Master's thesis 2026

Level Design Guidelines Applied to Procedural Open-World Games

How to influence experiences in Open-World games through level design using procedural content generation

Raúl Martín

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
University of Gothenburg
Chalmers University of Technology
Gothenburg, Sweden 2026

Level Design Guidelines Applied to Procedural Open-World Games
How to influence experiences in Open-World games through level design using procedural content generation
Raúl Martín

Supervisor: Michael Heron, N2GDT
Examiner: Staffan Björk, IxD

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Level Design Guidelines Applied to Procedural Open-World Games
How to influence experiences in Open-World games through level design using procedural content generation
Raúl Martín
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Procedural Content Generation (PCG) is widely used in open-world games to address issues of scale and content creation. However, it is often driven primarily by technical considerations rather than experiential intent. This thesis investigates how player experience in open-world games can be influenced by informing procedural generation algorithms with level design principles and practices. The concepts of game experience, open-world games, level design, and procedural content generation, are used as the four foundational pillars of this work. This thesis explores the interconnections between level design and the rest of the pillars, and synthesizes the findings into a set of design guidelines focused on goals, engagement, guidance, and spatial communication. These guidelines are then translated into two practical prototypes that explore how experiential design principles can be embedded into game systems through level layouts, terrain generation, and the distribution of game elements. The research adopts an exploratory approach grounded in Research Through Design, positioning implementation as a means of probing feasibility rather than empirical validation. While the prototypes are not evaluated through player studies, they exemplify how abstract level design knowledge can be used to inform procedural systems that are informed by player experience considerations. This work contributes a conceptual framework and practical grounding for future investigations into experience-driven PCG in open-world games, highlighting both the potential and limitations of using these generative systems to shape game experiences.

Keywords: Level Design, PCG, Open Worlds, Gaming Experience, MDA, Guidelines

Acknowledgements

I want to thank my supervisor for supporting me and giving me great advice to approach and accomplish this monumental task.

I specially want to thank my family and my close friends with which I could share my excitement and frustration about the topics explored in this work.

Raúl Martín, Gothenburg, 2026-02-21

«Twill be a cold, dark, and very gentle place. And one day, it will make someone a goodly home.»

The Painter, *Dark Souls III (2016)*

Contents

1

Introduction

PCG is a highly researched topic in the field of computer graphics and in the industry of video games. It is often framed in a very technical perspective, especially in the terrain generation field, where its research focuses on acquiring high-performance and realistic-looking results. However, in the context of video games, level design plays a very important role in the game experience. This feature is usually disregarded by terrain generation algorithms, which are frequently used in open-world games. The purpose of this thesis was to run deep research to understand both level design and PCG, and the different ways they could be integrated to form an engaging game experience.

The research in this thesis can be summarised in a sentence. 'How can we influence game experiences in Open-World games through level design using procedural content generation?' This research is split in four pillars, each with several questions to answer.

- **Game Experiences**. How can we define what makes a game experience? How can we measure how engaging and interesting an experience is?
- **Open-World Games**. What games are we talking about? What features do they share? What games are excluded from this research and why?
- **Level Design**. Why is level design important? What effect does it have in the game experience? What are the different conceptual design tools professionals use when designing their game levels?
- **Procedural Content Generation**. What is PCG and when is it okay to use it? What are its benefits and limitations? What are the different methods we can use to integrate it with level design?

The specific work of this thesis consisted of answering all these questions via a deep research in the four topics, and then synthetizing and summarizing all the findings in such a way that the information is more digestible. Research on game experiences and open-world games tends to be less in-depth, as a few well-chosen definitions can suffice to provide satisfying answers to their related questions. By contrast, level design and PCG are supported by a broader and more mature body of research, covering everything from low-level technical implementation to considerations of player psychology. This is why the thesis uses the first two pillars as context to situate a common ground for their integration with level design and PCG.

The outcome of this thesis reflects in two artefacts, a theoretical and a practical one. The theoretical one consists of a consolidated list of level design guidelines, which serves as a glossary of many different rules that can be used to understand how different design choices affect the player's experience in a game. The practical artefact takes some of these guidelines, along with the gathered background knowledge about PCG, and uses two prototypes to show different techniques on how these rules can be implemented in an actual game. It is important to note that the prototypes are not a game per se, but just a dummy environment to show how these guidelines can be translated from natural language into code. These prototypes demonstrate the effect of how a set of simple rules can generate an interesting world for players to traverse and interact with.

The scope of this thesis reflects an intentional breadth. Rather than addressing a narrowly defined problem, this work aims to construct a foundational taxonomy of level design guidelines while it explores several considerations such as their implementation through procedural methods, or their influence on game experiences. While this breadth enables the creation of this foundational construct, the scope necessarily constrains the depth with which each individual guideline can be developed and evaluated. The result of this work prioritises integrative understanding of the four pillars over isolated solutions to specific scenarios. This frames the thesis as an exploratory mapping that serves as a basis for further research.

2

Background

The research of this thesis can be summarized in one sentence. Its purpose is to explore how to influence ***experience*** in ***open-world*** games through ***level design*** using ***procedural content generation***. This research can be split by these four differentiable topics, where each has a deep research background of their own. This thesis has a heavy usage of acronyms [p.57] and key concepts [p.58], so two sections are included at the end of this document to provide definitions for both.

2.1 Gaming Experiences

To understand the first pillar of research, we need to ask what constitutes an open-world ***experience***. Hughes (2023) [1] explains in his thesis how there are two different types of experiences to consider: overarching and contextually situated. On the one hand, he defines the overarching open-world experience as the freedom the player has to self-pace and auto-impose goals. On the other hand, the contextually-situated experiences are defined as the moment-to-moment events the player goes through in a playthrough. The overarching experience can also be seen as the collection of all the contextually-situated experiences. The framework proposed in his findings is based on motivation and goals, which are deeply looked at through several studies, but the results are inconclusive. However, good insights can be extrapolated from this research. Hughes treats both motivation and goals as separate things, where gameplay is driven not only by player intent (motivation), but goals also influence their actions. He does mention how one has a strong effect over the other.

However, I propose that motivation comes directly from the desire to achieve goals, and the distinction comes from whether these goals are self-imposed or presented by the game. This implies that the motivation behind any action the player takes comes from their *desires*. The different types of desires can be found in Figure 2.1.

In this graph we can see how the definition of experience we use goes as follows: an experience consists of the desire, the behaviour, and the emotional response a player has in the game. This mirrors closely the MDA framework [2], one of the most relevant pieces of research in game design, where a game can be analysed from the perspective of its mechanics, dynamics, and aesthetics, and how one affects the others. In our case: **goals** (enabled by mechanics) *motivate* a **behaviour** (dynamics), that later evokes **emotions** (aesthetics).

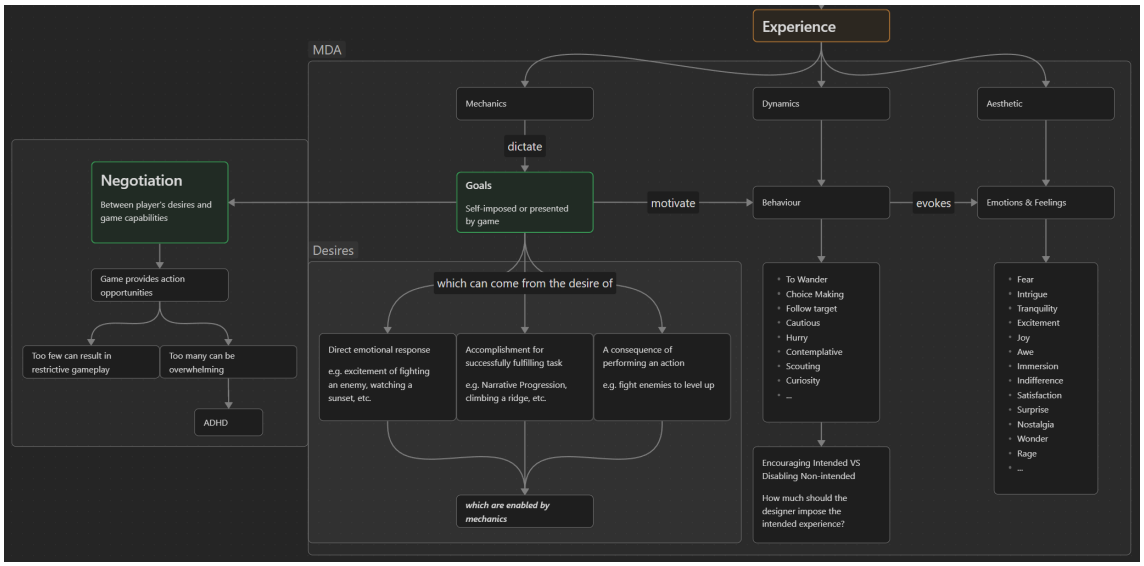

Figure 2.1: Experience concept diagram designed in *Obsidian*. It exemplifies how the MDA framework is structured, and how goals can be born from the player desires.

In his work, Hughes [1] explains how a gameplay experience consists of a constant 'negotiation' between the player and the game, where the player has a desire to perform certain actions, and the game responds with its available systems or goals. This is particularly important in open-world games, where the freedom to explore the world relies on a much richer and more profound aspect of *layering goals*. In this context, level design is the one in charge of distributing said goals to construct the intended experience. Here is where the connection between open-world games and level design lies. This research ignores what type of game systems enable the different types of goals that can be created and instead focuses on exploring how the distribution of these goals is presented to the player through level design.

2.2 Open-World Games

Open-world refers to a genre of video games where players can freely explore a large, usually continuous game space and decide for themselves what to do and when to do it. This is what we call *player agency*, where, instead of being pushed along a strictly linear sequence of levels or missions, the player can roam the world, ignore or postpone main objectives, pursue side activities, and often shape their own experience through *choice* of activities, playstyle, and priorities. The core idea is that the player can self-pace their gaming sessions. The narrative, the game progression, and the challenge are usually structured around this principle to support this player agency rather than dictate a fixed route.

To support this sense of freedom and exploration, open-world games tend to share several common mechanics and design elements. These elements can appear in many genres, but in open-world design they are usually arranged together [3].

- **Exploration and traversal**: Large maps, subdivided into multiple biomes or districts, and various traversal methods (e.g. on foot, vehicles, mounts, climbing, gliding, fast travel, etc.).
- **Non-linear objectives**: Different orders of importance for the goals: main story quests, side quests, activities, collectibles, and sometimes dynamic events. These can usually be approached in any order or even ignored by the player.
- **Progression and gameplay variance**: Experience systems, different types of gear, skill trees, and upgrades let players specialize (combat, stealth, crafting, exploration, etc.) and approach challenges in varied ways. Depending on what genre the game is there might be more or less freedom in this aspect. One such example is the RPG *Elden Ring (2022)*, where the build you choose drastically changes the way you approach combat, whereas in *Zelda BOTW (2017)* you can just upgrade your backpack size and your health and stamina.
- **Discovery-driven content**: Hidden locations, collectibles, environmental storytelling, and optional quests or encounters reward player *curiosity*.
- **Sandbox interactions**: These games often include many different tools which support multiple solutions to problems rather than a single correct approach. Some examples include physics, crafting, or building.
- **Systemic world simulation**: Some open-world games also include dynamic systems, such as day-night cycles, weather, NPC schedules, and ecosystem simulation (e.g. wildlife) to help the world feel alive beyond scripted sequences.

The open-world genre describes the structure of the game while also sharing these listed elements. However, it does not dictate its core activity or theme. Because of this, it acts as a layer that can sit on top of almost any primary genre. It is typically used as a qualifier (open-world RPG, open-world survival horror, open-world action-adventure...) rather than as a fully self-contained label, because the underlying playstyle is still defined by the primary genre.

This thesis aims to provide a set of level design guidelines that apply in any of these genres. This means that games like *Minecraft*, *Assassin's Creed*, *Firewatch*, *The Evil Within*, *Satisfactory*, *Elden Ring*, *Subnautica*, and *Goat Simulator* could benefit from the insights of this thesis, even when they are so different from each other. Not all of these games would benefit from an in-game procedurally generated world, since some of them have a special design considerations, like a strong narrative nature, but even then PCG could be used in the development phase to accelerate specific parts of the process.

It is important to remark that some genres are disregarded in this thesis, since their level design needs are quite distant from the genres listed before. These include genres like platforming or racing, which rely on highly controlled level layouts and fixed paths. These can coexist with the open-world genre, but the focus of the level design research in this thesis is not in how to space out platforms and make interesting circuits to drive. Level design in this thesis refers to the overarching structures and the cognitive influence different rules have over the player's intent.

2.3 Level Design

The foundation of level design consists of **guiding** players through space using **layouts** and visual cues to support gameplay, pacing, and narrative [4]. It can be defined as the process of planning, structuring, and implementing the spaces in which a game takes place. A level designer's role is to assemble the core gameplay elements in such a way that it encourages the intended player experience [5].

In the contemporary game development industry, level design is a crucial role due to its interdisciplinary nature [6]. It holds a deep connection with every other specialization. Game design defines the core mechanics, systems, and progression (along with narrative design). Game art first delivers thematic context and provides the building blocks of the scenery and then it gets constrained by the layouts provided. Technical roles need to discuss with level designers the capabilities and limitations of the systems being developed. In this matter, the level designer often operates as an architect, integrating all these elements together within coherent, navigable, and readable spaces [7].

Historically, level design emerged as a more informal practice tightly related to game design. It was usually carried out by programmers, who manually built stages using simple tools or even raw code [8]. The earliest games that featured modern properties associated with level design include *Super Mario Bros (1985)*, *The Legend of Zelda (1986)*, and *Metroid (1986)* as the most famous examples. Their rules were strongly dictated by hardware limitations and simple rule sets, but still they laid the foundation for principles such as difficulty curves, readability, spatial progression, exploration, visual cues, etc. Foundational examples are harder to enumerate, since they define a more blurry outline of the current practices associated with level design. Some of these games include *Colossal Cave Adventure (1976)*, *Adventure (1979)*, and *Pitfall! (1982)*. As games increased in complexity and production values, level design gradually evolved into a dedicated discipline with specialized tools, pipelines, and methodologies [9]. The result is that level design today is recognized as a core feature that significantly influences both player engagement and the overall experience of a game [10].

The impact and importance of level design has become especially pronounced in open-world games, where the player is granted a high degree of agency and control over pacing and goal selection. The is a strong difference with how level design works in games with a linear nature. Linear games provide more control to the designer in the sense of pacing, pathing, and all the factors that affect engagement. The effect of how much control the designer has over these factors is debatable, since a player deciding what to do usually grants them a more authored experience, but they can also get lost and choose to play in a way that stops being engaging after a while. In open-world games, designers must consider how exploration is motivated and account for different layers of goals at any time. This translates in a different approach, where they have to think about how Points Of Interest (POIs) are distributed across the world, and how players are subtly guided without overtly constraining them. Level design in open worlds involves crafting hierarchical structures (regions, hubs, landmarks, etc.) that support orientation and navigation,

while also accounting for localized gameplay instances that maintain variety and pacing. The design sometimes must also account for multiple playstyles, emergent interactions between systems, and the need to sustain engagement over extended play sessions. Hence, while level design in open worlds has a distinct nature, its quality is still crucial to provide engaging player experiences.

It is important to note that this thesis focuses on level design as in the context of world structure and integrating mechanics to convey the intended gaming experience. Level design can sometimes be used as an umbrella term to also refer to level art, which informs the placement of assets, lights, and other level elements to provide a specific aesthetic. Level designers should work hand in hand with level artists, but their roles should not be confused. Visual dressing and stylistic choices are acknowledged as influential in terms of player guidance, but fall outside the main scope of this work. This allows for a more focused investigation into the principles and practices that dictate level design, particularly within the context of open-world games.

2.4 Procedural Content Generation

Procedural Content Generation (PCG) consists of the usage of algorithms and randomness to automatically create game content [11]. This means that these game elements are generated by code and Random Number Generators (RNGs), as opposed to handcrafted content. Its purpose is to produce content that is both varied and scalable in a dynamic way, reducing the production cost. Some examples of content that can be generated procedurally include level layouts, quests, loot systems, character appearances, and terrain, being the latter the most heavily researched topic in the literature. Some examples of PCG methods typically used in video games include Perlin Noise, Wave Function Collapse, Grammars, and Random Walk, among many others.

2.4.1 Benefits and Drawbacks

The main benefit PCG provides is replayability, or infinite play. This is because with procedural levels or scenarios we can have an experientially infinite set of content or outcomes [12]. This is the case of *Noita (2019)*, a roguelike video game where a system of caves and the selection of items and enemies found in it are generated procedurally. Another benefit is adaptability, where the uncertainty of events leads to a more dynamic playstyle in which players have to constantly react to new scenarios [13]. Furthermore, some game elements can be tweaked by the PCG algorithms to adjust pacing and progression dynamically so that levels get more intricate and difficult over time, while at the same time they get balanced with the skill the player presents. Unique experiences can also be experimented, this referring to one player having an experience that no other player has had before. This is the case of exploration in *Minecraft (2011)*, where the idea of discovering never-before-seen lands can provide a very specific sensation that is impossible to recreate with handcrafted content [14].

However, PCG also presents some issues mainly related to the complexity of the implementation of algorithms and the 'interestingness' of the results. This complexity can be born from specific game needs, such as the requirement of existence of a *critical path* i.e. a valid path for the player to achieve its goals. For some algorithms it can be really complex to determine if this exists, even more if the game requires to only have a single solution, such as many puzzle games. Measuring 'interestingness' is another problematic task due to its subjective nature. One example of a missing interesting game element when using PCG is *noteworthy gameplay instances*. These are incredibly difficult to provide since they are usually meticulously crafted by hand and are most likely to be tied to narrative e.g. going back to your home town after defeating the main antagonist only to realize it has been invaded by goblins. At the time of writing this text, PCG algorithms and generative Artificial Intelligence (AI) struggle at generating compelling stories and lore that can have a meaningful impact on people [13]. The creation and relevance of this **meaning** is explored in the guidelines and discussion [Section 5.1.4, Section 6.1].

2.4.2 Historical Context

In recent decades, PCG has undergone a significant evolution within the field of video game development. Early uses of PCG focused primarily on simple randomisation to extend replayability or reduce memory usage (*.kkrieger (2004)*), such as generating basic dungeon layouts or random enemy placements. Today, however, the most advanced implementations of PCG are deeply integrated into the core of games. One such example is terrain generation, which has reached a high level of precision and efficiency, employing noise functions, layered heightmaps, and sophisticated erosion simulations to produce believable landscapes at scale. Another specific example that is gaining popularity lately is voxel engines. These enable the creation of highly detailed, fully mutable game worlds, allowing both developers and players to manipulate the environment in real time, as is the case of *Hytale (EA 2026)*. Voxel renderers have existed for a long time, being one of the earlier examples *Comanche Maximum Overkill (1992)*. They have later been widely popularised with *Minecraft (2011)*. Recently, with the advancements of technology and research, people are achieving finer results with higher resolution while still conserving the mutable property of the worlds. Such is the case of *Teardown (2022)*, where PCG can be applied to generate dynamic environments and highly interactable worlds.

Nowadays, PCG is also used in game development in the process of creating finite worlds by automatically distributing assets, 'painting' large sceneries with trees, houses, and other game elements. PCG techniques also extend to the automatic creation of game assets, such as NPCs, items stats, buildings, and even entire cities with level layouts. Using this technology as a creation tool can produce a variety of visually or semantically coherent results by combining modular elements of handcrafted content. These include clothing, body parts, and animations for characters; different architectural elements for buildings; and pre-designed rooms as building blocks for larger level design contexts. PCG is no longer limited to simple randomization but is instead a powerful design tool that supports game content creation and reduces the manual workload on development teams.

2.4.3 Taxonomy

We want to learn about many different procedural content generation methods so that we can understand which ones serve us for implementing in this thesis. A taxonomy in this context is useful because it helps analyse the specific characteristics we desire in our algorithm [15]. This taxonomy presents eight different ways to categorise PCG algorithms [16]:

- **Input Required**. What type of input the algorithm requires to run. It might require no input, user/designer input, or training data.
- **Data Representation**. Data output can be represented in many different ways: as textures, as splines, as vectors or graphs, as voxels, as rules, etc.
- **Controllability**. This defines how much control we have to alter the results of the generator. We can have no controllability (only seed-based), suggested control (e.g. parameters), or precise control (e.g. sketch-based). This category can be split up depending on if we are talking about user or designer controllability. Furthermore, we can also differentiate if this control is applied before running the algorithm or during its execution.
- **Evaluation of Success**. A successful result can sometimes be identified automatically, but some other times we require user-evaluated verification. Automatic evaluation of success is especially useful in algorithms that require parameter tuning.
- **Complexity**. We can measure complexity based on the number of features and layers our generator has. We can also identify characteristics such as testability or ease for implementing intricate rules.
- **Computational Cost**. We can measure memory usage or execution time, and evaluate which one is a stricter constraint for our application. We can have offline or runtime algorithms, which particularly influences the importance of execution time.
- **Determinism**. Depending on how we construct our algorithms, the implementation of a seed can provide deterministic results. If we do not require this functionality we can be more relaxed on our constraints.
- **Reusability**. An algorithm that can be tuned to many different purposes is considered reusable. This can be the case for a flexible world generator, where you can alter the frequency and intensity of specific features, such as mountains, water bodies, and biomes.

Some other characteristics are more closely related to the paradigm or purpose of the algorithm. This can be the case of chunk-ability in world generation, where the generation can be split up in small plots of land instead of all at once, or traverse-ability, which dictates if there is a path for the player to walk from place to place.

2.4.4 Methods

This is a specific type of categorisation where we distinguish algorithms depending on their paradigm [11][17][18]. Bear in mind the following categories are not mutually exclusive, but algorithms might belong to more than one of these. There are four major groups, which can be further divided into more specific categories:

- **Stochastic**. These are algorithms that try to imitate natural randomness. They use rules and parameters, but lack precise control over resulting features.
  - *Fractal*. These refer to algorithms whose rules are defined recursively. Some examples include layering noise textures like Perlin Noise, or implementing the Random Mid-Point Displacement algorithm.
  - *Grammars*. These are a specific type of fractal algorithms which define elements and rules like grammars do. These then converge into terminal symbols that represent the result of a generation.
  - *Tiling*. These refer to algorithms that subdivide the space. They can apply different rules at different depths. An example algorithm is Binary Space Partitioning (BSP).
  - *Parametrization*. These algorithms define parameters that can be used to adjust the behaviour of the generator. They are most commonly applied in Distribution Sampling, to define frequencies and probabilities of elements being selected.
- **Constraint-based**. These are algorithms whose generation is determined by a strict set of rules that has to be solved for in order to generate valid states. One specific example is Wave Function Collapse (WFC), which defines a set of tiles and rules on how they can be combined together.
  - *Constructionist*. These refer to randomly stitching modules to generate a variety of possible sequences. They are typical in dungeon generators for *roguelike* games. These modules may be generated by hand or by another PCG algorithm.
  - *Cellular Automata*. This is a specific category that works within a grid space and has rules to define what the next step of generation is based on its current state. The most famous rule set is the one from Conway's *Game of Life (1970)*. This category could also belong to **Simulation**.
- **Simulation**. Algorithms in this category try to recreate natural world occurrences. They have some initial parameters and state, which is then modified over time. Usually, the designer cannot intervene in the generation process until the final result is provided. They also tend to be slow and lack precise control over features.
  - *Geomorphological*. These simulate terrain generation occurrences, like river erosion, mountain ranges creation with tectonic plates, or mineral deposits stacking up.

  - *Ecosystem*. These refer to the evolution of the fauna or flora, or even the effect they have over the environment. They can dictate which populations of entities persist, or how the environment is modified by their presence, by changing the state of the soil or tracing paths. This category can also include human behaviour, where they can drastically modify their surroundings by creating towns and roads.
  - *Agent-based*. These refer to effectors that do not base their behaviour on geomorphological or ecosystem events, but they are straightforward modifiers that can be applied randomly. Some example effectors include rising, flattening, or smoothing surfaces, texturing effectors, or river effectors that connect points on two distinct bodies of water.

- **Learning**. These algorithms process provided data, usually in an iterative way, to replicate their intrinsic patterns. The quality of the provided data here is crucial to have a successful generation.
  - *Example-based*. Here usually real data is used as atomic modules that can be stitched together with other generation methods. The properties of these examples are used to replicate their looks while combining them all. This is similar to the constructionist approach.
  - *Search-based*. These algorithms are based on a fitness function they use to evaluate results and iterate on by modifying them slightly until a local minimum is found. The fitness function has to be defined around the properties of the data we want to replicate. This is typical of Evolutionary Algorithms, where different chromosomes define possible solutions, and they compete with each other until the most fit survives.
  - *Inverse Procedural Generation*. This consists of the extraction of parameters from the data. This could be real data or a provided handcrafted sample. One example could be creating a flora painting brush given the distribution of plants and trees in a plot of land.

- **Sketch-based**. This category refers to the PCG algorithms that act as tools designers can utilize to populate levels. They are the kind of generators with most precise control over the generated features. The name comes from the specific tools that can receive drawings as input to use as templates.
  - *2D*. These can be textures that define water bodies or mountain ranges in the world, or probability maps to spawn enemies or Points Of Interest.
  - *3D*. Although these are more complex to work with, they provide flexibility to the user to define characteristics of their games in a 3D space. Some common tools are splines or 3D effectors that specify how game elements should be spatially generated in specific parts of the level.

Some specific examples of procedural generation techniques include: layering noise, graph generation and traversal, irregular grids, and usage of mathematical distributions. Specific named algorithms include: WFC, Marching Cubes, Random Walk, Distribution Sampling, BSP, Delaunay Triangulation, Markov Chains, etc.

2.4.5 My Contribution

This thesis investigates how level design rules can be embedded directly into a game through the usage of different types of procedural generators, with a specific focus on level layouts for open-world games. Rather than treating PCG as a purely technical system that generates arbitrary content, the focus is on encoding design knowledge (e.g. constraints on navigability, pacing, difficulty curves, and visual language) into the generative process itself. By integrating level design rules, generators can produce content that is not only structurally valid but also aligned with the **intended player experience**. This involves translating abstract design principles from natural language into formal constraints and heuristics that can guide generation in all these ambits. The overarching goal of the thesis is to bridge the gap between the cognitive and often intuitive practice of level design, and the systematic nature of procedural algorithms.

A conceptual distinction to be made in this thesis is between what can be referred to as technically procedural and meaningfully procedural content. On the one hand, technically procedural content is any content generated by algorithms using some form of randomness or rule-based logic. For example, assigning enemies random amounts of damage values or generating loot with randomised statistics falls under this category. Such uses of randomness often have limited impact on the players qualitative experience of the game. While they may increase variability, they will not be fundamentally changing how the game is played or perceived. This was the usual case in early instances of PCG in video games, but it still has a strong presence in the modern industry. On the other hand, meaningfully procedural content refers to content whose generation has a deliberate, perceivable, and coherent effect on the overall experience or interpretation of the game. These are usually more complex to implement and affect aspects of a game such as encouraging exploration, supporting game pacing, or reinforcing particular design goals. With this distinction, we can set the research background focus in the latter type of Procedural Content Generation. The integration of **open-world level design** principles into these **generators** leads to following the intended design **experience** from a programmatic perspective, covering all four pillars of research.

2.5 Guidelines VS Design Patterns

In game development research, design patterns have been an influential tool for describing and analysing recurring solutions to common design problems [19]. These patterns are valuable because they offer a shared vocabulary for developers and researchers, making it easier to identify and compare design elements across different games. However, despite their usefulness, design patterns tend to remain relatively modular units of knowledge. Each pattern typically encapsulates a specific problem-solution pair, which can lead to a fragmented understanding of complex domains such as level design [20], where many interdependent factors must be considered simultaneously.

Design patterns are frequently compiled into large catalogues, which can become collections of loosely related entries rather than a tightly integrated body of knowledge. For a topic as intricate and layered as level design, this can make it difficult for designers and researchers to grasp the larger conceptual structure. One particular example of this is observed in the industry professional Peter Field's talk about spatial communication [4]. This talk goes over many different design patterns and tricks that give great insights about what the player experience might be when presented with different visual cues and layouts. However, all these tips feel somewhat disconnected from each other, and it can be overwhelming to try to apply all of them at the same time when designing your own levels. Among these patterns he explains concepts like clear goals, sightlines, several pathing tools, vistas, affordances, and even how to influence the player's curiosity with artistic features. These are all really useful concepts that ended up integrated in the guidelines, but in this thesis they are presented in a more structured and thematically cohesive narration. Here, guidelines do not replace the insights of design patterns, but rather build on them within a more educationally oriented framework.

On a more general note, many professionals in the industry have shared similar contributions before [21]–[23], providing their own practices and insights of their approach and their experience in many different contexts, from indie games to AAA industries. These resources have been essential to build the knowledge base for the level design pillar in this thesis, but still, most of them lacked cohesion in all the concepts presented. This has been a running issue throughout the literature, where design principles, practices, and considerations where presented alike, as if they provided the same kind of tools to the reader. They all also state how these insights are never sufficient to do proper level design, and that designers require a kind of tacit knowledge that is only acquired through practice. In addition to all of this, none of them explicitly mentioned the close relation level design has with game design itself, and how many decisions depend on a close discussion with the development team.

The work of this thesis aspires to deliver the **foundation** of a holistic and practically actionable body of knowledge. In this text, guidelines are presented following a textbook-like structure. They are not only structured into chapters, sections, and thematic clusters that progressively build an understanding of level design, but they are also separated from designer considerations and common practices, which belong to additional sections. Furthermore, this text provides a practical pipeline that explains in which order to take guidelines in consideration throughout the lifetime of a project. Additionally, unlike individual patterns, each guideline establishes its relationship with others, uses metrics, and outlines real industry examples to provide a complete context on what these rules are, when they should be applied, and why.

This way of structuring all the knowledge allows the material to support both learning and practice. Newcomers can use it as an educational resource that supports their learning process, while experienced designers can use it as a reference that they can check to refresh key concepts. The aim is not just to enumerate known solutions, but to offer a structured approach for reasoning about levels, from high-level design decisions down to moment-to-moment player experiences.

2.6 Why this research matters

First and foremost, the importance of this thesis relies on the relevance of the research question itself. We want to understand how level design can influence the game experience in an open-world game, and then apply it algorithmically through PCG. There are several links here: how game experiences and open worlds connect, how level design can influence these experiences, and how to implement level design with procedural generation.

To start off, we can use the definition of an open-world experience provided in Section 2.1. It explains how a experience is constructed upon the MDA framework, and how the mechanics dictate what are the possible goals. These goals can also be born from the desires the players have, and the discussion players have with the game to understand what are possible interactions is called goal negotiation. Level design is in charge of laying out these goals as opportunities so that the players engage with them. This is specially tricky in open spaces, since designers have to be more lenient on what the intended experience can be due to players choosing what to engage with at any point. On top of that, level design can be understood as the **guidance** of players through layouts. Non-linear level design is usually explored from the perspective of closed spaces with branching, and open spaces are left aside as simple empty playgrounds to fill up with scattered points of interest. This thesis suggests we can take the principles of linear level design and extrapolate them to a open space context.

The connection between level design and procedural generation is a fresh topic in research, but there is some evidence of people trying their hands on implementing it [13]. The approach of this thesis relies on formalizing the principles of level design and translating this natural language into algorithmic rules to follow. This thesis focuses on emphasizing the importance of this connection, rather than explaining in depth how this implementation could happen. Still, a practical example is presented in the Results [Section 5.2]. Integrating procedural generation with level design opens up a discussion topic about the interestingness of the results due to a possible lack of *meaning*. This is further explored in the discussion [Section 6.1].

2.7 Level Design and PCG Integration

When thinking about integrating PCG with level design there are two distinct parts. We can integrate it with hand-crafted content or with level design principles. Both are crucial to consider to have successful results, although the most interesting topic is the latter, which is the one integration that lacks more research in the literature and which this thesis tries to further explore.

The integration of PCG with hand-crafted content is straightforward. M. Johnson [13] explains two different approaches: horizontal and vertical integration. In horizontal integration, the hand-crafted content follows a modular nature [24]. Here, each piece of generated content present in the game can be replaced by hand-crafted one. This approach facilitates the implementation of set pieces, which directly fights

the struggle of PCG to create *noteworthy gameplay instances*. One example of this type of integration can be a dungeon generator where specific designed rooms provide story beats or carefully crafted experiences. A consideration of this approach is whether designers should blur the edges between both types of content or if they should make sure these set pieces are striking enough and not missable [25]. On the contrary, with vertical integration, instead of having scattered elements of handcrafted content all around the game, the procedural content is constructed around the former. This means that the game will always present the main story line or provide a set of goals, and then randomness is applied on top of that to add replayability. A real industry example is *Spelunky (2008)*, where a trail of specific actions, enemies, and items can be followed to reveal a new secret final area.

The integration of level design principles is substantially more complex [26]. This thesis proposes several level design guidelines, and the procedural implementation of each of them follows a completely distinct approach, most of which are just proposed theoretically. However, we can still have a look at some techniques proposed in the literature. J. Segree [27] explains how to use PCG in puzzle games following different methods. My approach builds on this by formally defining goals, rewards, and metrics to construct a method of generation that satisfies some common level design principles such as **pacing** and **Risk VS Reward**. Some useful metrics to define include difficulty, mechanics involved, time to solve, number of actions required, etc. Although this serves the purpose of automated categorization, it might lack designer intentionality if it is not specifically encoded in the generation. It might be harder to present new mechanics to players in the same way that tutorials are designed, and there might be a lack of cohesion since every task is presented in isolation. We can still apply some techniques proper of puzzle games like **batching**, which consists on combining small pieces of puzzles to create larger ones. This concept is later referred as *Gardens* in the level design guidelines. Other useful techniques are **incremental change** (adding constraints in stages with decreasing impact), or **mechanic checking** (forcing a situation where a mechanic is required so that the player can show their understanding of it).

In procedural generation, the approach of implementation can follow two distinct paradigms. The top-down paradigm explains how we first have an example of what the result is and then construct a generator that produces similar outputs, while the bottom-up paradigm first comes up with the algorithm and then tunes it for the most interesting results. The approach to implement the integration of PCG with level design principles following puzzle design concepts is similar. The concepts here are named solution-down and puzzle-up, respectively. The first one proposes a solution and then adds constraints around it to justify that solution. The second one generates a list of random constraints and then runs a solver to check if a solution exists. A mixed strategy can also be implemented which first generates a solution and then when adding the constraints it might modify slightly the solution to better fit the intended output. For level design we have to be specifically careful with some considerations, like the existence of a *critical path*, multiple solutions, undesirable results, and be aware of spatial constraints and the balance between chaos and lack of variety.

3

Theory

Since the scope of this thesis is quite ambitious and it includes a deep research of many topics, along with the implementation of both theoretical and practical artefacts, several theories are required for different purposes. Some of them are used to frame the research problem, while others are used to explain the structure or content of the knowledge, or to justify the approach of research. Here is an outline of this chapter, which is divided in 5 sections, each with their respective theories.

1. **Research Approach**. This section explains the nature of the research problem through *Wicked Problems* [28], and how the research approach is based on the *Exploratory Research* framework [29].

2. **Player Experience**. This section briefly explains how the *MDA Framework* works and its importance to construct a game experience. After that, it explains how the guidelines of **Goals** and **Engagement** are based on the *Self-Determination theory* [30].

3. **PCG**. Here, the *Systems Theory* [31] and the *Emergence Theory* [32], explain the nature and relevance of PCG. They also explain how this thesis proposes the integration of PCG with Level Design.

4. **Guidelines**. Theories in this section are used to explain the importance of a shared vocabulary in design practice through the use of the *Sapir-Whorf hypothesis* [33] and *Pattern Languages* [34]. Furthermore, it explains the importance of *Affordances* [35] in the context of Level Design.

5. **Prototypes**. Finally, this section explains the relevance of the implementation of the practical prototypes through the theories of *Research Through Design* [36] and *Constructivism* [37].

3.1 Research Approach

Wicked problems [28] are characterised by ambiguous boundaries, multiple stakeholders with conflicting values, and evolving constraints that change as one attempts to intervene. This type of problems are difficult to define or solve since they is strongly interconnected with the context they is situated in, and they often have a subjective nature. This is the case of our research question, where we are trying to evaluate how to influence player experience (an intrinsically subjective matter) through different tools such as level design and PCG. These experiences are strongly context dependent, since they rely on the demographic they are aiming to influence. Framing the problem as a wicked problem justifies the need for a flexible, iterative research strategy rather than a rigid, hypothesis-testing design.

The flexible approach in this thesis started by exploring the research field of both open-world experiences and level design in a broad sense, and then narrowing it down by finding commonalities and selecting the most interesting and related questions. This helped define a more concise research question. However, since wicked problems are context dependent, we need to clearly define the scope to narrow what type of answer we are looking for. In the context of this thesis, this part of the work consisted of carefully defining which specific kind of open-world game experiences we want to explore, as seen in Section 2.2.

The specific approach of this thesis consisted on an *exploratory research* [29]. These are typically used when the phenomenon under study is not yet well understood, theoretical foundations are incomplete, or prior empirical evidence is scarce. This is the case of level design in the context of open-world games. Rather than testing predefined hypotheses, exploratory studies aim to generate insights and uncover patterns that can inform future, more structured investigations. This approach allows the research to iteratively refine questions and potential solution directions as our understanding deepens, working along the theory of wicked problems. Together, the wicked-problem perspective and the exploratory research approach support the idea that the objective of this study is not merely to test an existing theory, but to construct and refine understanding in a complex and only partially understood problem space.

3.2 Player Experience

As explored in the background (Section 2.1), a game experience can be defined through the *MDA Framework* [2]. This framework is widely referenced in the level design literature, and it explains how a game experience is constituted of three parts: the mechanics or systems that define the rules of the game; the dynamics, which refer to the interactions and the emergent behaviour these mechanics allow the player to have; and the aesthetics, which relate to the subjective experience the player has when interacting with the media. This thesis uses this framework as a base for further researching what role level design plays in game experiences and what tools we can use to influence them.

This thesis states in Section 2.1 how the mechanics of a game serve the purpose to construct **goals** or objectives, and how these relate to the **desires** the player has to perform certain actions. Then, level design is in charge of laying out these goals to create a conversation with the player called **goal negotiation**, where the player comes up with desires and the game responds with its available systems. This knowledge about game experiences is collected and presented as the guidelines of **goals** and **engagement**, which explore concepts like *player agency*, *meaning*, *choices*, and *challenge*.

All these concepts are strongly tied to the *Self-determination theory* [30]. This is another framework that explains human behaviour through the satisfaction of three basic psychological needs: *autonomy*, *competence*, and *relatedness*. In the context of games, autonomy refers to the players perceived agency and freedom to make meaningful choices; competence relates to the players sense of mastery, challenge, and progression; and relatedness concerns the experience of meaning, coherence, and connection to the game world. This theory is relevant in the sense that when we design a game, we do it so that the player has a specific experience, and to achieve it, they have to follow paths or interact with the game in specific pre-considered ways, even when the target experience is based on *emergent gameplay*. The self-determination theory provides tools we can use to analyse games and to understand how to construct them.

3.3 Procedural Content Generation

Procedural Content Generation in games is commonly understood through the lens of *systems theory* [31], in which content is produced by the interaction of rules, parameters, and constraints rather than direct authored control. From this perspective, a PCG system is a structured set of interdependent components whose behaviour cannot be fully understood by examining individual elements in isolation.

A core concept within this framework is *emergence theory* [32], which in the context of video games can be tied to emergent systems or gameplay. Here, emergent behaviour refers to patterns or structures that arise from the interaction of simple rules, producing outcomes that are not explicitly specified by the designer. In PCG, this means that level layouts and gameplay interactions are generated indirectly, making the resulting content partially unpredictable yet constrained by the design of the rules. Emergent gameplay is also predominant in open worlds, since the scale and non-linearity of the space, along with the player agency, disallow for strict designer control. Instead of creating specific experiences, designers influence the space of possible outcomes by shaping generative rules and constraints. Player experience is therefore designed through influence guidance, instead of strict pathing sequences.

It is important to remark that here PCG is not treated as an automation of level design, but as a design medium that shifts control from hand-crafted content to the construction of generative systems. Designers can use these systems to produce certain experience opportunities while accepting *variability* and *unpredictability* as inherent properties, which are important factors of **engagement** (Section 5.1.4).

3.4 Guidelines

The creation of the guidelines as a theoretical artefact of this thesis is based on the *Pattern Languages* [34] theory, but also strongly inspired by the *Sapir-Whorf hypothesis* [33].

A pattern language is a structured set of named, reusable solutions to recurring problems in a given context. Each pattern labels a problem-solution pair and relates it to other patterns, forming a network of concepts that designers can draw upon. In design practice, such a language functions as a common conceptual toolkit that designers can use to coordinate and communicate complex ideas with each other. In this study, guidelines are based on design patterns, but they take an extra step by adding context on how they can be applied in different situations, providing metrics, and giving real industry examples.

The Sapir-Whorf hypothesis proposes that 'the language that we speak shapes the way we think' [33]. While this thesis does not formally apply or test this hypothesis, it is used to explain how a level design vocabulary can serve as a toolkit for the designer with which to reason about common design problems and solutions.

A specific theory that is widely common in design, and explicitly stated in level design is *affordances*. Popularized by Norman's *The Design of Everyday Things* [35], affordances explain how the appearance of objects should hint on their purpose or way of interaction. In level design, affordances become a primary channel of designer-to-player communication: they shape what players immediately perceive as possible or expected actions in a space (e.g. which surfaces can be climbed, which paths can be traversed, or which objects can be interacted with) without requiring explicit instructions. Affordances can also be transcribed from everyday life interactions to video games, but working against them can result in breaking the player's immersion. Such can be the case of an ajar door not being interactable.

Having a common set of terms for patterns and affordances in level design does not merely ease communication, but can shape how designers perceive design spaces, recognize problems, and construct solutions.

3.5 Prototypes

An important framework to tackle wicked problems is *Research through Design (RtD)* [36]. Rather than treating design artefacts as mere outcomes, RtD states that the act of designing and implementing artefacts serves as a method of knowledge gathering. This knowledge is created through the iterative process of creation, evaluation, and refinement of artefacts, allowing researchers to explore complex problems that cannot be fully understood through analytical reasoning alone. This is the case of level design, which has an intrinsic tacit knowledge associated. As many professionals in the industry put it, one can only truly learn through practice.

This approach is closely aligned with the *Constructivism* theory [37], which states that knowledge is actively constructed through experience rather than passively discovered. From this perspective, understanding how procedural level design influences player experience requires direct engagement with the design process itself. Implementing prototypes enables the exploration of how theoretical design guidelines, generative rules, and constraints translate into concrete spatial structures and playable scenarios. The prototypes therefore function as tools that support learning and prove the theory behind them.

Together, RtD and constructivism justify the implementation of the practical prototypes as an integral component of the research process. The prototypes are not presented as finalized solutions or empirically validated systems, but as exploratory pieces of work that demonstrate how the conceptual design guidelines can be translated into procedural systems. Their implementation simply serves to investigate the feasibility of this translation and to expose technical or conceptual challenges and limitations.

4

Methods

This chapter goes through the processes and design choices applied in the thesis. In particular, it explains what the research approach consisted of, and the reasoning behind it. This chapter also presents the roadmap that this whole project followed, and the outline of how both the theoretical and practical artefacts were conceived.

4.1 The Research Process

The main methodology used in the research process of this thesis is the *scoping review*. Unlike other more traditional methods like *systematic reviews*, the scoping review helps exploring the breadth of available research to summarize scattered knowledge [38]. The nature of open-world games is analysed in this research through many different perspectives, but little resources can be found that combine them to have a more broad knowledge of how it all works together. Some of this different research fronts include open-world experiences, level design, environment art (and its connection with *psychogeography*), mechanics, pacing, difficulty scaling, guidance, and the exploration of what makes a space interesting. The usage of a scoping review helps this research collect many different factors that enable a designer create a compelling open-world experience. This is necessary to later design a procedural algorithm that uses this knowledge to try mirror the handcrafting effort designers put when creating their worlds. The scoping review method also helps in finding gaps in the research field to later answer in the study.

The literature review of this thesis is based on many different sources, like articles, academic papers, theses, case studies, developer insights, interviews, and Game Developer Conference (GDC) talks, among others. This variety in knowledge sources represent why a scoping review is necessary, since there is no unified set of knowledge that can be applied to level design, specially in the context of open-world games. Throughout the literature review we can find how this is true, since every designer talks about their experience, and their design tools do not usually match. It is important to note that specifically for the research of level design and open-world games experiences there are not many academic resources, as they pertain to tacit knowledge and professionals' experiences. This is not necessarily bad, but it explains the less formal and more subjective nature of the findings. This is not the case for PCG, as it is presented from a more technical and practical perspective.

The research process was divided into 3 milestones: the literature research, the field research and design, and the implementation. The first milestone consisted of going through the published material related to the 4 pillars of this thesis. The second milestone focused on relating all this gathered knowledge to real game examples to get some insights on the presence of these concepts. This implied playing games like *Subnautica (2018)*, an open-world exploration game with narrative on the side, and *Death Stranding (2019)*, a strong narrative in a large, procedurally crafted world. The final milestone collects all the findings to implement the artefacts and prove their feasibility.

In particular, the literature research started by doing a broad search on topics related to the four pillars of this thesis. For every piece of research, I would summarise its contents, define key words, and link them in a diagram to build a network of knowledge. With this diagram structure, it would be easier to keep track of the scoping review and figure out where research gaps lie, which were particularly easy to point out in between each pair of pillars. In terms of content, the research process started by finding out what makes a gaming experience and how these relate to open-world games by defining what actually is a game in this genre. Then, the research continued by finding publications about tips and insights on several kinds of level design, which could be linked back to both open worlds and game experiences. The research of PCG was much simpler, since most of its insights can be formally defined, such as the benefits and drawbacks of using it against hand-crafted content, or its taxonomy and example methods. The most interesting parts of PCG are the implementation considerations and approach, which can be connected back to level design.

The literature research came to an end when new sources did not provide new key concepts to add to the diagrams. At this step, the gaps in the literature were clearly visible in the diagrams. These are the problems I focus on in my own contribution and explore in the following chapters.

4.2 Structure of the Guidelines

This thesis has a strong focus on generic level design research. This is due to the need to define formal rules so that level design principles can be more easily translated into procedural algorithms. Before starting to write down the guidelines, it was important to reflect on what their purpose was. They are supposed to serve as an actionable body of knowledge that can be used as a toolkit. For this purpose, some additional research was conducted to understand how to create such a practical piece of knowledge. This research led to understanding in what way to categorise all the findings from the literature review and in what way to present them so that they are easier to grasp and retain. Furthermore, throughout the findings in the literature, you can spot several authors talking about their own lists of level design tips, practices, and patterns, but they are always presented under the same categorical frame. For this reason, the theoretical artefact of this thesis is divided into the following sections:

- **Scope and Context**. This section puts in context all the information presented by explaining the game genres they are focusing on, the expertise of the target readers, and the consideration of PCG within the guidelines.
- **Design Considerations**. This section explains how to put the readers' specific game in context by presenting some game design questions they should answer themselves when starting their projects.
- **Common Design Practices**. This section goes through some common practices professional level designers apply in the industry.
- **List of Guidelines**. This section collects the most relevant concepts and design patterns from level design, and presents them in a particular way to be memorable, practicable, and measurable.
- **How to Apply in Production**. This section reorganises the list of guidelines in such a way that they follow a typical development pipeline.

Each individual guideline was structured to present its definition, techniques and considerations to apply it, its effects and counter-effects, some metrics and validations, and real industry examples that do or do not implement it. This provides a set o knowledge that is understandable, actionable, and measurable, and that is grounded in proven examples. Additionally, each guideline includes a section that explain its connection to other guidelines since they all work together like a network of concepts, not as individual and isolated tools. To gather the list of guidelines, all the key concepts collected in the first milestone where grouped by proximity into cohesive cascading blocks. The title of these blocks would then become each individual guideline.

An important consideration is that the shape in which you present your knowledge has different purposes and trade-offs. That is why the list of guidelines is presented in a specific order, and then they are revisited in a different order to understand how to apply them in production. The first sorting is useful as a presentation of the concepts to understand them and progressively and conceptually connect them in the reader's mind. Another useful representation was the original diagram itself, as seen in Appendix B, but these are harder to translate to a textbook format, so they were used as a stepping stone instead. In the written guidelines it is harder to visualize how all these concepts link to each other, but with the original diagram that relation is not lost. This means that the representation of knowledge in this thesis is not the only correct and universal way to do it, and other researchers or professionals can differ in this kind of presentation. One specific example is the video from the industry professional P. Field [4], from which many different concepts were extracted for this thesis. As seen in Figure 4.1, the order in which concepts are presented in the video have nothing to do with the contextual grouping arranged in the guidelines. However, this order does make sense for the video, since all the concepts are linked to a spatial and timely progression, in the same way they belong in the final product of a video game. The video uses this space to present how these concepts work and interact with each other, such as how to use sightlines to present clear goals, or how to apply pathing techniques through a vocabulary.

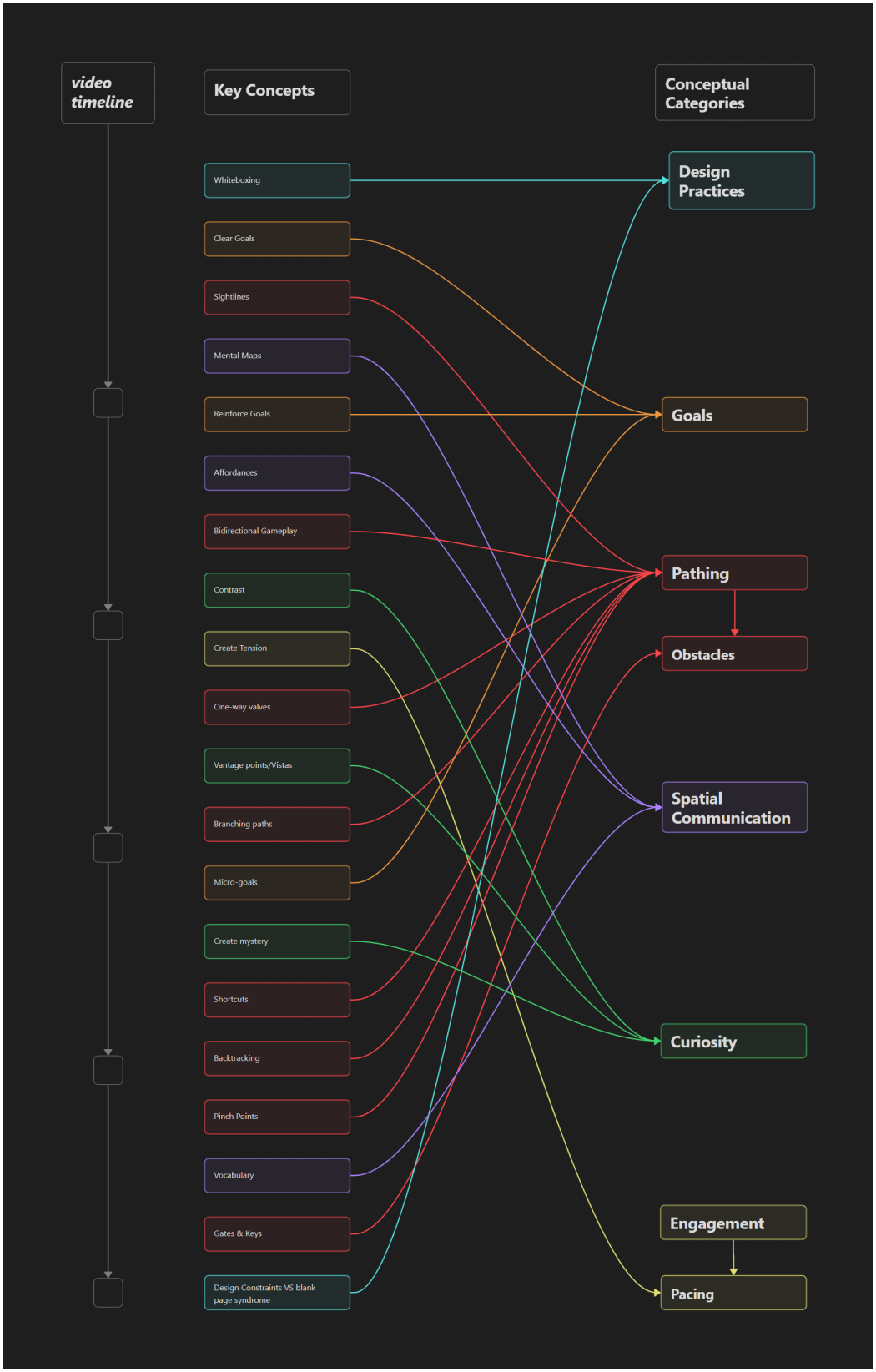

Figure 4.1: Visualization of the chronological order in which level design concepts are presented in Peter Field's video about spatial communication [4]. Each concept is connected to its respective guideline [Section 5.1.4].

4.3 Construction and Purpose of the Prototype

The design and construction of the prototype was heavily dependent on the results of the guidelines. Explaining these design choices and the purpose of the prototypes rely on being knowledgeable about the key concepts of the guidelines, and for this reason, it is encouraged to read its respective chapter [Section 5.1.4] before continuing with this section.

The first reason to implement the open-world prototype was to put to practice several of the design considerations and the development pipeline gathered in the research phase. Along with this, another reason was to transcribe some of the guidelines into algorithms. The main idea was to **design** the prototype using the guidelines, and to **construct** it using PCG. The design process started by selecting several key concepts, like goal layering, goal motivations, variety, POIs, obstacles, and more. With this selection, specific game details and algorithms could be defined to be implemented, such influence maps, biomes, spatial distributions, specific gates and keys, and the implementation of graphs to account for pacing, challenge, and sightlines.

After connecting guidelines concepts to design choices and algorithms, two major blocks were going to shape the prototype: the probability maps, and the progression graph. The former would use several techniques like recursive noise layering and distribution sampling to achieve design properties like POIs, challenge, variety, and goal layering. The latter would be focused on selecting locations for progression and pacing, and create a game structure to follow the *Pathing* guideline. However, this prototype came with a costly secondary implementation. The mechanics and all the systems required for the minimum gameplay took most of the effort. This, along some technical complications that arose from the 3D nature of the space, led to a change of paradigm.

The second prototype focuses strictly on the idea of translating design guidelines and concepts to algorithms, with no playable outcome at the end. This prototype envisioned to create a map representation of the design choices, but generated procedurally. For this prototype, the approach consisted of focusing on a wide variety of procedural methods and techniques, and linking them to the appropriate guidelines that could be represented using them. The selection included specific methods, like noise layering, distribution sampling, CCL, and A*, but it also focused on specific types of input and data representation like sketch-based algorithms, parameters, seeds, and grammars, among others.

5

Results

This chapter presents the results of both the theoretical and practical artefacts. The former summarizes all the gathered knowledge from the level design research. The latter is constructed using this knowledge to present some examples of practical implementation in the form of a simple world generator prototype.

5.1 Theoretical Artefact: The Guidelines

This artefact encompasses all the gathered knowledge from the level design research. Due to its long extension, not all concepts will be explained in detail. You can find a glossary at the end of this document [p.58] to help you throughout the read of this section. This section presents all the key concepts of level design, and sorts them into several categories. It is introduced with scope and context, and then followed up by level design considerations and practices. After that, the guidelines are outlined in a schematic way, since explaining all the concepts and relations in detail would vastly increase the scope of this thesis. However, two example guidelines have been fully developed to show what a final product would look like [Appendix A]. Finally, this section closes up with a guide on how to apply these guidelines in production.

5.1.1 Scope and Context

These guidelines are not a definitive tool that is to be applied anywhere mindlessly. The effort and practical experience of a level designer is required to make informed decisions and understand how to use them to have a successful result. This is the context where the guidelines apply. They can be extrapolated to other scenarios, but they were carefully chosen for this specific criteria.

Genre. These level design guidelines focus on 3D games with open exploration. Even though most of them are compatible with many other genres, e.g. stealth, shooters, metroidvanias, survival, horror, etc., they do not provide a complete set of rules for these genres, since each one of them has specific design needs. Some genres where these rules are harder to apply include platformers or racing games. These are specially discordant with the set of guidelines since their gameplay needs are tied to the structure of the level. While the genres this research focuses on use level design as guidance, the latter examples use level design as the scenery where the action takes place, so they are not explicitly accounted for.

Expertise. These guidelines are aimed for junior level designers with no previous experience, as every concept is explained in detail. However, the guidelines might also be useful for seniors as a toolbox to check from time to time and be reminded of the many design tricks one can use to influence the player's behaviour.

PCG VS Handcrafted. There is no explicit distinction between guidelines related to handcrafted or procedurally generated content. All of these guidelines are appropriate for handcrafted content, but some of them are more flexible and compatible with PCG due to their formal definition and metrics.

5.1.2 Design Considerations

Before using these guidelines, the designer has to stop and think what type of game they are creating. These are a series of helper questions to contextualize your game and have a more clear approach on how to bring it to life. Each guideline informs how different ways of applying it has different kinds of impact on the player experience. These questions will help you understand which of these different ways is more relevant to implement your intended experience.

What type of experience do you want to provide? This falls closer to a game design matter, but level designers must align with this vision, since they are the ones in charge of portraying this experience through the way they build their worlds.

What is your world scale and density? Thinking about how large and sparse your world is can help understand what type of feeling it can provide. A large empty world can represent how a civilization became desolated. A world that is crammed with structures and activities to perform can represent the zenith of the economy in another civilization.

How much presence does the narrative have? Clarifying how central the narrative is helps decide how strongly levels should guide the player towards story beats, how much space to dedicate to cutscenes or scripted moments, and how environmental storytelling should be prioritized. A narrative-heavy game might require tighter pacing and more controlled player flow, while a lighter narrative allows for more open-ended structures and optional content, where the attention of the player may be diverted more often.

How relevant is the exploration in your game? This question is strongly tied to the previous one, since both affect how you design spaces, paths, and rewards. If exploration is key, levels should encourage curiosity with visual landmarks, secrets, and meaningful payoffs for going off the main path. If exploration is secondary, layouts can be more focused, reducing the risk of players getting lost or distracted from core objectives.

Is it okay if the player misses content? Deciding this early affects how you place story elements, collectibles, and side activities. In this matter, you should decide what type of level elements are okay for your player to potentially miss. For missable content you can create more hidden areas and optional experiences that reward thorough players. For content that should not be missed, you must design clearer

guidance, redundancy in critical information, and more visible routes to essential content, making use of your *affordances*.

Balancing subtlety with too much exposition. Related to the last consideration, this refers to how explicit the guidance is towards the content, whether it is missable or crucial. Remember too much subtlety may cause players to miss important cues, while too much exposition can feel heavy-handed and break immersion.

5.1.3 Common Design Practices

This section goes through some common practices professional level designers apply in the industry. This is just a collection of the most relevant practices. They follow no narrative structure, so you can read them as independent pieces of information to have in mind when working on a video game.

Iteration Pipeline. The most basic practice that is always present in any project is iteration. Approach the design process in small steps and loop through them to find out what works and tune the result to your specific needs. Specifically for level design, a contender list of steps includes:

- **Design on paper.** Before you start putting too much time and effort into building your levels, you should first design them and make sure that the structure is built in such a way that it follows the design guidelines you want to implement. The process of building a level should only happen when you think it is ready to playtest.
- **Apply *greyboxing***. This concept refers to building your level with as simple geometry as possible (usually grey or white cubes), without paying any attention to the looks or details. Some specific level elements might be required in this step, such as point lights, sound sources, or moving elements, but these should be part of specific level design needs (as some guidance tools explained in Section 5.1.4). Once the level is greyboxed, it is usually playtested internally. Here, you iterate and go back to design on paper whatever did not fulfil your needs.
- **First Playable.** This refers to the version of the level that is ready to playtest in a wider context. It is usually denoted as a pre-alpha version, where game ideas are more consolidated, and the level works as a whole, and not individual pieces. At this stage the levels can be populated with temporary assets as decorations.
- **Level Art.** Once the level design has been polished and designers have a feeling their level is ready for production, the level artists can take the lead and start giving life to these environments. It is still important for level designers to be somewhat present in this step, since level art can drastically impact how a level is perceived. The use of colour, light, sound, or movement (*spatial lures*) can direct the player's attention away from the path they were intended to follow.

We can see represented in this process of iteration one of the most common recommendations: 'Early Design and Early Testing'. This refers to the introduction of level design early in the development process of the game. Level design is one of the most crucial steps for a game to be engaging, and getting it right is a difficult and tedious task. Making sure your designs work as early as possible allows for work parallelization and detecting core game design flaws early in the development process. It is also helpful for tracking different level design properties such as pacing, variety, downtime, risk VS reward, etc.

Blank Canvas. One common problem we can run into as designers is the blank canvas syndrome [39]. To overcome it, there are a few well known practices that can be used. To start off, we can build *metric playgrounds*. These are a type of playground used by developers to measure and evaluate the implementation of their mechanics. They are more widely used in games that heavily relay on movement or shooter mechanics. In these spaces, level designers can also understand what type of interactions the player can have with the game elements. In a way, these spaces are a collection of all the 'building blocks' level designers can use. Level designers can combine these elements to create *action blocks*, also known as 'Gardens'. These are defined as simple combinations of game elements that portray a specific gameplay instance. One example could be how to use a mechanic to surpass an obstacle in a specific way, such as bouncing on top of an enemy to jump over a wall. With this, levels can then be prototyped by combining different action blocks. As a final remark in the topic of the blank canvas syndrome, remember that **constraints** allow for new ideas. It is easier to create something interesting by solving your way around a constraint than it is to create it from thin air. An example could be designing the layout of a small town, where you impose yourself the constraint to arrange it in a tiny plot of land, or with uneven terrain, including cliffs or rivers.

Polymorphic Variety. One of the key concepts from the guidelines explains the importance of **variety**. It can be very resource intensive to create many different level elements. One practice to tackle this problem is applying some principles from object-oriented programming to level design. Polymorphism for example, can be implemented by taking level elements and mechanics and creating small variations. This implies saving development time and resources, while still providing value to the game. One example of this implementation occurs in the Zelda series, where the different tiers of 'bokoblins' introduce gameplay variation by having small differences. These differences include difficulty (health and damage variance), different types of weapons (bows, swords, spears...), and sometimes even different mechanics, like bokoblins that can use a horn to invoke more enemies. In the context of level design, this practice could be applied to the aforementioned *action blocks*, where simple mechanical ideas can present small variations so that you can easily expand your selection of building blocks.

Living Documentation. Senior level designer J. Burgess explained how his design planning process works for open world games [22]. He explained how a toolkit cannot be provided, since any game's needs are different from one another and *"... the creative goals of a game should shape the process of creating it"*.

This knowledge is only to be applied mindfully in the appropriate context, for a workflow with iteration as a core value. His planning process consists of a *'living documentation'* constituted of 3 parts that can be used to organise and track the design of a large scale open world.

- **The Map.** This is a literal representation of the space. It is used to visually understand how the different level elements relate to each other spatially. Burgess explained here different considerations that I already mentioned in Section 5.1.2, along with some principles included in several guidelines (Section 5.1.4). It is encouraged to read his specific input in this topic [22], although this thesis categorized these concepts where they seemed more fit.
- **The Master List.** This is a list of all the important locations with details on several aspects so that you can filter and compare them. This can be arranged in a spreadsheet software. Some example attributes to store in this list are: the coordinates of the location, the footprint radius, the relevance, the designer in charge, related quests, etc. With this information, you can later run different data visualization methods, like plotting the locations with their influence radius in the map, tracing out the path players might take when they are following a quest, or colour-code the different areas depending on how much progress has been made development-wise.
- **The Location Directory.** This is akin to the master list, but here every location entry is explained in detail. Long descriptions are used to explain the purpose, the looks, the feel, and many other aspects of the different locations. This is a more traditional representation of documentation that can be followed by designers to implement and introduce these areas into the game.

In-Game Documentation. Finally, I wanted to mention the importance of the usage of ***metrics***. These refer to the in-engine measurements developers carry out to track the relative scales the player has with the different level elements and the environment itself. These are quite useful at the early stages of prototyping levels (like *playgrounds*), since you can keep track how the player interacts with its surroundings and its limitations, such as how high the steps of a staircase should be, or how tall mountains should be to appear 'large'. Take into account that games rely on a *sense of scale*, not real scale. This means that using real-world metrics can mess with the sense of scale the players have. *Metrics* should be built around players, not humans. Metrics are usually built in development scenes called Gyms or *playgrounds*. Another type of in-game documentation is the Zoo [40], a type of space where you get to display all of the game elements and assets in an orderly fashion. Here a sample arrangement can be provided. The final type of in-game documentation is the Museum. These focus on displaying systems and interactions with in depth explanations on how they function or how they can be implemented.

5.1.4 List of Guidelines

This section enumerates the list of guidelines to understand what they consist of, and how to implement them. Two guidelines have been fully developed to serve as

an example of what a finished product would look like [Appendix A]. These include descriptions of counter effects, real industry examples and counter examples, metrics and validations, and a list of related guidelines.

Level design can be defined as the process of shaping spaces that guide players and communicate how to interact with them. From this definition we can extract to concepts: guidance and spatial communication. The core of the guidelines is based on these two concepts. Along with these, the guidelines include other core concepts that link level design to game design. These include the definition and implementation of goals (or objectives) and engagement. Some of these guidelines are quite dense, since they present many properties and definitions, and at the same time they intertwine with each other. To help with this, some guidelines are divided into sub-guidelines so the concepts are easier to digest. There is a total of 4 major categories, which are broken down into 11 guidelines. The list starts off with the 2 categories related to game design: **Goals** and **Engagement**. After that, another section explain different rules that define how player **Guidance** works, and the psychologic tricks behind them. Finally, **Spatial Communication** explains how building these levels influence how the player interprets the game world and how they learn to traverse it. In the following outline, each guideline briefly explains what it consists of and lists details on how to apply it.

- **Goals.** Goals refer to any objective or activity the player has to perform that comes from the **desire** to achieve something in the game. Desires can be born from the player itself or they can be induced from objectives presented by the game. Goals arise from these desires: player seeks the consequence of an action, player seeks accomplishment from challenge, or player seeks a direct emotional response. These desires form part of the **negotiation of goals**, where the player comes up with desires and the game responds with its available systems. This guideline is strongly tied to the MDA framework. Goals (which are enabled by mechanics), motivate behaviour (dynamics), which then evokes emotions and feelings (aesthetic). Here is how to implement them:
  - **Seek the intended experience**. Goals should align with the intended player experience.
  - **Layering Goals**. Several goals can be presented simultaneously to provide player choice and encourage player agency.
  - **Clear Goals**. Goals have to be stated clearly and repeatedly, so that players are aware of them and understand their purpose and shape.
  - **Rewards**. Goals have rewards. These can be cognitive or material, but players need to be rewarded in some way when they achieve a task they have been imposed.
- **Engagement**. Engagement describes how effectively a level or a game holds the players attention and motivates them to keep playing. This is built upon three major sub-guidelines: Choices, Meaning and Pacing. Along with that, there are other smaller concepts that strongly affect engagement:

- **Challenge.** This is a measure of how difficult the presented goals are in relation to the skill of the player. An objective that is too easy for the player results in boredom or disinterest, while the opposite case can lead to frustration or even induce anxiety. A good balance of challenge helps maintain the player engaged with the different activities. Challenge can be analysed with the *difficulty curve*, which evaluates the level of challenge in a game over its span.
- **Unpredictability.** It describes how the expectancy of the outcome of an event (e.g. not knowing how an entity can react to your presence) can get the player invested and create the *desire* [**Goals**] to figure it out. One common technique is the usage of *corners*. These are a layout resource utilized to hide game elements behind sharp turns in your paths. This tool helps both applying **Pacing** and boosting **Curiosity**. Unpredictability can also be achieved through randomness and emergent gameplay.
- **Variety.** This refers to the idea that a game should present a wide variety of elements to remain engaging. A good variety can keep the experience of a game fresh and avoid repetitiveness. This concept can be applied in every aspect of a game: game mechanics, themes, environmental art, openness of a level, density of goals, etc. We can borrow the concept of modularity from object-oriented programming to effectively increase the variability by summing up all the different combinations.
- **Game Feel.** This is an abstract and intangible concept that refers to a satisfying sensation experienced when playing games, which is usually linked to responsiveness. This responsiveness reflects on the reaction the game has when a player performs an action. Spaces can create these sensations through the shapes and openness. *Architectural theory* [**Spatial Communication**] can explain how to construct these spaces.

Now we can have a deeper look into the sub-guidelines of engagement.

- **Choices**. This refers to allowing the player to decide what type of activity they want to engage with, or what approach they want to take to face one task. This is strongly tied with *goal negotiation* [**Goals**], where **player agency** is a key element that drives the experience. Unlike cinema, literature, or other kinds of art, interactability (highlighted by choices) is what distinguishes games from the rest of the mediums. This is deeply entangled with many other concepts in these guidelines, such as *player motivation*, *variety*, **Points Of Interest**, and **Pathing**. To properly apply choices, they have to be:
  - **Meaningful**. For choices to be meaningful they have to have **consequences**, present a balance in terms of *risk VS reward*, the player has to have direct control over the action they are choosing (so no randomness involved), and options have to be varied enough for them to understand the differences of choosing one option over the others.

    * **Conscious**. This means that the player has to be aware of the action they are taking, and that they are willingly choosing to do so. For this purpose, they have to know that this is a possibility the game presents and what the possible outcomes might be. This is strongly related to how **Goals** are implemented, following the same principles. So two essential things are required: **player intentionality**, and **signposting**.

  - **Meaning**. This term refers to the sense of purpose that players experience while playing. It consists of the feeling that what they are doing in the game matters, whether to the story, to other players, or to themselves. Meaning can be born from **connections** of many kinds: cognitive (detecting patterns and solving problems), spatial (when meaning is embedded in spaces), personal (reflecting on ourselves), and emotional (attachment). For these connections to be created, the player has to spend **time** playing. It is crucial that they get engaged during this time, otherwise they will not have a chance to experience such types of connections. For level design to create connections we have to appeal to the spatial category. This is best explained from the perspective of ***psychogeography*** and ***environmental storytelling***.

  - **Pacing**. Pacing refers to the speed at which experiences unfold over time. It defines how quickly events, information, or actions are presented. Pacing can be measured in many different aspects of a game: narrative, challenge, mechanics involved, emotional charge, etc. A well-tuned pacing maintains curiosity, focus, and emotional investment. *Challenge* is a key concept in video games that has undergone a deep research in terms of pacing, and that is why it has its own dedicated section in the *engagement* guideline. In terms of level design, we are responsible for the order in which all the different mechanics, goals, narrative bits and other game elements are presented to the player, and we have to be mindful about applying a good **variety**. Furthermore, we have to be aware of the need for **downtime**, sections of the game where the player can rest, think strategies, and prepare for action. For this, resources like *checkpoints* and *safe zones* are commonly used. The concept of pacing is also used to properly apply **layering of goals**.

- **Guidance**. This is a key principle of level design. It refers to how all the design decisions are directed towards conducting the players though the intended experience by taking them through different spaces and game interactions. For this purpose, guidance can be split into three categories: **Points Of Interest** that attract the player's attention towards parts of the game that present gameplay opportunities; **pathing** techniques that dictate how the player can traverse this space and the impact it has on what they actively think; and **curiosity**, which in essence makes the player feel they are in control of the experience they are having. Here is a more in depth look at them.

- **POIs**. Points of interest are locations that have a visually distinctive appearance so that they can be distinguished from afar. They are usually linked to gameplay opportunities, and players can also use them to orient themselves in a space. They keep a close relation to *goal negotiation*, since they can be a source of activities such as enemy camps or quests. They can be analysed from the perspectives of *spatial distribution* and *level of interest*. The spatial distribution presents properties like density, scale, footprint (or reach), and approach direction, which dictate how the space feels in terms of openness or emptiness. They also dictate what type of traversal mechanics are required in the game, such as mounts or fast travel. The level of interest of a location is dictated by how frequently it appears (the less frequent the more interesting), and its relevance and complexity in a mechanical or narrative sense. The visual appearance of a location should match its level of interest. Their visual appearance should also follow the principles of *affordances* [**Spatial Communication**]. A specific type of POIs are *landmarks*, which refer to the larger scale locations that serve for global orientation for the player in large spaces.
- **Pathing**. This guideline serves as a list of resources that consolidates the building tools we have to construct our levels. These are common pathing patterns and considerations required to effectively apply all the concepts mentioned in the rest of the guidelines, such as **Meaning**, **Pacing**, *Variety*, and **Spatial Communication**. The practical implementation of these tools is straightforward, as they follow what is suggested by the rest of guidelines. The list of tools can be divided into four categories:
  - **Sigthlines**. *Sightlines* are a very important tool used by designers to transmit to the player specific bits of information, such as 'that is your goal', or 'here is a path you can follow'. To control sightlines we can make use of *corners* and *pinch points*. The latter can be effectively used to guide the player towards specific *vistas* and compositions.
  - **Flow Control**. We can make use of *one-way valves* to prevent the player from walking back to already visited areas and push them forward. We can also play with opening and closing the explorable space by making use of branching, which incentivises player **Choices** and *agency*. To close up the space we can use the concept of *choke points*, also referred as funnelling.
  - **Revisiting Areas**. If we require the player to go back to a previous area we can make use of *shortcuts* to avoid repetition and prevent them from walking an unusable space backwards. If this space is usable, we can benefit from the concept of *backtracking*, where the implementation of *bidirectional gameplay* is encouraged.

* **Obstacles**. This category constitutes its own sub-guideline due to its deep insights. It differentiates two different types of obstacles depending on their purpose: boundaries and gates. Both of these types of obstacles can be hard or soft, depending on how restrictive they are. Boundaries are a type of obstacle that limit the exploration space and that should discourage (or not encourage) the player from approaching them. Hard boundaries are definitive and delimit the walkable space of the world, while soft boundaries just remark what the intended paths are without restricting the player. Gates are a type of obstacle that is actionable, and its state can be altered to control when the player can get through. These usually require the player to complete a task before they are unlocked. A gate is considered to be soft only when its purpose is to briefly slow down the player from rushing through the level. In this case a simple task like an NPC that requires your attention before you leave an area could be considered a soft gate. Gates present deep insights about how to effectively create locks and keys. These define different categories of gates and different resources to unlock them. **Locks** can be categorized depending on their:

  - **Nature**. They can be *conditional* if they work as a traditional doorway. They can be *uncertain* if they can be avoided by other means that do not require unlocking the gate, like using a secret path. They can be *dangerous* if they pose a threat to the player and they have to acquire some new kind of ability to surpass it, like walking through fire.

  - **Unlock States**. Locks can be *permanent* if they remain open once unlocked. They can be *temporary* if their unlock status only last for a short span of time before requiring the use of a key again. They can be *reversible* if they can be unlocked and locked back again. They can be *collapsing* if the act of crossing them locks them permanently, like a collapsing bridge.

  - **Approach Direction**. They can be *Valves* if they can only be traversed in one direction. They can be *asymmetrical* if they can only be unlocked from one side.

  - **Guaranteed Solution**. They are considered *safe* or *unsafe* depending on if the player can miss the opportunity to unlock them.

  One important consideration is that a designer should always aim for locks to be found before their respective keys, since there is no challenge to the player if they run into a gate they can immediately unlock. Keys can also be categorized depending on their:

  - **Nature**. They can be a *physical object* the player has to find and store in their inventory, or they can be a *level element* they have to find and interact with.

        - **Purpose**. They are *single-purpose* if they just serve to open a lock, or they can be *multi-purpose* if they have other utilities, like a knife you can use to pry doors open or to fight enemies.
        - **Particularity**. They are *particular* if they only serve to open a very specific door.
        - **Persistency**. They are *non-persistent* if they are consumed when used. Persistent keys are useful to match with non-particular keys, and frequent for multi-purpose keys.

    - **Curiosity**. This concept talks about how the player decides on their actions to traverse the space by being influenced by the *desire* to satisfy their curiosity. It is a matter of constructing mystery to encourage player agency over the experience we expect them to have. To create this mystery we can make use of different tools:

        * **Theoretical**. The first and most crucial resource is showing *partial information*. This refers to the idea of not providing the player with all the answers at once, but slowly taking them through the process and give them room to think for what the possible outcomes might be. This is closely related to *Uncertainty* [**Engagement**]. To apply this concept we can make use of *sightlines* and *foreshadowing*. Another way of application is through *denial spaces*, which refer to those parts of the level where the goal is visually lost to the player and which create small 'pockets of tension', which reward the player once they leave this space. The approach the developers of *Outer Wilds (2019)* had consisted of firsts inspiring curiosity by planting the seed of a question, then guiding the player though the hints, to finally rewarding the players with answers to these mysteries.
        * **Practical**. A powerful and practical tool to guide players through their curiosity is by using *breadcrumbs*. These can be created with explicit level elements like trails of collectibles, or in a more subtle way by using *spatial lures*. These are constructed through contrast, which serve from artistic elements such as colour, light, sound, scale, or even motion. *Vistas* are also another resource that can help inspire questions in the mind of the player. These vantage points serve as spaces where the player can observe their surroundings and plan ahead what their actions are going to be. Here we have an interesting opportunity to apply the concept of *signposting*.

- **Spatial Communication**. This guideline explains how we can transmit information to the player with the different layouts we use to construct the spaces. This guideline is strongly influenced by storytelling, environmental art, and world building. In this sense, a level designer has to be informed and have active discussions with the different roles in the development team to successfully transmit this vision to the player. The concept used to explain how to communicate with the player is the *visual language*.

This type of implicit communication is implemented through *affordances*. These are recurring level elements or interactions that inform the player how future interactions with the game might be. A proper implementation of affordances implies the player will be making *conscious* **Choices**. For this, affordances have to be both *readable* and *consistent*. The player can find satisfaction both from confirming their expectations or them being subverted. Readability is a concept that is also important to apply to our spatial structures.

- **Spatial Structure**. The spatial structure describes the different shapes and attributes our space can have, and what influence they have on the player sensations. These spaces can have different properties that define them, like scale, verticality, openness, tightness, symmetry, etc. Making readable and comprehensible layouts for our levels help the player create *cognitive maps*, which are the mental representation they have of the space. A specific way to play with these cognitive maps is by getting the player lost so that later when they find the way back they can connect in their heads how these spaces are structured.

  In a more practical sense, layouts can be conceptualised as graphs. These are structures that describe locations or level elements as nodes, and the paths between these as edges. *Boundaries* are implicitly defined through the lack of edges between nodes. These graphs can be structured in a hierarchical way, e.g. the arrangement of towns in a game form a graph, but then every single town has its own sub-graph. This hierarchical structure is intuitively created in the cognitive maps of the players. Graphs can have attributes. These can be assigned to the edges, such as the distances or difficulty of traversing them, or they can be assigned to the nodes themselves, informing the value of these locations. The traversal of the graph depends on if it is directed or not, i.e. edges can only be traversed in one direction. The structure of the graph translates into different **Pathing** tools that make interesting playable spaces, like *cycles*, *backtracking*, *shortcuts*, *choke points*, and *gating* among others. Graphs are also a useful tool to validate the existence of a *critical path*. Spatial structures are an implementation of all the theoretical concepts explained in these guidelines by making use of all the practical **Pathing** tools. Here is where we implement sightlines, uncertainty, variety, choices, pacing, breadcrumbs, points of interest, and more.

5.1.5 How to Apply the Guidelines in Production

To better understand this section you should be familiarized with the guidelines and key concepts. This section lists the guidelines in a possible order of consideration to develop a game. However, this is not the only possible approach, since depending on the needs of your game and your development team priorities might they have to be rearranged. Furthermore, remember that the development process should follow an iterative paradigm, and although things are presented in a linear progression, every concept should be revisited and verified when needed. Additionally, this proposition

also requires to work in parallel with game design, implementation, and/or playtest at every milestone. Finally, bear in mind that the guidelines are by no means universal rules that everyone should follow. Remember that "... the creative goals of a game should shape the process of creating it" [22], so the designer intent and artistic vision should be the ones to listen first. With all this, we can have a look at the four distinct phases that level design can iterate through:

- **Game Design**. In this phase you have to first design your **game experience**. Decide what is it that you want your player to do and feel, and design your **core mechanics** accordingly. With these you can start thinking about what are going to be your major **goals** and some candidate secondary or lesser goals. As a level designer, you can also think about what kind of *connections* you want the player to have with your game, investing it with **meaning**. This concept should be revisited to ensure it is properly implemented.
- **Building Blocks**. This phase consists of deciding (and maybe starting to implement and playtest) what your building blocks are going to be. You can start by constructing tools for your **vocabulary**, such as identifying what objects are interactable or not, or what surfaces are intended to be walked on, or how something looks like if it is out of bounds. Additionally, you can work on your different **points of interest**, accounting for different levels of scale and relevance, and deciding how frequent or rare they should be. Along with these, you can also come up with the principal **landmarks** that are going to serve the player for their *cognitive maps*. To help you design these points of interest you can make use of the **engagement** guideline, which explains how to implement challenge, unpredictability, variety, and game feel.
- **World Structure**. This is the phase where you work on your first layer of your **spatial structure**. Here you decide where to place your *landmarks* and regions, start organising *sightlines* and roads that connect the different parts of your world, and start setting up the **pacing** and **choices** by laying out your *points of interest* following specific spatial patterns.
- **Gameplay Instances**. Once you are done with the overall structure of your world, and the building blocks are ready to be used, you can begin to put in place your moment-to-moment experiences, following all the **pathing** tools explained in the guidelines, and setting up your **obstacles** and boundaries. Here is the perfect place to implement **curiosity** by making use of all the theoretical and practical tools explained in the guidelines. You can use *denial spaces*, implement *breadcrumbs*, make use of *spatial lures*, organise *vistas*, and implement *signposting* through *sightlines* or other means.

This list of phases explains a top-down approach, where major structures and goals are first defined, and then designers and developers implement finer details until the moment-to-moment experiences are integrated. The inverse approach is also possible, where you come up with specific gameplay bits, and construct upon them and merge them to create larger and larger experiences until your game includes all these bits of gameplay. No approach is better than the other, just remember that which one is best fit for you depends on what type of game you want to build.

5.2 Technical Artefact: The Prototypes

The first intent of the prototype was to display a practical example of how one could implement some of the elicited guidelines of the theoretical artefact into a game making use of different procedural algorithms. The first approach consisted of implementing a simple open-world adventure game where modularity would be the key. Modularity would enable the creation of a substantial amount of content with minimum systems, so its variety and quantity would not pose a problem. However, most of the development time went into implementing many systems that had nothing to do with either level design guidelines or procedural generation. For this reason the project had to be trashed, and a change of approach was required. With this, the second prototype was designed to be a straightforward translation from guideline concept to implementation.

Despite this change of approach, we can get useful insights from both projects, so in the following subsections I will go through the main design choices and the design of the algorithms of each prototype.

5.2.1 Open-world adventure: Floating Islands

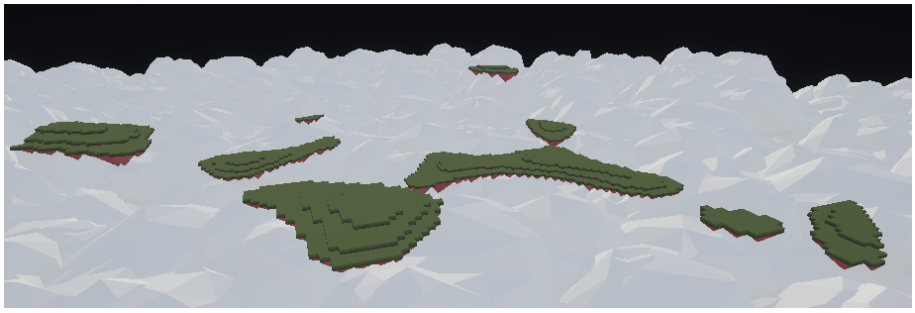

Figure 5.1: An example of a generated cluster of islands.

The development process of this prototype was split into two parts. The first one consisted of the game design and the implementation of many of the systems required to get the game loop running. The second part was in charge of implementing the world generation.

The design process of this prototype followed J. Burgess's approach [22] of having a living documentation, where we represent our game details in the shape of a map, a list, and a directory. With this, I could keep track of what were the different locations, where they could be found, and what characteristics defined them. All the design documents, both text and diagrams, were written in *Obsidian* [41]. I first started defining what kind of goals the player would have, and what kind of mechanics would be required for that purpose. The game idea consisted of a character that could glide between different floating islands in a finite world enclosed by a dome in the skies and a sea of clouds. All the goals were centred around the concept of exploration, with different narrative motifs behind them. These narrative motifs belong to three different categories:

- **Spatial**. Player could wonder about the boundaries of the world, and explore to break those limits, or understand why they exist.
- **Existential**. Learn about the origin of the world, or find out if they are alone in the world.
- **Practical**. The starting town is running out of resources and they need to expand and search for technological advancements.

With these conceptual ideas the core mechanics came right after, including: the movement (jumping, gliding, sprinting, stamina consumption, gliding leaps, etc), the items (interaction, trading, keys and locks, resource nodes), and the character upgrades (increase stamina or number of leaps). Then, the game loop was defined with these mechanics in mind, resolving in the diagram shown in Figure 5.2, where the four core game interactions are highlighted in green. In this diagram you can see how everything connects, and how a typical game experience could be.

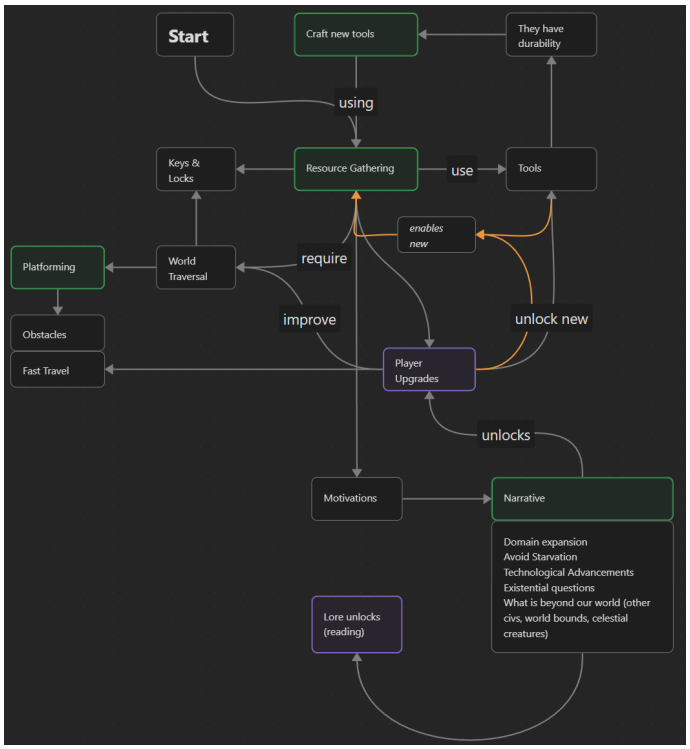

Figure 5.2: Game loop diagram of the first prototype.

The game progression was then defined following the guideline of ***Obstacles*** [5.1.4], where different items and narrative beats could be understood as *Keys & Locks*. Each of the keys could be found in different biomes, which had different traversal and tools requirements, encouraging the player to engage with the game loop.

After that I worked on the *Master List*, a list of all the different biomes and their properties: the available resources they have, the tools you could discover in them, the visual aspect, the islands layouts (verticality, scale, density, world position, etc), the POIs, and a brief summary of the culture of its civilizations.

The relevant part of all these design choices lie in the relation they have with the guidelines. Although the relation is mostly implicit, the following list explains some of these guidelines being integrated in the design of the prototype:

- **Choices**. Players would have to decide what type of activity to engage with, like where to go next, what quest to engage with, what resource to seek, or when should they return to the hub to update their progress.
- **Variety**. This is explicitly implemented through the biomes and all their differences of layouts and looks. Challenge would also vary on how incrementally difficult the traversal becomes when islands start being smaller and more sparse.
- **Spatial Structure**. This would be defined through a graph implementation, which both affects progression and pacing. It also represents obstacles as unreachable distances. The different traversal mechanics defined the scale of the world, and not the other way around.
- **Points of Interest**. These elements are also explicitly implemented as locations where the player can interact with the game, from resource nodes to towns where they can trade and craft tools.
- **Curiosity**. This is achieved through affordances, which are easily implemented thanks to the modular nature of the generation process. Players can have easier times reading when elements of the scenery are interactable or not, and then make better informed choices according to those preconceptions.

When the design phase was finished I translated everything into systems I would need to implement. The movement mechanics where straightforward, but implementing systems like health and stamina, inventory systems, interaction systems, item drops, dialogue boxes, and more turned out to be too much effort and time spent in systems that would not really contribute to the purpose of this thesis. At this point I jumped into the world generation instead, but shortly after realising this game would not be playable due to its cost of implementation I pivoted into the second prototype, with a more simple approach. The world generation of the first prototype still provides useful insights about the complexity and time that can go into implementing intricate rules following a procedural paradigm instead of just hand-crafting a finite world.

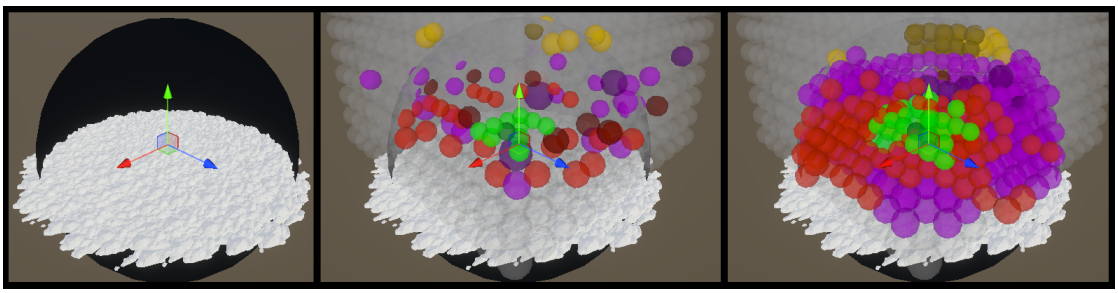

Figure 5.3: Process of biome cells generation inside the world dome of the first prototype. Middle picture represents random biome samples. Right-most image shows these samples expanded to fill the space of the dome.

The world generation process can be split into two differentiable parts: the generation of biomes and the generation of island clusters. Biomes are generated in a 3D grid space, where each cell defines what type of biome it is. For each of these cells, a cluster of islands is generated according to how the biome properties dictates.

The biome generation process decides what biome each cell belongs to. If a cell has no biome designated, it will not generate any islands inside. The process is portrayed in Figure 5.3, where a series of samples per biome are spawned in the grid, and then they are expanded to take over adjacent cells to fill the space. Finally, the cells that fall outside the boundaries of the world (outside of the black dome) are discarded. The algorithm defines a list of properties *per biome*:

- The number of initial samples to spawn.
- The sample growth radius, which defines how many units the sample will grow.
- The world height at which they can spawn (measured in a range).
- The priority they have, which dictates what biome survives when one grows inside another.
- The distance from the horizontal centre coordinates at which they can spawn. This can be a maximum radius from the centre (green biome), or a minimum one (which results in a doughnut shape, like the red biome).
- The specific settings of how a cluster of islands can be generated in a cell of this biome.

Once all the biome cells are generated, I iterate through all of them to generate each individual cluster of islands according to how each biome defines its generation properties. Some of these properties are defined as distributions to avoid repetition. This means that they have a mean value and a standard deviation (and min/max), so that every time that I need to read the value I can get a slightly different result. The generation properties of a **cluster** include:

- Height multiplier (distribution), which defines how many world units of difference there can be between the lowest point of the island and the highest.
- Verticality factor (distribution), which defines how much elevation difference there is between islands of the same cluster, as seen in Figure 5.4.
- Perlin Noise settings. These include the typical X and Y coordinates, the scale factor, and the resolution, along with min/max thresholds. Additionally, a parametrised sigmoid function has been defined to smooth out the generated results, with settings such as min/max factor, exponent (affects steepness of the slope), and distance (defines at which value lies the centre of the slope).

The island generation algorithm is quite more complex than the biome cells. Firstly, a noise texture is generated where each pixel represents the elevation of an island tile. Then the pixels outside the circle circumscribed in the texture are discarded to reduce the effect of islands looking cut off at the edges.

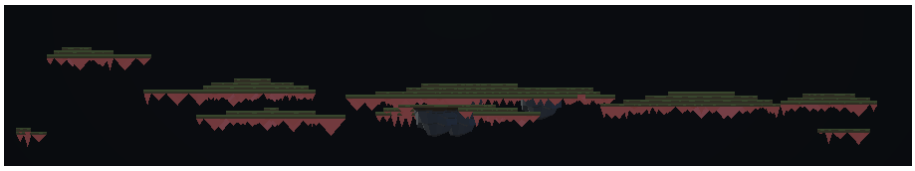

Figure 5.4: Side view of the altitude (verticality) of the different islands in a cluster.

With this new texture, a Connected-Component Labelling (CCL) algorithm is run to detect individual islands and operate them separately. Then, within each island, tiles of **equal height** are merged into squares. The same process is repeated for the remaining tiles, but this time with rectangles. This grouping allows for three benefits: performance, since less island pieces have to be instanced; appearance, since larger tiles can be appreciated from a bottom perspective; and practical, since we now have measurements of the size of the tiles, which allows me to instance large structures on top making sure they will not be clipping the ground. This tiling is visible in Figure 5.5. These steps provide a series of local cluster coordinates, which I can then phase in the Y axis using the verticality factor, and then transform into global coordinates using the coordinates of the biome cell. With all this information I finally instantiate all the island pieces and scale them to the pertinent calculated tile size.

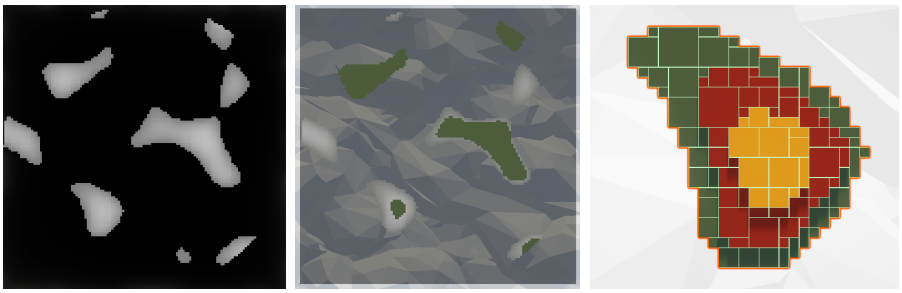

Figure 5.5: (Left to Right) Perlin Noise sample tuned with Sigmoid function and cutout inside circle, Islands Instanced with noise texture overlayed (some parts of the islands are hidden underneath the texture), and tiling of a specific island.

From here onwards, the development process stagnated due to gameplay requirements in the generation process. At this point I was required to measure distances between islands to generate a graph with which I could arrange progression and pacing. However, this posed a complex challenge with several factors. First of all, the shape of the islands is organic and not convex. This implies that, depending on the algorithm, the measurement could be highly over or under estimated. Furthermore, to create the reachability graph a single value would not suffice, since depending on the unlocked abilities of the player, they can traverse the space in different ways, and depending on how much stamina or glide leaps they unlocked, the resulting graph would look different. This means that the graph would have to be dynamic and react to player upgrades. Additionally, to understand if a measurement is valid for the player we have to account for height, duration of glide, number of available leaps, and so on. These factors greatly hindered measuring the reachability between islands.

Finally, design-wise another gimmick was required to allow the player to go back to already visited islands, since they can always glide down, but that does not guarantee they can come back the same way. This means that a directed graph was needed. With all these technical complications, the first prototype was discarded and the second one came to life as a simple representation of how one could translate a guideline into an algorithm rule, without any gameplay features getting in the way.

5.2.2 Open-world map: Guidelines to Generation Rules

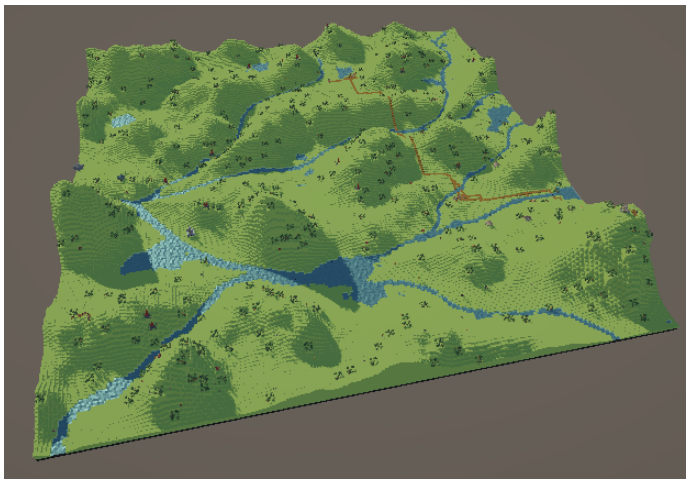

Figure 5.6: Overview of an example generated map.

The purpose of this prototype was to have a one-to-one representation of how some guidelines can be translated into different properties of an algorithm. The approach consisted of creating a maquette representation of a slice of playable space, without it actually being playable. This prototype was just in charge of creating the play space, without any of the game mechanics or interactions being implemented. Several of the implemented rules pair up a guideline with some specific types of procedural generation methods, e.g. parametrised distribution sampling with challenge pacing.

Before getting through the guidelines that are portrayed in this prototype we will go through the generation pipeline to put them in context. The prototype uses several textures and arrays in its generation that serve as representations of the calculations before finally constructing the world and instancing all the elements in place. The generation pipeline goes as follows:

- **Initialization**. First, the algorithm resets by destroying the previous world if there was one and then it initializes all the required textures and lists. At this step the algorithm can set the seed of the algorithm if one is provided.
- **Elevation Map**. The first step of the generation is to create the world elevation (hills and valleys). To do so, 3 octaves of Perlin Noise are layered to achieve a simple effect of terrain.

- **Water Bodies**. The elevation texture can generate small pockets of water, but for the generation of rivers (or any kind of water body), the level designer can input a hand-painted texture, as seen in Figure 5.7. The 'water bodies' texture is generated by mixing this input texture with the water locations generated in the elevation map. With this new texture, the algorithm calculates the 'near water bodies' texture (with a parametrised radius). The 'water bodies' texture is then used to mask the elevation texture, and then the 'near water' texture is used to smooth the terrain down to water level (Y=0).

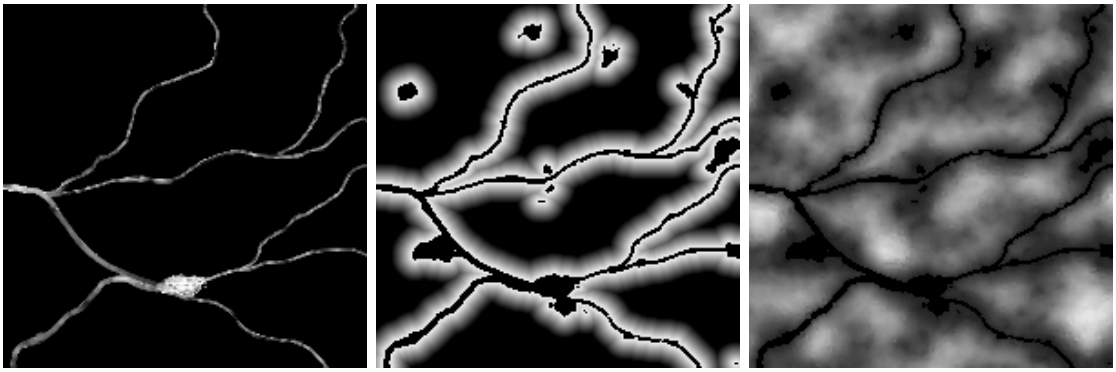

Figure 5.7: Hand-painted water bodies texture (Left), Near Water texture (Center), and terrain elevation map (Right) after applying the water bodies mask (brightness altered to be more readable).

- **Inclination Map**. Once the elevation texture is ready, an additional texture is generated by calculating the steepness of the elevation (using the gradient). This 'inclination' map is later used to decide the spawn probabilities for a variety of different types of elements. You can find this texture in Figure 5.9.
- **Towns Spawn Map**. At this step the probability spawn map of town buildings is generated. This is where special generation rules linked to game design start to apply. This spawn map uses the information of the near water texture and the inclination texture (along with some parametrized weights and thresholds) to define where towns are more likely to spawn. These rules dictate how towns will spawn near bodies of water and in rather flatter areas.

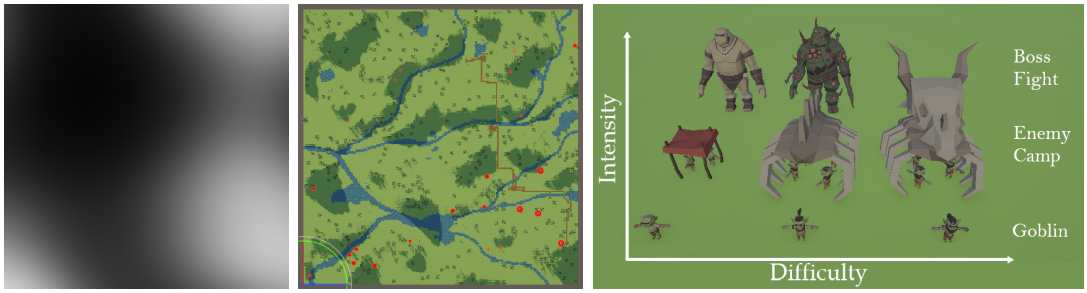

Figure 5.8: Intensity Map (Left), Enemies Spawned (Middle), and Enemy Ranking (Right). In the intensity map, a brighter value represents an area with more intense enemy challenges. In the world map, enemies are marked with red.

- **Enemies Preparation**. For the generation of enemy locations two additional textures are required: difficulty and intensity. These are generated through a single layer of Perlin Noise, where the level designer has parametrized control. The difficulty texture defines how challenging the type of enemy encounter is, by using different tiers of enemies, with increasing health and attack values. This texture also uses elevation as a factor, so harder challenges can be found at the top of the hills. Intensity, however, will determine what type of challenge the player will have to face, from single wandering enemies, to enemy camps, to boss fights. This is visualised in Figure 5.8.
- **Generate Locations**. Different types of POIs have different approaches when their location is generated. They all avoid invalid spawn positions by using the Global Map (Figure 5.9) as a mask. When a POI's location is decided, it is marked as no longer available (black) in this map. Some of the POIs generate a random selection of elements from a pool of prefabs while others, like the *unique locations* (player spawn, main goal, and enemy dungeon), are used as a raw list to instantiate only once. The structures that occupy more than one tile of space can alter the elevation map so that they will not clip through the ground. The most interesting locations generated are the towns. These have an intermediate step that generates a series of clusters around the world to use as a mask over the Towns Spawn Map, so that the location of every individual building is not completely random, but they clump around these clusters. To sample from the probability maps (Elevation, Inclination, etc.) I make use of a method that transforms the pixels of the texture into a Cumulative Distribution Function (CDF), where brighter pixels have a proportional higher chance of being selected. Once a value has been sampled, it gets removed from the CDF and the total sum value gets updated. The sampling of enemies follows an additional rule that uses both the intensity and difficulty maps to decide what type of element is going to be spawned, since they have different footprints and they might impact the elevation map.

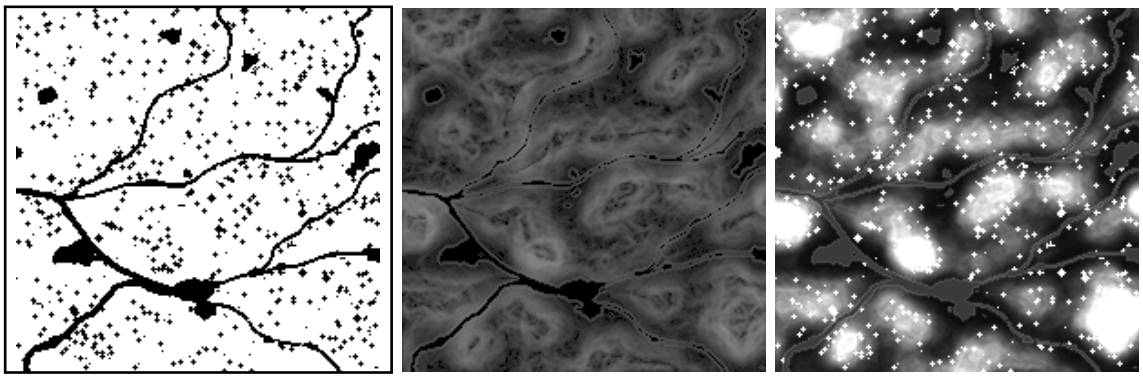

Figure 5.9: (Left to Right) Global map (instancing availability), Inclination map (derivative of elevation), and A* Weights for creating paths between towns.

- **Town Paths**. Once the town locations are generated, some dirt paths are created to connect them. These sample individual building locations and pair them up with another one that is inside a minimum and maximum distance, so that paths are not created between buildings in the same town or towns that are in opposite corners of the map. The pathing algorithm uses A*, which requires a list of weights. These weights are stored in a texture (Figure 5.9), that is generated by using the elevation, inclination, global maps, and water bodies. Paths will avoid cliffs and water when possible, and they will never run into an obstacle. This behaviour can be seen in Figure 5.10.

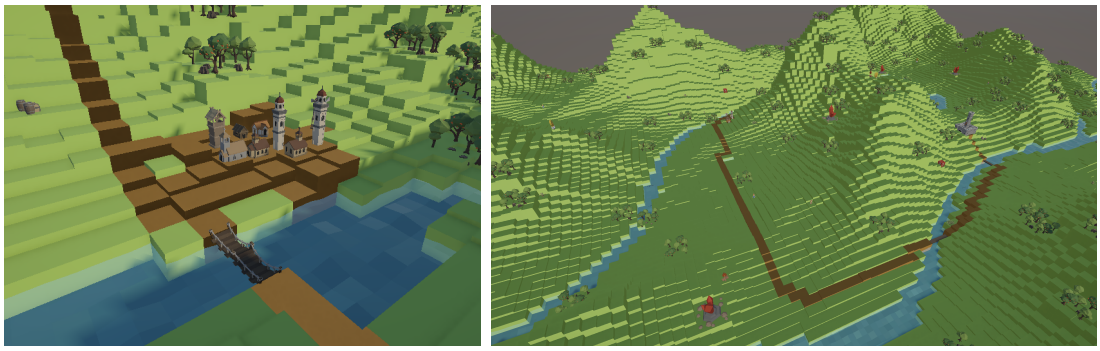

Figure 5.10: Preview of different implemented properties. Towns tend to spawn in low inclination areas near the water. Paths are generated avoiding water and hills.

- **Instantiation**. All the previous steps worked with different data structures like textures and arrays of coordinates. These are now used in the final step to instantiate all the elements in the world. The elevation, water bodies, and towns paths textures are first used to generate the land. With these, each tile can select to be one out of the 3 possible types: land, path, or water. Then, the algorithm runs through all the saved locations for POIs and instantiates them with a random orientation. You can visualize these results in Figure 5.11.

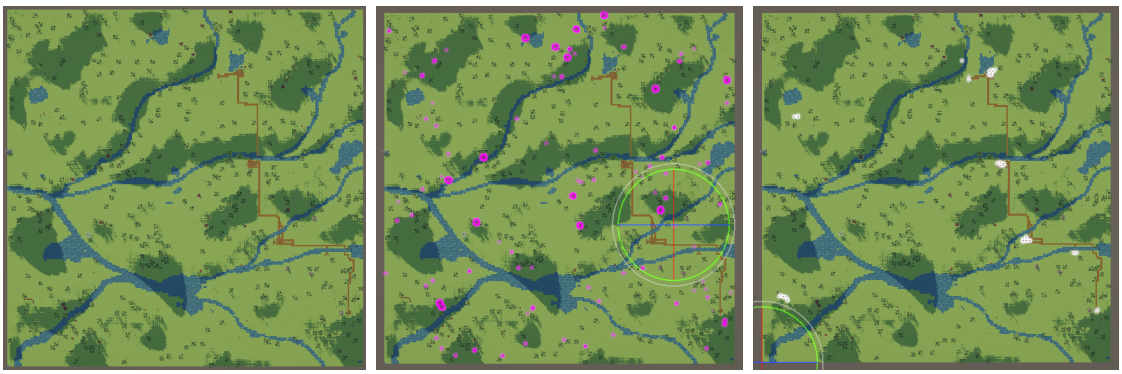

Figure 5.11: (Left to Right) Top-down view of the world map, then highlighted POIs in magenta, then highlighted towns in white.

The structure of these generation rules not only creates a playground where the player can spend some time at, but they can also provide a specific gameplay experience with designer intent behind. Tying the spawn location of all these level elements (NPCs, enemy encounters, quest markers) to spatial generation rules, like altitude or closeness to water, leads to specific gameplay opportunities that are supported by the level design guidelines. Furthermore, by defining goals that encourage the exploration of the environment, the player has an excuse to visualize their surroundings and come up with their own *desires*.

These goals can be formally defined through specific verbs such as "go to, find/explore, defeat, interact, or acquire". These verbs can then be attributed specific subjects or nouns to start constructing a **grammar**. You could go to or find a location, you could find a level element or an enemy, you could acquire a resource or a new quest, etc. These structures, along with some properties and constraints that could integrate pacing, difficulty scaling, or spatial rules, can become an effective grammar to further dictate the world generation in the shape of a *graph*. However, due to time constraints this quest generation system was no further developed. This means that guidelines like Pathing, Choices, and Spatial Structure (graph generation) were not implemented into this prototype, but have a strong potential. The guidelines that do have representation in this second prototype include:

- *POIs.* The usage of probability spawn maps and CDFs naturally work with POIs due to their parametric nature. Defining properties like density/sparsity, clustering, and uniqueness, and tying these to their level of interest and footprint or visual impact makes for a compelling implementation of the guideline.
- *Pacing and Challenge.* The prototype follows the simple approach of the difficulty and intensity maps to define these two properties. It would be interesting to see a more complex system where these maps are used along the progression graph with the quest generation system, but that is further to be explored in the future work.
- *Variety.* The modularity of these systems enables the addition of new elements to a potential game without much effort, just defining the type of spawn requirements they have, and setting up their spawn distribution parameters. One example could be adding new types of enemies, where instead of just goblins you could have lizards, knights, or ghosts, with their respective variants for different levels of intensity and difficulty.
- *Pathing: Vistas.* This one particular pathing technique is achieved organically by tying the spawn rate of different level elements to the elevation of the map. Some POIs are designed to spawn on top of the cliffs, while towns will mostly spawn near water in the valleys. This creates a gameplay loop where players can be commanded from a town to go complete a task at one of these POIs, where they can then scout their surroundings and find new objectives.

6

Conclusion

This thesis explored how to influence player experiences in open-world games through level design using procedural content generation. The research process of these four pillars followed an exploratory research by applying a scoping review. Concepts that are more self-contained like gaming experiences in open worlds and PCG had their own individual research phase, while the exploratory research of level design served to connect all the concepts together. Level design can be linked to game experiences through the concept of **goal negotiation**.

However, the scoping review showed how there is a gap in the literature about the integration of level design and PCG, where there is no common ground on what a correct approach would look like. This thesis served both to point out this gap in the literature, and to prove the feasibility of a practical implementation of level design principles with procedural generation methods in such a way that they can align with the intended game experience. This feasibility was explored by first formally defining all the level design concepts, and then implementing several of these rules into practical prototypes that would display them in a dummy environment.

6.1 Discussion

This section explores the takeaways and limitations of the resulting artefacts of this thesis, along with some ethical concerns about the usage of automatic generation methods in a designer context. This section also questions the nature of this thesis and challenges the idea of creating interesting open-world experiences with PCG.

6.1.1 Takeaways of the Results

The guidelines provided in this thesis aim to create an actionable body of knowledge in a textbook-like structure that can serve to learn and be used as a guide to apply level design concepts in a video game. This is based in the theory of *Pattern Languages* [34], where a common vocabulary can serve (designers and researchers in this case) to ease communication and have a representation of common problem-solution pairs. These guidelines are also heavily inspired in the *Sapir-Whorf hypothesis* [33], which explains how *'the language that we speak shapes the way we think'*. This can be translated into how designers envision the spaces they create based on the design concepts they are familiar with.

The guidelines also remark the importance of the realisation that different presentations of the same knowledge have different purposes, benefits, and drawbacks. This is portrayed by an example from the literature, from which several concepts were extracted (P. Field [4], Figure 4.1), but where the sorting of these concepts has nothing to do with the arrangement of the guidelines. This concept is later reinforced in Section 5.1.5, where an alternative sorting is provided with a different focus. This alternative sorting groups the guidelines in such a way that they are easier to apply in a development pipeline instead of just teaching the concepts to the reader.

One final consideration about the guidelines is that they try to encompass a broad field of research, where many of these individual concepts present extensive studies of their own. A problem arises with some parts of this knowledge being tacit, which professional developers try to explain in their insights, but which are better understood through practice. Furthermore, the intent of the guidelines in this thesis was not to serve as a final result of the research, but as a stepping stone to then be applied through procedural generation. For these reasons, the guidelines cannot be considered a complete set of level design knowledge, since they were not conceived for that purpose, and aiming for that would pose a monumental task. To exemplify, here are some of the missing concepts that were not mentioned in the results:

- Attraction versus repulsion of the player in a space. What are typical elements that can serve for these purposes, and what are the benefits and consequences of using one over the other.
- An exploration of the usage of *Game Design Documents* (GDDs), and their alternatives as different types of live or in-game documentation.
- Insights on how players usually read the spaces they are presented with. How they can ignore verticality if they are not prompted, or how they can read the space from left to right, and the first thing they see can be the one that sticks with them. An explanation on how some players will try to explore secondary paths before the apparent main path.
- How to effectively implement paper prototyping, as in, designing levels with pen and paper. This refers to the usage of layers, colour values, and line girth to clearly represent the ideas behind the design of a level.

One particularity of the level design research is that all the findings are not directly referenced in the main text. These have been moved to Appendix D. This is due to how many of the sources overlap in the topics they cover and how each individual source often addresses multiple concepts, so having in-text citation would clutter and confuse the reader rather than provide useful information.

The practical artefacts also have a substantial list of limitations. These were considered during the design and implementation phases, but ended up discarded due to scope constraints. It is important to be aware that the nature of the prototypes was to just serve as a display of the feasibility of implementation of the level design rules through PCG. Their purpose was to show the potential of this integration, not to actually prove how to do so with a functional and playable result.

Not having a playable prototype implies that no metrics could be applied. Most of the metrics proposed in the guidelines [Appendix A] require a complete implementation of said guidelines, which is not possible from the conceptual approach of the prototypes. Furthermore, not having a playable prototype means that its efficacy can not be proven by playtesting and surveys. Again, the purpose of the prototypes was not to show a working implementation, but to display its feasibility.

However, one specific thing that the prototypes proved is the increasing complexity of procedural generation implementation with abstract design rules. The prototypes display examples of both implemented and conceptualised algorithms, and the effort and difficulty that they pose. The first prototype displays the complexity involved in the generation of the islands, their variability, their biomes, and their structures. The generation process was complex enough, but at the end it showed how the true complexity arises from the integration of gameplay restrictions, in this case by trying to measure the distance between organic-shaped islands to arrange game progression. The second prototype also displayed the complexity of the implementation of several techniques with the different instantiation rules of the POIs (enemies, towns, quest markers, etc.). It also proposed a first approach to a grammar generation to organise quests both following the pacing and pathing guidelines.

6.1.2 Evaluation

The exploratory nature of this thesis enabled the creation of a foundational taxonomy of level design, while establishing the background for its connection with game experiences and its integration with procedural generation. However, this approach also resulted in an expansion of scope that constrained the use of formal evaluation methods. These include the use of metrics or user studies to validate the results, which were ultimately delegated to future work. Despite this, the application of research-through-design enables a process of self-evaluation, which can be applied both to the research process and to the artefacts of this work.

As explained in Section 4.1, the research process followed a roadmap divided into three milestones: literature review, field research and design, and implementation and evaluation. A fourth milestone could also be considered if the writing of this thesis is included, which took approximately three months. However, while the second and third milestones took 1.5 and 2 months respectively, most of the time was spent on the first milestone, which spanned five months. This literature review phase focused primarily on level design, which explains the strong presence of the theoretical artefact.

While research on game experiences, open-world games, and procedural content generation could be explored through a limited number of well-chosen sources, level design revealed the absence of a unified and formalised body of knowledge. This is apparent in Appendix D, where each source presents only a subset of the concepts collected in this thesis. Furthermore, some of these concepts are grouped differently in the literature, whereas the guidelines distinguish between design considerations, practices, guidelines, and operational resources.

The creation of the level design concept tree presented in Appendix B underwent an iterative process. During the review of sources, concepts were initially scattered and were gradually grouped and connected into cohesive categories. This process of connected thinking later allowed the identification of gaps in the literature when attempting to link level design to the other pillars. Finally, transforming these diagrams into guidelines not only provided a useful way of presenting knowledge in an applicable format, but also revealed how different forms of representation serve distinct purposes. While the guidelines are structured in a textbook-like and digestible format, I personally found the diagram representation more useful as a cognitive map of the concepts.

While the guidelines lack formal validation, they are grounded in reliable sources, including peer-reviewed papers, theses, books, and professional insights such as GDC talks. Although their structure has not been evaluated, they have personally helped me develop a deeper understanding of the concepts and operationalise them for practical application rather than leaving them as abstract ideas. Finally, regarding the practical artefacts, most of the implemented guidelines follow explicit translations from theory into system design. However, the creation of vistas has a more subjective nature. A collection of the generated vistas can be found in Appendix C.

6.1.3 Ethical Concerns

The majority of ethical concerns of this thesis arise from the usage of Procedural Content Generation. The most immediate effect is that, due to the integration of randomness, the design of the generation algorithms in the game can lead to unintended gameplay scenarios. These can be unbalanced situations or unexpected events that might be detrimental to the intended gameplay experience, even when these are rare occurrences. These unexpected results can range from situations in which the game is impossible to beat, to scenarios where harmful goals are proposed to the player (e.g. in games with a family-friendly context). A studio or company may not be aware of these problems, or they can succumb to lazy design and ignore them. The remark this thesis tries to make against this concern is that game design (level design in our case) can heavily influence the implementation of PCG, and that it should align with the intended game experience.

Another ethical issue that is closely related to the previous one is the existence of harmful biases [42]. PCG is usually informed by somewhat basic rules to generate a variety of scenarios, but these rules can be based on the designer's biases. This effect can be more or less noticeable, e.g. in how specific level layouts are structured in a way that NPC minorities are secluded or different ethnicities are segregated in a procedural city. It can also be more explicit in games where quests and characters are procedurally generated. It is up to the designers to be aware of this and look for these ethical issues and prevent any offensive content from being generated. One recent example is generative AI, which some game studies started integrating into their products [43]. This is clearly dangerous, since there are numerous reports of people managing to get these AIs to be toxic or produce harmful content, even when they were designed to explicitly avoid it [44].

A specific example of this integration is the Mantella mod for *Skyrim (2011)* [45], where NPCs are context-aware of their surroundings and some game elements, and they can memorise all the conversations they have with the player. With this mod, players are allowed to alter the behaviour and narrative of the characters to the point where hate speech can be generated.

Another less noticeable but equally important ethical concern is the job displacement. The usage of PCG in video game development can lead to less people being required to work on hand-crafting assets or levels [46]. People are still required to design and implement the procedural algorithms, but these jobs rely on a more technical skill set. Furthermore, there can be an increase of expectations over smaller teams that use PCG since once a content generator has been implemented it produces assets at a faster rate than a traditional team would do. This is usually related to how the roles that are not in charge of this design or implementation are not aware of the complexity of these systems. This can lead to development teams suffering from tight deadlines and crunch, which eventually lead to massive occupational burnouts and people dropping their jobs for health issues in extreme cases [47]. This thesis can help slightly mitigate this issue by portraying how much effort is required to implement the simple design rules integrated in the prototypes.

6.1.4 Meaning in Procedurally-Generated Worlds

Meaning is a crucial sub-guideline that serves engagement. It refers to the sense of purpose that players experience while playing, and the feeling that what they are doing in the game matters. Meaning can be born from the connections players experience, and these can be cognitive, personal, emotional, or spatial. Level design particularly focuses on the spatial category, which occurs when meaning is embedded into spaces [48]. The particular tools level design uses to create this type of meaning are *environmental storytelling* and *Psychogeography*.

This becomes an interesting discussion topic when introducing procedural content generation in the development process, since these two aforementioned tools heavily rely on narrative and author intentionality [49], which are two particular weaknesses of PCG. This complicates the embedding of meaning in spaces when using this kind of algorithmic approaches.

In terms of narrative, non-authored generated content may reduce perceived value or engagement in players. This is further to be explored, but a running debate topic with the recent rise of generative AI discusses how the perception of authorship can affect how meaning is constructed. Experimental evidence indicates that consumers evaluate AI-generated art more negatively than human-made art, suggesting that the source of creative output impacts perceived meaning [50]. In terms of author intentionality, it becomes a diffuse topic in the context of PCG. Here, there is still authorship over the intended experiences, but the control is indirect. Game developers design the generation rules instead of the specific instances, which can lead to unexpected or unintended results that deviate from the author intent.

This far, we have only referred to meaning as something that must be provided by the author, but interpretative meaning changes the focus from the author to the consumer. This reframes PCG from a model that lacks author intent to one that redistributes where the origin of meaning lies. This framing could work better along the use of procedural generation since, despite the lack of precise control, the author can instead create the possibility space that can lead to opportunities for the players to invest their own meaning in the worlds they explore. Author intentionality can be important here too, since although meaning here emerges from the consumer, their interpretative engagement can be reduced if they perceive this lack of human authorship [50].

From a more pragmatic perspective, the implications of this discussion can serve level designers better understand the implications of meaning, and how to ultimately and effectively integrate it into their worlds and systems. This discussion helps explain the trade-off between authored control and emergent meaning, although a more developed answer would be required for the fully developed guideline.

6.2 Future Work

This thesis managed to explore all the literature required to answer its research question. However, the thesis only *gathered* this knowledge, but it did not provide a conclusive answer. This answer would require a more analytical approach, and more formally defined artefacts.

Most of the research that belongs to the future work lies on the theoretical artefact: the guidelines. First of all, they would require to be fully developed, in the same way that it has been done for the first two (Goals and Engagement) in Appendix A. A particularly interesting addition would be an in-depth catalogue of examples that show how different implementations of each guideline evoke different game aesthetics. One example could explain how the frequency of goals creates different sensations, e.g. in a game where you are a bartender, a lack of customers could expose the player to apathy, while, an infinite hoard of customers could induce distress. To improve the guidelines, it would also be useful to have examples of their presence in games, which could be achieved through crowdsourcing, since one has to be familiar with a game to state if it implements said guidelines. Furthermore, the missing concepts and design practices mentioned in the takeaways (Section 6.1.1) should be integrated in the final product. The guidelines should also be validated through different means, like an MDA analysis over the listed games, or surveys to explore the effect these patterns have in the players themselves. Professional developers should also be contacted to directly provide their insights and opinions of the value of this knowledge. Additionally, different displays of this knowledge could be explored by rearranging their conceptual categories and their sorting. One particular concept that shows how this might be necessary is uncertainty. This concept is used in two guidelines: it is a tool used to apply guidance through the player's **curiosity**, but it is also an attribute of **engagement**. This concept showing up in two different guidelines can be used as a hint to indicate that a different arrangement of the concepts could help in their understanding.

As for the technical artefacts, their future work involves revisiting their current limitations listed in Section 6.1.1. This could involve either implementing a playable prototype with metrics and validation systems, like playtests, or the implementation of a quest generator with graphs that account for spatial pathing techniques, such as vistas, shortcuts, valves, sightlines, branching, and more in a greyboxed level.

6.3 Closing Remarks

Coming back to the research question: 'How can we influence game experiences in Open-World games through level design using procedural content generation?'. The answer that can be provided with the findings of this thesis is quite fragmented. For this purpose we need to acknowledge the four pillars and understand the connections between them. We first explored how game experiences could be defined and analysed through the MDA framework. Then, we described what constitutes an open-world game by defining common characteristics they share, like exploration and non-linear objectives. Here we identified that 'open world' does not constitute a game genre but rather acts as a qualifier on top of other genres. Level design acts as the linking pillar that unifies everything together. It connects to game experience through the concept of *goal negotiation*, where the act of distributing the game objectives throughout the world incentivises this targeted experience. The level design guidelines can then be used to implement the intended experience by following the pertinent rules that apply to open-world games. Finally, the implementation of these rules would be applied by translating them into algorithms thanks to the taxonomy and list of methods provided for PCG.

This could work as a tentative answer, since the guidelines have been constructed around the definition of game experiences, and all of them describe how they impact them. Particularly, the guidelines provided a concise but flexible definition on how to construct goals and engagement, while at the same time they offered example consequences of implementing them one way or another in the 'How to apply' and 'Counter effects' sections. Additionally, this thesis briefly showed some examples on how these rules can be translated into algorithms by using procedural generation. With this we can see how there is an answer to the question, but it will always depend on the type of experience the game designer wants to construct. There is also no definitive way to translate the level design rules into algorithms. Their implementation heavily depend on what rules you are combining, and what is your intended game experience. A complete set of examples for each of the guidelines could help refine this answer.

While a conclusive answer to this research question has not been provided due to the missing implementation of the guidelines, this thesis remarks the importance of this topic and points out the existing gap in the literature. Additionally, it sets the foundation of knowledge from which to build further investigations in many different directions. This could consist of deepening the current research of level design and extending the guidelines with player surveys, industry professionals' insights, and examples with different game experiences in mind, or taking a more technical approach and providing implementation examples making use of PCG.

Acronyms

AI	Artificial Intelligence
BOTW	(Zelda) Breath of the Wild
BSP	Binary Space Partitioning
CCL	Connected-Component Labelling
CDF	Cumulative Distribution Function
GDC	Game Developer Conference
MDA	Mechanics, Dynamics, Aesthetics
NPC	Non-Playable Character
POI	Point Of Interest
PCG	Procedural Content Generation
RNG	Random Number Generator
RPG	Role-Play Game
RtD	Research through Design
WFC	Wave Function Collapse

Ludography

- .kkrieger (2004)
- Adventure (1979)
- Assassin's Creed saga (2007)
- The Binding of Isaac (2011)
- Bloodborne (2015)
- Celeste (2018)
- Chants of Sennaar (2023)
- Colossal Cave Adventure (1976)
- Comanche: Maximum Overkill (1992)
- Cookie Clicker (2013)
- Dead Cells (2017)
- Death Stranding (2019)
- Doom series (1993)
- Elden Ring (2022)
- Enter the Gungeon (2016)
- The Evil Within series (2014)
- Firewatch (2016)
- Goat Simulator (2014)
- GTA series (1997)
- Hogwarts Legacy (2023)
- Hollow Knight: Silksong (2025)
- Hytale (EA 2026)
- Journey (2012)
- The Last of Us series (2013)
- The Legend of Zelda series (1986)
- Minecraft (2011)
- Mirror's Edge (2008)
- Noita (2019)
- Outer Wilds (2019)
- Pitfall! (1982)
- Rain World (2017)
- Sable (2021)
- Satisfactory (2024)
- Sekiro (2019)
- Skyrim (2011)
- Spelunky (2008)
- Subnautica (2018)
- Sunset Overdrive (2014)
- Teardown (2022)
- Uncharted series (2007)
- Universal Paperclips (2017)

– The Witness (2016)
– XCOM 2 (2016)

Level Design Glossary

Action Block – A modular piece of a level that has a focus on a very specific mechanic or challenge and that represents a specific gameplay idea. It then gets used as a building block to create levels.

Affordances – A concept that explains how recurrent and consistent visual cues in the environment are used to signal possible game interactions to the player.

Anchors – Small level elements that help the player recognise and orient themselves in a space.

Backtracking – Concept used to refer to gameplay instances where the player has to traverse backwards a path that they have already explored to get to a previous part of a level.

Bidirectional Gameplay – A level layout that supports different gameplay interactions when traversed in multiple directions, usually applied in backtracking routes.

Breadcrumbs – Subtle environmental cues or rewards placed along a route to lure the player through intended paths and experiences. They are common in open spaces and as indicators to secret paths.

Choke Point – A specific node in the level layout that converges many paths to a single one, constraining the openness of the space to enable control over specific progression bits.

Clear Goals – Concept that explains how objectives should be stated clear to the player so that they understand what they are intended to do to progress in the game.

Cognitive Map – The mental representation a player constructs of a game's spatial layout. This is a really useful tool that can be used to measure how players orient themselves in a given space.

Corners – Layout technique used to hide game elements from the player by using the corners of buildings or hallways. This serves the purpose of pacing the information provided to the player and also inducing curiosity.

Critical Path – The intended route or set of actions that the player has to take in order to achieve a goal in the game. The concept is usually brought up when talking about the importance of its existence.

Denial Space – A part of a level where the goal is visually lost to create 'pockets of tension'. Resolving this situation by revealing the goal again can be used as a reward or a resource for pacing.

Difficulty Curve – A metric that can be used to understand how the challenge presented by the game escalates over time in a game. This is represented in a line plot, from which we can extrapolate how fair the escalation of challenge is by measuring the steepness of the line.

Emergent Gameplay – Unplanned or unexpected player behaviours arising from interactions with the game systems. It holds a positive connotation, usually tied to the potential of infinite play.

Environmental Storytelling – Concept used to refer to levels conveying narrative information (and usually lore) through spatial arrangements and the placement of level elements instead of explicit exposition.

Foreshadowing – Environmental or narrative hints that signal future events or challenges.

Funnelling – Guidance technique that uses spatial composition to subtly direct players toward a specific location or objective while maintaining perceived freedom. This spatial structure usually precedes choke points.

Gates and Locks – A type of mechanical or spatial obstacle that restricts progression until certain conditions are met. This can range from the typical locked door with a respective key, to the use of mechanics like requiring a lantern to get through a dark cave.

Goal Layering – Game design technique of presenting multiple concurrent objectives of varying scope to support flexible exploration and sustained motivation.

Goal Negotiation – The process by which players decide what type of activity to engage with by assessing the goals and systems the game presents.

Greyboxing – Early level design phase that uses simple geometric shapes to prototype spatial layouts and convey the general structure and flows of a space, prior to integrating detailed level art.

Landmarks – A large type of anchor with a distinctive visual appearance that helps players orient themselves in a space. It can usually be distinguished from afar, and it heavily supports cognitive maps.

Metrics – Term used in early level design that refers to the practice of measuring distances and spatial relations of how the different interactions the player can have with the level. The most common ones relate to character controllers, like jump height or the height of coverages and attack range in shooters.

Onboarding Space – Space where the player first spawns in a level. Usually tied to tutorialization or introduction to new mechanics or themes.

One-way Valve – A pathing technique used to allows traversal in one direction only, restricting the player from accessing previous parts of a level to push progression forward.

Pacing – Metric used to define the speed at which experiences unfold over time. It defines how quickly events, information, or actions are presented.

Pinch Points – This is a type of funnelling technique that is specifically used to direct the player camera and point of view towards a specific vista or set piece.

Player Agency – The capacity of the player to autonomously make meaningful choices that influence outcomes.

Player Desire – Concept that defines the inception of the motivation that drives the player's actions. It can be of three types: desire of direct consequence, desire of direct emotional response, and desire of sensation of accomplishment.

Playgrounds – Open spaces designed to encourage experimentation through the interaction with systems, rather than goal-directed navigation. The difference with sandbox spaces is that in playgrounds the systems are integrated into the space. In sandbox levels you play in the space, in playgrounds you play with the space.

Point Of Interest (POI) – Location that attracts player attention. It usually has a strong visual impact tied to the importance of the location. They serve as anchors, and usually provide some kind of goals to the player.

Psychogeography – The study of how spatial arrangements and features of a space are shaped by the people that interacts with said space. These spaces induce strong emotional and behavioural responses, since people can relate and connect to them. This concept has been adopted in games and translated through the concept of environmental storytelling.

Rewards – Consequences of objectives that act as incentives for the player to work on achieving them.

Risk VS Reward – Concept that explains how goals that escalate challenge should be matched with a corresponding escalated reward. This concept is specially strong in combination with goal layering, where the player can decide what objective to engage with considering both the risk and the reward involved.

Safety Nets – Level design technique that reduces the impact of failed objectives so that players can retry them without going through a loading screen. An example would be a player jumping over a gap, but instead of a lava pit or getting voided out, they get some stairs that take them back to try the jump again.

Safe Zones – Areas of reduced threat or challenge that enable recovery, planning, and emotional decompression, helping control the pacing of the game.

Set Pieces – Scripted or designed high-impact moments in a game that deliver spectacle, emotional intensity, or narrative significance.

Sightlines – Concept used to describe the points from which the player has direct visual connection with goals or relevant level elements. These are usually utilised to reveal or reinforce important spatial information, orientation, or frame points of interest and goals.

Spatial Hierarchy – Concept that explains how the structure of levels and spaces follows a recursive nature, where layout patterns repeat at different scales.

Spatial Lures – A collection of artistic elements that can be used in level design to attract player's attention by creating contrast in the environment. Some examples include lighting, colour, movement, sound, etc.

Spatial Occlusion – A concept that explains how to create tension by partially or totally hiding elements of a level that are recognisable by the player. This tension is then resolved by revealing back the previously hidden elements.

Signposting – The deliberate use of visual, spatial, or systemic cues to direct the player's attention and communicate intended routes or interactions.

Triangle Rule – A compositional technique that derives from the concept of spatial occlusion. It consist of partially hiding points of interest behind other level elements (such as mountains, hence the triangle) to create mystery.

Vistas – Large-scale vantage point that reveals distant goals, landmarks, or layouts, reinforcing orientation and long-term navigation.

Bibliography

[1] N. G. J. Hughes, "Understanding specific gaming experiences: The case of open world games," https://etheses.whiterose.ac.uk/33608/, Ph.D. dissertation, University of York, 2023.

[2] R. Hunicke, M. Leblanc, and R. Zubek, "A formal approach to game design and game research," in *Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*, 2004.

[3] N. E. M. Vickery and P. Wyeth, "Exploration in open-world videogames: Environment, items, locations, quests, and combat in the witcher 3," in *Proceedings of the 34th Australian Conference on Human-Computer Interaction*, New York, NY, USA: Association for Computing Machinery, 2023, pp. 310–318, ISBN: 9798400700248. [Online]. Available: https://doi.org/10.1145/3572921.3572926.

[4] P. Field, "Spatial communication in level design," in *Develop Digital*, https://www.youtube.com/watch?v=AKeUZVikPV8, 2020.

[5] R. I. Zubek, *Elements of Game Design*. Cambridge, MA: MIT Press, 2020.

[6] J. Burgess, *Skyrim's modular level design*, Game Developers Conference, 2013. [Online]. Available: http://blog.joelburgess.com/2013/04/skyrims-modular-level-design-gdc-2013.html.

[7] C. W. Totten, *An Architectural Approach to Level Design*, 2nd ed. CRC Press, 2019.

[8] E. Adams, *Fundamentals of Game Design*, 3rd ed. Berkeley, CA: New Riders, 2014.

[9] T. Fullerton, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*, 4th ed. Boca Raton, FL: CRC Press, 2018.

[10] J. Schell, *The Art of Game Design: A Book of Lenses*, 3rd ed. CRC Press, 2019.

[11] M. e. a. Hendrikx, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1, Feb. 2013, ISSN: 1551-6857. DOI: 10.1145/2422956.2422957.

[12] D. Grey, "When and why to use procedural generation," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 1.

[13] M. R. Johnson, "Integrating procedural and handmade level design," in *Level design: processes and experiences*, CRC Press, 2017, ch. 11, pp. 217–242.

[14] M. R. Johnson, "Worlds," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 10.

[15] L. O. Valencia-Rosado and O. Starostenko, "Methods for procedural terrain generation: A review," in *Pattern Recognition*, Cham: Springer International Publishing, 2019, pp. 58–67, ISBN: 978-3-030-21077-9. [Online]. Available: https://www.doi.org/10.1007/978-3-030-21077-9_6.

[16] G. Smith, "Pcg overview," in *Level Design Processes and Experiences*, CRC Press, 2017, ch. 8.

[17] B. Bucklew, "Algorithms and approaches," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 26.

[18] R. Montserrat, "Procedural content generation for video games, a friendly approach," *Level Up [Game Dev Hub]*, 2025, https://www.levelup-gamedevhub.com/en/news/procedural-content-generation-for-video-games-a-friendly-approach/.

[19] S. Björk and J. Holopainen, *Patterns in Game Design*. Hingham, MA: Charles River Media, 2005.

[20] C. Alexander, *The Timeless Way of Building*. New York, NY: Oxford University Press, 1979.

[21] M. Barclay. "My level design guidelines." https://mikebarclay.co.uk/my-level-design-guidelines/. (2016).

[22] J. Burgess, "Level design planning for open-world games," in *Level design: processes and experiences*, CRC Press, 2017, ch. 12, pp. 243–272.

[23] J. Machina. "Define primero, diseña tu nivel después." (2020), [Online]. Available: https://youtu.be/1nq-FemmLrM.

[24] J. Grinblat, "Designing for modularity," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 4.

[25] L. Welton, "Aesthetics in pcg," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 3.

[26] G. Programmer. "Procedural generation in games." (2022), [Online]. Available: https://generalistprogrammer.com/procedural-generation-games.

[27] J. Segree, *The case for procedural generation in puzzle games*, 2024. [Online]. Available: https://www.youtube.com/watch?v=xqmTp20juZk.

[28] B. W. Head and J. Alford, "Wicked problems," *Administration & Society*, pp. 711–739, 2013, https://doi.org/10.1177/0095399713481601.

[29] R. A. Stebbins, *Exploratory Research in the Social Sciences*. Thousand Oaks, CA: SAGE Publications, 2001.

[30] E. L. Deci and R. M. Ryan, "The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior," *Psychological Inquiry*, 2000.

[31] L. von Bertalanffy, *General System Theory: Foundations, Development, Applications*. New York, NY: George Braziller, 1968.

[32] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. New York, NY: Scribner, 2001.

[33] B. L. Whorf, *Language, Thought, and Reality*. Cambridge, MA: MIT Press, 1956.

[34] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. New York, NY: Oxford University Press, 1977.

[35] D. A. Norman, *The Design of Everyday Things*. New York, NY: Basic Books, 2013.

[36] J. Zimmerman and J. Forlizzi, "Research through design in hci," in *Ways of Knowing in HCI*, J. S. Olson and W. A. Kellogg, Eds., Springer New York, 2014, pp. 167–189, ISBN: 978-1-4939-0378-8. DOI: 10.1007/978-1-4939-0378-8_8.

[37] C. Fosnot, *Constructivism: Theory, Perspectives, and Practice*. Teachers College Press, 2005, ISBN: 9780807745700. [Online]. Available: https://books.google.se/books?id=rvnFQgAACAAJ.

[38] Z. Munn, M. Peters, and C. e. a. Stern, "Systematic review or scoping review. guidance for authors when choosing between a systematic or scoping review approach," *BMC Med Res Methodol*, 2018, https://doi.org/10.1186/s12874-018-0611-x.

[39] S. Iyengar and M. Lepper, "When choice is demotivating: Can one desire too much of a good thing?" *Journal of Personality and Social Psychology*, vol. 79, pp. 995–1006, Jan. 2001. DOI: 10.1037/0022-3514.79.6.995.

[40] R.-Y. Storm, *Spatial documentation: Gyms, zoos, and museums*, YouTube video, https://youtu.be/5PJRCz0t7yY, 2021.

[41] S. L. Erica Xu, *Obsidian*, Personal knowledge management software, 2020. [Online]. Available: https://obsidian.md.

[42] M. Zhou, V. Abhishek, T. Derdenger, J. Kim, and K. Srinivasan, *Bias in generative ai*, https://arxiv.org/abs/2403.02726, 2024. arXiv: 2403.02726 [econ.GN].

[43] S. Buongiorno, L. Klinkert, Z. Zhuang, T. Chawla, and C. Clark, "Pangea: Procedural artificial narrative using generative ai for turn-based, role-playing video games," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 20, no. 1, pp. 156–166, Nov. 2024, ISSN: 2326-909X. DOI: 10.1609/aiide.v20i1.31876.

[44] C. Xiang, "People are jailbreaking chatgpt to make it endorse racism, conspiracies," *VICE*, 2024, https://www.vice.com/en/article/people-are-jailbreaking-chatgpt-to-make-it-endorse-racism-conspiracies.

[45] d&f et al, *Mantella - bring npcs to life with ai*, https://art-from-the-machine.github.io/Mantella/, 2024.

[46] S. Palvel, "Building worlds with code: The magic of procedural content generation in ai," *Medium*, 2024, https://subashpalvel.medium.com/building-worlds-with-code-the-magic-of-procedural-content-generation-in-ai-698a34b7f8ad.

[47] S. Anderson and S. Orme, "Mental health, illness, crunch, and burnout: Discourses in video games culture," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, https://doi.org/10.24251/hicss.2022.385, 2022.

[48] D. M. R. Johnson, "Meaning," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 27, pp. 301–311.

[49] L. Andreotti and X. Costa, *Theory of the dérive and other situationist writings on the city*. Museu dArt Contemporani de Barcelona eBooks, 1996.

[50] F. Tigre Moura, C. Castrucci, and C. Hindley, "Artificial intelligence creates art? an experimental investigation of value and creativity perceptions," *The Journal of Creative Behavior*, vol. 57, no. 4, pp. 534–549, 2023. DOI: https://

doi.org/10.1002/jocb.600. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jocb.600. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jocb.600.

[51] B. Wang, Z. Gao, and M. Shidujaman, "Meaningful place: A phenomenological approach to the design of spatial experience in open-world games," *Games and Culture*, vol. 19, no. 5, pp. 587–610, 2024. [Online]. Available: https://doi.org/10.1177/15554120231171290.

[52] A. Khalifa, F. de Mesentier Silva, and J. Togelius, "Level design patterns in 2d games," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8. DOI: 10.1109/CIG.2019.8847953.

[53] S. Rogers, "Levels," in **Level Up! The guide to great video game design**, John Wiley & Sons Inc, 2014, ch. 9.

[54] J. O. Fares Kayali, "Level design practices for independent games," in *Level Design Processes and Experiences*, CRC Press, 2017, ch. 6.

[55] B. C. João Raza, "The illusion of choice - how to hide linearity of levels," in *Level Design Processes and Experiences*, CRC Press, 2017, ch. 10.

[56] D. J. Dormans, "Graphs and cycles," in *Procedural Generation in Game Design*, CRC Press, 2017, ch. 9.

[57] E. Games. "Level design fundamentals." (2023), [Online]. Available: https://dev.epicgames.com/community/learning/tutorials/3VKJ/unreal-engine-fortnite-level-design-fundamentals.

[58] R. Y. Smith. "The level design book." (2023), [Online]. Available: https://book.leveldesignbook.com.

[59] D. Taylor, "Ten principles for good level design," in *GDC*, https://gdcvault.com/play/1017803/Ten-Principles-for-Good-Level, 2013.

[60] S. Lee, "An approach to holistic level design," in *GDC*, https://gdcvault.com/play/1024301/Level-Design-Workshop-An-Approach, 2017.

[61] A. Serr, "Designing radically non-linear single player levels," in *GDC*, https://gdcvault.com/play/1026398/Level-Design-Workshop-Designing-Radically, 2021.

[62] N. Oueijan, "Stop getting lost: Make cognitive maps, not levels," in *GDC*, https://gdcvault.com/play/1027206/Stop-Getting-Lost-Make-Cognitive, 2022.

[63] A. Yoder, "Playgrounds and level design," in *GDC*, https://gdcvault.com/play/1025884/Level-Design-Workshop-Real-World, 2019.

[64] Nintendo EPD, *The making of the legend of zelda: Breath of the wild*, Developer Interview, 2017. [Online]. Available: https://youtu.be/30jGWna4-Ns.

[65] K. Beachum, "Sparking curiosity-driven exploration through narrative in 'outer wilds'," in *GDC*, https://gdcvault.com/play/1027368/Independent-Games-Summit-Sparking-Curiosity, 2021.

[66] A. Beachum. "Designing for curiosity in outer wilds." https://youtu.be/vGnce1Dp9BU. (2023).

A

Example of 2 Guidelines

A.1 Goals

A.1.1 Description

What is it? Goals refer to any objective or activity the player has to perform that comes from the **desire** to achieve something in the game. These goals can be presented by the game or can be self-imposed by the player. The conversation the player has with the game to discuss which goals are viable through the game's systems is called **goal negotiation**. Goals are in charge of providing the players a reason and/or examples to use the **core mechanics** the game systems offer.

This guideline is closer to game design than it is to level design, but it is important for level designers to know about them. This is because it is up to level designers to distribute these goals in such a way so that they align with the intended game experience. Goals can arise from several types of desires. These desires can overlap, since objectives can be born from multiple desires at once. There are three types of desires:

- Desire of a direct consequence of performing an activity. This could be any type of action where the player would get a reward or achieve something in exchange of performing a task. This desire also applies in the scenarios where players do not know what the consequences of their actions might be, supporting the **Curiosity** guideline. Some examples include: fighting enemies to level up, gathering resources, building the defences of a base, or progressing the narrative.
- Sense of accomplishment for successfully performing a task. This is usually related to challenges and mastery of the mechanics. This type of desire includes tasks such as defeating a difficult boss, climbing a ridge, gathering all collectibles, or even completing the game without receiving damage.
- Desire of direct emotional response. This is explained by players seeking specific emotions when performing actions, such as climbing a ridge to watch a sunset, or fighting enemies because they enjoy the combat mechanics. This can also be seen from a more cognitive or personal level, where players might act in specific ways drawing the magic circle closer to reality. This can be the case of being kind to NPCs or following the traffic rules in a *GTA* game.

As you can see, the same action (e.g. fighting an enemy) can come from different types of desire. The player might want to gain experience, they may want to prove to themselves they are capable of defeating such enemy, or they might just enjoy the combat mechanics. Designers should seek how to motivate the intended behaviours through these types of desires.

What it achieves/focuses on. This guideline is strongly tied with the MDA (Mechanics, Dynamics, Aesthetics) framework. Goals (which are enabled by mechanics), *motivate* behaviour (dynamics), which then *evokes* emotions and feelings (aesthetic). This is one of the most relevant guidelines, since this ***negotiation of goals*** is the main driver of the **experience** the player will have with the game.

How to apply.

- **Seeking the intended experience.** One approach game designers have to bring game experiences to life is following the MDA framework. This can be done by first thinking about the mechanics, which then enable dynamics and evoke the aesthetics. The process can also be applied in reverse, where designers first seek a specific aesthetic, and then think what type of behaviours and mechanics would contribute to that experience.

  It is important to note that when designers are working towards an intended player experience, it is better to **encourage** a wanted behaviour rather than disabling an unwanted one. This is because the coerced freedom of the player can be perceived as an obstacle to play however they want to. There is an open debate in the industry whether designers should impose their intended experience to players or not. This has been a hot debate with topics like the difficulty of the *FromSoftware* games. One specific example is *XCOM2 (2016)*, a turn-based tactics game where some levels would present a maximum number of turns to **enforce** a more risky playstyle. Many players did not enjoy this restriction, since they preferred playing in a way where they could minimize risk. This feature was so disliked that the community implemented mods to overcome this restriction.

  One way to deal with this discussion between players and designers is to clearly state what the intended experience is without forcing the players to follow it. This can be done through accessibility modes as it is seen in *Celeste (2018)*, where you can enable settings like infinite dash or no stamina consumption, and *Satisfactory (2019)*, where you can completely disable the enemies hostility. However, this should be more of a last resource, since they can bring to the table more development time.

  "One of the responsibilities I think we have as designers is to protect the player from themselves" (Sid Meier) since "given the opportunity, players will optimize the fun out of a game" (Soren Johnson). Players might decide to play a game in a specific way that is less fun just because it guarantees success with a goal, or simply because that is a faster method. As a designer, make sure that whenever you are seeking an intended experience, do it in such a way that you find a balance between these two aspects.

- **Layering Goals**. *Player Agency* is vital to most game genres. This is achieved through **choices**. In relation to goals, this is done through presenting several goals at the same time, so players can choose which one to engage with. These goals should present some *variety* [**Engagement**] so that the choices the player makes are meaningful. This variety can be applied through: core mechanics involved, duration (long/short-term goals), relevance (main/secondary missions), and difficulty, among others.

- **Clear Goals**. This refers to the need of explicitly stating to the player what are the current goals, and frequently reminding them about their existence and purpose. Although games with an open-ended nature allow for a broader selection of goals, players usually need some guidance to not feel completely lost. One great example is *Outer Wilds (2019)*, where developers learned through playtesting that they needed to make sure to clearly present their soft goals at the beginning of the game. Another example of implementation is *landmarks*, which can tie a level element to progression, and then present it frequently in the *sightlines* of the player.

- **Rewards**. Goals have rewards. The player needs to feel some accomplishment after achieving a goal, and this is done through rewards. These can be framed from the *desires* definition, where every goal is born from the desire to achieve something. These can be explicit as levelling up, progressing in the narrative, or defeating a tough enemy; or they can be more abstract, such as fulfilling their **Curiosity** or expecting a specific emotional reaction. It is fair to say that players' self-imposed goals are more difficult to account for but, as designers, we should try to cover as many plausible desires as possible to the extent of our capabilities and resources.

Counter Effects. The negotiation of goals is a key element of level design. Too few options can lead to restrictive gameplay and make the game feel too linear. Beware of the opposite as well, too many goals can become overwhelming and players might lose track of the more important ones as their attention gets diverted into less interesting tasks.

A.1.2 Real Industry Examples

- **Intended Experience**. You can achieve the effect of encouraging specific playstyles (and hence enabling intended experiences) by using clever game and level design tricks. One such example is *Doom (2016)*, where they designed the game for a more aggressive playstyle where the player should always be in a rampage of violence. In this game they implemented the 'glory kills', a mechanic where killing demons using a melee attack rewards the player with health packs and ammunition. Another example is *Bloodborne (2015)*, where the 'rally' mechanic allows the player to regain some health they just lost if they become more aggressive and hit back the enemy that just damaged them. One last example is the *Uncharted* series, where enemies can destroy the player's cover by throwing grenades if they are playing it too safe.

- **Layering Goals**. *Zelda BOTW (2017)* always presents many different objectives simultaneously, such as enemy camps, shrines, areas to gather resources (e.g. hunting or fishing), villages, wandering NPCs, stables... As a counter example, *Zelda Skyward Sword (2011)* presents only the main goal overlaid with a few lesser secondary tasks such as bug hunting. The ratings of this second game have been controversial. People have deemed it to be too linear, since it does not present many chances to choose what to engage with in its world.

- **Clear Goals**. The *The Last of Us* series (2013, 2020) always conducts players to specific locations, and they make sure to explicitly state this at the beginning of each level. This is clearly marked and **reinforced** through the level with a *landmark* pointing at the destination (e.g. the museum, the bridge, the science building, the hospital, etc.). One example of a game that does not present clear goals is *Sable (2021)*. This game gives the player **absolute freedom** after the tutorial area is completed. The player is given the soft goal of getting one specific key item from anywhere in the world, and then they can decide to come back to finish the game or continue exploring on their own. Although the premise of complete freedom to explore different cultures and civilizations can be intriguing, many players showed discontent towards this game since there was no apparent reason for them to do so.

- **Rewards**. Another problem *Sable (2021)* had was underwhelming rewards after completing the different goals, which could make players feel discouraged from seeking more activities in its open world. As a counter-example, *Outer Wilds (2019)* managed to reward players that showed curiosity by providing them with answers to their mysteries, and unlocking new investigation threads and mechanics to use on further expeditions.

A.1.3 Metrics and Validation

Create a list of all your explicit goals, and how they belong to the game chronologically and spatially. With this, you can run an MDA analysis over your main and secondary goals. See what systems are involved and playtest how players react to those to see if they align with your intended experience. You can also compare how many possible goals players have at their disposal throughout the game both timely and spatially. You could aim to have a main goal, a few secondary ones, and lesser goals distributed sparsely. Go through the list and make sure every goal has a reward or a reason to be completed. Players can find it very frustrating to complete a task without being rewarded afterwards.

Ensure you have some means to remind the player of where they can find or follow their goals. This can be done, for example, through a notebook used as a registry (*Zelda Majora's Mask (2000)*), or by spreading NPCs across the land to talk about the relevance of those tasks (*Zelda: Tears of the Kingdom (2023)*).

A.1.4 Related to

- **Engagement, Choices, Meaning, Pacing**. Goals are responsible of providing the player with a gameplay experience through a display of the ***core mechanics*** and systems. Layering of goals should offer variety, risk VS reward, and choices. Rewards can act as consequences of choices. Goal negotiation is the main conductor of pacing, dictating which type of interaction the player can have with the game at any point. Furthermore, layering goals allows players to self-pace their game sessions.
- **Guidance - POIs**. POIs are a great source of goals, both presented by the game and self-imposed by the player. e.g. a player sees a *landmark* in the distance and decides to go there, which leads them to finding an NPC that provides a quest.
- **Guidance - Curiosity**. Goals can arise from the player's desire to satisfy their curiosity.

A.2 Engagement

This is the first guideline that is divided into different sub-guidelines. It provides a general definition of what engagement is constituted of, along with some brief introductions to the sub-guidelines. It also contains explanations on some other shallower concepts. While the description section will go over all of these concepts, the rest of the sections will only focus on the latter concepts, since the former already have a dedicated in-depth guideline for each of them.

A.2.1 Description

What is it? Engagement describes how effectively a level or a game holds the players attention and motivates them to keep playing. Engagement is built upon three major concepts.

- **Choices**. This relates to *player agency*, where the player decides what type of activity they want to engage with, or what approach they want to take to face a task. This is strongly tied with *goal negotiation*, a key concept that helps drive the intended experience. Unlike cinema, literature, or other kinds of art, interact-ability (highlighted by choices) is what distinguishes games from the rest of the mediums. Choices are deeply entangled with many other concepts in these guidelines, such as *player motivation*, *variety*, ***Points Of Interest***, and **Pathing**.
- **Meaning**. This term refers to the sense of purpose that players experience while playing. It consists of the feeling that what they are doing in the game matters, whether to the story, to other players, or to themselves. Meaning can be born from **connections** of many kinds: cognitive (detecting patterns and solving problems), spatial (when meaning is embedded in spaces), personal (reflecting on ourselves), and emotional (attachment).

- **Pacing**. Pacing is the speed at which experiences unfold over time. It defines how quickly events, information, or actions are presented. Pacing can be measured in many different aspects of a game: narrative, challenge, mechanics involved, emotional charge, etc. A well-tuned pacing maintains **curiosity**, focus, and emotional investment.

These are the main concepts that shape engagement in a game, but there are some other related concepts that are widely mentioned in the industry and that are worth looking at, since they form part of the vocabulary of players, game designers, and researchers alike.

- **Challenge**. This measures how difficult the presented goal is in relation to the skill of the player. Games have to be designed with a player base in mind, but it is also a common practice to try to reach for broad audiences, in which tuning the challenge is a more difficult task. An objective that is too easy for the player results in boredom or disinterest, while the opposite case can lead to frustration or even induce anxiety. A good balance of challenge helps maintain the player engaged with the different activities.
- **Unpredictability**. This concept helps shape engagement by influencing the player's **Curiosity**. It describes how the expectancy of the outcome of an event (e.g. not knowing how an entity can react to your presence) can get the player invested and create the *desire* to figure it out.
- **Variety**. This refers to the idea that a game should present a wide variety of elements to remain engaging. This concept can be applied in every aspect of a game: game mechanics, themes, environmental art, openness of a level, density of goals, etc. A good variety can keep the experience of a game fresh and avoid repetitiveness.
- **Game Feel**. This is an abstract and intangible concept that refers to a satisfying sensation experienced when playing games, which is usually linked to responsiveness. This responsiveness reflects on the reaction the game has when a player performs an action. It is best explained through an example. If your game consists of pressing a button, this can be mechanically achieved by just reading the input from the mouse button. However, game feel explains how to transform this simple action into something engaging, by giving this click some visual and auditory impact. This can be done through a satisfying clicking sound, animating the button bouncing with the click, adding particles and camera shakes, etc. This concept is usually closely tied with game design or game art, but level design can have this kind of impact too.

What it achieves/focuses on

All of these concepts have one goal in mind: to get the player to keep playing the game. Players usually have a preconception of what the game is when they are engaging with it. This means that they have an expectation of what the game experience is going to be, and they have some **motivation** to look forward to it. These concepts reassure the player and keeps them interacting with the game. Furthermore, a game is usually structured around fighting boredom and avoiding passivity.

There are some exceptions, such as narrative-driven games, where the only type of interactability is the dialogue options the player can choose from or *quick time events*. However, these games still present the aforementioned concepts, but instead of through game systems they do so through narrative resources.

One side effect of challenge and unpredictability is the creation of *immersion*. This refers to the flow state that players achieve on prolonged game sessions, where they fully engage with the game systems or narrative, and reality temporarily bends around the fiction they are experiencing. This only happens when the player is fully engaged with the game, so this is the direct outcome of a successful application of the guideline.

Finally, both variety and unpredictability work together to boost the player's **Curiosity**. This is a guidance technique that is deeply looked at in its respective guideline. Variety can influence unpredictability by creating the idea in the player that, since they have already run into a wide selection of events or scenarios, new experiences might be around the corner.

How to apply it

- *Challenge*. The first step consists of designing metrics to understand the challenge of our goals. The approach can be similar to the one taken to measure difficulty in puzzle games. There are different characteristics of our goals we can measure:

  - The number of valid solutions. A reduced amount of solutions forces the player to follow specific steps, which increases the challenge posed.
  - The number of mechanics involved. Depending on your game and the number of valid solutions this can both reduce or increase the difficulty of your challenge. More mechanics can introduce complexity or flexibility, depending on the number of valid solutions.
  - The duration of the goal. This can be measured in time, number of sub-goals, or number of steps. The longer a task is, the more challenging it becomes.

Once we have metrics to define the difficulty of our goals, we can sort them in ascending order. A specific metric used to define this progression is the *difficulty curve*. This curve can present different shapes and steepness across different games. Purely narrative games with little to no fail states usually have a flat curve with low steepness, while more demanding games like in the *Soulslike* genre usually have a more pronounced curve. The important consideration is that your curve matches the intended experience you want to provide. Difficulty curves also have a strong connection with **Pacing**, where altering the progression of challenge can positively impact the experience of our game. These curves do not necessarily need to be straight lines, they can have spikes. These spikes, both hills and valleys, serve pacing and produce moments of tension and relaxation in the player experience.

The organization of these goals in open spaces is trickier. With the inherent freedom of these spaces the player can decide where to wander at any point, so we cannot lay out our goals in a linear way. However, we can still trace paths by having *difficulty spikes*. Letting the player access areas with tougher enemies can serve

as a discouragement to approach such spaces before acquiring better gear or being more skilful. This also encourages the behaviour of *backtracking* and visiting old areas they did not face before, which can also help them find missed content in the areas they did explore. However, this difference in challenge has to be visually explicit. Players should know the space they are approaching is more dangerous just by the themes and looks of it. If that is not the case they might think that such is the intended path and then get frustrated due to them not being able to get through it. Getting this balance right is not a trivial task. This is one of the many reasons you should **playtest** your levels. Getting feedback from potential players is more valuable than any textbook, since their experience is specifically framed in the context of your game, unlike this kind of guidelines or textbooks, which aim for a more general understanding of level design.

- *Unpredictability.* One of the most common techniques to achieve unpredictability is the usage of *corners*. These are a layout resource utilized to hide game elements behind sharp turns in your paths. Although this is mainly a guidance tool that boosts **Curiosity**, this also serves the purpose of unpredictability, since players will not know what to expect. It is important to let the player know where paths or game elements might be hidden, since the purpose of this tool is to eventually let them explore every piece of content. In this sense, corners also serve as a **Pacing** resource, since they control the flow at which information is presented to the player so that it is more digestible.

Another more explicit and straightforward tool to integrate unpredictability in a game is through the usage of *randomness*. This can be applied to many different aspects of a game, determining, for example, the type of enemies to find in a dungeon, rewards of different value, and even the mechanics required to achieve a specific goal. In the context of level design, this is specially common in rogue-like games, where the layout of the dungeons, the loot, and the enemies found is different in every run. The implementation of this randomness in these more complex scenarios is done through *procedural generation algorithms*. Some example algorithms include *Random Walk*, *Wave Function Collapse*, and *L-systems*, but this is a deep topic I will not delve into in this document.

One more uncommon practice is the usage of **emergent gameplay**. This is achieved through the design of game systems and mechanics that have a high interoperability with each other, which is then further expanded by not explaining the player all these types of interactions, and expecting them to explore these on their own. The role of level design in emergent gameplay is providing the space where this exploration happens. There are two types of spaces for this purpose: *sandbox* and *playgrounds*. Both present similar definitions: they define a play space where the player can explore the different systems the game presents. However, there is a subtle difference, playgrounds have these systems embedded into their space, while sandbox spaces just provide the empty space. One example to follow is *Minecraft (2011)*, where even when world generation is an infinite flat plane, the player can use this space to play. On the contrary, a playground example would be *Sunset Overdrive (2014)*, where the space is populated with zip-lines and rails the player can grind to quickly move around while fighting hordes of enemies.

- *Variety*. One approach to create variety that is widely applicable and easily translatable to level design is through the concept of *modularity*. This concept refers to the creation of modules, or blocks of content, which then can be combined in many different ways to create different experiences. In game design this is achieved through the interaction of different mechanics and systems. Level design works in the same way, but instead they define these modular pieces as *action blocks*, where each piece represents a simple idea on how to interact or traverse a level. This is specially powerful combined with procedural generation, since encoding random permutations of these modules is a widely researched topic. However, the modules themselves are usually created by hand, and the implementation of the algorithm requires level design insights to follow specific rules such as **Choices** or **Pathing**.

One specific consideration is that the concept of variety can be applied to almost every aspect of a game. Level art (biomes, themes, types of obstacles...), pathing elements (branching, funnelling, vistas, backtracking...), game mechanics involved (resource gathering, combat, platforming...), and even challenge are some examples of elements with strong variance. Level designers act as the architects of games by deciding in what way to put all these pieces together. This should be done in such a way that it aligns with the intended game experience.

- *Game Feel*. In terms of level design, game feel can be explored from the sensations the different spaces create on the player. As explored in following guidelines, *architectural theory* [**Spatial Structures**] can explain how these spaces have an emotional influence with different shapes and scales. In the same sense that aesthetical elements achieve game feel in simple mechanics, level design can enhance the type of experience the game wants to provide. For example, specific movement mechanics like the ones we find in *Mirror's Edge (2009)* convey different sensations when applied in open spaces or in tight corridors. Sharp edges can convey a sensation of danger, while smooth shapes allow for a more organic environment where players might feel safer.

Counter Effects

- *Challenge*: If the difficulty does not match the intended or expected experience it feels unfair rather than engaging. Players will no longer read it as a fun challenge, but rather as *friction* that gets in the way of what they came for. This can be in a global sense, where the overall difficulty does not match the expected experience, but it can also happen in set moments in the game, where a spike in difficulty or a valley can discourage players from going forward, since they might feel the burden is not worth their time.

- *Unpredictability*: A game that is too predictable is going to lack **Curiosity**, which is a key concept that drives the player throughout the experience. On the other hand, systems that are too unpredictable or overly random make it hard for players to learn, anticipate, and strategise. When their attempts to understand the game never pay off, it feels incoherent and discouraging instead of mysterious or deep.

- *Variety*: On the same note as the previous point, overly repetitive encounters, mechanics, or rewards quickly drain excitement. Without a steady sense of novelty or progression, the objectives the game presents may start to feel like chores instead.

- *Game Feel*: Weak or absent game feel (no satisfying feedback, weight, impact, or responsiveness) turns otherwise solid mechanics into bland interactions. Even good ideas can feel lifeless when every action feels flat, floaty, or unresponsive.

All together, neglecting these elements does not simply make an experience slightly worse, but it directly contributes to frustration, boredom, emotional detachment, and ultimately players abandoning the game.

A.2.2 Real Industry Examples

- *Challenge*. An example of a game that has been called out for having a bland and dull difficulty curve is *Hogwarts Legacy (2023)* due to its challenges feeling too linear and scripted. It aims for a broad target audience by lowering the difficulty bar, making it a less exciting experience, specially in the moments the narrative demands it. On the contrary, *Celeste (2018)* has a perfect progression. It slowly introduces new mechanics and evolves them over time, combining them to increase complexity and challenge. It paces out the challenge by having narrative beats scattered around the levels, and difficulty spikes on specific rooms. It also uses open spaces and branching so that the player can decide what room to approach next, rewarding exploration and completion of more difficult rooms. Another example of a game with an imbalance in its difficulty was *Hollow Knight: Silksong (2025)*. At its release, this game presented an extreme challenge for newcomers, since most of the environmental hazards and the tougher enemies would deal double damage in the early game, which means players would perish in 3 hits while still learning the basic mechanics. This was solved in one of the first patches, acknowledging the existence of the issue. Talking about *soulslike* games is tricky, since they already have a high bar of difficulty on entrance, but players already expect this. With this high bar in mind, *Elden Ring (2022)* presents an open world where the player can decide where to go at any point. However, the difficulty of the areas is easy to read, and players can decide by themselves if they want to approach harder challenges earlier than expected. Such is the case of *Caelid*, a mid-game area that is pretty close by to the beginning of the game.

- *Unpredictability*. Pretty common examples of games that have a good implementation of unpredictability is *Rogue-likes*, such as *Noita (2019)* and *The Binding of Isaac (2011)*. These games have a good integration of randomness, in a way that they drastically affect gameplay, without taking away control from the players from achieving their goals. On the contrary, games like *Rain World (2017)* and *XCOM 2 (2016)* have randomness mechanics that can definitely determine if the player is going to succeed in their goal. Getting some creatures spawn in such a way that they corner you in a room, or missing a close-range shot to an enemy you are exposed yourself to can quickly end your game and the enjoyment.

- *Variety*. A very explicit implementation of variety is through visually distinct regions. This is present in most traditional open-world games, like *Subnautica (2018)*, *Elden Ring (2022)*, and *Zelda BOTW (2017)*. Other games that offer more realistic environments like in the *Grand Theft Auto* series present this variety in a different way. The structure of the city roads, the business area, the suburbs, or coastal areas differ both in looks and in gameplay experience. Modularity is a concept that is more easily attributed to rogue-likes, where specific rooms and enemies offer variation by the distribution and subtle differences in looks and behaviour. This is the case of *Enter the Gungeon (2016)* or *Dead Cells (2017)*.

- *Game Feel*. This characteristic of games is harder to define and diffuse to point out, but some examples of wonderfully implemented environmental storytelling are *Horizon Zero Dawn (2017)* and *The Last of Us (2013)*, where different displays of scenes, corpses, and NPC interactions portray narrative bits. In particular *The Last of Us* uses openness, light, colours, sound, and layout to convey specific sensations to the player, at the same time it drives narrative and tension.

A.2.3 Metrics and Validation

- **Challenge**. This is usually analysed with the difficulty curve, which evaluates the increase of challenge throughout the progression of the game. The skill set of the expected player base can be put in perspective with this graph, having five distinctive regions. When the skill surpasses the challenge the player can feel bored or even apathy if the difference is too extreme. In the opposite case, the game can feel too demanding and the player can feel frustration or even anxiety. When the challenge is just right, where it is demanding but not infuriating, the player can enter the flow state.

- **Unpredictability**. You can measure this aspect of a game by analysing the randomness integrated in your gameplay. You can also measure the distance and number of occurrences of corners, where the player is subtly guided to small hidden pieces of levels. A measure of validation is to make sure not all the required information is presented at one, let the player explore and find all the bits of game knowledge in small steps.

- **Variety**. In the case of working with modularity, you can mathematically calculate the number of permutations or combinations these modules present. Go through all your conceptual tools and assert you have some kind of variety where needed, these conceptual tools referring to the different **Pathing** and level design resources presented in this text.

- **Game Feel**. The best approach to ensure the game feel is right is through playtesting. Hand out the controller to your colleagues and ask them how do the controls feel in the space you created. What kind of sensations or thoughts come to their minds while experiencing such parts of the game. This process can also be carried out through public testing.

A.2.4 Related to

- **Engagement: Choices, Meaning, Pacing**. These are the three main pillars that define engagement. Each guideline explains in detail what each concept means, how to apply it, and real industry examples and metrics. Choices are related to challenge, unpredictability and variety. Meaning is related to game feel. Pacing is related to challenge and variety.
- **POIs**. Variety is an intrinsic characteristic of points of interest, and how their frequency should be inversely proportional to their relevance.
- **Curiosity**. Curiosity can be born from unpredictability. Players not knowing what to expect get the desire to figure out what the game is hiding from them.
- **Spatial Communication**. This dictates what the player is going to be able to read from their surroundings, which helps in the context of unpredictability. It helps keeping a balance by letting the player have enough information for them to wonder about the outcome of some events, so that hey are not completely unreadable.
- **Spatial Structure**. The spatial structure of your game determines in what order the player finds the different objectives, and that explicitly defines the challenge progression the player is going to face. The shape of the space also provides a specific game feel through *architectural theory*.

B

Obsidian Diagrams

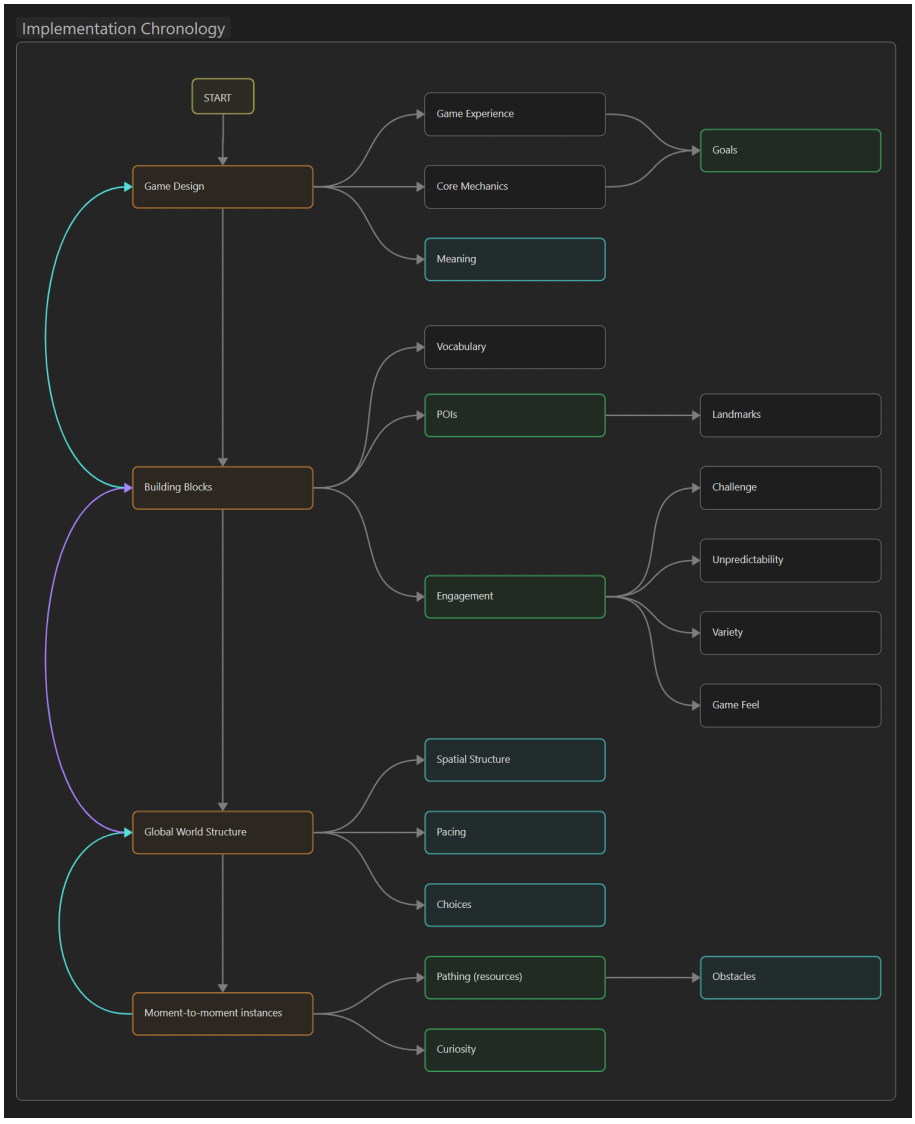

Figure B.1: Diagram that displays in what order all the guidelines can be applied for a specific iterative development pipeline.

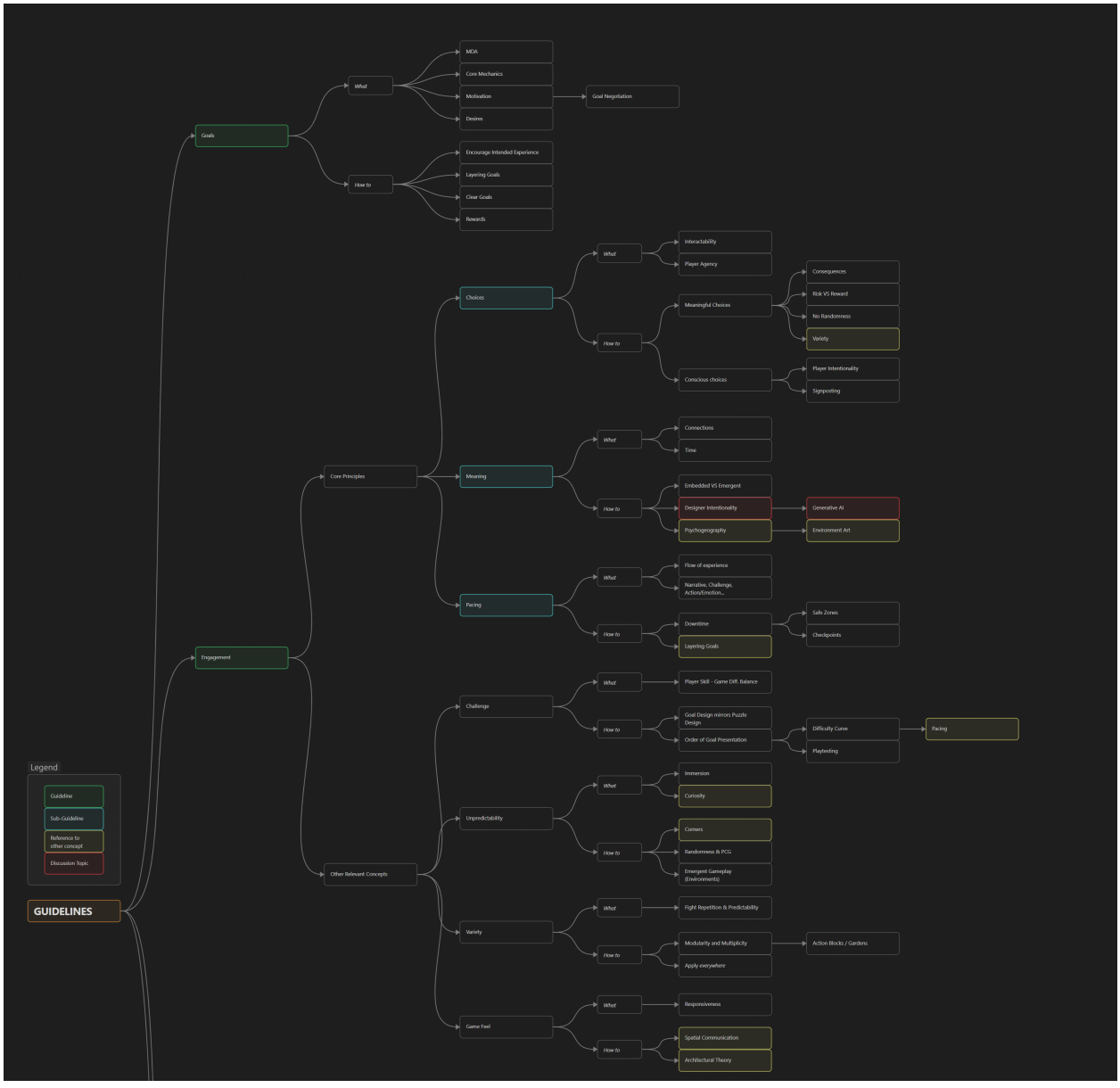

Figure B.2: Guidelines canvas [Part I] that links key concepts to Goals and Engagement.

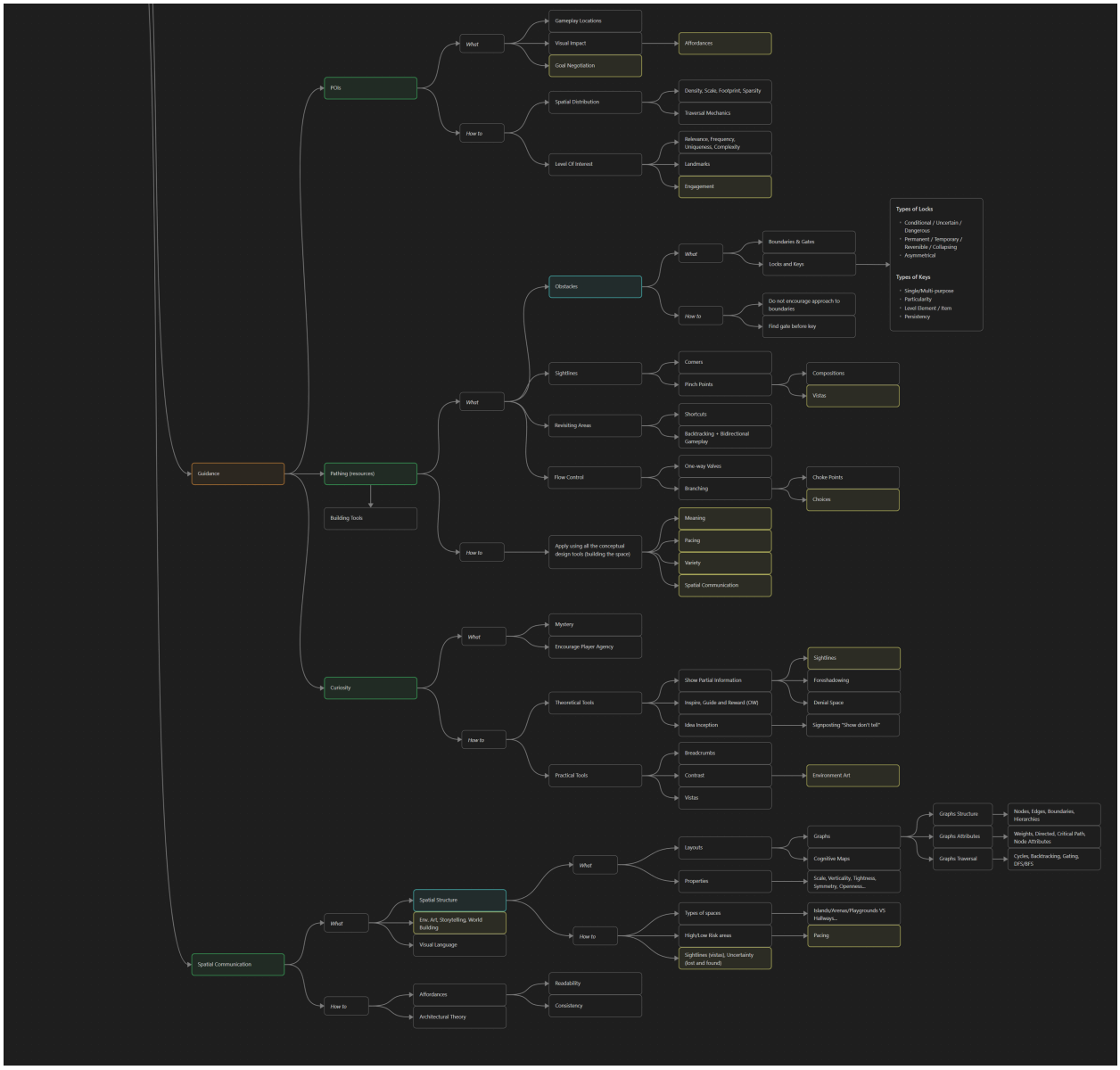

Figure B.3: Guidelines canvas [Part II] that links key concepts to POIs, Pathing, Curiosity, and Spatial Communication.

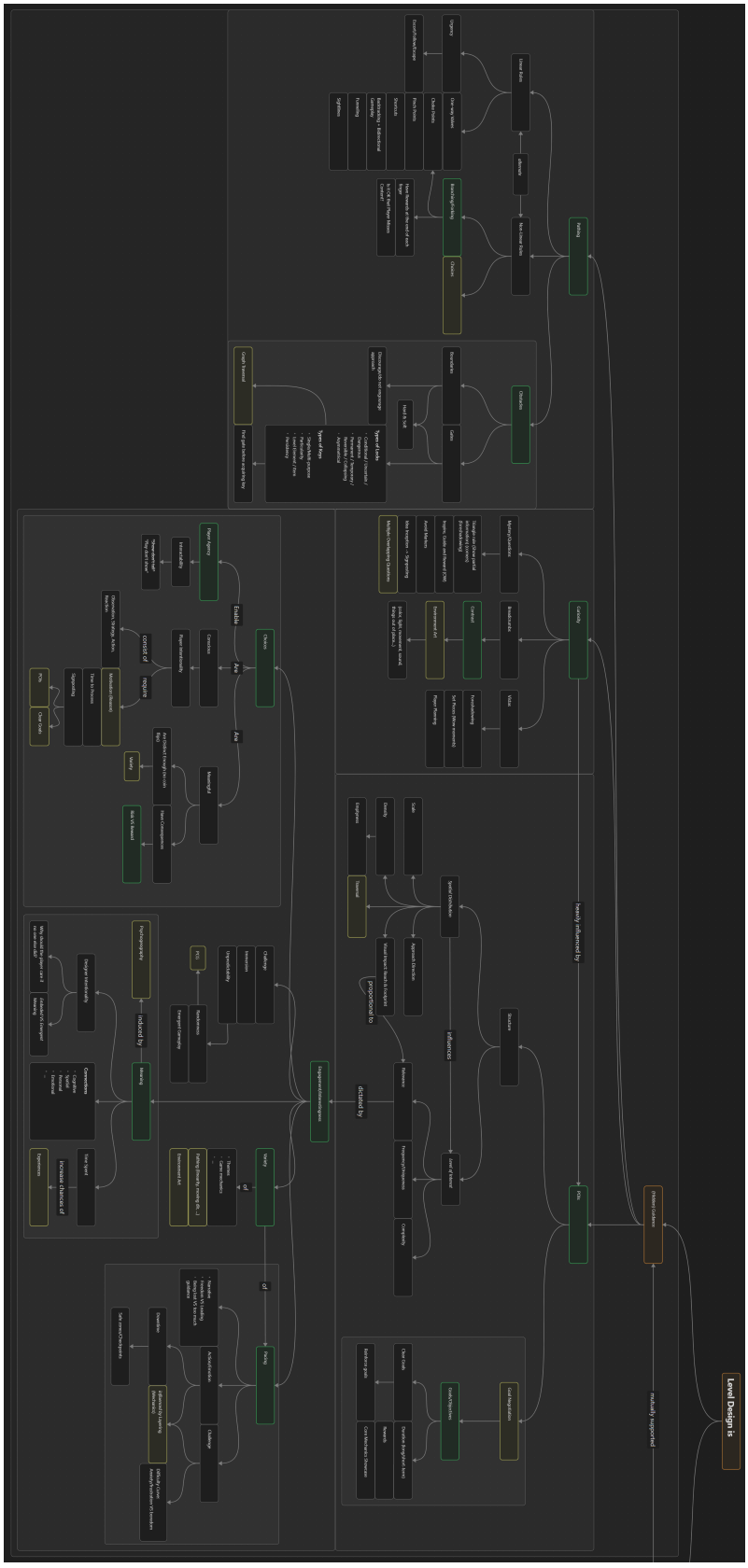

Figure B.4: Level Design canvas [Part I] that displays all the concepts related to **guidance**.

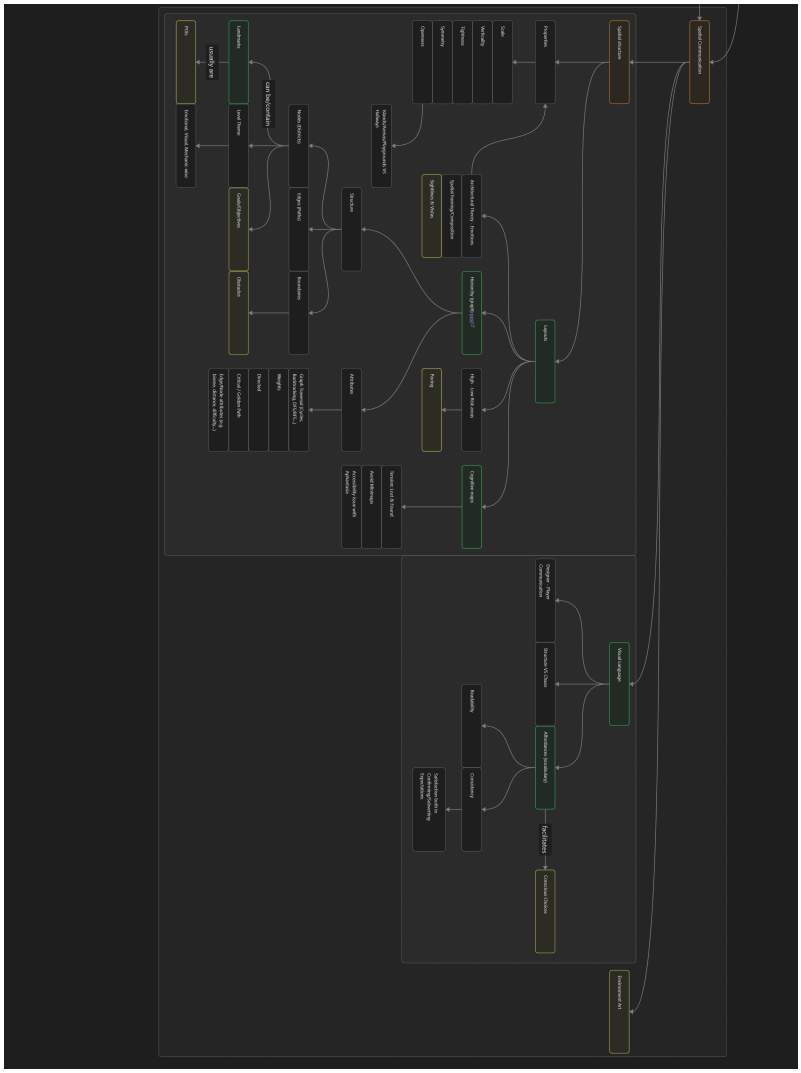

Figure B.5: Level Design canvas [Part II] that displays all the concepts related to **spatial communication**.

C

Example Vistas from the Second Artefact

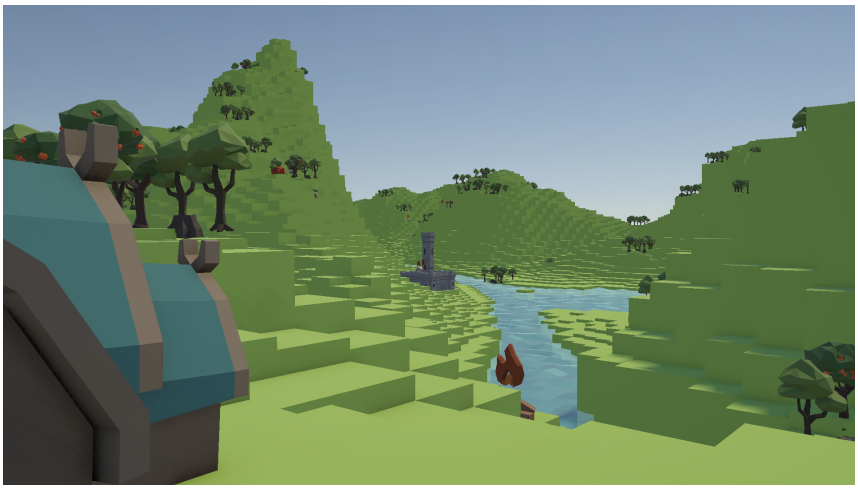

Figure C.1: Vista from the player spawn where the main goal (castle) is visible along the shoreline.

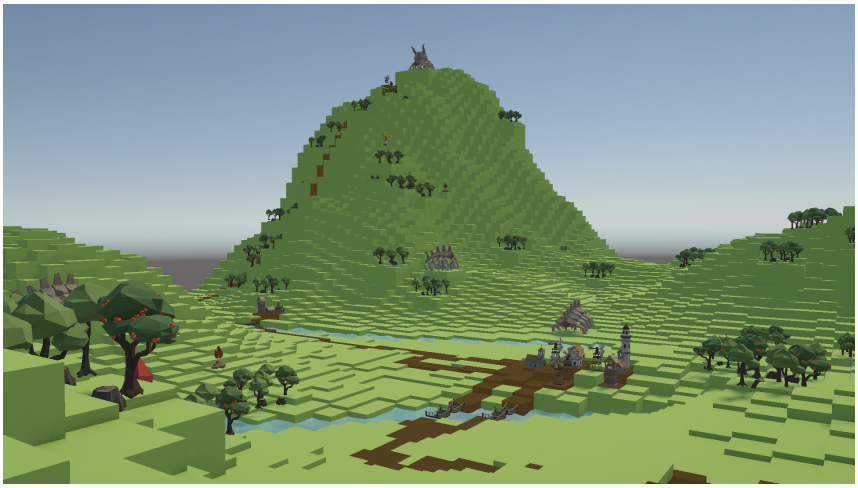

Figure C.2: Vistas from a path that leads to a village in a valley, with a foreboding enemy camp at the top of a mountain.

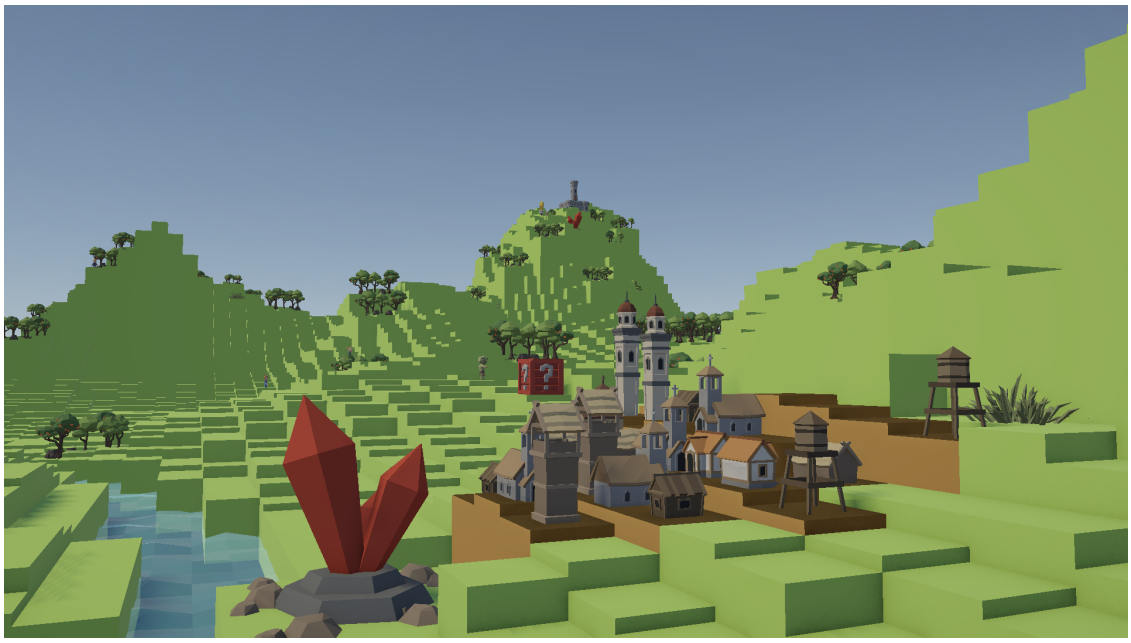

Figure C.3: Village from which the main goal (castle) is visible at the top of a mountain.

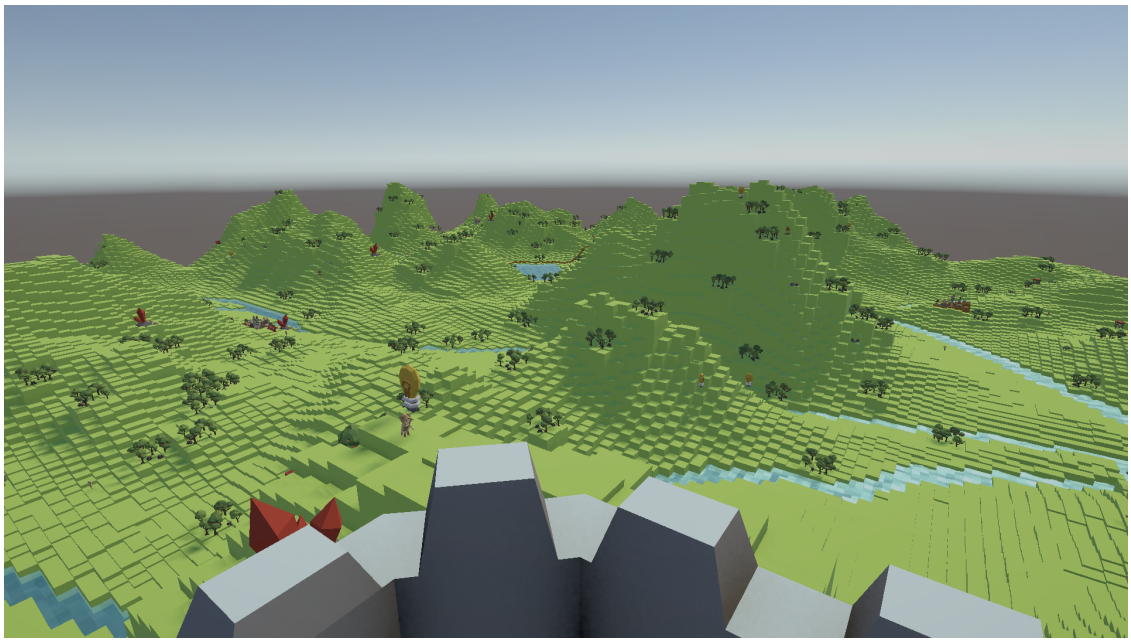

Figure C.4: Same generation as last figure, from the point of view of the castle. Here two towns can be observed in the distance.

D

Level Design Sources Outline

This chapter outlines the main sources used to extract all the knowledge about level design in the research phase of this work. This is necessary due to how many of the sources overlap in the topics they cover and how each individual source often addresses multiple concepts. Including these reference would not only be an arduous task, but it would also clutter all the text and result confusing to the reader, instead of providing useful information about the origin of the different concepts and statements. For these reasons this approach serve as an alternative. What follows is a list of sources about level design and its integration with PCG, where each entry summarizes the concepts it contributes to.

Publications

- **A Phenomenological Approach to the Design of Spatial Experience in Open-world Games**. [51] Research article that explores how players perceive and make sense of the world around them. This source explains the difference between a space and a place. It is used as a foundation to define the meaning guideline, and it also collects several concepts about hidden guidance and orientation.
- **Exploration in Open-World Video games: Environment, Items, Locations, Quests, and Combat in The Witcher 3**. [3] Research article that explores many concepts of open-world games pertinent to level design. These include unpredictability, immersion, curiosity, POIs, goals, and more. It defines exploration as the process of locating POIs.
- **Level Design Patterns in 2D Games**. [52] Paper that provides a list of many key concepts proper of 2D level design, which can be also applied to 3D spaces. These include guidance, breadcrumbs, safe zones, pacing, foreshadowing, curiosity, layering, branching, choices, risk VS reward, and more.

Book Chapters

- **The Art of Game Design: A Book of Lenses**. [10] This book is used as the main source to explain how level design can influence a game experience. It explains how to design with a focus on the intended player experience, and explores goals, motivation, meaningful choices, challenge, feedback, pacing, and more.

- **Level Up! The guide to great video game design: Levels**. [53] This source merges level design principles, with practices and game design considerations all in one. It explains types of spaces and their characteristics, how to approach level prototyping, it talks about different types of variety to keep the experience fresh, it mentions the importance of goals, and it also includes a section to talk about the integration of PCG.

- **An Architectural Approach to Level Design**. [7] This book explains most of the concepts explored in the spatial communication and spatial structure guidelines. It mentions layout hierarchies, wayfinding techniques (landmarks, sightlines, etc.), it talks about spatial readability and affordances, pacing through spaces, and the experiential impact different structures can have over the player.

- **Level Design Practices for Independent Games**. [54] This source is the main reference that explains how level design can be defined as the process that implements game design, and how it should be structured around the game's core mechanics. It focuses on practices for a development process, like the implementation of iteration, or the use of an object-oriented approach. However, it gets diluted in level design patterns and game design considerations, such as pacing, challenge balancing, guidance, sandbox spaces, and emergent gameplay.

- **The Illusion of Choice - How to hide linearity of levels**. [55] This source explicitly states the concept of 'hidden guidance'. It explains several guidance concepts like landmarks, branching, types of narrative deterrents, and more. It also explains the consequences of an unsuccessful implementation, like the lack of engagement or immersion of the player.

- **Level Design Planning for Open-World Games**. [22] This book chapter explains a useful practice of creating a living documentation. It explains how to define the world map and the list of important locations, along with all their metrics and properties. It also explains a possible workflow, from pitching the idea and greyboxing, to art and polish. However, it also explains how a toolkit for level design cannot be provided, since 'any game's needs are different from one another'. This quote was important to define the scope and purpose of the guidelines.

- **Procedural Generation in Game Design: Meaning**. [48] This source is used as a foundation of the meaning guideline. It explains how meaning is born from connections, and the different types of connections there are. Although this source sometimes confuses the concept of meaning with importance or relevance, it explores how this meaning can exist or not in procedurally generated games.

- **Procedural Generation in Game Design: Graphs and Cycles**. [56] This source explains how to understand game progression from a mathematical perspective using graph theory. It explores many concepts for graph traversal that implement tools presented in the pathing guideline, such as branching,

shortcuts, one-way valves, critical paths, and more. It also serves as the main source to define gates, through the theory of types of keys and locks mentioned in the obstacles guideline.

Articles and Web Sources

- **My Level Design Guidelines (M. Barclay)**. [21] Web article that presents several key concepts of level design from an industry professional. It explains concepts from several guidelines all at once, like themes, pacing, signposting, landmarks, denial spaces, POIs, breadcrumbs, spatial lures, boundaries, vistas, visual languages, set pieces, gates, objectives (goals), rewards, branching, player agency, sandbox levels, choke points, and one-way valves.
- **Level Design Fundamentals**. [57] Article published by Unreal Engine that explores a wide variety of concepts in level design. These include core mechanics, possibility space, shape language, space theory (negative space, occlusion, and player perception), metrics, scale, cognitive maps, graph theory, hierarchies, player motivation, rewards, and player choice.
- **The Level Design Book**. [58] Digital book which serves as a guide that explains how to create a game from the ground up. It presents a workflow pipeline, explaining pre-production, game design, layouts, scripting, environment art, and release, among others. It explains important theory about the use of lighting as a spatial lure along with framing theory. It explains layouts and greyboxing, and how it can be integrated in an industry pipeline. It also provides dozens of tools and resources to start practising game development.
- **'Define first, design levels later' (Translated from Spanish)**. [23] Industry professional shares his insights about game design documents and paper prototyping in this video. He explains different tools on how to properly use drawing tools to communicate ideas on paper. Other entries of his channel explore block outs, understanding levels as game experiences, types of spaces, and open-world guidance, among others.

Talks and Developer Insights

- **Spatial Communication in Level Design**. [4] Talk at Develop Digital (2020) that explores many concepts in pathing, curiosity, goal, and spatial communication. It provides a practical example on how the following concepts apply in a space: greyboxing, clear goals, sightlines, cognitive maps, reinforcing goals, affordances, bidirectional gameplay, contrast and spatial lures, denial spaces, one-way valves, vistas/vantage points, branching, mystery, shortcuts, backtracking, pinch points, vocabulary, affordances, locks and keys, and design constraints.
- **Ten Principles for Good Level Design**. [59] GDC talk that explores several principles of level design. These include: visual languages, space navigation, environmental storytelling, clear objectives, layering goals, expectations, consequences, difficulty, risk vs reward, modularity, bidirectional gameplay, architectural theory, and MDA.

- **An Approach to Holistic Level Design**. [60] GDC talk that explain how a game consists of a story, gameplay, and their presentation, and how designers control these through intentionality, affordances, world building, goals, desires inception, and more.
- **Designing Radically Non-Linear Single Player Levels**. [61] This GDC talk explains how to implement branching and choices, managing pacing and player tension. It explains different layouts of spaces with high and low risk, and the different types of level elements that are used to create these, such as attractors or deterrents. This same concept is used in one of the prototypes resulting from this thesis.
- **Stop Getting Lost: Make Cognitive Maps, Not Levels**. [62] This GDC talk explains in detail how players' cognitive maps work and how they follow graph theory, where nodes, edges, and hierarchies are the basics. It also explains landmarks, minimaps drawbacks, districts, and how to use all these elements to orient your players or to get them lost.
- **Playgrounds and Level Design**. [63] Another GDC talk that explains the nature of playgrounds and sandbox spaces, and matches them to graph theory, following nodes, edges, and hierarchies. It also explains the usage of modules, landmarks, goals, obstacles, and rewards.
- **The Making of The Legend of Zelda: Breath of the Wild**. [64] This is an official video entry from Nintendo explaining different considerations behind the creation of *Zelda BOTW (2017)*. This thesis uses it as a resource to explain how modularity can be used as a tool to implement with PCG.
- **Skyrim's Modular Level Design**. [6] This GDC talk explains how games like *Skyrim (2011)* use a modular approach to create content like its dungeons. In particular, this resource is utilised in this thesis to explain how level design has an interdisciplinary nature, and how level designers need to keep active communication with the rest of the team (level art, narrative, tech team, and game direction).
- **Gyms, Zoos, and Museums: Your documentation should be in-game**. [40] This talk explains different types of in-game documentation and their purpose. This is a useful practice for level design to keep all the building blocks organised. It can also be extended to organise action blocks.
- **Sparking Curiosity-Driven Exploration Through Narrative in 'Outer Wilds'**. [65] This GDC talk serves as the main source for the curiosity guideline. It explains how to inspire curiosity by allowing the players to pull information, instead of pushing it onto them. It then explains how to guide the player along these attractors maintain curiosity and push the flow forward. It also talks about layering goals and rewards.

- **Designing for Curiosity in Outer Wilds**. [66] This talk given in the Full Indie Summit provides a list of concepts used to make the player curious. It particularly focus on how *Outer Wilds (2019)* frames it, but its information can be extrapolated to other games. It mentions concepts like: layering goals, pacing, clear objectives, meaningful choices, meaning (narrative relatedness to the player), and signposting.