A COMPARISON OF PROCEDURAL-GENERATED AND HUMAN-DESIGNED TWO-DIMENSIONAL PLATFORMER GAME LEVELS

BALIM ALPAY

Thesis for the Master's Program in Design Studies

Graduate School
Izmir University of Economics
Izmir
2024

A COMPARISON OF PROCEDURAL-GENERATED AND HUMAN-DESIGNED TWO-DIMENSIONAL PLATFORMER GAME LEVELS

BALIM ALPAY

THESIS ADVISOR: ASST. PROF. DR. LALE BAŞARIR

A Master's Thesis
Submitted to
the Graduate School of Izmir University of Economics
the Department of Design Studies

Izmir
2024

ETHICAL DECLARATION

I hereby declare that I am the sole author of this thesis and that I have conducted my work in accordance with academic rules and ethical behaviour at every stage from the planning of the thesis to its defence. I confirm that I have cited all ideas, information and findings that are not specific to my study, as required by the code of ethical behaviour, and that all statements not cited are my own.

Name, Surname: Balım Alpay

Date: 13.02.2024

ABSTRACT

A COMPARISON OF PROCEDURAL-GENERATED AND HUMAN-DESIGNED TWO-DIMENSIONAL PLATFORMER GAME LEVELS

Alpay, Balım

Master's Program in Design Studies

Advisor: Asst. Prof. Dr. Lale BAŞARIR

January, 2024

This thesis analyzes the design and impact of procedural generation in two-dimensional platformer games, while also drawing comparisons to levels designed by humans. The study emphasizes the importance of comprehending the disparities between human-designed and procedurally generated designs, specifically in terms of their impact on the level designer's experience in two-dimensional platform games. It discusses the concept of level design in game design, analyzes the primary design elements utilized for level design, and provides an explanation of the two-dimensional platformer genre level design process with design patterns. The study employs a methodology that examines two unique two-dimensional platform games created particularly for this thesis. All variables, except for level design, are held constant to ensure an accurate comparison. One game's levels are human designed, while the levels of the other game are generated procedurally. The findings indicate that human-designed levels offer distinct instances of gameplay, narrative depth, and empathy. On the other hand, procedural generated levels stand out in terms of providing exceptional replay value, challenges, and opportunities for exploration. The thesis examines the merits and drawbacks of each approach, with a particular focus on achieving balance, magnitude, and harmony in level design. Ultimately, the research supports the idea of combining human creativity with the effectiveness of procedural generated level design, endorsing a hybrid approach. This combination has the potential to improve the player's experience and broaden the design possibilities in two-dimensional platformer games.

Keywords: Level Design, Procedural Generation, Human-Designed, Two-Dimensional Platformer Games, Design Pattern in Games, Graph-Based Generation.

ÖZET

PROSEDÜREL ÜRETİLEN VE İNSAN TASARIMLI İKİ BOYUTLU PLATFORM OYUN BÖLÜMLERİNİN KARŞILAŞTIRILMASI

Alpay, Balım

Tasarım Çalışmaları Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Lale BAŞARIR

Ocak, 2024

Bu tez, iki boyutlu platform oyunlarında prosedürel üretimin tasarımını ve etkisini analiz ederken, aynı zamanda insanlar tarafından tasarlanan seviyelerle karşılaştırmalar yapmaktadır. Çalışma, iki boyutlu platform oyunlarında seviye tasarımcısının deneyimi üzerindeki etkileri açısından insan tasarımı ve prosedürel üretilmiş tasarımlar arasındaki farkları anlamanın önemini vurgulamaktadır. Oyun tasarımında seviye tasarımı kavramını tartışır, seviye tasarımı için kullanılan temel tasarım unsurlarını analiz eder ve iki boyutlu platform türü seviye tasarım sürecini tasarım kalıpları ile birlikte açıklar. Çalışma, bu tez için özel olarak oluşturulan iki benzersiz iki boyutlu platform oyununu inceleyen bir metodoloji kullanmaktadır. Doğru bir karşılaştırma sağlamak için seviye tasarımı dışındaki tüm değişkenler sabit tutulur. Bir oyunun seviyeleri insan tarafından tasarlanırken, diğer oyunun seviyeleri prosedürel olarak üretilir. Bulgular, insan tasarımı seviyelerin benzersiz oyun deneyimleri, anlatı derinliği ve empati sunma açısından öne çıktığını göstermektedir. Öte yandan, prosedürel üretilen seviyeler, olağanüstü tekrar oynanabilirlik değeri, zorluklar ve keşfetme fırsatları sunma açısından dikkat çekmektedir. Tez, her yaklaşımın artılarını ve eksilerini inceleyerek, seviye tasarımında denge, büyüklük ve uyum elde etmeye odaklanır. Sonuç olarak, araştırma insan yaratıcılığını prosedürel üretilen seviye tasarımının etkinliği ile birleştiren hibrit bir yaklaşımı desteklemektedir. Bu kombinasyon, oyuncunun deneyimini geliştirme ve iki boyutlu platform oyunlarında tasarım olanaklarını genişletme potansiyeline sahiptir.

Anahtar Kelimeler: Bölüm Tasarımı, Prosedürel Tasarım, İnsan Tasarımı, İki Boyutlu Platform Oyunları, Oyunlarda Tasarım Desenleri, Graf Tabanlı Üretim.

Dedicated to spring of my life, my loving grandfathers.

ACKNOWLEDGEMENTS

First and foremost, I extend my heartfelt thanks to my supervisor, Asst. Prof. Dr. Lale Başarır. Her unwavering belief in my abilities and her continuous guidance have been the cornerstone of my academic growth. She has been more than a mentor to me; she has been a guiding light, illuminating the path of my research with her profound knowledge and infectious energy. Her support has not only enriched my academic experience but has also instilled in me a sense of confidence and enthusiasm.

Additionally, I wish to express my sincere appreciation to my family, which consists of my lovely sister Petek Alpay, my mother Dilek Alpay, and my father Mücalp Alpay. Their unending support, motivation, and confidence in my capabilities have served as the foundations of my fortitude. They have consistently provided me with solace and insightful words of this journey. I am eternally indebted to my sister for serving as my light and support.

Furthermore, I would like to extend my sincere appreciation to Cevher Eryürek and Meriç Eryürek for their remarkable patience and adaptability throughout the thesis phase. Their assistance transcended professional boundaries and had a substantial influence on my scholarly pursuits.

I would like to extend my sincere gratitude to Birkan Kader for his unwavering support and presence throughout this ordeal. He was with me every moment, both technically and spiritually.

Lastly, I would like to thank to my cat Arwen for sitting with me until the morning. I couldn't have gotten through this process alone.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1. Need for Study

The creation of spaces and levels for video games is a complex process that requires careful consideration of the player experience, gameplay mechanics, and narrative components. Every video game's potential is proportional to the quality of its design and concept. The success of a game depends on the game designer, who is responsible for the initial stages of the project. Game designers are tasked with crafting not only the visuals, narrative, and other fantastical aspects of the game but also the player's actual experience within the game. In many cases, the most common cause of a game's commercial failure is its poor design (Karlsson, Brusk and Engström, 2023).

Depending on the size and scale of the project, more than one game designer may work on the game, and the titles of these designers also vary (level designer, quest designer, narrative designer, gameplay designer etc.) (Schell, 2008). Level designers are responsible for designing the spatial layouts and environments that players navigate and encounter when playing video games. Although the number, complexity, and style of levels may differ across projects, the primary task of level designers is to build and arrange structures, while also designing different types of obstacles (Co, 2006). The level design of a game is crucial for its playability.

Game companies work with a limited number of designers to find qualified personnel and keep the budget of the project under control. Limited employment of level designers compared to the game's scale causes monotonous and repetitive game levels. This leads players to discontinue to the game or even abandon the game completely.

In this case, it is possible to facilitate and support the designer's job by using procedural content generation. By generating content automatically and integrating artificial intelligence (AI) into the process, more dynamic and responsive surroundings (which are called levels) can be produced than human-designed levels, which will improve the gameplay experience. Procedural Content Generation in video games refers to the utilization of artificial intelligence and algorithms to autonomously generate various aspects of game content, such as levels, settings, and enemies. This technique differs significantly from conventional, human-designed game development by including aspects of unpredictability and diversity (Adams, 2002). Instead of engaging in pre-determined, unchanging stages, players are presented with unique and dynamic gaming environments and obstacles that vary with each playthrough. This feature not only enhances the novelty and potential to be played again of games but also enables extensive and dynamic gaming experiences. Levels can even be generated in real-time based on algorithms, resulting in an ever-changing gaming environment.

1.2. Research Questions

Based on the data and observations collected during the study, the research questions have been developed as follows to guide the thesis:

RQ 1. What role might artificial intelligence play in level design in video games? How will AI-generated levels differ from human-designed levels?

RQ 2. How can procedural generation help the level design process of video games? What would be the difference between procedurally generated levels and human-designed levels?

RQ Final. What are the potential benefits and challenges of procedurally generated video game levels, and how can this approach contribute to creating varied and engaging gameplay experiences? What are the benefits of procedural generation from a level design perspective?

1.3. Aims and Scope

The purpose of this thesis is to examine the impact of procedurally generated levels on two-dimensional side-scrolling scroller platformer video game design. The study started with the aim of focusing on the design benefits and differences, between human-designed levels and procedurally designed levels. The research focuses on two-dimensional platformer games.

This thesis aims to,

- Explain what game level design is and production phases.
- Review level generation approaches.
- Analyze the differences between human-designed levels and procedural-designed levels.
- Develop two new games, one with procedural-generated levels and other human-designed ones, discuss their strengths, weaknesses, and benefits.

The study examines the principles of level design, its mechanics, and draws parallels between level design and design elements. Then, it explains what procedural generation is and how it works in design and information on which tools work and how they work. How designers approach these two design methods (human-designed vs procedural-generated), Accordingly, the advantages and disadvantages of the two methods are revealed.

1.4. Why Were Two-Dimensional Platformer Games Chosen for This Research?

The choice to focus on two-dimensional platform games in this thesis is based on the principles of simplicity, accessibility, and focused design. Creating games in two dimensions usually demands fewer resources, giving a notable advantage when dealing with limited time and resources in academic research. These games enable a focused analysis of game mechanics, design, and designer experience, without the added complications of detailed modeling and texturing required by 3D games. Moreover, two-dimensional platform games have a significant historical background that has played a crucial role in the development of video games. This allows for a more comprehensive examination of game design and culture within a broader context. These games possess the necessary characteristics for studying the relationship between procedural generation and design focused on human experience, such as the ability to be played multiple times and a wide range of design variations.

The study's research questions and objectives can be more efficiently tackled by utilizing 2D platforms. These platforms effectively showcase design principles and game mechanics, while also ensuring that the research process remains manageable. As a result, the selection of two-dimensional games aligns precisely with the scope and objectives of this research and should be regarded as a methodological requirement.

1.5. Methodology

Upon the questions of the thesis, developing two new 2D side scroller platformer games appeared as appropriate for the research goals. Both games have the same gameplay and have the same artistic style. While one of them has human-designed levels, the other levels are generated procedurally according to the designer's choices. The study collected its findings by examining encountered by the level designer during the game creation process for these two games.

The thesis is structured as follows; Chapter 2 is serves as a literature review of game-level design, design components which are important for procedural generation. Describes level design patterns to compare the differences between procedural generated levels and human-designed levels. Chapter 3 explains what procedural generation is and evaluate current approaches, and examine the tool used in the development phase of one of the games which procedurally generated levels, and the differences from others. Chapter 4 presents the two games which developed for the thesis, explain game mechanics and components, analyze the designer's approaches, explain benefits of both. Chapter 5 is conclusion and about future works.

CHAPTER 2: GAME DESIGN

2.1. Design in Video Games

The process of deciding what a game should be, also known as game design, is critical. Game design includes making decisions on a wide range of topics, such as rules, aesthetics, time, risk, rewards, and more.

Every created thing has a designer. A designer develops strategies for turning ideas into tangible products. A game designer is someone who creates every aspect of the player experience, not just the narrative. Creating the game's rules, visual and audio elements, rhythm, and general atmosphere are among the responsibilities. It is emphasized that engineers, animators, composers, writers, producers, and other game developers are included in the larger group of game developers, which includes game designers. When it comes to games, the designer is usually the one who comes up with the original concepts, puts them on paper to show to others (either as a rough demo or design document), and oversees the process of turning a design into a playable video game (Byrne, 2005).

As the name implies, game design is a continuous process that needs to be engaged in from the beginning to the end of the game's conception (Athavale and Dalvi, 2018). The differentiation between game developers and designers serves to emphasize that different team members who participate in decision-making during the development process can have a role in game design.

This thesis discusses level design and level designer, both of which fall under the category of game design.

2.2. Primary Elements in Video Game Design

The conceptual elements of point, line, plane, and volume are simply expressions of the human mind's imagination. We sense the existence of these basic elements even though we cannot see them clearly. We can perceive the point where two lines cross, visually detect a line that outlines the boundary of a plane, physically sense a plane that covers a three-dimensional space, and mentally grasp the volume of an object filling the space. When viewed on paper or in three-dimensional space, we see these elements in shapes, sizes, colors, and textures. While experiencing these forms around us, we should see the presence of the basic elements of point, line, plane, and volume in their structures (Ching, 2002). In this section, we will analyze these elements from a design perspective and see how they guide us in shaping digital environments. Let us now analyze the Breakout game (1976 – Atari, Inc.), Figure 1, created by Steve Wozniak, by considering four elements: point, line, plane, volume.

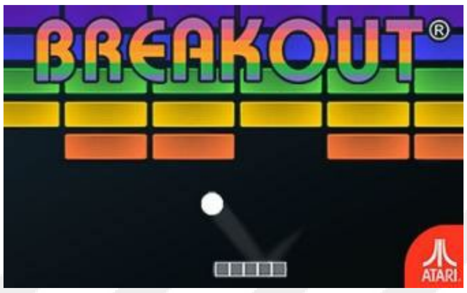


Figure 1. Breakout Game – 1976

2.2.1. Point

A point represents a specific position. Conceptually, it has a structure without any dimensions such as length, width, or depth, resulting in a fixed, centralized, and directionless nature. In theory, a point lacks both shape and form, yet when positioned in a visual field, it starts showing its existence (Ching, 2002).

The point serves as a fundamental unit in video games, frequently employed to establish the coordinates of objects or characters. Guide dots serve to assist players in locating their objective or mission within a game, but in a "Breakout" game, the ball is commonly depicted as a dot. The ball's position corresponds to a distinct location on the playing surface (Figure 2).

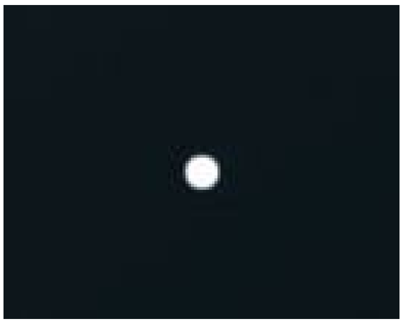


Figure 2. Breakout Game – The Ball represent the point.

2.2.2. Line

A point transforms into a line when it is expanded. A line is a one-dimensional geometric figure that possesses length but lacks width or depth. Although a point remains set, a line can visually represent orientation, movement, and expansion by tracing the path of a moving point (Ching, 2002).

Lines in video games serve as visual representations of highways, walls, corridors, and other structural components. In an action game, hallways and tunnels can be composed of various configurations of lines. Lines serve the purpose of defining the direction of paths or recognizing challenges. It offers instructions for players to navigate and engage with a game world. In the game "Breakout," the stick or platform serves as the primary character under the player's control. The stick is shown as a horizontal line and serves the purpose of supporting the ball or demolishing bricks (Figure 3).

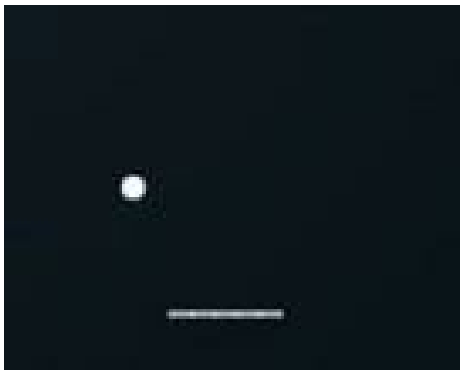


Figure 3. Breakout Game – The Platform represent the line.

2.2.3. Plane

A line transforms into a plane when it is extended in a direction that is different from its fundamental direction. Conceptually, a plane is a two-dimensional object that possesses length and width, but lacking depth. The primary distinguishing feature of a plane is its geometric form. The shape is determined by the contour of the line that defines the boundaries of the plane. Zooming in perspective can cause a change in our perception of shape. Therefore, the real form of a plane can only be observed when viewed from the front (Ching, 2002).

Planes are the physical boundaries that define the game world's surfaces. These surfaces can possess distinct attributes, such as being floors, walls, ceilings, or areas of water. In the game "Breakout," the playing field symbolizes the concept of a two-dimensional plane. The playing field (Figure 4), where both the ball and stick are in motion, can be seen as a plane where the player engages. The term "plane" describes a two-dimensional space in which everything moves inside the game place.

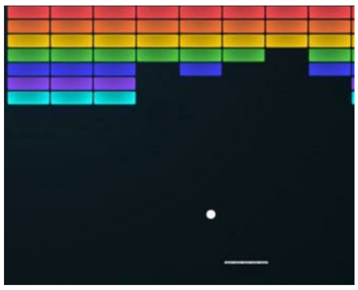


Figure 4. Breakout Game – The Playground represent the plane.

2.2.4. Volume

A plane becomes a volume when extended in a direction other than the internal direction. Conceptually, a volume has three dimensions: length, width, and depth (Ching, 2002).

Volume serves a crucial role in the three-dimensional realm of video games. Volume refers to the spatial dimensions of things and spaces in three dimensions. A "Breakout" game is often a 2D game, therefore, the sense of volume is not as apparent. In 2000, Supersonic Software attempted to incorporate a three-dimensional aspect into the game with Breakout 3D. However, this attempt did not achieve the same level of success as the original Breakout game (Figure 5)

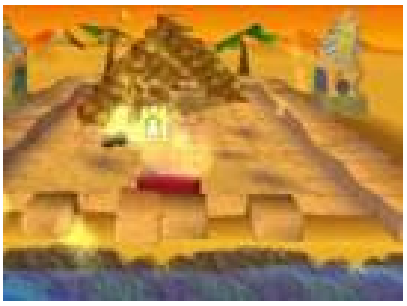


Figure 5. Breakout 3D Game – Volumetric Playground

Even in the design of a basic game like "Breakout," these four basic elements are present. The core mechanics of the game are established by these concepts and used to mold the player's experience. Players participate in the game by guiding the point (ball) along the line (platform) within the plane (playground).

2.3. *The 2D Platformer Genre*

The “platformer” genre gets its name from the jumping and landing areas that these games use in their levels (Co, 2006). Super Mario Bros. from Nintendo (the first side scroller two-dimensional platform game) and Sonic the Hedgehog from Sega are two examples of the two-dimensional "side scrolling" genre that dominated the video game market years ago and gave rise to these games (Suominen, 2012).

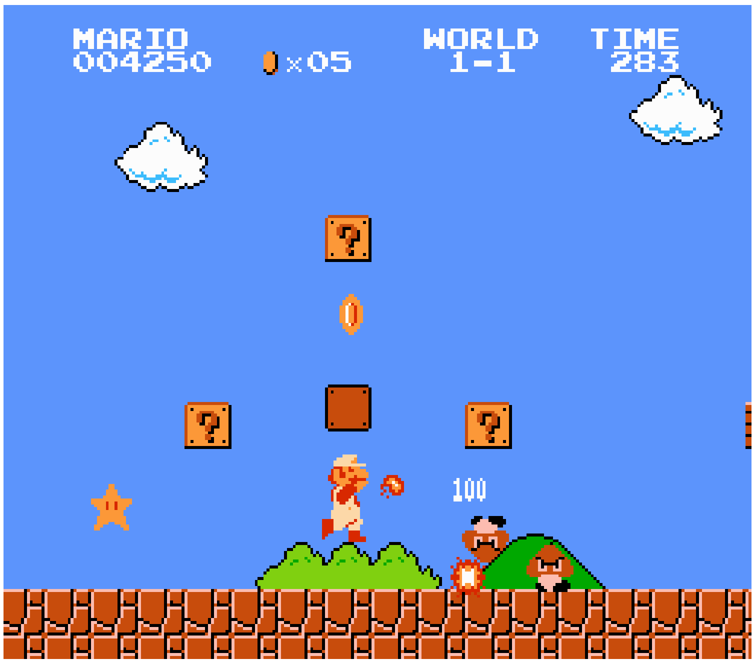


Figure 6. Super Mario Bros. was the first game in the series and the first 2D side-scrolling platform game with Mario, released in 1985 for the Nintendo Entertainment System.

As the player character progresses through the levels, the side scrolling genre "scrolls" the screen to keep up with character. In most platformer games, movement still serves as the major gameplay aspect, and reflexes and navigational skills are more often tested than deductive reasoning in most levels. To go from one end of the level to the other, player characters can move between them using a variety of abilities, such as sprinting, jumping, climbing, and sliding (Minkkinen, 2016). It's simple to comprehend and accept the game's objectives, which are to move a character from one place to another.

2.4. Level Design

Level design is defined as the implementation of the group's concepts into a playable format. In Smith, Cha and Whiteheads own words:

> *"Levels are the space where a player explores the rules and mechanics of a game; as such, good level design is critical to the game design process." (2008, p. 75).*

A level designer is the point of convergence for programming, cinematography, audio, art, and design. The fundamental structure of every game is created by game designers and put into place and maintained by level designers. Besides creating environments and captivating graphics, level designers also keep an eye on the game's performance, ensure that any technological issues are fixed before the product is released, and repair any bugs that may arise. The user experiences a game through its levels, which is why level design plays such a crucial part in today's production teams.

2.4.1. Defining Levels

In many games, the terms "map," "mission," and "stage" are synonymous with "level." The term "level" in games came most likely from the early home game systems and arcade machines, when the gameplay was separated into stages or levels, which were progressively harder than one another (Byrne, 2005).

A level may concentrate on a single objective or include almost all the game's systems and mechanics. Every game has an environment, and it is the ‘level designers' responsibility to create that setting putting the ground in playground.

2.4.2. Good Design for Levels

Good design for levels is vital for creating enjoyable games; as “containers for gameplay” (Byrne, 2005), levels serve as "containers for gameplay," giving the player an interactive area to explore within the confines of the game world. Considering this, creating levels is a difficult task that necessitates knowing how to combine each of the game's components. Additionally, the abilities needed vary depending on the genre: a 2D platformer level requires quite different skills than a role-playing game level. Since the scope of this thesis is 2D platformer games, it continues in this field.

2.4.3. Level Design Process

Game creation is a collaborative effort that requires the participation of a diverse variety of specialists, including artists, designers, engineers, writers, sound designers, and musicians (Whitson, 2020). As an outcome, many roles in game development are classified as flexible. According to McDaniel and Daer (2016), depending on experience, ability, game type, and current stage of production, the function of a game designer may include activities such as world design, system design, content design, game writing, level design, or user interface design. The level design process presents a significant management difficulty, as it must accommodate both creativity and efficiency. As a result, the traits and challenges discovered for game production in general are mirrored, like level design.

There are four steps to production of level design. These are knowing the game, knowing the audience, knowing the tools, dimensions, and scale calculation (YouTube, 2021).

2.4.3.1. Knowing the Game

To create high-quality content for any type of game, it is essential to have a deep understanding of the game's mechanics and complexity. It is essential to have knowledge of the responses to these types of questions: what the game is about, what tasks does the player take in the game, how does the player traverse the environment, is there vaulting and climbing, can the player crouch or slide under narrow gaps, what are the types of obstacles that player can face in the game, etc.

2.4.3.2. Knowing the Audience

In order to make informed design choices, it is important to possess an in-depth understanding of the audience you are targeting, or, in simpler terms, "Who are the players that will be engaging with your game?"

2.4.3.3. Knowing the Tools

One of the crucial aspects of level design is the creation of a blockout. Once sufficient information about the game and players is acquired, this stage becomes of paramount importance.

A blockout (Figure 7), also known as a blockmesh or graybox, is an early level constructed using primary elements devoid of intricate details or refined art assets. The objective is to create a prototype, conduct testing, and make necessary modifications to the fundamental structures of the level. For example, in this figure, orange boxes represent branches of the tree, which will break when jump on it. Black boxes represent solid, and red boxes represent enemies and environmental hazards. First designing the level as blockout and then making visual dressing is part of the game process.

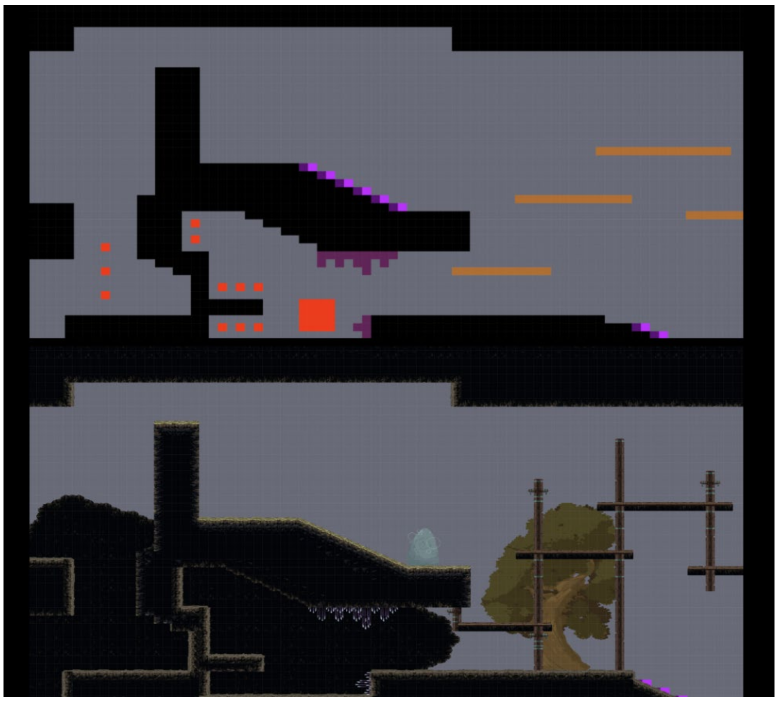


Figure 7. Blockout – Restless Lands Level, by Balım Alpay.

2.4.3.4. Dimension and Scale Calculations

Once the designer has a clear understanding of the fundamental mechanics, they should establish standardized measurements for the environments. What does the maximum height jump that a player can achieve? Similarly, this applies to gaps in the level that can be jumped over. The player is more likely to make a mistake if the gap is precisely the same length as their jump distance, and they may be annoyed when they die because they don't feel responsible if the gap is only slightly larger. On the other hand, scaling is also important for visual. The details will be examined in section *2.4.4. Basic Design Components in Level Design.*

The four steps of 'Level Design Process' will assist level designers in preparing more efficiently for any project they are designing levels for. However, since each project is unique, it is important to be aware of what is most applicable to the specific designer's work and adapt the production process accordingly.

2.4.4. Basic Design Components in Level Design

The more realistic the level designs of the game are, the more the player will feel involved in the game. Despite some trends, a game's level of realism is not directly proportional to the amount of high-resolution – realistic textures on the surfaces the levels cover. The use of high-quality textures and lighting can enliven the space; However, realism can only be achieved through the correct and appropriate application of certain design principles. These principles include concepts such as balance, scale, proportion, emphasis, rhythm, harmony, and unity. By consciously applying these principles, level designers ensure that levels feature references to real architecture, so players can experience spatial familiarity even in a virtual environment. (Çatak, 2003) Design components are used to ensure that the levels of a game are effective and satisfying and greatly affect the quality of the gaming experience.

2.4.4.1. Balance

The concept of balance in partition design can be represented by different situations, but the most prominent one is shown by the equitable allocation of the design in both the vertical and horizontal axes. At the same time, the division should be balanced, evenly distributed with each item, with the balance of enemies, sufficient resources, and obstacles to overcome.

2.4.4.2. Scale

Scale refers to the congruity between the scale of the section and the spatial relationships perceived by the player. It is important to both physical objects and two-dimensional textures. An inaccurately proportioned texture or object detracts from the player's perception of space and undermines its credibility. The player experiences discomfort and becomes disinterested. For example, when observing a building's door from the exterior, it is necessary to accurately adjust its proportions to match the size of the player.

2.4.4.3. Proportion

When comparing the sizes of objects, it is important to maintain accurate proportions and avoid distortion. When creating a racetrack for a racing game, it is important to ensure that the dimension of the vehicles is in proportion to the width of the track. Proper calculation of curves and assignment of lanes based on vehicle dimensions are crucial for maintaining player engagement and ensuring a fair competitive environment.

2.4.4.4. Emphasis

Emphasis or focal point is important for the object or place that will first attract the player's attention. Maintaining spatial composition and drawing the player's attention to the right place is one of the key design components of level design.

2.4.4.5. Rhythm

Moving the player's focus from one area to another within the episode is the element that creates a sense of rhythm through the recurring presence of similar objects, colors, light, or style. Changing the player's movements for the purpose of research is a fundamental responsibility of the level designer, but it is also one of the most difficult challenges. The sense of rhythm makes the player feel safe and contains predictable areas.

2.4.4.6. Harmony

Harmony refers to the harmony of visual elements such as theme, style, color palette, pattern and texture used in the department design. Harmony is important for aesthetic integrity. For example, the harmonious atmosphere created in a horror game is used to give the player a frightening experience.

2.4.4.7. Unity

It refers to the combination of the game's story, game spaces and the design of the spaces to create an experience. Objects need to give clues to the player about where they are and what is going on. The game world of the designed levels, the character's story and the missions must harmoniously combine with the theme.

2.5. 2D Platformer Games Level Design Patterns

2D Platformer Games Level Design Patterns describes the methodical design components and guidelines that are applied when developing 2D platformer game levels. Character movements, jump actions, and a variety of platform kinds (such as static, dynamic, and fading) are all included in these patterns. Different platform kinds and behaviors, like moving trajectories and unique features, define platformers. Features are used to characterize the movements of the player's character, and a directed graph linking platforms defines how to navigate the game (Aramini, Lanzi and Loiacon, 2018)

Silva, Khalifa and Togelius examined more than 30 different 2D games in 2019 and determined the six common patterns of them. In their work, they presented six patterns: Guidance, foreshadowing, safe zone, layering, branching, and pace breaking (Silva, Khalifa and Togelius, 2019). These six patterns were analyzed through their own research.

2.5.1. Guidance

During gameplay, players might get disoriented and lose sight of the intended route, particularly when the game design emphasizes exploration. The most straightforward method to direct the player is by utilizing the level's shape. In this design pattern, designers use solid tiles to focus the player's attention on the intended direction

(Figure 8).

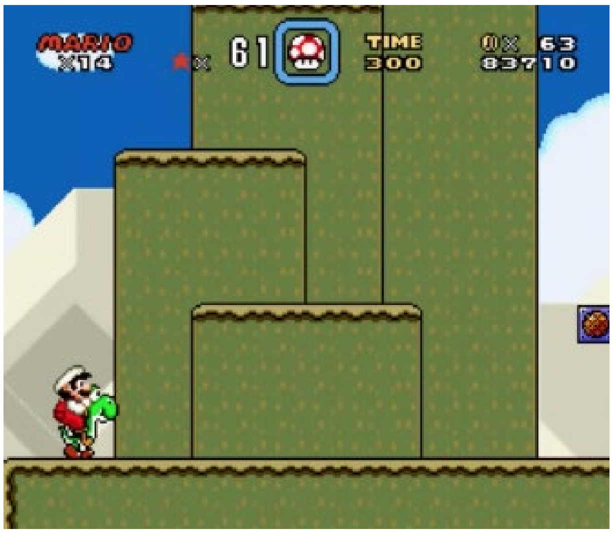


Figure 8. Stacked platforms guide the players upwards in Super Mario World

2.5.2. Foreshadowing

Foreshadowing is a literary pattern in which the level designer provides hidden clues or guesses about future events in the plot. This can also be called storytelling with the environment.

2.5.3. Safe Zone

The term "Safe Zone" describes one or many designated areas within the current level where players are protected from any harmful interactions. The establishment of Safe Zones allows for the creation of designated areas where players can carefully assess their surroundings and strategically plan what they will do next without any risk or danger.

2.5.4. Layering

Layering describes the process of combining multiple components to create a new and unique experience. Layering is mostly used to introduce more difficult challenges to the player, without the necessity of introducing new elements.

2.5.5. Branching

Branching involves providing players different ways to achieve their purpose. Providing players with options leads to a sense of empowerment. Levels may include unrestricted branching, allowing players to freely choose from any accessible paths. This enhances the sense of discovery experienced while playing the game.

2.5.6. Pace Breaking

Pace breaking refers to purposely changing the dramatic arc of the game from one level to the next. It is often used to either increase or decrease tension, with the aim of encouraging more player engagement in the overall experience. A typical characteristic that leads to Pace Breaking is the inclusion of a danger that is noticeably challenging.

CHAPTER 3: PROCEDURAL GENERATION

Procedural Level Generation in video games is the utilization of algorithms to automatically generate game levels. This encompasses diverse forms of content, including landscapes, rules, quests, and other components. PLG frequently employs methodologies such as novelty search, wherein the evaluation of levels is based on their distinctiveness rather than specific goals. Another method involves reinforcement learning, in which a system acquires the ability to create levels by considering specific factors such as solvability and novelty. These methods enable the generation of varied and intricate levels, frequently with limited involvement from designers, resulting in a cost-efficient approach for game development (Beukman, Cleghorn and James, 2022).

Rogue, developed by Michael Toy and Glenn Wichman and released in 1980, is a dungeon crawling game that is considered a pioneer in using procedural generation. It splats out up to nine rooms and connect them randomly. This game represented an important milestone in the history of game development, specifically within the rogue-like genre. In the game Rogue, the dungeon levels, and the placement of objects within them are generated randomly. This procedural generation approach was revolutionary at the time and established the basis for numerous subsequent games.

3.1. Procedural Generation in 2D Platformer Games

Procedural Generation in 2D Platformer Games is a branch of Procedural Content Generation that specifically deals with the algorithmic generation of game content, while considering the constraints of limited or indirect user input. This technique is utilized to minimize the expenses associated with game design and generate a substantial quantity of game content, which poses a particular challenge in the context of 2D platformers. The objective is to create levels that are not only accessible and distinct, but also captivating and enjoyable. The development of level-generating algorithms is crucial as it prevents the creation of monotonous and repetitive levels in 2D platformers (Egana, 2018).

Procedural generation is an approach that involves generating data algorithmically instead of manually. It is frequently employed in video games to autonomously produce a substantial quantity of content, providing advantages such as reduced file sizes, increased content diversity, and less predictable gameplay. Procedural generation in 2D platformer games can be employed to dynamically adjust the difficulty level based on a player's skills. One way to accomplish this is by utilizing techniques such as genetic algorithms, which facilitate the attraction of a wider variety of players by generating a multitude of levels (Latif et al., 2022).

3.2. Current Approaches

Multiple approaches exist for procedural level design in 2D platformer games. Each of these methods possesses unique advantages and difficulties, and they are often used in conjunction to attain the desired equilibrium of unpredictability, enjoyability, and aesthetic attractiveness.

3.2.1. Genetic Algorithms

This approach includes the simulation of the natural evolutionary process. Algorithms use efficiency criteria such as playability, difficulty, and aesthetic appeal to choose and reproduce generations of level designs. Through multiple iterations, this method can generate game levels that are both highly optimized and diverse (Zafar and Mujtaba, 2020)

3.2.2. Graph-Based Generation

This method involves the establishment of a predetermined set of rules, known as a graph, which is used to create different levels. These rules define the placement and interconnection of various elements within a level. This approach enables a methodical yet diverse level design process (Dormans and Bakkes, 2011)

3.2.3. Machine Learning-Based Generation

In this approach, machine learning models are trained on a dataset of existing game levels. Subsequently, the model produces new levels by utilizing patterns it acquired from the data. It has the potential to generate groundbreaking designs that seamlessly integrate different elements of the training data (Summerville et al., 2017).

3.2.4. Noise-Based Generation

Perlin noise and Simplex noise techniques are employed to generate level layouts that are both random and consistent. This method is particularly advantageous for generating realistic terrain and characteristics within a gaming environment (Smith, Treanor, Whitehead and Mateas, 2009)

3.2.5. Tile-Based Generation

Levels are generated by arranging predetermined tiles within a grid. The placement of these tiles is determined by algorithms, which follow specific rules or incorporate randomness. This ensures that each level is both distinct and logically consistent (Jadhav and Guzdial, 2021). On figure 9, blue tiles represent non-soil environment, such as oceans.

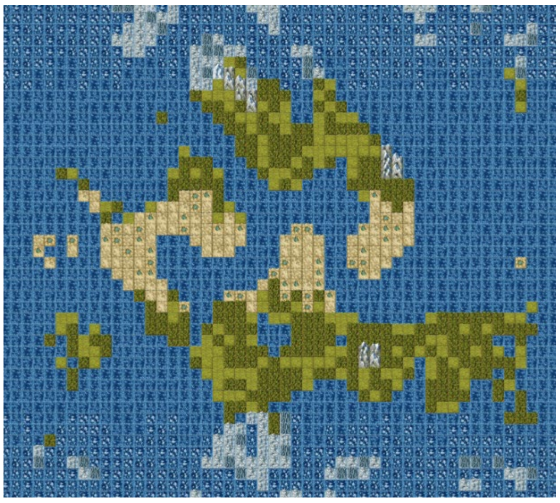


Figure 9. 2D Tile Based Generation Map Example

3.3. Edgar – The Level Generator

Level generators provide options for customizing the quantity of rooms, their dimensions, corridor length, and other parameters. To exert control over the arrangement of constructed levels, it is sometimes possible to specify the preferred degree of simplicity or complexity, as well as the quantity of corridors to be linked to each room. Utilizing the provided configuration, they create a randomized arrangement for the level.

An important advantage of these generators is their simple and efficient setup, enabling you to start designing levels in just a few minutes. An additional advantage is that these algorithms can effectively generate layouts consisting of many rooms, ranging from tens to hundreds, thanks to the constraints imposed on the structure of the level. Hence, the quest for an appropriate dungeon is relatively uncomplicated. These level generators are highly compatible with most games.

The main goal of this level generator is to empower game creators with complete control over the configuration of the generated layouts. This is achieved by defining level (connectivity) graphs, which specify the intended quantity of rooms to be produced and how they should be interconnected. Moreover, the visual aspect of rooms is governed by room templates.

3.3.1. Why Was the Edgar Level Tool Used?

Edgar is a Unity Game Engine plugin designed to generate 2D platformers using procedural techniques. Its main objective is to provide level designers with full control over the levels that are generated. The system utilizes a graph-based methodology for procedural generation, complemented by designed room templates, to produce levels that possess a sense of unity and consistency. It is a graph-based approach to the procedural generation.

This approach was preferred in this thesis and used in the developed game, because some successful games today were also designed with this method, like Dead Cells (2018).

3.3.2. Primary Elements

3.3.2.1. Room Templates

Room templates are utilized to regulate the visual characteristics of individual rooms. These are predetermined components that the algorithm selects from when creating a level. Unity tile maps are utilized to construct them, while they can also incorporate supplementary game objects such as lights, enemies, or chests containing loot. The designer can allocate various room templates to different categories of rooms.

3.3.2.2. Level Graphs

A level graph comprises interconnected rooms. Every room in the generated level pairs with another room, and each connection signifies that the algorithm requires the two rooms to be connected either directly or via a corridor.

Presented in figure 10, is a simple graph comprising five rooms and four connections. By applying this level graph as an input for the algorithm, each generated level will comprise five rooms, with room one connected with all other rooms within the level.

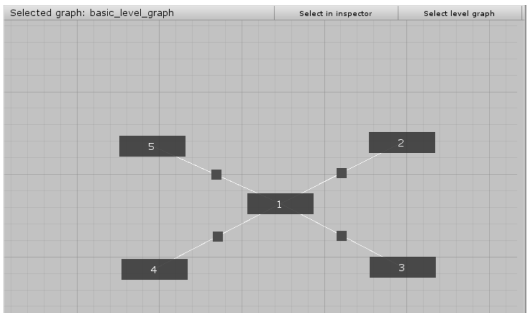


Figure 10. Simple level graph with five rooms and four door connections.

3.3.3. Workflow

3.3.3.1. Drawing Elements

The first stage is to design what are known as room templates, which specify the layout of each room and how it can be connected to other rooms.

3.3.3.1.1 Rooms

The rooms (sections) are designed as separate pieces by the level designer. These rooms can be grouped individually or kept in a common pool. The grouping of rooms can be separated by parameters such as their purpose, playing time and difficulty. In Figure 11, these two rooms' purpose is to start and finish. On Figure 12, there are four different room templates which originally taken from plugins example.

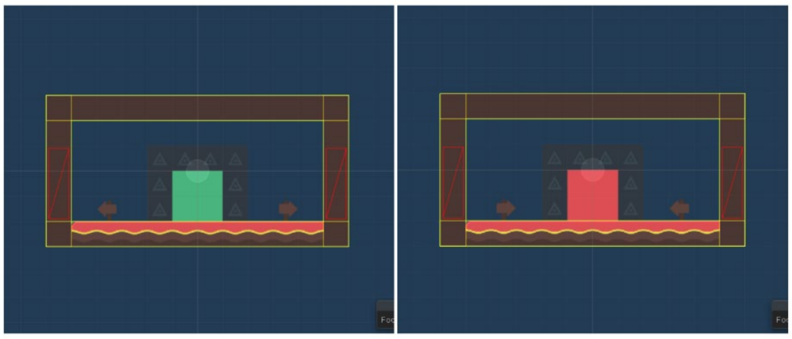


Figure 11. Start and goal template.

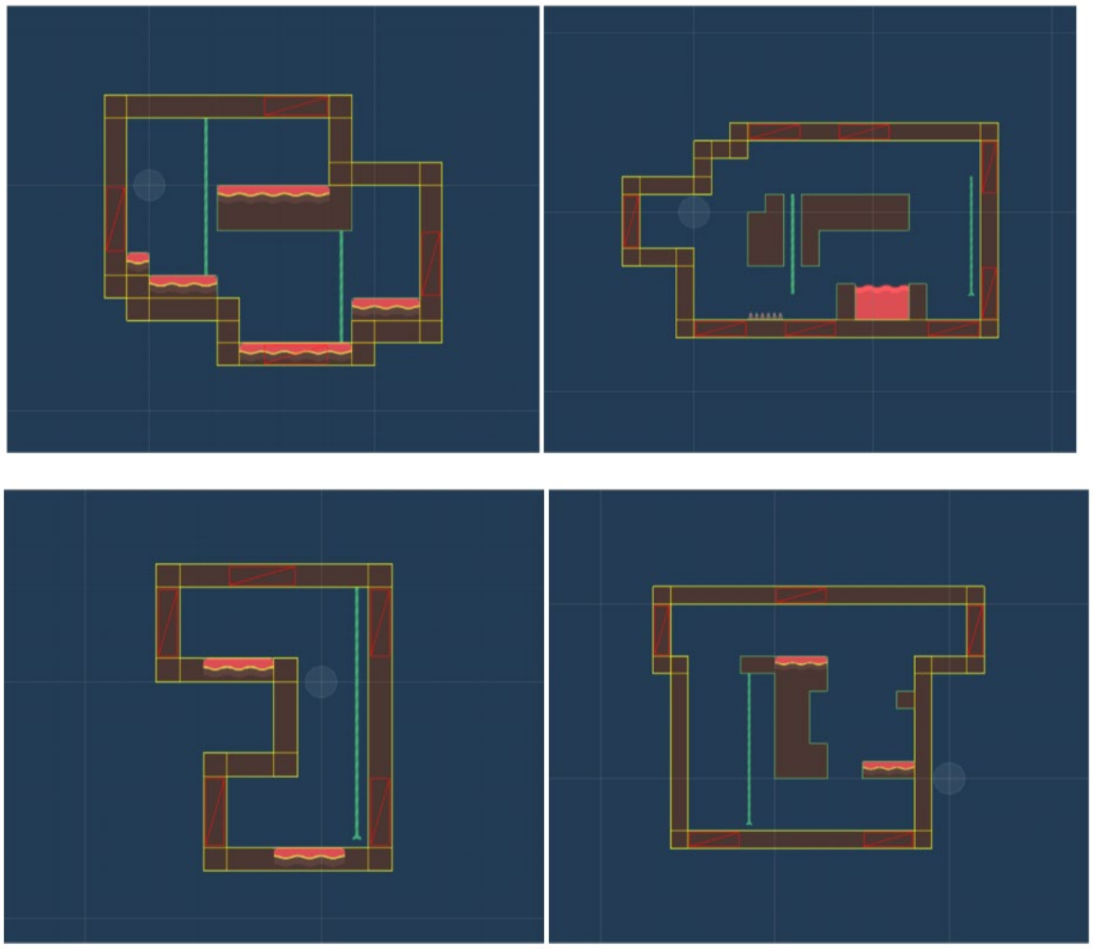


Figure 12. Four different room template examples.

3.3.3.1.2 Doors - Corridors

Doors-corridors are elements used to connect rooms together as seen in figure 13.

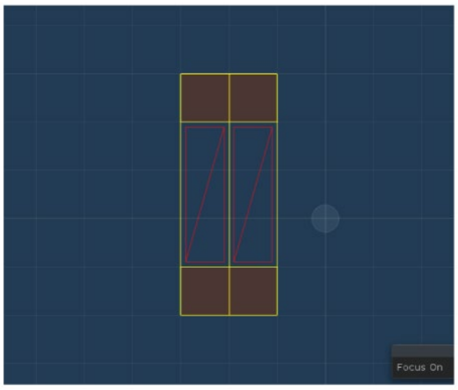


Figure 13. Door.

3.3.3.2. Prepare Structure

The process of creating a "level graph" is the second step (figure 14). The generator includes a graph editor that allows the designer to specify the number of rooms and their configuration. Additionally, a designer can decide which rooms will have distinct room templates from others; for example, a boss room and a spawn room will have different designs.

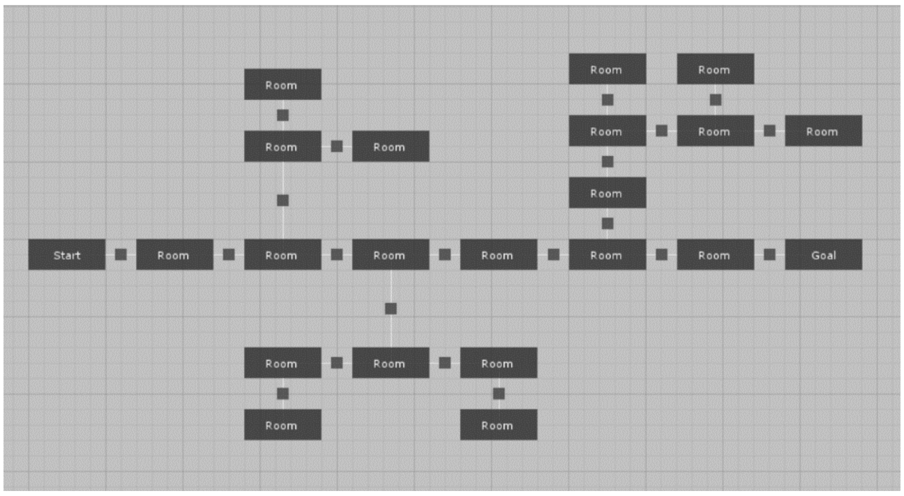


Figure 14. Level graph example.

3.3.3.3. Generate Levels

To complete the process, simply attach the generator component to a game object and assign the level graph obtained from the previous step. Once this is done, the system will be fully prepared to generate levels.

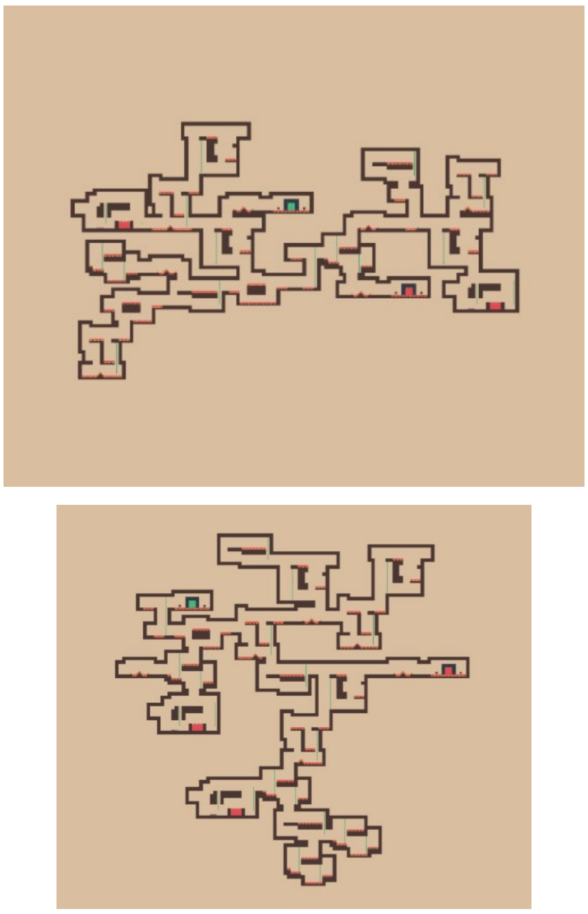


Figure 15. Example level results.

Figure 15 represents two level templates where the same values are entered, and different results come out.

3.3.3.4. Post Processing

Once a level is generated, it is frequently necessary to execute additional operations such as enemy spawning.

CHAPTER 4: CASE STUDY

4.1. Case Study Games – Analysis of 2D Platformer Levels

The game developed for this thesis focuses on fundamental game mechanics such as jumping, crossing gaps, avoiding spikes, and gathering apples along the horizontal axis.

Table 1. Entities used in two games.

Components	**Subheadings**	**Visual**	**Definition**
Level	-	-	Section of the game, where players control the avatar
Collectable	-		
Avatar	-		The entity that players control within the game
Moving Platforms	-		
Triggers	Checkpoint		Entity that will be activated when hovered over
	Trampoline		
TileSet	Tile		A surface on top of which the avatar can move
	Platform		
Obstacle	Spike		Entity that the player must avoid
	Gaps		
	Enemy		

The components of platformer levels are classified based on the specific functions they serve within a level. There are seven components in these two games, which are listed in the table below. Subheadings, visuals, and definitions of these seven components are given in Table 1. On Table 2, there are comparison of games.

Table 2. Comparison of both games

	Human-Designed	Procedural Generated
How many levels does game have?	6	6
What was the average time spent on designing each level?	60 minutes	30 minutes
Is the game replayable? How many times can player play over and have different level structures?	1	720 (permutation)
Does the game allow for dynamic adjustment of level/environment difficulty?	No	Yes
Does the approach to level design prioritize narrative integration?	Yes	No
Is there engagement and immersion of the designer to the game?	Yes	No

4.1.1. Game Mechanic

The avatar in case study games has the ability to jump. Considerations like jump height and distance should be made when designing levels. Moving platforms, enemies, gaps, and spikes may all test a player's ability to use the jumping ability. Also, the double jump feature is activated in the event of consecutive jumps.

4.1.2. *Progression and Difficulty Increase*

A level's design must provide a sense of progression that allows players to advance their abilities. While later stages might present more difficult obstacles, initial levels normally educate players in fundamental mechanics.
The degrees of difficulty should rise steadily. Players should have an engaging and difficult experience from elements like a rise in the number of enemies, intricate platform layouts, or time-based tasks.

4.1.3. *Prominent Game Elements*

In games, there are two kinds of triggers. Among these are the checkpoint, which is automatically recorded as the player progresses through the level, and the trampoline, which helps in the avatar's jumping and significantly simplifies the player's task.

4.2. *Human-Designed Benefits*

One of the key distinctions between human-designed levels and procedural generation is the incorporation of human emotions. Particularly in terms of storytelling, levels that convey various messages deepen the game's level designs. As discussed in Section 2.5 '2D Platformer Games Level Design Patterns', this aspect is more pronounced in human-designed levels. The level designer's ability to empathize with the player helps in understanding what the player expects from that level, allowing for a more effectively managed pacing. Special touches made by the designer in the levels can contribute to making the game feel 'more alive'.

4.2.1. *Specific Gameplay Instances*

The careful and deliberate creation of particular gameplay moments is matched by the human-design. This could be an important component of the storyline, a significant obstacle for the player, an important point in the narrative that all players must experience regardless of their prior choices, a notable item or tough enemy, or a small segment within a level. Procedural content generation, by its nature, is unable to achieve precise outcomes. If the generating algorithm's requirements and parameters were set to extremely restrictive limits, resulting in the same outcome every time, it would be practically identical to manually creating and saving the specific gameplay moment desired by the designer. In order for procedural level generation to truly exhibit procedural characteristics, it is essential to allow for flexibility and diversity. In certain instances, this can still allow for variation while guaranteeing the presence of essential elements. Therefore, there exists a continuum between procedural generation and human-design, encompassing all possible design choices. The creation of certain story elements or specific cinematic sequences poses a significant challenge in terms of generation. Therefore, manual design is necessary to carefully construct these gameplay moments that cannot be adequately produced by the game's procedural content generation system. This is particularly true when the game aims to convey a specific story in a particular manner, necessitating the inclusion of precise narrative elements (Johnson and Totten, 2017).

4.2.2. Narrative

The few games that have attempted to tell intricate procedural narratives frequently consist of peculiar character behavior, anomalies, and contradictions; rare moments of clarity are typically more due to the player's incorrect understandings than to the developer's intention and design. In contrast, when creating a landscape, people are more likely to notice gaps in stories than strange geological formations. Although there is a growing corpus of academic work on the subject, the methods for creating puzzles and narratives are still in their infancy when compared to the methods for creating gameplay spaces, and they haven't been integrated into games to the same degree (Li and Riedl, 2015). To respond fluidly and adaptively to in-game events, procedural narratives need to be heavily engaged with other game factors, such as other characters, factions, and locales, which may also be generated. This contrasts with static backdrops that are generated once and then played upon rather than played with. If procedural narratives are to create characters and narratives that are as compelling and nuanced as their handmade counterparts, they must also adjust to a generated world and the player's significant actions, which are frequently unpredictable.

In other cases, a procedurally generated game might want to tell a handcrafted tale rather than have the narrative generated; if this were the case, the game would run into the issues mentioned above.

As a result, a lot of games choose to write their narrative segments by hand, either because they have a particular story in mind or because procedural narrative generation seems too complicated and writing it by hand is easier.

4.2.3. Empathy

The most significant difference between human-designed levels and procedurally generated ones is the designer's capacity for empathy. The designer can experience many emotions during the design phase and can incorporate every detail they want the player to feel. Utilizing balance and rhythm (as outlined in 2.4.4 - Basic Design Components in Level Design) becomes more manageable, allowing for effective pacing control. This approach enables the designer to deliver the exact experience intended for the player. Emotional control rests in the hands of the designer, allowing for a more nuanced and impactful game experience.

4.3. Procedural Generated Benefits

There are three main reasons for using procedurally generated level design: to ensure replayability, to create challenging gameplay experiences, and to focus gameplay experiences on the theme or mechanic of exploration. The outcome is contingent upon the meticulousness of the level designer's documentation during the creation of the procedural level.

4.3.1. Replay

Using procedural generation, the game's gameplay varies with each playthrough. Enemies are positioned in different locations, collectable items are found in different places, level layouts change, and elements encountered in one playthrough may not be present in the next. This feature significantly enhances the replay value of these games compared to their manually designed counterparts, where nothing changes in each new playthrough. Ordering six levels in permutation offers much more replayability than playing six levels in order as seen on table 2.

4.3.2. Challenge

One of the most challenging tasks for a level designer is adjusting the difficulty of a game. As discussed in Section 2.4.3 'Level Design Process', the topic 'knowing the audience' is of critical importance. Factors such as who will play the game and the skill levels of the players play a significant role. In this context, distinguishing between human-designed levels and procedurally generated ones is also likely. The designer's ability to create and prepare levels for different difficulty levels can not only increase the targeted player segment but also allows for an active, evolving difficulty system.

4.3.3. Exploration

In platformer games, it is crucial to provide players with various paths for progression. This could be a path that becomes accessible later in the game with a newly acquired ability, or it could represent a 'personal choice' based on the player's preference for certain levels. Procedural level generation is a favorable choice as it can offer multiple alternatives and diversify these options.

4.3.4. Cost

It requires fewer human resources, although the process of creating complicated algorithms can pose challenges and consume a significant amount of time. As seen in Table 2, it takes two times longer to design human-designed levels than to design levels designed for procedural generation.

4.4. Dead Cells – Successful Hybrid Approach Example

Dead Cells, developed by Motion Twin, was released in the digital entertainment industry on August 7, 2018. This action-platformer game challenges players to navigate a large and ever-changing castle, depending on their ability to defeat the powerful guardians inside. The primary goal of the game is to achieve a high level of proficiency in complex combat mechanics, which is emphasized by the constant presence of permanent death. This factor presents a notable obstacle: there are no designated points of verification or progress tracking. The gameplay loop is defined by a recurring sequence of combat, death, acquisition of knowledge, and repetition. This design decision enhances the significance of strategic learning and the acquisition of skills in the player's progression.

An essential element of Dead Cells is the gradual and interconnected exploration of its world (Figure 19). This aspect is enhanced by the game's high replayability and the intense adrenaline-inducing tension caused by the permadeath feature. The combat system is characterized as demanding yet fair, showcasing a collection of over fifty weapons and spells, each providing distinct gameplay encounters.

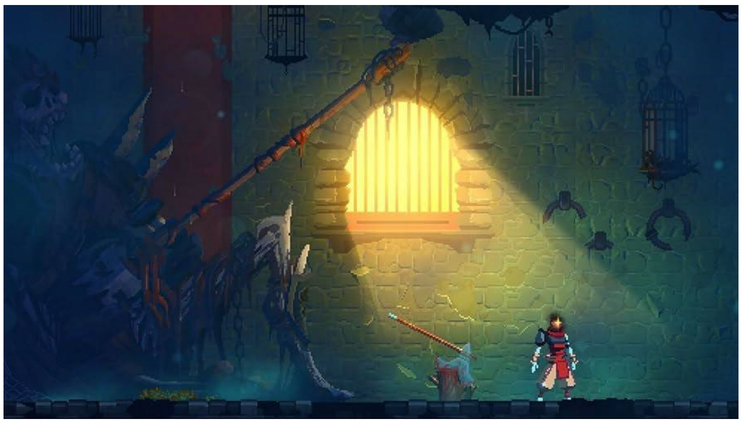


Figure 16. Dead Cells In-Game Screenshot

Significant aspect of the game is its non-linear progression. The game undergoes transformations depending on the player's decisions and the permanent abilities they acquire, encompassing locations such as the Sewers, Ossuary, and Ramparts. These capabilities enable the player to easily access new routes that are in line with their current character build, preferred style of play, or mood.

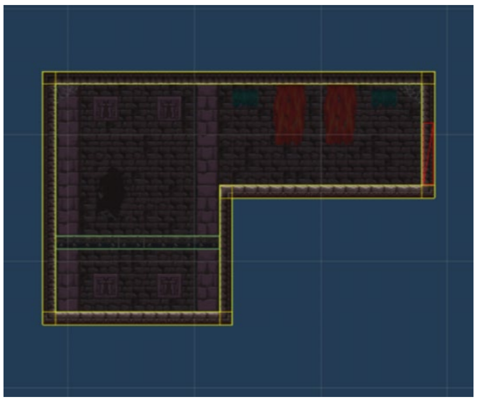
The game incorporates exploration as a prominent element, offering players the opportunity to discover secret rooms, concealed passages, and visually stunning landscapes. This design element promotes extensive exploration and interaction with the game world, thereby enhancing the overall player experience. Dead Cells represents the convergence of strategic gameplay, demanding mechanics, and captivating exploration, all taking place in a dynamic narrative environment.

About one year after its initial release in early access, Dead Cells achieved sales of more than 730,000 units, and exceeded 850,000 units before its official launch. Dead Cells had sold two million units by May 2019, ten months after its full release. With the announcement of their third DLC in March 2021, Dead Cells had achieved a sales milestone of 5 million copies sold (Stockdale, 2019). These sales showed that the game was successful.

Sebastien Benard is a lead designer on Dead Cells and explains their design method as a hybrid (both human designed and procedural generated) approach (Benard, 2017).

Initially, they started on the task of creating a prototype for Dead Cells, choosing for the traditional, human designed approach to level design. However, they soon realized that given the limited size of their team, they would not have sufficient time to execute it effectively. Team decided to use procedurally generated levels, thought of procedurally generated levels enhances the replayability of the game, particularly for game that feature permadeath mechanics. Moreover, the team discovered that it fundamentally transformed the sensation of the game's combat, shifting the focus to the player's instincts and reflexes instead of relying on memorizing a level to advance. However, although the core gameplay experience was enhanced by the novelty of the new arrangement of enemies, the narrative and the empathy lost significantly. Simply put, it was irrational, disordered, and failed to provide a sense of unity or connection with the world. The team was dissatisfied with either fully handcrafting or fully procedural generation, so they decided to combine both methods and create a hybrid approach.

The team developed their own “Artificial Intelligence Director”, adapted for Dead Cells, which quite similar system with Edgar Level Tool, graph-based generation.
First, the team designed a ‘rooms’ with a specific number of potential variations in each, depending upon their arrangement, there are several examples of Dead Cells Room template in figure 20.



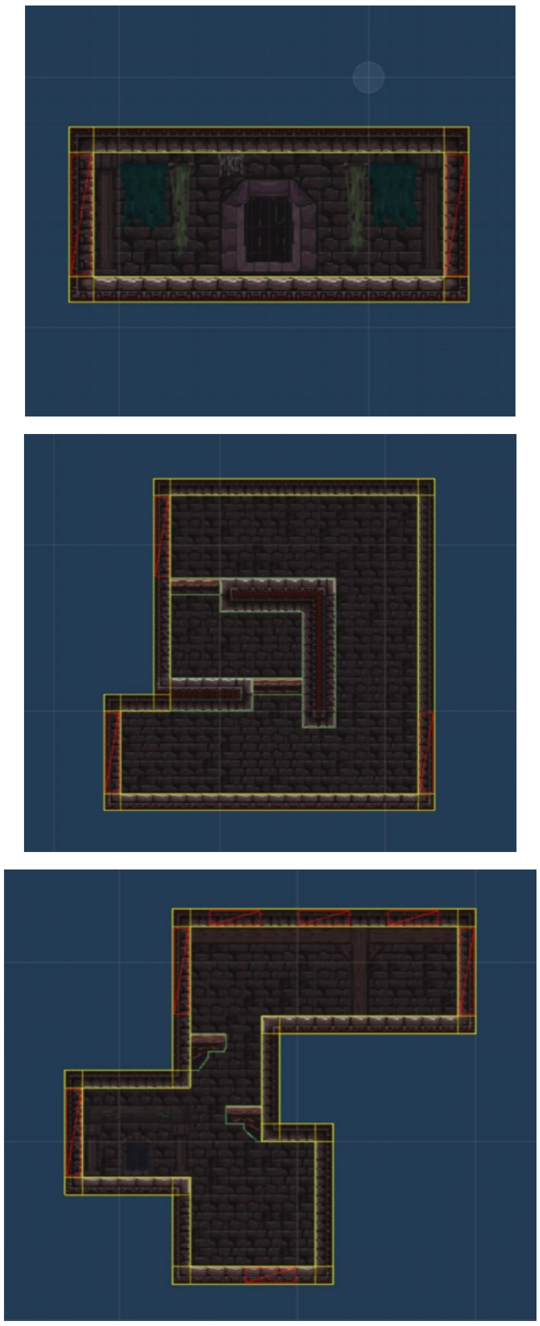


Figure 17. Dead Cells Room Examples

Practically, every room is equipped with a unique arrangement of platforms that are specifically designed for a specific purpose. A room specifically intended for concealing a valuable treasure will differ significantly from one intended for accommodating a merchant, and both will be markedly distinct from rooms designed for combat purposes. There are potential variations that might take place in the levels that are designed by human.

Each room belongs to a level. For example, rooms utilized in the prison are not repurposed in the sewers. This enables us to assign a distinct and well-defined identity to each level. For instance, the small width of the sewers severely limits the player's capacity to jump and dodge, compelling them to carefully consider how they handle the enemies.

Second, the fixed elements were positioned, serving as a framework for the procedural generation to display itself. The map's overall design encompasses the interconnections between different levels, as well as the strategic placement of keys that unlock new paths for subsequent runs. The global arrangement of the world is predetermined, and human designed (Figure 21).

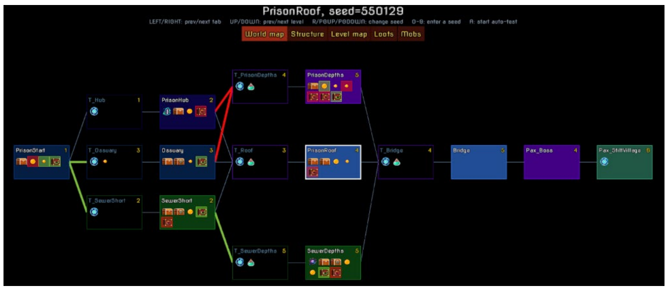


Figure 18. Dead Cells World Map

A concept graph was generated for every level. A graph is a diagrammatic representation of the arrangement of rooms within a level, represented by nodes. The entrance and exit of the level were positioned, followed by the incorporation of special rooms such as treasure chambers. The graph serves as a guide for the procedural generation algorithm, providing specific instructions regarding the level's length, quantity of special tiles, degree of labyrinthine structure, distance between the entrance and the nearest exit, and other relevant parameters.

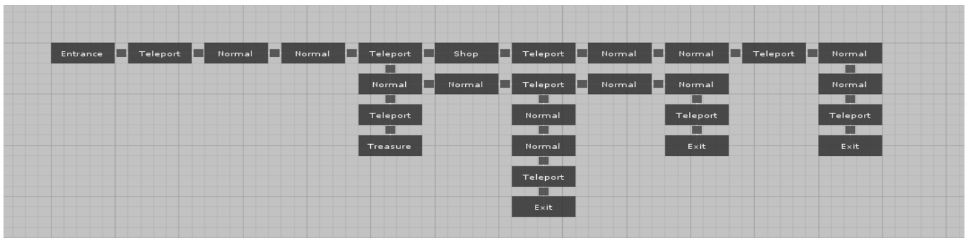


Figure 19. Dead Cells Level Graph Example

The algorithm selects a random room from the set of rooms specified for the current level of the node, and then checks if it matches the specifications outlined by the graph, such as location, number of entrances, and type (Figure 22). If there is no match, the algorithm proceeds to attempt another room until it discovers a suitable one and generates levels (Figure 23).

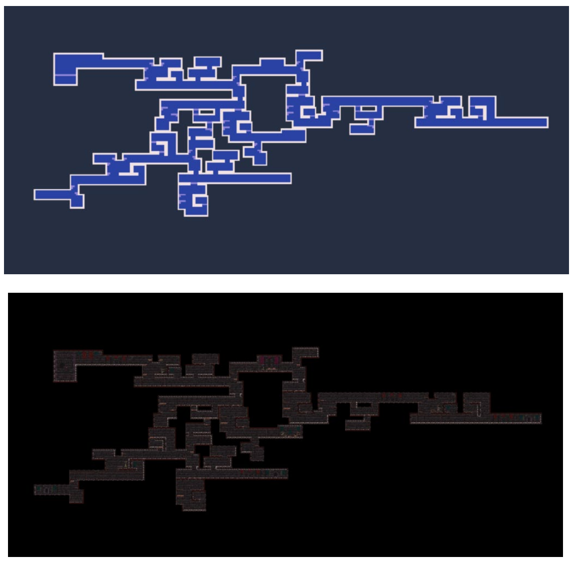


Figure 20. Dead Cells Generated Level Example

4.5. Result

The concepts of procedural level generation and human-design appear to be distinctly dichotomous. One generates a vast number of possible combinations, while the other provides only one specific example. One game emphasizes replayability, while the other lacks significant new content for players who engage in a second playthrough.

The study concluded that procedural generation is a design technique that prioritizes replay value, the duration of engaging gameplay, and the significance of novelty. It also found that procedural level generation can introduce strategic and reflex challenges that are not commonly found in human designed games, especially when considering multiple playthroughs. Furthermore, the study determined that games that focus on exploration and discovery are particularly suitable for a model of the game world where the player acknowledges the unpredictability of the unseen and the uncertain range of the game's variations (Figure 24). Nevertheless, procedural content generation is unable to generate precise instances of assured gameplay and faces difficulties in generating the narratological aspects that human design tends to excel in (Figure 25).

The combined use of procedural level generation and human design can produce significantly better games than using either method alone. This approach not only facilitates the designer's work but also ensures the retention of originality. For instance, manually designing levels that are crucial for the game's progression and narrative is appropriate. In contrast, procedural level generation can be employed for levels that follow a more uniform structure. This hybrid method allows for a balance between unique, story-driven content and efficient, varied level production, enhancing both the gameplay experience and the design process.

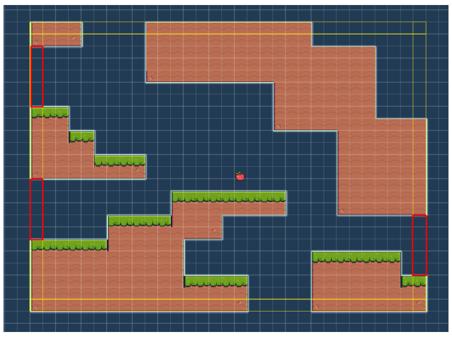


Figure 21. One of the 6 levels; designed for the procedural generated game, the junction points (doors) of the levels are marked and 5 fixed units.

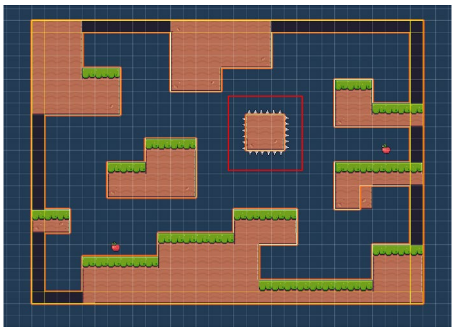


Figure 22. One of the 6 levels; designed for the human-designed game, the marked area represents the enemy that will appear in the next level.

Table 3. Comparison of Criteria

Criteria	**Human-Designed**	**Procedural Generated**
Specific Gameplay Instances	High-quality, balanced gameplay with uniform experiences due to designer oversight.	Gameplay quality varies with algorithm complexity, leading to potential inconsistencies.
Narrative	Human creativity at its peak with original, narrative-driven designs.	Limited in uniqueness and creativity due to computational constraints.
Empathy	Designer empathy allows for meeting player expectations.	Challenging for both player and designer to connect with the game.
Replay	Lower replayability due to consistent gaming experience per session.	High replayability potential with distinct, varied experiences in each playthrough.
Challenge	Consistent challenge in level design.	Difficulty adjusts to player style; focus on diverse level difficulties.
Exploration	Explorable areas as intended by designer, leading to similar gameplay maps for players.	Diverse level designs enhance exploration, offering multiple ways to play.
Cost	Needs more human power and time, increasing costs but yielding reliable results.	Fewer human resources needed, but complex algorithm creation is time-consuming.

4.6. Future Work - Three-Dimensional Game Levels

This thesis purposefully focuses on the analysis of two-dimensional platformer games, with limited scope and subject matter. The decision was made to have a concentrated, controllable, and thorough examination of design principles. Given the ongoing development and growing intricacy of game design and technology, it is highly important to expand the scope of this study to include the analysis of three-dimensional games.

In future work, it can expand upon the knowledge acquired in this thesis by specifically investigating three-dimensional game design and procedural generation techniques. This new method has the capacity to provide a more comprehensive perspective on elements such as spatial dynamics, depth of interaction, and player immersion in the field of game design. Three-dimensional games, equipped with advanced technological elements like physics engines, artificial intelligence, and visual effects, offer a more comprehensive understanding of game design and player experience.

Moreover, researching 3D games could provide a clearer understanding of the core distinctions and similarities between two-dimensional and three-dimensional game designs, thereby enhancing game design theory and practice. This comparative analysis aims to deepen our comprehension of how the utilization of procedural generation and human-design can lead to distinct outcomes in various aspects, ultimately influencing the overall gaming experience.

To summarize, the forthcoming research can focus on conducting an in-depth study of three-dimensional games. The objective is to gain a more extensive comprehension of game design and development, and to propose inventive design approaches that are in line with the current trends in the gaming industry.

CHAPTER 5: CONCLUSION

In previous chapters, we discussed the importance and principles of game design. There is a lot of design work involved in game development, which is a separate field. In this context, level design was examined, with data gathered from various sources to first interpret primary elements within a sample game. Information was provided on level design, one of the most crucial branches of game design, followed by an exploration of 2D platformer games, a genre that has remained popular for over 30 years and is still one of the most favored. A study was carried out to investigate the reasons behind the success of these games and what aspects they have done well, gathering relevant data. Contemporary approaches to procedural generation were studied, and the popular and proven 'graph-based generation' method was chosen. A game was designed using this method with the aid of a plugin. During this process, insights from level designers were gathered, and the advantages of both human-designed and procedurally generated levels were identified. The study concluded that the combined use of both approaches is the most effective method, as one compensates for the shortcomings of the other.

This study targets the experience and work of the game-level designer, showcasing how procedural generation can contribute to the design process. Its strengths have been demonstrated through hands-on game development.
The combination of human creativity and procedural generation in game level design signifies a major advancement in game development. This hybrid approach utilizes the advantages of both methodologies to generate captivating and dynamic gaming experiences. Human designers possess an advanced understanding of narrative, empathy, and specific gameplay instances, which are essential for crafting levels that leave a lasting impact and evoke strong emotions. On the other hand, procedural generation provides extensive opportunities for variety, unpredictability, and scalability, thereby significantly improving replay value and maintaining a consistently unique gameplay experience (Table 3).

In the future, this research can be extended to include 'player reviews and perspectives.' While this process may be enjoyable for the game designer, it raises curiosity about how players will respond to games that are similar in every aspect, with the only difference being that one features human-designed levels and the other includes procedurally generated levels. Investigating player reactions and preferences in this context could provide valuable insights into the effectiveness and reception of these design methodologies in the gaming community. Another potential avenue involves the study of advanced artificial intelligence and machine learning methods in the field of procedural generation. As these technologies progress, they have the potential to provide enhanced capabilities for generating content that is not only random or diverse but also rich in context and narrative. This could entail the utilization of AI systems that acquire knowledge from an extensive collection of level designs and player interactions to produce levels that are not only new but also deeply captivating and challenging in manners that hold significance for players.

Furthermore, by incorporating real-time feedback mechanisms into games, it becomes possible to implement adaptive-level design. This means that the game can automatically modify the difficulty, style, and content of levels based on the player's skill level, preferences, and in-game actions. This has the potential to result in gaming experiences that are highly customized to individual preferences, causing a blending of the boundaries between intentionally created and automatically generated content.

In conclusion, the combination of human and procedural level design has great potential for the future of game development. By further investigating this collaboration, researchers and developers can discover fresh opportunities for generating immersive, dynamic, and captivating gaming encounters. Following studies, based on extensive input from players and utilizing advanced artificial intelligence, have the potential to result in game designs that are increasingly adaptable, engaging, and captivating, ultimately pushing the limits of what games can achieve.

REFERENCES

Adams, D. (2002) *Automatic Generation of Dungeons for Computer Games.* [Online]. Available at: https://xenophule.com/girscloset/Dungeon_Procedural_Generation.pdf (Accessed: 24 April 2023).

Aramini, A., Lanzi, P. and Loiacono, D. (2018) *An Integrated Framework for AI Assisted Level Design in 2D Platformers.* IEEE Games, Entertainment, Media Conference (GEM).

Athavale, S. and Dalvi, G. (2018) *A Method to Study Purposeful Game Design Process.* IEEE 6th International Conference on Serious Games and Applications for Health (SeGAH).

Benard, S. (2017) *Building the Level Design of a Procedurally Generated Metroidvania: A Hybrid Approach* [Online]. Available at: https://www.gamedeveloper.com/design/building-the-level-design-of-a-procedurally-generated-metroidvania-a-hybrid-approach-#close-modal (Accessed: 29 September 2023).

Beukman, M., Cleghorn, C. and James, S. (2022) *Procedural Content Generation using Neuroevolution and Novelty Search for Diverse Video Game Levels.* Genetic and Evolutionary Computation Conference; July 9–13, 2022; Boston, MA, USA

Bycer, J. (2020) *Game Design Deep Dive Platformers.* 1st edition. New Jersey: CRC Press.

Byrne, E. (2005) *Game Level Design*. 1st edition. Boston: Charles River Media.

Camilleri, E., Yannakakis, G. and Dingli, A. (2016) *Platformer Level Design for Player Believability*. IEEE Conference on Computational Intelligence and Games

Ching, F.D.K. (2002) *Architecture: Form, Space and Order*. 3rd edition. New Jersey: John Wiley & Sons, Inc.

Co. P. (2006) *Level Design for Games, Creating Compelling Game Experiences*. 1st edition. California: New Riders.

Çatak, G. (2003) *Bilgisayar Oyunlarında Mimarinin Kullanımı*. Master Thesis. İstanbul, YTÜ.

Dormans, J. and Bakkes, S. (2011) *Generating Missions and Spaces for Adaptable Play Experiences*. IEEE Transactions on Computational Intelligence and AI in

Games, Vol. 3(3), pp. 216-228

Fisher, T. (2008) *Architectural Design and Ethics, Tools for Survival*. 1st edition. Amsterdam: Elsevier

[Game Maker's Toolkit]. (2020) *How Level Design Can Tell a Story* [Video]. Available at: https://www.youtube.com/watch?v=RwlnCn2EB9o&t=5s

[GDC]. (2017) *Level Design Workshop: Architecture in Level Design* [Video]. Available at: https://www.youtube.com/watch?v=XW7KvppTspc&t=482s

[GDC]. (2018) *Level Design Workshop: Blockmesh and Lighting Tips* [Video]. Available at: https://www.youtube.com/watch?v=09r1B9cVEQY&t=2799s

Jadhav, M. and Guzdial, M. J. (2021) *Tile Embedding: A General Representation for Procedural Level Generation via Machine Learning*. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.

Johnson, M. and Totten, C. (2017) *Level Design: Processes and Experiences*. 1st edition. New York: CRC Press.

Karavolos, D., Bouwer, A. and Bidarra, R. (2015) *Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation*. International Conference on the Foundations of Digital Games

Karlsson, T., Brusk, J. and Engström, H. (2023) *Level Design Processes and Challenges: A Cross Section of Game Development*. Games and Culture Vol. 18, Issue 6, pp. 821-849.

Kremers, R. (2009) *Level Design: Concept, Theory, and Practice*. 1st edition. Massachusetts: A K Peters, Ltd. Natick.

Latif, A., Zuhairi, M., Khan, F., Randhawa, P. and Patel, A. (2022) *A Critical Evaluation of Procedural Content Generation Approaches for Digital Twins*. Hindawi Journal of Sensors Vol. 2022.

Li, B. and Riedl, M. (2015) *Scheherazade: Crowd-Powered Interactive Narrative Generation*. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29(1).

McDaniel, R. and Daer, A. (2016) *Developer discourse: Exploring Technical Communication Practices Within Video Game Development*. Technical Communication Quarterly, Vol. 25(3), pp. 155–166.

Minkkinen, T. (2016) *Basics of Platform Games*. Kajaanin Ammattikorke Akoulu University of Applied Sciences.

Motion Twin (2018) *Dead Cells* [Video Game]. Motion Twin.

Nintendo (1985) *Super Mario Bros.* [Video Game]. Nintendo Entertainment System.

Risi, S. and Togelius, J. (2020*) Increasing Generality in Machine Learning Through Procedural Content Generation*. Nature Machine Intelligence.

[Roguelike Celebration] (2020) Herbert Wolverson - *Procedural Map Generation Techniques* [Video]. Available at: https://www.youtube.com/watch?v=TlLIOgWYVpI

Saltzman, M. (1999) *Secrets of the Sages: Level Design* [Online]. Available at: https://www.gamedeveloper.com/design/secrets-of-the-sages-level-design (Accessed: 13 October 2023).

Schell, Jesse. (2008) *The Art of Game Design*. 3rd edition. New Jersey: CRC Press.

Sega (1991) *Sonic the Hedgehog* [Video Game]. Sega.

Seraphine, F. (2014) *The Intrinsic Semiotics of Video Games*. Japan. The University of Tokyo.

Smith, G., Cha, M., and Whitehead, J. (2008) *A Framework for Analysis of 2D Platformer Levels.* Sandbox Symposium 2008, Los Angeles, California.

Smith, G., Whitehead, J., Mateas, M., Treanor, M., March, J. and Cha, M. (2011) *Launchpad: A Rhythm-Based Level Generator for 2-D Platform*ers. IEEE Transactions on Computational Intelligence and AI in Games. V.

Smith, G., Treanor, M., Whitehead, E., and Mateas, M. (2009) *Rhythm-based level generation for 2D platformers*. Vol. 3, pp. 175-182.

Silva, F., Khalifa, A. and Togelius, J. (2019) *Level Design Patterns in 2D Games*. IEEE Conference on Games.

Stockdale, H. (2019) *Dead Cells Has Now Sold 5 Million Copies Worldwide*. [Online]. Available at: https://www.nintendolife.com/news/2021/03/dead_cells_has_now_sold_5_million_copies_worldwide (Accessed: 11 November 2023).

Summerville, A., Snodgrass, S., Guzdial, M. J., Holmgård, C., Hoover, A. K., Isaksen, A., Nealen, A., and Togelius, J. (2017). *Procedural Content Generation via Machine Learning (PCGML)*. IEEE Transactions on Games, Vol. 10(3), pp. 257-270.

Suominen, J. (2012) *Mario's Legacy and Sonic's Heritage: Replays and Refunds of Console Gaming History*. Proceedings of DiGRA Nordic 2012 Conference: Local and Global – Games in Culture and Society.

[TEDx Talks Peter Burroughs]. (2016, April 15) *Creating New Worlds: A Journey Through Video Game Design* [Video]. Available at: https://youtube.com/watch?v=x313GWFiKgU

[TimDoesLevelDesign]. (2022, February 9) *Pre-Production for Level Design* [Video]. Available at: https://www.youtube.com/watch?v=lhNVxxv9KkY

[TimDoesLevelDesign]. (2021, December 29) *What is Video Game Level Design? A very brief Introduction* [Video]. Available at: https://www.youtube.com/watch?v=lAs5UngYPRs

Potocek, T. (2015) *Level Generation Techniques for Platformer Games.* [Online]. Available at: https://www.tobice.cz/publications/level-generation-techniques-for-platformer-games.pdf (Accessed: 25 August 2023).

Unity Technologies (2005), Unity Engine [Game Engine]. Unity Technologies Inc.

Whitton, N. (2016) *Digital Games and Learning*, Research and Theory. 1st edition. London: Routledge

Whitson, J. R. (2020). *What Can We Learn from Studio Studies Ethnographies? A "Messy" Account of Game Development Materiality, Learning, and Expertise*. Games and Culture, Vol. 15(3), pp. 266–288.

Zafar, A. and Mujtaba, H. (2020) *Search-based Procedural Content Generation for GVG-LG*. Applied Soft Computing Vol. 86, January 2020.

APPENDICES

The games created for this thesis were developed utilizing the Unity game engine. Source codes are provided in this section.

Player Controller

```
using UnityEngine;
public class PlayerController : MonoBehaviour
{
   public float moveSpeed = 3f;
   public float jumpForce = 5f;
   private bool isGrounded;
   private bool isJumping;
   private Rigidbody2D rb2D;
   private float jumpTimeCounter;
   [SerializeField] private float jumpTime;
   private float moveInput;
   private void Start()
   {
      rb2D = GetComponent<Rigidbody2D>();
   }
   private void FixedUpdate()
   {
      moveInput = Input.GetAxis("Horizontal");
      rb2D.velocity = new Vector2(moveInput * moveSpeed, rb2D.velocity.y);
   }
   void Update()
   {
      if (Input.GetButtonDown("Jump") && isGrounded)
      {
         isJumping = true;
         jumpTimeCounter = jumpTime;
```

```
        rb2D.velocity = Vector2.up * jumpForce;
    }
    if (Input.GetButton("Jump") && isJumping)
    {
        if (jumpTimeCounter > 0)
        {
            rb2D.velocity = Vector2.up * jumpForce;
            jumpTimeCounter -= Time.deltaTime;
        }
        else
        {
            isJumping = false;
        }
    }
    if (Input.GetButtonUp("Jump"))
    {
        isJumping = false;
    }
}
private void OnCollisionEnter2D(Collision2D other)
{
    Vector3 normal = other.GetContact(0).normal;
    if (normal.y > 0.5f) // To also take into account inclined surfaces
    {
        isGrounded = true;
    }
}

private void OnCollisionStay2D(Collision2D other)
{
    // If the character is still in contact with plane
    Vector3 normal = other.GetContact(0).normal;
    if (!isGrounded && normal.y > 0.5f)
```

```
        {
            isGrounded = true;
        }
    }
    private void OnCollisionExit2D(Collision2D other)
    {
        isGrounded = false;
    }
}
```

Player Life

```
using System.Collections;
using UnityEngine;
public class PlayerLife : MonoBehaviour
{
    private Animator anim;
    private Rigidbody2D rigidbody;
    private Vector3 startPos;
    public float deathHeight = -10f; // Ölüm için belirlenen yükseklik
    private void Start()
    {
        rigidbody = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        startPos = transform.position;
    }
    private void Update()
    {
        CheckDeath();
    }
    private void CheckDeath()
    {
        if (transform.position.y < deathHeight)
        {
            Die();
        }
    }
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.CompareTag("Trap"))
        {
            Die();
        }
    }
    private void Die()
    {
        rigidbody.isKinematic = true;
        rigidbody.constraints = RigidbodyConstraints2D.FreezePositionX |
RigidbodyConstraints2D.FreezePositionY;
        anim.SetTrigger("DeathTrigger");
        StartCoroutine(Respawn(0.5f));
    }
    IEnumerator Respawn(float duration)
    {
        yield return new WaitForSeconds(duration);
        anim.SetTrigger("RespawnTrigger");
        transform.position = startPos;
        rigidbody.isKinematic = false;
        rigidbody.constraints = RigidbodyConstraints2D.None;
    }
    private void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.CompareTag("Checkpoint"))
        {
            startPos = transform.position;
col.gameObject.GetComponent<Animator>().SetTrigger("CheckpointReachedTrig
ger");
        }
    }
}
```

Item Collector

```
using UnityEngine;
using UnityEngine.UI;
public class ItemCollector : MonoBehaviour
{
    private int collectibles = 0;
    private Text collectibleText;
    private Animator animator;
    private GameObject collectibleObject;
    private void Start()
    {
        collectibleText =
GameObject.FindWithTag("ScoreText").GetComponent<Text>();
        collectibleObject = GameObject.FindWithTag("Collectible");
        animator = collectibleObject.GetComponent<Animator>();
    }
    private void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.CompareTag("Collectible"))
        {
            collectibles++;
            animator.SetTrigger("CollectableTrigger");
            Destroy(col.gameObject);
            collectibleText.text = "Collectible : " + collectibles;
        }
    }
}
```

Camera Controller

```
using UnityEngine;
public class CameraController : MonoBehaviour
{
   private Transform player;
   private void Update()
   {
      player = GameObject.FindWithTag("Player").transform;
      transform.position = new Vector3(player.position.x, player.position.y + 4f,
transform.position.z);
   }
}
```

Waypoint Follower

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class WaypointFollower : MonoBehaviour
{
   [SerializeField] private GameObject[] waypoints;
   private int currentWaypointIndex = 0;
   [SerializeField] private float speed = 2f;
   void Update()
   {
      if (Vector2.Distance(waypoints[currentWaypointIndex].transform.position,
transform.position) < .1f)
      {
         currentWaypointIndex++;
         if (currentWaypointIndex >= waypoints.Length)
         {
                currentWaypointIndex = 0;
            }
        }
        transform.position = Vector2.MoveTowards(transform.position,
waypoints[currentWaypointIndex].transform.position,
            Time.deltaTime * speed);
    }
}
```

Platformer Post Processing

```
using System.Linq;
using Edgar.Unity;
public class Platformer1PostProcessing : DungeonGeneratorPostProcessingGrid2D
{
    public override void Run(DungeonGeneratorLevelGrid2D level)
    {
        RemoveWallsFromDoors(level);
    }
    private void RemoveWallsFromDoors(DungeonGeneratorLevelGrid2D level)
    {
        // Get the tilemap that we want to delete tiles from
        var walls = level.GetSharedTilemaps().Single(x => x.name == "Walls");
        walls.tag = "Ground";
        var platforms = level.GetSharedTilemaps().Single(x => x.name ==
"Platforms");
        // Go through individual rooms
        foreach (var roomInstance in level.RoomInstances)
        {
            // Go through individual doors
            foreach (var doorInstance in roomInstance.Doors)
            {
                // Remove all the wall tiles from door positions
                foreach (var point in doorInstance.DoorLine.GetPoints())
                {
                    walls.SetTile(point + roomInstance.Position, null);
                }
            }
        }
    }
}
```

Trampoline

```
using UnityEngine;
public class Trampoline : MonoBehaviour
{
    public float pushForce = 250f;
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            Rigidbody2D playerRigidbody =
collision.gameObject.GetComponent<Rigidbody2D>();
            playerRigidbody.AddForce(Vector2.up * pushForce,
ForceMode2D.Impulse);
        }
    }
}
```