Fakultät für Informatik
Technische Universität München

Master's Thesis in Informatics: Game Engineering

Field Level Design in "The Legend of Zelda: Breath of the Wild": Deriving Principles, Tools and Methods for Open Worlds in Video Games

Lukas Liu

Fakultät für Informatik
Technische Universität München

Master's Thesis in Informatics: Game Engineering

Field Level Design in "The Legend of Zelda: Breath of the Wild": Deriving Principles, Tools and Methods for Open Worlds in Video Games

Feld-Level-Design in The Legend of Zelda: Breath of the Wild: Ableitung von Prinzipien, Werkzeugen und Methoden für offene Welten in Videospielen

Author: Lukas Liu
Supervisor: Prof. Dr. David Plecher
Advisors: Daniel Dyrda
Submission Date: 15.10.2024

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.10.2024

LUKAS LIU

Abstract

Open world video games face the challenge of striking the balance between guidance and player freedom, as to provide enough guidance for the player towards objectives while keeping the game non-linear to preserve player autonomy. In order to achieve this, subtle guidance through level design choices such as environmental shape or topology can be utilized for that purpose. However, current development lacks tools and guidelines to assess whether an open world game is suitable to achieve the balance between player guidance and autonomy to deliver the intended game experience. This paper examines Nintendo's *The Legend of Zelda: Breath of the Wild* design philosophies as a case study in order to infer tools and guidelines for general open world level design. It's design philosophies and level design strategies prove to be useful to purposefully lead the player's attention through the game world by subtle means without actively disrupting player freedom. Finally, this paper proposes a tool developed for Unity that assists level design decision-making by indicating visibility information of objects in the game scene during the design process incorporating a key aspect of Nintendo's design strategy into the general level design workflow for open world games.

Acknowledgements

First and foremost, I would like to thank my supervisor, Daniel Dyrda, for supporting my thesis and providing valuable insights. The organization of collective meetings with fellow students, facilitated by Daniel, enabled productive discussions and the exchange of information, while still allowing for individual meetings to seek advice. This approach proved extremely helpful for the progress of this thesis. The feedback and insights from both my fellow students and Daniel were crucial in shaping the direction of this work and overcoming challenges along the way. I am truly grateful for Daniel's patience and expertise, which have greatly contributed to the completion of this thesis.

I am also very thankful to my family, who have supported me throughout my entire studies and this thesis journey. Their encouragement, wisdom, and advice helped me persevere through the tougher times and provided the foundation I needed to complete this work. The sage advice on work ethic and approaches to tackling my thesis enabled me to carry out this project more efficiently, playing a vital role in its completion.

Lastly, I want to thank all my friends, who were always there to discuss ideas, engage in endless conversations about this project, and offer different perspectives on the topics covered in this thesis. The much-needed breaks and distractions in between were invaluable in clearing my mind and motivating me to push forward. I am truly thankful for all your support and encouragement.

Contents

1 Introduction

In an interview [20] with Hidetaka Miyazaki, the CEO of *FromSoftware*, on whether there were any particular mechanics or gameplay systems that were difficult to get right in *Elden Ring* [36], he responds as follows:

"[...] A lot of it was related to the game tempo - the rhythm and the flow of the game, to keep the player from getting bored, to keep them interested, exploring and having fun. And, of course, in this brand-new huge world that we've created, we wanted to prioritise that fun and level of player freedom more than anything. So with that comes a lot of characters, a lot of events that you're trying to incorporate, and you don't want anything to tread on the toes of anything else - you want it all to mingle and to mesh nicely with the player and their own motivations as well."
(Hidetaka Miyazaki, 2021)

On a similar topic, in a *Rock, Paper, Shotgun* article [26], the author outlines an interview with former *Fallout* and *The Elder Scrolls* level designer Nate Purkeypile and Matt Firor, *ZeniMax Online* president and *Elder Scrolls Online* [42] director. The author expresses the frustration with modern open world games and their tendency to induce "choice paralysis" and regards "open worlds as chores". Purkeypile argues that the problem stems from how open world games are designed, where the design is more akin to a "checklist" rather than having exploration in mind. Nowadays, the open worlds "might cast you as explorers, but they don't feel like places you discover".

Figure 1.1: Portion of the in-game map of *Assassin's Creed: Unity* [41] showing all points of interests

This is illustrated by Fig. 1.1 with the in-game map of *Assassin's Creed: Unity* [41]. While

the game is considered an open world game, most points of interest are marked on the map. It enables the player to easily find any point of interest but the sense of discovery and exploration is lost in the process. Additionally, the aforementioned "choice paralysis" could emerge in this scenario, as players are presented with numerous options at the same time.

In contrast, the author states that in *The Elder Scrolls III: Morrowind* [35] the world design is "an obscure and formless landscape where the geography must be actively navigated based on scraps of information". Firor declares that with that kind of environment and vague quest directions, today "no one would play it". In other words, Firor expresses that without any kind of guidance, the player would become lost and not find their way to any objective or point of interest.

As Miyazaki explains, crafting an open world is a complex endeavor, filling the world with characters and events that mesh together while preserving the player's freedom and ability to explore. Through the aforementioned *Rock, Paper, Shotgun* article [26], it is suggested that an open world must strike a balance between allowing the player explore the world at their own pace while steering them towards key locations and objectives. Otherwise, an open world can result in a tedious or disorienting experience. This thesis examines the principles of open world level design and proposes guidelines and tools to facilitate more effective level design decision-making for open worlds.

1.1 Problem Statement

Open world games have to find a compromise between line letting the player freely explore the environment without getting lost, but also have to lead them to key locations. The difficulty in designing an open world to achieve this could stem from a lack of clear guidelines and tools to better judge level design choices. Level designers might lack the tools to properly evaluate whether their level design choices are suitable for their open world game experience. Due to this, some open world games suffer from being either too linear or guided, which takes away the opportunity for the player to explore at their own pace, or being too open, resulting in the player getting lost.

1.2 Solution Approach

This paper investigates the paradigms and approaches used in Nintendo's *The Legend of Zelda: Breath of the Wild* (BotW) [39] and formulates principles and methods in an attempt to alleviate the design problem. The developers of *BotW* used a unique design approach worth investigating, which could be applied to open world design in general. As *BotW* is critically and commercially successful and currently the most sold title [9] as the first open world title in the franchise, it is important to analyse its features to improve open world design quality in the future.

Following that, potential implementations to apply the examined design philosophies and strategies into level design workflows are investigated to facilitate improved decision-making in level design choices based on the paradigms established in *BotW*. If *BotW*'s design strategies can be effectively adapted to general open world level design accordingly, it becomes vital to incorporate them into a level design workflow, streamlining and improving the overall design process.

In the following chapter, the approach to the aforementioned method is outlined and discussed.

2 Methodology

In this chapter, the procedure of analysing *BotW* is described and how paradigms and tools are extracted for the purpose of general open world level design.

The research conducted in this paper qualifies as a qualitative case study on *BotW* and can be broken down into three distinct phases: analysing the design philosophies discussed in a *Computer Entertainment Developers Conference* (CEDEC) talk by Nintendo in 2017, investigating how these principles are applied within the game itself, and developing tools to apply these principles to open world game design.

The first phase comprises a detailed analysis of the Nintendo presentation, with a focus on identifying key design philosophies related to open world level design. These design principles are then examined in the second phase, where specific in-game examples from *BotW* are investigated with the design philosophies in mind. By observing how the game's world design supports non-linear exploration, this phase provides concrete examples of how the presented theories are implemented in practice. In the final phase, the findings are used to develop tools that can help implement similar world design principles in open world games more broadly. This process includes designing a set of systems and potential workflows that can assist game designers integrate these principles into their own projects.

By structuring the methodology in this way, the research ensures a comprehensive analysis of both theoretical design concepts and their practical applications, while also contributing practical tools that can be used in future game design. Each phase is described in detail below.

2.1 Analysing Nintendo's Design Philosophies

The first phase of this research involves a detailed analysis of the 2017 CEDEC presentation by Nintendo, where the developers discussed the world design of *BotW*. This presentation offers valuable insights into the key design philosophies that shaped the game's open world structure and player interaction with the environment. The objective of this phase is to systematically extract these design principles and organize them for further analysis and application.

To conduct a thorough analysis, Matt Walker's translation [32] of the 2017 CEDEC presentation by Nintendo serves as the primary data source for this phase. The reason for this is that there are no publicly available recordings of the talk itself, as it was held in private. However, the translation by Walker are based off Japanese news articles covering the CEDEC talks and serve as a summary of these news articles. Therefore, due to the private nature of the CEDEC presentation, this phase relies on Walker's account of the talk based off Japanese news outlets which are publicly available. The original news articles are credited on Walker's translation accordingly.

As this paper aims to elaborate on open world level design, the focus of this phase stays

on the Field Level Design section of the talks. While the talks encompass user interface (UI) design, sound design and project management, they lie out of scope for the purpose of this paper and are thus omitted. Key parts and explanations of *BotW*'s level design remains the focal point of this phase, and all references to design principles and strategies will be included.

Once key points of *BotW*'s level design are established, the next phase concentrates on the concrete implementations and manifestations in the game world itself.

2.2 Application in *The Legend of Zelda: Breath of the Wild*

In this phase, the focus lies on how the design principles extracted from the 2017 CEDEC presentation are implemented in *BotW*. The goal of this phase is to observe specific in-game elements that align with the core philosophies identified in phase 1, providing concrete examples of how these design principles are implemented in practice. This phase uses a combination of in-game observation and systematic analysis of the world design.

To explore the practical application of the design principles, a detailed observation of *BotW*'s game world will be conducted. This process involves systematically exploring the game environment, interacting with different mechanics, and observing player freedom. The following areas will be the primary focus:

- **World Map**: Investigating how the game's open world is structured to encourage non-linear exploration.
- **Level Design**: Examining how the game's environmental structure allows the player to choose their own path without the need for explicit guidance. Important emphasis here lies in the organic guidance through environmental cues such as landmarks or visual cues.
- **Terrain Navigation**: Observing how the player's freedom of movement affects the exploration and the environment, respectively.

2.3 Developing Tools for Open World Games

Lastly, this phase involved creating tools to facilitate the implementation of the previously extracted design principles from *BotW* in other open world games. The goal is to convert the identified design philosophies into usable tools and methodologies that can be used by game designers to improve design decisions to achieve an enjoyable open world experience.

For this phase, Unity is chosen as the targeted platform due to its widespread popularity in game development, its capabilities and the researcher's proficiency with the engine. Unity offers an extensive tool set such as terrain generation, meshes and scripting to directly integrate the identified design principles into its workflow. It is chosen over other game engines such as Unreal Engine or Godot Engine due to the researcher's expertise with the Unity Engine, decreasing learning curves and accelerating tool development and decision-making.

First, a conceptual framework based on the design principles from phase 1 and 2 needs to be developed. Design principles that can be directly translated into usable tools and prin-

ciples that are better suited as design guidelines and philosophies need to be identified and separated. Afterwards, designing and developing tool prototypes is necessary to facilitate the implementation of suited design principles. This paper proposes a visibility scoring tool as the most suited to establish the identified design principles and enhance the level design decision-making. The outline of the tool's processes and results finish the extent of this paper.

2.4 Limitations

While this study provides valuable insights into Nintendo's world design philosophy as presented in Breath of the Wild, there are several limitations to consider.

Scope of Analysis

This research is limited to *BotW* and limits the applicability to other open world games. The unique open world level design elements in *BotW* might not succeed in the same way as other open world games in its genre, as the level design is only part of the open world experience. Plus, the design principles are tailored to *BotW* and its vision, which furthers questions the applicability to other open world games with different mechanics in question. Additionally, other industry insights or presentations might offer additional perspectives that are not considered in this research.

Data Collection

The data source for Nintendo's design philosophies involves subjective judgments, as it is not the original presentation held at the CEDEC 2017. The accuracy of the recount of the talks is crucial, and omissions or errors in the translation or summary could impact the results of the analysis and skew the findings. Different interpretations of the design philosophies could lead to variability in results as well.

Validity

As this paper only proposes tools to improve the open world level design workflow, it is not validated or judged on its efficiency or usability in practice. The tools are developed based on theoretical design principles and are not empirically tested, which leaves the question on their effectiveness unanswered. As a result, while the proposed tools are designed to align with established design philosophies and findings of this research, their real-world applicability and performance remains open.

It is important to acknowledge these limitations to ensure objectivity and appropriately put the value of the findings accordingly into context for future open world game design.

3 Related Work

To fully understand the value of this paper's objective, it is crucial to place it amongst similar research and papers. Existing research on *BotW* or open world level design touch on similar topics as this paper and therefore, the following review will demonstrate the nuances of this paper.

3.1 Analysing *The Legend of Zelda: Breath of the Wild*

Several papers examined *BotW* as a game and its mechanics to different extents.

Vidqvist [31] analyses *BotW* as a whole and attempts to determine whether the game was successful as an entry to the open-world genre and which shortcomings or successes contributed to its achievements. While Vidqvist touches on the map design of *BotW*, they stay on a surface level as to what is seen and provided to the player and does not elaborate on how the map design affects the player's game experience. However, they elaborate on topics such as an open world definition, level design in general and *BotW's* progression system, which will be important later in this thesis.

Schnaars [29] introduces the concept of "airness" in the context of *BotW* which describes a design decision to fade the narrative of a game into the background to motivate an explorative approach during gameplay rather than putting the focus on narration or characters. The concept is then broken down into three central facets which support the concept inside *BotW*: quest design and its influence on map topology, tools of navigation and wayfinding and lastly the interactivity of the player with the world. Schnaars focuses here on the decision to fade the narrative aspect of the game into the background to incentivize an explorative approach to the game, which allows the mentioned three aspects to thrive in the game experience. Schnaars also mentions the "triangle rule" used in the map design to further support the concept of "airness". The "triangle rule" will serve as a focal point in this paper.

While Schnaars provides vital information to understand the intricacies of *BotW*'s open world design and how it supports an explorative gameplay approach, it is not extracted and applied to general open world design. The focus lies on how the concept of "airness" works specifically in the context of *BotW* and not how any of the three described facets could also work in general game design. This paper will focus on some aspects that Schnaars mentions as well and shift the focus on how to extrapolate and apply them for general use cases.

3.2 Procedural Content Generation

Procedural content generation (PCG) is an important aspect to consider when discussing level design as it is a prominent approach to quickly create vast and replayable envi-

ronments, especially for open worlds. PCG techniques are widely used in open world games such as *Minecraft* [38] and *No Man's Sky* [37] where the focus is on algorithmically generating content that can vary significantly between playthroughs.

Dunn [11] proposes a system to create large exterior environments with focus on terrain and vegetation for open worlds. Gao et al. [16] showcases different PCG approaches and ideas for different genres, including open world games. Shaker et al. [30] also outlines level generation methods through different approaches and means. PCG is a viable approach for level generation as it saves resources and enables replayability depending on the design goal of a game and its genre. However, using PCG trades off designer control to tailor specific parts of a world to be more optimal for a desired game experience when used in a fully-automated manner. Additionally, for the purpose of this paper, PCG does not render this paper obsolete as findings of this paper can be integrated into PCG algorithms and possibly improve the quality of level generation in the context of open worlds. Furthermore, handcrafted world designs or environments generated through PCG both retain their uses cases to this day, as it depends on the desired game experience a developer wants to achieve.

3.3 Open World Design

Many papers deliberate on features open world games need to exhibit to quality as an open world game and different approaches to manifest these features. Rotzetter [27] classifies different guidance systems in open-world games that subtly lead the player towards certain goals. While *BotW* features all the guidance systems in one shape or form, this paper focuses mainly on spatial guidance system's and how it manifests in *BotW* as a concept. Rotzetter provides a classification of different guidance system but to make them applicable, instances of each guidance system need to be formulated and modelled to be applied to general level design.

Fox [15] describes a hex tile map design system pattern for open worlds to enable modulation and reusability. While it is a different viable approach in creating an open world, its focus lies on efficiency to save development time and resources rather than improving the quality of the open world design. However, similar to PCG algorithms, the outcomes of this paper can be integrated into Fox' hex tile map design system to improve its result while retaining the efficiency gains from the system.

4 Background

To comprehend the context of this thesis, it is essential to first explore the foundational concepts and that form the foundation of the topic. This section provides an overview of the key principles and relevant theories that are vital to this thesis, offering a comprehensive background that provides necessary knowledge to understand the subsequent analysis and discussion.

4.1 Level Design

Level Design is a key aspect of any video game, as it is the "plate upon which we serve the gourmet dinner of our level" and serves one purpose: "to make the game more fun" [13]. Level Design describes the discipline to craft a virtual environment through which the player navigates and encounters gameplay mechanics that allow them to engage with and enjoy the game experience. Different games aim to create distinct game experiences and as such, “level design is different for every game, because every game is different" [28]. Many techniques and ideas exist nowadays to approach level design such as paradigms, design patterns, automated level design [30] or PCG [11, 16].

Bleszinski [6] describes different design guidelines to make a game more enjoyable and engaging. A game needs to control the player’s freedom to some extent as to letting the player think they are in control of where to go but is subtly guided towards their destination. Allowing the player unrestricted freedom to explore anywhere is likely to result in frustration and disorientation. By "giving the illusion that there are multiple paths", the player retains the freedom to choose and results in a more enjoyable game experience. Risk incentives are another example of viable game design. Giving the player the opportunity to weigh rewards against danger or risk lets the player choose whether to take a risk for a reward or play safe. The designer provides the player a choice, therefore retaining player’s feeling of control. Bleszinski also mentions the concept of "revisiting" or "doubling back". It describes the event when the player sees an inaccessible area of a level and wonders how to get there or past it. After progressing the game and unlocking the inaccessible area or the necessary tools to reach the area, players might revisit the previously inaccessible area, enhancing their sense of achievement.

While Bleszinksi describes game design principles, they ultimately manifest in a game’s level design to achieve the desired experiences.

In a similar vein, Khalifa et al. [19] examines levels from existing games and highlights common level design patterns and mentions a pattern they call "Branching". It characterizes the provision of multiple pathways for players to reach an objective while allowing them to retain control over their actions, akin to Bleszinski’s notion of constraining player freedom. In a pattern referred to as "Guidance" by Khalifa et al., the authors highlight the necessity of providing directional assistance to prevent players from losing sight of the intended path they should follow. Furthermore, they emphasize the need for guidance when "when exploration is an aspect of the design". The easiest way to provide guidance

to the player is "through the level shape" and the placement of enemies, collectibles or environmental cues.

Bartle [5] supports this claim as "getting lost is the main issue that virtual geography must address". In order to not get lost, "contextual and landmark commands are used, so that players can get back to somewhere they know relatively easily". In the case a player cannot see a point of reference, "graphical virtual worlds should provide players pointers to show directions" such as roads, rivers or signposts.

To summarize, level design plays a crucial role in making a game experience more enjoyable through shaping the virtual environment as to provide guidance, visuals, challenges and choices to the player. One of the main roles of successful level design is ensuring the player does not get disorientated through environmental cues, reference points or level shape. It is necessary to evaluate which design principles and patterns to use depending on the desired game experience, as it will differ from game to game.

4.2 Open World Design

4.2.1 Definition

Before analysing *BotW's* open world design, it is crucial to first define the term "open world". Despite its frequent usage by developers, critics, and players, what defines an open world game more often than not depends on varying interpretations and contextual factors, including the specific game in question, the perspective of a player, and the intended design goals of the creators. This lack of consensus complicates discussions about the defining characteristics and the criteria of an open world game, which hinders a proper evaluation of such. The following definitions are examined in an attempt to form an encompassing definition on which this paper will be based on:

"An open world is a nonlinear virtual world in which the player has the agency to roam freely and tackle objectives in the order they choose. [...] Obstacles can typically be overcome in multiple ways in order to facilitate player freedom and to achieve a heightened sense of player agency."
(Vidqvist, [31])

"In an open world game, players enjoy a great deal of autonomy in selecting from a complex system of mechanics, explorable spaces, and goals. Rather than experiencing a prescribed linear (or partially ordered) progression of challenges and plot points orchestrated by a game designer, players are free to experiment with the consequences of the game's mechanics and exhibit more creativity in deciding what to do, frequently devising their own goals. [...] Open world games emphasize emergence over progression: they are more interested in opening a wide space of explorable consequences of the game's mechanics than in delivering a specific sequential experience."
(Alexander & Martens, [3])

Fox [15] describes an open world by defining key features it has to exhibit. An open world needs to be an expansive game world that needs to deliver not only in scale but also in content diversity and density. Failing in this can result in the game feeling lacklustre. Fox also puts an emphasis on interactivity, as players expect to encounter meaningful engagements. Additionally, players seek areas of discovery manifested through unique and rewarding locations as it adds to the depth of the game. Lastly, the game needs to

display diverse visuals through various environments, which will otherwise lead to a sense of repetitiveness.

Common notions become apparent in the above definitions as to what comprises an open world game:

- **Non-Linearity**: The player is not constrained to a prescribed path through, for example, a narrative or the environment
- **Player agency**: The player can choose to engage with content and explore the game world at their own pace
- **Content diversity**: Diverse content inside the game world incites exploration and provides the player with different content opportunities

These seem to build a solid foundation for what constitutes an open world game. However, it is important to mention the notion of emergence. Emergence describes unexpected gameplay behaviour that stem from interactions between existing gameplay mechanics or rules of a game. This usually occurs organically when the player experiments with the game's systems. Emergence contributes to the player-driven non-linear gameplay of open world games and supports diversifying the content.

However, it can be argued that while emergence can enhance an open world game, it is not a necessary component. Games like *The Witness* [40] and *Assassin's Creed Odyssey* [34] both are considered open world games and have little to no emphasis on emergence through their gameplay. *The Witness*'s gameplay revolves around solving puzzles and exploring an island, while *Assassin's Creed Odyssey*'s gameplay focuses much on structured quests and scripted events. Both are expansive game worlds with diverse content, and a lot of the agency is with the player to engage and disengage with content as they wish.

For that reason and the sake of this paper, emergence is not included in the definition of an open world game.

With that said, Hughes & Cairns [18] offer a more nuanced definition in the form of Fig. 4.1 which encompasses the aforementioned concepts as well as elaborates on them. They break down an open world into five key themes:

1. **Players are situated to scale within the world**: The player is physically situated in the virtual environment and exists in it through an avatar or entity to scale within the world.
2. **The world is large, connected and accessible**: The world space is large enough for free exploration, fully connected in some way and areas are not barred by artificial barriers that are out of place in the context of the world.
3. **The main goal does not restrict players from engaging with other activities**: The main objective of the game should not be all-encompassing of the gameplay and should be integrated within the gameplay.
4. **Content density is more important than world size**: An open world needs to contain enough content in between goals and tasks, or risk feeling needlessly large. While this impacts the open world game experience, it is not mandatory to provide an open world experience.
5. **Players can self-pace gameplay through engaging/disengaging with tasks at will**: The player can engage and disengage with any content they are currently pursuing at will and unrestricted.

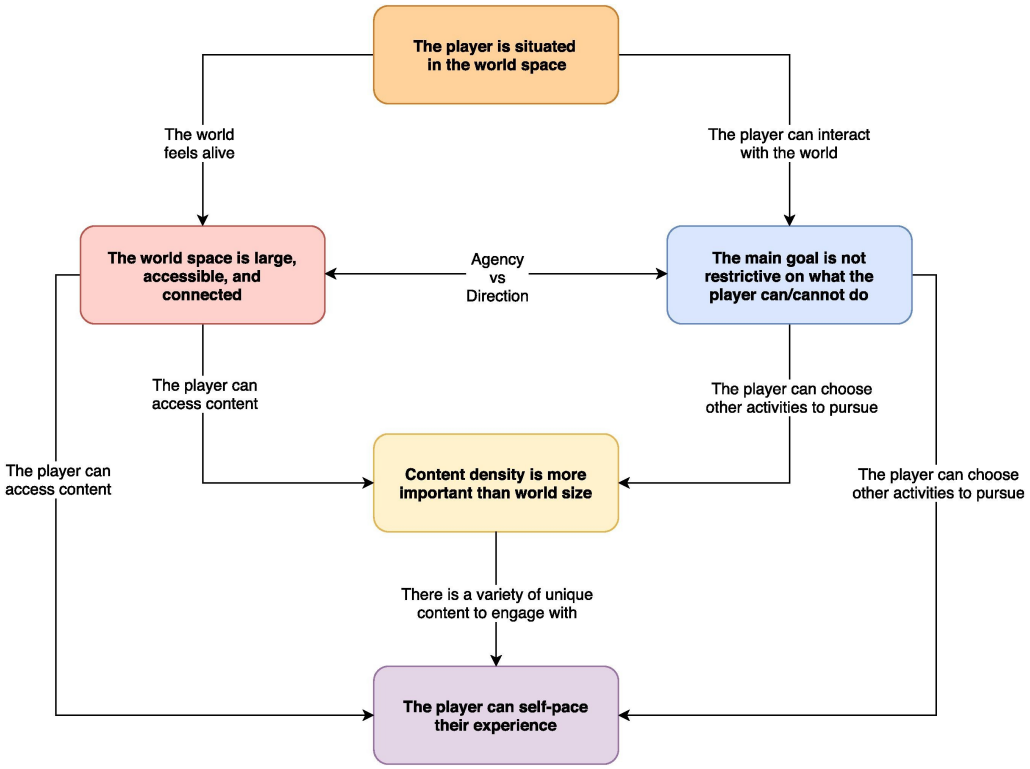

Figure 4.1: Hughes & Cairns [18] open world definition and how the different themes influence each other

As shown in Fig. 4.1, each theme is correlated and dependent on each other, where a lack of any themes would impact the open world experience negatively. The 1st theme relates to the player's feeling of being part of the world, rather than playing a spectator of a sort. This enables the next 2 themes: Since the player feels as though they are a part of the world through a sense of immersion, then the concept of creating an expansive world becomes available and shapes the 2nd theme. Similarly, since the player is within the world, they can physically interact with the world, enabling the 3rd theme. Through the expansive world and the free agency the player has, the player can access and interact with other types of content in the world, which creates theme 4. With the right amount of content density and diversity, the player is then able to freely self-pace their game experience and enjoy the open world experience resulting in the final theme.

Important to note is that theme 4 does not need to be fulfilled to create an open world experience. The ability to self-pace and choosing which tasks to pursue at any given moment without the sense of restriction can only occur with the first three themes in place. However, putting theme four in place adds to the open world experience with content diversity and a proper amount of content density in the world, enhancing the experience as a result.

In addition, the definition by Hughes & Cairns highlights the tensions between the themes as well and how there has to be a balance between the themes and in the themes themselves. Theme 2 describes the necessity of the world to be big enough to allow exploration, but at the same time it is detrimental if the world is too large to become repetitive or sparse. In theme 2, the requirement of having a non-restrictive main goal stands in conflict with the need for a meaningful and engaging main goal. The main goal has to

be present enough to be appealing but cannot be intrusive as to cause the player to feel restricted in their freedom. This tension is also presented between theme 2 and 3 in the form of the bilateral arrow in the diagram.

For this paper, this definition for an open world is utilized to evaluate variables through which open world level design can be improved upon. This paper will specifically address the aforementioned tension between player agency and guidance, as already previously outlined. In the following, existing paradigms for open world design are explored and examined.

4.2.2 Methods and Paradigms

Existing research already outlines some techniques and strategies to better understand the challenges of open world design.

Bonner [7] highlights the shift of computer games in the last decade to be more and more "vast open world game spaces" in new instalments of established franchises. Within this development, the atmosphere of city- and landscapes are conveyed to the player through "non-linear appropriation by the players" themselves and less via "strict narration". It seems that in open worlds games, narrative aspects are communicated progressively more through subtle cues on which the player picks up on by their own agency. This supports the statement by [1] which expresses that games that "may seem most story-like are the most spatially constrained and place-quest oriented". "The challenge for game designers who want to create rich, open game worlds and tell interesting stories at the same time" is to strike the balance outlined by Hughes & Cairns open world definition in Fig. 4.1. It is apparent that the narrative of an open world game has to shift towards the background and revealed to the player by other means, for example through player appropriation.

In a similar vein, Rotzetter [27] showcases different non-verbal guidance systems to subtly guide a player towards a goal. It is essential for guidance clues to strike a balance of understandability and subtlety. Information that is too difficult or complicated to understand is a failure of the guidance system to lead the player and clues that are too obvious results in "the essential aspect of open world games, namely exploring, vanish[ing]". Rotzetter classifies 6 different guidance systems in open world games:

- **Informative**: Pure information provided to the player directly; Examples: map, compass, location markers
- **Interactive**: Information obtained by the player's curiosity through interaction or properties of the environment; Examples: following a non-player character (NPC), cold/hot area for determent, roads and signposts
- **Processual**: Guidance through autonomous movements in the environment; Examples: river currents, wind currents, direction of a shot arrow through effects in the wind
- **Spatial**: Guidance through altering their freedom of movement, making the player execute a specific action or reference points for orientation; Examples: points of interest, (locked) doors, corridors, mountains, rivers
- **Emotional**: Evoke feelings to influence the player's movement; Examples: battle music to initiate combat, heartbeat sounds to implicate intensity

- **Narrative**: Information provided through narrative background, player using deduction to come to conclusions; Examples: smoke indicating a fire somewhere, an NPC being near a house concluding that the NPC might live in that house

It is crucial to understand which guidance system might work for what scenario, as "there is no guarantee that a guidance system works because it always depends on interpretable game aspects". However, "a combination of the six guidance systems and the deliberate use of their different capabilities reduces the risk of failure and may improve immersion and atmosphere considerably".

This might explain why *Assassin's Creed: Unity* [41] seemingly fails in part in its open world experience outlined in the previously *Rock, Paper, Shotgun* article [26]. As shown in Fig. 1.1, the game heavily relies on the map as its informative guidance system, which takes away the explorative aspect of open world games as most points of interests are provided to the player.

Lastly, Bonner [8] introduces the concept of two different types of space in which navigation is differently emphasized. They differentiate between *striated space* and *smooth space*. Striated space describes space "defined by lines and paths between two waypoints". This type of space puts reaching a destination as its main objective and is characterized by "following navigational points, overall goals of a quest or the NPCs' path routines". Smooth space on the other hand concentrates "the stretch of way, the distance itself". The notion of "the journey is the rewards" aligns entirely with the immersive nature of player interactivity in open worlds where there is an emphasis on exploration. Therefore, players who disregard informative guidance clues engage with the game world as a smooth space, traversing it freely without constraints. This practice is also known as "free roaming" depicting a form "of wildness in open world games". To quote:

"While an ideal example for smooth space is how nomads experience and navigate landscapes, the ideal example for striated space is how urbanites inscribe movement patterns into infrastructures. As constant as smooth space is transferred into striated space, striated space is reverted into smooth space. [...] This fits perfectly with the concept of smooth and striated space as reversible figure: whether the players follow an inscribed network of streets, paths and trails towards their next quest or destination (striated) or are guided by hyperreal impressions of aesthetics, luminosity, sound and topography (smooth). The latter might lead to detours but also to emergent events, rewarding prospects or rare items because free roaming (smooth space) is also intended and inscribed by game designers." (Bonner, [8])

The concept of striated and smooth space as a reversible figure also perfectly fits within *BotW*'s game loop. The player is constantly shifting between striated and smooth space while passing through the game's environment. As the player gradually explores and reveals the game's map, the revealed areas become striated spaces with waypoints and markers indicating important points of interest, allowing for systematic quick travel routes to be formed by the player. While discovering new areas however, the land is traversed as smooth space through its non-verbal guidance systems placing emphasis on discovery and free roaming giving opportunity for emergent gameplay and transitioning the smooth space slowly into striated space.

Understanding the distinctions between striated and smooth spaces, along with their respective characteristics and objectives, is essential for determining the appropriate navigational cues to utilize in order to facilitate the type of travel that players experience in each space. Effectively navigating striated space requires the use of explicit informational

systems to clearly direct the player toward their objectives. In contrast, smooth space is enhanced by more implicit cues derived from the environment, which serve to encourage exploration.

With that in mind, the next chapter elaborates and outlines the concept of wayfinding to understand the process of how a player might navigate a given game environment. As wayfinding and exploration plays an important role in open world games, it will be essential for the purpose of this paper.

4.3 Wayfinding

4.3.1 Definition

Farr et al. [12] explains wayfinding as "the process of finding your way to a destination in a familiar or unfamiliar setting using cues given by the environment". The effectiveness is mostly dependent on the interaction between human and environmental factors. The process of wayfinding is broken down into 4 tasks:

1. **Orientation**: Finding an individual's location with regard to nearby landmarks and the desired destination
2. **Route Selection**: Choosing a route that will lead to the destination
3. **Route Control**: Constant control and confirmation of the individual that they are still on selected route
4. **Recognition of destination**: Individual to realize and recognize that they have reached the desired destination

Arthur and Passini [4] further break down wayfinding as a "spatial problem-solving task" into 3 cognitive steps: information processing, decision-making and decision execution. These can be grouped together with the wayfinding tasks defined by Farr et al. Orientation mainly involves information processing as to take in environmental characteristics, pin down current location and evaluate possible routes. Route selection is a decision-making task assessing the desired route to take depending on the individual's goals. Route control requires decision execution by physically taking the chosen route and moving along it. It also comprises a repetition of orientation and route selection in order to evaluate whether the individual is still on route or has reached its destination. With that said, recognition of destination covers the last information processing task to be performed, which concludes the wayfinding task.

Depending on the specific scenario, it can be further divided into different types of wayfinding tasks, depending on the aim and circumstances of the task. Fewings [14] identifies 3 different types of wayfinding tasks depending on the aim of the task:

- **Recreational**: Wayfinding process a source of satisfaction and enjoyment through the opportunity to solve a puzzle or problem; Example: maze, labyrinth
- **Resolute**: Goal of the wayfinding task is efficiency and optimization reaching a destination in the most efficient manner; Example: finding items in a supermarket, navigating a new office building
- **Emergency**: Sole purpose of this type of wayfinding is reaching a destination as fast as possible under pressures such as time, stress or panic; Example: fire evacuation

The requirements for spatial aids to successfully carry out these types of wayfinding tasks differ depending on the task. For recreational wayfinding tasks, it may be beneficial to purposefully obscure or hide clues to increase the challenge in favour of the enjoyment of the task. Resolute wayfinding optimally has information such as route costs, obstacles and possible route options provided in order for an individual to arrive at an informed decision to find the most efficient routing. In emergencies, it is crucial for navigation aids to be as obvious, simple and precise as possible to not risk of an individual becoming confused.

To add to this, Wiener et al. [33] offer a taxonomy differentiating wayfinding tasks depending on the type of knowledge that is available.

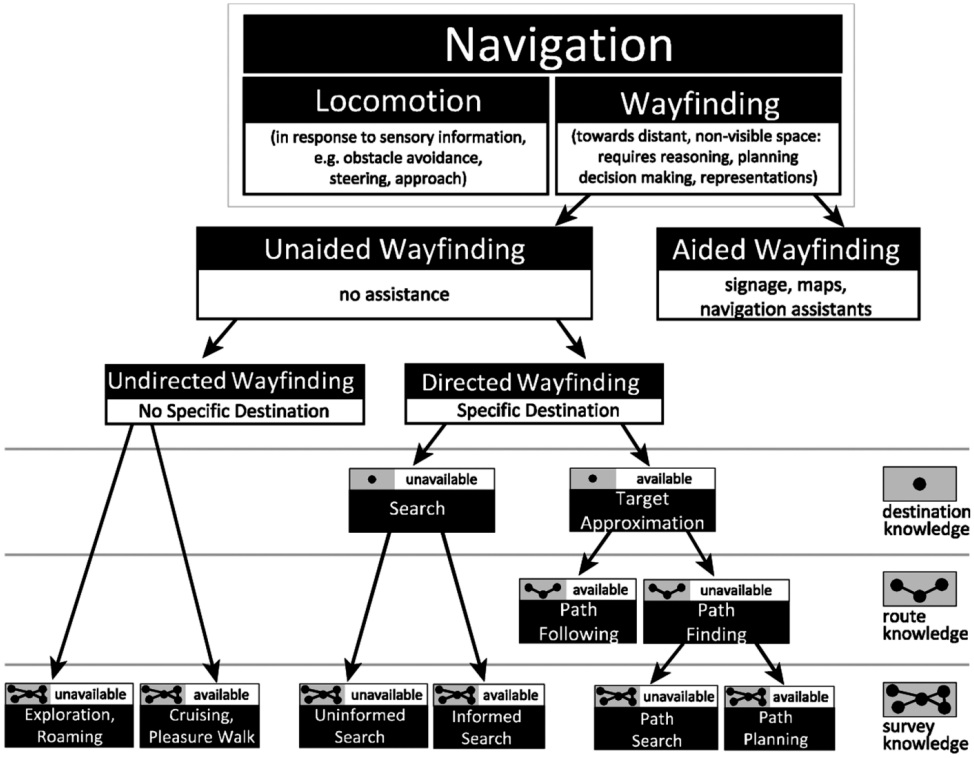

Figure 4.2: Taxonomy of wayfinding tasks differentiated by available spatial knowledge [33]

This is depicted by the flow diagram in Fig. 4.2. The act of navigating between places is first broken down into locomotion and wayfinding. Locomotion specifies navigational behaviour in "response to current sensory-motor input of the immediate surrounding and includes tasks such as steering, obstacle avoidance, and the approach of a visible object in vista space". In contrast to that, wayfinding requires planning, reasoning and decision-making to reach distant, non-visible space as previously mentioned. The wayfinding categorization branches out whether spatial knowledge is available to perform the wayfinding task. The types of spatial knowledge have been itemized into:

- **Destination knowledge**: Knowledge about a point in space such as a landmark or destination point
- **Route knowledge**: Knowledge about a sequence of points like a path or route to a

destination

- **Survey knowledge**: Knowledge about the spatial relation between at least two points; knowledge about an area

It is essential to identify the specific wayfinding tasks inside game loops, especially in open world games, in order to gain better insight on which navigational aids are required to properly navigate the game space.

With the foundation for wayfinding is established, in the following chapter the application of wayfinding in games is explored, how it manifests and its influences.

4.3.2 Wayfinding in Games

Wayfinding is an important task for players, as they need to be able to navigate the game environment first before they can begin to engage in gameplay and other content. As game worlds have become increasingly vast and complex, designers need to use wayfinding cues to inform players of where to go, especially when environments may not be inherently easily traversable [23]. It is apparent that designing wayfinding systems for games differ from wayfinding systems found in the real world in favour of, for example, exploration to progress [23]. Moura [10] identifies 3 major components that play an essential role in the wayfinding process in 3D game environments:

- **Game Environments**: Environments in which interactions between player and game world take place.
- **Wayfinding tasks**: Types of wayfinding tasks the player will face in order to progress the game.
- **Wayfinding cues**: Wayfinding cues added into the environment to provide information to the player.

The types of wayfinding tasks has already been expanded upon under chapter 3.3.1 as it will depend on the game which kind of wayfinding tasks a player might encounter. In the case of open world games, it is likely a player will encounter most if not all types of wayfinding tasks, especially when exploration is emphasized upon. In the following, the components of game environments and wayfinding cues will be elaborated and further examined.

Game Environments

Environmental or architectural design plays an important role in games to "support the gameplay" [10, 2] through providing challenges, puzzles or suggesting actions. It is not unusual for games to break the rules of conventional architectural design to create unique experiences, which necessitates the need to rethink environmental design with wayfinding in mind. Adams [2] proposes 4 ways a game environment supports and guides gameplay:

- **Constraint**: Architecture establishes boundaries to limit the player's freedom of movement to either create wayfinding challenges or provide guidance to steer the player somewhere else.
- **Concealment**: Concealment of, for example, enemies, important objects or items

creates opportunities of discovery. At the same time, subtle concealment such as through landscape or inconspicuous objects creates challenge by requiring the player to detect or discover the concealment.

- **Obstacles**: Obstacles such as traps, chasms or cliffs are common instances of obstacles to be overcome by a player and offers more challenge to them.
- **Exploration**: Environmental design can invite explorative challenges by requiring the player to understand the space in which they move in and, for example, learn which area leads to where. Subtle guidance such as sunlight coming through a window in a dimly lit area or a differently shaded patch of wall are named as well executed examples.

Moura [24] further highlights that, in addition to the environmental factors, designers must also take game design elements into account. Designers need to consider how much exploration the game environment should incite in context with the intended game experience, the amount of access the player will have to the game world and how much control the player has over visuals and camera. It is paramount to define how exploration and therefore wayfinding relates to the game at hand and take that into account when designing the environment.

The complexity of creating a virtual environment stresses the importance of wayfinding cues to further improve the wayfinding process once the environment itself becomes too complex to comprehend at first glance.

Wayfinding Cues

Wayfinding cues are "sensory information placed in game space by the designers to guide players when the environment becomes too convoluted to be navigated through its architectural features alone" [10]. Moura [24] offers an expansive list of navigational aids that details elements used in different games to support the wayfinding process. The navigational aids are categorized depending on what kind of information to aid the wayfinding process they provide to the player.

Marples [21] provides a similar list but suggests a different categorization: auxiliary and intrinsic cues. Depending on whether the wayfinding cue is embedded into the game world or stem from external elements that are not part of the games' environment, the type of cue is classified as intrinsic or auxiliary respectively. Auxiliary wayfinding cues such as pop-ups, markers, GPS or artificial are usually part of a Graphical User Interface (GUI) or Heads Up Display (HUD) which make up the interface element of a game. As a result, while auxiliary cues potentially disrupt immersion, they serve as crucial tools to provide vital information to the player that is not easily conveyed to the player through natural means. On the other hand, intrinsic wayfinding cues such as interactable objects, physical maps, doors, landmarks or lighting are more subtle in nature and usually work in a combined manner.

For the purpose of this paper, the distinction between auxiliary and intrinsic wayfinding cues will provide a sufficient framework to understand the environmental design of *BotW* and how its wayfinding cues work for an open world setting.

5 Analysis

After establishing the fundamental concepts, the following chapters will dive into the analysis of *BotW*'s design philosophies from the developer's perspective and how they shaped *BotW*'s open world experience.

5.1 Nintendo's Design Philosophy

In a video by Nintendo discussing the development process of *BotW* [25], Eiji Aonuma, producer for the *The Legend of Zelda* franchise, states the following:

"After all, people who play the Legend of Zelda games are often driven by their desire to explore. They're always trying to find secret places. That's why we decided to create a truly open-air world this time. We felt we really needed to make a game in which players could enjoy that sort of exploration. [...] I [...] already had the concept of an open-air world in mind back when we were making The Wind Waker. There, the idea manifested as a vast ocean through which players travelled from island to island by boat. You could pick a destination far away and work to get there so in that sense you could call it an "open air" environment". (Eiji Aonuma, 2017)

As *BotW* is the first "open world" title in the *The Legend of Zelda* franchise, the development team had a unique approach to tackle the design of open worlds. In a shareholder Q&A between Nintendo and its investors [22], Shigeru Miyamoto, game director and producer at Nintendo, states that he "prefer[s] not to use the generally used term "open world"" for *BotW* as it comes with pre-defined ideas from trends and established practices. Schnaars [29] dives into the "open-air" concept and explains the success of the concept through the game's quest system, world design, terrain navigation and emergent gameplay. Two particular aspects as part of the game's world design, the "triangle rule" and the concept of gravity, are further examined in the following chapters.

5.1.1 The Triangle Rule

From the 2017 CEDEC [32] talks, Nintendo showcases the concept of the "Triangle Rule" which manifests in *BotW*'s environmental design. They utilize triangles to model the environment, as the triangle serves 2 objectives: The player is given a choice between going around or over the triangle (Fig. 5.1). Since the triangle obscures the player's view, designers can employ them to surprise players as to what is behind the triangle and incentivize exploration through discovery (Fig. 5.2).

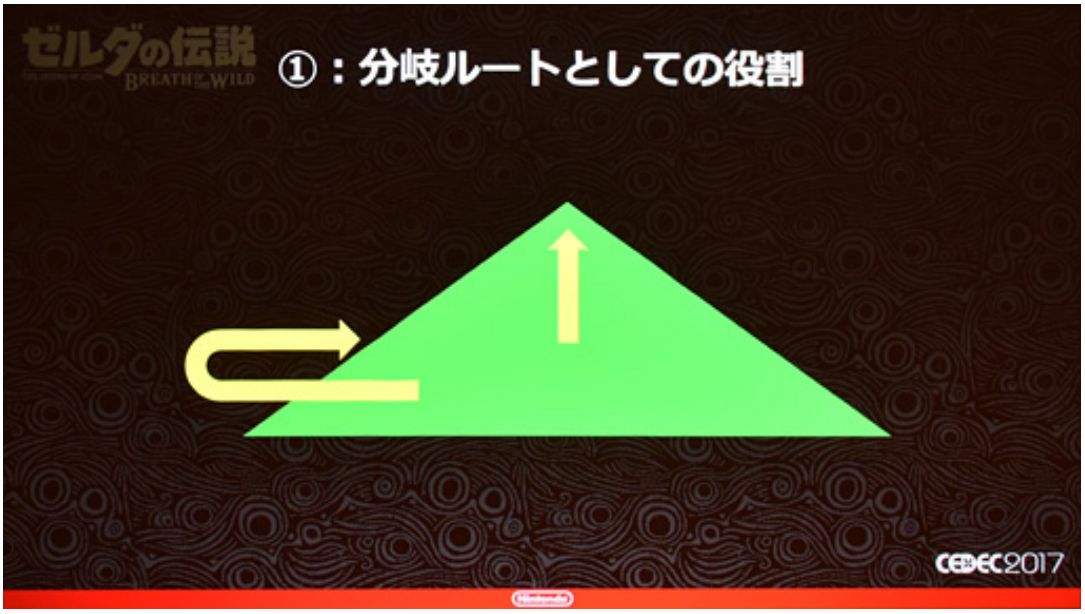

Figure 5.1: Triangle used to shape the environment and giving the player two choices of going around or over the triangle

.

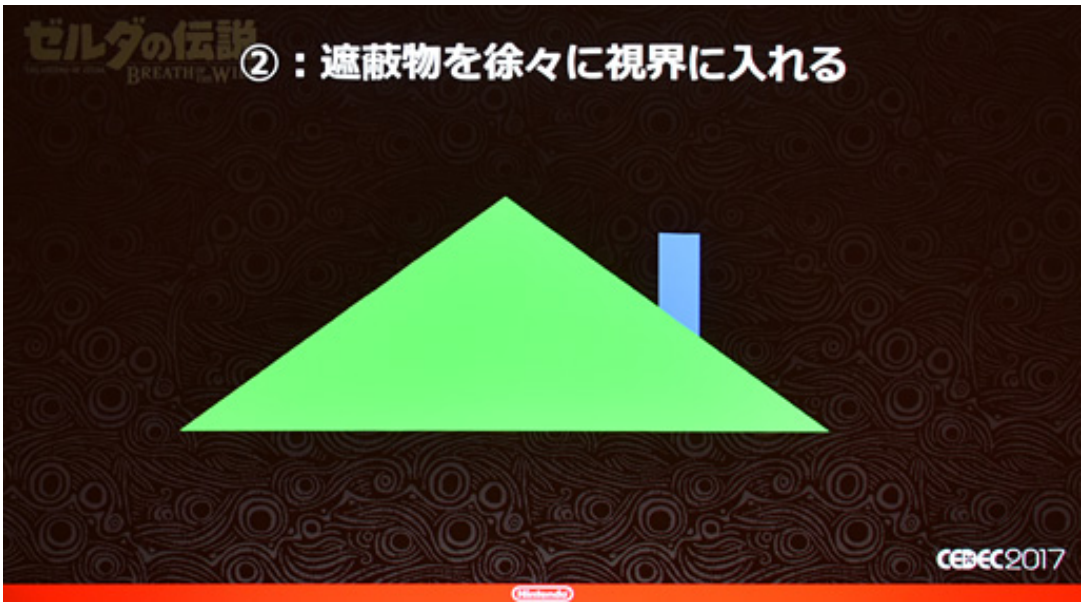

Figure 5.2: Triangle obscuring view of what is behind the triangle from the player.

Variations for these triangles exist to pique the player's interest and attract attention. Designers can apply this to place small puzzles or rewards to further incentivize exploration (Fig. 5.3). Nintendo further presents 3 different types of triangles varying in size to perform different tasks: The largest triangles act as a landmark and visual cue to aid in orientation and navigation. The medium-sized triangles serve to obstruct the player's view, concealing elements behind them and the smallest triangles serve the gameplay tempo affecting player inputs or adjusting the pace of the game by providing interaction opportunities (Fig. 5.4).

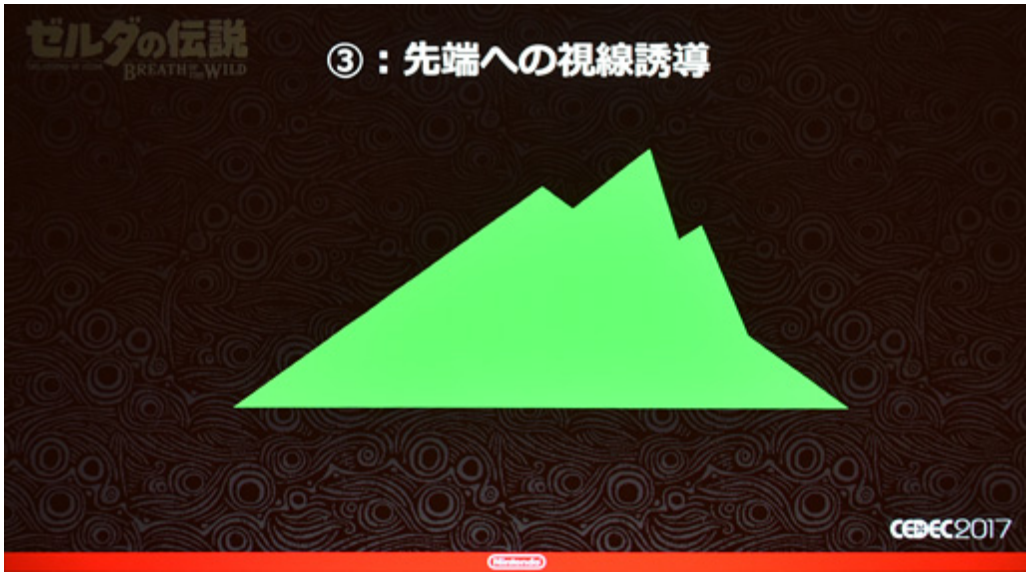

Figure 5.3: Example of uniquely shaped triangle model.

Figure 5.4: Showcase of the 3 utilized triangle types and their in-game examples.

Fig. 5.5 showcases the ubiquitous application of the triangle rule throughout *BotW*'s game world. Important to note here is the usage of rectangles as well as triangles in the environment modelling. Instead of gradually revealing an element in the environment, rectangles are utilized to fully hide an object from sight. This design strategy not only aids in navigation but also improves the immersion by manipulating visibility and engagement with the surroundings. The design process applying the triangle rule is demonstrated in Fig. 5.6 where a tower is initially placed and gradually obscured by applying the triangle rule to create a chain of interest (hill → bridge structure → tower).

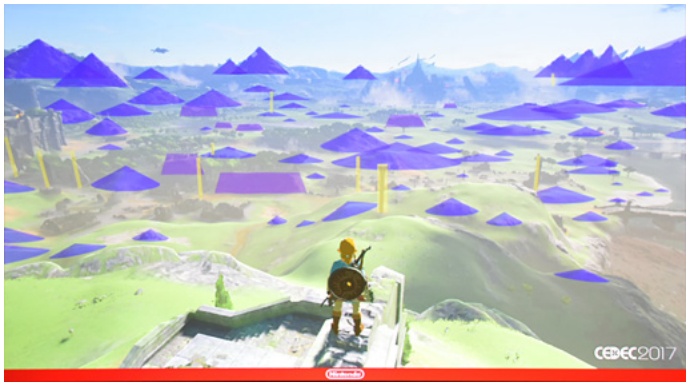

Figure 5.5: Instance of triangle rule used throughout the game world in *BotW*

(a) Tower without concealments

(b) Tower with added hills

(c) Tower with concealment structures

(d) Bridge structure revealed; hides tower

(e) Tower revealed behind hill and bridge structure

Figure 5.6: Gradual application of triangle rule, obscuring a tower in the background as one point of interest.

In summary, the triangle rule serves as a general design guideline to naturally incentivize exploration through creating curiosity. By providing a choice to the player through the

triangle concept, the player has the freedom to decide between different paths during exploration. Additionally, by concealing important environmental elements, it generates moments of discovery once the player reaches around the concealments. Through discovery, the player is provided more options and paths to take, further establishing opportunities for exploration.

In the following chapter, the concept of gravity is elaborated upon in conjunction with the triangle rule to explain the manner of how structures and events are placed in the world to facilitate explorative gameplay.

5.1.2 Gravity

To understand the idea of gravity [32], the tower structure as a game mechanic needs to be clarified. The tower acts as an important point of interest for navigational and gameplay purposes. Once activated by the player, the player is able to return to the tower at any given time in the game by teleporting to them, easing travel in already explored areas. Plus, the tower provides a useful vantage point, enabling survey of the local area from an elevated position. They are visible from far away due to their distinct look and height, lending itself as a reference point for orientation.

Lastly, a tower serves the essential purpose of revealing parts of the map to further ease the wayfinding process for the player. As seen in Fig. 5.8, each part of the map is initially obscured and needs to be revealed by activating the tower in each respective region. Once a tower is activated, the respective region of the tower is revealed and displays the topology and structure of the local area (Fig. 5.9).

Figure 5.7: Tower structure used to reveal the map, acts as a teleport point for travelling and serves as a vantage point.

Figure 5.8: The map of the world at the start of the game.

Figure 5.9: A section of the map revealed by activating the tower in the region.

Taking that into account, the tower structure exhibits substantial incentive for the player to reach the towers, which plays a pivotal role in the world design. The initial attempt by Nintendo to structure *BotW* the map was to spread these towers throughout the world and place game events in between without the concept of the triangle rule. After conducting playtests, player's felt the game was too linear and too guided, as they would either traverse from tower to tower or explore in an entirely different direction and get lost.

Figure 5.10: Example of structure placement with gravity in mind in between 2 towers.

As a solution to this issue, they introduced a concept termed "gravity". Each structure or game event is assigned a ranking based on its visibility and significance to the player, and is purposefully positioned to allow the player to be distracted while progressing toward a specific objective. Fig. 5.10 shows a model-like application of gravity between 2 towers. Starting from the left tower, players might discover the cave on their way to the tower on the right and faces the option to either explore the cave or remain on the previous path. If the player chooses to visit the cave, they might find other points of interest such as a small puzzle (top most icon) or the horse stable (right of the puzzle icon). They can freely elect to do either point of interest or dismiss both to resume on the original path based on their preference for gameplay.

The aspect of visibility extends to its distinct silhouette and appearance and is not limited to its size. Each structure has to be easily identifiable by the player in order for them to decide whether they wish to visit it or not. Fig. 5.11 showcases the prominent game structures and their visibility based on size, unique features and silhouette appearance. From left to right, the structures in Fig. 5.11 represent the following:

1. **Mountain**: Landmark, reference point, vantage point.
2. **Tower**: Landmark, reference point, vantage point, teleport point, reveals portion of the map.
3. **Korok seed puzzle**: Small puzzle, gives currency to expand inventory size. Indicated by a smoke plume going up.
4. **Horse stable**: Register/Fetch horse for travel, merchants, side quests, other Non-Player Characters (NPCs). Indicated by horse-shaped roof.
5. **Enemy encampment**: Combat encounters, opportunity for equipment, weapons and treasure. Indicated by a skull-shaped canopy.
6. **Shrine**: Small dungeon with puzzles or combat encounters, grants currency to increase maximum health or stamina, potentially treasure. Indicated by glowing rock entrance.

7. **Chests**: Contains useful items such as weapons, cooking ingredients, materials or treasure.
8. **Mushroom**: Cooking ingredient, used for recipes that grant buffs or recovers health or stamina.

Figure 5.11: Most prominent structures and game events in *BotW* ranked after visibility.

In contrast, the player's priority order is influenced by their gameplay preferences. For example, A combat-orientated player would prioritize events such as shrines and enemy encampments to increase health or obtain superior gear (Fig. 5.13).

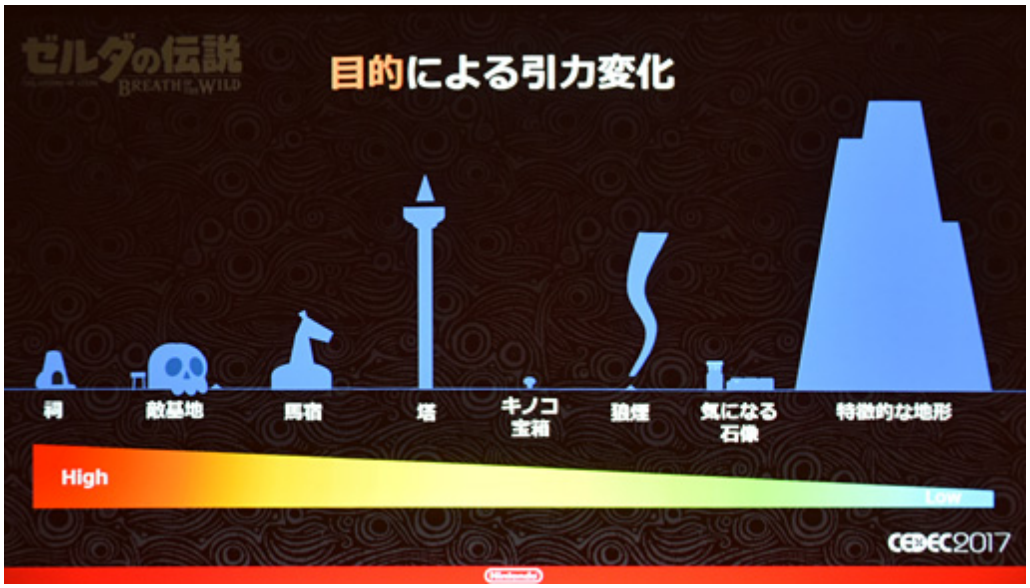

Figure 5.12: Example priority of structures and game events to a combat-orientated player.

This cycle of repeatedly visiting newly discovered points of interest continues until the player either reaches the right tower or opts to pursue a different goal at any point of

the route. This is made possible by employing the triangle rule in between the encounters for which the visibility of each encounter is paramount. By having the game events be discovered organically by the player and their location obscured by ways such as the triangle rule, the player is incentivized to explore and reveal the structures themselves and regularly experiences the feeling of discovery. Additionally, the importance of variety in game events or structures is equally emphasized as to offer different content to any type of player. If a player only encounters events they would rather avoid, then the discoveries will feel unrewarding and exploration becomes less engaging.

Figure 5.13: Left heat map: Paths players took without the concept of gravity; Right heat map: Player paths after the implementation of gravity. Traversed paths taken by players became more well distributed after the implementation of the gravity concept.

This stands in contrast to the *Assassin's Creed Unity* map in Fig. 1.1 where each point of interest is marked preemptively on the map and a high density of similar game events is present potentially taking away the appeal of exploration and discovery. In *BotW* however, the map simply displays the topology of the game world and only marking visited distinct locations such as towers and shrines to which the player can teleport to.

In conclusion, gravity serves as a crucial concept to incentivize exploration through organic moments of discovery. Coupled with the triangle rule, game events and structures are intentionally obfuscated depending on their visibility level in order for the player to find them themselves. From one point of interest to the next, a cycle of discovery is created by identifying new points of interest from a previous one keeping the player engaged by their own volition without becoming lost. The intentional concealment of important game elements is essential to not take away the moments of discovery by preemptively showing the player where points of interest are located. This allows for subtle guidance between major game events without the need for explicit guidance through, for example, auxiliary wayfinding cues preserving the opportunities for exploration.

5.2 In-game Application

To gain a deeper understanding of the aforementioned concepts, this section examined concrete examples from the game and explore their impact on the game experience. The player initially awakes in a cave and after an introductory sequence to establish basic input controls, the player is met with a view of the game view for the first time (Fig. 5.14).

Figure 5.14: First view of the game world after leaving the cave the player wakes up in.

The most notable points of interest in this initial impression to the player of the game world are circled in red: A distinct looking mountain on the left, a castle structure in the middle and a volcano on the right. As they are visible from far away and can serve as landmarks throughout the game, they are instances of the largest triangle in the triangle rule concept. The uniquely shaped mountain and the volcano are also showcased as examples for the largest triangle in Fig. 5.4.

The elements circled in green are less noticeable as they seem to be mere environmental elements and part of the landscape. However, these represent the medium-sized triangles and rectangles meaning they act as concealments for other game elements. The leftmost rectangular shaped hill hides a structure resembling the architecture of the Colosseum which contains strong enemies not meant to be encountered at the start of the game (Fig. 5.15). On top of the middle hill a Korok seed puzzle can be found after the triangle rule principle of playing events on top of a triangle to further incentivize exploration. At the same time, behind the hill an enemy encampment is revealed following the purpose of the medium-sized triangle (Fig. 5.16). Lastly, the right hill hides ruins and another Korok seed puzzle also acting as a medium-sized triangle (Fig. 5.17). The structure marked with a blue circle will be elaborated upon later in this chapter. It is noteworthy how seamlessly it blends with the environment rendering it seemingly unimportant within this view of the scene.

Figure 5.15: Structure hidden behind a rectangular hill according to the triangle rule.

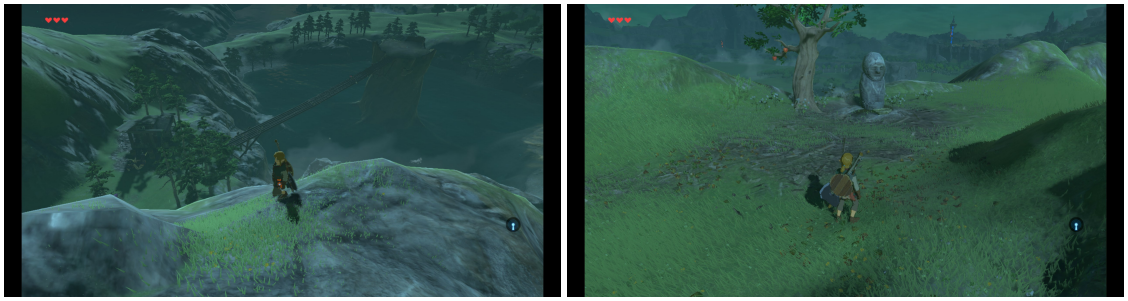

Figure 5.16: Enemy encampment hidden behind a medium-sized triangle after the triangle rule. The top of the hill reveals a Korok seed puzzle.

Figure 5.17: Ruins and Korok seed puzzle concealed behind another medium-sized triangle following the triangle rule.

The starting portion of the game plays out linearly as the player is introduced to the core game mechanics and important structures by an NPC. The area is also limited to an elevated plateau, which the player is unable to leave due to the height. Through the course of the starting area, the player is made familiar with the tower structure and their benefits, and they start appearing throughout the game world. After activating the first tower on the starting area, the player reveals the map portion for the introductory plateau area, gains access to the teleport point of the tower and the tower's vantage point to survey the area. The following figures show the views from the first tower:

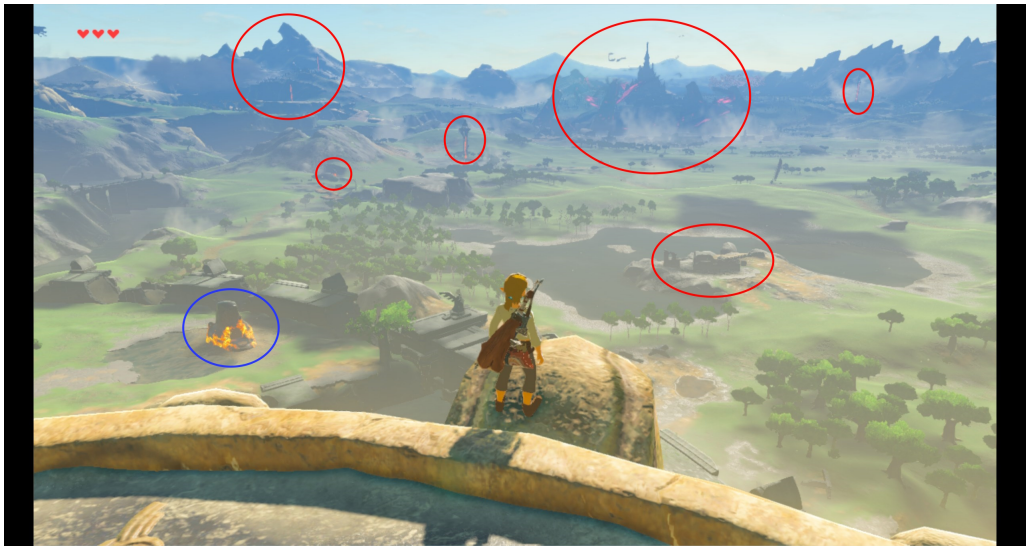

Figure 5.18: One perspective of the view around the first tower, showing shrines, towers and landmarks.

Figure 5.19: Another perspective of the view around the first tower showing another tower, shrines and an enemy encampment indicated with the skull-shaped rock.

Most notably are the shrines and towers that appear in a yellow-orange glowing hue,

contrasting the green, brown and grey landscape palette. Both perspectives of the views shown in Fig. 5.18 and Fig. 5.19 have the notable points of interest circled in red which consists of landmarks, towers, shrines or other apparent structures such as an enemy encampment or ruins. The game relies less on active guidance through auxiliary wayfinding cues and more through organically drawing attention to distinct structures by contrasting colours or uniquely shaped structures.

The shrine highlighted with a blue circle in Fig. 5.18 corresponds to the same structure circled in blue in Fig. 5.14, the primary distinction being the yellow-orange glowing hue visible after the activation of the first tower. This is intentionally designed to avoid diverting the player's attention from the scene view and its distinct landmarks, while also preventing the player from being misled toward locations that are not yet intended to be explored. A similar approach is applied to the structures obscured using the triangle rule (Fig. 5.15, 5.16, 5.17), as they are meant to be discovered by the player at a later point in time. Revealing them immediately would diminish the player's feeling of discovery.

One last example showcases the application of gravity in tandem with the triangle rule. After the player finishes 4 shrines on the starting area introducing the player to the core mechanics of the game, the player is given the paraglider from the NPC at the beginning which allows the player to safely leave the elevated plateau. In Fig. 5.20, the player is located at the edge of the plateau, surveying the lower area. In a hypothetical scenario where the player attempts to reach the tower circled in red by heading directly towards it, the application of gravity utilized along the path can also be observed. Although the view from above suggests no obstacles or event along the path, there are numerous opportunities for the player to encounter discoveries.

The following screenshots show the path if the player does not deviate from the path heading towards the tower. Interesting points of interest are marked and elaborated upon in the context of the gravity concept and the triangle rule.

Figure 5.20: The player standing at the edge of the elevated plateau, surveying the area. Tower as hypothetical destination circled in red.

(a) Forest with enemies immediately below the elevated plateau.

(b) Area behind the forest. Destination circled in blue, points of interest marked in red.

(c) On top of the hill circled red in the previous image. Destination circled in blue, points of interest marked in red.

Figure 5.21: Snapshots of stops on the path towards the marked tower as a destination. Points of interest marked respectively.

(d) On top of the same hill, observing towards the right.

(e) Past the hill in front of a forest with a small enemy encampment.

(f) Behind the forest and past the enemy encampment. Points of interest marked in red.

Figure 5.21: Snapshots of stops on the path towards the marked tower as a destination. Points of interest marked respectively. (cont.)

(g) On top of the hill marked in the previous image by the left circle. Points of interest marked in red.

Figure 5.21: Snapshots of stops on the path towards the marked tower as a destination. Points of interest marked respectively. (cont.)

Immediately after leaving the elevated plateau and entering the forest below, the player is met with enemies they can choose to engage or escape from (Fig. 5.21a). Beyond the forest are grasslands with several visuals cues that draw the player's attention (Fig. 5.21b. The hill in the middle serves as another instance of the medium-sized triangle of the triangle rule, posing the options of going on top or around the hill to the player. To the right of the hill, the player can identify a glimpse of a shrine structure due to its contrasting yellow-orange glow. To the left of the hill, a subtle path is visible, which the player can likewise choose to pursue. Going on top of the hill exposes ruins containing enemies behind (Fig. 5.21c and to the right of the hill (Fig. 5.21d. The previously seen shrine becomes more apparent to the player and more shrines are identifiable in the distance from the hill as a vantage point. Following the path to the left of the hill or venturing beyond the hill towards the tower destination, the player stumbles upon another small forest with an enemy encampment (Fig. 5.21e). Past the forest and the enemy encampment along the previously observed path, another hill with a Korok seed puzzle to the left of the path and a bridge at the end of the path can be spotted (Fig. 5.21f). On top of the hill with the Korok seed puzzle, the destination tower is once more visible and several other points of interest such as shrines and a horse stable can be noted (Fig. 5.21g).

While this is not the entire path towards the destination, it serves to show the application of the gravity concept and the triangle rule. At most points of the path, several points of interest can be observed which draw the player's attention to be explored, which follows the gravity concept. Through that, the player can either choose to focus on the goal at hand (reaching the destination tower) or pursue other activities such as shrines, ruins, engage enemies or other towers. If the player decides to pursue other tasks and resume the previous goal later, the player can easily orient themselves along landmarks such as the mountain that seemingly is split in half (Fig. 5.21b, 5.21c) or the volcano (Fig. 5.20, 5.21g), shrines, towers or other prominent structures such as the horse stable. The triangle rule aids in obscuring points of interest that are further away from the player's immediate area and providing an opportunity of discovery for the player to discover the concealed points of interest. At the same time, it also serves as vantage points for

re-orientation or survey of the area to aid the wayfinding process.

5.2.1 Takeaways and Results

The analysis of the gameplay path showcases the effective implementation of both the gravity concept and the triangle rule in guiding player exploration and providing opportunities for discovery. The combination of visual cues and the strategic use of terrain elements offers players multiple points of interest, which naturally draws attention and incentivizes exploration beyond the primary objective. Visual cues such as glowing shrines, hills, enemy encampments or other towers serve gravitational forces, subtly guiding the player's attention to these different locations without directly interfering with the primary goal of reaching the tower. This design enables players to decide whether to follow the intended path, pursue other points of interest, or explore alternate routes. It demonstrates how environmental design can influence player behaviour through subtle means, encourage organic exploration and preserve the player's freedom of choice. The use of gravity and the triangle rule ensures that exploration remains rewarding without overwhelming the player with too much information at once.

The triangle rule is utilized to create layers of visibility, obscuring distant points of interest until the player reaches a higher vantage point, such as the top of a hill. This design approach maintains the element of surprise and rewards exploration with the discovery of previously hidden landmarks, such as shrines or ruins. Additionally, vantage points offer opportunities for re-orientation, enabling players to regain their sense of direction after engaging with different tasks. This is achieved through the careful placement of recognizable landmarks like mountains, volcanoes, and towers, which all serve as intrinsic wayfinding aids.

Another key point to consider here is the designer's intention in arranging a specific view, determining which structures are meant to be immediately visible and which are intended to be discovered later. Questions such as "Which point of the interest is the player supposed to see here?" or "What point does this view serve to the player?" could be raised while crafting a scene. It needs to be considered whether an encounter, structure or event is supposed to be observed from further away or is supposed to be discovered close by, which in turn affects the scene composition of vantage points.

Both the gravity concept and the triangle rule aim to strike the balance of the tensions described by Hughes & Cairns [18] in 4.1. By providing a guideline to populate a large, open world with engaging activities to ensure content density and variety and at the same time not overwhelming the player with options, it solves the inner tension of theme 2 and at the same time tackles an issue that theme 4 might face. Additionally, both paradigms offer a solution to tackle the agency vs direction tension by maintaining the player agency through the unconstrained environmental design, incentivizing exploration and providing guidance by utilizing intrinsic wayfinding cues.

In the following chapter, the gravity concept and triangle rule are examined in context to the results from the in-game analysis, with the aim of developing tools to aid the level design process for general open-world applications.

6 Tool Development

6.1 Conceptualization

This section focuses on translating the previous findings into tangible tools that improve the level design process for open-world games. Several aspects from the earlier analysis could be implemented into tools to apply the same strategies into different games. While developing frameworks specifically for the gravity concept or the triangle rule could enable direct implementation of these methods, such frameworks may be too rigid or specialized to be applicable to open-world games in general. The triangle rule is derived from natural landscapes with the triangles representing hills and mountains to design the environment, while the environment design may differ from game to game. At the same time, the gravitational force of a point of interest to draw the player's attention is difficult to measure, rendering comparisons between points of interest's gravitational force infeasible.

With that said, an underlying element of the analysis that encompasses both the gravity concept and the triangle rule is the notion of **visibility**. Both concepts suggest that visibility is fundamental to exploration and wayfinding in open-world game design as they are both concerned with managing visibility of points of interest in a manner that facilitates guidance and discovery, while also ensuring the player is not overwhelmed by needless choices. It would prove useful to build tools that visualize the visibility of specific elements or objects in the game world given a player's position during the level design process.

Such tools would allow designers to simulate and analyse how game elements such as landmarks, waypoints or objectives are perceived from different player angles within the environment. Additionally, these tools would provide immediate feedback on the visibility of objects, which are obscured by the environment, and how changes in player position or movement might influence the overall visible landscape. This would ease the decision-making process of arranging the spatial structure of points of interests to ensure a similar flow in exploration as showcased in *BotW*. By providing information on the visibility of objects, this tool also aids in the implementation of the gravity concept and triangle rule. Once designers are able to determine which objects in the environment are visible, they can strategically place concealment elements, such as terrain features as stated in the triangle rule, to guide player attention. This allows for application of the gravity concept, ensuring that key points of interest attract the player's attention as intended and maintaining a balance between visibility and discovery.

As the chosen platform is Unity and the developed tools' purpose is to support the level designing process, the tools will affect the level editing process during Edit Mode in Unity.

6.2 Approach

The initial approach used to implement a visibility metric involved utilizing Unity's *Universal Render Pipeline (URP)* to tag game objects for visibility testing, assigning each object a unique colour based on its ID, and rendering them onto a *RenderTexture* object. This approach required the use of two cameras: one for rendering the entire scene and another dedicated to rendering the objects being tested. Both cameras would render their respective images onto *RenderTextures* instead of directly displaying them to enable the calculation of colour differences between the two images. The pixelwise colour difference would yield the visibility percentage of each tested object.

However, this approach depends on the current camera view to compute visibility scores, making it unsuitable to use during Edit Mode. Additionally, it only accounts for objects within the camera's field of view, limiting visibility calculations to what is captured by the camera. The approach also posed challenges related to the results and the potential errors that might occur due to environmental features. Since this method relies on colour difference calculations, if the environment around the object under test happens to show the same colour as the object's colour assigned to it by its ID, the results might become faulty. Due to these constraints, this method was discarded in favour of a more versatile solution that could better suit the level design process.

To address these limitations, an alternative method was explored, leveraging the Monte Carlo method [17] to perform visibility calculations independent of the camera's view during Edit Mode. Rather than relying on real-time camera rendering, a raycasting system is implemented to perform occlusion checks between the player's position and the objects under test. By firing multiply raycasts from an object's surface to the player's position, an estimate of the object's visibility can be determined. This method would allow for visibility tests from any point in the game world, including regions outside the immediate camera's field of view. Furthermore, it provides robustness in the form of enabling simulations independent of the environment's features and characteristics, such as colour or shape.

Moreover, the updated approach would be usable during Edit Mode, enabling designers to assess the visibility of objects or points of interest from different positions in the environment without the necessity of being in play mode. This real-time feedback facilitates more informed decision-making regarding object placement and line of sight during the level design process. By allowing designers to flexibly simulate and evaluate visibility in diverse scenarios, this method contributes to more efficient and effective level design decision-making.

The following chapter outlines the implementation process of the visibility metric using the Monte Carlo-based raycasting approach. It details the setup of the raycasting system, including the configuration of the ray origins, directions, and the criteria for determining occlusion. The chapter will also describe how the Monte Carlo method is applied to statistically estimate visibility percentages, integrating randomness to simulate various potential viewpoints and environmental conditions.

6.3 Implementation

The implementation process is separated into two primary scripts[1]: the script for the point of interests from which the visibility score is calculated and a script to flexibly position a point of reference to where the visibility is determined from. The first chapter outlines the implementation of the point of interest script, which handles the computation of the visibility score given a point of reference. The second chapter deals with the dynamic placement of a point of reference object.

To be able to run scripts in the Edit Mode in Unity, the decorator *[ExecuteInEditMode]* is needed above the class declaration in order to be executed during Edit Mode. This is the case for both scripts, as the execution of both scripts is essential for calculating visibility scores. The point of interest script must be attached to each game object for which the designer requires visibility information, while the point script only needs to be included in the project to be utilized.

6.3.1 Point of Interest Script

The following script can be applied to any game object in Unity for which the designer aims to determine the visibility score. By accessing the game object's mesh, the script retrieves the vertices and implicit information about the polygons constituting the mesh. Before initializing the raycasting system, the script first conducts a preliminary check on the object's bounding box to assess whether the object is fully occluded. This is achieved by casting raycasts from each corner of the bounding box towards the reference point, from which the visibility will be calculated. If the bounding box is entirely occluded, it is assumed that the entire game object is obscured, rendering further checks unnecessary.

If the initial check concludes that the bounding box is not fully occluded, the script proceeds to set up the raycasting system. For each triangle in the mesh, the normal vector of the triangle's surface is calculated via the cross product of two of its edges. If the dot product of the normal vector and the direction vector from the reference point to the triangle is negative, the triangle does not face the reference point and is excluded from further calculations. Simultaneously, the surface area of each triangle is computed and discarded in the same way if the triangle is not facing the reference point.

The script then samples points from the visible surface area of the mesh. Each visible triangle is selected for sampling based on a probability proportional to its surface area relative to the total surface area of all visible triangles, ensuring uniform sampling across the entire visible surface. Once a triangle is selected, a random point on its surface is determined by generating two random factors between 0 and 1, multiplying each factor with two of the triangle's edge vectors respectively, and adding up the results. If the sum of the two factors exceeds 1, a transformation is applied to ensure that the point remains within the triangle.

This process is repeated until a desired number of points have been uniformly sampled from the surface area. Raycasts are then shot from each sampled point towards the reference point, and the number of intersecting rays is counted. The ratio of intersecting rays to the total number of rays results in the initial visibility score.

[1]Both scripts can be found in the appendix or in the following repository: `https://github.com/Lukaboka/Master-Thesis`

To account for the effect of distance on perceived visibility, the script incorporates a distance factor. Even if a small object is fully visible, it may remain difficult to notice if it is far from the reference point. Thus, the script computes the perceived visible surface area by multiplying the total visible surface area by the visibility score and dividing by the distance between the reference point and the game object.

Finally, once the perceived visible surface area is determined, a label is drawn above the object in the scene to provide immediate feedback to the designer regarding the object's visibility level. The labels include "Not Visible," "Slightly Visible," "Moderately Visible," "Very Visible," and "Fully Visible." This process completes the calculation of the visibility score for the object of interest.

The script also offers customization options in the *Inspector* to improve the usability for designers. Parameters such as the number of raycasts, the option to display the visibility label, the label's font size and colour, and an option to visualize all performed raycasts can be customized. These settings allow for adjustments of the functionality without the need to modify the underlying code, making the tool flexible and adaptable to specific level design requirements.

6.3.2 Pointer Script

To calculate the visibility score, a reference point must be placed from which visibility is determined from. This script facilitates the dynamic placement of such a reference point. It does so by casting a ray from the cursor into the scene and attempting to place a reference point object on the surface of the first object that the ray hits. Once the reference point is created, all instances of the point of interest script are notified and receive a reference to the new object. Upon acquiring this reference, the visibility calculation for each point of interest object begins, providing the designer with immediate and continuous feedback.

The script also automatically deletes the previous reference point whenever a new one is created, updating the references in all point of interest objects accordingly. Additionally, the designer can suspend the pointer script by pressing "P," which allows for manual adjustment of the current reference point or temporary deactivation if the feature is not needed at a point in time. This functionality provides designers with a flexible and dynamic means of placing a reference point to calculate visibility scores in real-time for all objects with the point-of-interest script attached.

This concludes the description of the visibility tool, which provides visibility information from any point in the game environment. The following sections contextualize the significance and value of this tool with regard to the goals of this research and evaluate the application of *BotW*'s design philosophies within the open-world genre.

7 Discussion

The research goal of this paper was to develop improved tools for level designers to better assess design choices in open world games. Using Nintendo's *The Legend of Zelda: Breath of the Wild* as a case study, various strategies and methods were analysed to identify the paradigms that enhance the open world experience. Key among these are the gravity concept and the triangle rule, both of which play a crucial role in *BotW* environmental design. These principles offer opportunities for discovery to the player, maintain the player's agency and freedom, and provide guidance through intrinsic wayfinding cues. To support the implementation of these strategies, the visibility tool was developed. This tool provides essential visibility information for specific objects in a game world, enabling designers to easier apply concepts like the gravity concept or triangle rule, both of which manipulate visibility to intentionally obscure or highlight objects, encouraging exploration and drawing the player's attention.

7.1 Findings contextualized within research goal

As highlighted by Hughes and Cairns [18], an enjoyable open-world experience is not only a large environment filled with objectives and a narrative, but involves far more nuance. Key elements of an engaging open-world design include balancing player agency with direction, maintaining content density without overwhelming the player, and ensuring content variety. Breath of the Wild balances these tensions through the use of intrinsic wayfinding cues, visibility manipulation, and strategic environmental design, for instance through the gravity concept and triangle rule, subtly guiding players toward points of interest while preserving their sense of agency. In contrast, the heavy use of auxiliary wayfinding cues, as seen in *Assassin's Creed: Unity*'s map (Fig. 1.1), can negatively impact the open-world experience by limiting opportunities for exploration and diminishing player agency. This highlights the importance of providing minimal constraints on the player while offering sufficient guidance through unobtrusive methods, falling in line with the initial hypothesis of this paper.

7.2 Application for open world games in general

The gravity concept and triangle rule can be fundamentally applied to open world games, as their core principle revolves around the strategic obfuscation of environmental elements to incentivize exploration and provide opportunities of discovery. By carefully designing the environment so that only the most essential elements are visible to the player, designers can organically guide the player's attention, leading them toward their objectives in a subtle manner. To achieve this, designers must address key questions such as, "What is the purpose of this scene?", "Which elements must be visible?", and "Which elements are intended to be discovered?" This approach ensures that open worlds bal-

ance player agency while providing sufficient guidance. The visibility tool developed in this paper facilitates this decision-making in response to these design challenges, allowing designers to apply these principles more effectively. Moreover, this tool and its underlying paradigms can be extended beyond open world games, proving applicable to general level design as guidance plays an important role in various game genres.

7.3 Challenges and Limitations

Despite the broad applicability of the gravity concept, triangle rule, and visibility tool, there are challenges and limitations that must be considered when implementing these methods into different game types. One potential limitation is the reliance on visual cues to guide players, which may prove ineffective in games which are less visually focused or that prioritize auditory or tactile feedback (such as virtual or augmented reality games). In these cases, additional layers of guidance may need to be developed that align with different sensory experiences of these games.

Another challenge is ensuring that the balance between guidance and player freedom remains appropriate for the target audience. Games with highly diverse player bases may need to consider different play styles and preferences. Some players may prefer more explicit or auxiliary cues, while others might enjoy more abstract or minimal to no guidance. It is essential to recognize that effective level design varies between games and needs to be carefully considered within the context of target audience, intended game experience and underlying game mechanics.

While level design is crucial in enhancing the open world experience, this paper does not address other important factors such as narrative, quest design, game mechanics, or additional game elements. Although the gravity concept and triangle rule are integral to *BotW*'s environmental design and contribute to its success, the game also thrives due to its unobtrusive narrative, subtle quest design, and emergent gameplay mechanics, all of which contribute to the open world experience. It remains unclear to what extent level design alone influences the overall experience, but it is important to recognize the impact of these other contributing factors in BotW's success.

7.4 Future Research and Tool Development

The findings and tools developed in this research present opportunities for future exploration and optimization. Expanding the visibility tool to account for dynamic game elements, such as moving objects, would increase its functionality in more complex game worlds. Additionally, including player feedback or heat map data into the tool could offer real-time insights into how players engage with the environment, allowing designers to make iterative adjustments that align with practical player behaviour. The tool could also be expanded upon by displaying visibility radiuses around a point of interest, indicating from which locations the point of interest remains visible or is obstructed by the surrounding local environment. In other non-traditional media such as augmented or virtual reality, designers could integrate other sensory cues such as auditory or haptic feedback to enhance the player's interaction with the environment in other ways other than visual cues. Another potential direction for future research lies in developing a framework to directly implement the gravity concept or triangle rule. Although these

concepts may be too specific to certain game scenarios, it might be insightful to experiment whether they can be applied in other game genres outside of open world games.

8 Conclusion

This paper has explored vital strategies in open world design, with a focus on the gravity concept and triangle rule as applied in *The Legend of Zelda: Breath of the Wild*. These principles, along with the visibility tool, offer valuable guidelines and methods for guiding player behaviour while preserving agency and freedom in game exploration. The findings demonstrate that strategic obfuscation and visibility manipulation are powerful ways to creating immersive and engaging open world experiences where exploration is incentivized.

The gravity concept and triangle rule provide universal guidelines that can be applied to open world games. The visibility tool, proposed to aid in the implementation of these concepts, serve to improve the level design workflow, allowing them to make informed decisions about how players perceive and interact with their environments. By leveraging these insights, game developers can create more engaging worlds that foster explorative gameplay and curiosity while maintaining player agency.

Ultimately, the research presented in this paper contributes to the broader field of game and level design by offering tools and methodologies that support the creation of more explorative and player-driven open world experiences. As game worlds continue to grow in complexity, these principles will remain vital in helping designers craft open world environments more efficiently, improving the overall game experience.

List of Figures

Bibliography

[1] Espen Aarseth. From hunt the wumpus to everquest: introduction to quest theory. In *International Conference on Entertainment Computing*, pages 496–506. Springer, 2005.

[2] Ernest Adams. The Construction of Ludic Space. In *DiGRA Conference*, 2003.

[3] Ryan Alexander and Chris Martens. Deriving quests from open world mechanics. In *Proceedings of the 12th international conference on the foundations of digital games*, pages 1–7, 2017.

[4] Paul Arthur and Romedi Passini. *Wayfinding: people, signs, and architecture*. 1992.

[5] Richard A Bartle. *Designing virtual worlds*. New Riders, 2004.

[6] Cliff Bleszinski and Epic Games. The art and science of level design. In *Game Developers Conference*, 2000.

[7] Marc Bonner. Ambiguous play pattern: A philosophical approach to the prospect-refuge theory in urban open world games by merging deleuze/guattari and de certeau. *space*, 106:108, 2014.

[8] Marc Bonner. On Striated Wilderness and Prospect Pacing: Rural Open World Games as Liminal Spaces of the Man-Nature Dichotomy. In *Digra conference*, 2018.

[9] J. Clement. Lifetime unit sales generated by The Legend of Zelda: Breath of the Wild on Nintendo Switch worldwide as of June 2024. `https://www.statista.com/statistics/1248052/zelda-botw-unit-sales/`, 2024. Accessed: 2024-08-26.

[10] Dinara de Moura Barbosa. And then you hit play: Investigating players' responses to wayfinding cues in 3D action-adventure games. 2017.

[11] Ian Thomas Dunn. Procedural generation and rendering of large-scale open-world environments. Master's thesis, California Polytechnic State University, 2016.

[12] Anna Charisse Farr, Tristan Kleinschmidt, Prasad Yarlagadda, and Kerrie Mengersen. Wayfinding: A simple concept, a complex process. *Transport Reviews*, 32(6):715–743, 2012.

[13] John Feil and Marc Scattergood. *Beginning game level design*. Thomson Course Technology, 2005.

[14] Rodney Fewings. Wayfinding and airport terminal design. *The journal of navigation*, 54(2):177–184, 2001.

[15] Nate Fox. Building an Open-World Game Without Hiring an Army. `https://www.gdcvault.com/play/1012448/Building-an-Open-World-Game`, 2010.

[16] Tianhan Gao, Jin Zhang, and Qingwei Mi. Procedural generation of game levels and maps: A review. In *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 050–055. IEEE, 2022.

[17] John Hammersley. *Monte carlo methods*. Springer Science & Business Media, 2013.

[18] Nathan G.J. Hughes and Paul Cairns. Opening the world of contextually-specific player experiences. *Entertainment Computing*, 37:100401, 2021.

[19] Ahmed Khalifa, Fernando de Mesentier Silva, and Julian Togelius. Level design patterns in 2D games. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[20] EDGE Magazine. Elden Ring - The Hidetaka Miyazaki Interview. https://www.yumpu.com/news/en/issue/120123-edge-issue-022022, 2021. Accessed: 2024-08-23.

[21] Daryl Marples. *The Influence of Intrinsic Perceptual Cues on Navigation and Route Selection in Virtual Environments*. PhD thesis, University of Huddersfield, 2017.

[22] Shigeru Miyamoto. The 74th Annual General Meeting of Shareholders. https://www.nintendo.co.jp/ir/en/stock/meeting/140627qa/index.html, 2021. Accessed: 2024-09-14.

[23] Dinara Moura and Lyn Bartram. Investigating players' responses to wayfinding cues in 3D video games. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 1513–1518. 2014.

[24] Dinara Moura and Magy Seif El-Nasr. Design techniques for planning navigational systems in 3-D video games. *Computers in Entertainment (CIE)*, 12(2):1–25, 2015.

[25] Nintendo of America. The Making of The Legend of Zelda: Breath of the Wild Video – Open-Air Concept. https://www.youtube.com/watch?v=vLMGrmf4xaY.

[26] Rock, Paper, Shotgun. How the checklist conquered the open world, from Morrowind to Skyrim. https://www.rockpapershotgun.com/how-the-checklist-conquered-the-open-world-from-morrowind-to-skyrim, 2024. Accessed: 2024-08-24.

[27] Francine Rotzetter. Nonverbal Guidance Systems. Seamless Player-leading in Open-world Games. 2018.

[28] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC press, 2008.

[29] Cornelia J Schnaars. Taking a Breath of the Wild The Concept of Airness in Nintendo's take on Open World Games. *Game| World| Architectonics*, 115, 2021.

[30] Noor Shaker, Julian Togelius, and Mark J Nelson. Procedural content generation in games. 2016.

[31] Joel Vidqvist. Open-World Game Design: case Study: The Legend of Zelda: Breath of the Wild. 2019.

[32] Matt Walker. "Field Level Design in "The Legend of Zelda: Breath of the Wild" Hydral Earth Created", Translation of CEDEC 2017 talks by Nintendo. https://gist.github.com/idbrii/e39fe96279aa1670319bfa521d907399. Accessed: 2024-04-27.

[33] Jan M Wiener, Simon J Büchner, and Christoph Hölscher. Taxonomy of human wayfinding tasks: A knowledge-based approach. *Spatial Cognition & Computation*, 9(2):152–165, 2009.

Ludography

[34] Ubisoft Quebec. Assassin's Creed Odyssey. Ubisoft, 2016. Google Stadia, Windows, PlayStation 4, Xbox One.

[35] Bethesda Softworks. The Elder Scrolls III: Morrowind. Ubisoft, 2002. Windows, Xbox.

[36] FromSoftware. Elden Ring. Bandai Namco Entertainment, 2022. Windows, PlayStation 4, PlayStation 5, Xbox One, Xbox Series.

[37] Hello Games. No Man's Sky, 2016. Windows, PlayStation 4.

[38] Mojang Studios. Minecraft, 2011. Windows, macOS, Linux.

[39] Nintendo Entertainment. The Legend of Zelda: Breath of the Wild. Nintendo, 2002. Wii U, Nintendo Switch.

[40] Thekla. The Witness, 2016. Android, iOS, macOS, Windows, PlayStation 4, Xbox One.

[41] Ubisoft Montreal. Assassin's Creed Unity. Ubisoft, 2014. Windows, PlayStation 4, Xbox One.

[42] ZeniMax Online Studios. The Elder Scrolls Online. Bethesda Softworks, 2014. Windows, macOS, PlayStation 4, Xbox One, Google Stadia.

Appendix

1 PointOfInterest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Unity.Mathematics;
using UnityEditor;
using UnityEngine;
using Random = UnityEngine.Random;

[ExecuteInEditMode]
[RequireComponent(typeof(MeshFilter))]
[RequireComponent(typeof(Renderer))]
[RequireComponent(typeof(Collider))]
public class PointOfInterest : MonoBehaviour
{
    [Header("Number of Raycasts")]
    public int precision = 200;

    [Header("Draw raycasts into scene")]
    public bool drawRays;

    [Header("Font size for label")]
    public int fontSize = 12;

    [Header("Colour for label")]
    public Color fontColour = Color.white;

    [Header("Point of reference")]
    public GameObject referencePOV;

    [Header("Toggle label display")]
    public bool showLabel = true;

    private bool targetPointOnThisPOI;

    private float distanceToTargetPoint;
    private bool isVisible;
    private Mesh mesh;

    private MeshFilter meshFilter;
    private GameObject pointerObject;

    private Vector3 povPosition;
    private Renderer renderer;

    private List<Vector3> surfacePoints;
```

```
46    private float totalVisibleSurfaceArea;
47
48    private List<Triangle> triangles;
49    private List<float> triangleSurfaceAreas;
50    private Vector3[] vertices;
51    private float visibilityPercentage;
52    private List<float> visibleSurfaceAreas;
53
54    private List<Triangle> visibleTriangles;
55
56    public bool TargetPointOnPOI
57    {
58       set => targetPointOnThisPOI = value;
59    }
60
61    private void Update()
62    {
63       if (!Pointer.Instance.poiList.Contains(transform.gameObject))
64          Pointer.Instance.poiList.Add(transform.gameObject);
65
66       if (meshFilter == null) meshFilter = GetComponent<MeshFilter>();
67
68       if (renderer == null) renderer = GetComponent<Renderer>();
69
70       if (referencePOV != null && !targetPointOnThisPOI)
71       {
72          triangleSurfaceAreas = new List<float>();
73          visibleTriangles = new List<Triangle>();
74          visibleSurfaceAreas = new List<float>();
75          surfacePoints = new List<Vector3>();
76          triangles = new List<Triangle>();
77
78          Initialize();
79
80          CalculateRoughVisibility();
81
82          if (isVisible)
83          {
84             CalculateVisibleTriangles();
85             CollectUniformSampledSurfacePoints();
86             PerformRaycasts();
87          }
88       }
89    }
90
91    private void OnDrawGizmos()
92    {
93       var guiStyle = new GUIStyle();
94       guiStyle.fontSize = fontSize;
95       guiStyle.normal.textColor = fontColour;
96
97       if (showLabel && referencePOV != null)
98          Handles.Label(transform.position, DetermineVisibilityLabel(),
               guiStyle);
99    }
100
```

```
101     private void Initialize()
102     {
103         if (meshFilter == null) meshFilter = GetComponent<MeshFilter>();
104 
105         if (renderer == null) renderer = GetComponent<Renderer>();
106 
107         triangleSurfaceAreas = new List<float>();
108         visibleTriangles = new List<Triangle>();
109         visibleSurfaceAreas = new List<float>();
110         surfacePoints = new List<Vector3>();
111         triangles = new List<Triangle>();
112 
113         povPosition = referencePOV.transform.position;
114         visibilityPercentage = 0;
115         isVisible = false;
116 
117         var mesh = meshFilter.sharedMesh;
118         vertices = mesh.vertices;
119         var triangleVertices = mesh.triangles;
120 
121         for (var i = 0; i < triangleVertices.Length; i += 3)
122         {
123             // Save triangles of mesh filter
124             var v0 = transform.TransformPoint(vertices[triangleVertices[i
                    ]]);
125             var v1 = transform.TransformPoint(vertices[triangleVertices[i
                    + 1]]);
126             var v2 = transform.TransformPoint(vertices[triangleVertices[i
                    + 2]]);
127 
128             triangles.Add(new Triangle(v0, v1, v2));
129 
130             // Calculate surface area of triangle for more uniform point
                    sampling
131             var v01 = v1 - v0;
132             var v02 = v2 - v0;
133 
134             triangleSurfaceAreas.Add(Vector3.Magnitude(Vector3.Cross(v01,
                    v02)) / 2);
135         }
136     }
137 
138 
139     private void CalculateRoughVisibility()
140     {
141         if (renderer == null) renderer = GetComponent<Renderer>();
142 
143         var bounds = renderer.bounds;
144         var boundPoint0 = bounds.min;
145         var boundPoint1 = bounds.max;
146         var boundPoint2 = new Vector3(boundPoint0.x, boundPoint0.y,
                boundPoint1.z);
147         var boundPoint3 = new Vector3(boundPoint0.x, boundPoint1.y,
                boundPoint0.z);
148         var boundPoint4 = new Vector3(boundPoint1.x, boundPoint0.y,
                boundPoint0.z);
```

```
149         var boundPoint5 = new Vector3(boundPoint0.x, boundPoint1.y,
                boundPoint1.z);
150         var boundPoint6 = new Vector3(boundPoint1.x, boundPoint0.y,
                boundPoint1.z);
151         var boundPoint7 = new Vector3(boundPoint1.x, boundPoint1.y,
                boundPoint0.z);
152
153         Vector3[] boundPoints =
154            { boundPoint0, boundPoint1, boundPoint2, boundPoint3,
                   boundPoint4, boundPoint5, boundPoint6, boundPoint7 };
155
156         foreach (var boundPoint in boundPoints)
157         {
158            var hits = Physics.RaycastAll(boundPoint, povPosition -
                   boundPoint,
159               Vector3.Magnitude(povPosition - boundPoint));
160
161            Array.Sort(hits, (hit1, hit2) => hit1.distance.CompareTo(hit2.
                   distance));
162
163            if (hits.Length > 0)
164            {
165               if (hits[0].transform == referencePOV.transform)
166               {
167                  isVisible = true;
168                  break;
169               }
170            }
171            else
172            {
173               isVisible = true;
174               break;
175            }
176         }
177      }
178
179      private void CalculateVisibleTriangles()
180      {
181         var povPosition = referencePOV.transform.position;
182         Triangle triangle;
183
184         for (var i = 0; i < triangles.Count; i++)
185         {
186            triangle = triangles[i];
187
188            var v0 = triangle.v0;
189            var v1 = triangle.v1;
190            var v2 = triangle.v2;
191
192            // Perform backface culling
193            var normal = Vector3.Cross(v1 - v0, v2 - v0).normalized;
194            if (Vector3.Dot(normal, (v0 - povPosition).normalized) < 0)
195            {
196               visibleTriangles.Add(triangle);
197               visibleSurfaceAreas.Add(triangleSurfaceAreas[i]);
198            }
```

```
199         }
200     }
201
202     private Triangle PickRandomVisibleTriangle()
203     {
204         totalVisibleSurfaceArea = visibleSurfaceAreas.Sum();
205         var numberOfTriangles = visibleTriangles.Count;
206
207         float[] probabilityList = new float[numberOfTriangles];
208
209         for (int i = 0; i < probabilityList.Length; i++)
210         {
211             probabilityList[i] = visibleSurfaceAreas[i] /
                    totalVisibleSurfaceArea;
212         }
213
214         var rng = Random.Range(0, 1f);
215
216         float minInterval = 0;
217         float maxInterval = 0;
218
219         for (int i = 0; i < probabilityList.Length; i++)
220         {
221             maxInterval += probabilityList[i];
222
223             if (rng >= minInterval && rng <= maxInterval)
224             {
225                 return visibleTriangles[i];
226             }
227
228             minInterval = maxInterval;
229         }
230
231         return visibleTriangles[^1];
232     }
233
234     private void CollectUniformSampledSurfacePoints()
235     {
236         for (var i = 0; i < precision; i++)
237         {
238             var triangle = PickRandomVisibleTriangle();
239
240             var v0 = triangle.v0;
241             var v1 = triangle.v1;
242             var v2 = triangle.v2;
243
244             var v01 = v1 - v0;
245             var v02 = v2 - v0;
246
247             var r1 = Random.Range(0f, 1f);
248             var r2 = Random.Range(0f, 1f);
249
250             if (r1 + r2 > 1)
251             {
252                 r1 = 1 - r1;
253                 r2 = 1 - r2;
```

```
254             }
255
256             var randomSurfacePoint = v0 + r1 * v01 + r2 * v02;
257             surfacePoints.Add(randomSurfacePoint);
258         }
259     }
260
261     private void PerformRaycasts()
262     {
263         float hitCount = 0;
264
265         distanceToTargetPoint = math.INFINITY;
266
267         foreach (var surfacePoint in surfacePoints)
268         {
269             RaycastHit[] hits;
270             hits = Physics.RaycastAll(surfacePoint, povPosition -
                    surfacePoint,
271                 Vector3.Magnitude(povPosition - surfacePoint));
272
273             if (drawRays) Debug.DrawRay(surfacePoint, povPosition -
                    surfacePoint, Color.green);
274
275             Array.Sort(hits, (hit1, hit2) => hit1.distance.CompareTo(hit2.
                    distance));
276
277             if (hits.Length > 0)
278             {
279                 if (hits[0].transform != transform && hits[0].transform !=
                        referencePOV.transform) hitCount += 1;
280                 else if (hits[0].distance < distanceToTargetPoint)
                        distanceToTargetPoint = hits[0].distance;
281             }
282         }
283
284         visibilityPercentage = (precision - hitCount) / precision;
285     }
286
287     private string DetermineVisibilityLabel()
288     {
289         var perceivedObjectSize = totalVisibleSurfaceArea *
                visibilityPercentage / distanceToTargetPoint;
290
291         if (perceivedObjectSize < 0.02f) return "Not Visible";
292         if (perceivedObjectSize < 0.1f) return "Slightly Visible";
293         if (perceivedObjectSize < 0.2f) return "Moderately Visible";
294         if (perceivedObjectSize < 0.3f) return "Very Visible";
295
296         return "Fully Visible";
297     }
298 }
299
300 internal struct Triangle
301 {
302     public Vector3 v0;
303     public Vector3 v1;
```

```
304     public Vector3 v2;
305 
306     public Triangle(Vector3 v0, Vector3 v1, Vector3 v2)
307     {
308         this.v0 = v0;
309         this.v1 = v1;
310         this.v2 = v2;
311     }
312 }
```

2 Pointer.cs

```
1  using System;
2  using System.Collections.Generic;
3  using Unity.Mathematics;
4  using Unity.VisualScripting;
5  using UnityEditor;
6  using UnityEngine;
7  
8  [ExecuteInEditMode]
9  public class Pointer : MonoBehaviour
10 {
11     private static Pointer instance;
12 
13     public static Pointer Instance
14     {
15         get
16         {
17             if (instance == null)
18             {
19                 instance = FindObjectOfType<Pointer>();
20 
21                 if (instance == null)
22                 {
23                     GameObject singletonObject = new GameObject("
                           SceneViewPointer");
24                     instance = singletonObject.AddComponent<Pointer>();
25                 }
26             }
27 
28             return instance;
29         }
30     }
31 
32     [Header("Toggle target point placement with 'P'")]
33     public bool enableTargetPoint = true;
34 
35     [Header("Target point material colour")]
36     public Color targetPointColor = Color.blue;
37 
38     [Header("Target point size")]
39     public Vector3 targetPointScale = new(0.35f, 0.1f, 0.35f);
40 
41     private List<GameObject> pointOfInterests;
```

```
42
43      private GameObject currentObjectHit;
44
45      private GameObject oldTargetPoint;
46      private bool surfaceHit;
47      private GameObject targetPoint;
48
49      public List<GameObject> poiList => pointOfInterests;
50
51      private void Update()
52      {
53          if (pointOfInterests == null) pointOfInterests = new List<
                GameObject>();
54
55          pointOfInterests.RemoveAll(item => item == null);
56      }
57
58      private void Awake()
59      {
60          if (Instance != null && Instance != this)
61          {
62              DestroyImmediate(this);
63          }
64          else
65          {
66              instance = this;
67          }
68
69      }
70
71      private void OnEnable()
72      {
73          if (Instance != null && Instance != this)
74          {
75              DestroyImmediate(this);
76          }
77          else
78          {
79              instance = this;
80          }
81
82          EditorApplication.hierarchyChanged += OnHierarchyChanged;
83          SceneView.duringSceneGui += OnSceneGUI;
84          pointOfInterests = new List<GameObject>();
85          surfaceHit = false;
86          if (targetPoint != null)
87          {
88              DestroyImmediate(targetPoint);
89          }
90          else
91          {
92              targetPoint = GameObject.Find("Target Point");
93          }
94
95          if (oldTargetPoint != null)
96          {
```

```
97          DestroyImmediate(oldTargetPoint);
98       }
99    }
100
101   private void OnSceneGUI(SceneView sceneView)
102   {
103      Event e = Event.current;
104      if (e.type == EventType.KeyDown && e.keyCode == KeyCode.P)
105      {
106         enableTargetPoint = !enableTargetPoint;
107      }
108
109      if (enableTargetPoint)
110      {
111         var ray = HandleUtility.GUIPointToWorldRay(Event.current.
              mousePosition);
112         var hits = Physics.RaycastAll(ray, math.INFINITY);
113
114         if (hits.Length > 0)
115         {
116            foreach (var hit in hits)
117            {
118               if (targetPoint == null)
119               {
120                  targetPoint = GameObject.CreatePrimitive(
                       PrimitiveType.Sphere);
121                  targetPoint.gameObject.name = "Target Point";
122
123                  var targetPointRenderer = targetPoint.GetComponent<
                       Renderer>();
124                  var tempMaterial = new Material(targetPointRenderer.
                       sharedMaterial);
125                  tempMaterial.color = targetPointColor;
126                  targetPointRenderer.sharedMaterial = tempMaterial;
127
128                  targetPoint.transform.localScale = targetPointScale;
129                  currentObjectHit = hit.transform.gameObject;
130                  targetPoint.transform.position = hit.point;
131                  targetPoint.transform.up = hit.normal;
132               }
133               else
134               {
135                  if (hit.transform == targetPoint.transform) continue;
136                  currentObjectHit = hit.transform.gameObject;
137                  targetPoint.transform.position = hit.point;
138                  targetPoint.transform.up = hit.normal;
139                  break;
140               }
141
142               surfaceHit = true;
143            }
144         }
145         else
146         {
147            surfaceHit = false;
148            DestroyImmediate(targetPoint);
```

```
149             }
150
151             if (surfaceHit && Event.current.type == EventType.MouseDown &&
                    Event.current.button == 0)
152             {
153                 if (oldTargetPoint != null) DestroyImmediate(oldTargetPoint
                        );
154
155                 foreach (var poi in pointOfInterests)
156                 {
157                     var poiScript = poi.GetComponent<PointOfInterest>();
158                     poiScript.referencePOV = targetPoint;
159                     if (currentObjectHit == poi) poiScript.TargetPointOnPOI
                            = true;
160                     else poiScript.TargetPointOnPOI = false;
161                 }
162
163                 oldTargetPoint = targetPoint;
164                 targetPoint = null;
165                 surfaceHit = false;
166             }
167         }
168         else
169         {
170             if (targetPoint != null) DestroyImmediate(targetPoint);
171         }
172     }
173
174     private void OnHierarchyChanged()
175     {
176         foreach (var poi in pointOfInterests)
177             if (!GameObject.Find(poi.name))
178                 pointOfInterests.Remove(poi);
179     }
180 }
```