UNIVERSITY OF LINCOLN

How does the order in which Procedural Generated Content is generated affect the narrative of a game?

By

Michael Christie

A dissertation submitted in partial fulfilment with the requirements for the degree of BSc (Hons) Games Computing

School of Computer Science.

University of Lincoln

2020

Contents

Acknowledgments

I just want to thank my supervisor for supporting my work during this study, all those who participated in the study and everyone who read my work and gave me feedback to improve it.

Abstract

This project looks at how the order in which different procedural systems generated content can affect the narrative of a game. In this paper we look at current research around quest systems and village generators, to find methods to building our own systems and then evaluating our system by conducting a user study. Quest generation creates a narrative which leads the user through the world and creates a goal to motivate the user to complete. The village generator creates an environment for the quest to be played out in. The results of this paper shows that participants could not see a notable difference in which generation order was used.

Introduction

With more companies heading towards implementing Procedural Content in their games, Procedural Content systems will have to heavily interact with each other. This interaction can change the type of content that is generated at the end. In this study we aim to explore how a Quest system and Village generator's content changes depending upon the order in which they're content is generated. There is a lot of research around generating quest procedural and generating settlements, however there is little research on implementing the two generators together or the affects of this. To do this, we plan to look at how immersed the player is with a game that generated a quest around a village and explore whether the player can notice a difference between having a village generated first or having a quest generated first.

To achieve this aim, there are multiple milestones that were set. The first milestone was creating a prototype, which was used to explore the concepts that we learned during the research phase of the project. This gave us time to understand how each system would work at a basic level so a method could be created for linking the two systems together. After creating a prototype, we could then focus on creating the actual artefact for this project. The next milestone was creating the village generator. This would create a village given a specific space. This was broken up into multiple goals, including designating building space, generating buildings in the space and generating roads. Once this milestone was hit, the quest system could be implemented using data from the village to populate the quest to begin with. Once the quest system worked with the village generator, the reverse had to be done, where the quest system influences how the village is generated. The final milestone was evaluation, collecting participants to play the game and collect data back from them, then analysis that data to form a conclusion for the project.

Background and Literature Review

This work is intended to study the current research on quest and village generation in order to understand the field. We will demonstrate our knowledge using an implementation of our method to create interactive quests and explorable villages and evaluate it to determine if there is a correlation between the order of procedurally generated content.

Quest Generation

There is a lot of research on Quest Generation and understanding how and what makes a good quest. There is also different styles of quests to consider. Ashmore and Nitsche focused on generating lock and key quests in 'The Quest In A Generated World' (Ashmore and Nitsche, 2007). A lock and key quest requires an obstacle to be generated, such as a locked door, and then requires the player to find an object to unlock the door, such as the key, or an axe to chop the door down. The lock and key style of quests are great for explorative constant generative worlds due to the fact is uses *spatial progression* to generate content, which means each lock can generate a new connected room, building into a sort of dungeon crawler. However, this type of generation style is simplistic and wouldn't work on more complex quests as it generates content linearly, instead of allowing breaching quests to develop. This linear type of quest requires the player to keep moving through the world to complete the quest, whereas generating quests around a village requires the player to be able to loop round and back to previous areas of the map.

Quests in A Generated World also talks briefly about the purpose of quests, as a means of structuring gameplay in an virtual environment (Jenkins, 2004) which relates to how quests can be used as a basis to generating an environment. Nitsche and Ashmore base the idea of a quest off of

Aarseth's definition of a quest which has three elements *'space, challenges, and goals'*, developing this further to add their own point, 'setting'. Creating a setting for the quest impacts *the players engagement due to the framing of the game world.* This framing can hugely impact the narrative of a quest line. It would feel wrong to be collecting swords and attacking zombies when the game world is set up in a 21st century city, so designing a setting for the game to sit in to align with the quests is an important aspect to take away.

There are many important details which offer a connection between generation quests and worlds procedural, however during their testing phase there is a lack of detail as to how they achieved their conclusion. "…In our tests, the open nature of the game world in Charbitat was appealing to players, but it lacked context…The game has an overall narrative and dramatic setting with several goals within the space, but the space was not confined or limited in any way."(Ashmore and Nitsche, 2007). There is little detail as to what type of test was conducted or how many participants were used to base this statement off of, however later on in the paper they states how comparing two players experience against each other was difficult due to how uniquely different each players quest and game worlds were.

Quests require a purpose to be able to fit into the game world. Doran and Parberry analysis hundreds of quests from multiple MMORPG's to create a classification of quests and the structure in which they follow in 'A Prototype Quest Generator Based on A Structural Analysis of Quests from Four MMORPGs' (Doran and Parberry, 2011). This is so that a quest can be *formed suitably and in a manner which ensures the quests makes logical sense given the current game state.* A quest generator requires constraints and definitions as to what an appropriate quest is based on the current game state. By classifying quests into several types, the current game state can be checked, and the most appropriate quest classification can be picked to generate the quest off of, as Parberry's and Doran's *decomposition is closely related to the game state.*

These classifications are defined as motives which is "a statement of the most important concern an NPC has, and the quest is intended to address this concern."(Doran and Parberry, 2011). There are nine motives that Doran and Parberry use, splitting up the vast amount of possible quests into a smaller pool to use. For example, a Kill quest can fit under the motive of protection or reputation, depending on the target and setting in which the quest would take place. By choosing a motive, it helps to limit what quests can be generated by labelling each quest to a type.

Motive	Description
Knowledge	Information known to a character
Comfort	Physical Comfort
Reputation	How others perceive a character
Serenity	Peace of mind
Protection	Security against threats
Conquest	Desire to prevail over enemies
Wealth	Economic Power
Ability	Character skills
Equipment	Usable asset

Table 1 Shows an overview of motives from Doran and Parberry

Generating quests requires planning to ensure the quest is completable and fits into the game setting, as discussed earlier. Doran and Parberry explain that a *Quest consists of an initial state, a goal and a set of actions* which is the underlying structure in which every quest follows, however planning a quest to fit into the game world and game state requires an *extensive search.*

By picking a motive, we would have evaluated the current game state to pick a classification of quest that would be most fitting to the current game situation, which can then be used to pick the initial state and end goal for the quest, however we are yet to consider the actually actions to be taken inside of a quest. Doran and Parberry also discuss the most common actions from quests they analysis. These actions consist of Go to, Attack, Give, Talk, Defend and Escort to name a few. By implementing these actions, we would be able to cover and generate any type of quest, using the motive from before to limit and choose which action to pick next. This would require each action to have a pre and post condition, which can then be searched for to see if it is a viable action in the game state.

With an understanding about how to generate quests, we need to look at a way in which to search the current game space to be to form a narrative around. Generating stories is a major corner stone of this project, as it feeds directly into the narrative as it *establishes context and motivates players to complete tasks* (Hartsook *et al.*, 2011). 'Towards Supporting Stories with Procedural Generated Game Worlds' focuses on a search-based approach to generating a narrative based off of sequence of points. Hartsook et al. builds stories off of these points, which are *a high-level specification of a period of time with a semantic and recognizable meaning*. These plot points can be interpreted by the same keywords that quests are generated by from the work of Doran and Parberry. These plot points are then joined together to form a structure for the narrative to follow, using the keywords to generate meaningful sentences.

This paper also looks at the way the game world can be generated with the understanding of the story, creating a node tree of how areas of the map connect to each other and flow into the story. Harksook el al. describes this process as creating island and bridges, where islands are *areas where critical plot points occur* and bridges are where *non-plot-specific game play occurs* but connect the islands together, linking the world and narrative together.

There is however an issue with generating in this sort of way, where the spatial layout of a level isn't taken into consideration when generating a narrative, which can cause the player to play in a small section of the map, or to use points far across the map making the player go back and forth between points ruining the players experience. As well, because this approach generates off of a node tree, it can generate more complex quests, however, generate a large interweaving story line requires a large amount of space searching and hardware capacity. This could be broken down however, generating an open starting and ending area, then coupling quests together over time, rather then doing this calculation all at once however this would have an affect on the overall story arch.

Village Generation

'A time-line approach for the generation of simulated settlements' by Williams and Headleand explore how cities are generated over time. It covers a wide variety of areas that are applicable to the artefact for this paper. There is a heavy focus around using a historical period to generate a city, however the means in which they generate cities will still be applicable to this artefact. They also state that there is a large body of research around generating cities, using map data, L-systems and model patterns, however *generating smaller, non-urban settlements (such as villages and hamlets) is an under-researched area.* (Williams and Headleand, 2017). Current research around villages are limited to generating spare road networks using Voronoi tessellation which was then built upon later by (Emilien *et al.*, 2012) to add settlements around this road network.

Williams explores the idea of generating settlements around a seeded position, which is the centre point of the settlement that it expands for, similar to a town centre. Multiple seeded positions are created and evaluated based off of the terrain level they are at, to check that this area would be a valid location to generate a settlement in. Points close to each other are grouped to form the size and possible growth of a settlement.

Road networks are generated along creating a directed street graph, where vertices are added at intersections, and road meshes generated on top of this graph data. The graph can be modified to expand the road network by building upon the current points and branching outwards. Roads are added into the graph by checking existing junctions and finding the angle between these roads. A new road can then branch off of this junction from a minimum angle. In the case of an urban area, roads are perpendicular (90 degrees) off of each other, forming a grid-based approach, whereas villages are more laxed in this grid-based generation. As well as this, new roads have to be a minimum distance away from each other so that they do not overlap causing unusual road layouts.

The final step is to estimate the amount of free space between roads. A ray is cast along each road perpendicular to its direction, returning when it hits either an obstacle, like terrain, another road or a max distance. This is then combined to estimate an area next to the roads which can then be split up and buildings generated in these spaces. The buildings are rotated to fit with the direction of the roads, creating a realistic settlement.

Overall this paper is helpful at evaluating and explaining the core concepts of generating a settlement, of various sizes depending upon the clustering of seeded positions, as well as key concepts about generating a believable road network. Taking this process of generating a village would be a key point of the end artefact.

‘Evaluating Models for Virtual Forestry Generation and Tree Placement in Games’ explores different methods for generating forests and evaluates each method on playability and realism. The three different generation methods are Naïve (random generation), Propagation (simulation) and Clustering (generating around points). By looking at the data collected from this study, it shows that *the Naïve approach scored highly from a* first person perspective. (Williams and Headleand, 2019). A random generation of trees scored higher than the simulated and clustering method on average when looking at the playability in the forest from a first-person perspective. However, from a top down view, a clustering approach on par in ratings with the propagation approach. Using this study, it is clear that *the perspective in which the artefact is to be played from should decide which tree generation method should be chosen.*

NPC’s

NPC’s help to bring the game world together for quests. They can create starting and end points, as well as to help deliver the narrative to the play.

Grey and Bryson build upon the use of motives built by Doran and Parberry but focused around its use in NPCs. They purpose creating relationships between each NPC to design an emotion which can then be used to dictate what type of quest to give to the player in their paper ‘Procedural Quests: A focus for agent interactions in role-playing game’ (Grey and Bryson, 2011). This design creates a more local quest space compared to a global quest space and is designed by how the quest giver feels towards another NPC or event at that game state. This relationship between NPC’s will become important later on when discussing Crowd Simulation.

A quest should be able to help direct the player while also adding narrative to the gameplay. Grey and Bryson believe that Believable Social Agents (BSA) can help give the quest narrative a foundation in the game world. “Without a narrative context, such side quests can be unmotivated and tedious for players.“ (Grey and Bryson, 2011). BSA’s not only help the believability of the quest but can also use quest themselves to create a realistic behaviour and interaction between NPC’s and the game world.

Creating a strong NPC system which can direct the flow of quests around a more local scale and interaction would be impactful to the narrative of the game, as a weak narrative would go against the purpose of this study.

Syzmanezyk and Cielniak explores how to create realistic interaction between NPC’s to generate a crowd affect. Creating simulated crowds *helps to achieve believability into the game world.* (Szymanezyk and Cielniak, 2010). Having too few NPC’s in an area will achieve a lifeless scene, whereas too many will make the game feel too claustrophobic, requiring NPC’s to be distributed across the map depending on the current game state.

Syzmanezyk and Cielniak uses 3 states to control crowds, a physical state, an emotional state and a goal state. The physical state describes the NPC’s position, velocity, direction and mass. The emotional state defines the NPC’s emotions toward the game world and other NPC’s and is used when picking a goal state for the NPC. If an NPC is feeling adventurous, it would choose a goal in an open green space, compared to an NPC that is feeling revengeful. This concept can then be portrayed onto groups of NPCs’, to simulate friends, family or even a functional role (like guards, kings, merchants).

Overall this paper is great at exploring the key concepts of group-oriented NPC that would interact in an open world setting, explaining how the relationship between NPCs’ affects the current goal. Implementing this system would help control the distribution of NPCs around the village, as well as creating realistic behaviour which would increase the narrative and immersion of the player to the game world.

Testing and Evaluation

‘Development and validation of the player experience inventory: A scale to measure player experiences at the level of functional and psychosocial consequences’ (Abeele *et al.*, 2020) explores ways to measure, analyse and understand players experience from a game design perspective. Finding the right questions to ask can often be difficult, leading to questions that can be understood as misleading, however this paper purposes a structure to what questions should be asked to be able evaluate a player’s experience. This paper purposes using a scale type questionnaire, but off of a means-end chain which includes questions which look at the functionality of the application to how the user values the product. Based upon the data collected from analysing different survey methods, *PXI is useful for conducting research during active game development, and looks at collecting information on how design choices contribute to the player experience.* (Abeele *et al.*, 2020).

‘Quantitative Studies: How Many Users to Test?’ (Nielsen, 2012) is a statistical analysis paper that looks at how to achieve statistically reliable data. From Nielsens study they found out that 20 participants could be used to reach a 90% confidance for the study and a 15% margin of error which will provide a large enough data set to discover any trends with the project.

Methodology

This section is here to describe the method's that were used while working on this study.

Project Management

At the planning phase of this study a Gannt chart (Figure 1) was created to help break down time and tasks into a weekly objective. Over time, this chart has changed slightly, giving more time on some areas that were expected to take less time, but from researching, became a more important system to work on. Using this plan, as well as updating it as issues arise so that work can be planned out correctly (Figure 2).

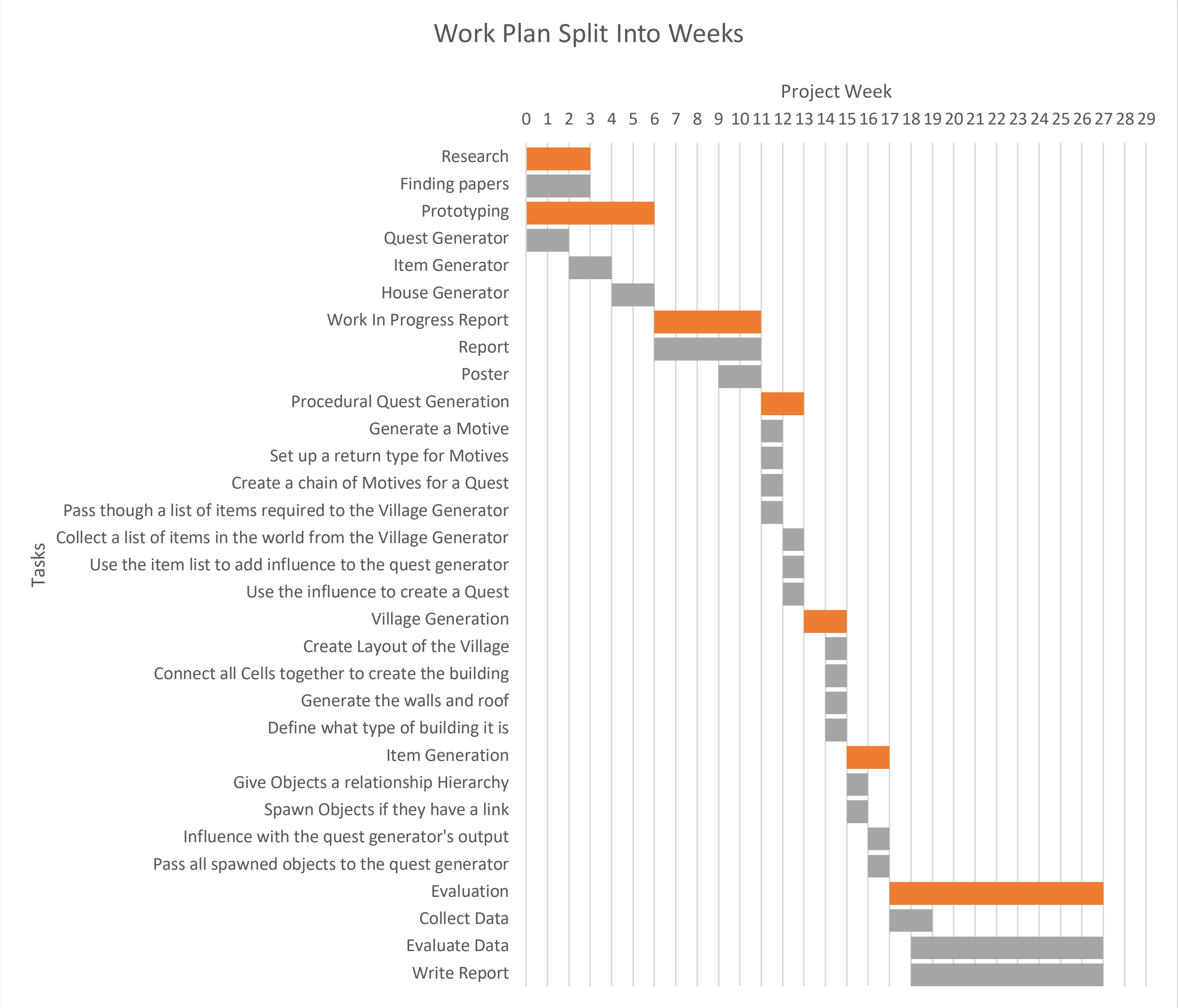


Figure 1 Shows a Gannt chart that was created during the research phase of the project.

There are several reasons why the Gannt chart changed over time, mainly due to the unexpected amount of time that was required to work on generating the village procedurally. This time was needed however to ensure polish to the village, placing buildings close together, districts and testing that the village has enough constraints to re-generate a village that requires more detail.

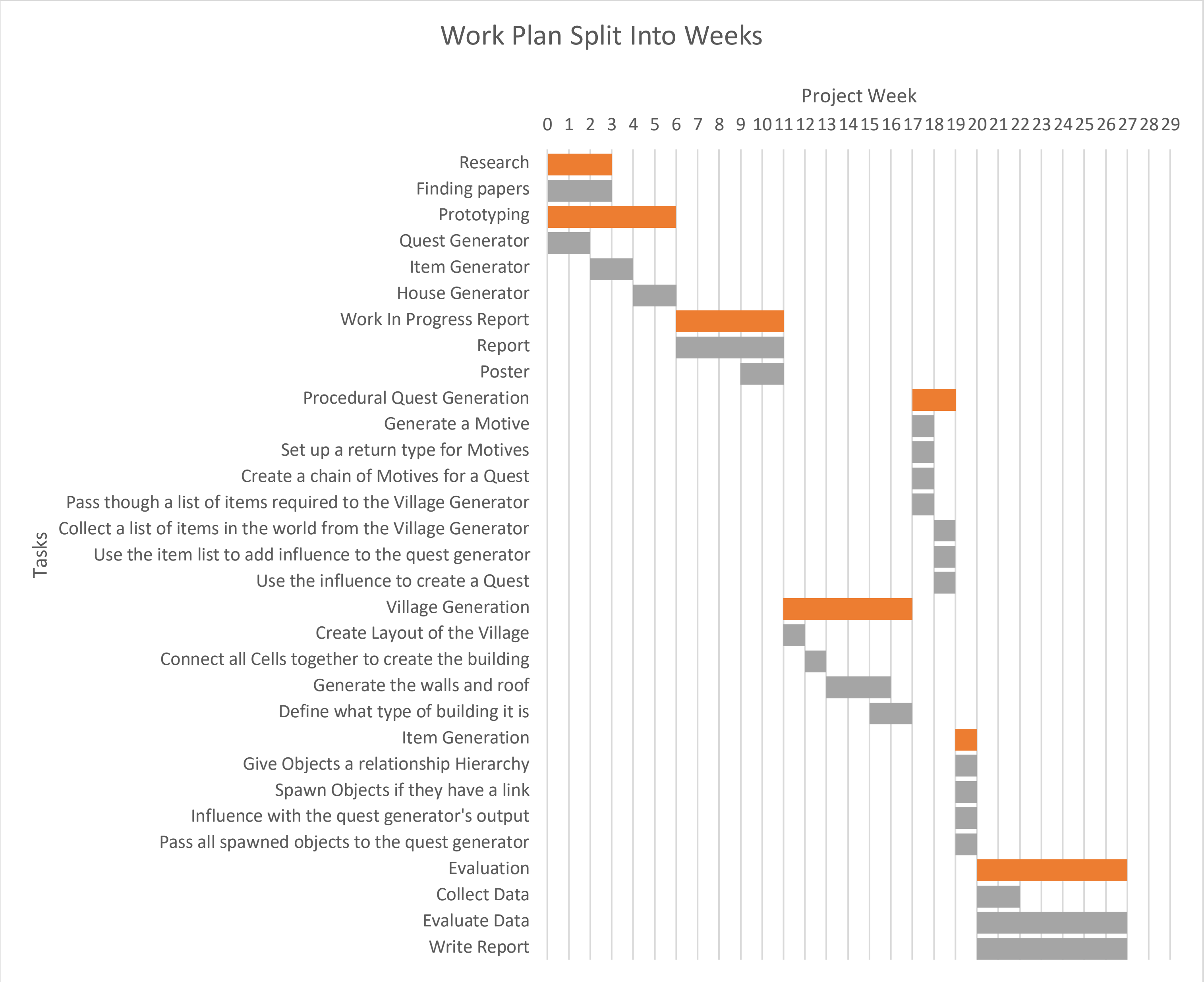
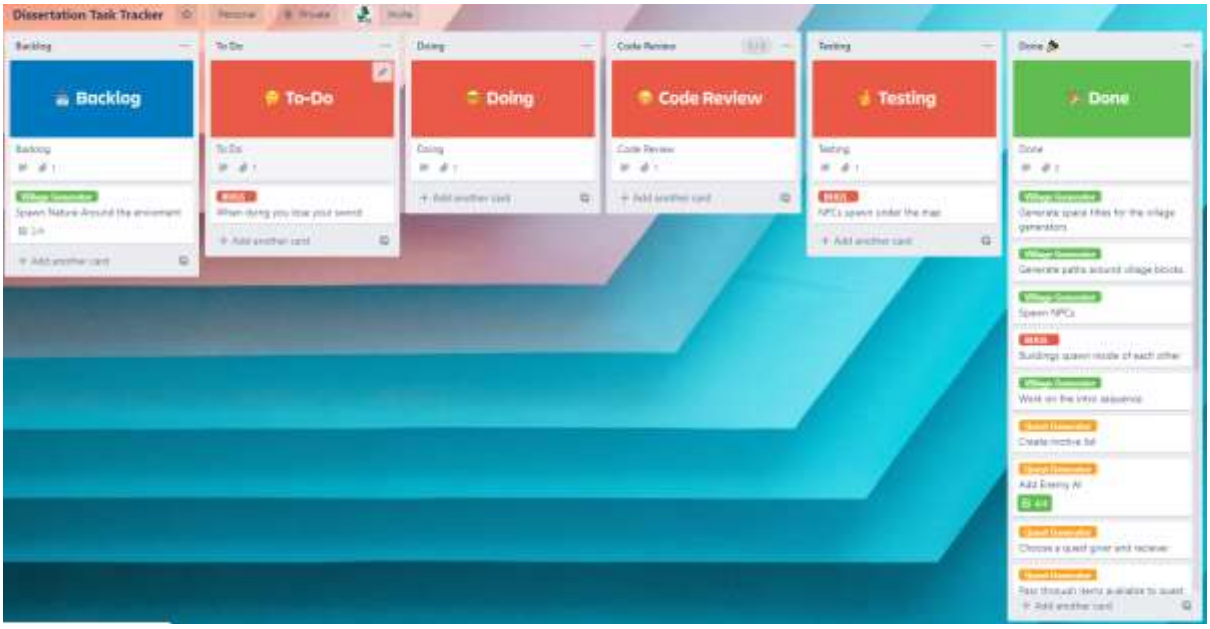


Figure 2 Shows the updated gannt chart at the end of the project.

Trello (Figure 3) was also used to help manage the workload and scheduling of this project. Categories were created for what was currently being worked on, what was complete and what was left to do. Coloured labels were also used to divide each task into groups, such as research, village generator and quest generator. This was a great way to be able to manage what tasks are currently being worked on, what is left to do and what needed reviewing. With Trello we are able to create alerts when categories have too many cards in them. This was used for the code review so that we didn't end up creating bad code which would have affected how productive and efficient the artefact would run at.



Dissertation Task Tracker

Backlog

Backlog

Backlog

Village Generator

Spawn Nature Around the enviroment

1/4

+ Add another card

To Do

To-Do

To Do

BUGS

When dying you lose your sword

+ Add another card

Doing

Doing

Doing

+ Add another card

Code Review

Code Review

Code Review

+ Add another card

Testing

Testing

Testing

BUGS

NPCs spawn under the map

+ Add another card

Done

Done

Done

Village Generator

Generate spare tiles for the village generators

Village Generator

Generate paths around village blocks

Village Generator

Spawn NPCs

BUGS

Buildings spawn inside of each other

Village Generator

Work on the intro sequence

Quest Generator

Create motive list

Quest Generator

Add Enemy AI

4/4

Quest Generator

Choose a quest giver and reciever

Quest Generator

Pass through items available to quest

+ Add another card

Figure 3 Shows the Trello Board at the end of the Project.

A waterfall method was used for this project as it requires each step before it to be completed before moving onto the next step. This was chosen due to the nature of the generation, the village generator and quest generator needed to be linked together, but they couldn't be linked together until one of them worked, so that data can be used to generate the other. As well as this, research into quests and villages had to be done for that research to then be implemented and used as a base foundation to build the artefact up from. Due to this, a Gannt chart was create and update, which laid out a task list and order in which the tasks would be completed in.

Software Development

The artefact was created in Unity, due to the fact that it was created as a study prototype rather than a fully developed game, which a game engine like unreal would have been more suitable for. As well as this, complex mesh building features, as well as features from previous projects, could be drag and dropped into this artefact, modified and used. The path that is generated around houses uses the Marching Squares algorithm, which was taking from a learning project, with slight modification to accept a map size which caused the development time to be subsequently shorter by reusing previous code.

Other game engines were considered during the research process of this project. Godot is a newer game engine that is open source, which has a friendly version control system integrated in it which would be useful when working on a large project like this. However, Godot isn't well documented, and would require trial and error programming which would be a huge risk to this project's completion. As due to Godot's architecture, it is difficult to optimise which would cause issues when procedurally generating large maps. (Slant, 2020). Another engine that was considered was Unreal Engine, which is really well optimised and uses C++, a low-level programming language which can be more efficient than C#. Due to Unreal Engine's workflow however, which is designed around long team-oriented games, it wasn't acceptable. Unity is great at making prototyped versions of games, whereas Unreal Engine is great at making full market value games, which is not what is required for this project and would end in more time spent creating an artefact than is allocated for this project.

Blender was also used to create and modify models. While some models could have been downloaded off of the Unity asset store, some more simple models had to be created by hand. Blender was used due to the fact that it is free to download and has a clean user interface which is easy to navigate. Compared to Maya, which is a more specialised 3D modelling software, which would have produced higher quality models however would have taken longer to create causing the project to go behind schedule.

Tools

There were several tools that I used while working on this project:

- Unity - The game engine the artefact is built on
- Blender - To help model some smaller details for the artefact
- Audacity - To modify some of the sounds that were implemented in the artefact
- Trello - To help plan out and schedule tasks
- Microsoft Excel - To create and manage the Gannt chart as well as for evaluating the collected data in graphs
- Google Forms - Used to collect and ask questions to the user about their experience

Research Methods

There are multiple research papers around conducting research for games studies, the approach that was taken for this study was PXI (Abeele *et al.*, 2020), which is a player experience survey. There are other survey methods available PENS, Players Experience and Needs Survey (Johnson, Gardner and Perry, 2018), and GEQ, Game Experience Questionnaire (Johnson, Gardner and Perry, 2018), which were considered then researching different research methods however GEQ reflects more upon the experience the game creates, and PENS focuses on experiences that motivates engagement.

PXI was used in this study due to the fact it gave a large spectrum of analytic data and confidence levels, which can be used to show how confident my conclusion is, as well as being scalable by Means-End theory, which means that players choose an objective based not just on means but also on the consequences of the action. The study from (Abeele *et al.*, 2020) concluded that PXI "measure player experience at both the level of Functional and Psychosocial consequences" and "provide actionable insight, enabling a better understanding of how game design choices impact the player actions during the runtime of the game, and how they shape emotional responses". The questions used in the study were based from PXI, collecting data of the player experience to the game, focusing on the quest, but also general player experience of the game. Questions are split into several categories, including the meaning of the game, the immersion of the game and feedback of the game, which will focus the collected data on how the player felt while playing the game and the game world. As well as this PXI was designed around how the changes in a game affects player experience, such as what if we made the player faster or jump higher. This style of questioning would therefore work really well for this study as the game design decision we are changing is the generation order which will yield different gameplay results.

Design, Development and Evaluation

This section explains the design and development of the artefact, as well as analysing the data collected and what concussion can be gained from study.

Design Document

This game is a first-person RPG game set in a small village. The village is procedurally generated and split into several areas, the main of which would be a castle / keep, liveable houses and ruins. The town is filled with NPC's and when necessary, will provide monsters for the player to destroy. The purpose of the game would be to complete the generated quest by fulfilling each step of the quest.

Village Generator

The Village generator is one of two systems that are part of the game. The village generator is responsible for generating the buildings, roads and controlling the NPC's that are active around the village. This village generator will use the method explored by Williams and Headleand in their settlement generation study explored earlier in this paper. (Williams and Headleand, 2017). When designing the size of the village, its overall size should make it so to get from one side of the village to the other, it will only take the player a maximum of 1 minute, which works out as 20-unit radius in Unity. The village should also be able to control how sparsely or densely populated an area is, so that the centre and surrounding areas of the castle are packed whereas the outer edges are more spaced out. This value of densely or sparsely packed areas will be linked to the district areas that are created.

There are several districts that should be included. A castle district should include the keep as well as surrounding buildings that are for fortification. Liveable district will include houses in which the NPC's are assigned as their home. These districts should be placed around the centre of the village. Shops / stalls should be located at the centre of the city and be less common at the outskirts to closely match the distribution in villages. The final district are ruins, which are located around the outskirt of the village. This district are where monsters will come from when the quest requires it.

District area	Location in Village	How packed are the buildings
Castle	Centre	Densely
Houses	Around Centre	Densely – Sparsely
Shop / Stalls	Centre	Densely
Ruins	Outskirts	Sparsely

Table 2 Shows an overview of the districts in the village

Roads are generated around the outskirts of each building group, and then those chains should be joined together at its closest point to create a loop of pathing for the player and the NPC's to use. The path will be generated using the marching square algorithm to define the paths shape.

Quest Generator

The quest generator is the second system of the game, delivering the quest to the player and dealing with the players actions at each step to determine whether to proceed or to wait for more player interactions.

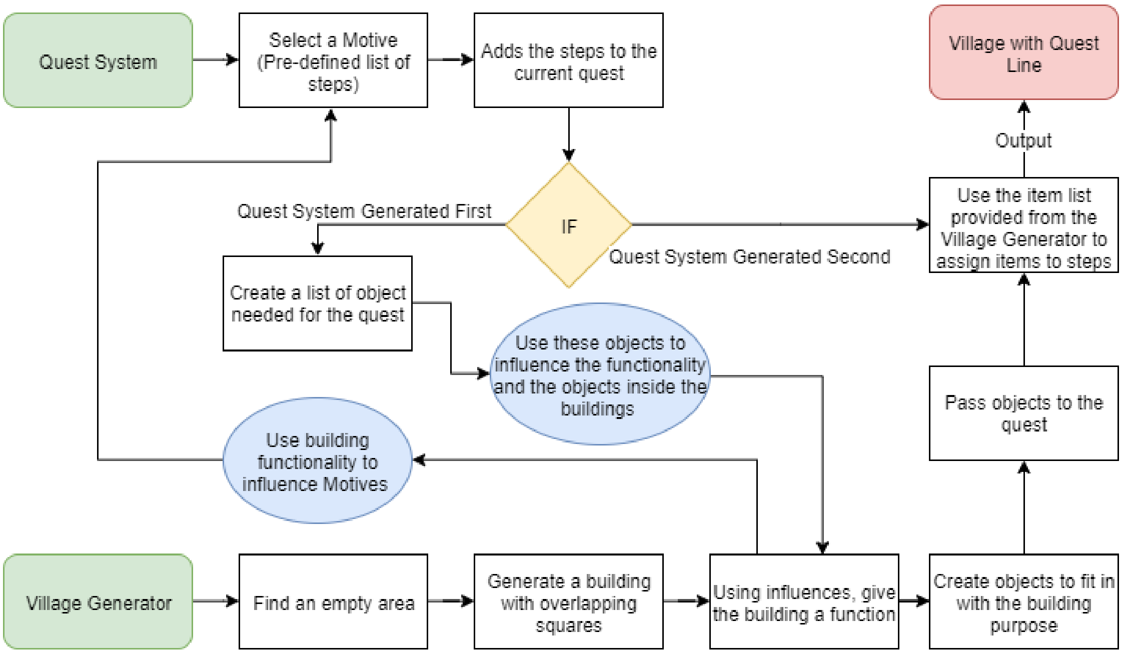
Each quest is based upon a motive, where a motive holds steps that is used to generate the quest. (Doran and Parberry, 2011) These steps fall into several categories described by Doran and Parberry which refer to its generation grammar. These categories include go to, get, kill and talk, which can be used in a combination of orders to generate quests.

There is also a configuration stage, where the quest generator pulls in information from the village generator, about objects, location, NPC and items that have been spawned, and the selects which item / items will be used for the quest that the player is going to play. This can be done through either pure seeded randomness or can be done through checking the distance from the location of the quest before it so that it is not too close or two far away.

The quest generator will also be responsible with updating the quest completion, and so will have to handle spawning trigger boxes for when the player has entered a specific location, spawning monsters when the player is close to the ruins and the quest line has set the player up to be ready to fight monsters, and when the player interacts with objects and NPC's.

The Interaction Between Village Generator and Quest Generator

These two systems need to be able to interact with each other and form some way of being able to influence outcomes from the system. Figure 4 shows how the two systems will interact, and at what stage data is passed between these two systems.



Quest System
Select a Motive (Pre-defined list of steps)
Adds the steps to the current quest
Village with Quest Line
Output
Quest System Generated First
IF
Quest System Generated Second
Use the item list provided from the Village Generator to assign items to steps
Create a list of object needed for the quest
Use these objects to influence the functionality and the objects inside the buildings
Pass objects to the quest
Use building functionality to influence Motives
Village Generator
Find an empty area
Generate a building with overlapping squares
Using influences, give the building a function
Create objects to fit in with the building purpose

Figure 4 Shows how the two systems will interact with each other

Terrain

Terrain will be used as a way to break up the flatness of the village, however, will be generated after the village has been generated in areas that is not covered by either a building or a path. The terrain will be generated using simple layered Perlin noise to create hills and depth in the landscape around the village, with any terrain outside of the village radius being have some value changed to create larger hills. Trees will also be generated, choosing between two types of trees. Inside the village, cleaner, more pruned trees will generate, whereas outside the village, a rougher and more natural tree will be generated. At the side of the path, there will be some green space separating the path to the buildings. This space should be used for grass and flowers to again, help break up the flatness of the village.

NPC's

Having NPC's are a pretty important part about keeping the narrative of the game feeling right. The NPC's need to have a feeling of realism to help keep the illusion of interactivity between the quest narrative and the players experience. To achieve that there needs to a relationship between each NPC (Szymanezyk and Cielniak, 2010). As part of this, there is several actions that each NPC group will include:

Wandering ~ Randomly walking around the map following the curve of the road.

Patrolling ~ Walking around a district or in a specific loop.

Walking ~ Picking an end point and walking to a desired specific location.

As well as this, a distribution system that can spread the NPC's across the map or into a specific area would be useful and can be used to link into the quest system so that NPC's run away from monsters when they spawn.

Player Controller

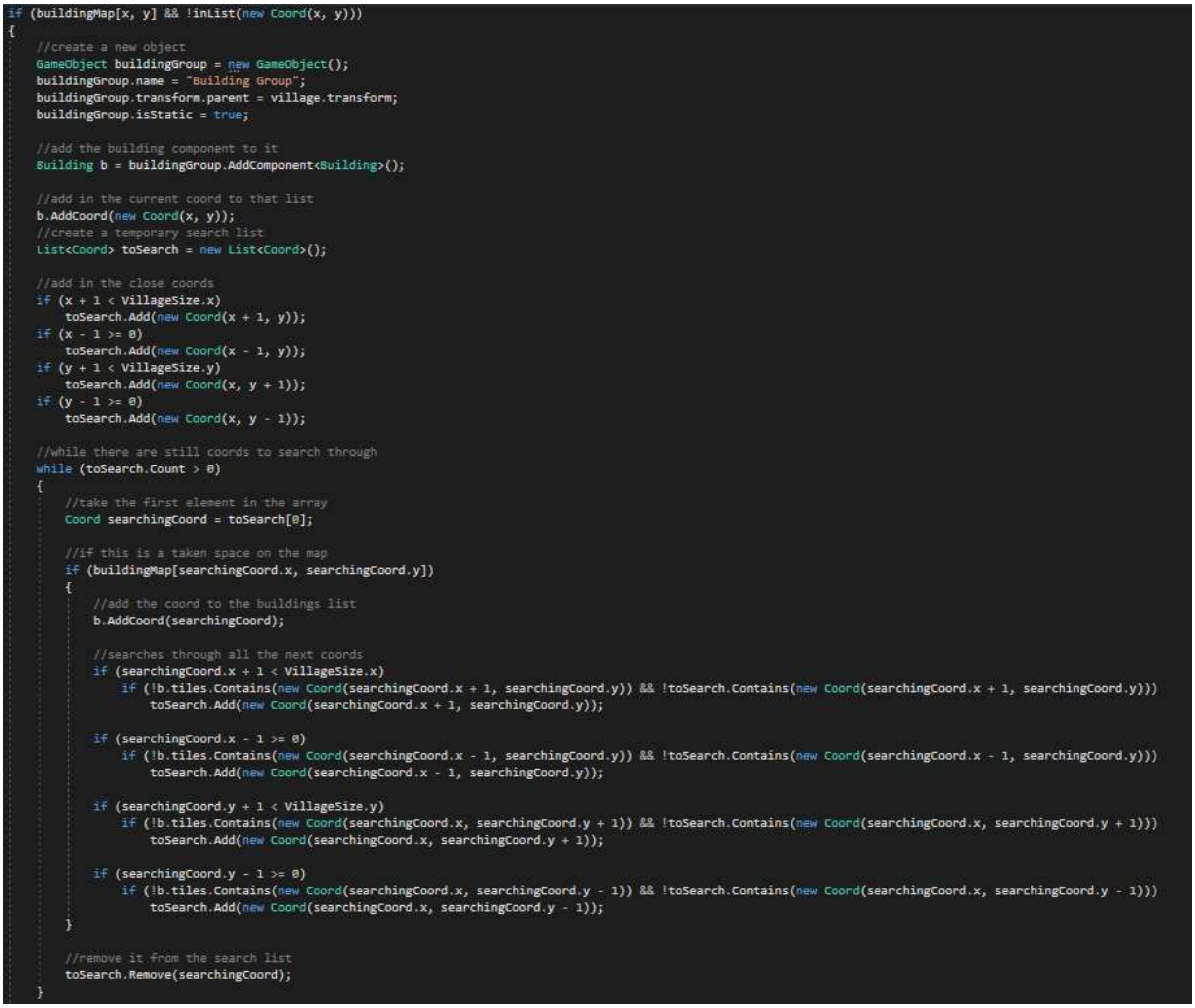
The player controller is important to get right as it is the way that the player interacts with the world. The controller will be a standard First-Person Controller, using the mouse to look around and WSAD to move around the map. There is several interactions the player needs to be able to complete too. Left mouse button will be used for attacking, and E will be used to interact with items / NPCs (picking up /using / talking).

Development

The artefact had 5 weeks planned for its overall development, however it ended up overrunning by 3 weeks due to under expecting how long some of the tasks would have taken.

The Village

Building blocks were first implemented for the village, where square shaped areas are created before a smoothing algorithm is applied creating a more defined shape of each building zone. Once a zone is generated, the edges are determined and buildings are spawned around the edges of each block, calculating the amount of free space available then picking a building from a list that will fit into the space. Each block has its own constraint as to how filled up each block can be, so that ruins are more sparsely filled compared to the houses area which are densely filled. Figure 5 shows a code snippet taken from the space searcher part of the village generator. It finds connected active cells and adds them to a search list, which then returns a full list of cells that are formed as the district for the buildings to generate in.



```
if (buildingMap[x, y] && !inList(new Coord(x, y)))
{
    //create a new object
    GameObject buildingGroup = new GameObject();
    buildingGroup.name = "Building Group";
    buildingGroup.transform.parent = village.transform;
    buildingGroup.isStatic = true;

    //add the building component to it
    Building b = buildingGroup.AddComponent<Building>();

    //add in the current coord to that list
    b.AddCoord(new Coord(x, y));
    //create a temporary search list
    List<Coord> toSearch = new List<Coord>();

    //add in the close coords
    if (x + 1 < VillageSize.x)
        toSearch.Add(new Coord(x + 1, y));
    if (x - 1 >= 0)
        toSearch.Add(new Coord(x - 1, y));
    if (y + 1 < VillageSize.y)
        toSearch.Add(new Coord(x, y + 1));
    if (y - 1 >= 0)
        toSearch.Add(new Coord(x, y - 1));

    //while there are still coords to search through
    while (toSearch.Count > 0)
    {
        //take the first element in the array
        Coord searchingCoord = toSearch[0];

        //if this is a taken space on the map
        if (buildingMap[searchingCoord.x, searchingCoord.y])
        {
            //add the coord to the buildings list
            b.AddCoord(searchingCoord);

            //searches through all the next coords
            if (searchingCoord.x + 1 < VillageSize.x)
                if (!b.tiles.Contains(new Coord(searchingCoord.x + 1, searchingCoord.y)) && !toSearch.Contains(new Coord(searchingCoord.x + 1, searchingCoord.y)))
                    toSearch.Add(new Coord(searchingCoord.x + 1, searchingCoord.y));

            if (searchingCoord.x - 1 >= 0)
                if (!b.tiles.Contains(new Coord(searchingCoord.x - 1, searchingCoord.y)) && !toSearch.Contains(new Coord(searchingCoord.x - 1, searchingCoord.y)))
                    toSearch.Add(new Coord(searchingCoord.x - 1, searchingCoord.y));

            if (searchingCoord.y + 1 < VillageSize.y)
                if (!b.tiles.Contains(new Coord(searchingCoord.x, searchingCoord.y + 1)) && !toSearch.Contains(new Coord(searchingCoord.x, searchingCoord.y + 1)))
                    toSearch.Add(new Coord(searchingCoord.x, searchingCoord.y + 1));

            if (searchingCoord.y - 1 >= 0)
                if (!b.tiles.Contains(new Coord(searchingCoord.x, searchingCoord.y - 1)) && !toSearch.Contains(new Coord(searchingCoord.x, searchingCoord.y - 1)))
                    toSearch.Add(new Coord(searchingCoord.x, searchingCoord.y - 1));
        }

        //remove it from the search list
        toSearch.Remove(searchingCoord);
    }
```

Figure 5 Code snippet which shows how areas are connected for building districts

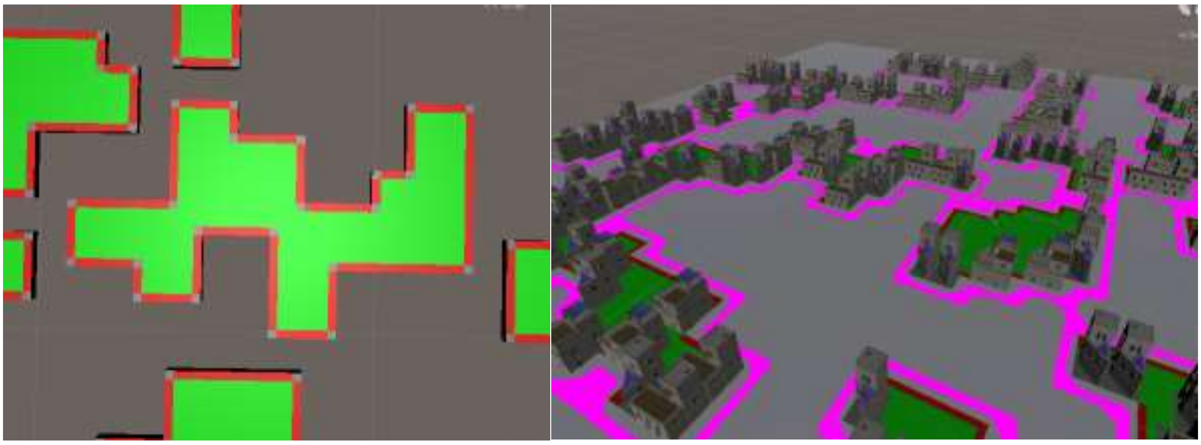


Figure 6 Shows each district area (left) and the beginning of building generation (right).

The number of regions has been condensed down to 3 areas, removing the shop / stall area from the design document. This was due purely to time restraints, however having another area will not change the way the artefact is to be played. As seen in Figure 6, there is a clear divide between areas. A castle that is located near the centre, surrounded by houses, and then around the outskirts of the village is several ruins.

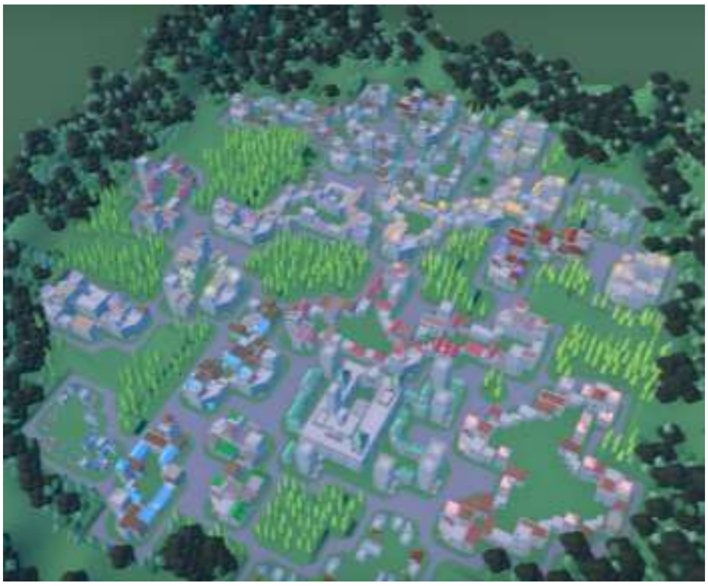


Figure 7 Shows the village that is generated

Every map is created with a set of options of possible spawn positions of unique buildings such as a castle, which is large in size. If the map requires a castle, and a castle cannot be spawned, then the map would be reloaded and recreated till a castle has a valid location to spawn.

Paths are generated around the empty space of buildings, creating a loop around the building block using the method described in 'A timeline approach for the generation of simulated settlements' (Williams and Headleand, 2017). These loops end up hooking together in some places, however sometimes a building block is too far away from other blocks. In these situations, every path is evaluated to find the longest path that has been created, and then all connected paths that aren't in the main path, the closest point to the nearest path is connected together. This looping path means that NPC's are accessible to all areas across the map.

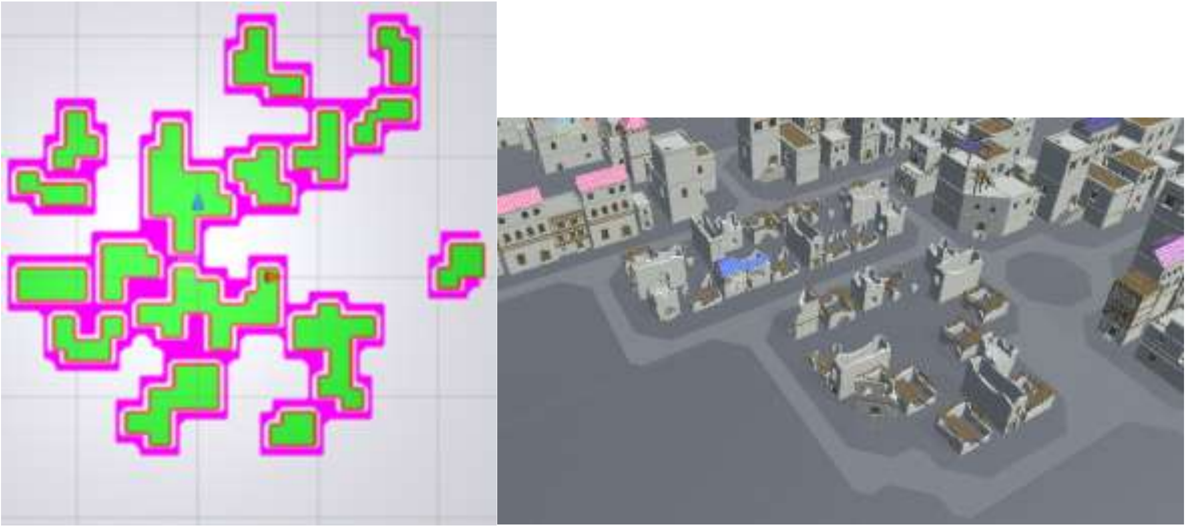
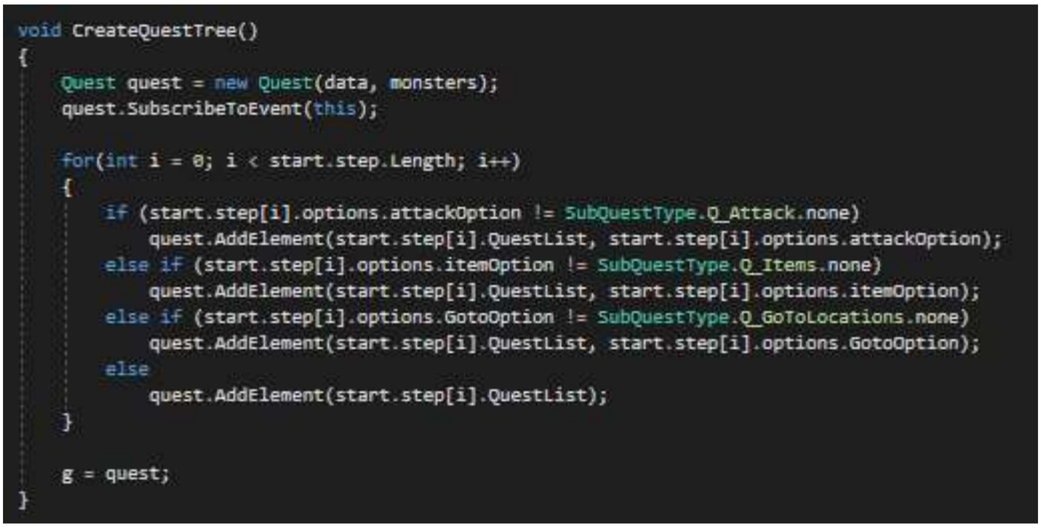


Figure 8 Shows how paths around areas started (left) and how they look now (right).

Quests

Quests are generated off of a motive, which describes the actions which occur in a quest. These motives are meant to be linked to an emotional in game value between NPC's, however due to the small nature of the game, a motive is selected at random out of the pool and that motive is used as the relationship between the quest giver and the rest of the NPC world. This was due mainly to the fact that creating an entanglement of NPC's relationship would have never been used as the game revolved around a single quest rather than generating a several quests, where a relationship would have been seen between NPC's. (Doran and Parberry, 2011) (Szymanezyk, Dickinson and Duckett, 2011). Figure 9 shows the set up for a quest tree, where a motive is taken in and its steps translated into data for the node tree to process. This node tree uses the research from 'Toward supporting stories with procedurally generated game worlds' (Hartsook *et al.*, 2011) to set up the quest in a spatial way so that it can then be used for the village generator later.



```
void CreateQuestTree()
{
    Quest quest = new Quest(data, monsters);
    quest.SubscribeToEvent(this);

    for(int i = 0; i < start.step.Length; i++)
    {
        if (start.step[i].options.attackOption != SubQuestType.Q_Attack.none)
            quest.AddElement(start.step[i].QuestList, start.step[i].options.attackOption);
        else if (start.step[i].options.itemOption != SubQuestType.Q_Items.none)
            quest.AddElement(start.step[i].QuestList, start.step[i].options.itemOption);
        else if (start.step[i].options.GotoOption != SubQuestType.Q_GoToLocations.none)
            quest.AddElement(start.step[i].QuestList, start.step[i].options.GotoOption);
        else
            quest.AddElement(start.step[i].QuestList);
    }

    g = quest;
}
```

Figure 9 Quest Tree Code Snippet

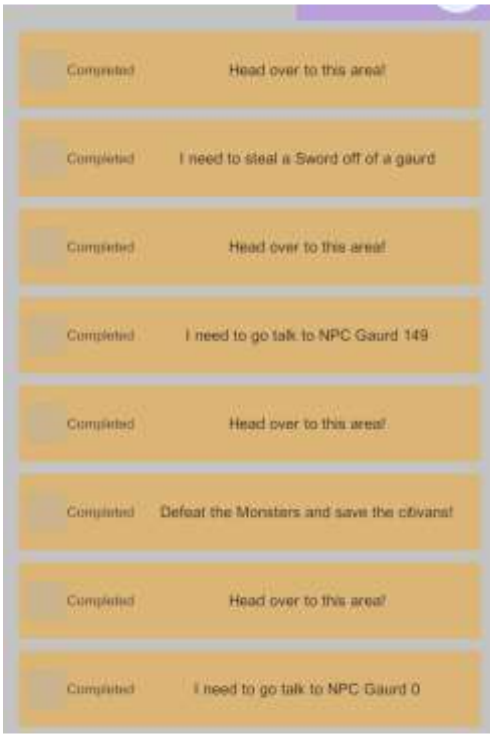
There are 6 main motives types:

- Comfort
- Knowledge
- Protection
- Reputation
- Serenity
- Wealth

These are the building blocks for a relationship between others and gives an overall cover of any type of relationship that NPC's can have to each other.

There are several different types of sub quests that are being defined in this game. These all come from the research from (Doran and Parberry, 2011) which looked at hundreds of quests and categories each goal. These categories have been used because they are the base foundation to any quest and make translating motives into steps more generalised while still being able to add extra detail on top of the general steps if need be.

The motive selected is translated into a list of sub quest steps, and then searches for data to populate the sub quest criteria. For example, a sub quest of type attack, searching for a ruined location to spawn monsters in, and then assigns a call back function what once monsters in that area are killed. This is repeated with all other sub quest types. These sub quests are key node points, created to help lead the narrative of the quest. (Hartsook *et al.*, 2011)



Completed Head over to this area!

Completed I need to steal a Sword off of a gaurd

Completed Head over to this area!

Completed I need to go talk to NPC Gaurd 149

Completed Head over to this area!

Completed Defeat the Monsters and save the citivans!

Completed Head over to this area!

Completed I need to go talk to NPC Gaurd 0

Figure 10 Shows a Quest generated in the game

NPCs

Initially NPC all spawned in one area while setting them up to co-exist with the village and quest generator. They would randomly wonder around the map.

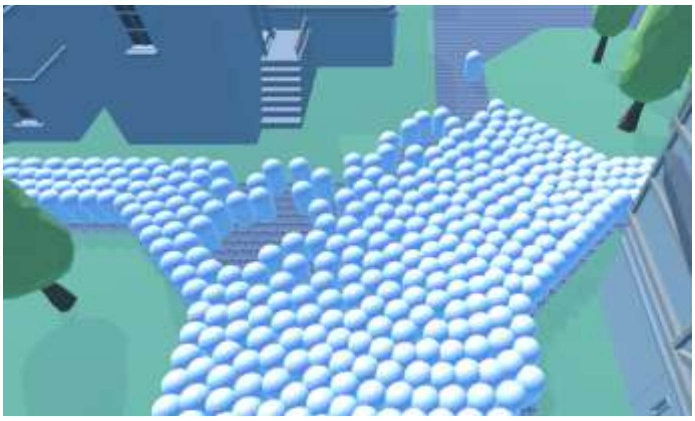


Figure 11 Shows how NPC's started out.

NPC’s have several properties to help create a realistic and believable crowd. Each NPC has relationships to other NPC’s in which they are grouped with. These groups vary with different NPC types. Guards will have larger groups, compared to citizens which has a range of small groups of 1 or 2 people to larger groups of 5 or 6. (Szymanezyk and Cielniak, 2010)

NPC Type	Min Group Size	Max Group Size
Nobel's	1	2
Guards	4	8
Citizens	1	6

Table 3 Shows the min and max sizes of groups for each NPC type.

Groups request a location to head towards from the NPC manager, which evaluates the current distribution of NPC's that are across the map and populates areas that have a lack of NPC's. This system can also be used to create a high-density area of NPC's in a single sector, however because of the small scale of this map, this is not used.

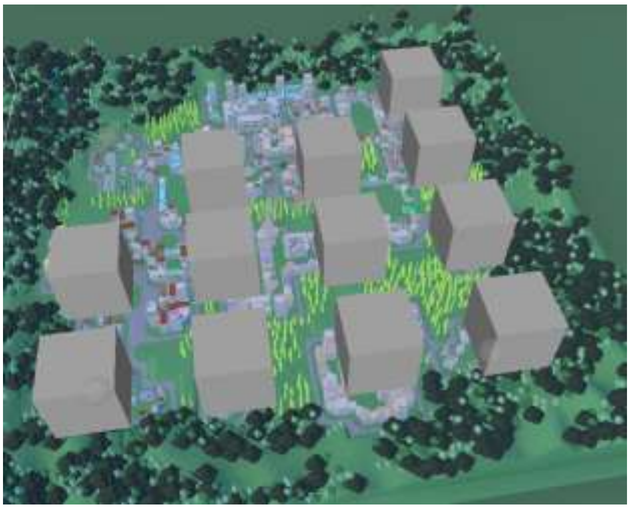


Figure 12 Shows a visualisation of the distribution system which keeps track of the position and density of NPC types in an area.

Nature

Trees are distributed around the map to help conceal its boarder of the map as well as to obstruct lines of sight in open spaces inside of the village itself. Trees use noise to determine whether they can or cannot grown in an area, as in (Williams and Headleand, 2019) which states that while viewing from a first-person perspective there is no difference to how trees look for clustering and simulation compared to using noise. Noise was therefore used as it was computationally quicker for the CPU than simulation tree growth and at the stage of implementing tree generation there was already issues in frame rate drops and thus a simple solution was necessary.

Terrain was added to help make the village feel less flat. A building map is passed into the terrain generator, which cycles through each point and sets its height equal to layers of Perlin noise or 0 depending whether a building is there or not. This terrain would have no impact on the overall experience of the study.

Birds were also implemented by using boid code taken from Sebastian Lague to creating a feeling of immersion within the game world (Lague S, 2019), adding small details like birds flying around the map to bring everything together. Birds flock around the player, so that a small number of boids only had to be used helping to reduce FPS issues that were occurring in the project.

Research

The study was designed to be self-contained in the game, so that any participant could just be given the game and need nothing else to be able to partake in the study. To do this, the questionnaire had to be created and completed online, and Google Forms was chosen for this due to its simplicity and setting up multiple forms. It was also decided to create two different forms that contained the same questions, where one form would be for having the village generate first, and the other would be for having the quest generated first. This would make it easier later on down the line, to compare the two sets of data, while not letting the participant know about which generation order they have.

There also needed to be a balance between how many players played the village generation first vs those who played the quest generation first, to remove any bias from the first generation they got, and this should be as close to a 1:1 split in which order they got first. Therefore, the participants gives the game their ID number, and if this number is odd, they play the village generation first, and if it's even they play the quest generation first. This would achieve a 1:1 split in which generation order participants play first with the 20 participants that are required for this study.

When answering the survey at the end, participants would give consent to use their data for this study. Participants would be able to withdraw consent at any point by using their ID and sending an email.

A step process was created to follow in order to give each participant the same information and format to complete the study in.

1. Talk to the participant about what the study is, how they will play through two scenarios, and after each scenario a questionnaire will appear for them to answer. During this phase too, explain the game controls, and how to play the game.
2. Allow time for each participant to answer any questions they may have from step 1.
3. Give each participant their unique ID and get them to enter it into the input field on the main menu.
4. Observe the participant as they play through the first scenario, only intervening when they are absolutely unsure of what to do.
5. Once the first scenario is done, a questionnaire will appear. Give the participant space to answer this honestly without the feeling of being watched.
6. Once they've completed the questionnaire, instruct them to close the webpage down, and click done, progressing the game onto the second scenario. Again, observe the participant as they play, and intervening when necessary.
7. After this scenario is complete, the final questionnaire will appear, again give them space to answer this without feeling watched.
8. Once done, thank the participant for participating in the study.

This step order was created so that ever participant was treated equality, and that there would be no bias in the data.

The Results

15 people were able to participate in this study, which is less then what was recommended by Nielson (Nielsen, 2012), and while this will affects the overall accuracy and reliability of this study, it has yielded some useful results. Due to having less participants, there is a 20% margin of error with 90% confidance.

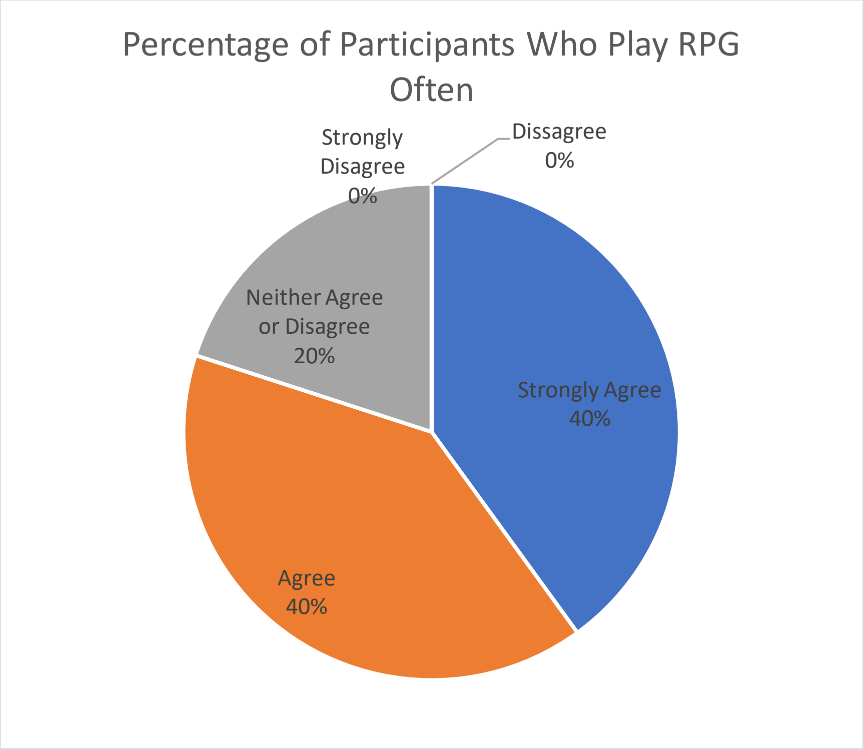


Figure 13 Shows the percentage response of whether participants play RPG games often

The first thing to analyse is the distribution of people who play RPG games. 80% of participants said they play RPG games often, with the remaining 20% neither agreed nor disagreed with this statement. (Figure 13) The large majority of people who partook in this study play through quests and understand how quests work. This remains a vital point for this study, the results do not reflect from an average casual gamer, but more from the viewpoints of people who play this genre often and are more critical to what is a good and bad narrative and quest line.

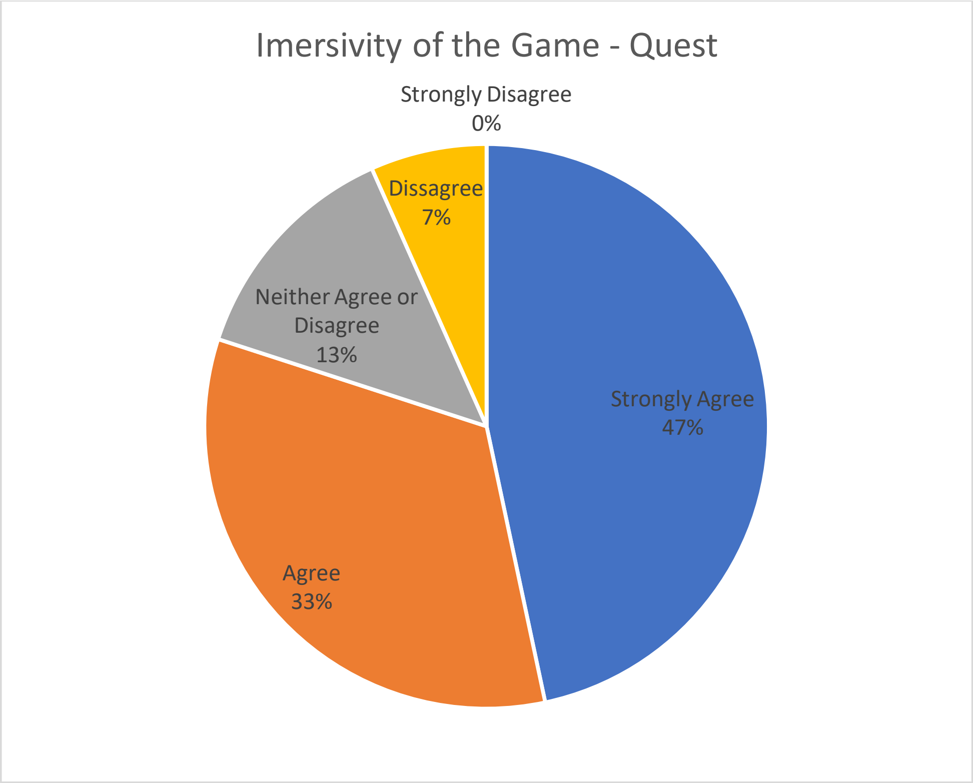


Figure 14 The interactivity for Quest Generation first.

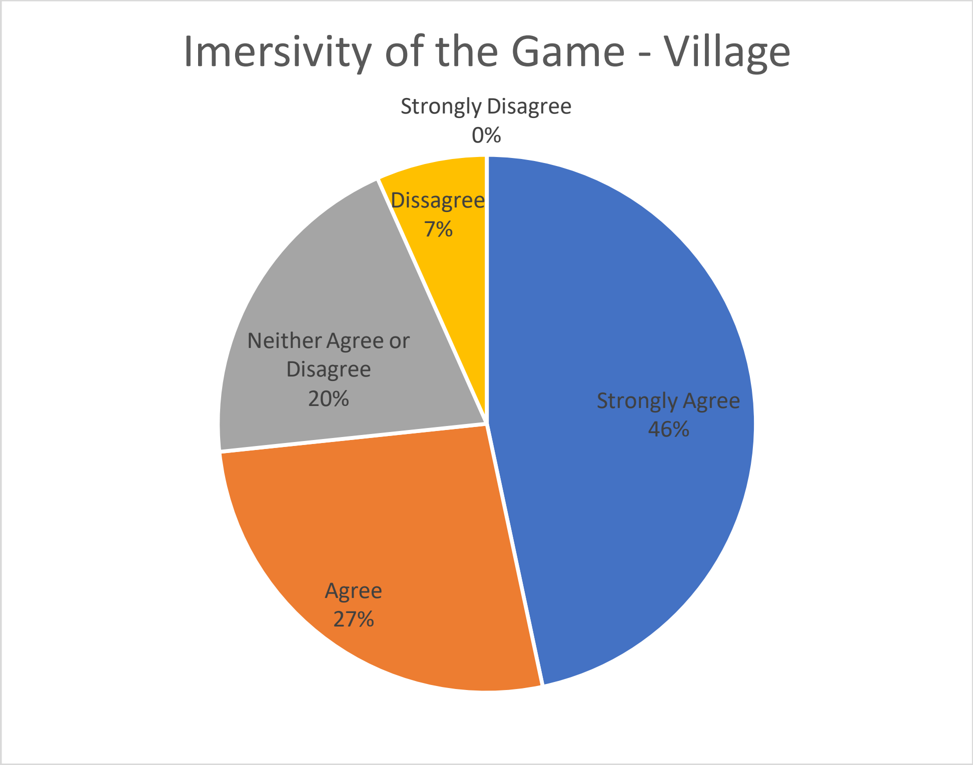


Figure 15 The interactivity for Village Generation first.

Participants also felt immersed while playing the game in both scenarios, 73% of participants were immersed in the village generation and 80% felt immersed in the quest generation, which is the equivalent of 1 participant difference. (Figure 14, Figure 15). Overall there is a high level of interaction between the players in both generations' cycles, and we can therefore at least conclude that players are feeling engaged into the game.

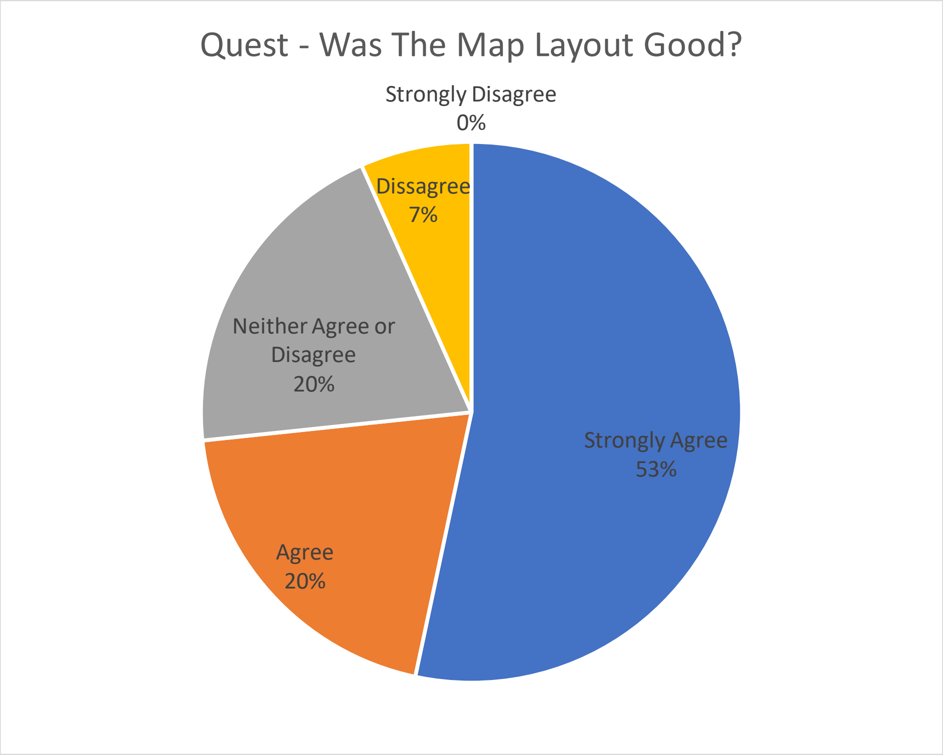


Figure 16 How good was the map you played in.

This immersion can be linked to the village that was created, due to how similar the data looks. As the map is procedurally generated, there is a chance that the participant received a bad map, and that would have been a factor to consider as to how immersive the game would be. An example of this is seed 13 (Figure 17) which generation cycle shows 2 isolated building clusters, with forest generated in-between them. A generation such as this would hinder the immersion and feel of the whole game so there was a small issue with the constraints used in the village generator. While this is only a small factor, there is a clear link between the map that the player interacts with and how immersive it was to the participant.

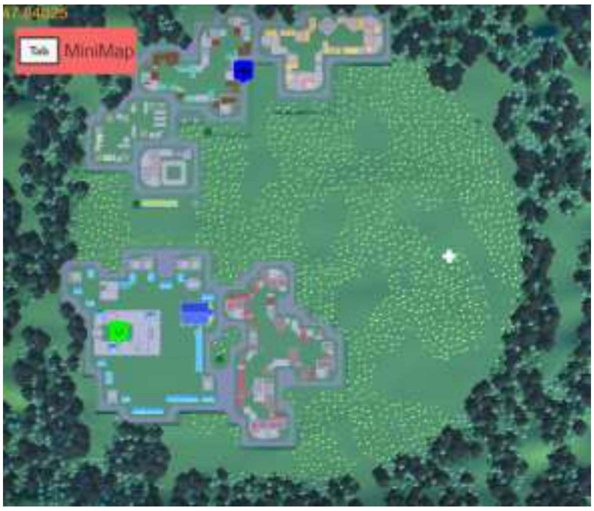


Figure 17 Shows a badly generated map.

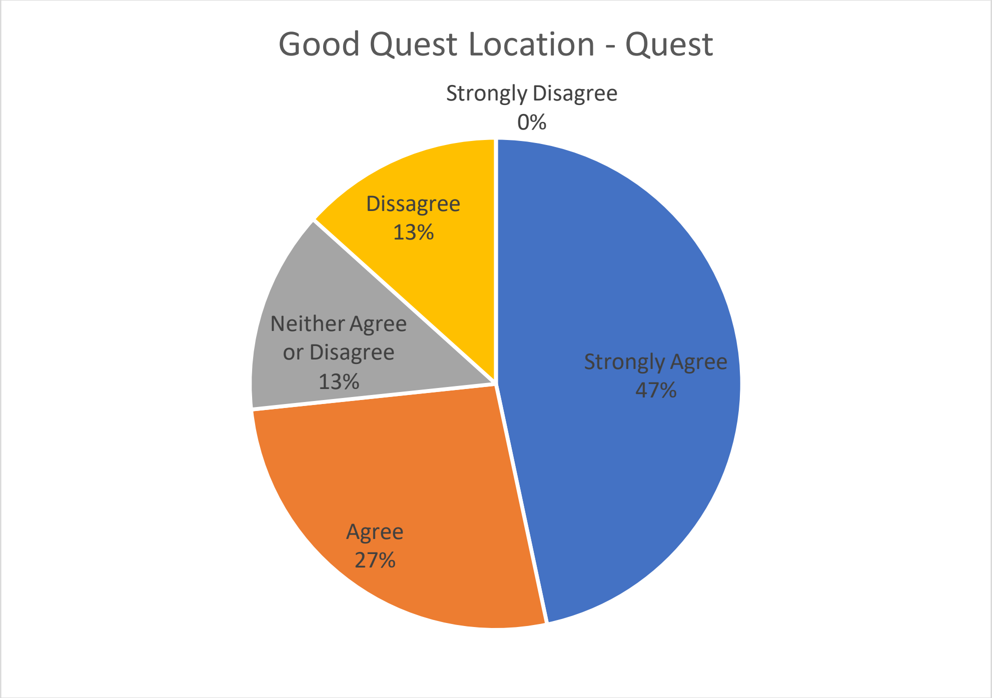


Figure 18 Shows how the participants felt about the quest distribution across the village.

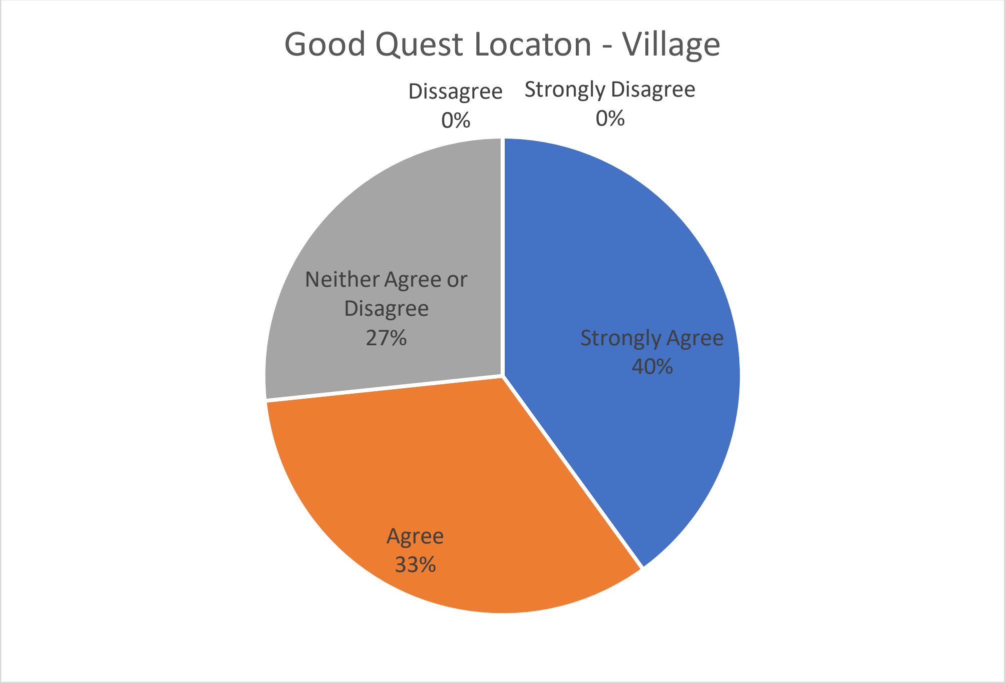


Figure 19 Shows how the participants felt about the quest distribution across the village.

Overall there is a strong connection between the how 'good' participants felt the quest was distributed across the map, making use of all the space available, and how immersed the game felt to the player.(Figure 18, Figure 19) A quest played in a small area, without making use of the whole map, felt underwhelming, where as a quest that pulled the player around the whole map to explore felt more interactive and immersive. Both data sets, for immersivity and good quest location are very similar in its distribution, with both Quest data sets having the exact same percentage distribution for the Agree and Strongly Agree category.

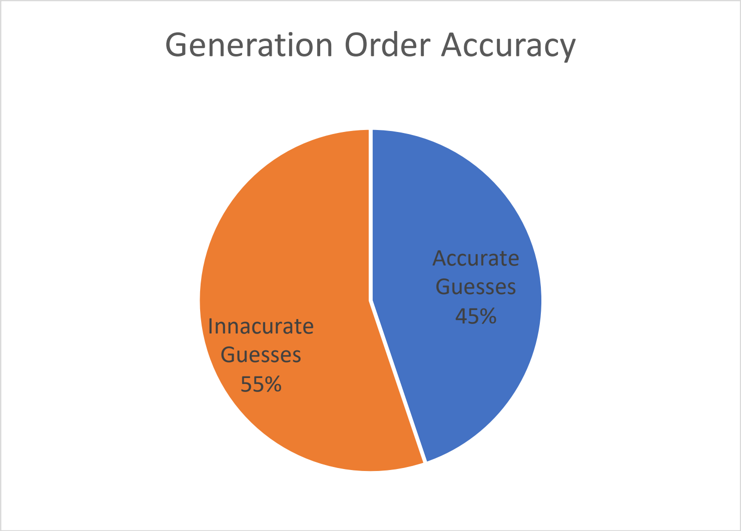


Figure 20 The total accuracy of guessing the generation order.

Finally, 45% of participants could correctly identify the generation order used to generate the game world but looking into the data also shows the 67% also voted for the same generation order in both scenarios. This indicates that participants were unsure as to which order they were playing, and often would doubt themselves on the second scenario, voting for the same generation order as before.

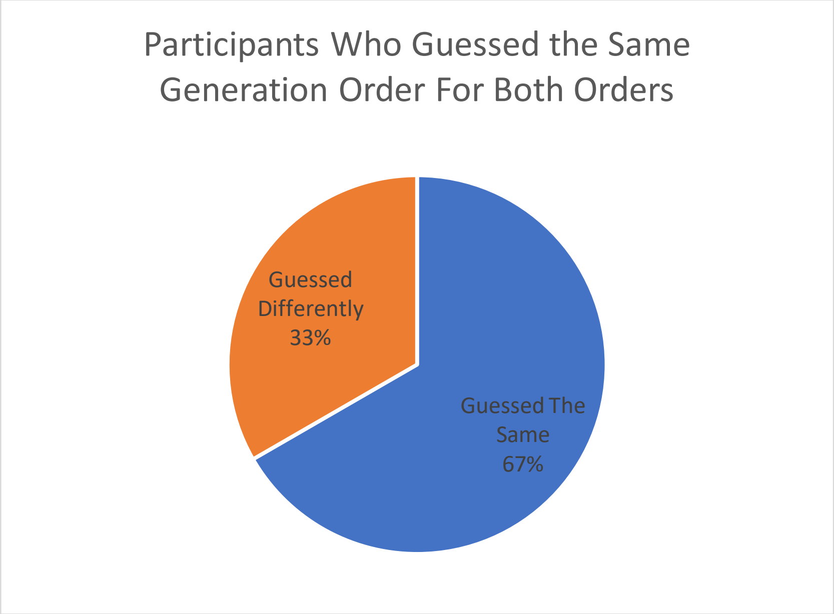


Figure 21 Participants who guessed the same generation order twice.

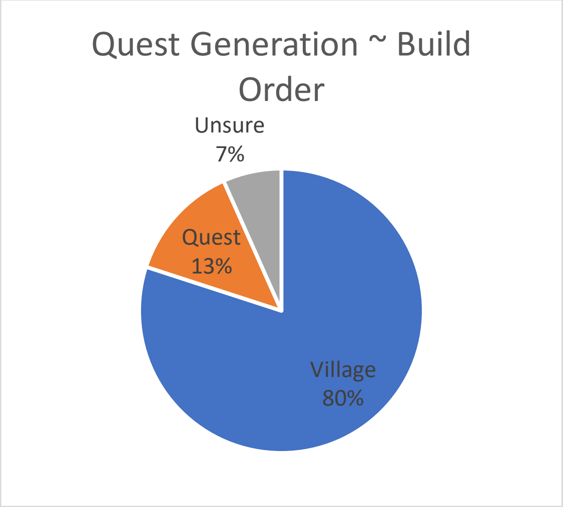


Figure 22 Participant guesses for the build order for Quest generation first.

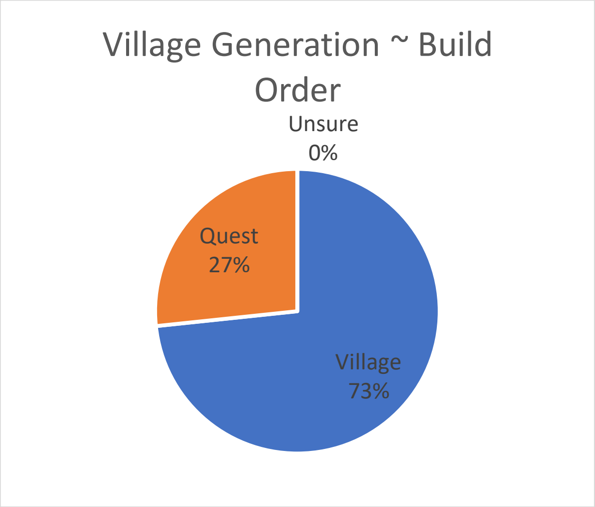


Figure 23 Participant guesses for the build order for Village generation first

From the data, there is a clear bias towards Village generation as the build order for the world in both cases with more than ¾ of participants. This could be due to how visual the city is compared to the quest line. While there is visual indicators as to where to travel to, what to pick up and what to do, they are often dwarfed by the towering village around the player, as well as an eye captivating intro scene of the village rising from the ground.

Project Conclusion

From the data collected, there is no clear correlation between whether the order in which content is generated, showing that only 45% of participants could correctly guess the generation order, with 66% of participants also picking the same generation order. This accuracy of 45% is mainly due to the fact that 76% of generation guesses were that the Village was generated first, and only 20% thought the quest was generated over both studies. This therefor concludes that there is no data to really support the question, resulting in a null conclusion. There is a case to be made that this projects data is less reliable due to its small sample size, however, the data that has been collected does show a very specific trend.

From this data, it is clear that there is no noticeable difference between the order in which content is generated. Each objective of this project was fulfilled, we created a working village generator and quest system that interact with each other while generating content, but participants could not notice any difference between the two orders of generation.

Overall the artefact created was effective, with it being immersive and providing players a clear feeling of freedom while leading them through a quest, however there are other approaches that could have been considered when developing the application. A village in a pure data term is nodes (road cross sections) connected together into a web. Quests can also be expressed in terms of nodes, but in a liner form. (Hartsook *et al.*, 2011). By overlapping a liner node graph onto a web of nodes, there is multiple paths that can be made to get to the same end goal, which can ruin the quest chain. If this was to be put into more linearly progressive generator, such as a dungeon generator, where each room is connected to only another room, overlaying a quest node graph, or using the node graph would have generated rooms which had more off an effect in the order and meanings behind the purpose of the room for the quest, which would of heavily influenced the narrative. This idea is also supported by Nitche and Ashmore who created a land generator that generated enclosed sections, creating a liner path through their game world, compared to an open world.

There is also some questions that could have been asked that we're not. Due to the fact that every level was created differently for each player, there is no clear way to be able to determine that generation order x is better than generation order y, as every participant received something different, which in some cases actually created worse scenarios which would then put that generation order at a disadvantage. To approach this issue, it might be worth conducting an interview, to learn more about what was wrong and why this could have been a factor. Completing this in a pre-test could then lead to improving the current solution to be more refined, however this just was not possible due to how short this projects time frame is.

Reflective Analysis

Reflecting on a project like this is often difficult, there is so many moving parts and features that could be improved upon. Overall, I'm happy with the way this study has turned out, as well as the artefact that has been created. I've learnt a lot about my own skills, as well as improving upon some of the things I find most enjoyable, such as the village generator, which has been the most enjoyable part of working on this project. There are still some issues with the current development, such as some spacing between buildings, which are so small it wasn't worth fixing for the study, though to fix this issue I would need to model each building into a select size scale.

I feel like I personally could have put more work into the quest generator, as it stands, its functional rather than efficient. I would have loved to have added a sentence generator to form the task description, even something like a Markov Chain, however 6 months to research, develop and evaluate would of made it difficult to add small details into the artefact and I could not fit everything I wanted to do for this project. Therefor some things had to be missed out or removed so that something complete could be used to conduct research and allow time for evaluation of this data.

Some things that didn't go well occurred during the data collection stage. The current situation around the world when writing this, Corona-virus pandemic, which started just as I had begun collecting data. I managed to get 15 participants before the pandemic really caused an issue, however the number of participants is low, and therefore when evaluating this study, I had to clearly state that there is an unreliable sample size. Another issue that didn't work out was with the AI. Generating a nav-mesh during runtime caused some problems, but having responsive agents seemed problematic with my complex system I had set up, funnelling agents through areas to add realism. This resulted in me creating a more simplistic AI system, which wasn't as complex but kept the agents moving around.

From building this, there are several approaches I would take differently. The first is really about how the village generates, where I would look at using real world data of cities and road layouts and use those to generate a village rather than procedurally generating the whole thing. This worked really well for Marvels Spiderman. (Santiago, 2019) As well as this, I would also look at setting up the quest system as more real time events, rather than a specific thing to do. This would mean I could take the current game state and position and generate a quest / event around what is in the immediate area, not what's in the area at the beginning of the generation cycle. Another thing I would change would be the district structures, and how building districts are decided. The current implementation of a district is determined by each building block, which then limits which buildings are selected to generate in this area. I would change this to be determined more like biomes are determined in games, using Voronoi noise, which could use data points from key areas like the castle, to generate more appropriate buildings in the immediate space surrounding it.

The final thing I want to talk about is the structure that the artefact took. When I started implementing, I tried to have a clean structure, and I had some idea as to how to structure it from the design process. However, a lot of the classes and function rely heavily on each other. Without buildings there would be no roads, without roads no people, without people no quest. Once the project was midway through development, the current structure didn't work well with each other, which ended up with messy code, where little consideration was given to where I was pulling and pushing data to, which caused a lot of issues at the end of the project. Luckily, I was able to do a lot of quick fixed, however if I had worked on creating classes more independently rather than intertwined together, I could have had a lot less hassle.

References

Abeele, V. Vanden *et al.* (2020) ‘Development and validation of the player experience inventory: A scale to measure player experiences at the level of functional and psychosocial consequences’, *International Journal of Human Computer Studies*. Elsevier Ltd, 135(January 2019), p. 102370. doi: 10.1016/j.ijhcs.2019.102370.

Ashmore, C. and Nitsche, M. (2007) *Situated Play, Proceedings of DiGRA 2007 Conference The Quest in a Generated World*.

Doran, J. and Parberry, I. (2011) *A Prototype Quest Generator Based on a Structural Analysis of Quests from Four MMORPGs*.

Emilien, A. *et al.* (2012) ‘Procedural generation of villages on arbitrary terrains’, *Visual Computer*. doi: 10.1007/s00371-012-0699-7.

Grey, J. and Bryson, J. (2011) *Procedural quests: A focus for agent interaction in role-playing-games, AISB 2011: AI and Games*.

Hartsook, K. *et al.* (2011) ‘Toward supporting stories with procedurally generated game worlds’, *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011*. IEEE, pp. 297–304. doi: 10.1109/CIG.2011.6032020.

Jenkins, H. (2004) ‘Game Design as Architecture Narrative’, in *First Person: New Media as Story, Performance, and Game*.

Johnson, D., Gardner, M. J. and Perry, R. (2018) ‘Validation of two game experience scales: The Player Experience of Need Satisfaction (PENS) and Game Experience Questionnaire (GEQ)’, *International Journal of Human Computer Studies*. Elsevier Ltd, 118(February), pp. 38–46. doi: 10.1016/j.ijhcs.2018.05.003.

Nielsen, J. (2012) *Quantitative Studies: How Many Users to Test?, Nielsen Norman Group*. Available at: https://www.nngroup.com/articles/quantitative-studies-how-many-users/.

Santiago, D. (2019) ‘Procedurally Crafting Manhattan for Marvel’s Spider-Man’, in. Available at: https://www.youtube.com/watch?v=4aw9uyj9MAE&t=731s.

Slant (2020) *Slant - Godot Review*. Available at: https://www.slant.co/options/1068/~godot-review (Accessed: 14 April 2020).

Szymanezyk, O. and Cielniak, G. (2010) ‘Group emotion modelling and the use of middleware for virtual crowds in video-games’, *Proceedings of the 3rd International Symposium on AI and Games - A Symposium at the AISB 2010 Convention*, pp. 25–30.

Szymanezyk, O., Dickinson, P. and Duckett, T. (2011) ‘From individual characters to large crowds: Augmenting the believability of open-world games through exploring social emotion in pedestrian groups’, *Proceedings of DiGRA 2011 Conference: Think Design Play*.

Williams, B. and Headleand, C. (2019) ‘Evaluating Models for Virtual Forestry Generation and Tree Placement in Games’. doi: 10.2312/cgvc.20191259.

Williams, B. and Headleand, C. J. (2017) ‘A time-line approach for the generation of simulated settlements’, in *Proceedings - 2017 International Conference on Cyberworlds, CW 2017 - in cooperation with: Eurographics Association International Federation for Information Processing ACM SIGGRAPH*, pp. 134–141. doi: 10.1109/CW.2017.32.

Lague, S. (2019) Coding Adventure: Boids [Video] Avaliable from https://www.youtube.com/watch?v=bqtqltqcQhw [accesed 14th April 2020]