Evaluating the Effect of Procedural Level Generation on Movement-Based Gameplay Mechanics

Harry Findlay

BSc (Hon) Computer Game Applications Development, 2025

School of Design and Informatics

Abertay University

Contents

Table of Figures

Table of Tables

Acknowledgements

I would like to thank my Wife, Dana and my kids, Roman and August for their continuous support throughout my time at university.

I would also like to thank my advisor, Dr. Naman Merchant. Naman has been so flexible yet available and helpful along every step of the way and his advice has been greatly helpful.

Abstract

Context:

Procedural content generation (PCG) is widely used throughout the games industry, covering many genres, the type of content generated and the uses of content that is generated. However, many genres of games seem to avoid the use of PCG as, in many cases, many different genres are better suited to hand-crafted worlds. The goal of this study is to investigate the level-building side of PCG and how this relates to movement-based gameplay mechanics such as wall-running and sliding.

Aim:

To investigate, compare and evaluate the differences between static-made game worlds and procedurally generated game worlds. The focus will be on the how movement-based and movement-enhancing gameplay mechanics relate and adapt to the game-space that they are in. Should developers and companies be more open to the idea and use of procedurally generated levels and worlds in movement-based video games.

Method:

An algorithm will be developed (or tool used) to create a very simple procedurally generated level. All tools can be found via the epic store or unreal engine forums/documentation. There will be a focus on wall-like structures with a secondary focus on building-style structures, being more box-shaped. This environment would then be tested against a static environment and compared in certain areas of interest to the research. I then aim to have a group of individuals of no specific set of skills to test PLG against a static level and fill out a questionnaire.

Results:

I expect that PLG would be a fun addition to movement-based video games, however, would be too unstable in its generation to properly compliment the mechanics. I hypothesize that PLG would work for sub-genres of the movement-based video games such as movement-based shooters (Titanfall 2, Respawn Entertainment, (2016)) but be less effective in core movement-based games such as Mirrors Edge DICE (2008).

Conclusion:

The conclusion is determining whether procedural generation is an appropriate and effective alternative to level/map creation when using movement-based gameplay mechanics. This paper will be investigating Wave Function Collapse specifically, and whether this is a valid alternative for developers looking to save time on level design, time and development costs.

Abbreviations, Symbols and Notation

Procedurally Content Generation – PCG

Procedurally Generated Level(s) – PGL

Procedural Level Generation – PLG

Wave Function Collapse – WFC

UE – Unreal Engine / Unreal Engine 5

1. Introduction

1.1 What is Procedural Content Generation

Procedural Content Generation (PCG) is the process of algorithmically creating content using a set of pre-defined rules and processes created by a developer.

PCG has been a programming technique since the 1980's in, not just video games, but in many areas of computer programming to create content at run-time. Generally, PCG follows rules or guidelines when generating content as to ensure that the content fits with the profile of the overall application or video game. Some strong examples of early uses of PCG in video games include "Beneath Apple Manor" (The Software Factory, (1978)) and "Rogue", (originally developed by Michael Toy and Glenn Wichman ((1980)).

PCG was originally developed to produce semi-random content for video games or applications in a fast and low-cost manner, at runtime. PCG was also used in other ways, some uses being: ways to bypass system restrictions and to compress file sizes on older, lower memory computers[1].

However, modern uses of PCG are very different when compared to the historic uses. In video games, modern uses of PCG is generally aimed at enhancing the video game's content, replayability and general purpose, rather than the earlier uses of combatting system restrictions and file size issues that plagued older systems and computers. Another example of modern used of PCG in video games is procedural level generation (PLG), to which is where this research paper will be focused. Levels, structures, terrain and other object-based topics can be semi-randomly generated at run-time to allow for replayability and to offer the video game with a wider arsenal of content to be used. Other, none-related areas of PCG include procedurally generated stories, quests, weather patterns, designs and in-game surfaces, and in some cases, even gameplay mechanics.

PCG, however, does come with negatives as well, especially if poorly executed. Some examples may include: PCG stories lacking depth and purpose. Some PCG levels can feel empty and meaningless, many PCG creatures and NPC's (non-player characters) may appear illogical or too random to make sense. An example of this is the idea of 'PCG Paradox'. This is the paradox where PCG can generate an infinite amount of seemingly unique items of levels. However, despite this infinite capability, all content seems to lack individuality and can feel repetitive and boring. [2]

[1] Kenny. (2021) "Procedural Generation: An Overview". Available at: https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41 (Accessed: 2 Nov. 2024).
[2] Schier, G. (2015) "Pros and Cons of Procedural Level Generation". Available at: https://schier.co/blog/pros-and-cons-of-procedural-level-generation (Accessed: 17 Jan 2025).

1.2 Procedural Generation in Movement-Based Video Games

The development of video games has become one of the world's largest and most successful industries, and with this, comes bigger and better computers capable of processing much more in a shorter span of time. With these improvements, PCG has been able to really thrive and bloom. This can come in a multitude of different forms ranging from entire worlds being generated to a complete arsenal of weapons, armour, characters or in some cases, entire solar systems or galaxies. Some examples to look at when investigating the expanding capabilities of PCG in video games: Valheim[3] Coffee Stain Studios (2021) with its early access release in 2021 and its official release in 2023. Valheim is a flawless example of large-scale world generation featuring different biomes, enemies, bosses and more. Another example is No Man's Sky Hello Games (2016)[4], though No Man's Sky had a shaky release regarding player experience, upon release, No Man's Sky used PCG to generate not one solar system, but 4.3 trillion[5]. No Man's Sky has put in a lot of work since release though, and has delivered the content and by extension, the PCG that was promised by Hello Games.[6] These, however, are video games that do not have an emphasis or primary gameplay loop related to movement-based gameplay mechanics.

Movement-based video games as a genre in whole tends to neglect the use of procedural level generation (PLG), and this is usually for a multitude of reasons. Titanfall 2 Respawn Entertainment (2016) is an example of a successful movement-based video games with an entire gameplay loop with strong ties to movement-based gameplay mechanics yet has absolutely no PCG content throughout the entire product. The same goes for other successful movement-based gameplay mechanics including Call of Duty Black Ops 3 Treyarch (2015) and Mirrors Edge DICE (2008), again, both extremely successful movement-based video games with absolutely no PCG content or levels. To add perspective, Call of Duty Black Ops 3 generated over
$550 million in revenue during its first three days of release, and Mirrors Edge generated approximately $125 million in revenue.

The reluctancy to use PCG in movement-based video games is very clear, and, as stated prior, this is for some obvious reasons. Angel, J. (2014)[7] shows that making well-flowing maps for parkour or movement-based video games requires "parkour vision" as these levels

[3] Coffee Stain Studios (2023) Valheim [Video Game]

[4] Hello Games (2016) No Man's Sky [Video Game]

[5] Contributors to No Man's Sky Wiki (no date) *Galaxy*, *No Man's Sky Wiki*. Available at: https://nomanssky.fandom.com/wiki/Galaxy (Accessed: 30 April 2025).

[6] Mahmoud, M. (2024) "The Redemption of No Man's Sky is INCOMPARABLE". Available at: https://www.kitguru.net/gaming/mustafa-mahmoud/kitguru-games-the-redemption-of-no-mans-sky-is-incomparable/ (Accessed 11 Nov. 2024)

[7] Angel, J.(2014). Game Maps: Parkour Vision and Urban Rel
ations. In: Schiller, G., Rubidge, S.
(eds) Choreographic Dwellings. New World Choreographies
. Palgrave Macmillan, London. https://doi.org/10.1057/9781
137385673_11

or game worlds need fit well enough together to grant a smooth and enjoyable user experience. This source sparked the start of this research as the goal here is to develop a prototype PLG algorithm and a series of movement-based gameplay mechanics and have users test this and relay their opinions and experiences. If done correctly, the hypothesis is that PLG will, not only, reduce development time and costs, but can add replayability, fluidity and overall enjoyability to movement-based video games going forward. This includes triple A industry-level products as well as indie-level development projects and video games.

Some video games lean very heavily into PCG for example, Spore (Maxis (2008))[8] that uses PCG on multiple scales ranging from creature design and animations, complete civilisations, planetary ecosystems, galactic scale exploration and even the games music. Another example being Borderlands (Gearbox Software (2009))[9] and its use of PCG to generate weapons with various elemental damage types, parts – that change the stats and behaviour of the weapon, and visuals, weapon sights included. Again, both are examples of PCG in video games however, yet again, both examples are not movement-based video games, furthering the evidence of PCG being popular, but overall neglected in movement-based video games.

Many movement-based video games do use PCG and despite this being on a smaller scale, the movement-based video games that do generally use PCG tend to be two dimensional (2D). This is usually as 2D games are generally considered easier to create flowing maps and levels in a way that is entertaining and well-flowing. Some examples including Prince of Persia, Jordan Mechner (1989), and Spelunky Mossmouth (2008).

The popular use of PCG and PLG in two dimensional games is a very wide topic however, three dimensional games tend to shy away from use of PLG and PCG. There are three dimensional games that do make use of PCG and PLG however, these are fewer and further between when compared to two dimensional games and such games and environments do not tend to make use of movement-based gameplay mechanics. This notice is what ignited the start of the research found throughout the paper you are reading now.

1.3 Aim

This project aims to research the implementation of Procedural Level Generation (PLG) in Movement-based video games and evaluate whether PLG is reliable and suitable enough to be used in future instalments into the movement-based video games genre.

1.4 Research Question

Does procedural level generation positively affect user experience of movement-based gameplay?

[8] Maxis (2008) Spore [Video Game]
[9] Gearbox Software (2009) Borderlands [Video Game]

1.5 Research Question Critical Analysis

PCG is an evolutionary technique and step in game development and is avoided in many situations when it should be embraced and enhanced via skilled developers and trial and error. The consistency of PCG in movement-based video games is very important as the play-worlds and levels require a flawless flow and path to keep the movement-based gameplay mechanics intact and enjoyable. These traits introduce a wide range of difficulties and challenges to maintain a smooth player experience and keep the generated worlds quality and worth-while. An example of this would be the use of WFC regarding the models and rules used to restrict the algorithm to generate nodes in a specific set of ways. If the rules and models are of poor quality or, by extension, not set up properly or of a strong enough quality, then the generation of the environment or level will contain flaws. Some flaws may include inconsistencies including potentially negative features such as incompletable, illogical or un-enjoyable flowing environments. An example being the pre-mentioned game No Man's Sky. On launch, No Man's Sky was the prime example of "PCG Paradox", where all planets felt and looked similar, lifeless and objectiveless with no goal and lacked unique aspects to draw the player in and have them spend time in each environment. Another example would be 7 Days to Die created by the Fun Pimps (2013)[10] where roads would run into rivers and would contain unusable geometry.

This raises the question, can PCG be used within movement-based video games to develop complete and playable levels/maps, in a smooth and effective manner, whilst able to keep the fluidity and enjoyability of movement-based gameplay mechanics.

1.6 Objectives

The primary objectives of this research project include:

1. To study the uses of procedural level generation in movement-based video games and investigate how movement-based gameplay mechanics, such as wall running, will respond to the non-manually sculpted game worlds.

2. To study how the use of movement-based gameplay mechanics can vary in different or procedurally generated environments.

3. Investigate the development and uses of level-based procedural generation tools and algorithms inside of Unity Engine.

[10] Vote4Wes. (2019) "7 Days To Die Review". Available at: https://thecouchcoopcouple.home.blog/2019/07/27/7-days-to-die-review/ (Accessed: 3 Feb. 2025).

2.0 Literature Review

For this project to be successful and accurate, external sources were investigated and analysed. This research included Wave Function Collapse (WFC) as well as other methods of PCG, all of which were related to PLG. The following sub sections review these sources and their relevance to the project.

2.1 Procedural Generation Techniques

2.1.1 Wave Function Collapse and Noise Techniques

Büyükşar et al (2024) [11]started by introducing and explaining a range of PCG techniques and discussed their strengths and weaknesses. Such PCG techniques that were discussed included, Particle Swarm Optimization (PSO). PSO was a strong technique due to its fine-grained control offering strong results. PSO is an optimisation-based algorithm that was inspired of the natural and social behaviour of swarms such as bees or a flock of birds. PSO is used in PCG to calculate the best parameters for generating content such as levels, music or anything that has a pattern. PSO works as a swarm of "particles" which represents a solution, or parameter(s), which move through the space where their position is influenced by its own best-known positions, the position of its neighbour and the velocity. This data aims to lead the swarm towards the optimal solution for the generation. Digital Elevation Map (DEM) was also discussed, explaining that this uses a 2D grid system paired with elevation values in order to achieve efficient and effective 2D PLG results. Erosion Based Simulation was expanded upon, explaining that this is a viable approach for generating terrain that resembles actual landscapes, however, struggles to stitch together neighbouring tiles, which can result in unnatural aesthetics. Noise techniques, in general, were discussed, explaining that, despite being a valid and efficient choice in some cases, noise generated outcomes are very efficient at generating terrain, however, lack depth when producing features and can create irregular patterns. WFC is then introduced and explained to be a very reliable approach to PLG but can be high maintenance due to requiring pre-made assets/models and can be expensive due to the overall resources needed. The paper then proceeds to analyse and explore their research and findings regarding a hybrid approach where WFC and noise are used to create a two-in-one system, where the noise generates the terrain and the WFC generates the textures and assets. This hybrid approach was generally considered a success, producing diverse and interesting maps. Büyükşar's research is overall a very strong source due to the un-biased and effective approach of discussing the strengths and weaknesses of various PCG techniques and their uses.

[11] Büyükşar, O., Yıldız, D. and Demirci, S. (2024) Enhancing wave function collapse algorithm for procedural map generation problem, Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi. Available at: https://dergipark.org.tr/en/pub/ngumuh/issue/86158/1361413 (Accessed: 13 October 2024)

2.1.2 Realtime Procedural Terrain Generation

Olsen's (2004) aim within this literature is to investigate and evaluate various types of erosion-based procedural generation techniques in computer games[12]. With the increase of general computer processing power, erosion-based techniques are a very solid and fast technique it used, giving near run-time results when emphasising speed over quality. Two types of erosion algorithm are used within this research thermal erosion and hydraulic erosion. These were first described by Ken Musgrave et al (1989) and has been described as a foundation to which various improvements have been suggested and made. An overview, thermal erosion simulates the breaking of material, and how said material would slide down a slope and rest at the bottom. Hydraulic erosion is the simulation of the effects in which flowing water has to terrain and dissolving materials, usually transforming the position of the material and leaving it elsewhere. These erosion types were also paired with a different type of PCG algorithm, which yielded some very interesting and powerful results. This secondary PCG technique is Voronoi Diagrams, and this algorithm focuses on the procedural generation of textures described by Steven Worley. Overall, this approach and research produced sturdy and robust results and is an interesting approach to level-generation. This literature was used to gather background and general knowledge based on PCG however, none of the content discussed was directly used within this study. This literature was investigated when trying to evaluate the best form of PLG for the prototype developed for this study.

2.1.3 Time-Space Wave Function Collapse

Facey (2024) introduces the process and goals of wave function collapse[13]. It is explained that wave function collapse is a method of PCG which is generally easy and fast to implement as well as being usually low maintenance and setup if done correctly. It is discussed, however, that WFC can be slow when in-process and require a designer to sit and implement the constraints, which act as 'rules' for the generation to follow when producing generated content. Additionally, Facey and Cooper's approach add an additional feature to the traditional WFC, adding time (T) as a factor when generating maps. Normally, WFC uses either a 2D or 3D grid (usually consisting of height, width and length, if 3D), splitting each section of the grid into nodes, however the approach used within this source expands this, turning the nodes into 'space-time blocks'. The point of adding time to the, generally considered, stable algorithm, is because instead of their algorithm only focusing on, only the level – adding time adds an additional focus: the steps to solve the generation. These generated maps were then tested against a series of gameplay mechanics: Maze, Field and Sokoban. It was found that Sokoban was extremely reliable, generating successful maps 70% vs. 41%, and accurate when used as a testing environment, however, took over 100 times longer to generate a single level – averaging at 51 minutes, 10 seconds. Comparing this to Field that took only 27 seconds, however, was only able to generate a successful map (52% - 64% of the time). Finally, Maze, when compared to Field, it was less

[12] Olsen, J. (2004) 'Realtime Procedural Terrain Generation', Department of Mathematics and Computer Science (IMADA) University of Southern Denmark.

[13] Facey, K., and Cooper, S. (2024) 'Toward Space-Time WaveFunctionCollapse for Level and Solution Generation', Khoury College of Computer Science.

likely to return a successful map and was also found to take slightly longer to do so as well, despite the Maze map being less than half the size of Field's map. Overall, it was found that the generation, as a whole, was very successful. This was due to the fact that the generation was easy, fast, produced content visually like the data passed in and, lastly, the levels were completable. It was found, however, that additional game data was required to be passed in as global constraints which, in turn, increased the complexity and time of the setup of this PCG technique. This did prove the WFC to be a reliable algorithm.

2.2.1 In-Depth Wall Running Mechanic

This source is a very strong source when exploring the background and mechanical side of strong wall running mechanics – which translates to other movement-based gameplay mechanics[14]. Throughout pages 363 to 375, the chapter starts off with a general guide to player movement mechanics in a three-dimensional space, with a general explanation of WASD movement, jumping and wall running (or "push off" mechanics). Momentum is then introduced and explored, explaining that many movement-based gameplay mechanics work with a powerful emphasis on momentum as a baseline for fluidity. This, however, is accompanied with many complexities and difficulties as well. Additionally, it is explained that the movement, in a three-dimensional environment, needs to be local to the player – unlike in many two-dimensional games, and this is especially important for the process of the wall running and general movement-based gameplay mechanics as the player needs to be able to easily guide themselves through obstacles as well as keep the fluidity of the gameplay. The source then teaches the reader about Vector 3 coordinates, and how they are important to the player transform within the context, using this as an introduction to vectors in general, and then advancing on to explaining magnitude. The reason behind why the vector magnitude is important is because this is the process of converting a vector into a direction, and the normalised direction is required for smooth and reliable movement-based gameplay mechanics. Additionally, the source talks about how the behaviour of the movement-based gameplay mechanics change depending on whether the user is on-ground or in-air. When airborne, the player direction is then locked to a global state, disabling the player's ability to turn based on camera (look) direction, which is then re-enabled when grounded. The chapter then C# Unity script and a briefing of what the methods called within the script do. This, overall, is a very detailed and effective approach to wall running and had a strong influence in the mechanics used within this research.

[14] Hardman, C. (2024). Advanced 3D Movement. In: Game Programming with Unity and C#. Apress, Berkeley, CA. Available at: https://doi.org/10.1007/978-1-4842-9720-9_38 (Accessed: 14 Dec. 2024)

2.2.2 Momentum-based Gameplay in FPS Games

Within this source[15], there was a heavy emphasis on movement-based gameplay mechanics via a momentum-based movement system. Within this research piece, Unity was utilised to craft a three-dimensional first-person shooter (FPS) game. This game made use of a first-person perspective as, like many other games, first person perspective allows for immersion and, in some cases, a better relationship with certain gameplay mechanics (such as movement-based gameplay mechanics). This also applies to the research conducted here. Though, this game did not contain any advanced mechanics such as wall running or dashing, it did contain a slide mechanic which allowed the player to maintain their momentum – pairing this with the ability to shoot and a series of enemies which attack the player. Both the player and enemies have health points and so can be 'killed' if enough damage is done to the target. Although the movement-based gameplay mechanics were polished and well-functioning, many of the other mechanics (such as shooting) were unpolished and missing some aspects such as visuals or audio queues. Another example being the fact enemies only had melee-style damage outputs. This negatively affected some of the feedback gathered within this study, however, it was generally stated that the movement-based gameplay mechanics were well received and 'welcome feature based on the optional feedback'. Some challenges were stated throughout the development of this study's game, some examples including: the idea of the setting and design of the game, how the health system should be implements and how it should function, and how the shooting to function as a whole – to which a raycast approach was elected. Zhu et al goes into much detail regarding many aspects of the game, such as the fire rate of the gun and enemy spawning. After two experiments: the first being the testing of the movement within the game, and the second experiment being aimed at the overall game experience. It was found, as mentioned, that the movement was very well received and welcomed however, the lacking and less polished areas, such as shooting, affected the results in a negative manner.

[15] Zhu, C., and Zhang, Y. 'A First-Person Game Designed To Educate And Aid The Player Movement Implementation', Beckman High School, University of California. Available at: https://csitcp.org/paper/13/132csit03.pdf (Accessed: 15 Dec. 2024)

3.0 Methodology

3.1 Overview

To investigate the research question fully, it was mandatory to create a prototype application that contained a static, man-made level and had the ability to generate a multitude of stable and reliable environments. The primary purpose of the application was to gather user-driven data in relation so the developed movement-based gameplay mechanics can be utilised and compared within a static and generated environment. This application made use of Unity 3D and its C# scripting to supply the users with movement-based gameplay mechanics and a series of grid-based generated levels. The user then controlled a playable character (PC) with access to all movement-based gameplay mechanics and were tasked with navigating the level(s) with the aim of finding one of two hidden objectives. One objective was a simple objective to give less-experienced users a means to complete the level, and the second contained a platform-based objective that was raised and required the user to make use of the more advanced movement-based gameplay mechanics to reach and trigger this objective. The overall aim is to gather real-player statistics which will be used to investigate and evaluate the use of PGL in movement-based video games.

3.2 Application

3.2.1 Application Design

The PCG within the prototype application was designed the traditional use of WFC. This means that the generation is calculated via a grid that runs along the X and Z axis and places nodes with attached models depending on the neighbouring and pre-existing nodes (propagation). [16]Each grid tile contains a node, as previously mentioned, and each node contains a series of data within them. The data includes: a node name, a 3D model which represents the in-world asset (such as a wall, corner, tree or empty space), and a series of rules which drive the WFC, only allowing certain nodes/models to be places next to other, specific nodes. Once all grid slots have been assigned a node, the level grid will 'collapse' and produce a level.

3.3 Node and Model Design

3.3.1 Node Uses and Overall WFC Generation

The node is arguably the most important component of the WFC. This is due to the fact that the nodes are slotted into the grid when the world is generating and is the sole container for all data out with the main generating algorithm. Though, without said nodes, the main algorithm is rendered useless.

As mentioned in the overview, the node contains a multitude of different data. The first field contained within the scriptable object that is a node, is the node name. The node name is simply an identifier for the node, allowing the client/user to identify the node being used. The

[16] Boris. (2020). "Wave Function Collapse Explained" Available at: https://www.boristhebrave.com/2020/04/13/wave-function-collapse-explained/ (Accessed: 5 Mar. 2025)

second piece of data contained within the node is the prefab game-object. This prefab is used as the physical body of the node, containing all in-world assets such as the ground, walls and extras (such as trees) which, by extension, contains all colliders and any interactive-based properties which the user may need to play the prototype. Lastly, the neighbouring relationships which act as the sole rules that the generation follows. The node contains relationships for neighbours north, east, south and west of the current node in question, and are labelled in-script as up, down, left and right.

The main algorithm starts off by setting up the fields and data required, especially the grid size which are exposed to the game engine as public fields. From this point, the Collapse() method is called – this is the main functionality behind the generation. The algorithm starts by iterating through the entire grid, cell by cell, and records the cells that still need to be assigned nodes and collapsed. Additionally, a list of potential nodes for each un-collapsed cell is initialised which is one of the main drivers of the algorithm's propagation. From here, the iteration of the propagation and collapsing initiates, starting by ensuring the cell being checked is valid and within range of the grid before adding the cells neighbours to a list for later generation. The neighbour each node is then initialised in a recursive fashion, and each neighbour's propagation is then pruned. Pruning is the action of removing the possibility of generating an incompatible neighbour node – driven by the rules within each node.

3.3.2 Propagation

Propagation is a fundamental aspect of the WFC algorithm. Propagation is what drives the generation of an already generated node's neighbouring nodes (offsets). The propagation of a constantly updating possibility of what a neighbouring node could be. For example, if we have nodes X, Y and Z where X and Z as well as Y and Z are compatible – this means that the algorithm will actively neglect to generate nodes X and Y beside one and other, as per stated in the node rules. As each node is generated, the act of pruning occurs to keep the propagation up-to-date and reliable. To summarise, propagation is the act of updating the neighbouring possibilities of generated nodes, and pruning is the active removal of incompatible nodes.

3.3.3 Prefabs and Models

The main component of the WFC from the player's perspective is the models to which are attached to the nodes. The model is the visible and interactive property of the working nodes and is what the player will have direct contact with. Within this prototype, there are a series of different models used, and these models are formatted as prefabs. Prefabs are a reusable and preconfigured game object template that is saved as an asset to be used within a Unity project. Prefabs can take on many forms ranging from enemy templates, player templates, system manager templates or, in the case of this project, objectives and in-world scenery including walls, floors and extras such as trees. The prototype's prefabs take form of various world pieces, usually in the form of a floor and a type of wall. The wall types include straight walls (horizontal and vertical), parallel walls to encourage wall running and corner walls. All wall types include flipped variants as well, this way the algorithm was made easier, excluding a rotating algorithm with the addition of some extra node rules.

3.4 Movement-Based Gameplay Mechanics Design

3.4.1 Wall Running and Sliding

Another focus of this research is the movement-based gameplay mechanics, as they will be closely paired with the PLG to research the relationship between them. The first of the movement-based gameplay mechanics is the wall running mechanic, arguably the strongest of the gameplay mechanics and the mechanic with the strongest relationship the playable environment. The wall running mechanic has a couple of strongly defining features and provides the player with an alternative means of navigating the environment – and this goes for all games and applications that contains this mechanic, this prototype included.

The wall running mechanic was built up from Dani's movement script[17] as this movement considers momentum, interactivity and is a strong basis for building on top of. As the base movement script was split into 2 sections, the camera and the player body object, this allowed for some fine tuning and manipulation when adding additional features. For example, the wall running mechanic makes use of the camera being segregated from the main players body (being a child object) and allowed for camera tilting when wall mounted.

Starting with the basic movement script, there are many layers of depth to the scripting. Firstly, the script casts a series of checks to investigate whether the movement should behave a certain way, or some cases, not at all. These checks being: can the player jump and is the player currently standing on the ground layer. Behind the Movement() method, the player is applied a force downwards to add a more responsive gravity to the movement before calculating the magnitude of the player and setting each magnitude for the X and Y axis. This magnitude is then utilised alongside the player's X and Y position to calculate the friction of the player's movement, this being used to help the player feel realistic when traveling fast as well as being well-flowing. Once the movement has been applied, the 'is player grounded', 'is player jumping' and 'is player crouching' are checked to ensure the movement doesn't need alteration before clamping the player's max speed and then applying the forces to move the player accordingly.

Player jump is a simple implementation, simply checking if the player is standing on a surface and can jump before adding an upwards force to the player's rigid body and resetting the jump cooldown. Where the jump gets complicated is an addition to the script – where wall running functionality is implemented. The wall running implementation within the jump method checks, firstly, to ensure the player is not already wall running, and if it is found that they are not, the player is attached to the wall via a conditional check (checking for the direction of the wall to the players local position). Once attached, the player gains a forward force to add to the momentum-based mechanic.

The wall running component is where a majority of the functionality for this movement-based gameplay mechanics is contained. Within this script, the wall is checked for using a left and right raycast and is attached or detached according to the result of the raycast finding a wall or not. Additionally, the input is read, ensuring that the correct inputs are inserted to connect the player to the wall. Lastly, the two methods responsible for starting and stopping the wall running

[17] D, Devy. (2019) FPS 'Movement Rigidbody'. Available at: https://github.com/DaniDevy/FPS_Movement_Rigidbody (Accessed 22 Dec. 2024)

which either ensures the player doesn't exceed the clamped max speed, a forward force is applied, and the player is attached to the wall via a directional force towards to the wall – or the player dismounts the wall and normal gravity is reapplied to the player.

The player also has access to a slide mechanic which can be used to 'bunny hop' and if done correctly, can grant speed and maintain momentum.

3.4.2 Objective

The objective is the user's main objective within the level and plays a crucial part in the user experience of each level. The objective gives the user a goal for when they are within the level, allowing them to navigate and use the movement-based gameplay mechanics with a goal, avoiding having an aimless environment.

There are two main objective types: an on-ground objective and a parkour-accessible objective.

The on-ground objective is for less experienced users who may struggle to use the movement-based gameplay mechanic effectively for whatever reason. This was added to ensure every user, despite their level of experience and exposure to video games, has a means of completing the level. The parkour-accessible objective was added as a means for more experienced player with prior experience of gaming. It was also added to encourage players of all levels of experience to utilise the movement-based gameplay mechanics to complete each level. Without this additional objective type, users would have little-to-no incentive to use the movement-based gameplay mechanics, which would directly oppose the research goal.

After the objective's trigger is activated through direct contact with the user, the user is moved to a win-screen, with options to replay or access the main menu of the application. All levels generated and complete will be tracked, separately and this data is persistent across scenes. The level tracking is automatically increased by the trigger activation of one of the objectives, increasing the level complete tally before loading the win-screen. The reason behind why levels generated and levels complete are tracked separately is due to the possibility of a level being incompletable which, in turn, would prevent the user from ever reaching either objective.

3.5 User Testing

To address the research question, a series of testers were obtained and tasked with testing the application to gather data around the research topic. The goal was to obtain both qualitative and quantitative data in the form of prototype experience-based from two structured questionaries, a gameplay experience orientated questionnaire and the being technical and PCG orientated.

The testing process will have a series of testers playing through two levels, a static, man-made level and a, or a series of, procedurally generated levels. Throughout the levels the player must navigate through with the goal of seeking one of two objectives. One objective was a sphere simply sat on the ground, intended to give less experienced testers a means to completing the level whereas the second objective was raised onto a floating platform, warranting the use of the mode advanced movement-based gameplay mechanics to access, for more experienced testers.

The prototype was developed in Unity Engine as this allowed for easy and fluid development and naturally contained more source material when compared to other, niche, game engines such as unreal Engine.

The application was made to be as user friendly as possible, making it so that testers required very little setup to access the prototype. All that was required, after the initial unzipping of the directory and clicking the .EXE file, was to select what they wanted to do from the main menu. The main menu consisted of a controls menu which described the use of controls, a goals menu which explained what was needed from the tester and how they could achieve the requirements and two play modes. A button that started the static level and another that generates a level using the WFC algorithm.

From the tester's experiences and answers from questionnaires, the gathered data was used to reach a conclusion and answer for the research question.

4.0 Results

4.1 Questionnaire Data

A series of testers where hand selected and asked to test the application before filling out two questionaries. These testers varied in experience regarding playing games, some being avid gamers and others rarely ever touching video games. These testers were chosen to produce accurate and non-bias results as when a game is released, there is no guarantee that all players will be of a certain skill level. Some gamers play daily, others, a couple of times a month, or even year. There was a total of 7 testers.

4.2 Playtest Data

4.2.1 Player playtest data

The first questionnaire was aimed at gathering results related to the tester's gameplay experience, focusing more on the 'fun factor' more so than the technicalities of the implementations. Within this questionnaire, there were a variety of questions ranging from content to question type, though, most questions were on a scale between one to five. One usually being the negative experience and five the positive.

Briefly, the user experience information gathered from the first questionnaire was mostly positive and this will be expanded upon throughout the next sections.

4.2 Playtest Data

4.2.1 Player Playtest Data

The first questionnaire was aimed at gathering information related to the tester's general experience with the application. Questionnaire one aims to retrieve data from the tester in a mix of different formats in relation to the tester's general 'fun factor' focused experiences. There also contains some general questions regarding the tester's background with games in general with the goal of utilising this information to extract any relation between the tester's exposure and experiences with games and their gameplay experience as a whole.

Question Number	Question	Response Type
Q1	Do you play video games often?	1 - to - 5
Q2	I consider myself a gaming hobbyist	Yes/No
Q3	I have much experience playing first person games with movement based mechanics	1 - to - 5
Q4	I had a good experience with the static (human-made) enviroment	1 - to - 5
Q5	I had a good experience with the procedurally generated enviroment	1 - to - 5
Q6	Overall, I feel that the procedurally generated enviroment was just as enjoyable as the static (human-made) enviroment?	1 - to - 5
Q7	Overall, I feel that the procedurally generated enviroment was better then the static (human made) enviroment	1 - to - 5
Q8	How many procedurally generated levels did you try (generate)?	Typed Answer
Q9	How many procedurally generated levels did you complete?	Typed Answer
Q10	I would like to see procedural generation used by more companies going forward	1 - to - 5
Q11	What general feedback would you give? This could be regarding potential improvements, overhauls/changes/expansions on content, or aspects you liked?	Typed Answer
Q12	Did you like the relationship between the gameplay mechanics and the procedurally generated enviroments, and would you like to see this technique used for level-building for more games in future?	Typed Answer

Table 1: Table of all questionnaire one questions

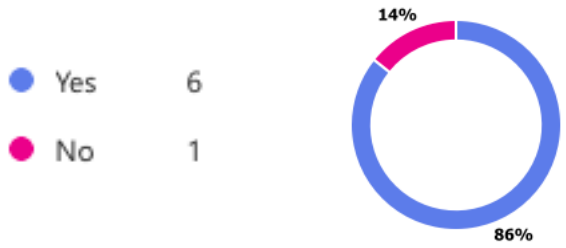

Table 2: Pie Chart of questionnaire 2 - question 2 results

Table 1 contains a list of all questions that the tester was issued within the first questionnaire alongside their response types. These questions were the main method of gathering information regarding the tester's experience of fun-based element of the prototype. The results played a crucial role in investigating and analysing the data to format an answer for the research question.

Question	Overall Statistic	Overview of Full Results
Q1	Average: 4 / 5	3x (5), 3x (4), 1x (1)
Q2	86% -> Yes	6x (Yes), 1x (No)
Q3	Average: 4 / 5	4x (5), 1x (4), 1x (3), 1x(2)
Q4	Average: 3.57 / 5	2x (5), 1x (4), 3x (3), 1x (2)
Q5	Average: 3.86	4x (5), 1x (4), 1x (2), 1x (1)
Q6	Average: 4.14	4x (5), 2x (4), 1x (1)
Q7	Average: 3.29	1x (5), 2x (4), 3x (3), 1x (1)
Q8	Average: 5.7 maps generated	20, 1, 2, 2, 6, 6, 3
Q9	Average: 4 maps completed	14, 1, 2, 2, 3, 3, 3
Q10	Average: 3.86	2x (5), 2x (4), 3x (3)
Q11	No Average - Written Feedback	Will be descussed seperately
Q12	No Average - Written Feedback	Overall, everyone would like to see more generation.

Table 3: Table of questionnaire one results

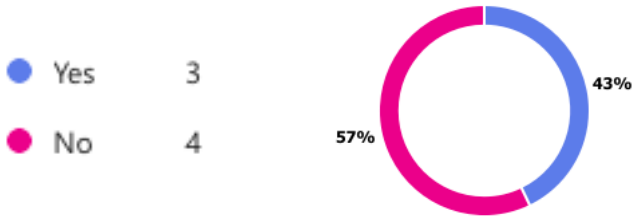

Table 4: Pie Chart of questionnaire 2 - question 2 results

Some questions required written feedback which contained some similarities as well as contrasts, seen from the answers given.

We can clearly see from the results above, a couple of main points. Firstly, an average score of 4 was calculated showing that the majority of testers play video games regularly, expanding on this, stating that 86 per cent of testers consider themselves to be gaming hobbyists. All average-based data is based on the tester's answers between 1 and 5, one being the worst case and five being the best-case response. All averages are calculated form this approach. An average of 3.57 was given regarding the tester's prior experience and exposure to movement-based gameplay mechanics, analysing that most had exposure in some sense. Most also consider themselves to have a strong grasp of movement-based gameplay mechanics, having an average score of 3.86. A strong majority shared that they found the PGL to be just as entertaining as the static environment as an average of 4.14 was calculated. Over half of the testers also shared that they found the PGL to be better than the static level, having an average score of 3.29. This data was analysed from an average level generated of 5.7 levels and an average of 4 of these levels completed. Lastly, an average of 3.86 testers shared that they would like to see more PCG map generation used by industry games going forward, this being over half of the testers. All testers showed interest in some form, however, some selected a range of interest between 3 and 5, with no testers providing one or two as an answer, warranting the results stating all testers are interested in seeing more PLG going forward. It is important to consider that all averages sit on a scale between 1 and 5, 1 being the lowest and 5, the highest score.

The first written responses were a simple numeric value where the tester stated how many levels they completed. The number of levels generated alongside an average number of levels generated can be found on Table 2.

The second written responses were in relation to the number of levels completed by testers. The following results are in the same order regarding users when compared to the levels generated above. The number of levels completed alongside an average number of levels completed can be found on Table 2. These results are in the same order as the levels generated for example, the two first values of: 20 levels generated, and 14 levels completed were the same user.

The third written responses were answers to question 11. It was generally found that half the users liked the simplicity of the controls whereas the other half found the controls to need some polish. However, it was found that testers who liked the simplicity of the controls did not fully utilise the more advanced movement-based gameplay mechanics mechanics due to a lack of gaming experience whereas the testers who stated polish was required were more experienced and claimed to be gaming hobbyists, and in turn, used the more advanced movement-based gameplay mechanics .

The final response asked if the tester liked the relationship between the movement-based gameplay mechanics and the PGL and asked if the tester would like to see more WFC PCL used for more games in future. All responses agreed and confirmed interest in PLG being used in game development going forward. One person stated that ever level felt ‘different and new’. Others expanded and stated that they ‘liked the tight corners and narrow spaces combined with the wallrunning’. Another tester showed interest in PLG for 3D games, as PLG is generally done in 2D environments rather than 3D. Generally, all feedback regarding the PCG was well received.

4.2.2 Player Generation Data

The second questionnaire aimed to gather information focused on the technical side of the tester’s experience. The second questionnaire aimed to gather both qualitative and quantitative data regarding the tester’s experience.

Question Number	Question	Response Type
Q1	My experience contained bugs (technical issues) that ruined the enviroment and its flow	1 - to - 5
Q2	Were any levels generated that made it impossible to complete the level?	Yes/No
Q3	If yes, please explain why this made the level difficult or impossible to complete	Typed Answer
Q4	I feel that the general preformance was stable and flowed well	1 - to - 5
Q5	Character movement and mechanics were reponsive and useable?	1 - to - 5
Q6	Were there any out-of-place objects (floating, clipping, etc)?	1 - to - 5
Q7	What general improvements would you make?	Typed Answer

Table 5: list of questionnaire two questions

Table 3 contains a list of all questions found within questionnaire two. These questions were developed and formatted with the intention of gathering both quantitative and qualitative data in order to conclude the best answer for the research question.

The final written response regarded general improvements of the prototype. These responses varied in content that all related to the movement mechanics and what can be improved about them. The first asked for the adding of more variety in movement mechanics and generated environmental factors to allow for a more varied play experience. A few asked for the first-person controller to be more responsive however, one expanded and stated that there was a ‘floating’ feel to the movement due to long accelerations and decelerations, which is a byproduct of applying forces to a rigidbody. The same was stated again however expanded, stating that the jump was also slippery and unpredictable when trying to reach the objective platform. Two people said they would add no improvements, and a final response simply asked for less pit-asset based generation.

Question	Overall Statistic	Overview of Full Results
Q1	Average: 2.71	2x (4), 3x (3), 2x (1)
Q2	43 % Yes	3x(Yes), 4x (No)
Q3	3 Responses	Will be descussed seperately
Q4	Average: 4.71	6x (5), 3x (1)
Q5	Average: 3.43	2x (5), 2x (4), 3x (2)
Q6	Average: 3.43	2x (5), 4x (3), 1x (2)
Q7	7 Responses	Will be descussed seperately

Table 6: Questionnaire two answers

The results from the testing can be seen in table 4 however, there were also some typed response-based questions which required the tester to manually type a semi-detailed answer regarding their experience. Some results worth noting include an average of 2.71 testers contained bugs within their experience, 43 per cent of testers stated that they experienced in-completable levels being generated, an average score of 4.71 was given, showing that the performance was generally considered stable and that the prototype flowed well, an average score of 3.43 was given showing that there were some out-of-place objects however, most found it to be little or none. Lastly, an average score of 3.43 was concluded, proving that most had a stable time with the movement mechanics however, much polish could be done to improve the experience for many testers if this project were to be developed further. It is important to consider that all averages sit on a scale between 1 and 5, 1 being the lowest and 5, the highest score.

5.0 Discussion

5.1 Playtest Session

The play test was conducted by participants in a mix of remote and in-person environments. Typically, the less experienced testers opted for an in-person session as this allowed me to verbally explain controls and goals whereas more experienced testers participated in a remote fashion.

The play test consisted of the tester(s) being provided with the prototype and questionnaires and asked to start the prototype. The testers were then confronted with the main menu consisting of four input buttons: play static level, play generated level, instructions and controls.

It was found that most testers opted to read both the instructions and controls pages before continuing, though, a few of the more experienced gamer-hobbyists decided to jump straight into the game after a brief verbal introduction. Overall, the test sessions were successful and there was little-to-no confusion on part of the testers. The sessions were successful at showing and acting as a means of gathering data regarding static levels and PGL, and their relationship with movement-based gameplay mechanics.

5.1.1 Questionnaire Data Analysis – Questionnaire One

The first questionnaire was directed on the tester, their background with video games and their general, non-tech related experience with the PGL. Generally, the results from questionnaire were positive, this doesn't mean that the generation or movement-based gameplay mechanics were of industry standard state.

The first question directly asked the tester if they play video games often. The response to this was an overwhelming yes, with only a single participant stating that they did not, seen from the average from the question. This is important as there is a difference between playing games often and considering oneself to be a gaming hobbyist. Hobbyists may not have as much as experience as a player who plays often but not as a hobby. This is a nice bridge between questions one and two, as question two asks about the tester being a hobbyist. The results here were clear as the only tester who does not consider themselves a hobbyist, also stated that they do not play often. This will be crucial in providing steady and accurate results.

The third question is in relation to the tester's experience with movement-based gameplay mechanics. The vast majority of tester's stated that they were comfortable with movement-based gameplay mechanics. This generated desirable data as most people who buy video games and keep up to date with upcoming titles and technologies tend to be hobbyists, which usually have much experience with video games and, by extension, movement-based gameplay mechanics.

Questions four and five focus on the tester's general experience with the static and generated levels. Just over half of the tester's reported that they had good experiences which shows that PGL are valid technique, even for smaller projects or prototypes. The overall experience of the tester is just as important as the in-depth, technical experiences as this includes the opinions and experience of more casual and less tech-savvy players. This data was intended to gather data regarding whether the levels were fun to play as, if not, there would be no reason for PGL.

Questions eight and nine are aimed at gathering data regarding the levels generated against the levels completed. The data shows that an average of 5.7 maps were generated and that an average of 4 levels were completed per user. This shows that with much more time and fine tuning, PGL can produce stable and reliable experiences.

Question ten was a general question, asking testers if they would like to see more PLG going forward. It was analysed that every tester expressed, at least, some interest in seeing more PLG in future video games.

Questions eleven and twelve ask the testers to give written feedback and opinions on their experience and what they would like to be changed/improved.

5.1.2 Questionnaire Data Analysis – Questionnaire Two

The second questionnaire was a more technology focused approach as to gather data surrounding the movement-based gameplay mechanics and PLG algorithm. Throughout the series of questions asked, there was a mix of both qualitative and quantitative data to answer the research question accordingly.

Questions one and six were aimed at gathering information regarding if the tester's experienced contained bugs. Such bugs could be out-of-place objects and other similar qualities. This data returned with average of 2.71, stating that most people's experiences contained little to no bugs.

Question two aimed to research whether or not the testers experienced any PGL which were impossible to complete, whether there were no progress paths, if the user was simply boxed in or if the tester was spawned above a generated hole. It was found from the data analysed that 43% of testers did experience at least one impossible level. Question three simply asked the users to expand and give reason for their answer if they responded 'yes'.

The fourth question was aimed at having the tester analyse and report the general and overall performance of the prototype within their experience. This is important as, in ways, the data gathered here is important to the analysis of the relationship between the PGL and movement-based gameplay mechanics.

The seventh and final question was tasked with gathering data focused on general improvements, and in what way and form the tester would like to see such improvements made to the prototype, gameplay mechanics and generation algorithm.

5.2 Project Findings

5.2.1 Project Summary

The findings gathered as a result of the prototype showed to be mostly positive. The prototype showed that there is definite potential in using PCG to generate environments and levels that can work and flow well with movement-based gameplay mechanics, even if the prototype was not a flawless success. The developed mechanics and WFC algorithm proved to be a solid contender for the generation of movement-based environments however, the prototype would need more time and improvements to be strong enough to generate content for a full-fledged movement-based video game as there were both flaws and strengths to the algorithm developed. One strength being that WFC will only use models supplied however, would require a designed or engineer to develop and implement a full-fledges and concrete set of rules for each node type. Despite this, tester feedback was positive and produced promising results.

5.2.2 Research Question

The research question being researched throughout this research was: "Does procedural level generation positively affect user experience of movement-based gameplay?". The results indicate that PLG can, indeed, be used to generate strong and well flowing environments for games with a focus on movement-based gameplay mechanics and reinforces that the developed approach is an effective way in doing so. By producing a playable entity featuring the movement scripts alongside a world generation object (or static level), the positive results were generated, though, it was stated by the testers that both the movement mechanics and PLG technique needs improvement, something that would require more time.

5.3 Critical Evaluation

5.3.1 Development Evaluation

Development of the prototype has remained consonantly positive throughout development, after a certain point. After the initial proposal, a strong method and plan was developed alongside a gnatt chart that was followed very tentatively throughout the entire development process.

Originally, the chosen game engine was Unreal Engine 5 (UE), as UE has a very respectable reputation throughout developers of three-dimensional games. After some research and two and a half weeks of trying to implement the chosen WFC algorithm into UE to produce a simple environment, no progress was made and all implementations made to the algorithm ended up generating nothing. This included when feeding the algorithm three models to use as templates within its generation attempts. After some time, the decision was made to transition to Unity, where a generation algorithm was produced after only one week and a half.

Unity was a much simpler engine to work with when implementing the WFC algorithm. This was due to a multitude of factors including the simplicity of using scriptable objects to the sheer amount of research materials when compared to UE. The first mistake was where the generation of the map was generating the wrong way however, this was a simple fix within the

script when generating the grid, where the X and Y coordinates of the generation was changed to reflect on the X and Z axis.

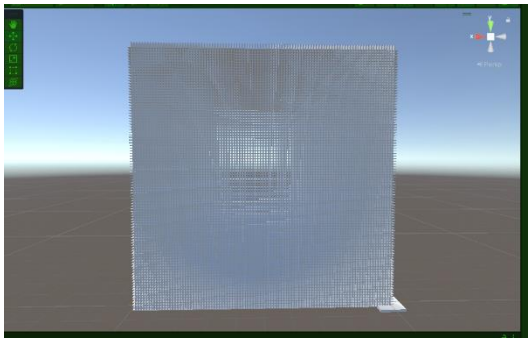

Figure 1: First generation attempt

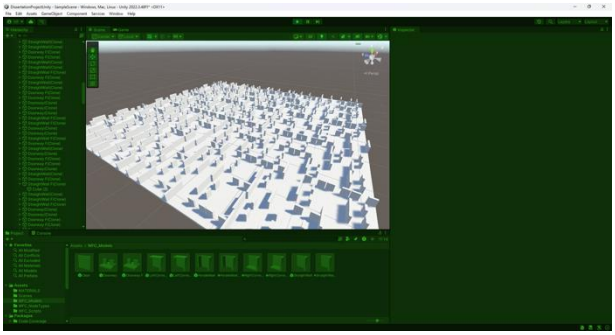

Figure 2: Second generation attempt

From this point, development accelerated as the main foundation had been constructed as the next step was developing more models and nodes. The nodes and models were an easy task to work through and get working however, was the longest stage of the development of the algorithm. This was because every single node needed a model, where each model needed to be created within Unity and converted to a prefab, which sounds simpler that it was. The process was simple however, the main challenge was ensuring the scale, alignment and position of each model was correct, alongside ensuring each child objects of the model, such as walls, also aligned with the children objects of other prefabs. This required a lot of trial and error as well as the running, note-taking and then stopping and changing to the prefabs as required. The nodes and models took up sixty per cent of the entire development time. Once the nodes were created, each node then needed to be analysed and planned before progressing to the rules of each node. Each node possessed a series of rules which determined what nodes can be placed next to others, the introduction of the generation's propagation. The propagation required a lot of calculations, note-taking and on-paper recording of how each ruleset should appear for each node. This also contained a lot of trial and error, generating a series of environments, studying the environments and pruning out rules that did not work well or hindered player progression though the level. The level was tested as a square but was then refined to be more of a rectangular shape to mimic the static level. Walls were also added to the generation, as shown below.

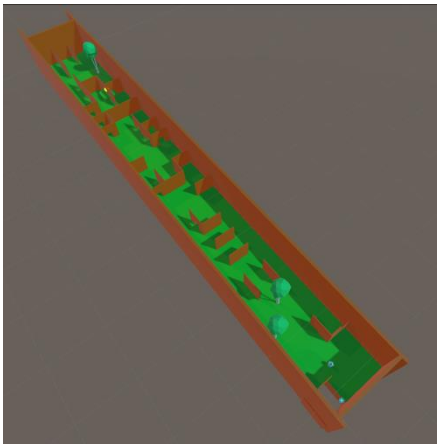

Figure 3: Final display of generation

Similarly to the PLG, the first version of the movement-based gameplay mechanics were implemented successfully, though, in UE5. The gameplay mechanics were originally implemented from a passion project done throughout the summer and worked very well. Such gameplay mechanics included sliding, wall running (with smoothed camera tilt) and a player dash, though the wall running was a product of UE5's blueprints system whereas the dash and dodge were implemented via C++ scripting. Eventually, with the transition to Unity engine, the gameplay mechanics were scrapped and restarted form a clean solution.

A basic player movement script was sourced from the credited game developer and YouTuber, Dani, as his efficient player movement script included all of the basic movement required as a base plate for the advanced movement. Once the basis for the movement was implemented and altered where required, an additional C# script was developed as an extension of the basic player movement, featuring the wall running mechanic. The wall running script was tightly coupled with the basic player movement script for close communication between scripts, and the basic movement script acted as a dependency for the wall running script. Once the wall running script was developed and revised, they were then fine-tuned, alongside the basic movement, to ensure the movement was controllable and polished enough for testing. On reflection, the movement mechanics feel floaty and light however, the aim was to connect this floatiness to the generated map and player with the aim to allow the player to build momentum and maintain speed. The floating feel is a result of applying forces to the player object's rigidbody. On reflection, after feedback, another method of movement with a heavier player would be researched and implemented instead.

Overall, I was less comfortable with Unity as a whole and only decided to make the switch from UE5 to Unity due to UE5's lack of source material when compared to Unity's. A lot needed to be learned throughout the development process of the prototype however, this results in a successful prototype and generated valid and strong results.

5.3.2 Playtest Evaluation

The playtest consisted of a medium group of testers of varying skill levels. A total of seven testers partook in the play test and the results from their experiences varied. The prototype was compiler using Unity's build system, which was compiled, linked and packed into an exe file which was useful as this allowed for a complete test to be done fully remotely, which was the preferred approach here with 71 per cent of participants partaking fully remotely.

After each participant extracted the executable file, they partook in the testing of the prototype where the results were gathered and later analysed. As mentioned, the results varied as a major factor in this range was the mix of user ability, experience and general exposure to video games. It can, however, be argued that the results may be inconsistent or unreliable because of the range of player experience, and that this may generate inaccurate results however, it is important to remember that not all gamers are avid enjoyers of video games and some may be casual player or first time players and so it is important to test a wide range of people. Due to this, the conclusions drawn from the analysed data are accurate to a wider audience and are an accurate representation of how player-bases work as a whole.

The tester was allowed to test as many maps as they desired as it is believed that giving the use a choice in how many maps they generated would provide more authentic and accurate results. The only rule was that at least one static and one generated level was to be tested in order to

provide valid and accurate results on the questionnaire. Though it was found that most testers generated and fully tested more that a single level and played, averaging at five point seven levels generated. This is desirable as this provided a range of accurate and valid results from each tester's experience.

All data generated by the play tests have benefited the study greatly as this game human-based results rather than simulating the results thorugh artificial intelligence algorithms or bots. This, similarly to the previously mentioned point, has provided stable and more meaningful results as only so much data can be extracted from AI and bots as they are usually numbers driven and so, struggle, or in some cases, cannot provide data as to how relationships and mechanics/features feel to a human player. Additionally, the cons of using human-based testers have been discussed prior, explaining that varying levels of tester experience can impact a tester's opinions and results given however, as discussed, this is a more accurate representation of a wider player-base. Another unavoidable factor would be that only people with interest in movement-based gameplay mechanics would actually buy games effected by this study, and as a result it can be argued that some of the data produced is unfaithful due to the testers who may not be overly enthusiastic about video games. However, all games released will have new players interested in the game or genre of game as a whole.

6.0 Conclusion

6.1 Main conclusion

This study initially set out to determined if procedural level generation positively affect user experience of movement-based gameplay, and whether it is a viable form of level and map creation in industry-standard games going forward. Throughout the testing, data gathering and analysis of said data, it was determined that, through direct tester feedback, that PLG is a valid form of third dimensional map generation for use within movement-based video games. It was also found that WFC is a valid approach to this level generation and that, with additional polish and development time, can be used as a technique to achieve this map creation. The algorithm was able to generate valid and successful map generations with very little invalid or un-completable levels or maps. The aim of the maps is to provide a playable and well flowing environment for players to test the movement mechanics which were, overall, successful.

The feedback and data gathered prove that the approach used throughout this study state that, in most cases, the WFC algorithm provided successful levels for the testers to play through and worked well with the provided mechanics. The feedback did state that the movement-based gameplay mechanics could use some polish and general improvements but seemed to work well enough, as a prototype, to generate accurate results which resulted in the answering of the research question.

The primary use of the prototype developed in its current form is produce partial to full levels for game developers to implement within their games. It would be the developer's responsibility to provide the algorithm with their desired nodes and models as well as set up any propagation rules. An additional responsibility would be to polish the algorithm to the desired state and develop any gameplay mechanics to work with the format given by the generation, as well as model and node-based data such as assigned tags to objects on the created models.

Overall creating a flexible, yet robust level generation algorithm is a challenge in and of itself. If the generation algorithm is done correctly, it provides a strong and replayability factor to the game in question however, if done poorly, it adds undesirable and incompletable levels, bugs, illogical item placement, terrain and texture glitches and much more. Especially for first person perspective games where the player view is closer to the terrain and objects, this can be the making or breaking of an experience. Overall, PCG and PLG is unpredictable and takes a lot of time and refinement to get correct and by extension, flawless.

5.2 Implications

The research and work carried out throughout the process of this study stands as an example for future efforts, attempts and extensions to better PLG for movement-based video games going forward. The study demonstrates how PCG can be used for level generation in movement-based video games and this confirms that generating levels for such games is a valid and effective approach and should be considered and developed further throughout the development of movement-based video games. The results also state that this approach can definitely become a more commonly utilised approach in future work throughout the genre.

6.3 Future Improvements

For the prototype to be improved and made release-ready, a couple of major improvements would need to be implemented to improve the general quality of life of the project.

One improvement regards the generation itself. Some testers reported issues with the generation, one example is being spawned above a drop, causing the player to fall in an endless loop. An approach to this would be to improve, as well as add more nodes and models to increase possible starting nodes and generation possibilities as well as a total review and refinement of the generation and propagation rules.

Another improvement that would be required is the movement-based gameplay mechanics. Many testers reported a floating feeling when moving the character and this is a result of applying forces to the player's rigidbody component. One approach to improvement would be a complete overhaul of the movement mechanics, starting from scratch. The issue here is that it more math and effort would be required if a momentum build-up was desired. Another approach would be to revise and refactor the current movement on a script level as well as an inspector level. Some inspector level alterations would include but not be limited to changes to the player mass and forces applied.

Another unmentioned improvement would be to add more versatility to the generation, such as adding a second floor or floating platforms. Maybe even adding an underground level, inspired by Mario, where tunnels or tubes are used to access these areas. More versatility would add more depth and replayability to the levels, however, would require a lot more work and development time.

7.0 Appendices

Appendix A – Raw Questionnaire Data

All raw data gathered and questions can be found in the excel file ` Questions Excel Sheet` located in the same directory as this research paper.

Appendix B – Consent Forms

All consent forms can be found in the sub-directory ` Signed Consent Forms` located in the same directory as this research paper.

Appendix C – GDPR Research Data Management Data Sign Off Form

Abertay University

GDPR Research Data Management
Data Sign Off Form

For undergraduate or postgraduate student projects supervised by an Abertay staff member.

This form MUST be included in the student's thesis/dissertation. Note that failure to do this will mean that the student's project cannot be assessed/examined.

Part 1: Supervisors to Complete

By signing this form, you are confirming that you have checked and verified your student's data according to the criteria stated below (e.g., raw data, completed questionnaires, superlab/Eprime output, transcriptions etc.)

Student Name:	Harry Findlay		
Student Number:	2102552		
Lead Supervisor Name:	Naman Merchant		
Lead Supervisor Signature	Naman Merchant		
Project title:			
Study route:	PhD ☐	MbR ☐	MPhil ☐
	Undergraduate ☑	PhD by Publication ☐	

Part 2: Student to Complete

	Initial here to confirm 'Yes'
I confirm that I have handed over all manual records from my research project (e.g., consent forms, transcripts) to my supervisor for archiving/storage	HF
I confirm that I have handed over all digital records from my research project (e.g., recordings, data files) to my supervisor for archiving/storage	HF
I confirm that I no longer hold any digital records from my research project on any device other than the university network and the only data that I may retain is a copy of an anonymised data file(s) from my research	HF
I understand that, for undergraduate projects, my supervisor may delete manual/digital records of data if there is no foreseeable use for that data (with the exception of consent forms, which should be retained for 10 years)	HF

Student signature : 

Date: 29/04/2025

8.0 Bibliography

Angel, J.(2014). Game Maps: Parkour Vision and Urban Rel
ations. In: Schiller, G., Rubidge, S.
(eds) Choreographic Dwellings. New World Choreographies
. Palgrave Macmillan, London. https://doi.org/10.1057/9781
137385673_11

Bhojan, A. and Wong, H. (2014). ARENA -Dynamic Run-time Map
Generation for Multiplayer Shooters. [online] Link.springer.com. Available
at:
https://link.springer.com/content/pdf/10.1007%2F978-3-662-45212-7_19.
pdf (Accessed: 27 Feb. 2025)

Boris. (2020). “Wave Function Collapse Explained” Available at:
https://www.boristhebrave.com/2020/04/13/wave-function-collapse-explained/ (Accessed: 5
Mar. 2025)

Büyükşar, O., Yıldız, D. and Demirci, S. (2024) Enhancing
wave function collapse algorithm for procedural map
generation problem, Niğde Ömer Halisdemir Üniversitesi
Mühendislik Bilimleri Dergisi. Available at:
https://dergipark.org.tr/en/pub/ngumuh/issue/86158/1361413
(Accessed: 13 October 2024).

Coffee Stain Studios (2023) Valheim [Video Game]

Contributors to No Man’s Sky Wiki (no date) *Galaxy*, *No Man’s Sky Wiki*. Available at:
https://nomanssky.fandom.com/wiki/Galaxy (Accessed: 30 April 2025).

David, M. (2016). Tile-based Method for Procedural Content Generation.
[online] Available at:

41
https://etd.ohiolink.edu/pg_10?0::NO:10:P10_ACCESSION_N
UM:osu1461077485 (Accessed 18 Jan. 2025)

David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin and Steven Worley:
Texturing and Modeling: A Procedural Approach (Third Edition). Morgan Kaufmann
Publishers, 2003.

D, Devy. (2019) FPS ‘Movement Rigidbody’. Available at: https://github.com/DaniDevy/FPS_Movement_Rigidbody (Accessed 22 Dec. 2024)

DICE (2008) Mirrors Edge [Video game]

Facey, K., and Cooper, S. (2024) ‘Toward Space-Time WaveFunctionCollapse for Level and Solution Generation’, Khoury College of Computer Science.

Fun Pimps (2013) 7 Days to Die [Video Game]

F. Kenton Musgrave, Craig E. Kolb and
Robert S. Mace: The Synthesis and Rendering of Eroded Fractal Terrains. Computer Graphics, Volume 23, Number 3, July 1989,
pages 41-50.

Gearbox Software (2009) Borderlands [Video Game]

Hardman, C. (2024). Advanced 3D Movement. In: Game Programming with Unity and C#. Apress, Berkeley, CA. Available at: https://doi.org/10.1007/978-1-4842-9720-9_38 (Accessed: 14 Dec. 2024)

Heaton, R. (2018) The wavefunction collapse algorithm explained very clearly, Robert Heaton. Available at: https://robertheaton.com/2018/12/17/wavefunction-collapse-algorithm/ (Accessed: 13 October 2024).

Hello Games (2016) No Man’s Sky [Video Game]

Ilyan O. O. et al. (2024) Розробка системи процедурної генерації ігрових карт на базі алгоритму колапсу хвильової функції, Development of the system of procedural generation of game maps based on the wave function collapse algorithm. Available at: https://journals.dut.edu.ua/index.php/sciencenotes/article/view/2948 (Accessed: 12 October 2024).

Ludwig, J. (2020). Procedural content is hard. [online] Programmerjoe.com. Available at: http://programmerjoe.com/2007/02/11/procedural-content-is-hard/ (Accessed 3 Feb. 2025).

Kenny. (2021) “Procedural Generation: An Overview”. Available at: https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41 (Accessed: 2 Nov. 2024).

Maxis (2008) Spore [Video Game]

Millonig, A. and Mitgutsch, K. (2014) ‘Playful mobility choices: Motivating informed Mobility Decision making by applying game mechanics’, ICST Transactions on Ambient Systems, 1(4). doi:10.4108/amsys.1.4.e3.

Mahmoud, M. (2024) “The Redemption of No Man’s Sky is INCOMPARABLE”. Available at: https://www.kitguru.net/gaming/mustafa-mahmoud/kitguru-games-the-redemption-of-no-mans-sky-is-incomparable/ (Accessed 11 Nov. 2024)

‘PARKOUR VISION’ (2013) Headway Parkour Blog. Headway Parkour Blog, 20 October. Available at: https://headwayparkour.weebly.com/blog/parkour-vision (Accessed: 13 October 2024).

Olsen, J. (2004) ‘Realtime Procedural Terrain Generation’, Department of Mathematics and Computer Science (IMADA) University of Southern Denmark.

Respawn Entertainment (2016) Titanfall 2 [Video game]

Schier, G. (2015) “Pros and Cons of Procedural Level Generation”. Available at: https://schier.co/blog/pros-and-cons-of-procedural-level-generation (Accessed: 17 Jan 2025).

Id Software (2020) DOOM ETERNAL [Video Game]

Vote4Wes. (2019) “7 Days To Die Review”. Available at: https://thecouchcoopcouple.home.blog/2019/07/27/7-days-to-die-review/ (Accessed: 3 Feb. 2025).

Zhu, C., and Zhang, Y. ‘A First-Person Game Designed To Educate And Aid The Player Movement Implementation’, Beckman High School, University of California. Available at: https://csitcp.org/paper/13/132csit03.pdf (Accessed: 15 Dec. 2024)