UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Procedural Content Generation for Emergent Gameplay

by Kyle T. Hammer

Thesis submitted in fulfilment of the requirements for the degree of

Master of Science (Research) in Computing Sciences

under the supervision of Dr. Jaime Garcia & Prof. William Raffe

University of Technology Sydney

Faculty of Engineering and Information Technology

October 2023

Certificate of Original Authorship

I, Kyle Hammer, declare that this thesis is submitted in fulfilment of the requirements for the award of Master of Science (Research) In Computing Sciences Degree, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Kyle T. Hammer

Acknowledgement

I would like to express significant gratitude to my supervisor, Dr. Jaime Garcia, for his consistent guidance, mentorship, and invaluable insights throughout the research process. He always made time to offer advice and critiques whenever required and has been supportive both inside and outside the academic space. Dr. Garcia encouraged and inspired me to take up research, which I will always be thankful for. His dedication and expertise have been the pillars supporting my work, and the work within this thesis could not have been completed without his guidance.

I would also like to extend my sincerest thanks to Associate Professor William Raffe, my co-supervisor, whose constructive critiques and support significantly contributed to shaping this research. In addition, he has also supported me in my future game development endeavours by giving me the necessary skills throughout my Bachelor's and Master's Degrees, which I will be forever thankful for. His expertise in the field provided valuable insights for my research journey and undoubtedly shaped the foundations of my research.

A special thank you goes towards Associate Professor Nico Pietroni, who served as the chair for my progression assessments, offering valuable feedback and direction for the project throughout multiple stages of my research journey. I would also like to offer special thanks to Dr

Baki Kocaballi, the assessor expert for my progressions assessments, for their thorough evaluations and constructive feedback, which were indispensable in enhancing the quality of this thesis.

Special recognition goes to the Australian Government for its Research Training Program, which has provided me with a scholarship to support my research endeavours financially. This opportunity was critical in allowing me to focus on my academic goals.

My appreciation goes to the University of Technology Sydney for providing a supportive and conducive academic research environment. Their facilities, resources and community were integral to completing my research project.

I would like to give thanks to the volunteers who expressed interest in my research project by participating in the "Procedural Content Generation for emergent gameplay" prototype experiment. Their input and time spent were invaluable to this research, and I could not have obtained significant enough data without them.

To all my friends who have been with me throughout this journey, your emotional and moral support has been invaluable. You helped me through the difficult moments in life and made the good times even better.

Finally, a very special thank you to my family: My parents, Nenita and Alex Hammer, my siblings, Liam and Chloe Hammer, and my fellow academic Uncle Norman Domigpe. Your love, encouragement, and faith in my abilities have motivated me through this path in academia. Thank you for your support throughout my research journey and life.

Abstract

The addition of emergent gameplay is a highly sought-after feature in games for both players and developers alike. Emergent gameplay refers to occurrences in games where the interaction of systems results in something unexpected, exciting and surprising, similar to a chain reaction that provides additional benefits or challenges for the player. Emergence in games does come with its own design challenges, but its inclusion offers excellent benefits by rewarding players' game knowledge when experimenting with interactions.

Many well-renowned games that involve emergent gameplay are games that make use of procedurally generated content (PCG), which allows for a more expansive spectrum of possibilities for the player to take. A few factors that increase emergence have been identified in the current literature; however, the list can be expanded and investigated further. For instance, the instability factor, most commonly achieved through procedural generation, has been seen to create more opportunities for emergent gameplay but is only occasionally necessary. Games with minimal randomness, like *Legend of Zelda: Breath of the Wild* (2017), have the potential for many emergent possibilities despite this.

This thesis aims to investigate the following research questions to obtain a better understanding of the nature of emergence and how to create it:

- How do developers design for emergent gameplay?
- How can PCG be used to assist with the creation of emergent scenarios?

The first part of the investigation examines academic literature, developer conferences and talks to find the common factors between systemic games that help create emergence. The findings from these sources will provide multiple perspectives to answer how developers design games for emergent gameplay. This will also be used to form the foundations of an emergent framework. The framework's objective will provide developers with an easy-to-understand chart to evaluate their emergent game development projects and compare them with other emergent games. Another part of the investigation will involve investigating several generative techniques and creating a framework to identify which PCG methods will be suitable for facilitating an emergent environment. This will reveal the relationship between PCG and emergence and show how generative techniques can support the creation of emergent scenarios. After the completion of the frameworks, a testing session will commence where participants will play a short roguelike prototype with their in-game actions observed for emergent behaviour. The results of the testing session can then be charted onto the framework to understand what elements of the prototype can be improved to create more emergence.

Contents

Certificate of Original Authorship 2
Acknowledgement 1
Abstract 2
Contents 4
List of Figures and Tables 8
1. Introduction 1
1.1 Introduction 1
1.2 Overview and Research Questions 1
1.3 Hypothesis 5
1.4 Research Purpose and Significance 6
1.5 Research Methodology 6
1.6 Key Contributions 8
1.7 Thesis Structure 8
1.8 Conclusion 10
2. Literature Review 11
2.1 Introduction 11
2.2 Roguelikes 11
2.3 Emergence 15
2.4 Procedural Content Generation 22
2.5 Methodology 27
2.6 Conclusion 29
3. Development of The Framework 30
3.1 Introduction 30
3.2 Factors for Emergence - Established Factors 30
3.2.1 Increasing Interactions 30
3.2.2 Reducing Constraints 31
3.2.3 Adding More Subjects 32
3.2.4 Creating Instability 32
3.2.5 Adding Meaning 33
3.3 Factors for Emergence - Additional Factors 33
3.3.1 Keeping Rules Consistent 34
3.3.2 Offering More Player Agency 34

3.3.3 Encouraging Experimentation.......35
3.4 PCG for Emergence.......37
3.4.1 PCG Games Following Emergent Factors.......37
3.4.2 Tile-Based Approaches.......37
3.4.3 Emergence through Storytelling.......38
3.4.4 Creating Instability.......38
3.5 Conclusion.......39
4. Applying the Framework....... 40
4.1 Introduction.......40
4.2 The Emergent Framework.......40
4.2.1 Spelunky.......41
4.2.2 Noita.......41
4.2.3 The Legend of Zelda: Breath of the Wild.......42
4.2.4 Minecraft.......43
4.2.5 Don't Starve.......43
Table 4.2: Classifying games under the Emergent Framework.......44
Figure 4.2: Chart representing the games under the Emergent Framework.......44
4.3 PCG for Emergence Framework.......45
4.3.1 Spelunky Map Generation Method.......47
4.3.2 Spelunky Room Generation Method.......47
4.3.3 Random Walk.......48
4.3.4 Binary Space Partitioning.......48
4.3.5 Cellular Automata.......48
4.3.6 Wave Function Collapse.......49
4.3.7 Grammars.......50
4.3.8. Constraint Solvers.......50
Table 4.3 PCG for Emergence Framework. A low score represents high instability, while a high score represents high consistency.......51
Figure 4.3 Chart representing the PCG for Emergence Framework.......51
4.3.9 Framework Effectiveness.......52
4.4 Conclusion.......52
5. Developing a Prototype for Emergence....... 53
5.1 Introduction.......53
5.2 Features vs. System Requirements.......53
Figure 5.2.1 The input samples for Wave Function Collapse.......55
Figure 5.2.2 The output samples for level 1, generated by Wave Function Collapse 56
Figure 5.2.3. The Spelunky Map Generative Method, shown to generate a path from the start to the goal.......57
5.3 Using the Prototype - Project Quiver.......59
Table 5.3.1: Movement and Camera Controls.......60
Table 5.3.2: Bow and Arrow Controls.......60
Table 5.3.3: Application controls.......60
Figure 5.3.1: The Project Quiver menu screen.......61

Figure 5.3.2: The player performing a perfect shot........ 62
5.4. Prototype Potential........ 62
5.5. Conclusion........ 62
6. Validating the Prototype through the Framework........ 64
6.1 Introduction........ 64
6.2 Preparation for Prototype Testing........ 64
6.3 Prototype Observations........ 67
6.4 Prototype Analysis........ 68
6.4.1 Emergent Occurrences........ 68
6.4.2 Analysis against the Emergent Framework........ 70
Figure 6.4.1a: Project Quiver's perceived emergence rating on a radar chart........ 71
Table 6.4.2b: Project Quiver's perceived emergence rating vs emergent games.. 72
Figure 6.4.2b: Project Quiver's perceived emergence rating on a radar chart vs emergent games........ 72
Table 6.4.3: Participants perception of emergent factors........ 74
Figure 6.4.3 Participants' perception of emergent factors on a bar chart........ 74
6.4.4 Participant Data Analysis........ 75
6.4.5 Justification behind the ratings - Emergent Factors........ 77
6.4.6 Justification behind the ratings - Project Quiver........ 79
6.5 Conclusion........ 81
7. Outcomes and Conclusion........ 82
7.1 Introduction........ 82
7.2 Research Outcomes........ 83
7.3 Limitations and Future Directions........ 84
Bibliography........ 87
1. Referenced Papers........ 87
2. Other Sources........ 88
Appendices........ 90
Appendix A - Designing Project Quiver........ 90
A.1 Project Quiver's Design Map Part 1........ 90
A.2 Project Quiver's Design Map Part 2........ 91
A.3.0 Project Quiver Feature to Emergent Factors - Key and Ratings........ 92
A.3.1 Project Quiver Feature to Emergent Factors - Procedural Elements........ 93
A.3.2 Project Quiver Feature to Emergent Factors - Game Systems........ 94
A.3.3 Project Quiver Feature to Emergent Factors - Items........ 96
A.3.4 Project Quiver Feature to Emergent Factors - AI........ 97
A.3.5 Project Quiver Feature to Emergent Factors - Challenges........ 98
Appendix B - Project Quiver Observation Data........ 99
B.0 Appendix B data........ 99
B.1 Participant understanding of emergence........ 99
B.2 Elements of the Emergent Definition*........ 99
B.3 Participant Playstyle/Prioritisation........ 100
B.4 Frequently used resources........ 100
B.5 Bow Charge Style........ 101

B.6 Participant Initial Skill Level ... 101
B.7 Participant Skill Level Change Over Time ... 101
B.8 Participant Furthest Level ... 102
B.9 Participant Highest Score ... 102
B.10 Participant Understanding of Level Generation ... 102
B.11 Participants Perception of Emergence ... 103
B.12 Presence of unintended scenarios ... 103
B.13 Participants on Factors that Increase Emergence ... 104
Appendix C - Emergent Scale Rubric ... 105

List of Figures and Tables

4. Applying the Framework....... 40
Table 4.2: Classifying games under the Emergent Framework....... 44
Figure 4.2: Chart representing the games under the Emergent Framework....... 44
Table 4.3 PCG for Emergence Framework. A low score represents high instability, while a high score represents high consistency....... 51
Figure 4.3 Chart representing the PCG for Emergence Framework....... 51
5. Developing a Prototype for Emergence....... 53
Figure 5.2.1 The input samples for Wave Function Collapse....... 55
Figure 5.2.2 The output samples for level 1, generated by Wave Function Collapse 56
Figure 5.2.3. The Spelunky Map Generative Method, shown to generate a path from the start to the goal....... 57
Table 5.3.1: Movement and Camera Controls....... 60
Table 5.3.2: Bow and Arrow Controls....... 60
Table 5.3.3: Application controls....... 60
Figure 5.3.1: The Project Quiver menu screen....... 61
Figure 5.3.2: The player performing a perfect shot....... 62
6. Validating the Prototype through the Framework....... 64
Figure 6.4.1a: Project Quiver’s perceived emergence rating on a radar chart....... 71
Table 6.4.2b: Project Quiver’s perceived emergence rating vs emergent games.. 72
Figure 6.4.2b: Project Quiver’s perceived emergence rating on a radar chart vs emergent games....... 72
Table 6.4.3: Participants perception of emergent factors....... 74
Figure 6.4.3 Participants' perception of emergent factors on a bar chart....... 74
Appendices....... 90
Appendix A - Designing Project Quiver....... 90
A.1 Project Quiver’s Design Map Part 1....... 90
A.2 Project Quiver’s Design Map Part 2....... 91
A.3.0 Project Quiver Feature to Emergent Factors - Key and Ratings....... 92
A.3.1 Project Quiver Feature to Emergent Factors - Procedural Elements....... 93
A.3.2 Project Quiver Feature to Emergent Factors - Game Systems....... 94
A.3.3 Project Quiver Feature to Emergent Factors - Items....... 96

A.3.4 Project Quiver Feature to Emergent Factors - AI.....97
A.3.5 Project Quiver Feature to Emergent Factors - Challenges.....98
Appendix B - Project Quiver Observation Data.....99
B.0 Appendix B data.....99
B.1 Participant understanding of emergence.....99
B.2 Elements of the Emergent Definition*.....99
B.3 Participant Playstyle/Prioritisation.....100
B.4 Frequently used resources.....100
B.5 Bow Charge Style.....101
B.6 Participant Initial Skill Level.....101
B.7 Participant Skill Level Change Over Time.....101
B.8 Participant Furthest Level.....102
B.9 Participant Highest Score.....102
B.10 Participant Understanding of Level Generation.....102
B.11 Participants Perception of Emergence.....103
B.12 Presence of unintended scenarios.....103
B.13 Participants on Factors that Increase Emergence.....104
Appendix C - Emergent Scale Rubric.....105

Chapter 1

Introduction

1.1 Introduction

The project investigates the concepts of emergence and procedural content generation (PCG) in games and explores their relationship. Emergent gameplay refers to the occurrences where the interaction of game systems results in something unexpected and often exciting, similar to a chain reaction that provides additional benefits or challenges for the player. Emergence in games does come with its own design challenges, but its inclusion offers excellent benefits by rewarding player's game knowledge when experimenting with interactions. While not a requirement for emergence, PCG has been shown in existing games to be an effective tool for generating unique game scenarios, requiring extra creativity for players to solve. The relationship between PCG and emergence is evident in genres like the generated open-world and roguelikes.

1.2 Overview and Research Questions

Emergent behaviour in games is typically defined as "...the appearance of new possibilities that arise from the interplay between game mechanics" (Juul, 2005). Many games are designed around this fact, where game systems interact with each other to create scenarios that lead to unpredictable situations and more exciting player stories.

Examples of interplaying mechanics are easily encountered in The Legend of Zelda: Breath of the Wild (2017), thanks to its many systems and interactable objects. As an example of an emergent scenario, a player may come across an enemy encampment filled with treasure chests and exploding barrels during their travels. Their initial plan is to shoot a flaming arrow to set off the exploding barrels from a distance and clear out the camp; however, after firing the arrow, they realise that the thunderstorm overhead extinguishes the flame, making the arrow useless to their plans. With this new revelation, the clever player decides to pivot their plan and sneak closer, throwing a rusty broadsword to the barrels once in close enough proximity. Although this does not have an immediate effect, it is not long before the sword begins to spark and soon enough, lightning strikes down on the sword, setting off the explosive barrels and clearing the enemy camp for the player to reap all the rewards. The interplay between the player's weaponry and the game's weather system initially thwarted the player's plan but also opened new routes. Both are examples of emergent gameplay in action.

Emergent gameplay has a widespread appeal to players and developers for several reasons. Players can use a game's dynamic systems to formulate unique, meaningful plans and make choices that are not limited to the binary options of other games. An interplay of mechanics can generate new situations throughout a game's world, allowing developers to bypass the need to generate individual scripted events. Players who come across these situations can observe the behaviour of these interactions and discover ways to utilise them to their advantage. A player might even discover a system interaction that was unintended by the game's designers (Leijnen & Veen, 2016) but later refine and implement it as a more common mechanic to facilitate more player options. Emergence can also create drama when an unexpected interplay of mechanics occurs to benefit or put a foil in a player's plans.

Building a game for emergent gameplay is challenging for developers for several reasons. One common hurdle for developers is the unexpected consequences that can arise from game mechanic interaction. For example, if a developer is to add water to their game for the player to dive and swim through, they will need to decide how the water will interact with other game elements. Will enemies also have the ability to swim? Does the player require a breath meter? Should enemies also have a breath meter? These mechanics and their corresponding interactions must be considered and reconsidered upon introducing new dynamic systems.

Even before implementing dynamic systems and player testing, it can be daunting to identify effective ways to make a game more emergent. A lack of knowledge of the nature of emergence could result in an under-utilisation of emergent systems in the games industry, resulting in fewer systemic games, a term used to describe games that contain systems linked and designed to influence each other.

Another challenge for emergence is the reliance on the player's actions. Unlike a game's tutorial or storyline, emergence cannot be forced upon the player due to its nature, only incentivised. This player-driven design philosophy makes it so that even if a developer plans to create an emergent game, there is no guarantee as to whether the players will take the opportunity to encounter these unique scenarios. Although there are ways to increase the likelihood of an emergent scenario, it will take frequent player testing and observations to get the emergent behaviour right. The challenge of creating emergence formulates the first research question, "How do developers design games for emergent gameplay?"

Some research has already been conducted on increasing the frequency of emergent scenarios. A design tool has been developed that lists additional methods to create more emergence: creating new interactions, removing player constraints, adding more subjects, adding connections between game mechanics and real life and creating instability (Paananen, 2020).

The design tool proves to be successful in creating instances of emergence. However, further experimentation may need to be conducted to evaluate the tool's effectiveness. Investigating additional ways to create emergence can expand on the existing tool. Another extension could be adapting the tool to apply to specific genres or areas of games, such as roguelikes or procedural content generation. Procedural content generation, or PCG for short, is a programmatic way to generate content. Through the use of specialised algorithms, an extensive range of varied content can be created through automatic generation (Raffe, 2014). For example, PCG can be responsible for generating something simple like the weapon stats in Borderlands (2009) or something far more complex like the vast worlds of No Man's Sky (2016). A strong focus on the emergence of PCG or a particular genre like roguelikes can increase the appeal to players, especially the archetype of players that enjoy experimenting and exploiting a game's systems.

The instability property of generating emergence is a particular area of interest as it involves using randomness to create novel scenarios and allows the player to experiment within those scenarios. Randomness is commonplace in games and forms the backbone of many games that rely on PCG to create unique scenarios for the player to encounter. The flexibility of PCG algorithms has made it very popular for game developers, and the randomness of PCG often results in a higher replayability factor due to the extra gameplay variety.

PCG is a vast research space with various ways to randomise countless subjects. It can be challenging to pick out a suitable algorithm that works best for the designer's vision while maintaining the emergent nature of a game. Furthermore, it is very challenging to predict whether the chosen procedural method will positively increase emergence and whether it will be more effective than an alternative algorithm. These issues pose another research question: "How can PCG be used to create emergent scenarios?"

The thesis aims to extend the knowledge of emergence in games past the state represented in current academic literature. The extended outcome will then be developed into a framework that can be used as an evaluation tool to see why existing games are successful or are struggling with creating emergence. In addition, the relevant factors should also be applied to PCG to measure their emergent possibility space, which may reveal a more significant link between PCG and emergent gameplay. Finally, a prototype will be developed based on the framework's principles and tested for occurrences of emergence and its impact on gameplay. The prototype should be created in a game genre that promotes PCG and emergent interactions. The roguelike genre was chosen for this purpose as it heavily relies on its PCG and has the potential to facilitate an emergent environment. The open-world genre was also considered, however, roguelikes were considered in scope for the project and better suited for rapid prototyping.

1.3 Hypothesis

After conducting the literature review, the following issues were identified:

- The current state of literature represented for emergence in games (RQ1. How do developers design games for emergent gameplay?)
- The use of procedural content generation to create emergent scenarios (RQ2. How can PCG be used to create emergent scenarios?)

These would then be developed into the following hypotheses:

- The current state of the literature on emergence in games can expand to include additional factors and techniques
- Procedurally generated content can be evaluated under the lens of emergence to help promote its inclusion in games

1.4 Research Purpose and Significance

Most sources published mention emergence as one of the advantages of PCG but often choose a different topic path to accompany it, such as using AI in PCG or finding the best-suited PCG method to suit their project needs. This leaves a gap for further investigation to be done on how PCG creates emergence as well as how it can have an impact on gameplay.

Emergent gameplay is a sought-after feature in many games as it gives players a sense of self-accomplishment for finding solutions the developers may not have planned. However, due to the unpredictable nature of emergence, creating numerous or meaningful emergent scenarios proves challenging to design. A better understanding of emergence and its applications in games can be obtained by studying the nature of emergence.

Procedural content generation is a more heavily researched field of study in game development for many reasons. It has versatile applications for many game systems and a solid appeal to developers who use it for varied content and replayability. The area of PCG continues to grow with new methodologies and techniques being developed for broad and niche applications. A deeper look into several different PCG methods can help shed light on what PCG attributes and techniques are better suited to creating emergent gameplay scenarios. It could also result in discovering new PCG techniques or finding a new hybrid of PCG techniques that work well together to create emergent gameplay.

1.5 Research Methodology

The research phases are as follows:

Phase 1: Literature review

- Evaluate the existing literature on procedural content generation, roguelikes and emergent gameplay.

- After gaining a foundational knowledge of the topics of interest, conduct a literature review to present the current state of the literature.
- Implement different procedural algorithms to understand better how they function.

Phase 2: Development of the framework

- Widen the search for procedural algorithms to consider for emergent testing.
- Discover the factors that PCG has that enable the creation of more emergence.
- Perform more PCG experiments to understand the generative methods further.
- Develop a framework to make comparisons between generative methods to streamline the process of choosing a generator for an emergent game.
- Look into factors for increasing emergence that may be underrepresented or absent from current literature.
- Develop a framework based on established principles and newly discovered factors of creating emergence.

Phase 3: Applying the framework

- Apply the framework to commercial emergent games as an evaluation tool.
- Choose a combination of PCG algorithms that would be a suitable mix of randomness and consistency for an emergent game prototype under the newly established framework.
- Develop a PCG prototype that will be tested for emergence scenarios, keeping the emergent factors in consideration.
- Recruit participants to test the prototype for emergent gameplay.

Phase 4: Analysing the results under the framework

- Participants to test the prototype, with observations made on emergent gameplay occurrences.
- Analyse the results from the noted observations.
- Resulting data to be applied to the Emergent Framework to identify areas of strength and weakness.

1.6 Key Contributions

The key contributions of this thesis are as follows:

- A modern, wide-ranging literature review on emergent gameplay, with information on the benefits it provides to players and developers and the current methodology used to create it. Supplementary literature reviews on procedural content generation and emergent gameplay have also been conducted for their link with emergent gameplay.
- The Emergent Framework: A framework used to support the development of emergent games, built upon the factors that promote emergence to point out areas of weakness.
- The PCG for Emergence Framework: A supplementary framework to the Emergent Framework to compare generative methods for their instability versus consistency to find the ideal method or combination of methods for their emergent game.
- The validation of the framework by applying it to commercial emergent games and to *Project Quiver*, a game prototype looking to increase interesting and engaging emergent interactions.

1.7 Thesis Structure

Chapter 1 introduces the topic of emergence and the challenges of developing a game with emergent systems in mind. Through this chapter, research questions have been developed to gauge the current understanding of emergence, how the field can be expanded, and discover what role procedural content generation plays in this process.

Chapter 2 establishes a background for the research by investigating the current literature on emergence, PCG and roguelikes. The investigation results in the development of a literature review that aims to be an in-depth look at the topics and establish potential gaps in the literature, some of which will be investigated further in the thesis.

Chapter 3 covers the foundational development of the framework by compiling the currently known literature into factors that are used to increase emergence. It also expands on this list by adding additional factors to creating emergence and takes a deeper look into how PCG can be used to facilitate an emergent game.

Chapter 4 uses the framework foundations covered in **Chapter 3** and applies them to released titles to evaluate and compare them under the lens of emergence. In addition, several PCG algorithms were also selected and evaluated under the PCG for Emergence Framework, which compares the instability and consistency of the algorithms against each other.

Chapter 5 uses the Emergent Framework's factors to support the development of an emergent game prototype, *Project Quiver*. The prototype considers the effect each feature has on emergence and selects a suitable mix of PCG algorithms through the PCG for emergence framework.

Chapter 6 looks to test the Project Quiver prototype for emergence in an experiment that observes the participant's interactions. The experiment participants also take part in an interview to gauge their familiarity with the concepts of emergence, revealing what the general gaming audience believes to be emergent. At the end of the experiment, the results will be analysed and charted on the Emergent Framework to identify the factors the prototype might be lacking.

Chapter 7 gives a summary of the information learnt from the literature review, the development of the framework and the experiment. The results will provide an overview of emergence, PCG and roguelikes and specify the directions the prototype can take to become more emergent.

1.8 Conclusion

In summary, this chapter establishes the groundwork for an in-depth exploration of emergent gameplay and a glimpse into how procedural content generation PCG contributes to emergence. To further understand these mechanics and their influences on player experiences, the research methodology breaks down this process into the phases that will be covered in the following chapters, starting with the literature review in Chapter 2. This study aims to compile a set of principles that game developers can refer to and chart on the emergent framework. The emergent framework is to be used as a tool to support the creation of more emergent games.

Chapter 2

Literature Review

2.1 Introduction

This literature review aims to delve deep into three intertwined concepts—roguelikes, emergent gameplay, and procedural content generation (PCG), and discuss the links between the concepts. The information covered in this chapter is from various sources, mainly from academic journals and conference papers. The methodology section goes into more detail in the selection process behind the sources selected for the review. The literature review hopes to shed light on the current state of knowledge and identify research gaps that future studies can cover.

2.2 Roguelikes

The roguelike genre of games has its roots in 1980 with the release of the game known as *Rogue* (Izgi, 2018). Developed by a small team in Unix, *Rogue* was a game that used an ASCII display to represent a dungeon crawling experience, complete with a main character, fightable monsters and items all represented through different letters. Its thematic inspirations tie back to fantasy novels like J. R. R. Tolkien's books (Izgi, 2018) and tabletop experiences such as the increasingly popular *Dungeons & Dragons* (Ho et al., 2016), which also contributed to its growth in popularity upon distribution. Thanks to its appealing theme and unique game mechanics, which was not seen then, *Rogue* had grown massively in popularity.

A combination of *Rogue's* core features was what ended up spawning the roguelike genre. A roguelike game's primary game design components are PCG, short for procedural content generation, and permadeath, short for permanent death (Wilson, 2020). PCG techniques are considered the backbone of roguelike games as they are responsible for generating land, maps, items, rewards, enemies and resources of the game. PCG is not necessarily limited to those factors as it can also be applied to audio, dialogue trees, mission objectives, behaviour patterns and more (Melotti & Moraes, 2019) depending on the needs of the roguelike. Permadeath is a mechanic that works contrary to many modern games. It punishes players for failure by permanently killing the player character upon losing, removing their current progress (Wilson, 2020).

Although permadeath may seem like a harsh mechanic at a glance, this system can support the growth of player mastery, where the player must progress through the game multiple times to learn from their previous mistakes and make improvements accordingly. Ebia and others in *Influencing Game Dynamics in A Roguelike Game Through Procedural Content Generation Using Genetic Algorithm* mention how permadeath in roguelikes works well with its PCG counterpart (Ebia et al., n.d.). Permadeath can incentivise replayability thanks to new and exciting playthroughs and experiences whenever a new game starts. Wilson (2020) evaluates permadeath in the context of roguelikes to gauge its effect on player engagement. It helps shed light on how the permadeath mechanic may affect playtime, perceptions of fun, perceptions of content, perceptions of difficulty and discourages death. The findings show that permadeath increased the user's perception of fun and discouraged deaths but did not increase user playtime or the player's perception of difficulty.

The discouragement of death corroborates Parker's (2017) evaluation of the effects of permadeath in roguelikes. The paper, *The Culture of Permadeath: Rogulikes and Terror*

Management Theory, evaluates roguelikes through the lens of terror management theory and discusses how it can be applied to increase the resilience of the roguelike community.

Despite the previous studies on the permadeath mechanic, its inclusion in games and its effects on players still have little coverage, leaving a gap for more studies to understand its nature better. The advantages and disadvantages of the mechanic could be analysed more thoroughly to discover whether its inclusion in games would benefit the core gameplay loop. Permadeath is known to have a profound effect on replayability but also has a reliance on its pairing factors like PCG. This area could be investigated to discover how the mechanic could increase game replayability. Permadeath is also known to affect player mastery, how much a player improves in a game over time, although how much it affects players compared to a non-permadeath game has yet to be evaluated.

Standard features in roguelike games include a high level of difficulty (Wilson, 2020), character progression (Ebia et al., n.d.) and factors that are a part of the Berlin Interpretation. The Berlin Interpretation, created in 2008, was a definition developed to better identify games as roguelikes through a series of high and low-value factors (Izgi, 2018). These factors are as follows:

High-Value Factors

- Procedural World Generation
- Permadeath
- Turn-Based Combat
- Grid-Based Combat
- Non-modal
- Complexity
- Resource Management

- Player vs. World
- Exploration and Discovery

Low-Value Factors

- Single player character
- Player-like monsters
- Tactical challenge
- ASCII display
- Dungeon like level generation
- Presentation through numbers

With the Berlin Interpretation being defined in 2008, it has had its fair share of controversy as to whether the definition holds up to the modern-day games industry. The roguelike community have argued that its definition is outdated and restrictive for such an open genre (Izgi, 2018) and that adding selective factors to the definition does not necessarily make a game more roguelike. With roguelikes becoming more commonplace in modern games, these combative arguments resulted in the definition being condensed down to its core elements, PCG and permadeath.

A comparison between some of the earlier and later roguelike games may help better identify how the genre has evolved. Xavier Ho and others discuss some of the earliest roguelike influences in their paper *Finding Design Influence within Roguelike Games* (Ho et al., 2016). *Hack* (1982) and later *NetHack* (1987) could be seen as some of the earliest examples of roguelikes as they built upon the mechanics of *Rogue* with an expanded arsenal of items and monsters.

Further evolutions included *Angband* (1990) with its one hundred levels of generated dungeons and the Japanese title responsible for spawning the Mystery Dungeon series *Torneko no*

Daibōken: Fushigi no Dungeon (1993), which translates to *Torneko's Great Adventure: Mystery Dungeon*. More recent titles of the roguelike genre include *Spelunky* (2008) and its sequel, *Spelunky 2* (2020), a cave exploration game that uses roguelike concepts with platformer mechanics. *Slay the Spire* (2017) is a deckbuilding game that cleverly uses roguelike progression and player choice to build the player’s deck. By comparing the modern iterations of roguelikes to the past, the genre's evolution becomes clearer. Many of today’s roguelikes are less restricted to the dungeon-like approaches of the past and experiment with mixing unique mechanics and genres to create novel experiences and new subgenres.

2.3 Emergence

Emergent behaviour or emergence is a general term that can be used to define the occurrence of unpredictable behaviour resulting from dynamic systems (Ampatzidou, 2019). With the interaction of systems creating new behaviours, emergent products end up being referred to as a whole that is worth more than the sum of its parts. This definition is no different in games, where the simple systems of a game create complex behaviours, often resulting in emergent gameplay (Hokkanen et al., 2018).

Developers typically start planning for emergent gameplay by looking at systems as a series of inputs, outputs, and rules. British video game journalist Mark Brown, most well known for his work as *Game Maker’s Toolkit*, explains this phenomenon by taking an example from the *Far Cry* series. Within *Far Cry's* world, wild animals roam around with specified inputs that they listen for, such as if players or enemies are nearby, and outputs where they broadcast themselves to the world (Brown, 2018). When an input and output match, a rule is followed, referring to the action taken by the systems involved. In the case of a wild animal and enemy interaction, the rule that follows is likely one where both parties initiate combat. This example gives a simple proposal to generate system interactions, and with enough of these interactions taking place in unexpected circumstances, more emergent gameplay can be produced.

Games that are characterised by their many interacting systems are referred to as systemic games and can result in games with high emergence. The open-world genre contains many examples of systemic games, often including numerous systems with multiple interactions. The somewhat recent Nintendo installation of the *Zelda* franchise, *Breath of the Wild* (2017), is a systemic game with many emergent gameplay opportunities thanks to its many dynamic systems. The electricity system, a subset of the game's elemental system, is built to have many interactions with other systems. These interactions include increasing shock damage and the blast range while it rains and attracting lightning to metal weapons during a thunderstorm. Another system in the game has the Bokoblins, the grunt enemies of the game, run towards weapons on the ground to arm themselves when they are alerted by the player's presence. By knowing these systems, an observant player may decide that during the next thunderstorm, they can place a metal weapon in an enemy's near vicinity, resulting in them picking up the weapon upon becoming alerted and getting a nasty lightning strike soon after.

Harvey Smith, a co-creative director for Arkane Studios, heavily promotes this systemic design in the award-winning game series *Dishonoured* (2012) and *Deus Ex* (2000). Harvey is known to be a big early proponent of immersive sim games and has given conference talks and interviews about systemic level design (2002). In an interview conducted by *NoClip*, Harvey mentions how he wanted the *Dishonoured* series to service multiple playstyles whilst still maintaining a "power fantasy" aspect (2018). Thanks to the game's many systems, a player could adopt a more offensive ability approach to deal with enemies or utilise the game's stealth systems to take out enemies without being noticed. In these games, the threat and tension of enemies are very much present, which makes it even more satisfying when a player outsmarts the enemies through their knowledge of the game's systems.

Emergence has a significant appeal for players and developers alike, making it a popular feature in modern games. For players, it allows more freedom and possibilities through the interactions between systems. Players are given a series of familiar systems, a game's 'building blocks,' so to speak, to build their own solutions to solve problems and scenarios thrown at them by the game's designers. Some solutions players come up with might not even have been considered by the game's designers, which may seem like a downside. However, it often leads to a more rewarding experience for the clever players who experiment with the game's ruleset (Leijnen & Veen, 2016). Sometimes, if the game's designers discover the interaction early on, it can transform into a main mechanic.

Another appeal for emergence is that it can result in more drama and surprise. A player may be able to put their devised solution in motion; however, they might not have accounted for all the other factors in their plan. These unaccounted factors can result in a chaotic chain of events where the player must learn and adapt to their changing situation. In the mission-driven indie roguelike *Streets of Rogue* (2017), a player might hire a gang of people to help assassinate a target, only for them to be put in more danger by encountering an aggressive rival gang on the trek to the target. Emergence allows freedom over how players play the game and creates unique and memorable experiences (Hokkanen et al., 2018).

In Penelope Sweetser's thesis paper, *An Emergent Approach to Game Design – Development and Play*, she describes a prevalent problem in game worlds at the time that still holds relevance in games today (2006). "Within games, enjoyment of the gameplay hinges on the game world. However, game worlds are often static and highly scripted, which leads to restricted and shallow gameplay that can detract from player enjoyment." To enhance the player experience in a game world, it is proposed that emergence in games could enhance player enjoyment in areas where game worlds are weak. Sweetser would then develop the *EmerGEnT system*; a proof of concept to show how emergence can be incorporated into a game's environment, objects, and agents.

Her work would continue in the following years with the release of her publication *Emergence in Games* (2008).

The implementation of emergence is not without its issues. Being a very player-focused way to design a game, attempting to build a game for emergence typically results in extra playtesting and development hours to see if players are utilising emergent strategies as intended. A developer may give players hundreds of unique ways to approach a problem in-game; however, the solution will not be emergent if players only use one or two basic strategies to solve the problem. One of the reasons players might do this is a lack of incentive to use unique strategies. Another reason might be the issue of poor game balancing.

Balancing systems in a systemic game is important so that dominant strategies are not the only options chosen. A systemic game's strength comes from its variety of options to tackle different in-game situations; however, as mentioned by *Civilisation* developer Soren Johnson, "Given the opportunity, players will optimise the fun out of a game" (Johnson, 2011). Because of this, players are likely to be drawn to the most powerful and often quite boring game strategies. This makes balancing all the more important to encourage experimentation, one of the attributes that contribute to emergent gameplay. Developers must spend extra time balancing systems to ensure players are not always selecting the same option.

Emergent strategies also run the risk of operating outside the developer's vision. This form of unintended emergence runs the risk of breaking the immersion of a game and, in some cases, may even break the game entirely. In cases where developers are implementing new interactions and subjects, they must also observe the effects of the new interactions in case previous mechanics break due to their introduction.

Emergence that was once unintended might be a welcome addition to a game if discovered early enough, offering new, unexpected ways to play a game. The project *Plusminus* is an example of this, developed to use simple systems to create complex behaviour (Hokkanen et al., 2018). A player had discovered an unintentional mechanic by launching themselves through the game's magnetism system. This process was set up by layering two objects on top of one another and having the player stand on the top object. The player could then set the bottom object to 'repel', resulting in the top object launching into the air alongside the player standing on top. After discovering this interaction through playtesting, the developers were able to implement the mechanic as a potential solution to some of the scenarios the player could encounter in the game. *Breath of the Wild* has similar unintended system interactions that are present today and are commonly used for speedrunning, a practice where an objective needs to be achieved at the lowest possible time. If a player happens to be drawing their bow while shield surfing whilst also landing on top of an enemy, they unexpectedly get launched super high into the air from an exploit in the game's physics System. This technique can be perfected to traverse the extensive plains of the game more quickly, making it an essential time saver for the speedrunning community.

Creating emergence is not as simple as implementing multiple system interactions, however, as there are many other factors that developers have to take into account. The rules of a system must maintain consistency; otherwise, players will have issues pulling off emergent plans. Suppose a player can set wooden objects on fire but cannot set an enemy wooden watchtower on fire. In that case, the game's believability and the player's willingness to experiment with that system in the future are reduced. Systems also need a reason or catalyst to interact with each other. The open-world Bethesda studio game *The Elder Scrolls V: Skyrim* (2011) contains many different factions, some of which hold hostile rivalries with each other, but does not capitalise on these interactions enough throughout the large world of *Skyrim*. As a result, the modding community developed mods to add more factions and increase the number of random encounters on Skyrim's roads to add more chaos and instability while travelling.

A systemic game also needs to encourage experimentation of its systems so that players can discover the game's interactions. A stealth game might involve several tactics to bypass security through distractions, hacking or messing with the ventilation system. However, if the easiest solution is to kill all the guards, then there is little room to test emergent strategies. Players must be incentivised to try strategies that are not always the same or have restrictions to stop the dominant strategies. Missions or quests in a systemic game that only have one or a few solutions also fall prey to the same issues as players will continue to follow the linear instructions given by the game. Emergence should be captured by offering players different game systems and letting them use their knowledge of those systems to solve problems.

Despite emergent gameplay being a problematic feature to capture in games, developers and academics have studied emergence to understand its nature better. Stefan Leijnen and Fjodor van Veen write a short paper describing the mechanics of a game as "...building blocks that, properly combined, provide new degrees of freedom left for the player to explore" (2016). It provides a unique perspective on emergence where the limitations of the player's experience can result in an expansion of tactics from the player. More information about the nature of emergence was further investigated with the identification of emergent factors in Paananen's thesis *Designing a Game for Emergent Gameplay* (2020).

Identifying ways to apply more emergence is highly relevant, especially in a game heavily relying on PCG elements. Paananen categorises different ways to design for emergent gameplay, which are as follows:

1. Increase interactions - Interactions both apply to player actions as well as non-player subjects and gameplay systems.
2. Reduce constraints - Have multiple ways to achieve goals and avoid limiting player actions.
3. Add more subjects - Including more player actions and systems that interact with the previous systems.
4. Create instability - Utilising randomness for novel scenarios and encouraging the player to experiment. Have subjects act without player input.
5. Add meaning - Connections between game mechanics and real life.

Combining these techniques increases the likelihood of more emergent scenarios appearing significantly.

While a significant identification of highly impactful factors for emergence, more factors still can come into play to extend this framework. Factors such as maintaining game rule consistency and increasing player agency are additional factors that can be used to increase emergence. Further investigations could also be conducted on the impact of certain factors, such as a deeper look into instability and, consequently, PCG.

This thesis uses the existing factors and literature on emergence and expands on that knowledge with additional emergent factors. These factors were constructed into a tool that was then tested by applying it to a custom game product, which was then play-tested and observed to find more emergent gameplay scenarios.

2.4 Procedural Content Generation

Procedural content generation, or PCG for short, is defined as the programmatic generation of content (Raffe, 2014) and has been prevalent in many games in some form or another. While a game without PCG would have artists and designers use tools to build the content within their game, PCG allows developers to expedite the process through algorithms and parameters that become responsible for generating the content. The term content can be used to cover a wide variety of game elements and does not necessarily have a limit on what can be procedurally generated. In the context of roguelikes, the most commonplace randomly generated elements are the maps, enemy placements, and items. However, other roguelikes and other genres play with PCG to create new creatures, stories, behaviours and more.

PCG can prove especially useful for developers needing more resources to manually create vast, detailed worlds or a diverse pool of items and equipment. This versatility has made PCG very popular in the indie scene, but it also has its fair share of use by larger studios looking to create a large variety of content. One of the more common applications for PCG in games is terrain generation. The game *Darwinia* (2005) generates its landscapes through simple fractal methods on a low-resolution height map (Raffe, 2014). Having game elements be procedurally produced allows developers more time to focus on other aspects of their projects, like polish and handcrafted content.

Another significant appeal of PCG for developers and players is the content variety produced through random generation. PCG can create a near-infinite amount of content (Gellel & Sweetser, 2020), resulting in a near-infinite amount of game variations. This extensive content variation can allow players to experience unique playthroughs with their own stories, items, personal goals, and more. It may also result in a higher quantity of emergent scenarios, where the player can tailor their own stories through their gameplay experiences or feel clever for finding solutions to unique problems that a designer did not explicitly plan. The high content variation also helps support a game's replay value and hours of play, so long as each variation of the game feels different enough from the last playthrough. PCG variation is not only limited to gameplay elements but can also be used for visual diversity. *SpeedTree* is a tool that procedurally generates natural-looking trees in games to add extra detail and variation (Gellel & Sweetser, 2020).

With PCG being such a versatile tool for games, it is natural that games use a wide variety of different PCG methods. Some game developers will retrofit existing PCG methods into their products, while others will make custom algorithms from scratch to best suit their needs. *Spelunky* (2008), for example, uses a custom PCG algorithm to generate its many rooms and levels. Each overall level is split into a four-by-four grid of rooms. A random room in the top row is assigned as the entry point room, which determines the player's spawn point. The algorithm then needs to generate what is referred to as the 'critical path', a series of rooms that connect to reach the goal. The critical path process starts by having the entry room pick an adjacent room to the left, right or below. The selected room then picks another adjoining room, excluding the previous one, continuing until the bottom row is reached. The room's chosen direction is randomised; however, it is weighted to favour specific directions to ensure that most levels are not just a pitfall to the exit. All critical path rooms will have open areas connecting to adjacent path rooms so the player can traverse the level without additional tools. If the critical path attempts to go down while at the bottom row, then that room is considered the goal, the room the player must go to complete the level. This results in a snake-like level structure that the player must descend to get to the next area.

The procedural level generation of *Spelunky* does not end there, however, as each room that is not assigned to the critical path still generates a layout with random open and closed-off entrances. There is a possibility that these rooms contain extra loot for the player or a shortcut to the goal if the player possesses bombs and ropes, but because the rooms are not on the path, there is no guarantee as to what may appear in those side rooms. These additional rewards for exploration add more surprise and luck elements to the game. The individual rooms also contain a bit of PCG within them by randomly spawning enemies, treasure, and platforms for more variety and to make it less evident that the player may have seen the same room before.

The way that *Spelunky's* PCG and gameplay are designed is to encourage more emergent gameplay scenarios. Some emergent scenarios arise as a product between the game's PCG system and other gameplay systems, like utilising exploding enemies to create new paths to traverse.

Many other roguelike projects have taken different approaches to their PCG to achieve a particular goal. Melotti and Moraes (2019) use a Novelty Search approach to generate roguelike Dungeons that have quality outputs and fulfil the specified needs of the designer. The quality of generated content can be evaluated in several ways, such as having diverse output variations and reducing malformed content. Malformed content is used to describe content that affects the dynamics of a game in a negative way, such as issuing players challenges that are impossible or are a jump in difficulty, leading to less balance and player engagement overall. The Novelty Search approach is used to create quality generation by encouraging diverse, innovative results that also contain a solution to deal with the malformed content problem efficiently.

Gellel and Sweetser (2020) evaluate several different PCG approaches to roguelikes and state their strengths and weaknesses. These techniques are Markov Chains and Random Markov fields, Cellular Automata, Grammars, Machine Learning, Evolutionary Algorithms and Player Experience Models. After thoroughly evaluating the different methods, a hybrid approach of Context-Free Grammars and Cellular Automata-Inspired Behaviour is chosen. The Context-Free Grammars approach is selected to enforce generation rules, resulting in only valid missions being generated. The structural rules allow for a good level of control, and the process quickly enforces its specified rules. The Cellular Automata approach complements the Grammar system's weaknesses by producing natural-looking spaces without the structure and order drawbacks. Alongside the benefits of building naturally convincing environments, the system is fast-acting and runs at a low cost. By combining the two approaches, Gellel and Sweetser produce a roguelike dungeon generator that capitalises on the selected approaches' benefits while minimising the drawbacks.

Togelius and others, in their paper *Search-Based Procedural Content Generation: A Taxonomy and Survey*, give one of the earliest examples of how PCG can be dissected, in this case, for better classification (2011). In their work, PCG is broken down into factors, such as offline vs. online generation, to better distinguish PCG techniques from each other.

With the rise in popularity of PCG methods in research, there are still PCG options that can be applied to roguelikes that may not have been considered before, including the ability to facilitate the occurrence of more emergent gameplay. Paananen (2020) investigates numerous ways to increase emergence in the context of a PCG game. Included are some excellent foundations for how a game can be designed for more emergent scenarios, with the support of the game’s level generation. Despite this, there is still room to expand upon the foundations of emergent gameplay, and additional research could be conducted on other PCG methods to promote more emergent gameplay scenarios. One example of this is Wave Function Collapse.

Wave Function Collapse, or WFC for short, is a PCG algorithm developed by Maxim Gumin that generates bitmap patterns through an input sample bitmap pattern (Gaisbauer et al. 2019). This algorithm, more specifically, is referred to as the overlapping model of WFC. The algorithm starts by observing an input pattern provided by the game designer, which will be deconstructed and used to generate the output. The size of the input can be scaled to be smaller, larger or the same size as the output and will be broken down into input sections that are a size of N x N. N is a significant parameter of WFC as it determines how big the input sections should be which is particularly relevant for the output.

The output process of WFC is split into three phases: observation, propagation, and backtracking. The observation phase picks an input tile of the size N x N and gives it a single random solution from the remaining possibilities (Parker, 2018). The propagation phase has the choice further propagated throughout the output grid, removing any tile possibilities that don't match the input model. The backtracking phase occurs if the observation and propagation phases end up with an unsolvable contradiction and cannot be continued, reverting to a new observation phase attempt. Due to the nature and importance of the value N, lower values tend to result in a more chaotic, random output, while a higher value will look more similar to the input model.

There also exists a more widely adopted model of WFC with specified rules for its tile placement, as opposed to inferring adjacency rules through an input image. The simple tiled model is a WFC algorithm that uses complex tile rules to restrict the adjacency between tiles with different semantics (Chen et al., 2020), making it a popular choice for developers looking for more control over their game's level generation.

WFC has seen use in projects like *Bug with a Gun* (2018) to generate platform levels and by Gaisbauer and others to build cities (Gaisbauer et al. 2019). In the context of the city builder, a section of the map from *The Bard's Tale* (1985) was adopted as the input for WFC to produce new city outputs. The simplest versions of WFC contain rules that determine which tiles can go next to which; however, more expansions of this model can be considered, such as tile reflections and symmetry (Karth and Smith, 2017). The algorithm has also seen some applications in 3D space, such as in Marian 42's infinite procedurally generated city (2019) or Martin Donald's island generator demonstration (2020).

WFC potentially has applications as the level generator in a roguelike game; however, one considerable issue to overcome would be the malformed content issue. The roguelike designer would need a way to evaluate the quality of the WFC output, then modify the parameters and adjust the algorithm to create more ideal results. Furthermore, WFC does not account for elements that are mandatory for the level, which means that factors like the player's spawn point and goal will likely need to be added after the generation output. An additional pass must be made to ensure the goal is reachable from the spawn point.

2.5 Methodology

Literature was found through a number of different peer-reviewed databases, with the most common one being IEEE Xplore. Other databases include Springer, ACM Digital Library, Charles University Digital Repository, Proquest, AAAI publications and the 1st International Joint Conference of DiGRA and FDG. Other relevant papers were shared through the supervisor and co-supervisor's recommendation.

To find relevant academic sources, the two primary searches carried out were for PCG and emergent gameplay. The first search on PCG contained quite a few papers on the topic, so the literature selected were either papers that focused on a new or novel approach, such as WFC or Novelty Search, or papers with a similar topic, such as classifying characteristics of PCG to achieve a specific goal.

The second search involved looking for papers focusing on emergence and emergent gameplay. Since there is less literature on emergence in games, most modern sources were used upon discovery. Most of the current literature focuses on how emergence can reinvent game rules or look into ways to create more emergence.

The term roguelike was also used early to filter and narrow down the results of PCG sources. It was not uncommon to see papers that covered roguelikes also mention emergence as one of the genre's many benefits. This benefit created a helpful link between the two leading search terms.

Sources were filtered to be less than ten years old to stay up to date with the current point of literature. Emergence is also a topic that has seen more popularity in recent years, which has resulted in most sources being recent and relevant. There were some exceptions to the literature where some older sources were used. *Search-Based Procedural Content Generation: A Taxonomy and Survey* (2011) was chosen for its early dissection of PCG into factors. Rabin's definition of emergence was one of the five definitions of emergence given to participants for the experiment in section 6. The ending part of the definition, "... behaviour that occurs when simple independent rules interact to give rise to behaviour that was not specifically programmed into a system" (Rabin, 2004), provides an interesting take that not all participants had agreed with, making it worthy of discussion.

Ebia and others' paper *Influencing Game Dynamics in A Roguelike Game Through Procedural Content Generation Using Genetic Algorithm* provides some valuable information on roguelikes and procedural content generation, however, the date of the paper is not listed. Despite this, It is assumed to be at least from 2014 or after, as the most recent source used in the paper is from 2014. Sweetser's thesis paper, *An Emergent Approach to Game Design –Development and Play* (2006), is considered a very early source; however, the topic and usage of PCG algorithms make it highly relevant for this thesis.

2.6 Conclusion

In synthesising the literature on roguelikes, emergent gameplay, and procedural content generation (PCG), it becomes evident that these concepts, while distinct in their origins and nuances, have many links with each other. Roguelikes can become a testbed for emergent gameplay, with the support of generated unique scenarios thanks to its procedural environment. Emergent gameplay is shown to be both an appealing feature and a design challenge for developers. Still, its inclusion can be facilitated by correctly using certain factors and enabling player freedom. Procedural content generation is a large field of study encompassing many different algorithms, with room for more investigations on new or novel generators or expanding the use cases for specific algorithms. Ultimately, there is room for each of these topics to be expanded upon in current literature, and the following chapters will use the three topics to gain a further understanding that hopes to expand the current literature.

Chapter 3

Development of The Framework

3.1 Introduction

The first question concerning emergence concerns how developers design for emergent gameplay. How can developers plan for systems that result in emergent behaviour for their players? An investigation was conducted on what was known about emergence from games and academic literature. This investigation will result in several factors being identified and adapted into a framework that can be employed in the following thesis chapters. The established factors were discovered through Paananen's paper "Designing a Game for Emergent Gameplay" (2020), while the additional factors are an extension of the work.

3.2 Factors for Emergence - Established Factors

3.2.1 Increasing Interactions

This is considered the core of emergent gameplay. Emergent plans strongly focus on utilising the interactions between different systems, and it is the variety and surrounding factors of these interactions that create emergent gameplay. The more interactions a game has, the more varied plans players can concoct. Games with a high interaction space are called systemic games. Many systemic games are within the Roguelike, open-world or immersive sim genre, although they are not necessarily limited to those categories.

Interactions are made up of a series of inputs and outputs. These may also be referred to as awareness and rules, respectively. Subjects such as the player, non-playable artificial intelligence (AI), weapons, items and environmental set pieces may be aware of other factors. A rule must be followed when that other factor is brought to them. An AI might be mindful of another AI with a fight ensuing based on their interaction rules, while the same AI could also be mindful of elements like fire and flee from the scene based on another interaction rule. Awareness can also listen out for external factors like weather or day-night cycles, and more interactions exist other than fighting and fleeing, such as environment manipulation.

Maintaining and adding many interactions can be challenging, so some developers have devised an elegant way to deal with the issue. Rather than having subjects react specifically to an enemy, item or environmental hazard, they respond to the stimuli given off by the other subject. The stimuli approach allows developers to implement subjects more easily with interactions rather than hard-coding new interactions. The shared stimuli also create more consistent rules for interactions, which has been shown to help generate more emergence.

3.2.2 Reducing Constraints

This marks the most significant difference in design between emergent and linear games. While a linear game will restrict the player from finding an intended solution, emergent games will have minimal restrictions on how players execute their plans, with even the chance of unintended solutions occurring. A constraint might be a failure state for a mission when specific steps are not adhered to or when player tools are taken away before a particular gameplay segment. When it comes to emergent design, a good philosophy is to give the player a goal and not worry about how it is achieved.

Removing constraints is the contrary in Stefan Leijnen and Fjodor van Veen’s paper *ArkaNet: Investigating Emergent Gameplay and Emergence* (2016), where they argue that a game's limitations can allow the expansion of tactics and other emergent gameplay possibilities. There is some truth to this. Suppose the player is given a multitude of options. In that case, there is a good chance that they will find an optimal strategy that works for them and use it in any scenario, resulting in fewer emergent possibilities. The overuse of optimal strategies makes the additional factor encouraging experimentation all the more important, as a mix of fewer constraints and more experimentation will arguably lead to more emergence than more constraints.

3.2.3 Adding More Subjects

The utilisation of interactions can only occur if there are subjects in the level that can interact with each other. Having a good number of subjects with exciting interactions in the scene allows for various options that the player can use. More subjects are generally better but require more balancing; otherwise, it may risk overwhelming newer players. Consistency and meaning should also be considered when introducing more subjects.

Developers should also consider player plans that change subjects' behaviour or introduce new subjects. Some examples of this could be hacking a terminal to turn a security system against the owners or opening the door to allow more subjects into the scene, causing more chaos. By enabling the player to use strategies that do not just remove subjects, the possibility of future emergence occurring in the area increases instead of decreasing.

3.2.4 Creating Instability

Instability allows for the most variety between player emergent storytelling, with its influence coming in many forms. It can come from the ability of subjects to act without the player's influence, giving them their own motives and actions to follow, which can result in chaotic and unusual scenarios. The player might not have agency over the initial actions, but they can utilise these behaviours to benefit their plans. Alternatively, instability could backfire on the player's plan, adding another layer of conflict for the player to deal with.

Instability can also come in the form of randomness, such as randomness in items, events, the actions of an AI and, most commonly, procedurally generated content. This technique is frequently used for roguelike and open-world sandbox games to create a new experience for each player and playthrough, leading to many unique scenarios and stories.

3.2.5 Adding Meaning

Meaning in this context is to add a correlation between the mechanics of a game and something that is commonly known. A typical example is fire, where commonly known flammable objects in real life, such as oil and wood, can also be set alight in the game. If this rule is to be broken, it risks breaking the consistency principle, which can reduce the player's desire to use emergent strategies.

Meaning does not necessarily need to be realistic as it can also be based on common game knowledge or troupes. Red barrels being explosive or green and purple goo being poisonous are some game elements players can infer for themselves and later use their respective interactions for their emergent plans. This common knowledge gives players a better understanding of a system, which goes hand in hand with the two factors being encouraging experimentation and maintaining consistency.

3.3 Factors for Emergence - Additional Factors

The additional factors are an extension of the current factors that have been established in Paananen's paper. These factors have been observed in several current emergent game titles and have been compiled in Game Maker's toolkit video *The Rise of the Systemic Game* (2018).

3.3.1 Keeping Rules Consistent

A common misstep for systemic games is when a rule is not consistently followed or when one interaction does not carry over to other similar subjects. For example, if a game allows the player to set fire to some wooden objects but not others, this results in a break in consistency. While consistency is a good principle in any game, it is essential to encouraging emergent experimentation. If a player sees their plan fail due to an inconsistent interaction within the game's world, they are less likely to use that emergent strategy in the future. On the other hand, players can make emergent strategies thanks to a game's consistent ruleset. By discovering a specific interaction in one area and employing that same interaction in another location, players can feel rewarded for using their ingenuity to solve problems.

Developer and games journalist Tom Francis discusses how consistency can be introduced into a game without the need to explicitly code every scenario (2018). Using any *Deus Ex* game as an example, the player will learn about three interactions: They can move crates, crates block vision, and turrets only attack targets they can see. With a large enough crate placed in front of a turret, the player can use this emergent strategy to bypass turrets with ease. Nothing in the code explicitly states 'If the player moves this crate to location x, don't let the turret fire'. Instead, the developer writes a rule that objects can block vision, and ensure those rules are consistent across to other objects so that players are incentivised to come up with strategies using the game's ruleset.

3.3.2 Offering More Player Agency

Agency is a relatively simple concept in a game, ensuring the player has plenty of actions to play the game and react to situations the way they want to. If a player is limited to a small set of tools, they will likely use all those tools to solve the situation, suitable for genres like puzzle games that guide the player through its limitations. On the other hand, systemic games offer players a large set of tools, not expecting them to use all of them at once but to use what they find most suitable for their situation, allowing for more freedom in their actions.

In the GDC talk *Breaking Conventions with The Legend of Zelda: Breath of the Wild* (2017), technical director Takuhiro Dohta describes how multiple engines were developed for Breath of the Wild. Going into detail about the mechanics of the custom chemistry engine, elements can change the state of materials (e.g. fire burning wood), and the state of other elements (e.g. fire disappearing on contact with water). Natural occurrences of the chemistry engine in action commonly serve as a great "show don't tell" tutorial to players about the elements, such as rain putting out fires and torches. However, the knowledge of the elements is of little use unless the player has the tools to utilise these elements, which thankfully the game has a plethora of. An explicit tool that uses the game's chemistry engine is the elemental wands and arrows, such as using a fire arrow to explode a red barrel. Another tool the player gets access to is the ability to move metallic objects with ease, which allows players to drop objects to deal damage (using the physics engine) or attract lightning in a thunderstorm to a particular point (using the chemistry engine). Regardless of what approach the player takes, the important part is that the player is offered these tools so that more emergent solutions can be found, instead of being fixed to fewer actions with fewer interactions.

3.3.3 Encouraging Experimentation

A systemic game cannot be fully realised without encouraging player experimentation. This concept is a little more abstract and challenging to achieve, resulting in various proposed approaches.

The more familiar a player is with their toolkit, the more likely they will use it. Suppose a player finds that an action is too niche, difficult to use, or the interaction effects are too inconclusive. In that case, they are discouraged from using that set of interactions. To encourage experimentation, developers need to teach their players the applications of their tools so that they can use them to exploit those system interactions as part of their plans.

Furthermore, developers must be careful of creating dominant strategies as this encourages players to play the game in a restricted way to be optimal. To work around this, genres such as Roguelikes will have more powerful items show up less frequently than other items on their run. Rarity is not a perfect solution, as many players can opt to reset their runs until they receive that powerful rare item before continuing to play.

Some games encourage players to use different strategies by providing explicit benefits when they switch their tools, like damage buffs or extra experience points. Game designer Clint Hocking, known for his work on the early *Tom Clancy's Splinter Cell* titles (2002-2005), *Far Cry 2* (2008), and *Watch Dogs: Legion* (2020) talks about the effectiveness of this approach in his GDC talk *Designing to Promote Intentional Play* (2006). Clint mentions how GTA titles offer rewards for specific challenges such as "...large cash rewards for completing rampages that require the player to use all the different weapons in the game, or for completing unique stunts that reward him for elite driving maneuvers" (2006). Other games take the discouragement approach, where overusing items makes them less effective. *The Legend of Zelda: Breath of the Wild* (2017) controversially gives its weapons very low durability, encouraging players to use other tools and newly acquired weapons.

Encouraging experimentation is not an easy attribute for developers to capture. However, many of the stated factors can support experimentation. For example, pairing consistency and meaning can encourage creative players to discover new strategies, increasing the possibility of emergence.

3.4 PCG for Emergence

3.4.1 PCG Games Following Emergent Factors

While emergent gameplay is a highly sought-after feature by developers, its factors are not suited to every game genre. More linear games like puzzle or narrative-focused games require many restrictions to guide players to the solution. However, genres like open-world games and roguelikes can make great use of all the factors to create a highly systemic game, such as *Minecraft* (2011), *Terraria* (2011), *Spelunky* (2008) and *Noita* (2019), to name a few examples.

PCG algorithms can even be evaluated under the Emergent Framework to determine if they suit a systemic game environment. The instability of a PCG algorithm can provide much variety for the player through a game's generated levels or randomly generated items. PCG could be responsible for adding subjects to a level, resulting in more emergent possibilities if the subjects are distributed appropriately. Finally, the consistency of a generated level can also increase the player's knowledge of the game, giving them more opportunities to use emergent strategies at the cost of less level variety through instability.

3.4.2 Tile-Based Approaches

Many Roguelikes known for their emergence use a tile-based PCG approach for generation. Tile-based approaches are 2D PCG methods that are given a set of shapes (often squares) to generate the world, dungeon or other location that the player is to navigate through. There are many variations this approach encompasses, such as *Spelunky's* (2011) custom generation, *Noita's* (2019) Wang Tiler and *Dwarf Fortress'* (2006) tiles and fractals system. There are some advantages when using this approach compared to other PCG methods. One is that the resulting world or dungeon is expected (i.e. less subject to malformed content) yet still varied enough for each playthrough. Tile-based approaches can also ensure that certain elements are always present, such as chests or a pathway that always leads to an exit, and often have a ‘second pass’ to add extra details to keep the level interesting.

3.4.3 Emergence through Storytelling

Emergent games are known for allowing players to play how they want, which often results in experiences that are unique to them. The actions they take and the reactions that follow result in player-driven stories, which can be made unique to the player if the game uses PCG. Some examples may include the lifecycle of a *Spore* (2008) creature or finding an escape route of a generated world/dungeon when fleeing from an enemy. Some programs, like *AI Dungeon* (2019) or *Wildermyth* (2021), use PCG to produce unique stories for the player to experience.

3.4.4 Creating Instability

Also mentioned as a factor of emergence, a subset of instability is randomness, which can be achieved through PCG. There is some uncertainty about what can be generated, often leading to unique stories being produced. The game could also generate subjects with the ability to interact with each other without the player’s involvement. Examples of this include AI with custom objectives, faction fights and dynamic environments.

PCG can also be applied to the interaction space, generating unique interactions never seen before; however, this is not without issues. Most notably, randomly generated interactions may go against other emergent factors like consistency, resulting in less emergence overall. There is also the possibility of creating procedural AI to create novel scenarios, which could be an exciting challenge to investigate. Still, it has the potential to break consistency if the AI becomes too random for the player to predict.

3.5 Conclusion

To develop the foundations of the framework, factors to promote emergence needed to be identified. The first factors to be identified were the emergent gameplay attributes that have been covered and proven to increase emergence in existing academic literature. These attributes include increasing the number of interactions, adding meaning to those interactions and adding more subjects to give a few examples. The list was then expanded to include additional factors like rule consistency, more player agency and the encouragement for experimentation. Finally, to strengthen the link between PCG and emergence, a couple of methods were listed on how PCG can be used to create emergence.

Chapter 4

Applying the Framework

4.1 Introduction

The Emergent Framework was designed as an evaluation tool to better understand the correlation between emergent factors and emergent gameplay. In Chapter 6, the framework will be used on a game prototype to help identify the effectiveness or lack of emergent factors. To create a comparison for the prototype against existing emergent games, five emergent games were chosen and evaluated under the framework: *Spelunky* (2008), *Noita* (2019), *Legend of Zelda: Breath of the Wild* (2017), *Minecraft* (2011) and *Don't Starve* (2013).

The PCG for Emergence Framework was created to measure the consistency and instability of procedural algorithms and evaluate their ability to add subjects within a game's level or world. The framework would be used to evaluate different algorithms against each other to support the process of choosing the most suitable generative method or combination of generative methods.

4.2 The Emergent Framework

Each emergent factor is rated on a scale of zero to five (see Table 4.2). Zero represents an absence of the factor, while five represents frequent usage. Full rating explanations from one to five on the emergent scale rubric are provided in Appendix C. As an example, L*egend of Zelda: Breath of the Wild*, or BOTW for short, has a five rating for interactions as they are very commonplace within the game, but a one rating for instability as it infrequently uses randomised elements. It is important to note that these ratings are used both as an example of how games can be charted on the framework and to create a reference point for the developed prototype of Chapter 5. These ratings, while given justifications, are empirical data and should not be used outside this context.

A table and radar chart represent the framework for emergent factors (see Figure 4.2). The table is used as a more traditional representation of data to clarify the correlation between the games and the emergent factors. The radar chart is used as a comparison tool to show the similarities and differences between each game's abilities to create emergence.

4.2.1 Spelunky

A 2D platformer roguelike game set with a cave and temple exploration theme. It is considered the first game to mix platformers and roguelikes. Games under the platforming genre were thought to be easy to pick up and play, although they could get repetitive due to static levels. Roguelike's random levels resulted in a highly replayable experience but were challenging to learn at the time due to their cryptic systems. *Spelunky* cleverly used each genre's advantages to outweigh the other genre's disadvantages, resulting in a highly successful indie title.

There is a good variety of items and enemies within the game, but not an overwhelming amount compared to other roguelikes like *The Binding of Isaac: Rebirth* (2014). Instead, the game focuses on the interactions of its elements and the procedural generation of its levels to create emergent moments, of which it is considered one of the best.

4.2.2 Noita

A premise similar to *Spelunky*, *Noita* is a 2D platformer Roguelike game set with the theme of cave exploration but with emphasis on different parts of the game. In *Noita*, you play as a wizard who can use wands with randomly generated properties. There is less focus on precision platforming and instead more emphasis on the interactions of the game, of which there are many, some of which the game designers may not even have seen.

The higher focus on emergent factors like interactions and the reduction of constraints make it arguably a more emergent game than *Spelunky;* however, this does come at the cost of being more challenging to pick up and play, making the rules of the game a little more difficult to understand for the first couple playthroughs. Once the player can get a good grasp of the game's systems, the possibility for emergence skyrockets, making it one of the most emergent games today.

4.2.3 The Legend of Zelda: Breath of the Wild

Breath of the Wild, or *BOTW* for short, is an action-adventure game set in the large open world of Hyrule. It has the appeal of many different genres, from the exploration and collectathons of the open world to the puzzles from the shrines and divine beasts, as well as the combat scenarios of its bosses and encounters throughout the sprawling world.

BOTW is considered a very emergent game thanks to the many different interactions that exist within the game's world. From its chemistry engine of different elements to the rolling weather system and the way NPCs and mobs interact with these elements (as well as each other), something interesting can always happen within the game's world. Furthermore, the game's puzzle shrines do a fantastic job of tutorialising the unique ways players can utilise the game mechanics while also sprinkling the world of Hyrule with plenty of enemy camps and set pieces for them to try out emergent strategies.

4.2.4 Minecraft

Minecraft's open world is created through a series of random generative methods, allowing players to explore new worlds that are unique to them. The overwhelming success of the game has allowed it to grow and continue updating with more and more subjects that have their own interactions with the world, creating new emergent scenarios.

With Minecraft being as popular as it is, players commonly find ways to use systems in unexpected ways that the developers might not have considered. For example, a player could place doors underwater to create a pocket of air, which may not make sense compared to the real world but has been kept as a fun, relatively harmless interaction for players to exploit.

4.2.5 Don't Starve

Don't Starve is a procedurally generated open-world game that focuses on surviving the elements like its everchanging seasons or the powerful emerging threats of its Burton-esque world. Like *Minecraft*, its emergence comes from its procedural elements and system interactions, most of which involve the game's AI creatures. Many of the creatures within *Don't Starve* not only have interactions with the player but also interactions with the other creatures and elements within the world. To survive and thrive in the world of *Don't Starve*, optimal players have found ways to use the enemy AI for their benefit, such as using destructive enemy attacks to help gather resources or utilising one powerful threat to take care of another.

Table 4.2: Classifying games under the Emergent Framework

Emergent Factors	Spelunky	Noita	BOTW	Minecraft	Don't Starve
Interactions	4	5	5	5	4
Reducing Constraints	4	4	4	5	3
Adding Subjects	4	5	4	5	5
Instability	4	5	1	5	4
Meaning	5	4	5	4	4
Consistency	5	4	4	4	5
Experimentation	4	4	4	4	4
Agency	4	5	5	5	4

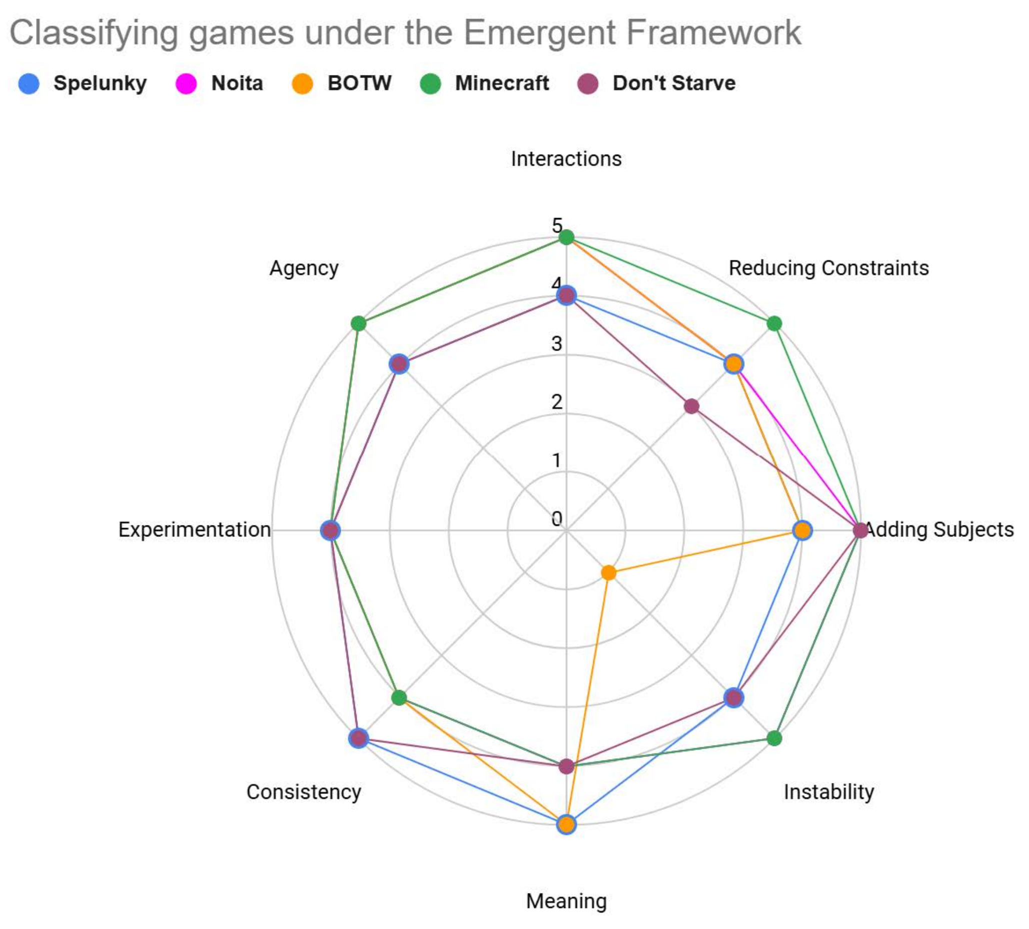


Figure 4.2: Chart representing the games under the Emergent Framework

4.3 PCG for Emergence Framework

PCG has previously been dissected into factors in academic literature, such as in *Search-Based Procedural Content Generation: A Taxonomy and Survey* (Togelius et al., 2011). However, classifying PCG techniques for emergence is more challenging to identify when compared to evaluating the whole game experience for emergence.

Section 3.4.1 (PCG games following emergent factors) and 3.4.3 (emergence through generated storytelling) are ways PCG can be used to create emergence. That said, these elements are not suited to be put onto a framework to evaluate PCG algorithms. Additionally, the emergent factors of increasing interactions, reducing constraints, adding meaning, encouraging experimentation and player agency are challenging to include in this framework. They are more applicable to an emergent game's design as a way to promote more emergence from the player. Some attributes may be slightly relevant (e.g. Adding meaning to the environment of PCG), but their effect plays a significantly smaller role than the traditional principles normally would. Three possible factors from the Emergent Framework apply to PCG techniques: instability, consistency and adding subjects. In this framework, instability describes the variation and randomness of generated content, while consistency refers to the familiarity of the generated content. Because of this, the two factors are at odds with one another for this framework, creating an interesting balancing act. Too much instability results in a reduced space for player plans due to a lack of game knowledge, while too much consistency results in a lack of diverse strategies.

Adding subjects is another potential factor for emergence through PCG. PCG algorithms may be responsible for adding subjects to a game level, although there are quite a few cases where it might not be applicable. In cases where it is practical, it can result in more subject variety and a higher possibility space, opening up the way for more emergent player plans. Typically, the more control the developer has over the PCG algorithm, the easier it is to add and distribute more subjects.

Eight PCG methods have been evaluated under this framework, with a mix of tile-based methods, general PCG methods and the constraint solver. These eight factors were discovered from their inclusion in roguelike discourse or frequent usage in PCG talks. Herbert Wolverson discusses how Random Walk, Binary Space Partitioning and Cellular Automata can be used in roguelikes at the streamed Roguelike Celebration event (2020). Kate Compton details situations where Grammars and constraint solvers can be used in her GDC talk *Practical Procedural Generation for Everyone* (2017). The two *Spelunky* (2008) generative methods were selected for analysis as the game is an early market example of how roguelikes can be emergent. Finally, Wave function collapse was selected as it is relatively new in the academic space and is still finding more use cases where it can be applied to games and other projects. These methods were chosen for the scope of a 2D platforming roguelike, which is discussed in more detail in Chapter 5. Many possible PCG methods could be chosen for analysis depending on the objective, such as looking into more 3D-centric generative methods for developing an open-world game.

The scale for this framework is rated on a spectrum from one to seven, with the lower end of numbers representing algorithms with more instability and the higher end representing the opposite, more consistency. For example, constraint solvers score seven on the spectrum as their purpose requires a highly consistent nature (see Table 4.3). A bar chart has been used to show the score range for each PCG method to assist with the visualisation of the instability vs. consistency score (see Figure 4.3).

It is also possible for algorithms to have multiple scores if their parameters allow them to be more consistent or unstable. The random walk algorithm, for example, could be rated as a four, five or six as the input parameters can significantly alter the consistency and instability of the output.

Adding subjects can also be involved with some PCG algorithms, potentially creating emergent gameplay. It is factored as one of four classifications: low, medium, high or non-applicable. The non-applicable option is included as not all PCG methods included are responsible for adding subjects to the game.

4.3.1 Spelunky Map Generation Method

The *Spelunky* map generation method is responsible for placing rooms in a particular order, with the result becoming the level the player has to navigate through. The algorithm is straightforward, starting from the top of the level and snaking its way down to the bottom to create the level goal. Increasing the interaction space by adding subjects plays a minimal role in this method, with the only applicable space being the subjects of one room synergising with other subjects in adjacently generated rooms. The variation or instability of this method is low as it is often less noticed by the typical player. A more adept player could use the consistency of the level layouts to complete specific objectives faster or more effectively (i.e. better learn where treasure or exit locations are).

4.3.2 Spelunky Room Generation Method

The *Spelunky* room generative method assists the map generation method as it can add details to rooms for more variation. *Spelunky's* rooms consist of multiple parts, the overall room layout and the designated spaces to generate set pieces inside. Set pieces typically contain environmental blocks but may include enemies or interactable items. There are many different room layouts and set pieces, which can result in many room varieties. That said, those who play the game quite often may recognise some of the reused level pieces. The developer does design these elements, so there is only a slight amount of instability. Regardless, emergence can still be created through this generative method as many subjects can be introduced through set pieces to increase the interaction space.

4.3.3 Random Walk

Random walk is a versatile and straightforward algorithm that generates varied room shapes and corridors to other rooms. The algorithm itself would not be responsible for adding subjects, with the subjects needing to be added later through an alternative algorithm. Despite the room variation, a lot of parameters will need to be tweaked from room to room for more instability, resulting in a high level of consistency.

4.3.4 Binary Space Partitioning

Binary space partitioning is a division-based dungeon generation method that ensures that placed rooms are an appropriate distance apart. To achieve this, the generative method splits a designated area somewhere around the halfway point, resulting in two sub-sections being produced. The process repeats for each section an appropriate amount of times until the number of sections matches the desired number of rooms to be placed. The rooms are placed in each section and connected through corridors to create the dungeon's layout. Similar to the random walk, the consistency of the level layout will be high with slight instability. Subjects would also be added through an alternative algorithm.

4.3.5 Cellular Automata

A heavily rule-based procedural method that creates interesting results akin to a cave-like or islandlike structure. Each individual tile has a set of rules that determines if a new tile is to be generated, destroyed or stay the same. These rules are based on whether the surrounding tiles are alive or dead (present or not present), and the rules can be modified through the developer's control.

Adding subjects can be performed through Cellular Automata, although using custom rules to place subjects might be more favoured depending on the level of control the developer wants. The level layout is quite varied but still has a bit of predictability thanks to the inputted rules.

4.3.6 Wave Function Collapse

Wave Function Collapse is a novel and relatively recent method of generating tile maps, which can even be expanded to generate 3D spaces. There are currently two main implementations of the Wave Function Collapse algorithm: the tiled and overlap models. The tiled model requires each tile to have specified adjacency constraints, with tiles continually placed and their rules propagated throughout a grid. The overlap model functions similarly but defines its adjacency rules through an input image, which it breaks down into pattern chunks for tiles (Parker, 2018).

The output can change quite a bit depending on the set rules, ranging from forests, islands and even cities, as seen in Gaisbauer's paper, *Procedural Generation of Video Game Cities for Specific Video Game Genres Using WaveFunctionCollapse* (2019). The algorithm is based on a set of specified adjacency rules, the observation phase, which is then applied throughout the remaining tile space, the propagation phase. Backtracking is also implemented in some algorithms to assist the generative process.

The method can be challenging to design for adding subjects due to the nature of its generation. However, with a better understanding of the generative method and rules, it is certainly possible. Developers can encourage the algorithm to generate subjects in certain positions based on the adjacency rules they specify for the algorithm. As for instability and consistency, these factors can range from high to low due in part to the rules set by the developer. More rules and less tile entropy will result in more consistent levels, whereas fewer rules and higher tile entropy will result in more instability.

4.3.7 Grammars

Grammars are a more computer science-focused way of generating content, where complex items are comprised of smaller items, which may be comprised of more minor items. Because of this, through good design rules and constraints, a bunch of subjects can be added through this algorithm. Grammar generation can result in a high variation of generated content if given many different divisions of items to choose from. Consistency in level design can be challenging but can eventually be figured out with enough experimentation and understanding of the algorithm.

4.3.8. Constraint Solvers

Constraint solvers are a valuable tool for many different PCG methods to enforce difficult-to-capture constraints with a lot of flexibility. While not an applicable generative method by itself, it can be beneficial for adding more subjects when paired with less stable algorithms. Instability is not typically achieved through the solver itself, and the rules of the solver can be figured out quite quickly by astute players.

Table 4.3 PCG for Emergence Framework. A low score represents high instability, while a high score represents high consistency.

PCG Methods	**Instability vs. Consistency Score**	**Adding Subjects**
Spelunky Map Generative Method	6	Low
Spelunky Room Generative Method	5-6	High
Random Walk	4-6	N/A
Binary Space Partitioning	6	N/A
Cellular Automata	3-6	Medium
Wave Function Collapse	2-6	Medium
Grammars	2-6	High
Constraint Solvers	7	High

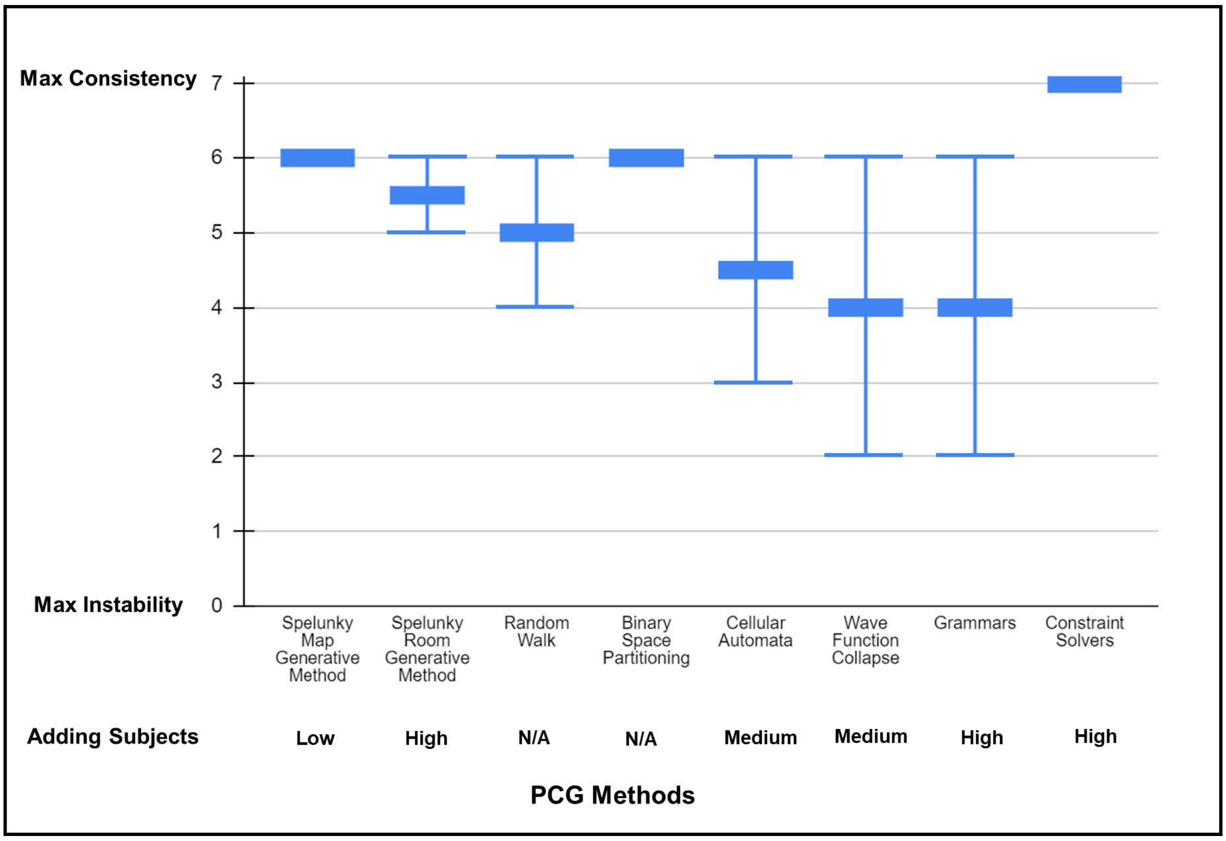


Figure 4.3 Chart representing the PCG for Emergence Framework.

4.3.9 Framework Effectiveness

While some generative methods are relatively straightforward regarding measuring instability, consistency and their ability to add subjects, it is clear that some rule-based approaches are highly variable and can range from being highly consistent to highly unstable. There is not necessarily a perfect PCG solution for creating emergent gameplay; however, with enough rule modifications to the algorithms, a near-ideal balance between instability and consistency can be discovered. Furthermore, constraint solvers can also be used to find that balance and can be responsible for adding more subjects for a higher emergent possibility space.

4.4 Conclusion

The emergent factors established in Chapter 3 have been adapted into the Emergent Framework, which can be used to evaluate a game's ability to facilitate an emergent environment. The framework was then applied to five released games that successfully captured emergent gameplay and compared them to provide an example of how the framework can be used. The framework was also applied to the prototype in Chapter 6 section 6.4, where it evaluated the presence of the emergent factors.

Eight generative methods were also tested in the PCG for Emergence Framework, evaluating them for their instability, consistency and ability to add subjects. The framework's purpose is also to assist in developing an emergent game. More specifically, it will be used to find a generative method or methods to facilitate an emergent environment. The PCG for emergence framework was used in Chapter 5 section 5.2, to find a suitable combination of PCG methods for the roguelike prototype.

Chapter 5

Developing a Prototype for Emergence

5.1 Introduction

The Emergent Framework has previously been used to evaluate emergent market games. The framework aims to become a valuable tool for developers by providing them with a list of factors they can keep track of during their development process. These factors can be considered when developing new systems, enemies, interactions, player actions, etc.

The PCG for Emergence Framework can also be used to find suitable PCG methods for an emergent game. This chapter uses the framework to find a combination of generative techniques that work well with each other, striking the perfect balance between instability and consistency.

5.2 Features vs. System Requirements

The prototype utilises the theory behind the Emergent Frameworks to create a game emphasising emergence. Including emergent factors from the framework was a priority when developing the framework, considering each factor while each game system was being developed. The prototype strongly relies on bow-wielding combat for several reasons that can lead to emergence. Bows are a common concept in games, with this prototype having similar controls and functionality to other bows in 2D games, increasing the likelihood of player familiarity (add meaning). Bows were chosen over other ranged systems like guns so that the game's physics system could impact projectiles more (more interactions and meaning). Bows and arrows will have randomly generated attributes (instability) depending on what the player has equipped so that the player has more options to solve in-game challenges their way (player agency).

Another essential factor to consider is the level of design. The levels will use 2D tile-based generation with additional randomly generated elements (instability) through many different items, enemies and interactable elements spawning throughout the level (adding subjects). Section 3.4.2 goes into more detail about the benefits of using a tile-based approach. The *Classifying PCG for emergence under the framework* (see Figure 4.2) was used to pick more specific, suitable generative methods to fit the needs of the prototype alongside other justifications.

The level generation will use a mix of generative approaches, one to add more variation and instability and the other to add more consistency and reduce malformed content. Cellular automata was one of the first generative approaches considered for instability; however, this procedural method was already effectively used for emergence in Sweetser's thesis, *An Emergent Approach to Game Design – Development and Play* (2006). Grammars may also be a suitable candidate for instability. Still, their generation requires several additional design considerations on top of being a topic with much coverage in current literature. Wave function collapse, or WFC, is relatively new in the academic space and is still finding more use cases where it can be applied to games and other projects. For this reason, WFC will be used to generate the levels.

WFC will be responsible for generating the floors, ceilings and traps of the level to add a bit of extra variety compared to a linear level. More specifically, the overlap model of WFC will be used to generate the levels. The tiled model of WFC is still a viable option for level generation and is more commonly adopted due to its developer control through specified tile rules. However, with instability being more of a focus for the project and the overlap model being great for rapid prototyping with its input sample, it will be the method of choice. More information on the functionality of WFC functions can be found in Chapter 4, section 4.2.6.

Figure 5.2.1 provides the input examples for the game's first three-level variations. The key to the tiles are as follows:

- Brown square tiles represent standard cave tiles
- Flat rectangular tiles represent platforms
- Grey triangle tiles represent spikes

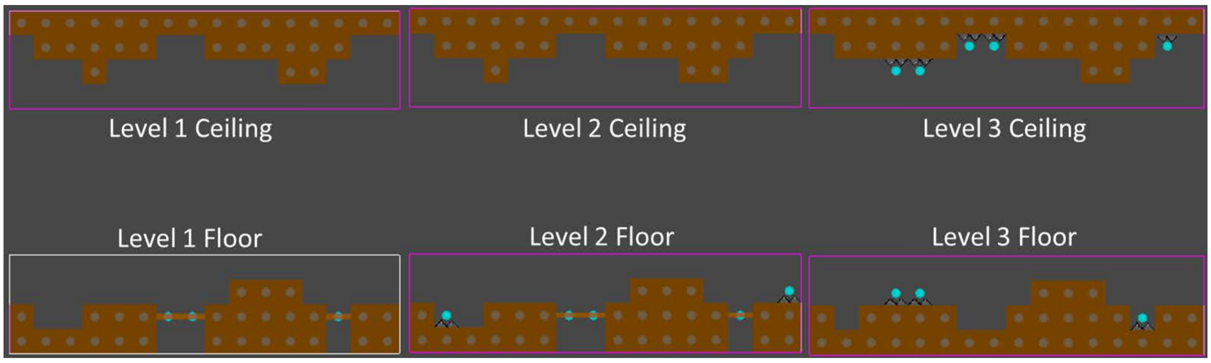


Figure 5.2.1 The input samples for Wave Function Collapse

With each level, subtle changes will be made to increase the difficulty of the levels. For example, level two will introduce more spikes within the level, while level three will remove the support given from platforms. After some internal testing, an *N* value of two was chosen for the overlap model due to higher *N* values resulting in very similar levels with slight variation. A larger *N* value could be appropriate if the input sample and levels were made on a larger scale. Figure 5.2.2 shows how the input samples are applied to the level generation.

The level will start with a big hollowed-out square surrounded by the default cave tiles. Each level will have four layers, each consisting of a floor and ceiling section. The floor section is represented with the orange boxes in Figure 5.2.2. The ceiling sections are represented with the cyan boxes. The boxes are the output samples of WFC and use the input from Figure 5.2.1 to generate a similar pattern in the output section. Each output box has a width of 40 tiles to suit the width of each level and a depth of 4 to match the input size and get similar results. For the sections that overlap, the generation of the floor layer takes priority in the overlapping area.

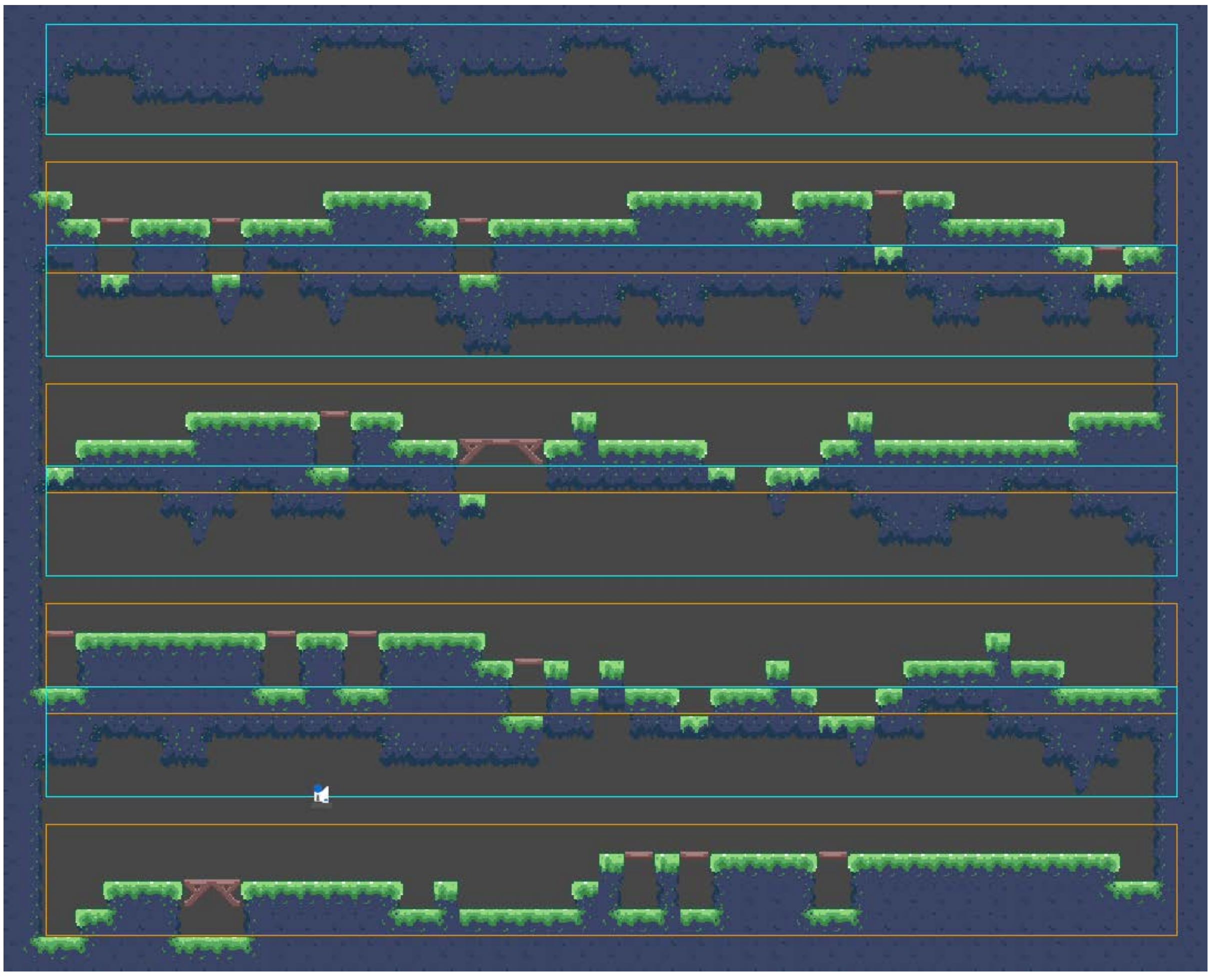


Figure 5.2.2 The output samples for level 1, generated by Wave Function Collapse

There will be an entry point and exit that the player will need to reach, with a critical path always being in place to reduce malformed content and impossible levels (consistency). However, the player should have the option to stray from the critical path to reach the exit if they have the required game knowledge and the equipment to do so (reducing constraints).

To generate the critical path, the Spelunky Map Generative Method will be used. The method generates a random path that carves holes in the level's floors and ceilings, allowing the player to drop through sections of the level to reach the goal. Figure 5.2.3 demonstrates the level carving process. More information on this generative method can be found in Chapter 4, section 4.2.1.

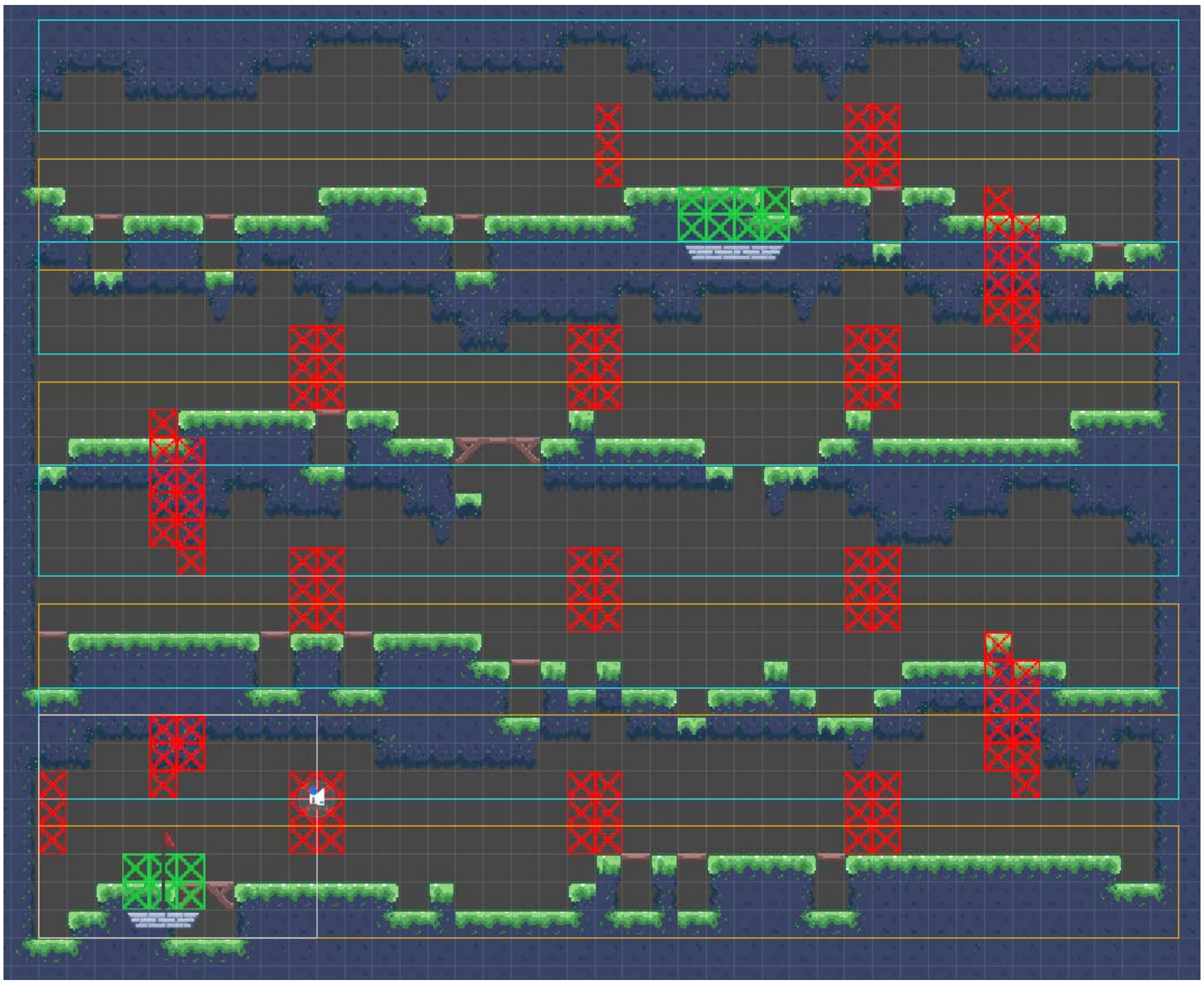


Figure 5.2.3. The Spelunky Map Generative Method, shown to generate a path from the start to the goal.

The generation and items were tweaked and balanced with a smaller group of testers before the participants could engage with the prototype. Having a smaller group of testers will ensure that dominant strategies won't overtake the game's meta (encourage experimentation) and provides an opportunity to patch any game-breaking bugs discovered. The placement of the items and enemies was reasonably straightforward, spawning both subjects in the open-air spaces between the cave floors and ceiling on each level.

Interactions between each item, enemy and piece of level generation will require implementation (increasing interactions), with those adhering to a consistent ruleset (maintaining consistency) that the player can learn and adapt. Appendix A.1 and A.2 are design maps that show the implemented items and interactions for the game. These design maps represent the core idea behind the prototype.

In addition to completing the level, players will be able to complete various challenges within each level, known as Totem Challenges. These challenges will be chosen randomly from a pool of challenges, including taking no damage, escaping the level in a certain amount of time, reaching a difficult spot and more. These challenges aim to encourage the player to try different systems and items (encourage experimentation) to succeed in these challenges for a rare reward.

Further mapping of the game's features to emergent factors was completed in the Project Quiver Matricies (see Appendix A.3.0 - A.3.5). The matricies combined are an exhaustive list of features that explains how each feature considers an emergent factor or multiple factors. These factors are also given a simple rating as to how much they increase the emergent factor. A rating of one corresponds to increasing that emergent factor, while a rating of 0.5 similarly increases the factor, but to a lesser degree. A blank space means that the feature does not affect that particular factor. To provide an example, Appendix A.3.2 notes that enemies are given the ability to knock back for the following emergent reasons:

- As a new interaction (increasing interactions) - Rating of 1 as it increases the emergent factor
- To maintain consistency as other elements of the game cause knockback (keeping rules consistent) - A rating of 1 as it increases the emergent factor
- Players can use the knockback of an enemy to their advantage (player agency & encourage experimentation) - Rating of 0.5 as it slightly increases the emergent factor

5.3 Using the Prototype - Project Quiver

Project Quiver is a 2D roguelike platformer game where the player controls an inquisitive archer who enters the depths of cave Quiver to learn its secrets. The player can strive for two explicit objectives of the game. One is to reach the furthest point of the caverns possible without dying. The other is to obtain the highest score, with points awarded for various tasks such as defeating enemies, completing challenges and reaching the goal of each level.

Project Quiver is available for download on the game-sharing site Itch through the link: https://devsledge.itch.io/project-quiver. The prototype is currently only available for Windows download. After downloading the project, the file must be extracted, and "ProjectQuiver.exe" will be run to play the prototype.

Project Quiver opens with a menu screen (see Figure 5.3.1), allowing the player to jump straight into the game, play the tutorial, adjust the audio settings or quit the application. The tutorial is highly recommended for new players. The tutorial will teach the fundamentals of the game, which has been broken down in the following Tables (see Table 5.3.1 - 5.3.3):

Table 5.3.1: Movement and Camera Controls

[A]	Move left
[D]	Move right
[space]	Jump
Hold [W]	Look up OR Climb up ropes
Hold [S]	Look down OR Descend ropes
Hold [S] + [space]	Drop through platforms

Table 5.3.2: Bow and Arrow Controls

Numbers [1] - [6] OR Scrollwheel	Select* an item
[R]	Equip* a selected item
Hold [R]	Drop an item from your inventory
Aim using the mouse cursor	
Hold Left Mouse Button	Draw an arrow (Must have a bow and arrow equipped)
Release Left Mouse Button	Fire an arrow

**item selection is represented with a white frame*

***Equipped items are represented with a red frame. Only one bow and one arrow can be equipped at a time.*

Table 5.3.3: Application controls

[esc]	Quit the application.
[R]	Restart from level 1.

In addition to the basic controls, the tutorial teaches players some game mechanics. Players are taught that releasing an arrow in the green zone results in a perfect shot (see Figure 5.3.2) and how to interact with totems and the end goal. The rest of the game is left up to the player to learn. Some game features are apparent to the player throughout the game, such as the perfect shot bonus appearing in the UI or the way to gain points being revealed at the end of each level. Other features are left for the player to figure out, like discovering item interactions, enemy interactions, and learning how to navigate around the levels.

Project Quiver may have explicit objectives for the players to work towards; however, it is up to the player to decide whether to go for these objectives or to create their own. Some of the participants during testing opted for their own alternative goals, such as using the game's interactions to break mechanics, reaching the goal as fast as possible or completing as many totem challenges as possible.

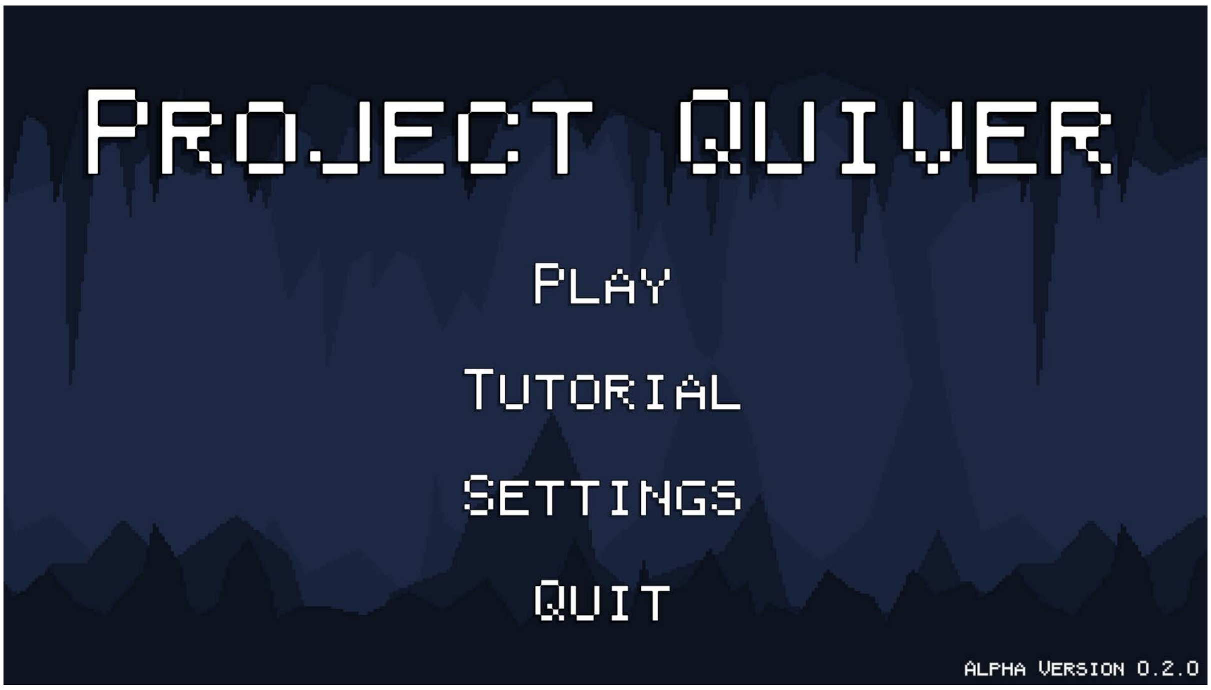


Figure 5.3.1: The Project Quiver menu screen.

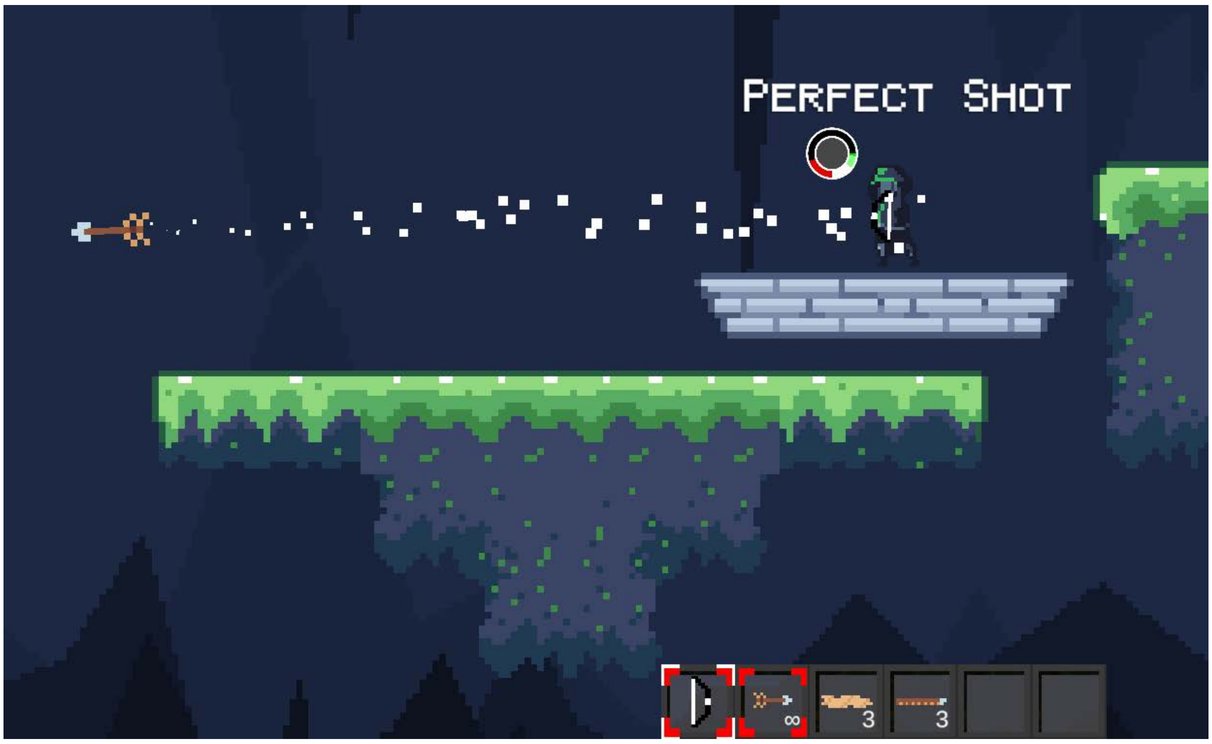


Figure 5.3.2: The player performing a perfect shot.

5.4. Prototype Potential

The prototype will have an appropriate number of items, mechanics and interactions for a play session of approximately thirty minutes. Within this session, the player may discover some interactions to create emergent gameplay, although these interactions are limited to the currently implemented items, enemies and environment available in the game. Increasing the number of subjects, interactions, and instability of the game can increase the possibility of emergent gameplay. Furthermore, by adding meaning to the interactions, balancing the constraints, keeping interaction rules consistent and rewarding player experimentation, we can encourage the discovery of more emergent scenarios, resulting in a more emergent game.

5.5. Conclusion

This chapter investigates how factors of emergence can be used to assist the development of an emergent game prototype. The Emergent Framework can then be applied after prototype testing to measure each factor's inclusion more accurately, as seen in the following chapter. The PCG for Emergence framework was used to pick a suitable mix of algorithms with an appropriate blend of consistency and instability to complement each other. The result of the emergent theory is the prototype Project Quiver, a roguelike platformer game with procedurally generated level layouts and an arsenal of items for the player to experiment with.

Chapter 6

Validating the Prototype through the Framework

6.1 Introduction

Due to the nature of emergence in general, a game cannot be considered genuinely emergent without the usage of players, even if it does abide by emergent factors. Instead, the factors are in place to facilitate an experimental environment, which the players can then use to their advantage in their emergent plans. The environment should also contain unpredictable elements and instability to create emergent moments for the player. This chapter aims to test the finished prototype with a small sample size of gamers to observe occurrences of emergence and chart the game against the emergent framework to find its strengths and weaknesses.

6.2 Preparation for Prototype Testing

Creating a prototype with emergent factors will not be enough to be considered an emergent product. Because of this, playtesters will be invited to try out the prototype and have their gameplay actions monitored to find instances of emergent behaviour occurring. Emergence will go by an early definition given by Jesper Juul, "...the appearance of new possibilities that arise from the interplay between game mechanics" (2005).

The project aims to act as a preliminary study for the field, using a small sample size of people with the potential to lead to a more extensive future study. Previous studies have been conducted with fewer participants, such as Paananen's paper "Designing a Game for Emergent Gameplay" (2020), which has four players during the prototype's second iteration but was still successful in promoting emergence. Initially, twenty participants were planned as the quantity allows various player strategies to emerge, which is very important in emergent games. In the end, eleven participants were involved in the experiment due to a data saturation cap being hit. The later tests resulted in repeated player actions, resulting in a decrease in any new emergent behaviour; thus, the experiment data collected was deemed sufficient.

Each participant selected for this experiment was familiar with video games and fulfilled the following inclusion criteria:

1) Fluent in English.

2) Have not contributed to the design or development of the game.

3) Does not have a relationship with the researcher.

4) Has a good understanding of game design theories.

5) Is an active gamer.

6) Is not underage.

7) Can consent.

8) Has no cognitive impairments.

9) Has no prior history of epilepsy.

Recruitment was done through various game development communities, both online and in-person. Many of these communities emerged organically through social media and other online channels. These communities are not companies or organisations but groups of people with a common interest in game development. After calling out to these communities, participants expressed their interest through a custom Microsoft form. Some examples of these communities include Beer and Pixels, Games with Wings Sydney, Game Developers of Australia and various game development clubs.

A ten-minute pitch was given to gather interest in the research project, which discussed topics including procedural content generation, emergent gameplay and roguelikes. Flyers were also handed out for potential participants to register their interest in the project. Since users were informed about these concepts before the experiment, it is important to disclose that this knowledge may have affected the experiment's results. As an example, a user may have felt more or less inclined to use emergent strategies after being informed that the experiment looks to investigate emergent gameplay.

The total time of the experiment session for each participant was one hour. To play through the prototype, participants were given thirty minutes maximum. Participants were allowed to end their session before the thirty-minute mark, which may risk their results not being included in the experiment. This time limit was chosen based on the content within the game. Thirty minutes is enough time for the player to grasp the game’s systems, allowing them to develop interesting plans that may not have been intended while making the game.

Participants were given an application to sign before the experiment to ensure their consent was given to collect their answers and observe their in-game actions. The testers could withdraw their consent anytime, although this did not occur with the selected playtesters. Hypothetically, if consent were withdrawn before the experiment, the tester would no longer be required to try the prototype. The tester's withdrawal would also mean that the tester’s actions would no longer be recorded, and any previous actions they took in the prototype would be removed from the records.

Before and after the prototype testing session, participants would also participate in an interview to get their thoughts on emergence in games. The interview segments would take a combined total of approximately thirty minutes, with the same consent rules applying as the prototype testing session (i.e. participants could withdraw their consent at any time). The starting interview would evaluate the level of knowledge the participant had on emergent gameplay. In contrast, the later interview would further test their knowledge and collect feedback on whether they found the prototype to be emergent.

Any emergent occurrences that the participants experienced were recorded in the prototype observations. This included interactions that occurred naturally from the game's systems interplaying with each other, as well as less conventional interactions that resulted from unintended features or exploits.

6.3 Prototype Observations

Participant data was collected in two different experiment sections. Observational notes were taken during the time that participants would be testing the prototype. The type of data collected includes the participant's playstyle, their perceived skill level, the occurrences of emergence (both intended and unintended), and their preferred choice of items. Other notes were taken based on the participant's answers during the interview section. These notes included what users perceived as emergent, their understanding of the level generation, their perception of emergent throughout the prototype and what factors they believe affect emergent gameplay.

This data would then be mapped to a user-testing spreadsheet to collate the data and make observations based on the numbers (see Appendix B.0 to B.13). The categories marked with an asterisk (*) mean that participants may have selected multiple subcategories, whereas categories without an asterisk mean only one category can be selected. The figures provided focus on the sum and percentages to obtain people's consensus on the game and its emergence. Some significant individual data and comments are mentioned in the emergent occurrences section.

There were a total of eleven participants. Percentages were taken by dividing the sum that agrees by eleven. The only exception is in Appendix B.2 where users with a low understanding of emergence were not considered. After the testing session, users were given a definition for emergence to answer questions for the data in Appendix sections B.10 through B.13. The definition given was the one defined by Jesper Juul as "...the appearance of new possibilities that arise from the interplay between game mechanics" (2005).

6.4 Prototype Analysis

6.4.1 Emergent Occurrences

Throughout the testing session, several identified interactions were considered emergent. The most common discovery was "Arrow Planting". When the player shoots an arrow into the wall or ground, it remains there for approximately half a second before disappearing; however, during this half a second, the hitbox of the arrow is still active and can damage enemies. A couple of participants used this to their advantage, predicting the enemy's path and planting the arrow trap for them to take damage. Although this happens to be an unintentional byproduct of the arrow despawning system, players enjoyed employing this strategy. Perhaps it could be worked upon to be a more typical game mechanic.

Another somewhat unintentional interaction was the "Quick-Fire" strategy some participants used. While there was a minimum charge requirement for an arrow to fire, some astute players noticed that no matter the charge of the bow (except for the perfect shot), the arrow would always deal the same amount of damage. Some players took advantage of this interaction and opted for this strategy instead of aiming for perfect shots. This strategy was unintentionally encouraged since the player starts with an infinite basic arrow, making ammo conservation less of a worry to the player.

The original intention for ropes was to access higher points within the level generation; however, some players had other plans. Some players would use ropes as an escape plan to avoid enemies better, while others used them to create a vantage point over grounded enemies. One player even found a glitch with the rope, where attaching themselves to it in a certain way could result in the player character phasing through tiles. Unfortunately for that player, the bug was difficult to replicate, so they could not use it for their emergent plans.

Initially, before the prototype was used in the experiment, the level-generated spikes would kill the player in a single hit. This game version was tested with some early access testers who were not a part of the experiment, and the feature was later adjusted for two reasons. One was that the game was perceived to be too difficult, with the most significant challenge being avoiding the deadly spikes. If players were to die too early in the game, they would be unable to play around with the game's later interactions, reducing the possibility of emergence. The other reason was that dying to the spikes felt like too linear of an interaction. While logically making sense and reinforcing the meaning factor of emergence, players perishing to spikes would prevent future emergent moments from occurring in that playthrough due to the death. Later, the spikes were made to deal damage and knockback to the player, which was seen to increase emergence from the experiments.

The combination of the level generation and enemy knockback resulted in some entertaining emergent gameplay scenarios, albeit against the players rather than for their benefit. The later levels would generate more and more spikes, which also had knockback properties. The knockback properties resulted in a pinball-like effect, where an enemy would knock back the player into spikes, which would then knock them back into another enemy or more spikes. This type of behaviour is especially prevalent in similar games like *Spelunky* (2008) and *Noita* (2019), and despite players occasionally dying in these scenarios, they appeared to enjoy the emergent occurrence.

When it came to discovering emergent strategies, there was no particular time where more emergent occurrences would occur than others. When it comes to commercial games, many emerging strategies appear later in the game when the player base is familiar with the base mechanics and interactions. With Project Quiver's scope being relatively small compared to more commercial games, emergent occurrences could appear anywhere from the start to the end of the testing session. For example, some users would discover and utilise the Quick-Fire strategy while others would find it in the final two minutes of gameplay.

There was also no perceived relation between the number of sessions and the observed emergent behaviour. Instead, the sessions that resulted in the most emergent sessions occurring were the ones where users had the most familiarity with games in the same genre, such as platformers or other roguelikes. These participants were more likely to experiment with the game system, occasionally resulting in emergent behaviour.

6.4.2 Analysis against the Emergent Framework

The testing session also served as an excellent way to observe what emergent factors were lacking in the prototype, even after planning the features through the Project Quiver Matrix (See Appendix A.3.0 - A.3.5). Table and Figure 6.4.1a below use the Emergent Framework from Chapter 4.2 to evaluate the prototype's perceived emergence level. Table and Figure 6.4.2b compare the prototype to the market emergent games evaluated in Chapter 4.2.

Table 6.4.1a: Project Quiver's perceived emergence rating.

Emergent Factors	**Project Quiver**
Interactions	3
Reducing Constraints	4
Adding Subjects	3
Instability	4
Meaning	5
Consistency	4
Experimentation	4
Agency	5

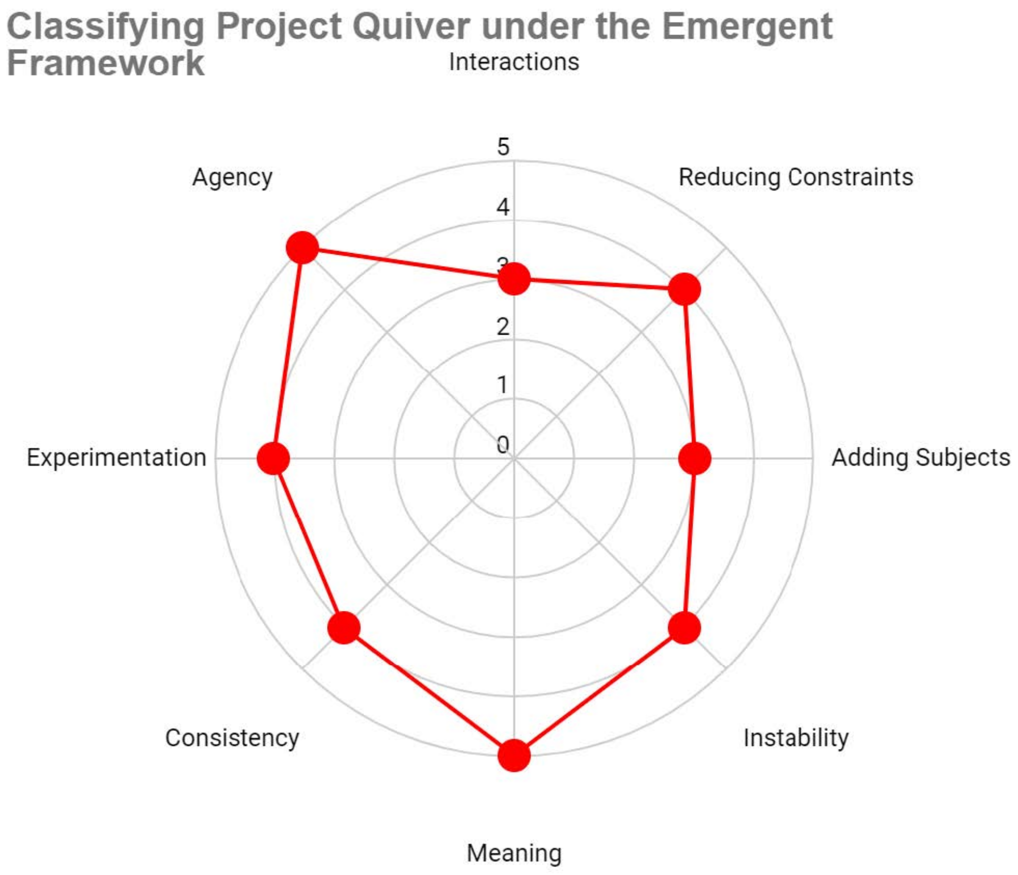


Figure 6.4.1a: Project Quiver's perceived emergence rating on a radar chart

Table 6.4.2b: Project Quiver's perceived emergence rating vs emergent games

Emergent Factors	Project Quiver	Spelunky	Noita	BOTW	Minecraft	Don't Starve
Interactions	3	4	5	5	5	4
Reducing Constraints	4	4	4	4	5	3
Adding Subjects	3	4	5	4	5	5
Instability	4	4	5	1	5	4
Meaning	5	5	4	5	4	4
Consistency	4	5	4	4	4	5
Experimentation	4	4	4	4	4	4
Agency	5	4	5	5	5	4

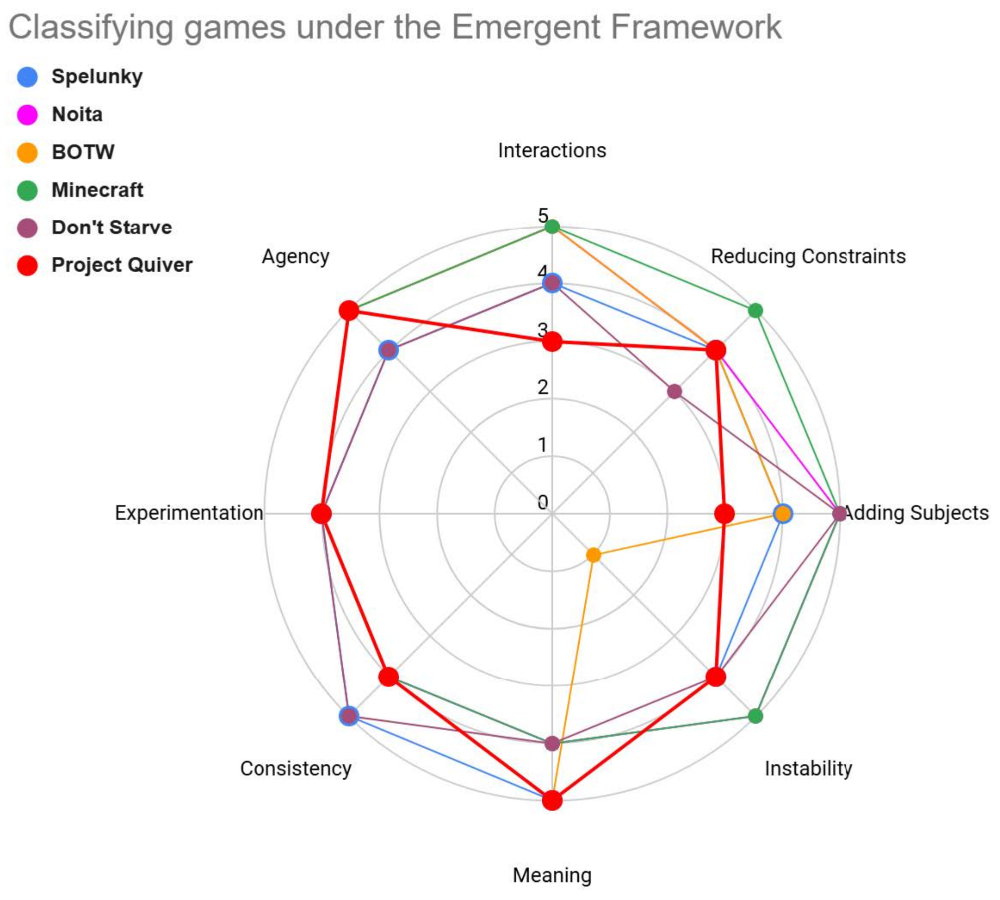


Figure 6.4.2b: Project Quiver's perceived emergence rating on a radar chart vs emergent games

Table 6.4.3 shows what users believed to be the most impactful factors for emergence. Users were given a couple of choices to score these factors. A score of one means the user agrees that the factor will increase emergence. A score of 0.5 means the user slightly agrees that the factor could increase emergence; however, its impact would not be as significant as other factors that scored a one. A score of zero means that the user believes that the factor has no effect on emergent gameplay or even has a detrimental impact on emergence. Figure 6.4.3 displays the same data in a bar chart format:

Table 6.4.3: Participants perception of emergent factors

Factors	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	Sum	%
Increase Interactions	1	1	1	1	1	1	1	1	1	1	1	11	100
Reduce Constraints	1	0	1	0.5	1	0.5	0	0	1	0	0	5	45.45
Add Subjects	1	0.5	1	0.5	1	1	1	1	0.5	1	1	9.5	86.36
Instability	1	1	1	1	1	0.5	0	1	1	1	1	9.5	86.36
Meaning	1	1	0.5	0.5	1	0.5	1	1	0.5	1	1	9	81.82
Consistency	1	0.5	1	0.5	0.5	0	0.5	1	1	0	1	7	63.64
Encourage Experiment	1	0.5	0.5	1	0.5	0.5	1	1	1	0	1	8	72.73
Player Agency	1	0	1	1	1	1	0.5	0.5	1	1	1	9	81.82

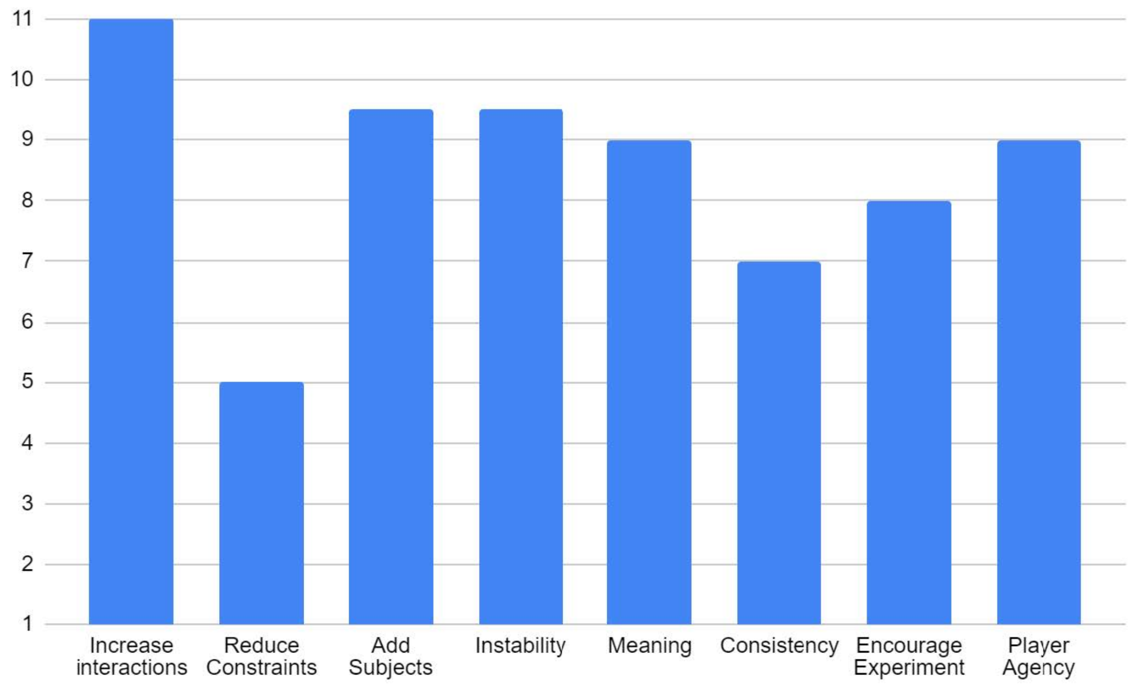


Figure 6.4.3 Participants' perception of emergent factors on a bar chart

6.4.4 Participant Data Analysis

Appendix B.1

There was a mixed level of knowledge when defining emergence among the participants. It is also important to consider the fact that the selected participants each have a good familiarity with a variety of different games. Participants in the low category could point out an advantage of emergent gameplay (e.g. allowing players to play a game the way they want) but had difficulty defining it and describing how it can be created. Those in the middle category could briefly define emergence and state some of its benefits. Those with a high understanding of emergence gave an accurate definition and could identify a couple of ways of increasing emergence in games.

Emergent gameplay is more commonly known among specific gaming collectives, such as those who enjoy roguelikes, open-world games and immersive sims. Although all participants are familiar with games, the resulting percentages are not too surprising as the participants likely play other types of games.

It is also important to note that, as mentioned in section 6.2, participants were provided with an explanation of emergence in games before the experiment during the recruiting process. This explanation is likely to have affected the level of knowledge of emergence amongst the participants, resulting in each participant having some understanding of the concept. The time between the recruiting process and experiment was two months, so it was not uncommon for some participants to forget this topic and have less knowledge than others.

Appendix B.2

A notable part of the data is the participant's perception about whether emergent behaviour must not be specifically programmed in. This notion was taken from Rabin's definition, "... behaviour that occurs when simple independent rules interact to give rise to behaviour that was not specifically programmed into a system" (2004). Most users agreed with this definition, although there was some disagreement with some participants due to the wording. They believed that emergent behaviour can result from something programmed specifically into a system, though the interaction would still need to have the qualities of the other emergent definitions.

As for whether emergence needs to be unintended by the developers, five participants disagreed with this statement, while two agreed. These seven participants either had a medium to high level of knowledge when defining emergence in games. Even still, the conflicted results further demonstrate the confusion that comes with defining emergent gameplay, and perhaps it would be best to split the definition into two interpretations, similar to what Paananen has proposed in “Designing for Emergent Gameplay” (2020).

Appendix B.3

There was a good mix of playstyles adopted by the participants. Some of the more common styles were the challenge completionist, who enjoyed the challenges and rewards the totems offered; the experimenter, who liked messing around with the different game interactions; and the speedrunner, which was popular for the last 15 minutes of the session due to the participants wanting to experience as much of the game as they could before the time for testing ran out. There was little incentive to kill all enemies, and testers generally did not pay attention to their score, reducing those playstyles from appearing.

Appendix B.4

While participants would use other bows when they got them, they had little preference for which one they preferred, making the default bow win out against the others. These bows had to be discovered throughout the level, making the other bows a less popular choice.

The basic arrow being infinite made it a popular choice among the participants, perhaps sometimes too popular as they did not get to play around with the other arrows all that much.

Rope arrows were commonly used for traversal but also overshadowed the platform arrow quite frequently since the ropes were considered more beneficial to the structure of the generated environment. Finally, participants who obtained bomb arrows either experienced an accidental death due to the self-damage or found them too good to use, making their usage rather unpopular.

Appendix B.5

Regarding the bow charging style, most opted for perfect shots as they understood the benefits it provided in battle. However, a notable number of people opted to release the arrow as soon as the minimum charge requirement was met, later dubbed the “Quick-fire” strategy. Some participants used this strategy as they found it more optimal for damage, while others simply enjoyed using a mechanic that was perceived as unintended.

Appendix B.6 - B.7

Participants’ initial skill level and skill level change were about as expected. Some participants did improve as they continued playing while others were at the same skill level, either because they were already quite good at the game, to begin with, or because there was not enough time to make a significant enough improvement in skill.

Appendix B.11

When it came to emergent gameplay, most had experienced a couple of interactions they considered to be emergent. The consensus was that there were emergent occurrences, but for a game to be truly emergent, there needed to be many more emergent interactions.

6.4.5 Justification behind the ratings - Emergent Factors

One element that every participant could agree on was that increasing the number of interactions was an integral way of creating more emergence. There is no surprise here, as emergent gameplay must involve interactions. Adding subjects and meaning were also agreed upon by all participants as ways to increase emergence, with some believing it has a more impactful effect than others.

Although instability was mostly agreed upon, one participant noted that instability does not need to be present when creating emergence. Their explanation also held some truth as there are examples of released games with minimal randomness that can create emergence. Another participant believed that player agency did not matter much in the emergent space, as emerging occurrences can appear without the player's intervention, while others believed player agency to be quite significant.

Consistency and encouraging experimentation were mostly agreed upon but with some outliers. The participants who rated the factor as 0.5 thought these factors could help emergence slightly, but not in every case. Those who rated the factor zero did not believe the factors could create emergence, making sure to establish a distinction between creating emergent behaviour and encouraging it.

It became clear that reducing constraints was the most divisive factor for increasing emergence, which also mirrors the divisive takes in literature. While most sources agree that reducing constraints results in more player actions and, by extension, potential emergence, sources such as *ArkaNet: Investigating Emergent Gameplay and Emergence* discuss how limitations can result in an expansion of tactics, resulting in more emergence (Leijnen & Veen, 2016). Approximately half of the participants did not agree that removing constraints would result in more emergence. They believed that constraints should instead be viewed as a balancing act rather than something to remove altogether.

6.4.6 Justification behind the ratings - Project Quiver

Although there were a good number of different items and interactions at the player's disposal, many participants felt that there needed to be more use cases for the items and more interactions with the enemies. Since interactions were still commonly used within the game, but not a wide enough variety was available, a rating of three was appropriate.

Participants did not feel too constrained when playing through the game and felt that six was a good number for the inventory slots were a good number. The one shortcoming was that the item resource quantity felt too small, bumping constraints from a five rating to a four.

Adding subjects was rated at a three due to an appropriate number of subjects involved in the level (spikes, enemies, etc.), but there was still much room for improvement. Participants felt there could have been more variety amongst enemies and obstacles and more ways to take advantage of the subjects for their emergent plans.

With the levels being entirely procedurally generated through a mix of Spelunky's map generative method and the overlap model of Wave Function Collapse, it becomes clear that instability is one of the game's core values. Levels had some interesting variations, making them a replayable and entertaining experience for the participants. Even still, having more input samples for WFC to draw from could help increase the level variation, or procedurally generated stats and attributes for items like in *Noita* (2019) could also support more instability. In the end, an instability rating of four was appropriate.

The meaning factor was apparent to the player throughout the game, giving it a rating of five. Players could quickly get the hang of using the bow due to their familiarity with other games featuring bows (i.e. holding to draw and releasing to fire). The items were intuitive enough that the players could infer some meaning without using them, such as the sprite of the bomb arrow looking like a small explosive on the tip of an arrow. Enemies behaved in an easy way for the player to understand, like the slime that would predictably bounce towards the player. Although spikes did not kill the player in a single hit, players were still cautious around them, knowing they would deal damage on contact.

The variety of items and their interactions with other elements in the world, as well as a relatively high meaning and consistency rating of the game, did help facilitate an experimental environment. In Appendix B.3, it can be seen that 45% of participants opted for an experimenter playstyle, looking to test the bounds regarding items and enemies. The users who adopted that playstyle also mentioned that they would have liked to see more interactions, particularly those that affect the environment. With more interactions at the player's disposal, the current rating of four for encouraging experimentation has the potential to be increased.

The game events and interactions were very consistent as well. Enemies had predictable patterns that the player could exploit, levels generated in a way where players would always know where to go next, and items behaved in an expected way to their use case. This level of consistency would have resulted in a five consistency score; however, one failure was that spikes did not have a consistent interaction between enemies and the player. Some players initially speculated how they could use spikes to defeat enemies, but after seeing that enemies were immune, those plans were lost. This oversight, unfortunately, resulted in some emergent plans being lost and was enough to reduce the consistency score to a four.

The player agency factor was never an issue with the player for the prototype. Players always felt comfortable with the tools they had at their disposal, whether it was the player's general movement and physics system or the different items they could discover throughout the levels. The inventory slot limit never felt too limiting on the player's items and some players made informed decisions about what items they wanted to keep for the next level. The factor was given a rating of five.

6.5 Conclusion

After testing the prototype with participants, it can be seen that there were indeed cases of emergent gameplay occurring. Participants agreed the prototype could facilitate a low to medium level of emergent gameplay moments upon this first iteration. If the prototype continues development, the main areas of focus to increase emergence would be to add more interactions and subjects. Constraints, instability, consistency, and encouraging experimentation are other areas of consideration for improvement to create more emergence.

It is worth considering that although this prototype was made with emergence in mind, a game that aims to capitalise on emergent elements would need to go through many more iterations with players in order to facilitate an emergent environment. The Emergent Framework could be used in each iteration to focus on the weakest factors and improve them in future iterations. The games evaluated in Chapter 4.2 have been in development for quite some time and make great use of their emergent factors, hence their high factor and emergence rating when compared against Project Quiver.

Chapter 7

Outcomes and Conclusion

7.1 Introduction

When it comes to game emergence, it can be challenging for developers to understand how it is implemented, especially with the added unpredictability of certain PCG algorithms. Despite the strong design and testing requirements, the benefits of emergence in games outweigh this, thanks to its player appeal, allowing for more freedom in player plans and resulting in more surprising and unique playthroughs compared to more traditional linear games.

Procedural content generation has been shown to support the creation of emergent gameplay, thanks to its ability to create unique scenarios and instability within game worlds and systems. Many different algorithms can be used to generate content within games, so it is up to the developer's discretion to select a suitable PCG algorithm to suit their needs. A good amount of instability can help support new experiences for the player. However, without some consistency, it can be difficult for the player to develop emergent plans when too many variables involved are unpredictable.

Roguelikes are one of the common genres to utilise PCG for emergent gameplay, alongside open-world sandbox games. Their high replayability factor allows players to try out different strategies in the various environments generated by the game. With enough systems to experiment with in the game, players can experience a game for hundreds of hours whilst still finding new interactions and synergies.

7.2 Research Outcomes

The following research questions were proposed in Chapter 1:

- RQ1. How do developers design games for emergent gameplay?
- RQ2. How can PCG be used to create emergent scenarios?

A literature review was conducted on emergent gameplay to answer the first research question. Within the review, a number of common factors and agreements were found on how emergent gameplay can be promoted within games, such as increasing the interaction space and adding more interaction subjects, to name a few examples. In addition, procedural content generation was another research topic investigated to find out its relationship with creating emergent scenarios. One common link that was mentioned between the two topics was roguelike games, as they were commonly known for using PCG and having emergent gameplay, which made it another topic of interest for the literature review.

The following hypotheses were proposed after the literature review stage:

- That the current state of the literature on emergence in games can expand to include additional factors and techniques
- That procedurally generated content can be evaluated under the lens of emergence to help promote its inclusion in games

While the first research question can be answered using past literature, it can be expanded past its current state with the addition of new factors. New factors were identified in Chapter 3: rule consistency, more player agency and encouraging experimentation. While not physically increasing the interaction possibility space, these factors can help promote player's emergent plans, resulting in a more emergent environment. To answer the second research question, PCG was investigated under the lens of emergence; more specifically how it can be used to facilitate an emergent environment. The procedurally generated levels were the most common way to support an emergent game, however, there exist other applications. Story generation and the inclusion of more unstable game elements may result in more emergent stories that are a result of PCG. In this sense, both of the research questions have been answered.

The Emergent Framework in Chapter 4 uses the extended list of emergent factors and adapts it into a framework. This framework can then be applied to games to visualise better the impact emergent factors have on systemic games. The PCG for emergence framework exists to evaluate a PCG algorithm's potential for creating emergence. PCG methods analysed through the framework will show the balance between instability and consistency and find ways to introduce more emergent possibilities through subjects. The Emergent Framework has been applied to five commercial games in Chapter 4, although its usage can also be applied to the development process of a game, as seen in Chapter 5.

In Chapter 5, a prototype was developed with the framework's principles in mind, with each feature being mapped to an emergent factor to justify its inclusion. The PCG methods selected for the prototype were Wave Function Collapse and the Spelunky map generative method, for a suitable mix of instability and consistency. In Chapter 6, the prototype was tested amongst participants to see how effectively it creates more game emergence. Instances of emergent occurrences were recorded, and an analysis was done for the emergent factors that should be focused on.

7.3 Limitations and Future Directions

Regarding the work completed for the thesis, some limitations must be considered. With eleven participants for the experiment, the insightful findings may not be entirely generalizable to a broader audience or demographic. Such a limited participant pool can introduce potential biases and reduce the robustness of the results. Observing emergent behaviour in released emergent game titles may be worthwhile to gather qualitative data, as there will be an already established demographic to obtain data.

The time and scope are other factors that are important to consider. More emergent gameplay moments can occur with a longer duration to develop the prototype (e.g., more systems and interactions) and longer testing sessions. Should the prototype continue development to become a highly emergent game, the framework can be used as a useful tracking tool to identify what factors of emergence need improvement.

An issue that could have been further considered and investigated is the variation of generated content. Kate Compton likens this problem to “10,000 bowls of oatmeal” in her GDC talk *Practical Procedural Generation for Everyone* (2017). In her example, a generator for oatmeal is made to produce 10,000 unique oatmeal variations, however, despite the differences between each bowl, none of it matters if the changes are unable to be identified by the consumer. This issue is very applicable to PCG games, where many games boast about their random generation but end up having content that is difficult to distinguish from one another.

Isaac Karth discusses this problem and reveals that the problem can be mitigated through variation in *Preliminary Poetics of ProceduralGeneration in Games* (2019). More specifically, he goes into the principles of multiplicity, style and cohesion which each have their role when creating varied content that is noticeable and impactful to the target demographic. While variation was not a particularly prevalent issue in the prototype, a longer play session with the participants may have made the variation problem more prevalent, and something that would need to be addressed should the prototype continue development.

Finally, with the ever-growing gaming industry, more PCG algorithms could be considered and evaluated for emergent games. The generated game content could come from a mix of existing algorithms to create something new or the introduction of a new algorithm that better suits the needs of the game designer. Another potential direction of interest is AI-driven Procedural Content Generation (PCG), which can be developed to dynamically craft game environments and narratives that adapt to player behaviour for even more emergent experiences. As the gaming industry and research on emergence continue to evolve, the mix of PCG and emergent design shows much promise for a future of unique gameplay experiences.

Bibliography

1. Referenced Papers

1. Ampatzidou, C., 2019. Reinventing the rules: Emergent gameplay for civic learning. In *The Hackable City* (pp. 187-203). Springer, Singapore.

2. Cheng, D., Han, H., & Fei, G. (2020). Automatic generation of game levels based on controllable wave function collapse algorithm. In *Entertainment Computing–ICEC 2020: 19th IFIP TC 14 International Conference, ICEC 2020, Xi'an, China, November 10–13, 2020, Proceedings 19* (pp. 37-50). Springer International Publishing.

3. Ebia, J.M.S., Manalansan, J.C.R., Guino, K.D.D. and Bunagan, J.K.V., Influencing Game Dynamics in A Roguelike Game Through Procedural Content Generation Using Genetic Algorithm

4. Gaisbauer, Raffe, W., Garcia, J., & Hlavacs, H. (2019). Procedural Generation of Video Game Cities for Specific Video Game Genres Using WaveFunctionCollapse (WFC). *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, 397–404.

5. Gellel, A. and Sweetser, P., 2020, September. A Hybrid Approach to Procedural Generation of Roguelike Video Game Levels. In *International Conference on the Foundations of Digital Games* (pp. 1-10).

6. Ho, X., Tomitsch, M., & Bednarz, T. (2016). Finding design influence within roguelike games. In *Meaningful Play 2016 Conference Proceedings*.

7. Hokkanen, Holmes, T., Koivuranta, H., Sandberg, A., Sorva, H., Toikka, J., Hämäläinen, P., & Kaos, M. (2018). Plusminus: Augmenting Physics to Promote Emergent Gameplay. *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, 321–327.

8. Izgi, E. 2018, Framework for Roguelike Video Games Development, Master Thesis, In *Charles University Digital Repository.*

9. Karth, I., & Smith, A. M. (2017, August). WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (pp. 1-10).

10. Karth, I. (2019). Preliminary poetics of procedural generation in games. *Transactions of the Digital Games Research Association*, 4(3).

11. Leijnen, S. and Van Veen, F., 2016. ArkaNet: Investigating Emergent Gameplay and Emergence. In *1st DiGRA/FDG conference, 1 augustus 2016, Dundee, Schotland.*

12. Melotti, & de Moraes, C. H. V. (2019). Evolving Roguelike Dungeons With Deluged Novelty Search Local Competition. *IEEE Transactions on Games*, 11(2), 173–182.

13. Paananen, K.E., 2020. Designing a Game for Emergent Gameplay: Developing Gatedelvers.

14. Parker. (2017). The culture of permadeath: Roguelikes and Terror Management Theory. *Journal of Gaming & Virtual Worlds*, 9(2), 123–141.

15. Rabin, S. (2004). Common game AI techniques. AI game programming wisdom, 2, 3-14.

16. Raffe, W. (2014). Personalized procedural map generation in games via evolutionary algorithms.

17. Sweetser, P., 2006. An emergent approach to game design: Development and play. PhD diss.: University of Queensland.

18. Togelius, J ; Yannakakis, G. N ; Stanley, K. O ; Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey In *IEEE transactions on computational intelligence and AI in games.* Vol.3 (3), p.172-186

19. Wilson, M.J.B., 2020, September. Playing with Permadeath. In *2020 IEEE Games, Multimedia, Animation and Multiple Realities Conference* (GMAX) (pp. 1-5). IEEE.

2. Other Sources

1. Brown, M. from GMTK. (2018, Feb 15). *The Rise of the Systemic Game*. YouTube.

2. Compton, K. at GDC talk. (2017, June 1). *Practical Procedural Generation for Everyone*. YouTube.

3. Donald, M., (2020, Aug 1). *Superpositions, Sudoku, the Wave Function Collapse algorithm*. Youtube.

4. Dohta, T., Fujibayashi, H., & Takizawa, S. at GDC talk. (2017, March 11). *Breaking Conventions with The Legend of Zelda: Breath of the Wild*. YouTube.

5. Francis, T. (2018). *What Works And Why: Emergence*. Rock Paper Shotgun. Online Article

6. Hocking, C. at GDC talk. (2006). *Designing to Promote Intentional Play*. GDC Vault.

7. Juul, J. 2005. Half Real. *Videogames between Real Rules and Fictional Worlds*. MIT Press.

8. Johnson, S. (2011). *GD Column 17: Water Finds a Crack*. Designer Notes.

9. Marian, (2019). *Infinite procedurally generated city with the Wave Function Collapse algorithm.* Article.

10. Parker, J. (2018). *Generating Procedural Game Worlds with Wave Function Collapse*. GitHub.

11. Shaker, Togelius, J., & Nelson, M. J. (2016). *Procedural Content Generation in Games* (1st ed. 2016.). Springer International Publishing.

12. Smith, H. (2002, March). Systemic level design. In the Game Developers Conference (pp. 21-23).

13. Smith, H. interview with NoClip. (2018, January 6). *Deus Ex to Dishonored with Harvey Smith*. YouTube.

14. Sweetser, P. (2008). *Emergence in Games*. Charles River Media.

15. Witney, O., 2019. *Usage of Random Number Generators and Artificial Intelligence to improve replayability of a Roguelike Game*.

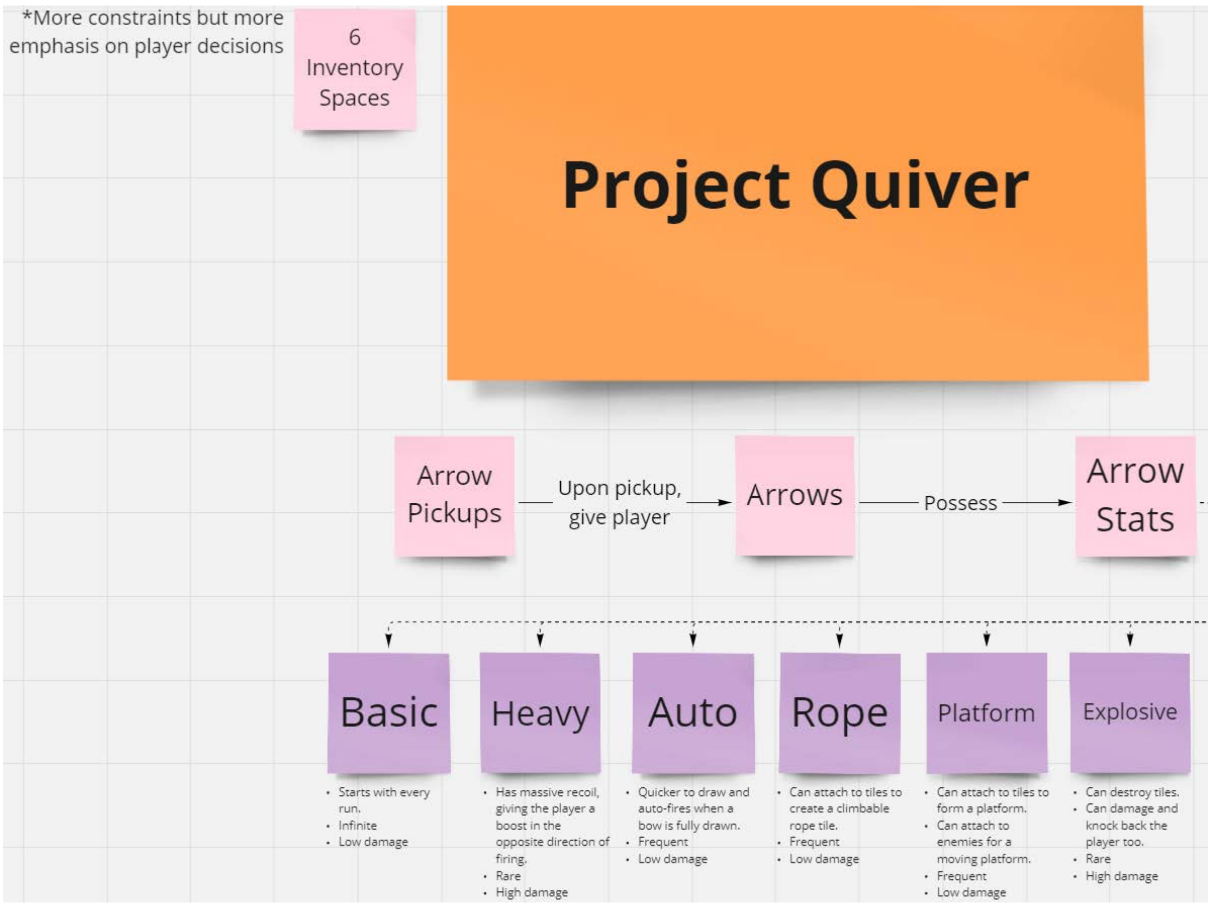
16. Wolverson, H. at Roguelike Celebration talk. (2020, October 16). *Herbert Wolverson - Procedural Map Generation Techniques*. YouTube.

17. Yu, D. (2016). *Spelunky by Derek Yu* (1st ed). Boss Fight Books.

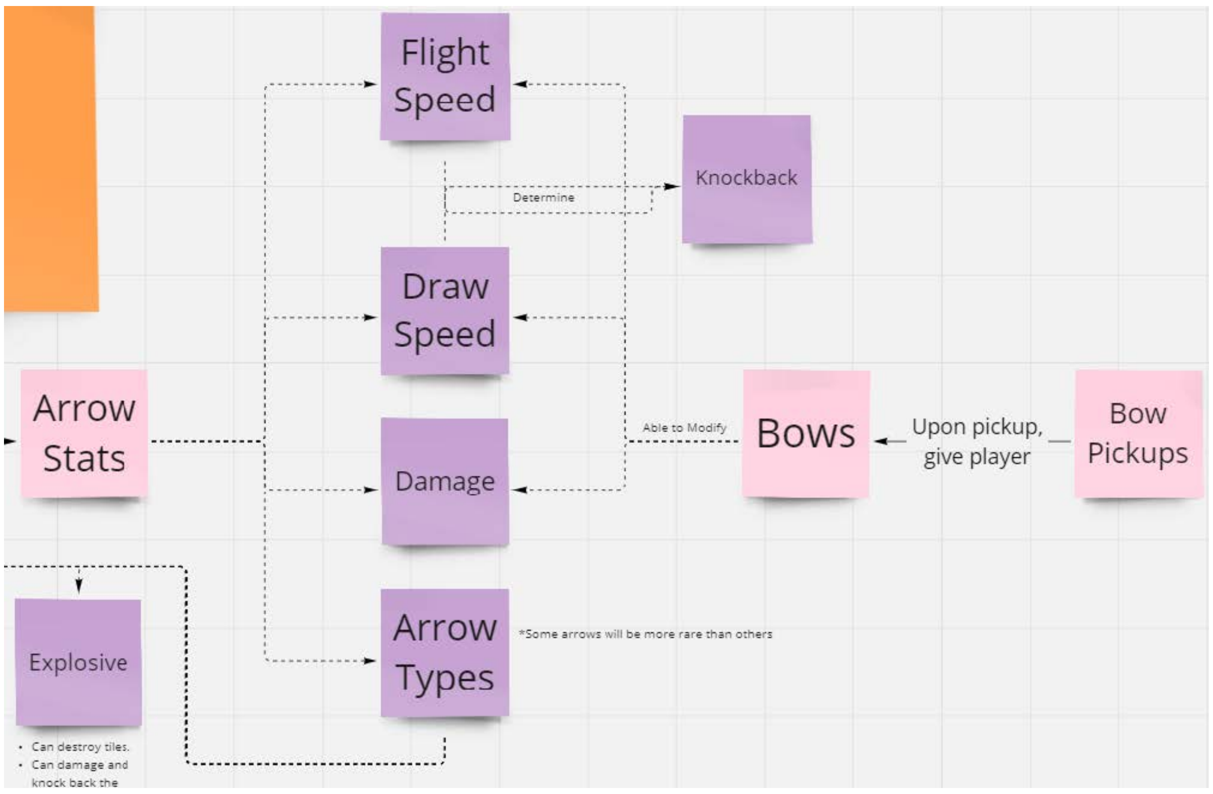
Appendices

Appendix A - Designing Project Quiver

A.1 Project Quiver's Design Map Part 1


*More constraints but more emphasis on player decisions
6 Inventory Spaces
Project Quiver
Arrow Pickups
Upon pickup, give player
Arrows
Possess
Arrow Stats
Basic
Heavy
Auto
Rope
Platform
Explosive
• Starts with every run.
• Infinite
• Low damage
• Has massive recoil, giving the player a boost in the opposite direction of firing.
• Rare
• High damage
• Quicker to draw and auto-fires when a bow is fully drawn.
• Frequent
• Low damage
• Can attach to tiles to create a climbable rope tile.
• Frequent
• Low damage
• Can attach to tiles to form a platform.
• Can attach to enemies for a moving platform.
• Frequent
• Low damage
• Can destroy tiles.
• Can damage and knock back the player too.
• Rare
• High damage

A.2 Project Quiver's Design Map Part 2



A.3.0 Project Quiver Feature to Emergent Factors - Key and Ratings

Key:

A. Increasing Interactions
B. Reducing Constraints
C. Adding Subjects
D. Creating Instability
E. Adding Meaning
F. Keeping Rules Consistent
G. Player Agency
H. Encourage Experimentation

Ratings:

- 1 = Feature increases the emergent factor
- 0.5 = Feature slightly increases the emergent factor
- Blank = Does not apply

A.3.1 Project Quiver Feature to Emergent Factors - Procedural Elements

Refer to A.3.0 for Key and rating system.

Procedural Elements	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**Explanation**
WFC Level Generation			1	1					Instability through random generation. Spikes are added to the level for more possibilities
Spelunky Generation						1			Ensures a path is always generated to the exit
Randomised Pickup Item Locations				0.5				0.5	Location of items is not always the same, encouraging exploration
Randomised Pickup Item Variety						0.5		1	Later the level, the rarer the items. Randomized items can lead to different player approaches
Randomised Enemy Locations			1	1					More enemies, more subjects. Enemy variety is random (unstable)
Randomised Enemy Variety				1		0.5			Later the level, the more difficult the enemies. More enemy types lead to a more unstable environment

A.3.2 Project Quiver Feature to Emergent Factors - Game Systems

Refer to A.3.0 for Key and rating system.

Game Systems	A	B	C	D	E	F	G	H	Explanation
6 Inventory Slots		1					1	1	Choice through constraints. The player must carefully decide which items they want to proceed with
2D Platformer Physics	1				1	1	1		More interaction possibilities when applying forces and using gravity. Forces should emulate real-life physics. Physics should apply all things affected by gravity.
Bow Knockback							1	0.5	Due to 2D physics constraints, the player can use the knockback of the bow to find new movement tech or accidentally land themselves in worse situations
Enemies Dealing Knockback	1					1	0.5	0.5	Knockback adds an interaction. All enemies must deal knockback. Players may use enemy knockback for additional possibilities
Enemies Dealing Contact Damage						1			Traditional game system where enemy contact deals damage. All enemies must deal contact damage
Players Dealing Knockback	1					1	1	0.5	Knockback adds an interaction. All physics driven enemies should take knockback. Players may knockback enemies for additional possibilities
Spikes	1	1			1	1	0.5		Spikes add damage and knockback interactions. They used to kill the

									player in one hit, but this behaviour was too constrained. Spikes symbolise danger for those who touch it. Players can use spike knockback to their advantage.
Bow Draw and Firing					1				Simulates drawing and firing a real bow with physics (longer draw = longer distance travelled)
Perfect Shot							1	1	Perfect shots provide additional effects for new possibilities
Point System						1		1	Points will always follow the same system. Players are incentivised to change their strategies to achieve more points
Health System					1			1	Traditional health system where reaching 0 kills the player. Players with lower health will likely change their strategy. Players health is kept low so that obstacles are more impactful

A.3.3 Project Quiver Feature to Emergent Factors - Items

Refer to A.3.0 for Key and rating system.

Items	A	B	C	D	E	F	G	H	Explanation
Quick Bow							0.5	0.5	Faster draw
Heavy Bow							0.5	0.5	Slower draw for much higher damage
Infinite Arrow							0.5	0.5	Ensures that the player always has a starting option, with reduced damage
Auto Arrows							0.5	0.5	Faster draw for lower damage
Heavy/Great Arrows	1	1					0.5	0.5	Propels the player with higher knockback, allowing them to reach previously unreachable areas. Has a slower draw for higher damage
Rope Arrow	1				1		1	0.5	If the level has some difficult to navigate areas, these can provide more adency over those areas
Platform Arrow	1				1		1	0.5	If the level has some difficult to navigate areas, these can provide more adency over those areas
Explosive Arrow	1	1			1		1	0.5	If the level has some difficult to navigate areas, these can provide more adency over those areas + arrow deals high damage and knockback which can also affect the player

A.3.4 Project Quiver Feature to Emergent Factors - AI

Refer to A.3.0 for Key and rating system.

AI	A	B	C	D	E	F	G	H	Explanation
Magic Arrow	1		1			1			Dash movement adds to the interaction possibility space. Arrows will always dash to the player after spotting them for a designated time
Jumping Slime	1		1		1	1			Jump movement adds to the interaction possibility space. Slimes will always jump on intervals and towards the player when spotted

A.3.5 Project Quiver Feature to Emergent Factors - Challenges

Refer to A.3.0 for Key and rating system.

Challenges	A	B	C	D	E	F	G	H	Explanation
Reach the Totem								1	Encourage the player to use new strategies to reach the totem for a reward
Take No Damage								1	Alters the player's play style by offering rewards for playing more cautiously
Escape in Time								1	Alters the player's play style by offering rewards for playing more quickly
Kill All Enemies								1	Encourages the player to interact more with the enemies for a reward
Shoot All Targets								1	Encourages the player to explore the level for a reward
Reach the Totem								1	Encourage the player to use new strategies to reach the totem for a reward

Appendix B - Project Quiver Observation Data

B.0 Appendix B data

Each number in the following tables represents a participant (i.e. 1 was the first participant to take part in the experiment)

B.1 Participant understanding of emergence

1 represents the participant's understanding of emergence prior to the experiment

Level of Understanding	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Sum**	**Percentages**
Low					1			1		1	1	4	36.36%
Mid	1	1				1			1			4	36.36%
High			1	1			1					3	27.27%

B.2 Elements of the Emergent Definition*

1 represents the participant's agreement with the element described in the definition.

*Participants with an * were not asked about this question due to not understanding emergence at the start of the experiment. This means that the sum of participants was out of 7*

Element of Emergence	**1**	**2**	**3**	**4**	**5***	**6**	**7**	**8***	**9**	**10***	**11***	**Sum**	**Percentages**
New possibilities from interplaying mechanics	1	1	1	1		1	1		1			7	100.00%
Not intentionally designed	1	1	1	1		1	1		1			7	100.00%
Lets players explore more ways	1	1	1	1		1	1		1			7	100.00%
Not an explicit rule	1	1	1	1		1	1		1			7	100.00%
Not specifically programmed into the system		1	1	1			1		1			5	71.43%
Must be	1					1						2	28.57%

unintended					▨			▨		▨	▨		

B.3 Participant Playstyle/Prioritisation

1 represents the participant's style of play at any point throughout the game's duration. Multiple playstyles could be adopted within the session.

Playstyle/ Prioritisation	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Sum**	**Percentages**
Explorer		1			1			1		1		4	36.36%
Exterminator	1											1	9.09%
Experimenter			1	1			1	1			1	5	45.45%
High Score Achiever (furthest level/score)						1	1					2	18.18%
Speedrunner	1	1		1		1	1	1				6	54.55%
Challenge Completionist		1		1	1				1	1		5	45.45%

B.4 Frequently used resources

1 represents a frequent use of the resource

Frequently used resources	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Sum**	**Percentages**
Basic Bow	1		1	1	1	1		1		1	1	8	72.73%
Great Bow		1					1					2	18.18%
Quick Bow		1							1			2	18.18%
Infinite Arrow	1	1	1	1	1	1	1	1	1	1	1	11	100.00%
Auto Arrow									1	1	1	3	27.27%
Great Arrow										1		1	9.09%
Bomb Arrow			1									1	9.09%
Platform Arrow						1						1	9.09%
Rope Arrow	1	1	1		1		1	1	1	1	1	9	81.82%

B.5 Bow Charge Style

1 represents the participant's preference of bow charging at any point throughout the game's duration. Multiple styles could be adopted within the session.

Bow Charge Style	1	2	3	4	5	6	7	8	9	10	11	Sum	Percentages
Aimed for perfect shots	1	1	1		1	1	1	1	1			8	72.73%
Rapid fire							1	1		1	1	4	36.36%
Held charge				1								1	9.09%

B.6 Participant Initial Skill Level

1 represents the participant's skill level.

Skill Level	1	2	3	4	5	6	7	8	9	10	11	Sum	Percentages
Low	1					1						2	18.18%
Mid					1			1	1		1	4	36.36%
High		1	1	1			1			1		5	45.45%

B.7 Participant Skill Level Change Over Time

1 represents if the participant's skill level change.

Skill Level Change	1	2	3	4	5	6	7	8	9	10	11	Sum	Percentages
Got worse												0	0.00%
Stayed the same		1	1			1	1	1	1	1		7	63.64%
Improved	1			1	1						1	4	36.36%

B.8 Participant Furthest Level

Each data set represents the highest level the user reached which are as follows:

- *1-0*
- *1-1*
- *2-0*
- *2-1*
- *3-0*
- *…*
- *3-Infinite*

Furthest Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Level Number	2-1	3-2	3-7	3-4	3-0	2-1	3-2	3-0	3-0	3-4	3-2

B.9 Participant Highest Score

Each data set represents the highest score the user reached.

Highest Score	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Score	2850	11600	23200	14700	8500	2550	10650	4600	4650	9050	7350

B.10 Participant Understanding of Level Generation

1 represents the level of understanding the participants had for the level generation.

PCG Understanding	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Sum**	**Percentages**
None												0	0.00%
Low	1	1		1	1				1			5	45.45%
Mid						1	1			1		3	27.27%
High			1					1			1	3	27.27%

B.11 Participants Perception of Emergence

1 represents the level of emergence the participants believed the prototype to be.

Perceived Emergence	1	2	3	4	5	6	7	8	9	10	11	Sum	Percentages
None												0	0.00%
Low				1	1		1			1	1	5	45.45%
Mid	1	1	1			1		1	1			6	54.55%
High												0	0.00%

B.12 Presence of unintended scenarios

1 represents the appearance of unintended scenarios for potential emergence.

Unintended scenario quantity	1	2	3	4	5	6	7	8	9	10	11	Sum	Percentages
0							1			1		2	18.18%
1	1		1		1	1		1	1		1	7	63.64%
2		1										1	9.09%
3+												0	0.00%

B.13 Participants on Factors that Increase Emergence

Each participant was asked about each of the factors for emergence and what factors actually contribute to more emergent gameplay.

- 1 represents an agreement that it does increase emergent gameplay.
- 0.5 is a slight agreement that the factor does increase emergent gameplay.
- 0 is a disagreement that the factor increases emergent gameplay.

Emergent Factor	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Sum**	**Percentages**
Increase interactions	1	1	1	1	1	1	1	1	1	1	1	11	100.00%
Reduce Constraints	1	0	1	0.5	1	0.5	0	0	1	0	0	5	45.45%
Add Subjects	1	0.5	1	0.5	1	1	1	1	0.5	1	1	9.5	86.36%
Instability	1	1	1	1	1	0.5	0	1	1	1	1	9.5	86.36%
Meaning	1	1	0.5	0.5	1	0.5	1	1	0.5	1	1	9	81.82%
Consistency	1	0.5	1	0.5	0.5	0	0.5	1	1	0	1	7	63.64%
Encourage Experiment	1	0.5	0.5	1	0.5	0.5	1	1	1	0	1	8	72.73%
Player Agency	1	0	1	1	1	1	0.5	0.5	1	1	1	9	81.82%

Appendix C - Emergent Scale Rubric

Emergent Scale Rubric (1/2)						
	0	**1**	**2**	**3**	**4**	**5**
Increasing Interactions	Interactions are kept to a minimum, only existing to progress the game.	Interactions are within the game but are very infrequent to have a more **constrained** design	Interactions are few within the game	Interactions may be involved in the solution/scenario or are used commonly throughout the game	Many different interactions within the game are made available to the player	Interactions are abundant throughout each gameplay scenario
Reducing Constraints	The game is constrained to the point where each player has an almost identical gameplay experience.	Many constraints and limitations are put on the player as per the game's design	Constraints exist, but multiple paths might still exist for the player to choose from (i.e. choose your path, correct and incorrect options)	Constraints are reduced from time to time to allow more player freedom over their actions	Constraints are used sparingly to allow for many emergent possibilities	Constraints are minimal to the point where the player may have freedom over everything (i.e. Creative or sandbox modes)
Adding Subjects	Subjects are kept to a minimum, only existing to progress the game.	Very few subjects are available in the game to add more emphasis on the few interactions involved	The number of subjects involved is limited and does not have much variation of interactions between them	There are a good number of subjects involved in the game/scenario	There are plenty of subjects present that the player can take advantage of for their emergent plans	A huge variety of subjects are present, with various interactions with the game world and other subjects that are commonly utilised
Instability	No sorts of randomness are involved	No sorts of randomness are involved (except for certain maths calculations and simple random range functions)	Randomness is only used far and few between (i.e. Changing the patterned solution of a puzzle each playthrough)	Some procedural elements may be involved such as random enemy or weapon spawn locations	More procedural elements are involved and made use of such as level generation and weapon stats	The game has many different combined randomised elements that lead to less predictable scenarios

Emergent Scale Rubric (2/2)						
	0	**1**	**2**	**3**	**4**	**5**
Meaning	The meaning of interactions was not considered (i.e. a byproduct of a bug, nonsensical mechanics)	Little meaning is put into the interactions.	Some meaning or common knowledge may be applied but it isn't a relying factor for the player to know	There are a few parallels between in-game mechanics and real life (or common knowledge areas). It is not unrealistic to expect a player to stumble upon a new interaction this way (**experimentation**)	There is enough meaning to the interactions in-game that the player can commonly apply real-life actions in-game and be rewarded with an expected outcome	There are a great number of parallels between in-game interactions and real life, almost providing a tutorial through **experimentation.** Meaning is also kept **consistent** for new interactions with common factors
Consistency	Rules are kept inconsistent as an element of **instability** (i.e. randomly generated rules/interactions)	Very few rules exist to maintain consistency.	Rules may be consistent but are commonly broken for the sake of certain missions or objectives (i.e. forcing the player to perform actions a certain way, like taking away their tools)	The rules are kept consistent for the most part, but still missing in certain areas (e.g. being able to set some but not all wooden things on fire)	The consistency of the game rules enables the player to **experiment** with new interactions and achieve an expected result, even if it happens to be their first time experiencing the interaction	The game world's rules are kept consistent enough that the player can create emergent plans and rely on the outcomes being consistent enough to follow through
Encourage Experimentation	No experimental tools are provided to the player	The game is highly linear, discouraging different forms of play	The game has options to play different ways (**agency)** but does little to incentivise different playstyles	Certain interactions are tutorialized and incentivised to show players how to use the mechanic. The hope is to have players take advantage of this knowledge to apply it in their scenarios	There is enough **consistency** and **meaning** present that the player can come up with a variety of different ways to approach a situation	Through great use of **consistency, meaning**, and having a high level of **agency**, the player can come up with plans that may be unique to what may have been expected by the developers (bugs or unintended emergence)
Agency	The design of the game is meant to be played in one way and only that way	The player may occasionally be given binary options for certain situations, however, most of the game is played with one solution in mind	The player is often given binary options to solve their situations	The player has access to a couple of different tools to approach their situation	The player has access to a great number of tools with different interactions to approach their situation	The player has access to a massive number of tools with different interactions to approach their situation. These are often scattered throughout a huge open world or located in different 'runs' in a roguelike