representing time or age are assigned to each symbol and a maximum for each parameter is defined. On each iteration the vitality and age of each symbol is incremented, and the plant grows by creating new symbols with ages and vitalities beginning from their minimums. When a symbol reaches its maximum age, it is replaced with a symbol that represents a dying symbol with the same parameters and all child symbols will be replaced with dying symbols. Dying symbols have their vitality reduced by one each iteration until it reaches 0 whereupon there are 2 choices of rule whether the user wants the plant part to be removed or remain as a dead part (Castellanos, Ramos, & Ramos, 2014). This is a well thought out system with the only criticism being that it would make sense to implement it as a timed L-System to allow smooth animation rather than being limited to discrete time steps.

### Drawing L-Systems with a Free-Form Stroke

Making minor adjustments to how L-Systems look can be challenging because minor changes can have large effects on the overall appearance due to their fractal nature and, they become increasingly complex as the number of iterations increases and you add features to make them more flexible such as context-sensitive rules. A clever solution to allow an artist to have some control over the general shape of the final model is to make the L-System generate along a user defined free-form stroke that can be drawn. This works by interpreting the stroke drawn by the user as the central axis of the L-System or the axis along the direction the turtle is initially facing (Ijiri, Owada, & Igarashi, 2006).

## Architectural Modelling

### Generating Buildings for a Racing Game

In racing games there are usually buildings or other scenery in the background and although these assets do not have to be of the highest quality because they are not the focus of the game variation is important to keep the players engaged. L-Systems are used to create a footprint of a building and then a genetic algorithm is used to evolve this model where the fitness of candidate models is determined using anonymous participant ratings submitted through an online service. Selected models are then combined together in different configurations and the genetic algorithm is repeated for a number of iterations leading to novel combinations of shapes which mimic modern architecture (Yoon & Kim, 2016). L-Systems are only used here for generating a footprint with seemingly no real reason to use them over a simpler approach, but the overall outcome is interesting as it shows the potential of combining L-Systems with incremental user feedback to make better models.

### Geometric Modelling of Buildings

Generating a wide range of complex shapes as seen in building architecture is a difficult task which if performed entirely by an L-Systems may become un-manageable due to the vast array of symbols required. By replacing the terminal symbols, in other words the final symbols that have no rules that replace them, with functions that can generate geometry the overall shape can be decided by the L-System while the smaller geometric details can be designed in traditional functions. Furthermore other symbols can be replaced with functions and all functions mentioned do not have to be executed at the end of the re-writing process as may be intuitive but can be executed at any point allowing more flexibility (Marvie, Perret, & Bouatouch, 2005). The presented system is very robust and produces high quality examples with little room for criticism, some of the value of procedural generation is eliminated however as the functions that define the small details for generated buildings need to be created by hand.

### City Generation

When generating a full city many factors must be considered. Transportation infrastructure is designed to meet the needs of the city’s occupants, the layout of the city evolves over time leaving a mixture of old and new architectural designs intermingled, and even social and legal factors play a role in the final composition. CityEngine takes geographical statistical maps and generates a roadmap based on the space available and population density, the space between roads can then be subdivided into allotments and buildings generated. Extended L-Systems are used to create the road network and parametric stochastic L-Systems are used to generate building geometry (Parish & Müller, 2001). CityEngine was later acquired by ESRI, further developed, and renamed to ArcGIS CityEngine. CityEngine is still in active development today with Pascal Müller being a director in the company, it can be integrated into Unreal Engine (Environmental Systems Research Institute, Inc., 2025) giving it the potential to be used for games development in modern titles. This is a clear case of where L-Systems have been used very successfully to generate content that could be used in games, while no evidence has been found that it has been used in a production game this information is unlikely to be widely publicized.

## Environmental Modelling

### Underground Cave Systems

Generation of a network of underground caves is accomplished with the combination of L-Systems and cellular automata. A blank 2D bitmap has an L-Systems operate on it from each edge for a total of 4 L-Systems. The shape created is then operated on by cellular automata which fills some of the space between the L-Systems paths making larger chambers and fitting any additional constraints required. Several bitmaps are made in this way and placed at differing heights then a heightmap is applied to combine and smooth the cave chambers together into a larger cave system with multiple distinct levels that can be traversed by a character more easily increasing playability (Antoniuk & Rokita, 2016).

### River Deltas

Generation of river deltas is an interesting research area that does not appear to have had significant investment historically. There are clear similarities, visible even to a casual observer, between the patterns seen in river deltas and the output images of basic L-System used to generate tree like structures, which make L-Systems appear to be a good choice for this kind of application. Stochastic L-Systems are first used to generate a skeleton for the river delta and a graphical interpretation is created using a turtle. The surrounding land for the delta is generated by finding the smallest convex shape that can completely enclose the delta and then this shape goes through another algorithm to make it appear rougher and more natural. A coastline is generated with its overlap of the delta being determined based on whether the river or tide is more dominant, again being made to appear rough. Finally, a set of conditional Generative Adversarial Networks (cGAN) are used to generate the final surface image (Valencia-Rosado, Guzman-Zavaleta, & Starostenko, 2022).

### Forest Generation

L-Systems are well known for their ability to produce realistic tree models and as such the generation of entire forests may seem like an obvious target, however, rendering L-Systems through the use of a turtle is slow due to the number of operations performed and so does not scale well where many trees need to be rendered on every frame. A collection of ‘hero’ trees can be generated using a stochastic L-System, and geometry of their branches cached in graphics memory. This allows a graphics shader to be written that is given positions and ages of trees to be generated, it then randomly selects branches from these ‘hero’ trees to generate a reasonably distinct tree of the desired age in the desired location. Since the turtle rendering process is only done once it is significantly faster, pair this with the fact that most of the rendering is done on the GPU and can be done in parallel using instanced rendering allows around 10,000 trees to be rendered in a few seconds, which while not interactive is significantly faster than a simple CPU based implementation (Carrey, 2019).

## Level Generation

### Escape Room Levels

Level layouts in puzzle games like escape room games have the potential to be procedurally generated rather than having a designer create them by hand. L-Systems are used to generate unique levels for each playthrough while the puzzles are preset in a virtual reality environment in a paper included in the Journal of Advances in Information Technology (Yaswinski, Chelladurai, & Barot, 2024). L-Systems appear to be used to create a maze between the different set puzzles in this game in an attempt to prevent repeat playthroughs from being boring to players. Unfortunately, the only L-System presented in the paper is entirely deterministic and no mention is made of randomized axioms being used so it is unclear whether this is how the layout is modified or whether stochastic L-Systems are in fact being used.

### L-Systems as a Game Mechanic

L-Systems being used as a Game Mechanic is an interesting concept, although it is difficult to find substantial literature exploring the topic. One example is a system where a stochastic L-System is used to generate trees where there is a 10% chance of a branch generating as a poisonous branch while the other 90% of the time it will just be a regular branch. The poisonous branches release toxic fumes that will damage the player if they come too close, the player must navigate to some goal by travelling across the branches and the player has a mechanism by which they can cause the L-System to grow. Growing the tree is necessary to reach the goal but comes with risk as branches that generate may not be safe to use, additionally branches have a chance of falling off and continuing to grow as a separate tree from where they land (Fornander, 2013). It is not clear how the mechanism which allows the player to control the growth of the L-Systems functions and the L-Systems used are very simple leaving the actual shape of the tree to a different system and only defining how many branches a tree has and what types of branch they are. Excluding these minor flaws the paper exposes an interesting use case not commonly seen in the literature for L-Systems.

## Dialogue and Quest Generation

Dialogue and other text-based systems seem to be an obvious target for L-Systems since they fundamentally work on strings of text, however, little evidence was found in the literature exploring this.

Tracery

## Research Gap

What doesn’t exist or is underdeveloped? I can’t say it doesn’t exist though maybe just not prevalent.

Textures

Audio

Terrain Generation

Maze Generation

Dialogues and Quests

## Conclusion?

How do I use this going forwards.

Architecture and plants well developed. Audio outside knowledge

# Method

This chapter should explain in detail how you carried out your project. This includes several aspects, although not all projects will need all of these. Use your judgement to decide which of these are appropriate to your project. Add anything I may have left out, if you think it’s necessary.

1. the development methodology you have chosen (waterfall, agile,…)
2. testing method
3. how you gathered requirements
4. the tools (languages, software, platforms, libraries etc.) you chose
5. research methods used, if there is a “research” aspect to your project, such as conducting a usability study of your program or web site, checking the effectiveness of educational tools or environments etc.

As well as explaining the method/tool etc., you should also provide some argument justifying your choice. In the best dissertations, this will include consideration of the alternatives.

## Development Methodology

The project does not have a clearly defined destination and the timescales of individual tasks was not predictable. Therefore the simple waterfall method where timelines and tasks are planned then plotted out ahead of time doesn’t work particularly well. Instead, an iterative and incremental development methodology will be used as these provide more flexibility and are “associated with many successful large projects and recommended by standards boards” (Larman & Basili, 2003).

A specific named methodology will not be used as these are typically designed for teams creating commercial software instead of lone developers without involved clients. The chosen methodology will use a Kanban board to track what tasks are pending, in progress and complete, and Scrum inspired sprints will be used where tasks are chosen each week based on a ranking of their estimated value. The pending column of the Kanban board will serve as this ranking and will be updated each week reconsidering the value of all the tasks in the list as well as uncompleted tasks from the previous sprint which may be moved to pending if their estimated value drops below other tasks. New tasks will be added to the pending column as they are formulated. I have used this methodology in past projects to good effect and so am confident it will be effective here.

## Design Methodology

Find concepts in papers or come up with concepts, hypothesize how these concepts could work, test hypothesis, repeat until working prototype is created or an impasse is reached.

## Testing Methodology

Since not product just for exploring use cases no usability just developer regression testing as features are created to ensure nothing is accidentally broken as new features are implemented.

## Tools

Options, why one was selected Unreal, Unity, roll your own..

### Unreal Engine

Using a game engine will significantly reduce the workload while producing better results compared to learning about and then writing all the rendering code from scratch which will allow more focus to be given to the main task of researching and developing L-System based algorithms.

Unreal engine 5 has been chosen due to its extensive use in the industry with Epic Games’ own titles such as Fortnite, Rocket League and Fall Guys as well as big hitters like Kingdom Hearts III (Kayser, 2020) and Hogwarts Legacy (Crecente, 2023).

There are other options such as Unity and Godot. In this case it would not make much difference which of these options was used due to the student having no experience with any of them, however, they do have extensive experience with C++ which is what Unreal Engine is written in and a pre-existing desire to learn Unreal Engine.

C++ will be used to create the procedural generation code as it easily integrates with Unreal Engine, and it is a high-performance low-level language well-suited to producing code that may run under performance critical conditions such as during initialization or within tight game loops.

# Results

This chapter is where you present the results of your project and explain what you achieved.

If you were collecting data for some reason then this is where you could present it. If you built something then this is where you would describe it, including a discussion of how far you got with the original plan.

If you built something and tested it (and you should have…) then this is where you would present your test results.

Actual data u (at least if it’s large) and code listings should be placed in an appendix and referred to from here. This chapter would then discuss those maybe using descriptive statistics.

## Walkthrough of Solution

## Testing Tables

## Summary

No interpreting or conclusions, mention unexplored or halted directions (Dialogue / Quests).

# Analysis

This chapter is for a critical evaluation of the results obtained and of the project in general.

If you collected data, this is where you should interpret it.

This is also the place for an evaluation of the project, of its methods and outputs. (Product **and** process.) Markers will expect a high level of reflection, analysis, thought, perception, insight… This is where you are expected to demonstrate high level academic skills. Going back over your project log book (you remembered to keep one, right?) will be helpful here.

## Fulfilment of Hypothesis

## Versus Existing Literature

## Compared to Other Techniques

## Implications

Shows that there is room for further development of L-Systems and probably untapped potential. Used as an aid to artists is fairly interesting with the way it can be split into stages and the concept of visual rules rather than text.

# Conclusions & Future Work

What it says on the tin.

You might want to talk about interesting/worthwhile/possible future directions for this project. Perhaps put that in a section called “Further work”.

## Research Question / Hypothesis

## Conclusions

Maybe list your original objective(s) and aims then add a sentence or two explaining how (if!) you met them.

## Future Work

* Dialogue, Quests, Textures
* Visual Rules – this shape replaces that shape.
* Terrain that allows deforming in 3D to create caves rather than a simple heightmap
* Maze turned into a full game where something is done about the player being inside the maze when it regenerates.
* Direct interpretation of the L-System output as a grid of values or equivalent.
* Integration of more L-System types (parametric, etc) and swap-in / out functionality for both turtles and l-system types.

# Critical Reflection & PDP

Write about how the project has helped you develop both personally and professionally. Include any ethical issues that have arisen and discuss how they have been resolved. For example, maybe before the course you were awful at planning and management and that has changed. Or maybe you have experienced and overcome adversity during the course? Please don't put any private/personal details in here but loosely refer to the issue and how you managed to overcome it to help you develop.

Provide a convincing discussion of personal and professional learning that has taken place and how this learning will influence your future direction. This should also include personal development that has taken place, with examples to back up your discussion wherever possible.

Where appropriate, clearly and convincingly discuss how ethical and professional issues have influenced decisions taken throughout the project.

## Challenges

Unreal Engine

Time – self doubt (surpass your limits)

Limited relevant research

## Skills Acquired!?

Unreal Engine is not a complete unknown. Research skills?

## Future Goals

Reaffirmed interest in game development. L-systems write and publish a paper exploring more in depth in attempt to convince more people to research and consider it an option.

## Lessons Learned (Conclusion)

Unreal

# Bibliography

All your references should be listed here, following the UHI Harvard Referencing standard. You’ve been doing this for a while now, so there is no excuse for doing it wrong. Remember to list the references in alphabetical order of the first author’s last name. **Refworks** may be a good solution here.

# Appendices

## Appendix A: Source Code Listing

## Appendix B: Exemplars

## Appendix C: Test Log

These are like chapters, but the numbering restarts: Appendix A, Appendix B,...

Each should start on a new page.

[I don’t know how to restart the numbering and get these included in the TOC, but you’re all expert IT users so it shouldn’t be a problem for you. If you don’t know how to do it, you could always Google it! I’m sure we’re not the first people to want to create a dissertation with appendices in Microsoft Word. Headings for appendices should be like this: Appendix A: Source code listing, Appendix B: Test Plan, Appendix C: Test Log, etc.]