An exploration into the effectiveness of procedurally generated worldbuilding versus traditional handcrafted narratives in games

Dominic Cohen

COH16656067

16656067@lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the

Degree of BSc(Hons) Computer Science

Supervisor: Dr. Olivier Syzmanesyk

April 2022

Acknowledgements

To Jacob, for not only making it possible for me to be here, but for helping me to maintain positivity throughout every one of the numerous catastrophes that have occurred over these three years.

To Mum and Nan, for putting up with the Goblin that has lived solely off of your generosity for the last 6 years.

And finally, to Harry, Jasmine, Haris, Ethan and Isabelle – for fostering a character that otherwise would not have been able to make it through first year, let alone all three. You all deserve the very best the world has to offer and more.

Abstract

Games often use procedural generation to build digital environments and distribute handmade items, enemies and decorations within them - but rarely do they use it to produce the context for such placements. Most developers handcraft this part themselves, giving the generation system rules of what to place where, rather than letting it produce the content it places on its own. This study aims to compare these two techniques, examining player responses during gameplay. Players played through a simple dungeon crawler multiple times, in which they encountered various worldbuilding elements placed in the level which they can interact with. Half of the playthroughs contained a handcrafted series of events written by the developer, and the other half utilised a replacement grammar system to generate a narrative for the player to discover piece by piece. After experiencing each of these, the player filled out a form detailing their experience of the game, through which we can determine if such systems make a difference and if so, whether or not they improve the experience they are a part of.

Table of Contents

Introduction

Background

In games, procedural generation is a tool often used in the games industry to create digital worlds, generate and place unique items, and distribute resources and obstacles throughout the game for the player to discover - as an alternative to the time-consuming development of such artefacts on their own.

When creating more qualitative elements, such as item descriptions, developers often handcraft these elements themselves, placing them in the level after generation. The advantage to this is having absolute creative control on the background of the game - however meaningful worldbuilding is a time-consuming endeavour, one that grows exponentially alongside the size of the level.

If there was a way to procedurally generate narrative events within a game, and then distribute items and obstacles across the level that conveys that narrative effectively, then the developer only needs to specify what *can* happen rather than what *will*. Whilst this does inhibit the ability to tell specific stories, it does allow for unique events that “employ apophenia, the human tendency to perceive patterns, in their interpretations” (Grinblat and Bucklew, 2017, 2)

Qualitative Procedural Generation (QPG) is still being explored. Games like *Caves of Qud* and *Dwarf Fortress* have each explored methods of generating history and culture within their games, but both of these methods were developed as tertiary systems within an already developed product, limiting their ability to have a marked effect on the player. These sorts of systems should lend “depth and, crucially, meaning to all of the forms of procedural generation a player encounters in a playthrough” (Johnson M., 2016, 5)

This project aims to explore the effectiveness of these systems further by putting it into the foreground for the player to interact with. In addition, rather than placing historical events into the world at complete random, the project aims to distribute events in a way that makes sense within the context of the level itself - Whereby the beginning of the narrative is found near the beginning of the level, the end near the end etc. This is because good level design is driven by mechanics, “whose primary function is to leverage your mechanics to create a great experience” (GDC, 2018b) - and in order to incentivise engagement with this system, giving the player a piece of the puzzle early in the game serves as both a taster of the narrative and a tutorial of the game mechanics.

Naturally, once a system like this has been created within a game, it would need to be tested against a more traditional worldbuilding method. Using the Game Experience Questionnaire (GEQ), we can have players run through the game multiple times, with half the playthroughs using procedurally generated narratives, and half using traditional handcrafted worldbuilding. This way, we can compare the flow and immersion of the two techniques directly, as well as analysing the players’ ability to tell the difference between each method of narrative generation - deducing the effectiveness of the new system in the process.

Aims and Objectives

The project aims to develop a procedural narrative generation system that enriches the existing level design of a game through novel worldbuilding. In order to achieve this aim, the project must complete the following objectives:

- Produce a set of 20 generic entities and events that can be strung together and fed into a replacement grammar system.

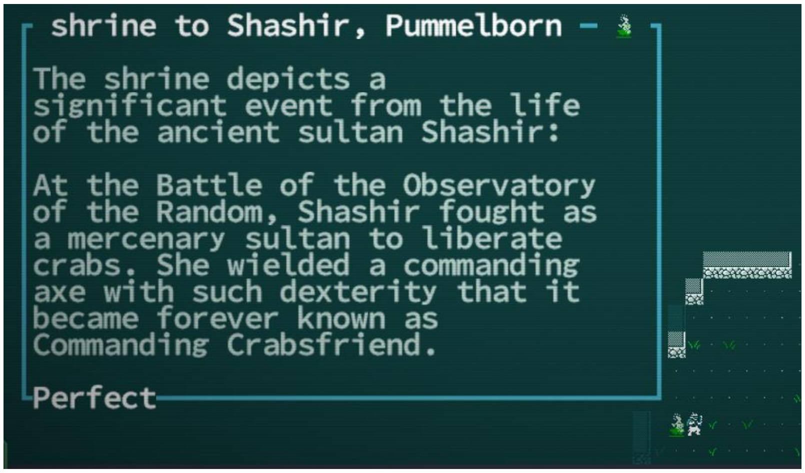
- Develop a simple, playable, 5-10 minute dungeon crawler to house the project.
- Devise a means of interacting with and displaying 8 generated text snippets within the created dungeon crawler.
- Construct a series of 3 narratives complete with a blueprint for their distribution within the game world.
- Measure player responses to 3 instances of each generation technique using the Game Experience Questionnaire.
- Collect the data from player responses and identify any major patterns that form between the handcrafted and procedurally generated levels.

Many instances of these objectives, such as the 20 entities for the replacement grammar system or the number of handcrafted levels were based purely on what seemed achievable at the time of writing – in many cases these numerical elements were modified as the project progressed.

Literature Review

Background

There are many ways to go about procedurally generating content for use in games or other media. Procedural content generation is essentially random generation (or as close as we can get with a computer) overlaid with a series of rules to curate the results. When designing these rules, you are aiming to create a “possibility space (all the kinds of artifacts [the generator] can generate) where most of the artifacts have the good properties, and few (or none) of the artifacts have the bad properties” (Compton, K., 2016). For example, a possible generative method for this project was a constraint solver, where you generate every possible variant of content, testing it against a series of tailored constraints until a desired outcome is reached. This is a viable method, but “too many choices will create a number of possible artifacts to search that is greater than the number of atoms in the universe", which isn't ideal.



shrine to Shashir, Pummelborn

The shrine depicts a significant event from the life of the ancient sultan Shashir:

At the Battle of the Observatory of the Random, Shashir fought as a mercenary sultan to liberate crabs. She wielded a commanding axe with such dexterity that it became forever known as Commanding Crabsfriend.

Perfect

Figure 1: *An example of a shrine depicting a historical event in Caves of Qud.*

A system such as this relies on a decent replacement grammar system to generate text snippets from the given data - snippets that make sense within the context of both the content of the game and the rules of the English language. This makes it "easy to encode knowledge about a particular artifact and its structure and its options all in one piece of data" (Compton K., 2016). Tracery.io, a free generative text tool, is used by the people at Freehold Games (*Caves of Qud*) as well as a variety of existing projects both in academic and developer communities. Its "scalable complexity allows for a low barrier of entry and a simple JSON file format [that] supports a culture of sharing and remixing" (Compton K., 2015, 1). This means that it will both be easy to embed into the game engine and allow for easy modifications should we wish to alter the games themes or narrative scope.

The creators of the game *Caves of Qud* wrote a paper on the devising of a system to generate the biographies of historical 'Sultans' through using a replacement grammar system, "whose rules map sultan properties to text fragments for a variety of narrative circumstances" (Grinblat and Bucklew, 2017, 4). It then modifies the Sultans properties based on the randomly determined results of that event, before moving on to the next event - provided that the Sultan is still alive.

The text snippets from these events - such as the one in figure 1 above - are then scattered across the game world, which the player can unearth through interacting with the environment. This system could easily be replicated on a smaller scale, mapping actions spanning a few hours/days rather than an individual's lifetime.

There are many ways to analyse and measure the game experience. Of those that are academically backed, two that stand out are the Player Experience of Need Satisfaction (PENS) and the Game Experience Questionnaire (GEQ). “The PENS questionnaire assesses player experience in the dimensions: Competence, Autonomy, Relatedness, Intuitive Controls and Presence/Immersion” (Johnson D., 2014, 2). Some of these dimensions are useful in comparing the effectiveness of our two systems, but there are ultimately a lot of dimensions of the questionnaire that present no real use to this project.

On the other hand, the GEQ is a much more concise self-report measure of the gameplay experience. Participants indicate their agreement on a number of questions using a 5 point Likert scale, the answers to which measure the player's sense of Competence, Immersion, Flow, Tension, Challenge, Negative affect and Positive affect. In addition, there is a shorter ‘In-game’ questionnaire, which “has an identical component structure” and is used for “assessing game experience at multiple intervals during a game session” (IJsselsteijn, W.A., 2013, 3). This makes it suitable for this project since we can compare the feel of multiple short playthroughs played in quick succession, rather than an extended test at the end of 6+ playthroughs.

Methodology

Project Management

This project is focused primarily on the design and development of a procedurally generated narrative, and the evaluation of that system within a game. However, in order to carry out this evaluation, a game must be created to house the procedurally generated content, preferably in such a way that the game can be played with or without the content allowing for players to compare and gauge whether or the game is improved by its inclusion.

Because of this, the project requires careful management, since many elements cannot be started without the completion of others. However, elements such as the game that house it need to be managed in such a way that they are only developed to the point of being functional – it is important that they are not overdeveloped at the expense of other tasks. A lean approach would cater to this rather well, reducing wastefulness and streamlining the projects progress as much as possible.

As for the actual day-to-day handling of the project, I intended to implement a simple waterfall model, outlining the requirements before developing the game, implementing the two narrative systems and finally testing and evaluating the merits of each. However, as mentioned previously the potential for the game aspect of the project to become an endless cycle of iterative development means that a bottleneck is far too likely, meaning that a waterfall model wouldn’t be very useful in determining how far along the project actually is.

Breaking the project down into smaller steps and managing those individually would better suit the project, especially if it allowed for the prioritisation of certain tasks over others.

Creating a KANBAN board would allow for tasks to be added and ordered within a todo list, letting us visualise the project overall as tasks are moved across the board overtime. It could also allow for the use of a ‘back-burner’ section, allowing wasteful tasks to be pushed to the back to make way for more important elements, preventing the bottlenecks which would otherwise be likely in other methodologies.

Software Development

There were two main elements of software development for this project: the system for worldbuilding generation itself, and the simple game that houses it. Naturally our focus is on the system itself, however in order to effectively test and compare such a system there needs to be a game to grant it context.

The pieces of software being developed are different enough to warrant different methodologies for each – the PCG system is using an unfamiliar tool created by a third party, whereas the game is using a well-established game engine to create something that fits a specific set of requirements.

As established, the game needed to be completed in as short a timeframe as possible – it needed to meet the bare minimum requirements for the implementation and testing of the PCG content, nothing more. As such, sticking to the KANBAN approach employed in the project overall was a suitable approach – it meant that since each required element of the game was visualised on the board, going off-task would be discouraged – any development on non-critical features of the game would visibly have no effect on the progress of the project.

Whilst the PCG system originally started out being developed in the waterfall method, it quickly proved to be an unreliable method for creating a system of such complexity. The generative texts were designed first on whiteboards, then put into the inbuilt editor on Tracery’s website before finally being converted over into Unity, where they could be tested on actual in-game items. This quickly proved ineffective, as the syntax between the websites JavaScript and the C#

library in Unity were different enough that small changes to the script required a great deal of editing before testing could occur – an issue that quickly got out of hand as the history generation systems got more complex.

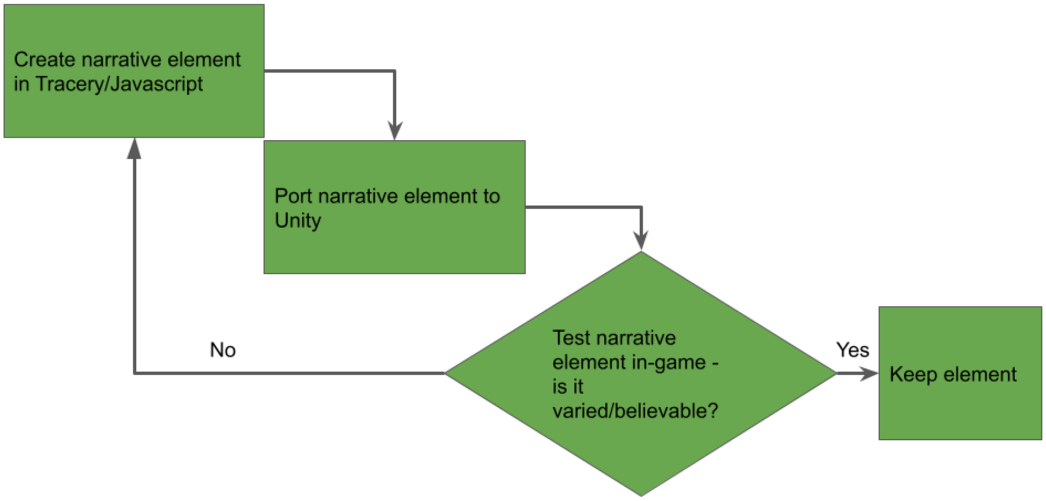
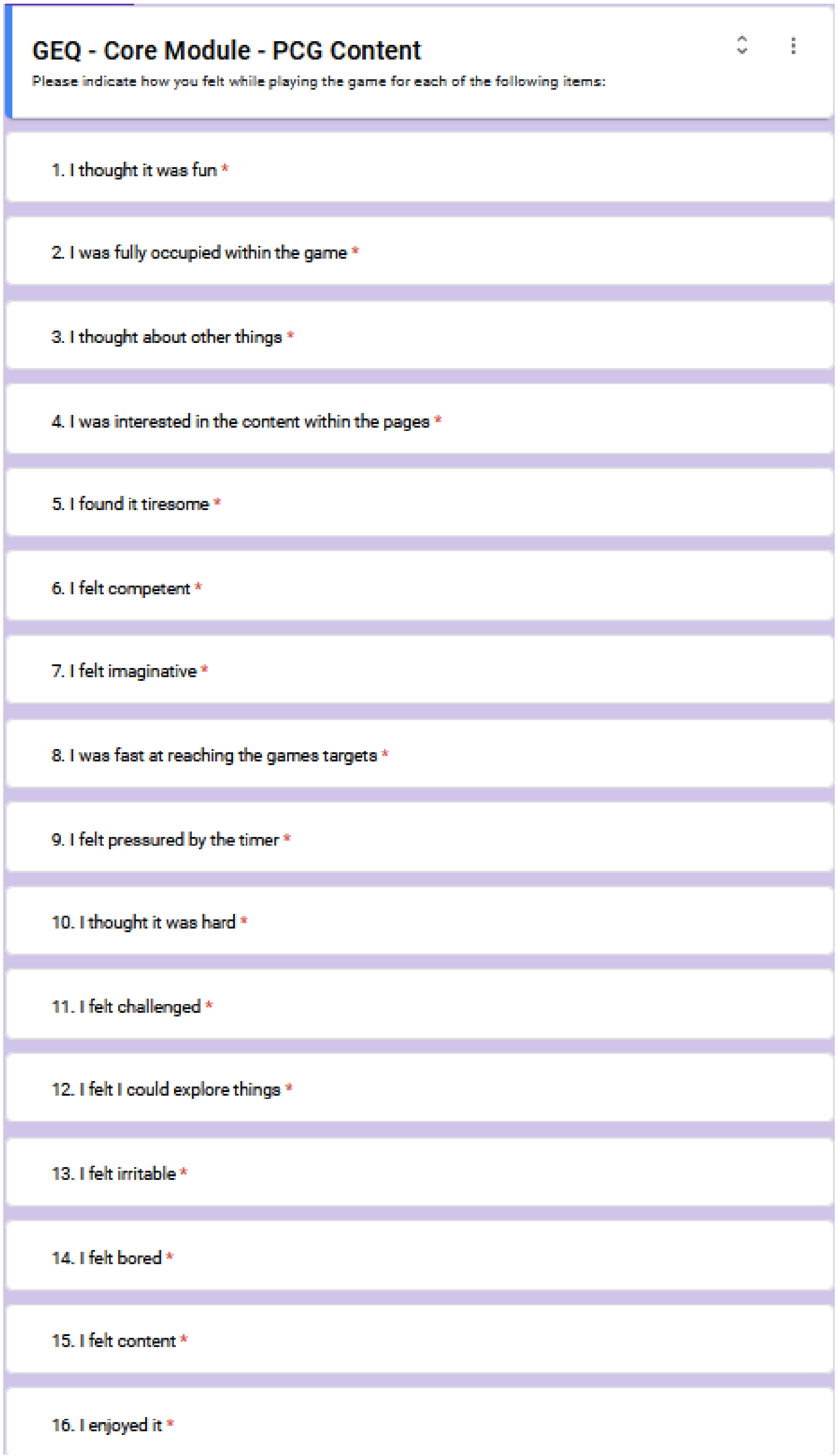

Create narrative element in Tracery/Javascript
Port narrative element to Unity
Test narrative element in-game - is it varied/believable?
No
Yes
Keep element

Figure 2: *The original waterfall methodology for the QPG system.*

To solve this, an agile methodology was adopted – the JavaScript element of designing the system was cut entirely, in place of translating planned worldbuilding elements directly into Unity script. This allowed for a more iterative approach, where every addition/change to the system could be reviewed quickly and efficiently and any defects or design changes identified. Removing Javascript development also meant that the time to complete each iteration was much shorter, letting there be more iterations in a smaller timeframe and allowing the system and the game to be developed alongside each other within the KANBAN board.

Research Methods

Our primary method of research for this project will be the acquisition of data from players using a modified version of the Game Experience Questionnaire (IJsselsteijn, W.A., 2013), a questionnaire designed to take qualitative data about the players opinion of the game and turn it into quantitative data which can be more easily analysed and compared with other datasets. Other questionnaires were considered early on in the project, but ultimately the GEQ was settled on due to its more concise format, making it easier to have players complete multiple times without too much frustration. The GEQ contains multiple ‘modules’ which each serve a particular purpose depending on the type of data you wish to obtain. The core module is the standard variant, and condenses 32 questions into 7 components for analysis. However, there is also an ‘In-game’ version of this module, which is designed to be completed during gameplay and as such is stripped down to only 14 questions whilst still resulting in the same seven components. As the player is going to be playing the same game over and over, it makes sense to use this more concise version, having the player complete the form after each individual playthrough such that a version of their results exists for both handwritten and procedurally generated narratives. In addition, a few modifications can be made to cater the results more heavily towards the background and worldbuilding of the game, since certain questions such as “I had to put a lot of effort into it” will likely be the same regardless of which version of the worldbuilding is in play.



GEQ - Core Module - PCG Content

Please indicate how you felt while playing the game for each of the following items:

1. I thought it was fun *

2. I was fully occupied within the game *

3. I thought about other things *

4. I was interested in the content within the pages *

5. I found it tiresome *

6. I felt competent *

7. I felt imaginative *

8. I was fast at reaching the games targets *

9. I felt pressured by the timer *

10. I thought it was hard *

11. I felt challenged *

12. I felt I could explore things *

13. I felt irritable *

14. I felt bored *

15. I felt content *

16. I enjoyed it *

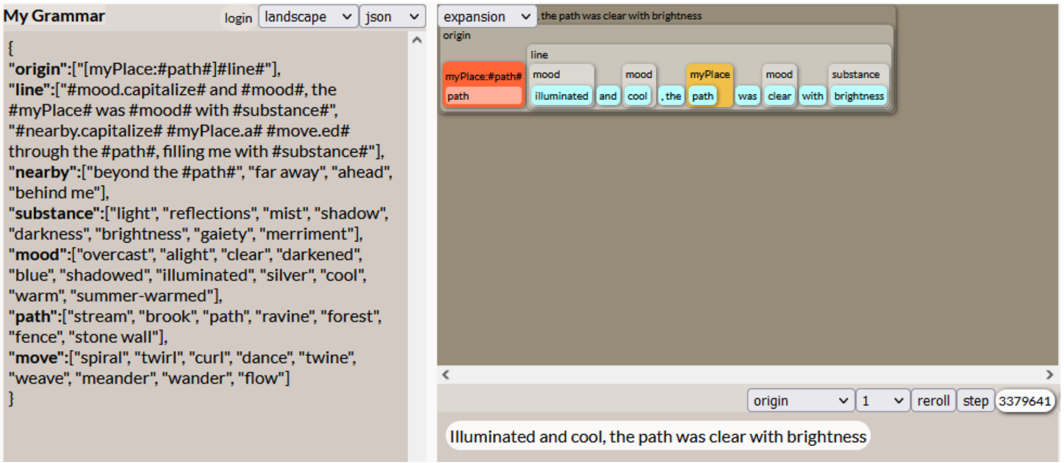
Figure 3: *The modified version of the GEQs in-game module.*

In order to make the questionnaire as concise as possible, a number of questions were removed. Typically, these were questions which had an element of overlap with other questions in the module. Questions such as "I found it frustrating" were deemed acceptable remove as the feelings associated with frustration could easily be assigned to existing questions such as 10, 11 and 13. Overall, the number of questions in the questionnaire was cut by about 60%, leaving a little room for some additions specific to this project – such as questions 4 and 9.

Finally, a third section was added asking the player whether or not they believed the content experienced in their playthrough was procedurally generated or handwritten. This wouldn't necessarily show the impact of PCG content, but it might be interesting to compare the results of players who believe their content is PCG to those who believe it is handcrafted, as well as whether not players were typically correct in their assumptions.

Toolsets and Machine Environments

When it comes to creating a simple game within which a procedurally generated history system can be effectively and easily integrated and analysed, there are a multitude options. The most popular choices are naturally going to be Unity and Unreal, not just for their ease of use but also for the plethora of online content (assets, tutorials, entire libraries) that has been created over time to allow developers to get prototypes off of the ground fast. Alternatives such as building the game from scratch were never seriously considered, as the amount of work to get basic functionality off of the ground would outweigh the usefulness of a more customised engine experience. In the end, Unity was chosen for its simplicity – Unreal is geared more towards 3D titles, and Unity has a vast asset library of free content which can be used to further speed up the prototyping process. One major example of this speeding up production is a module dedicated to Tracery, integrating it into the engine saving us valuable time.



```
{
"origin":["[myPlace:#path#]#line#"],
"line":["#mood.capitalize# and #mood#, the
#myPlace# was #mood# with #substance#",
"#nearby.capitalize# #myPlace.a# #move.ed#
through the #path#, filling me with #substance#"],
"nearby":["beyond the #path#", "far away", "ahead",
"behind me"],
"substance":["light", "reflections", "mist", "shadow",
"darkness", "brightness", "gaiety", "merriment"],
"mood":["overcast", "alight", "clear", "darkened",
"blue", "shadowed", "illuminated", "silver", "cool",
"warm", "summer-warmed"],
"path":["stream", "brook", "path", "ravine", "forest",
"fence", "stone wall"],
"move":["spiral", "twirl", "curl", "dance", "twine",
"weave", "meander", "wander", "flow"]
}
```

```
Illuminated and cool, the path was clear with brightness
```

Figure 4: *Tracery's online editor, useful for prototyping new narrative events.*

As mentioned in the introduction, initially the primary element of the project – the procedural generation of historical events – was handled using Tracery, a JavaScript library created by Dr Kate Compton that uses grammars to create interesting and varied texts. It's a library that I have experience in before, as well as a library that pairs well with game engines for creating unique and interesting content. Time was spent looking at other options such as GPT-2, an AI that autocompletes text when given prompts. However, GPT-2 is known for quickly becoming nonsensical when writing entirely original content – "although its grammar and spelling are generally correct, it tends to stray off topic, and the text it produces lacks overall coherence" (Vincent, 2019). This ultimately means that a system like this isn't fit for purpose – the text that players' encounter in the game needs to relay a specific set of information through the historical content, as well as making sense – straying off topic would pull the player out of the experience. In addition, the use of an advanced AI trained on over 1 billion datasets is a tad overkill for a project of this scope – not to mention the time that would need to be spent integrating it into Unity and feeding it the necessary data to generate text of the correct tone and style.

There are some requirements of the procedurally generated history system that cannot be achieved with Tracery alone – with each event generated, elements such as the name and age of the subject must be saved and reused across each event, giving the impression of continuity. Tracery has methods to save data for use within its system, however there are no methods of preserving that data between generations. The solution to this is to use Unity scripts to generate texts piece by piece and storing the important elements as class properties, allowing for them to be initialised and updated in-between each historical event. Each instance of said class would then represent a particular figure and their ‘story’, the pieces of which can be jumbled and distributed throughout a level in the game.

When it came to gathering data on the players experience of the game, two methods were explored – Microsoft forms and Google forms. Both allow for the creation and distribution of questionnaires with neither standing out in particular, with the primary differences being in how the results are displayed. Microsoft forms allows for the direct and easy export of gathered data to Microsoft Excel, a useful feature which could prove useful especially for larger datasets. However, Google Forms automatically collates results into a variety of visual aids making analysis and evaluation of the data simpler without having to configure our own methods for viewing and comparing the results.

Design, Development and Evaluation

Software Development

Requirements of the Game

It has already been stated that the game developed for use in this project must be simple: it needs to have a way to display the generated historical content, and said content must be spread across multiple areas such that the player discovers them bit by bit – allowing them to form a narrative themselves through exploration. However, beyond this, the ‘game’ element can take any number of forms.

During the design phase, several ideas were presented and eventually narrowed down. These included a traditional dungeon crawler, a text-based exploration game, a card game etc. There was even the idea for a simple linear display of the PCG worldbuilding elements where the player could just click through them one by one, however this was quickly put aside as it would not provide any insight into the effectiveness of the system within games – only an insight into the quality of the system itself. Ideas like the dungeon crawler held promise – the use of a map of rooms for a player to walk around in would allow for an even distribution of worldbuilding elements, much like how games like Skyrim (Bethesda, 2011) distribute books and other reading materials across the game world. A map similar to Enter the Gungeon (Devolver, 2016) could be simple to implement, though creating a PCG landscape to traverse would be a complex task that would have no effect on the project objectives. The game needs to have an element to drive the player – there needs to be cause to explore and collect the texts outside of being told to play the game for research purposes – but these motivations must be simple so as to not alienate the player. Finally, the game must be able to be completed start to finish in a relatively short space of time, as the aim of the project is to have the game played multiple times over to compare differences between different worldbuilding methods.

Overall, the requirements of the game can be summarised as follows:

- Mechanically simplistic.
- A level design that is simple to prototype, test and iterate upon during development of the PCG system.
- Allow for the PCG system to be swapped out for traditional handwritten text.

- Drive the player to interact and engage with the narrative content presented.

Designing the Game

Once the ideas were narrowed down, proper design could begin. The game is going to be from a top down perspective – This serves multiple purposes, but chief among them is asset creation. Creating 3D assets alongside a 3D environment is not only more time consuming, it also adds another dimension to procedural content generation in both a literal and figurative sense. Populating a 3-dimensional level with items would require more complex code to handle placing those items, and in addition creating a character controller in a 3D space is often more difficult to get right than a 2D one. This is especially true if you consider that with a top down perspective, I won't even need to consider jumping or gravity within the game.

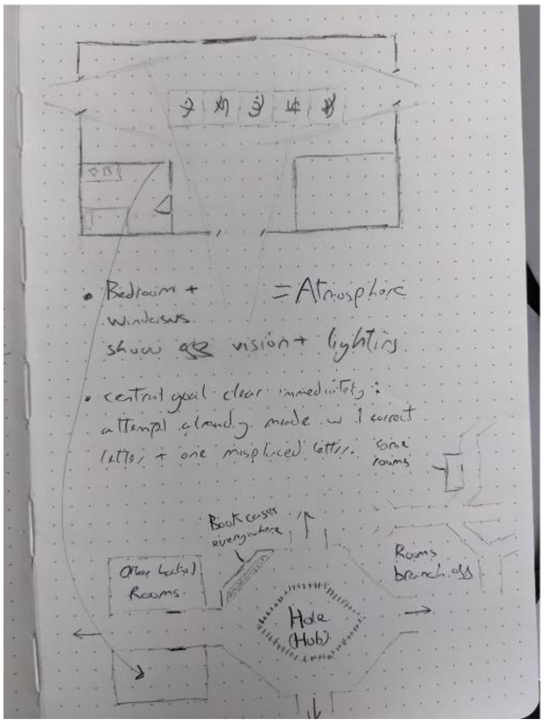


Figure 5: Initial sketches of how to give the player multiple options for exploration

Driving the player is going to prove difficult – naturally players like to explore when given options (needs citation), so creating a level with multiple pathways out of the gate would passively incentivise exploration. However, we want to keep the experience short, and so having a central area with multiple branches is only a good idea if the branches themselves are kept short. An initial sketch was made early on of a possible level for the game – this level had the central location mentioned, with 4 corridors heading to the North, South, East and West. Surrounding this central area was 4 'wings' each of which could have a distinct feel whilst being accessible via the two adjacent corridors. This early prototype was promising, but as time went on it became clear that 4 wings was too many – better to craft 2 well-designed areas than 4 samey ones.

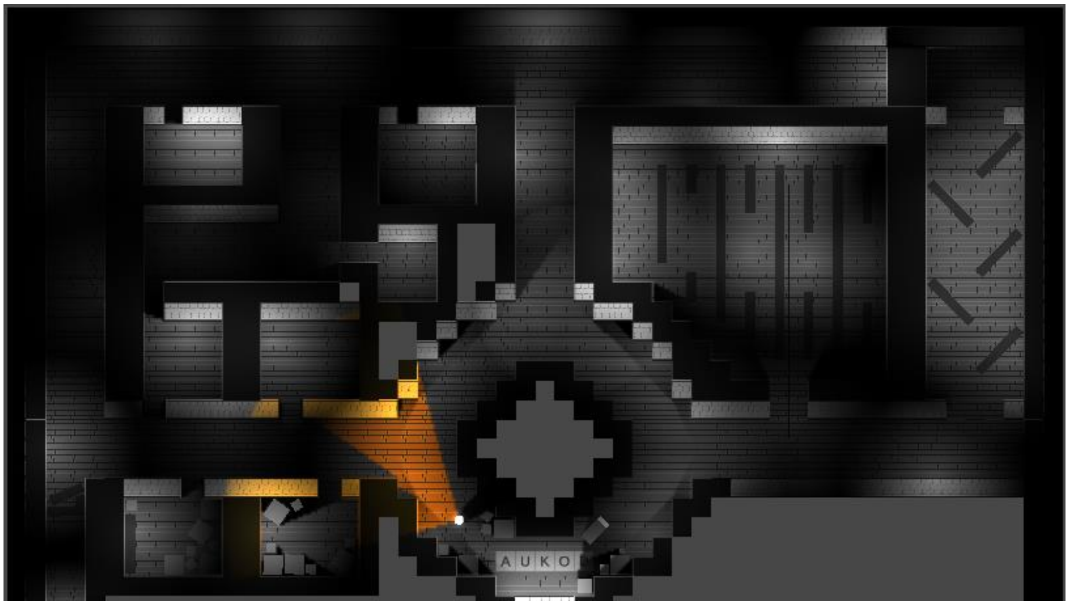


Figure 6: *A birds-eye view of the map once completed.*

Ultimately, this decision was fruitful – with 3 corridors leading away from the centre, the player has not lost that initial choice when exploring, and the amount of work required has been halved. A final design decision that contributes to driving the player was to have the game take place over a number of in-game 'days'. This is made clear at the start with the player being informed of the game starting at 'Day 1' coupled with a timer in the top right counting down the amount of time left in the day. The lights implemented later on were then used to reinforce this day/night system, hopefully driving the player to complete the game in as little time as possible.

***Figure 7**: Hotline Miami (Devolver, 2012) breaks up its narrative with fast-paced violence.*

One requirement of the game is the 'game' element – analysing procedurally generated content on its own can be done without a game at all, but since the goal of this project is to look at its effectiveness and use within games, there needs to be an actual element of play involved. As a top down game based in an interior area, a common method of conflict in games is *literal* conflict. Games like Hotline Miami (Devolver, 2012) and Darkwood (Acid Wizard, 2014) have simple combat mechanics that involve swinging/firing various weapons at enemies, most often of which lunge at the player in melee. Whilst this could allow for an element of variety between the exploration and puzzle-solving element of the game, it would ultimately distract from the goal of the project – bringing procedurally generated history to the foreground. As such, something else is needed.

One minor mechanic developed during the prototype stage was a light switch. Whilst relatively simplistic and if anything irrelevant to the game as a whole, it eventually sparked an idea – if time was to be spent on the lighting of the level, why not make said lighting a part of the mechanics? A script was devised that over time switched off the lights in the level, section by section – this was paired with code in the character controller that depleted a 'panic' meter (this was later renamed to sanity, as it makes more sense to 'lose' sanity over time) from 100 to 0 the longer the player spent in darkness, replenishing that meter should they make it back into the light before it hit 0.

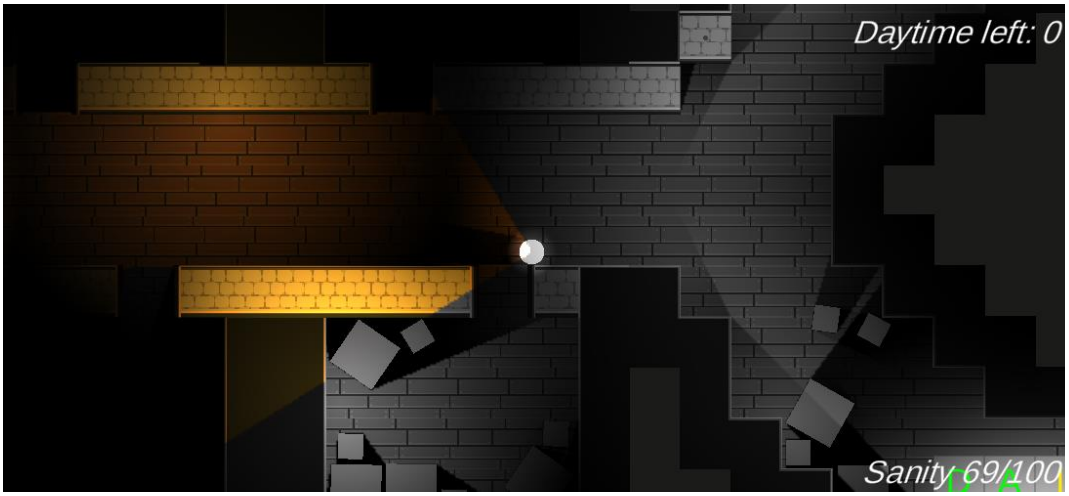


Figure 8*: An example of the panic meter depleting after time in darkness – the player has just exited the room in the top-left corner.*

This proved an effective primary mechanic – thanks to the combination of the lights and the timer, the player now has a limited window to head out into the level to find pieces of information before rushing back to the centre as it grows dark. The centre of the level becomes a safe haven, populated with lights that don't turn off at the stroke of night.

In some of the initial playthroughs of the game, it was identified that whilst the day/night system does add the needed danger element that drives the player, it does not add much to the moment-to-moment gameplay – the player is still limited to just moving the player with the WASD keys and interacting using E. Two minor additions were made to combat this, each of which complemented each other to hopefully bring some depth to an otherwise bare level containing only walls and floors. A vision system, which obscures objects not in the direct 'vision cone' of the player, whilst also employing a shader that leaves the cone as the only source of colour in the scene, focusing the players attention – and the ability to push/pull physical objects such as furniture.

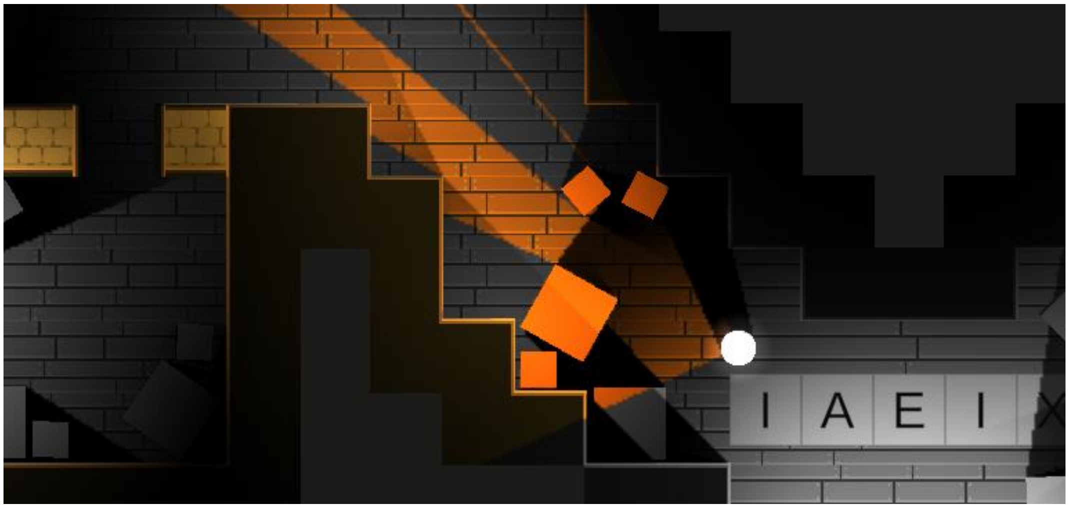


Figure 9: *An example of obstacles obscuring the players view.*

These pieces of furniture would also obscure the vision cone, meaning the player may be forced to push objects out of the way to see if there is an entity or item behind it. For example, should a crate be blocking the vision cone of the player, a page of information placed behind it would be obscured, causing it to be invisible to the player. This lends an element of claustrophobia to the game *despite* being a top down experience, where it is traditionally hard to obscure things. This is a mechanic largely borrowed from *Darkwood*, which uses it primarily to enhance the horror element of its design, whereas we are using it to further add to the tension – the player is rushed for time, so any obstacle is only going to increase that pressure.

Requirements of the PCG

The system for procedurally generating a narrative is going to mimic that of the system used in Caves of Qud: using a generative grammar system to create an individual then applying that individual to a series of scenarios, modifying the individual as needed (Grinblat and Bucklew, 2017). When an individual is created in this system, they begin with a set of properties: their name, age, occupation etc – these properties will need to be stored in memory somewhere outside of the system itself, so that Unity can access them. This is so that we can manipulate the game world in accordance with these events.

A series of events must be selected from a pool of pre-written event 'templates' these templates will contain blank elements which can be filled from a pool of pre-written details, hopefully such that in the event that an individual were to have two events that originated from the same template, the player would not notice. Increasing the number of event templates can mitigate this.

The number of events generated can be arbitrary for the purposes of this project, though it would be sensible to keep them to a number reflective of the size of the map being used for the game. If the game uses a PCG map that alters in size with each playthrough, then the generative system used will need to work in accordance to generate an appropriate number of events before they are distributed. If the game uses a hand-built map, then the number of events generated can similarly be handpicked to whatever feels appropriate – though ensuring that both systems ultimately have the same number of events placed in the level is important in order to keep the player in the dark as to which is which, as well as ensuring that difficulty is not affected by which narrative system is currently in use.

Finally, the number of possible events that can occur within the narrative needs to be enough such that there is little in the way of repetition – if the player is to replay the game, then the experience needs to be different to keep things interesting. However, since each of these events will warrant additional assets to reflect the effect of that event on the game world, too many events will become an overly time-consuming task that will negatively affect the quality of each event. Thus, it is important to find a sweet spot between these two extremes. If additional assets cannot be produced to mimic the effects of each possible event, then the overall narrative of the game will need to shift such that such an omission goes unnoticed – if paced quickly then the player would spend too much of their time acquiring the events to take in their surroundings.

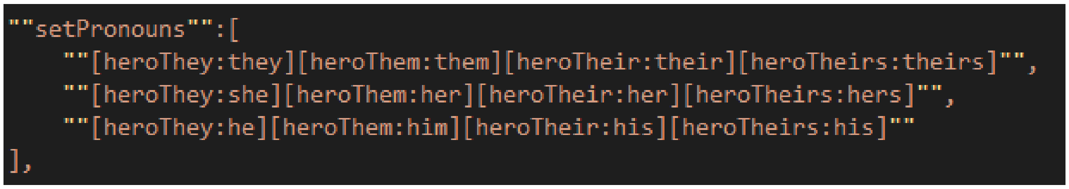
Overall, the requirements of the PCG can be summarised as follows:

- Generate events from a pool of templates, filling those templates with a variety of details to ensure even repeated templates appear distinct.
- Tie a series of events to a specific individual – keep track of that individuals properties
- Generate multiple individuals each with their own set of events.
- Generate a number of events that make sense for the size of the map they're being placed into.
- Little to no repetition in event structure or details.

Designing the PCG

Players are going to be replaying this instance multiple times in order to show the variety of results the system can produce. As such, it is important that the instance of gameplay overall is rather short. This will also mean that the player will be able to recall the game in its entirety during subsequent playthroughs – preventing them from forgetting previous instances of procedural generation.

In initial development, the first port of call was to create the figure by which events will occur – Since I decided early on to set this game in a library, I decided to name these figures 'archivists'. At creation, each archivist would have generated for them a set of properties – their name, age, particular field and specialty, as well as their pronouns – this was a particularly tricky process since each of these properties was then saved to a class instance, and some compound properties such as the pronouns were rather difficult to separate from the generator using the Tracery library. Once this was done however, I had a system whereby I could create entire sets of properties related to one another without the random element of the system jumbling it.



```
""setPronouns"":[
    ""[heroThey:they][heroThem:them][heroTheir:their][heroTheirs:theirs]"",
    ""[heroThey:she][heroThem:her][heroTheir:her][heroTheirs:hers]"",
    ""[heroThey:he][heroThem:him][heroTheir:his][heroTheirs:his]""
],
```

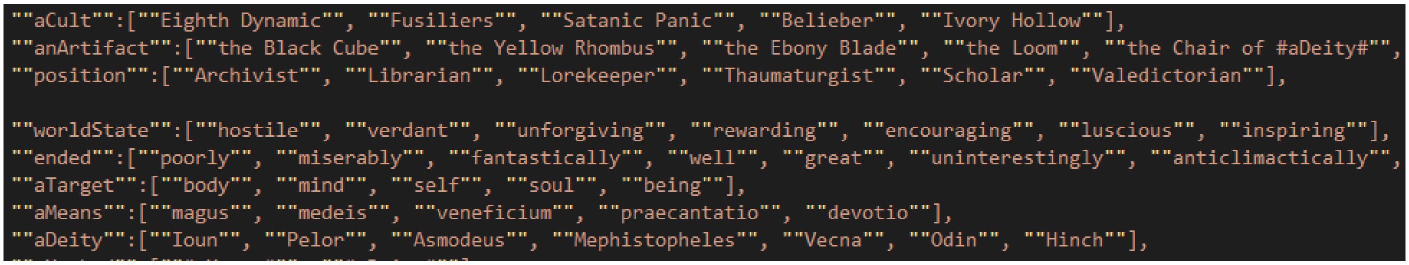
Figure 10: *The compound property system. Using Tracery code to assign multiple properties at once, a random option of the three is chosen, and all four tenses are assigned as properties in tandem.*

To use the pronouns as an example, generating a random simple past tense pronoun could produce ‘her’ – the system could not then use ‘he’ or ‘they’ as a present tense pronoun. Once one is chosen, it must be grouped with the correct counterparts.

The events themselves were rather simple to generate once the initial groundwork was laid out. I hand built a series of template event structures through which the archivist properties could be inserted to create a random historical event.

“Archivist #archivist# gained the title of #title# after aiding #name# and #name# in the banishing of #aDeity# from the mortal plane.”

Above is an example of one of the event templates created early on in development. #archivist# and #name# both derive from the same set of possible names to generate, however #archivist# is referring to the property of the individual associated with the event, whereas #name# will simply pick a new name from the list each time it is invoked. This allows the name used in #archivist# to persist across multiple events, as will the result of #title#. In some cases, an archivist may not have a property such as #title#, meaning that a new property will need to be created should an event like this be chosen from the pool. If the property does already exist, then it can simply be overwritten. This process will also allow us to perform simple mathematics on numeric properties such as the archivists' age, increasing it by a random amount with each event generated to give the impression of time passing. In addition to this, a birth year could be determined for each archivist, increasing along with the age such that each event found in the game could have a timestamp next to it – this would bind all events across all archivists to the same rough timeframe, letting the player subconsciously employ apophenia to form their own connections depending on the order of which the events are found.



```
""aCult"":[""Eighth Dynamic"", ""Fusiliers"", ""Satanic Panic"", ""Belieber"", ""Ivory Hollow""],
""anArtifact"":[""the Black Cube"", ""the Yellow Rhombus"", ""the Ebony Blade"", ""the Loom"", ""the Chair of #aDeity#"",
""position"":[""Archivist"", ""Librarian"", ""Lorekeeper"", ""Thaumaturgist"", ""Scholar"", ""Valedictorian""],

""worldState"":[""hostile"", ""verdant"", ""unforgiving"", ""rewarding"", ""encouraging"", ""luscious"", ""inspiring""],
""ended"":[""poorly"", ""miserably"", ""fantastically"", ""well"", ""great"", ""uninterestingly"", ""anticlimactically"",
""aTarget"":[""body"", ""mind"", ""self"", ""soul"", ""being""],
""aMeans"":[""magus"", ""medeis"", ""veneficium"", ""praecantatio"", ""devotio""],
""aDeity"":[""Ioun"", ""Pelor"", ""Asmodeus"", ""Mephistopheles"", ""Vecna"", ""Odin"", ""Hinch""],
```

Figure 11: *A section of the 'Details' that are incorporated into the event templates.*

It is important that enough of these events are created to prevent repetition in the overall set, as well as enough entries in each of the subsections created, such as #aDeity#. In order to save on time, care should be taken to ensure that a subsection is only created if it is deemed useable in a number of different scenarios, so that the time spent building said subsection is not wasted enhancing the detail of a single event. In addition, some events are compounded with the inclusion of *other* events – such as in figure 10 where "The Chair of #aDeity#" is an option in the #anArtefact# detail. This further abstracts the details themselves whilst also aiding in ensuring the lack of repetition, and adding to the players sense of apophenia by further fostering connections between events.

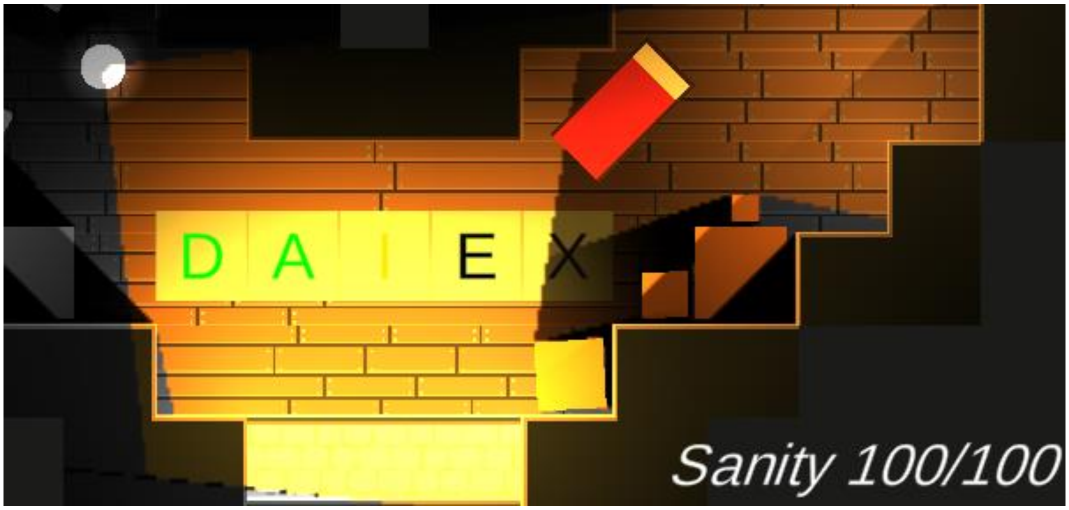


Figure 12: *The 'wordle' puzzle by which the game is centred.*

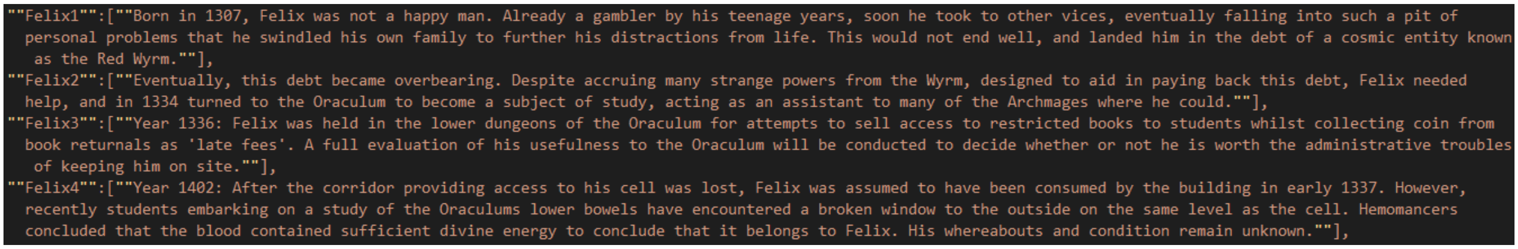
To tie the generated content to the gameplay, the names of each of the generated archivists must be saved once created, with one being chosen at random to be the answer to a basic puzzle in the centre of the level. Rather than spending valuable time devising a clever puzzle with which to connect the name to the game, this puzzle shall be laid out similar to a wordle – a simplistic word game that gained immense popularity at the time of the design phase of this project. The 'wordle' in question will be 5 letters, and centred on the level – the player can explore the level finding pages containing various historical events, many of which contain figures that *coincidentally* all have 5 letter names.

One issue that came about was the distribution of the generated events – the list of events are generated in order, with the first 4 corresponding to the first archivist, the second 4 to the second and so on. This created a problem as it meant that often events associated with a particular archivist would be found in close proximity to one another, instead of spread about for the player to find and piece together over time. To solve this, the list of events would need to be shuffled before put into distribution, so that they are more or less random in their placement on the map. To handle this, a simple version of the Fisher-Yates algorithm was implemented, which shuffled the data sufficiently.

Designing the Handcrafted elements

Naturally, since this project involves comparing procedurally generated content against more traditional methods of worldbuilding, the game will need some traditional worldbuilding to put in place of the PCG. This way, players can experience both methods, and after completing a GEW on each the results can be analysed to see the difference. The handcrafted pieces of worldbuilding will follow the same requirements as that of the PCG – allowing them to be plugged into the game easily, and hopefully in such a way that the player is not aware which version they are playing.

In the game, 5 different ‘Archivists’ are created, each with 4 significant life events. These events are generated in order, with the game assigning a birth year and keeping track of the Archivists age throughout, such that when distributed in the world each entry can be correctly dated despite being randomly distributed throughout the level. So, the task is to create 5 different Archivists, each with a distinct personality and style of events to try and differentiate each of them from one another.



```
""Felix1"":[""Born in 1307, Felix was not a happy man. Already a gambler by his teenage years, soon he took to other vices, eventually falling into such a pit of
  personal problems that he swindled his own family to further his distractions from life. This would not end well, and landed him in the debt of a cosmic entity known
   as the Red Wyrm.""],
""Felix2"":[""Eventually, this debt became overbearing. Despite accruing many strange powers from the Wyrm, designed to aid in paying back this debt, Felix needed
  help, and in 1334 turned to the Oraculum to become a subject of study, acting as an assistant to many of the Archmages where he could.""],
""Felix3"":[""Year 1336: Felix was held in the lower dungeons of the Oraculum for attempts to sell access to restricted books to students whilst collecting coin from
  book returnals as 'late fees'. A full evaluation of his usefulness to the Oraculum will be conducted to decide whether or not he is worth the administrative troubles
   of keeping him on site.""],
""Felix4"":[""Year 1402: After the corridor providing access to his cell was lost, Felix was assumed to have been consumed by the building in early 1337. However,
  recently students embarking on a study of the Oraculums lower bowels have encountered a broken window to the outside on the same level as the cell. Hemomancers
  concluded that the blood contained sufficient divine energy to conclude that it belongs to Felix. His whereabouts and condition remain unknown.""],
```

Figure 13: *One of the five Archivist narratives written up for use as comparison against the PCG.*

For this, I initially planned to write all of the data myself - however, I thought as a method to save time I could employ the aid of several writers, each being given a single Archivist. This would increase the distinctness of each historical figure in the game whilst also allowing for a variety in writing style, further adding to the believability of the games environment. I reached out to such a group in a local Discord server I am a part of, dedicated primarily to roleplaying games and other fantasy-based pastimes. In the end, 2 of the 5 handwritten narratives were written by me, with the remaining 3 being written by the aforementioned writers.

I gave them a list of parameters to ensure that any created content would fit within the code structure of the game, and to make it easier to plug it in to a system that was designed with PCG in mind, and set them loose.

Evaluation

Participant Recruitment

Development of both the procedurally generated content and the game to house it were primarily done in the university labs, it was easy to gather a wide selection of participants – both with and without prior experience of PCG in games. I was present for all instances of the participant playing the game, which became useful later on when a problem was identified in the study procedure. When filling out the questionnaire, the participants were required to fill out a small demography section – this was so that any patterns in the results that emerge as a result of age or gender could be identified, however I don't anticipate that there will be any. Whilst age and gender are recorded as a part of the questionnaire, at no point are the names of the participants taken in – this is to maintain anonymity.

Overall, 12 participants were gathered – 10 of which were from the university labs as mentioned, the remaining 2 being family members that were emailed builds of the game to complete during online meet-ups using Microsoft Teams. This was primarily to try and bring some diversity in both age and gender into the study, as otherwise the study would have been 100% male.

Ethical Considerations

There are no inherent physical dangers to playing the created game. Early on in the proposal, it was identified that there was a possibility that procedural grammars could create harmful or offensive content due to a degree of randomness – this ultimately won't be a problem, as the final design of the procedural history system uses prewritten events that include randomised sections, meaning that unless one of the templates was offensive, the final output is very unlikely to be.

The Study

Participants played through the game multiple times each, with at least one playthrough utilising PCG in its content and one using traditional methods. After each individual playthrough, they were presented with an online Google Forms version of the Game Experience Questionnaire. They were two versions of this questionnaire – one for each method of worldbuilding employed in the project. Having two versions allowed the results for each worldbuilding method to be viewed individually, making comparisons easier. It also prevented the need for the player to be informed which version of the game they were playing - the game randomly selects between the two methods whenever you select 'Play' from the main menu, helping to eliminate bias towards any one method of worldbuilding.

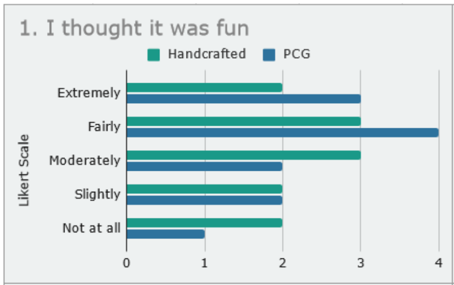
The problem with this however was that playing the game through twice could result in two playthroughs of the same method. Luckily, since I was physically present for each of these playthroughs, I could silently verify which version of the game the participant was playing and bring up the appropriate form once they finished. Since there is no definitive amount of playthroughs needed, I could simply let the player continue playing and replaying the game until they have played each method once.

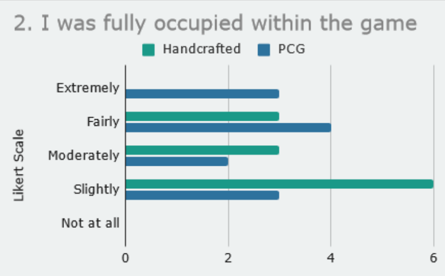
Results

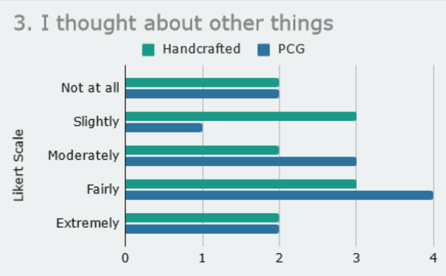
The Game Experience Questionnaire (GEQ) is designed to analyse a number of elements of a game. It does this by asking the player how they felt about a series of generic statements about the game, rating their agreement with those statements on a scale from 1 to 5 – 5 being extremely and 1 being not at all. After this data is collected, The results are used to present quantitative data on a series of components. It does this by averaging the scores of questions relevant to the component being calculated. As the questionnaire being used for this project is a modified version of the In-Game questionnaire, these scoring guidelines will need to be altered to suit the study:

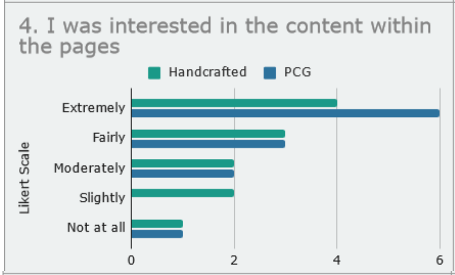
- **Competence:** Items 2, 6 and 8.
- **Sensory and Imaginative Immersion:** Items 4 and 7.
- **Flow:** Items 3 and 12.
- **Tension:** Items 9 and 13.
- **Challenge:** Items 10 and 11.
- **Negative affect:** Items 5 and 14.
- **Positive affect:** Items 1, 15 and 16.

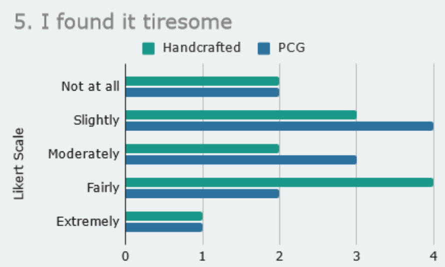
Below is the initial data gathered from the GEQ, organised such that it directly compares the results of each question between the PCG and handcrafted variants of the questionnaire.

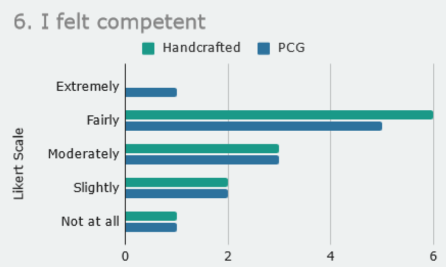

1. I thought it was fun

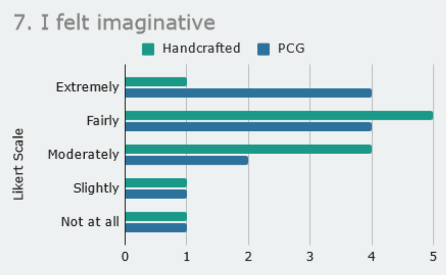

2. I was fully occupied within the game

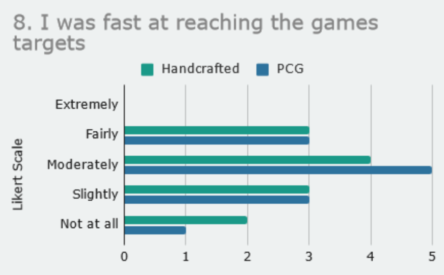

3. I thought about other things

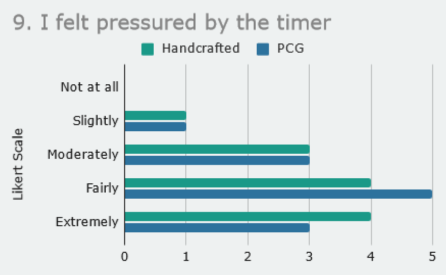

4. I was interested in the content within the pages

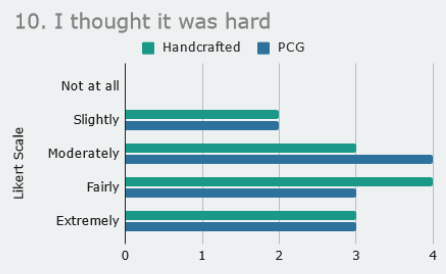

5. I found it tiresome

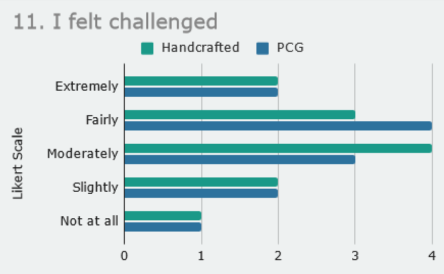

6. I felt competent

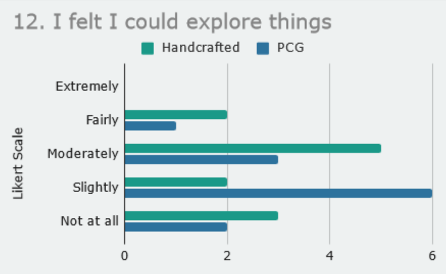

7. I felt imaginative

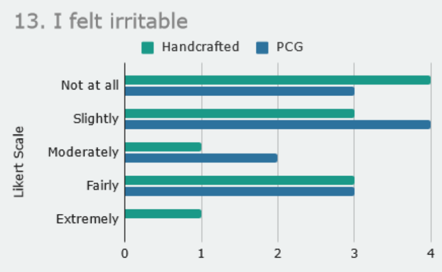

8. I was fast at reaching the games targets

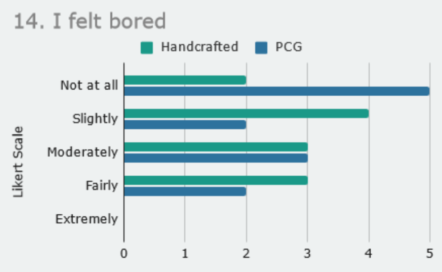

9. I felt pressured by the timer

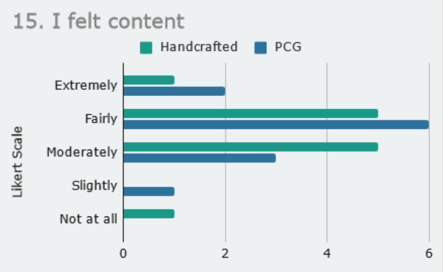

10. I thought it was hard

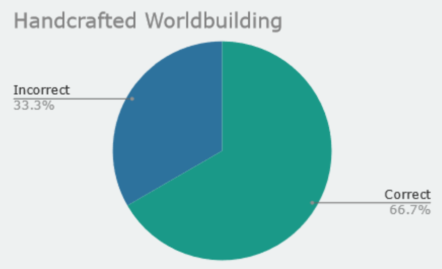

11. I felt challenged

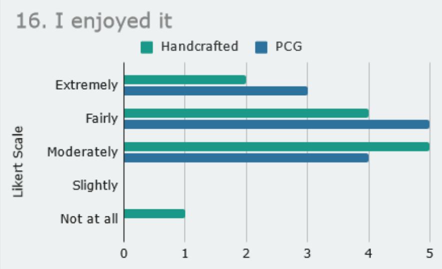

12. I felt I could explore things

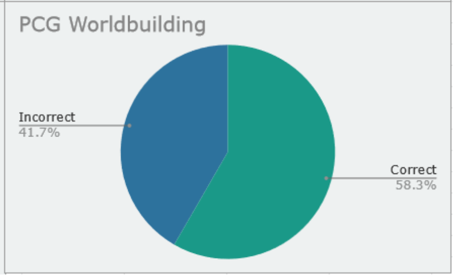

13. I felt irritable


14. I felt bored


15. I felt content


Handcrafted Worldbuilding


16. I enjoyed it


PCG Worldbuilding

A few details can already be spotted without the need to derive an average of all PCG and Handcrafted playthroughs. People seemed (thankfully) interested in the pages spread across the level regardless of what method was in place, but players felt more imaginative and less bored when playing with procedurally generated worldbuilding than they did with the handwritten content. This is especially interesting given that players were 8.4% more likely to recognise that they were playing with handwritten content than they were with PCG – it's possible that players are more interested in reading worldbuilding elements when they are aware its unique to their playthrough, which in turn explains why players on average felt more content during playthroughs employing procedurally generated content.

To draw a direct comparison between the two systems, an average result of each question will be calculated for each of the two methods of worldbuilding. From there, the averages can be combined into their associated components to form an average for that method as a whole.

GEQ Component	Handcrafted	PCG	Difference
Competence	2.86	3.33	+0.47
Sensory and Imaginative Immersion	3.45	3.91	+0.46
Flow	2.75	3.27	+0.52
Tension	3.20	3.12	-0.08
Challenge	3.45	3.45	0.00
Negative affect	2.75	2.41	-0.34
Positive affect	3.33	3.72	+0.39

Figure 14: *A table directly comparing the results of the GEQ for each system.*

Conclusions

On the whole, the data suggests that games are improved by the inclusion of procedurally generated worldbuilding over traditionally written background. Outside of the games difficulty and tension, which remained unaffected (tension showed a difference of 0.08, which is negligible), every component of the GEQ showed an improvement. However, most of these improvements were only minor, suggesting that the answers gained from this study are far from conclusive.

The very small sample size increases the variability of the data, which can lead to bias. Bias could also stem from the methods used to obtain the participants – accruing participants from the university labs limits the data to students, more specifically students that are currently in the midst of computing or computing-adjacent degree fields. This in turn would likely increase the odds of them having encountered various examples of PCG in games before, especially should they happen to be fellow Games Computing students. In an ideal world, the study would include these types of participants with an equal number of people who have not much experience in games and their various systems, allowing them to approach both systems with equal footing and opinion. This could have been mitigated by sending out builds of the game to a greater variety of candidates, increasing both the variability of experience in the subject matter and the diversity of the candidates themselves. Computer science is a male dominated field, and the low sample diversity makes it difficult to determine if things like age or gender would have any marked effect on the results.

It was expected that the systems being compared would not have any effect on the difficulty of the game – in a game concerning the divining of information from text, the difficulty would come with the complexity of the text, not the method by which the text is created. When handwriting the texts used for comparison to PCG texts, care was taken to write them in a similar structure to that which the PCG created during testing – that way we could ensure that participants couldn’t be certain whether the version they were playing was handcrafted or not.

But whilst the systems themselves have no effect on the difficulty, the difficulty of the game overall was much higher than anticipated. Players often found the limited time available to head out into the world and gather pages frustrating – especially those with a degree of awareness as to the projects goals. They were trying to engage with the systems whilst being actively pushed away from them, and whilst this was intentional as part of creating a challenging and engaging experience, the degree to which it affected the players ability to engage with the system resulted in the data being skewed towards those with a higher degree of skill and experience in games of a similar nature.

Overall, the study proves that these systems are indeed at fostering a narrative for the player to experience, and that even with a rather simplistic set of algorithms the types of artefacts generated can rival or even surpass that of their handwritten counterparts. However, there is not enough evidence to suggest that these systems will always be the better option – nor could there really ever be. Despite the impressive variety of content available with a small amount of generative input, creators will sometimes want complete creative control over the narrative decoration in their games, meaning a system like this will not be the natural option even if it were to be provably better in its capability. The natural use for a system like this is always going to be in tandem with handwritten content, particularly when aiding in the populating of much larger levels. The amount of work needed to create 5 archivists and their life stories was relatively easy, but in games where hundreds or thousands of background elements would need creating, PCG is the natural choice to avoid employing and/or overworking large teams of writers.

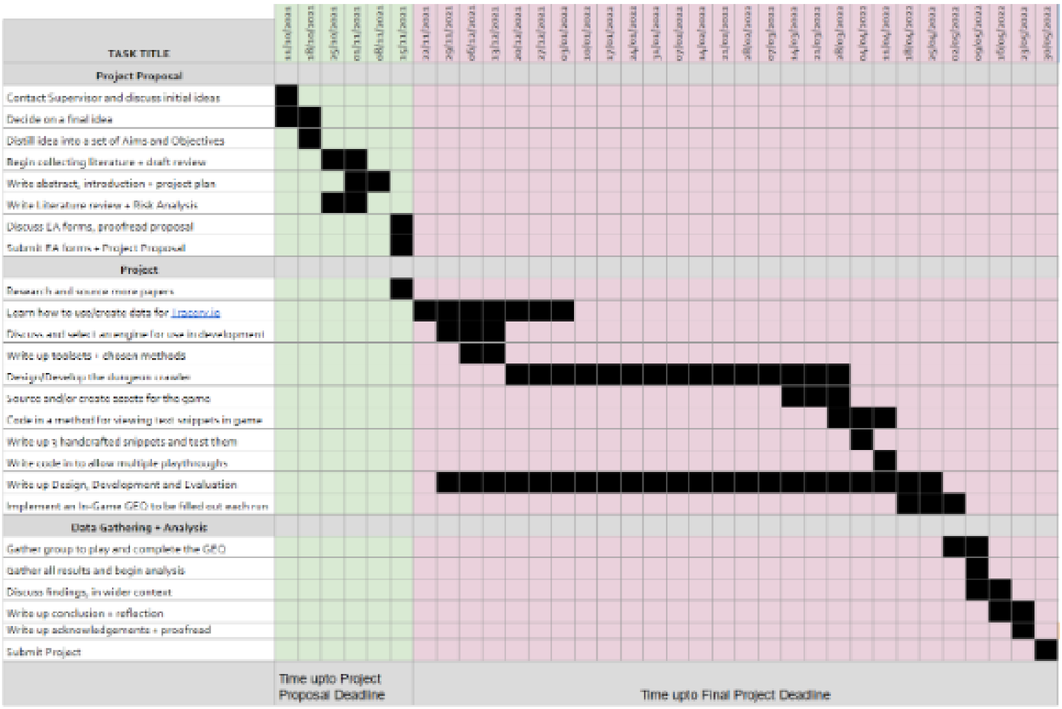
Reflective Analysis

There were many elements of the project that did not meet expectations. There were a multitude of times whereupon poor time management lead to a reduction of scope which – whilst the right move – meant that the obtained results were not as valuable as they could have been. That being said, a lot of valuable lessons were learned as a result of this, and the data obtained was still rather eye-opening.

Far too much time was spent building upon the core mechanics of the game made to house the PCG – this was to the detriment of the PCG, but also the game, as the time spent working on the vision system and perfecting the movement meant the reduction in scope regarding the level design. Some decisions, like the decision to use a single handcrafted environment over levels that are themselves procedurally generated was a smart move – this allowed the player to memorise the locations that pages were likely to be in, and drove them to rush out from the safe zone in the game begun.

One benefit of the extra time spent in development is the engagement – very few players were bored or irritable during the gathering of data, meaning many of them were engaged enough with the content to ensure that the answers we did get were a result of the quality of the systems, and that any negatives identified were not simply the player becoming uninterested. Still, had more time been devoted to the PCG, a more complex algorithm could have been developed to create truly unique and interwoven narratives – the ease players had discerning when content was handwritten was likely due to the variance in writing style, something the PCG content did not have the capability of producing.

A lesson learned from this project is the importance of proper time management and planning – during the proposal of the project, a simple Gantt chart was created that laid out the intended time each section of the project would take, as well as the times and dates of when each following section should be started.



TASK TITLE	11/10/2021	18/10/2021	25/10/2021	01/11/2021	08/11/2021	15/11/2021	22/11/2021	29/11/2021	06/12/2021	13/12/2021	20/12/2021	27/12/2021	03/01/2022	10/01/2022	17/01/2022	24/01/2022	31/01/2022	07/02/2022	14/02/2022	21/02/2022	28/02/2022	07/03/2022	14/03/2022	21/03/2022	28/03/2022	04/04/2022	11/04/2022	18/04/2022	25/04/2022	02/05/2022	09/05/2022	16/05/2022	23/05/2022	30/05/2022
Project Proposal																																		
Contact Supervisor and discuss initial ideas	■																																	
Decide on a final idea	■	■																																
Distill idea into a set of Aims and Objectives		■																																
Begin collecting literature + draft review			■	■																														
Write abstract, introduction + project plan				■	■																													
Write Literature review + Risk Analysis			■	■																														
Discuss EA forms, proofread proposal						■																												
Submit EA forms + Project Proposal						■																												
Project																																		
Research and source more papers						■																												
Learn how to use/create data for Tracery.io							■	■	■	■	■	■	■																					
Discuss and select an engine for use in development								■	■	■																								
Write up toolsets + chosen methods									■	■																								
Design/Develop the dungeon crawler											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■									
Source and/or create assets for the game																							■	■	■									
Code in a method for viewing text snippets in game																									■	■	■							
Write up 5 handcrafted snippets and test them																										■								
Write code in to allow multiple playthroughs																											■							
Write up Design, Development and Evaluation								■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
Implement an In-Game GEQ to be filled out each run																												■	■	■				
Data Gathering + Analysis																																		
Gather group to play and complete the GEQ																														■	■			
Gather all results and begin analysis																															■			
Discuss findings, in wider context																															■	■		
Write up conclusion + reflection																																■	■	
Write up acknowledgements + proofread																																	■	
Submit Project																																		■
	Time upto Project Proposal Deadline						Time upto Final Project Deadline																											

Figure 15: The initial Gantt chart.

Whilst efforts were made to adhere to this chart overall, the primary element of the chart was the large block of time allocated in the centre for 'Game Development'. Not only was this a naïve estimate as to how long the game would take, it also failed to break down precisely what part of the game should be worked on during any one week, only that it should be finished by the 28th of March. In truth, the game was being worked on right up until the end of the project, as many of the elements proved far more difficult for a lone developer to tackle in short order. With the benefit of hindsight, not only would a more thorough plan of the games mechanics and requirements be taken out, but a proper breakdown of those elements in Gantt form would allow for issues and bottlenecks to be identified far earlier – I definitely would have better balanced my time between game development and data gathering given the chance.

Ludography

Caves of Qud (Early Access). 2015. PC [Game]. Freehold Games: Berkeley.

*The Elder Scrolls V: Skyrim (*Legendary Edition). 2013. PC, Xbox 360 and PlayStation 3 [Game]. Bethesda Softworks: Rockville.

Enter the Gungeon (Standard Edition). 2016. PC and PlayStation 4 [Game]. Devolver Digital: Austin.

Darkwood (Standard Edition). 2014. PC, Xbox one and PlayStation 4 [Game]. Acid Wizard Studio: Poland.

Hotline Miami (Standard Edition). 2012. PC, Xbox and PlayStation (various) [Game]. Devolver Digital: Austin.

References

Compton, K. (2015) Tracery: *An Author-Focused Generative Text Tool.* [online] Available from: http://fdg2015.org/papers/fdg2015_extended_abstract_18.pdf [Accessed 7 November 2021]

Compton, K. (2016) *So you want to build a generator?* [blog]. 22 February. Available from: https://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator [Accessed 2 November 2021]

Floridi, L. and Chiriatti, M. (2020). *GPT-3: Its Nature, Scope, Limits, and Consequences* [commentary] February 20. Available from: https://link.springer.com/content/pdf/10.1007/s11023-020-09548-1.pdf

GDC (2016). *Dwarf Fortress, Moon Hunters, and Practices in Procedural Generation.* [vlog]. May 27. Available from: https://www.youtube.com/watch?v=v8zwPdPvN10&list=PL0yMAu5TQBt8iL-Sdd5h1wNvFiFcYK9Hs&index=5 [Accessed 13 September 2021].

GDC (2021). *End-to-End Procedural Generation in Caves of Qud.* [vlog]. June 23. Available from: https://www.youtube.com/watch?v=jV-DZqdKlnE&list=PL0yMAu5TQBt8iL-Sdd5h1wNvFiFcYK9Hs&index=4 [Accessed 13 September 2021].

GDC (2016*). Interior Design and Environment Art: Mastering Space, Mastering Place.* [vlog]. November 12. Available from: https://www.youtube.com/watch?v=WWXsmnlmADc&list=PL0yMAu5TQBt8iL-Sdd5h1wNvFiFcYK9Hs&index=2 [Accessed 6 November 2021]

GDC (2018a). *Procedurally Generating History in Caves of Qud.* [vlog]. June 12. Available from: https://www.youtube.com/watch?v=H0sLa1y3BW4&list=PL0yMAu5TQBt8iL-Sdd5h1wNvFiFcYK9Hs&index=3 [Accessed 5 September 2021]

GDC (2018b). *Ten Principles for Good Level Design.* [vlog]. February 9. Available from: https://www.youtube.com/watch?v=iNEe3KhMvXM&list=PL0yMAu5TQBt8iL-Sdd5h1wNvFiFcYK9Hs&index=1 [Accessed 4 November 2021]

Grinblat, J. and Bucklew, C., (2017*). Subverting Historical Cause & Effect: Generation of Mythic Biographies in Caves of Qud.* [online] Available from: https://www.freeholdgames.com/papers/Generation_of_Mythic_Biographies_in_CavesofQud.pdf [Accessed 27 October 2021].

Hall, H., Williams, B. and Headleand, C., (2017) *Artificial Folklore for Simulated Religions.* Lincoln, UK [online]
Available from: https://ieeexplore.ieee.org/document/8120332

IJsselsteijn, W.A., de Kort, Y.A.W., Poels, K. (2013) *Game Experience Questionnaire.* [online] Available from: https://pure.tue.nl/ws/files/21666907/Game_Experience_Questionnaire_English.pdf [Last Accessed 23 May 2022]

Johnson, D., Gardner J., Perry R. (2018) *Validation of two game experience scales: The Player Experience of Need Satisfaction (PENS) and Game Experience Questionnaire (GEQ).* [online]
Available from: https://bit.ly/3qPJFBV [Accessed 6 November 2021]

Johnson, D., Watling, C., Gardner, J., Nacke, L. (2014) *The Edge of Glory: The Relationship between Metacritic Scores and Player Experience.* [online] Available from:

https://dl.acm.org/doi/pdf/10.1145/2658537.2658694?casa_token=7quzFk9KTMEAAAAA:lJgwxXjEJkdS_F89cHIVkzqlADmirpmeYgCd-RbZvrqNfJI1-UnsVDOx2lpaxaKE2wli23DBekUk [Accessed 6 November 2021]

Johnson, M. (2016) *Towards Qualitative Procedural Generation.* [online] Available from: https://web.archive.org/web/20170610155631id_/http://www.ccgworkshop.org:80/2016/Johnson.pdf [Accessed 10 November 2021].

James Vincent (2019) *OpenAI's new multitalented AI writes, translates, and slanders.* The Verge [article] Available from: https://www.theverge.com/2019/2/14/18224704/ai-machine-learning-language-models-read-write-openai-gpt2 [Accessed 12 January 2022]

Word Count: 9392