1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Francisco Campos de Sá Nogueira Caiado

Generative Game Design as Common Interactive Space - Transmediating and Developing the Meerplay Digital Game

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering, advised by Professor Licínio Gomes Roque and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Đ
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Francisco Campos de Sá Nogueira Caiado

Generative Game Design as Common Interactive Space - Transmediating and Developing the Meerplay Digital Game

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering, advised by Professor Licínio Gomes Roque and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2023

Acknowledgements

I would like to thank both Licínio Roque and Mariana Seiça for their help and guidance throughout the development of the project.

Major thanks to Valéria Pinto for always being available to discuss her game and design implications and, most importantly, for setting up the gameplay sessions with the target audience, which would not have been possible without her.

Thanks to João Calhau, for helping me in integrating the Erlang Server with the game and explaining how to use its console.

A big thank you to all the friends I have made throughout five years studying Informatics Engineering for their support and companionship.

Special thanks to my family, for being so supportive, caring, and always keeping me on my toes.

Final thanks to my girlfriend, Stéphanie, who had to endure countless nights of button and mouse mashing noises, entire days filled with stress, and was always ready to help in anything necessary even though she did not understand any of the projects' concepts.

Abstract

Games are a source of entertainment with the potential to offer interactive experiences on a scale not achievable in other types of mediums. Although there are many forms of games, with each offering different types of experiences, all of them share similar characteristics that allow them to be translated into different forms. This work's aim, is the transmediation of a board game created with the intent of developing computational thinking by promoting self-efficacy and collaboration, into a video game. This transmediation process aims to be faithful to the source material by maintaining the original game's overall design and goals in mind, whilst simultaneously taking advantage of the digital medium by providing content unique to this medium. Furthermore, the main goal of this paper is to serve as a case study on the types, applications, and potential of Procedural Content Generation as a game design tool. To achieve these goals, a Procedural Content Generation algorithm was incorporated into the digital version of the board game previously mentioned. The original game only provides one set of arena configuration and interactive content, the PCG algorithm aims to change both the arena and interactive content in the digital version in every game session, creating diverse gaming scenarios and increasing the replayability value of the game. Subsequently, the entire development process is detailed, from the initial planning performed to create a structured workflow, to early concepts and design ideas and their validation and finally, the actual implementation of said concepts through the Godot game engine. Gameplay tests were also made with participants in the game's target audience age range in order to evaluate the digital game and assess the quality of both the game's design in terms of visual appeal, interaction, and goals in comparison to the original one's, and the Procedural Content Generation algorithm implemented to promote its replayability.

Keywords

Play, video games, transmediation, game design, procedural content generation.

Resumo

Jogos são uma fonte de entretenimento com o potencial de oferecer experiências interativas em uma escala não alcançável em outros tipos de mídia. Embora existam diversas formas de jogos, cada uma oferecendo diferentes tipos de experiências, todos eles compartilham características semelhantes que permitem que sejam traduzidos para diferentes formas. O objetivo deste trabalho, é a transmediação de um jogo de tabuleiro criado com a intenção de desenvolver o pensamento computacional, promovendo a autoeficácia e a colaboração, num jogo de vídeo. Este processo de transmediação tem como objetivo ser fiel ao material de origem, tendo em conta o design geral e os objetivos do jogo original, simultaneamente tirando proveito da mídia digital para fornecer conteúdo único a esta mídia. Além disso, o principal objetivo deste trabalho é servir como um caso de estudo sobre os tipos, aplicações e potencial da Geração de Conteúdo Procedimental como ferramenta de design de jogos. Para alcançar esses objetivos, um algoritmo de Geração de Conteúdo Procedimental foi incorporado na versão digital do jogo de tabuleiro mencionado anteriormente. O jogo original oferece apenas um conjunto de configurações de arena e conteúdo interativo, o algoritmo de GCP tem como objetivo alterar tanto a arena quanto o conteúdo interativo na versão digital em cada sessão de jogo, criando cenários de jogo diversos e aumentando o valor de rejogabilidade do jogo. Posteriormente, todo o processo de desenvolvimento é detalhado, desde o planeamento inicial realizado para criar um fluxo de trabalho estruturado, até os conceitos iniciais e ideias de design e a sua validação, e finalmente, a implementação desses conceitos através do motor de jogo Godot. Testes de jogabilidade foram também realizados com participantes na faixa etária-alvo do jogo, a fim de avaliar o jogo digital e avaliar a qualidade do design do jogo em termos de apelo visual, interação e objetivos em comparação com o original, e o algoritmo de Geração de Conteúdo Procedimental implementado para promover sua rejogabilidade.

Palavras-Chave

Jogar, videojogos, transmediação, design de jogos, geração procedural de conteúdo.

Contents

Acronyms

AI Artificial Intelligence.

PCG Procedural Content Generation.

List of Figures

List of Tables

Chapter 1

Introduction

Games are an inevitable part of any person's life (Crawford, 2011). Whether playing or watching someone else play, everyone makes contact with some type of game almost every single day. Artifacts found all over the globe serve as proof that people from different cultures have been creating and taking part in games for centuries (Solly, 2020). Everyone longs to feel entertained, to be able to do something that provides them with joy and relieves any stress. Games have been time and time again proven to be sources that greatly satisfy these previously mentioned needs.

Even though games are predominantly used for entertainment and are usually created with entertainment in mind, they can also be used as teaching devices. A game can, without disregarding entertainment, serve as a disguised way of teaching. People in general tend to be more willing and open-minded to do something they might regard as fun rather than work. This willingness for entertainment is taken advantage of by games, in which players perform actions that, in a way, mimic real-life scenarios or academic subjects, under the pretense of an entertainment experience, providing a way for them to learn, whilst having fun. The inherent concepts of trial and error and losing versus winning games, help the players to better understand the consequences of their actions and learn from their mistakes (Pivec, Dziabenko, & Schinnerl, 2003). This immersion, however, may be increased if the game is designed in a way that is of interest to the players.

Good gameplay is one of the most important aspects of any game. If a game becomes stale people will stop playing it. If a game looks uninteresting people will not play it in the first place. If a game is fun but always offers the same experience players might get bored of the repetitiveness. For a game to be designed in a way that will make players want to play it over and over again it needs to offer a somewhat different experience every time it is played.

A technique that greatly helps designers create game experiences that feel unique, by promoting replayability and adaptability, is Procedural Content Generation (PCG) (Smith, 2017). With PCG, content that would take many hours of human labor can be done efficiently in a very short amount of time. This content can be generated before the game is finished, before the game is launched by the player, or even during gameplay.

The purpose of this thesis is to develop a case study of procedural content generation and its potential as a technique to be used in games and improve gameplay. This case study will be developed as a transmediation of a board game previously developed at the lab and its adaptation as a digital video game.

To achieve this, a rigorous study will be conducted on the main properties of game design and the origins, types, needs, and goals of procedural content generation algorithms. Additionally, there will be an attempt to adapt a single-scenario educational board game into a digital format and incorporate PCG into its design to create several different gaming scenarios, with the desired result being a better understanding and clarification of PCG as a technique as well as a functional game prototype that offers uniquely distinct replayable gaming experiences to its players whilst still preserving the goals and identity of the original game.

The report is structured as follows: Chapter 2 corresponds to the State of the Art section, where the majority of the research performed for this paper is displayed and discussed. This section includes research on the definition and clarification of Play and Games, Video Games, and Board Games, the process of transmediating a board game into a video game, approaches to game design, and a clarification of the uses of PCG and its different forms. Chapter 3 is the Objectives and Methodology section, which presents the main objectives of this paper, the process that will be followed to achieve these goals, the calendarization planned based on this process, possible risks that may occur, and mitigation plans. Chapter 4 highlights the design process of the project, from initial ideas to mockups and their validations. Chapter 5 describes the actual development of the project. This includes a description of the chosen game engine, the game's architecture, the developed PCG algorithm, the integration of the game with a Multiplayer online server, a description of all of the game's interfaces, and finally, a description of the iterative process behind the development. Chapter 6 exposes the process behind gameplay testing and its evaluation based on results gathered from sessions performed with participants from the game's target audience. Finally, in Chapter 7, a resume of what was achieved with this project is presented, as well as possible future work and what was learned from the making of the project.

Chapter 2

State of the Art

2.1 Play and Games

The concept of play has been a topic of discussion that has spanned several centuries. Play is usually associated with the idea of a game: to play is to be involved in a game. Games, which are a form of play, have been played all around the world since the beginning of human civilization and, because of this, it is commonly believed that playing is a part of human culture.

The idea that play is something that derives from culture, however, is refuted by Huizinga (1938), who argues that play does not appear with culture, it precedes it. Huizinga (1938) argues that since culture tends to be used only about humans, stating that play originated from culture is stating that play started with humans. Recurrent study of observations shows that other animals, particularly birds and mammals, also engage in play, without any human intervention. This presents the idea that humans did not create the meaning of play itself, only many new forms of play and new ways to experience play. But what constitutes play?

According to Huizinga (1938), play is an activity that is voluntarily performed by players who are aware of its detachment from ordinary life but are engrossed in it. The player's motivations for play are entirely intrinsic, with no rewards being expected or coming from outside of it. Additionally, play has its own spatial and temporal boundaries and happens under a predetermined set of rules known by each player. This definition is sometimes referred to as "the magic circle", as it describes play as an activity independent from the world outside of it.

Much like Huizinga (1938), Caillois (2001), characterizes play as being a voluntary activity that is detached from reality, constrained by a set of rules, and fundamentally unproductive since the rewards lay inside play itself. Caillois (2001) however, adds that play is uncertain, in the sense that no one is aware of how it is going to develop or end.

Garvey and Lloyd (1990), argue that most people who study play agree that play can be described by four elements. Firstly, play is enjoyable and is seen as something positive by the player. Secondly, motivation for play is fundamentally intrinsic, it derives from goals found entirely in play, not outside of it. Thirdly, play is a spontaneous and voluntary action, with players choosing to play of their own volition. Finally, play involves some sort of action performed by the players.

Salen and Zimmerman (2003) defines play as “free movement within a more rigid structure”, referring to the number of possible actions that can be performed under certain rules. One of the examples the authors use to exemplify this definition is the act of bouncing a ball against a wall. When bouncing the ball, the player interacts with the wall, the floor, gravity, the ball’s composition, and their physical ability. All of these aspects constitute the rigid structure that constrains the limits of what the players can and cannot do. Inside these constraints, however, the players are free to act as they want. Furthermore, Salen and Zimmerman (2003) divide the concept of play into three categories: play that happens during games - the one this section is focused on -, play that happens during ludic activities, and play that happens when someone is being playful. Each of these categories encompasses the one before it, with “being playful” being the broadest form of play.

The previous categorization done by Salen and Zimmerman (2003) indicates that games are not the only way of experiencing play and are, in fact, play in its simplest form. This idea is supported by Huizinga (1938), who claims that any sort of cultural activity, like dances, performances, etc., is considered play. However, although most definitions of play and games share the same attributes, it is relevant to show some definitions that are aimed directly at characterizing games.

Crawford (2011) provides a definition of games that is summarized by four qualities. A game is a complete structure situated outside of reality that has all of the required assets inside its space, without it being necessary to access anything outside of it. Games provide an unprecedented level of interaction by allowing players to explore the entire extent of its possibilities. Games provide conflicts by presenting obstacles to the players in their pursuit of a goal. Games are safe spaces that simulate the risks and conflicts of reality without real consequences.

After comparing several definitions and characterizations of the concept of play and games done by some of the authors previously mentioned as well as others, Salen and Zimmerman (2003) proposed a new definition that consolidated some of the aspects of the previous definitions and discarded the ones they deemed irrelevant. According to them “a game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome”. This definition was later used by Juul (2005), who, much like Salen and Zimmerman (2003), compared several play and game definitions to create his own by combining the aspects he thought to be more relevant and removing the ones he believed to not be truly verifiable, such as the idea that the act of playing a game is a voluntary action: “Is it not a game if social pressure forces the player to play?” (Juul, 2005). This definition is made up of six features:

- Games have rules;
- Games have variable quantifiable outcomes;
- The different outcomes of the games are valorized in different ways;

- In order to influence the outcome the players must exert effort in the game;
- Players are emotionally attached to the possible outcomes;
- The same games can be played with or without real-world consequences;

As the author puts it: "A game is a rule-based system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort to influence the outcome, the player feels emotionally attached to the outcome, and the consequences of the activity are negotiable" (Juul, 2005).

Although Salen and Zimmerman (2003) and Juul (2005) definitions are the two most quoted game definitions in the context of video games, they do, according to Aarseth and Calleja (2015), present some flaws. The authors claim that even though both Salen and Zimmerman (2003) and Juul (2005) mention that games that do not fit their definition are simply exceptions to the norm, the number of games that do not follow these definitions is increasing and so new definitions might be needed. Games like The Elder Scrolls V: Skyrim (Bethesda Softworks, 2011), do not have an explicit end so a quantifiable outcome can not be derived from it. Likewise, narrative-driven games like The Last of Us (Naughty Dog, 2013), might have an ending, but it is usually not quantifiable. Aarseth and Calleja (2015) also mention that some games have different sets of rules, depending on what is being played. Titanfall (Electronic Arts, 2014) for once, is a game that offers several different multiplayer modes to its players with each one of these modes having distinct objectives and rules. However, although these rules are different and regardless of the mode that is being played, the game is still Titanfall.

Despite this, these definitions help one understand the concepts of both play and games, however, much like play, games are not of a singular form. Even though most forms of games can be defined by the previous qualities, there are several genres of this media, with each one having unique attributes. In the context of this thesis, in particular, the two genres of games that are of relevance are video games and board games.

2.1.1 Video Games

A video game - also known as a digital game - is a type of game that relies on electronic equipment for visual output and interaction. It follows the same characteristics traditional games do, but for players to interact with it they must rely on a screen where the game environment is displayed. The screens can be of different types, qualities, or sizes (computer screen, television, phone, handheld console, smart watches), as long as they can display the game content. For players to interact with what is shown on the screen they also need some electronic device that allows them to control their actions in the game. These devices, often referred to as controllers, are usually specific not to the game itself but to the system where the game is played.

Salen and Zimmerman (2003) enumerate four traits that, though not exclusive to video games, are more prevalent in this medium:

- **Immediate but Narrow Interactivity** - Video games can react immediately to players' actions and provide instant feedback on their behaviors. The amount of interaction players have with video games, however, is very limited. Their interaction is constrained by the controller that is being used, which usually only allows players to interact with the game through very simple movements like pressing a button. Whilst in a game of tennis players have to use their entire body, in a tennis video game that does not use motion controllers players only use their wrists and fingers.

- **Information Manipulation** - Video games can store and display images, videos, audio, text, and other types of data to enrich the game environment in ways not possible to most of the other gaming media. Most video games also present information directly or indirectly to the players in ways that help them understand how to interact with its content. This allows players to learn how to play the game as narrative-driven instead of having to know the rules prior to starting to play, hence why most video games' first level serves as a tutorial where the game teaches the players the controls and how to navigate through its environment. Battlefield 4 (Electronic Arts, 2013) first level, for example, teaches players how to look around, sprint, jump, crouch, reload and fire their guns, open doors, throw grenades, change equipment, change firing mode, use the tactical binoculars and spot enemy units. Although this may seem like a lot of information to be learned in one go, the game does a good job of building the level in a way that helps the players get used to all of these actions.

  Additionally, video games also allow information to be hidden and displayed in certain ways or only at specific moments. The fog of war in League of Legends (Riot Games, 2009) for example, limits the player's vision of the map and only shows non-allied units when the players or one of their allied units is in direct contact with them. This adds to the game a feeling of tension for not knowing where the enemy is and what he is up to, as can be observed in the following image.

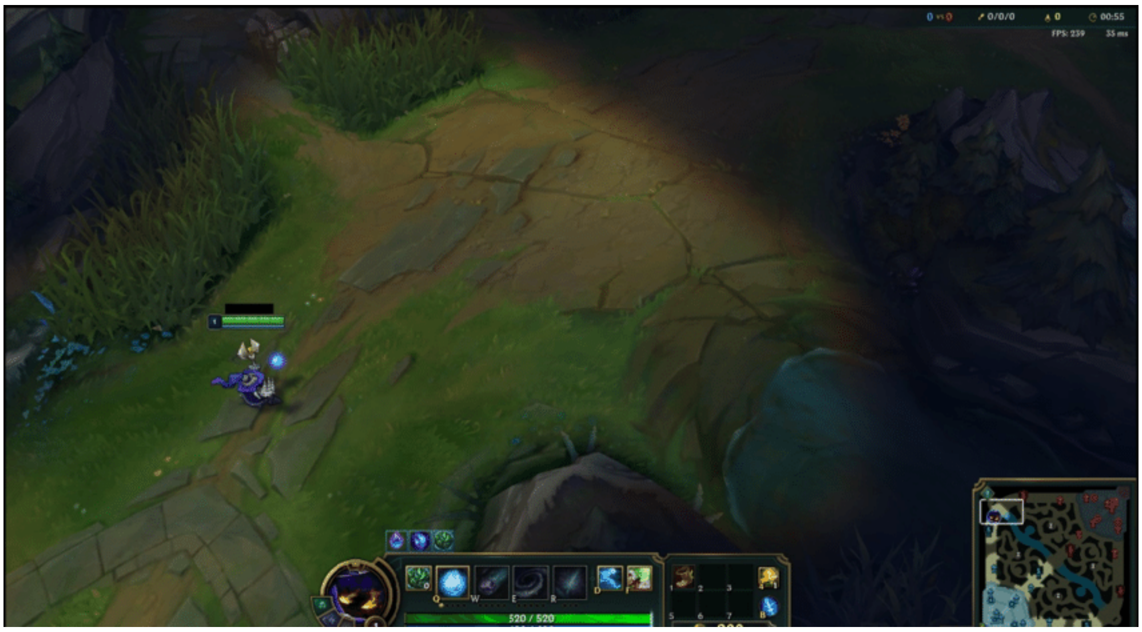


Figure 2.1: League of Legends' fog of war (Ekdahl, 2021)

- **Automated Complex Systems** - Video games, like most digital systems, can automate procedures that could otherwise be too complicated or too tedious for a player to perform by himself. Role-playing video games are a great example of video games that manage information for the players. Mass Effect (Electronic Arts, 2007) for example, calculates and stores the player's character level, status, reputation, relationships, and equipment. In most non-digital games of the same nature, this type of data would have to be saved and calculated by the players themselves.

  Video games can also change the game environment in response to the player's actions, such as removing a defeated enemy unit or placing a flag that symbolizes who owns a certain territory, instead of forcing the player to do it. This automation allows video games the ability to perform and give the players more complicated tasks than in board games. However, video game automation has a downside. Unlike non-digital games, which present all of their rules and ways the players and pieces must interact with the players, video games usually hide their inner workings, which, to some players, might diminish their experience.

- **Networked Communication** - Some video games allow players to communicate more easily over long distances. Since video games can display different types of information, this communication can be done in many different ways, such as text, voice, emoticons, or even the way players play the game itself.

As previously mentioned, these characteristics are not exclusive to video games but are usually more prevalent in this media. Additionally, a video game doesn't need to present all of these qualities, as they do not represent a definition for the concept of video games, but rather a list of attributes they tend to have. Thanks to these and other qualities, video games have skyrocketed in popularity to become one of the most popular forms of entertainment. However, although more popular than ever, video games have been around and entertaining the public for more than half a century. Tennis for Two (Higinbotham, 1958) and Spacewar! (Russell, Graetz, Saunders, & Piner, 1962), are two of the very first video games created for entertainment purposes.

Tennis for Two (Higinbotham, 1958) was created in 1958 to be exposed to guests during a Brookhaven National Laboratories three-day exposition. The game was made up of a vertical line, which represented a tennis net, a horizontal line, which represented a tennis court, and a ball, which would be controlled by the players using a controller made up of a button and a knob. Pressing the button would hit the ball and twisting the knob would control the angle of the shot. Tennis for Two proved to be so popular during the exposition that visitors would line up in massive lines to have the chance to play it for a brief moment (Stony Brook University, 2013).

The following depicts a game session of Tennis for Two.

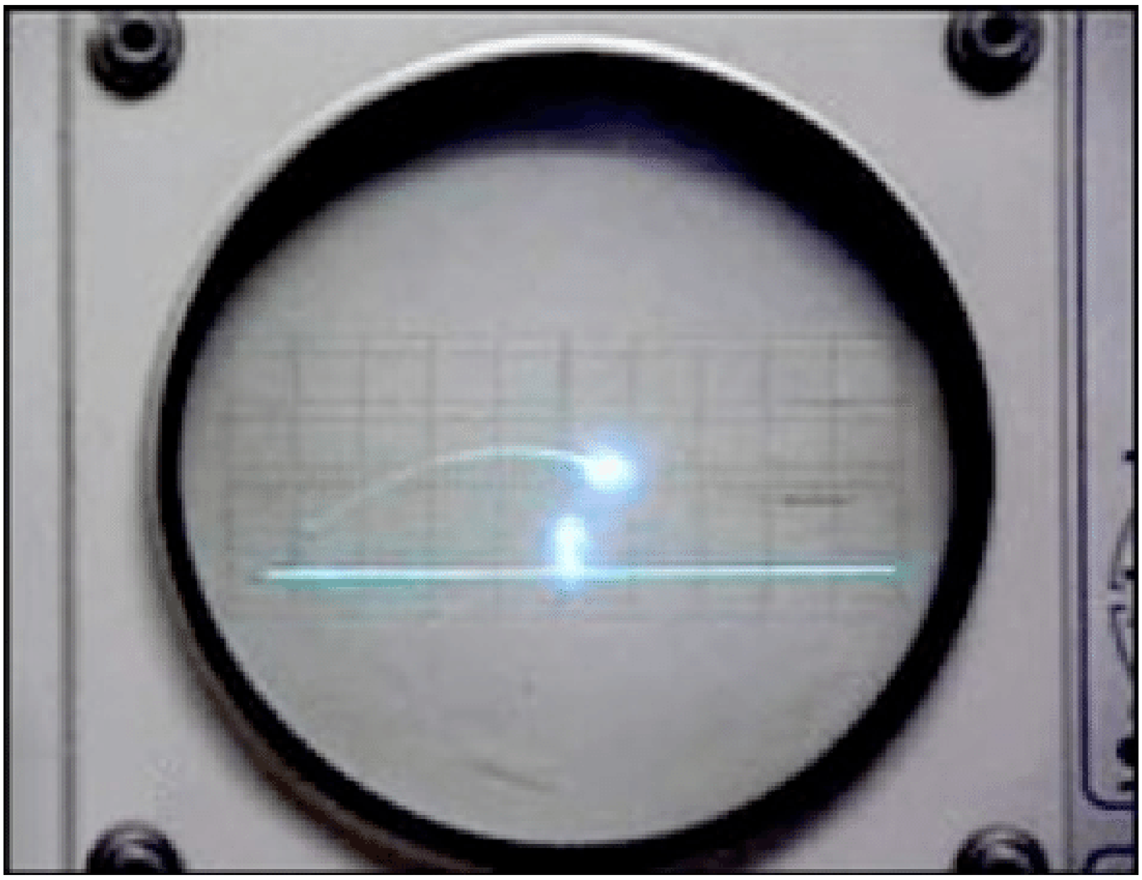


Figure 2.2: Tennis for Two (Sweidan & Darabkh, 2018)

Spacewar! (Russell et al., 1962) was a program developed in 1962 at MIT to show the capabilities of the PDP-1 minicomputer (Graetz, 1981; Monnens & Goldberg, 2015). It consisted of two spaceships, each controlled by a different player, fighting each other around a star that offered a gravitational pull positioned at the center of the screen in a 2D perspective. The spaceships would be destroyed if they collided with each other, the star, or one of the other ship's torpedoes.

The following depicts a game session of Spacewar!.

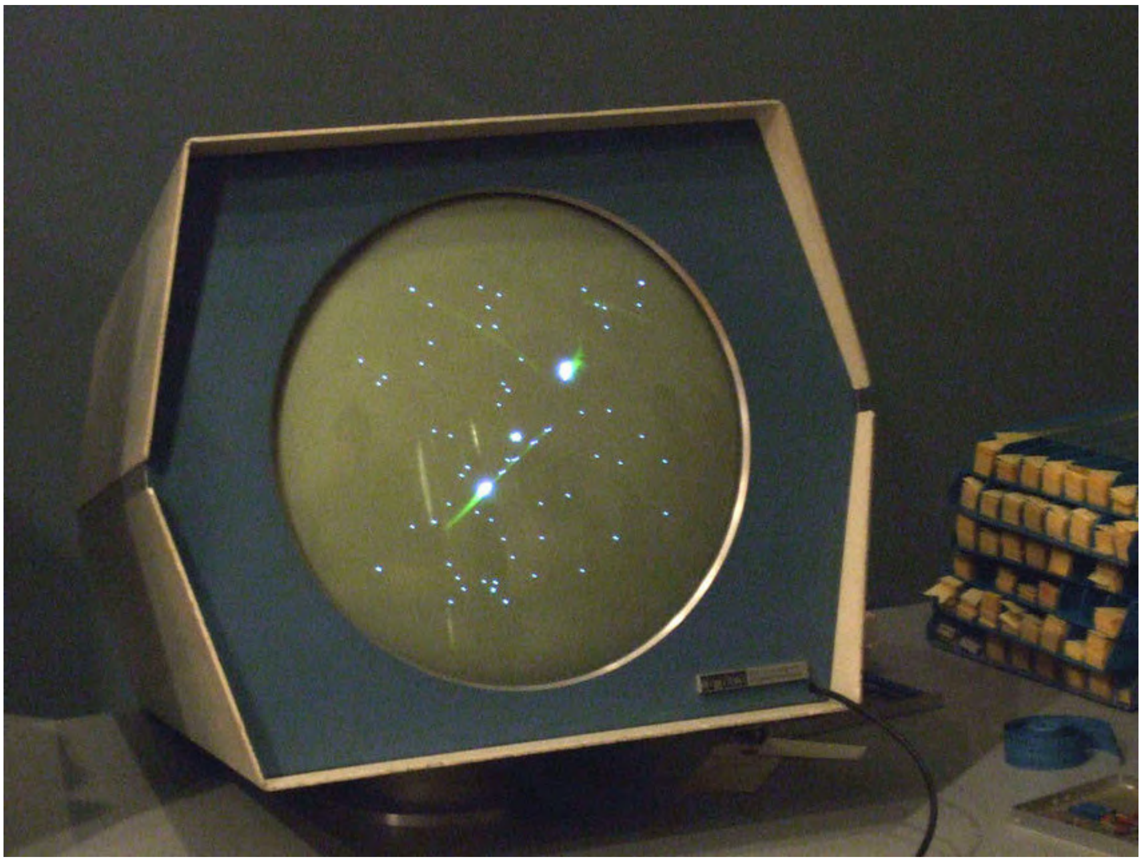


Figure 2.3: Spacewar! (Engström, 2020)

Even though several other video games would eventually be released after Tennis for Two (Higinbotham, 1958) and Spacewar! (Russell et al., 1962), it would not be until a decade later, with the release of Pong (Atari, 1972), that a video game would grab wild-scale attention from the public, launching the video game market into what it would eventually be today (Lowood, 2009).

With the public interest and computer technology growing, so did the video game space. More demand from the public led companies to develop a higher number and variety of video games and the advances in technology allowed them to create increasingly greater experiences.

2.1.2 Board Games

A board game is a type of game, constrained by a set of particular rules, that offers a tangible playing surface in which the players use objects with specific purposes to interact with it (Barbara, 2015). Unlike most video games, board games focus more on the ease of interaction between players than that of the players with the system, by requiring that all players be present in the same place at the same time.

Additionally, most, though not all, board games are intended to be played by at least two players, usually pitting them against each other.

The origins of board games are a debating point between historical experts, with evidence dating several centuries being found all over the globe. The discussion stems from the high degree of difficulty in extrapolating the exact historical date of the findings as well as their true purpose. In some areas of Africa, Arabia, and the Middle East there have been found stone carvings dating back between 7000 B.C. and 9000 B.C. (Park, 2021). These carvings are formed in rows that resemble the modern board game Mancala, with some believing it to be a predecessor to this game, while others claim that these holes are a sort of analog calculator. According to Park (2021) however, Irving Finkel, an expert on Mesopotamian culture at the British Museum, claims similar carvings have been found in several houses during an excavation in Israel and that this most likely indicates that the carvings were used for entertainment and not for calculations, although this is not known for certain.

One of the first known board games ever created is the game Senet. According to archeological evidence, Senet was being played as long as 5000 years ago in Egypt (Solly, 2020). Although time eventually forgot this game, with its exact rules being open to interpretation by professional Egyptologists, it is known for a fact that the board of the game was constituted by a grid of thirty squares laid out in three rows of ten squares each.

The following is an image showing the board of the game Senet.

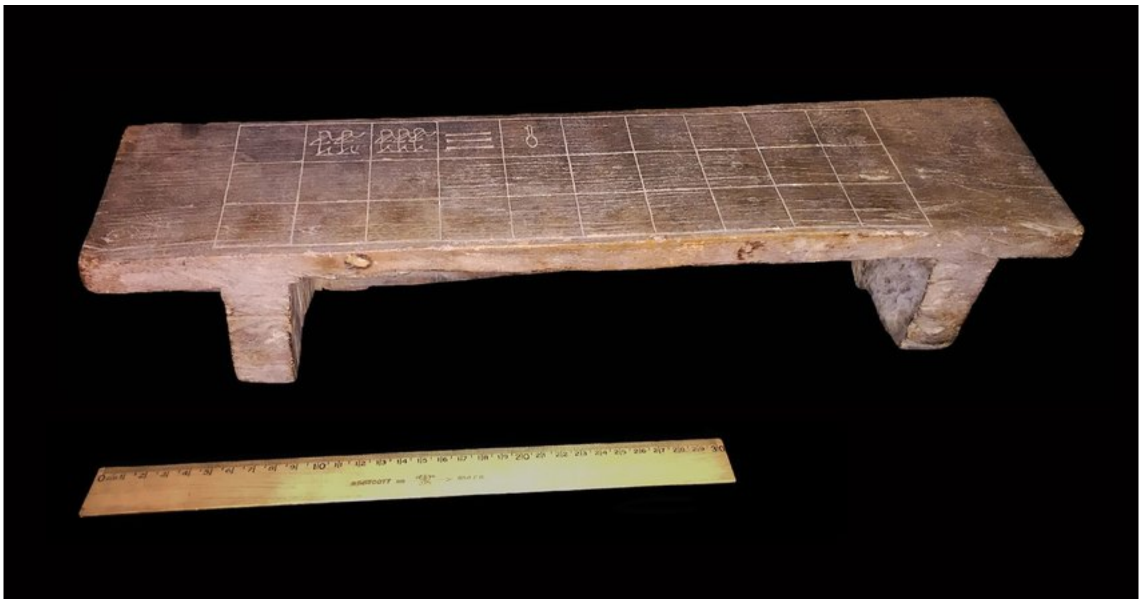


Figure 2.4: Senet board(Crist, 2020)

Some board games, however, have withstood the test of time and have been played with little to no changes to their rules, boards, or pieces for centuries. According to Kraaijeveld (2000), Chess originated sometime between 200 BC and the early 7th century. Stahlhacke (2003) mentions that Nine Men's Morris has been a popular game as far back as the 14th century, with iterations with less than nine pieces being found dating back to 1400 B.C. Shotwell (1994) goes into great detail on the many different instances in time in which the game Go was created, with most of these references dating back to sometime between 1000 BC and 2000 B.C. which would make it the oldest continuously played board game in history.

2.1.3 Challenges and Opportunities in Game Transmediation

With the increasing popularity of video games the creation of games in this medium that are either inspired or adapted from board games has also become a growing trend. Asmodee for example, one of the world leaders in board game development and publication, with 43 million game units sold according to their website (*Asmodee*, 2022a), has been publishing board games as well as video game adaptations of board games since 1995. Some of the most successful digital board games they have published include the likes of Scythe: Digital Edition (Asmodee & The Knights of Unity, 2018), A Game Of Thrones: The Board Game Digital Edition (Asmodee, 2022b), and Catan (Asmodee & Dovetail Games, 2019). As is the case in any transmediation process, however, adapting a board game into a video game has its share of challenges and benefits.

In 2018, Philippe Dao - Asmodee Digital's Chief Commercial and Marketing Officer -, hosted a panel presentation in Casual Connect Europe about the challenges of adapting a board game into the digital medium (GameDaily Connect, 2018). Two years later in 2020, Piotr Sobolewski, CEO of The Knights of Unity, Jacek Iwanicki, developer at The Knights of Unity, Jamey Stegmaier, CEO of Stonemaier, and Yann Corno, CTO of Asmodee Digital, discuss the process of adapting a board game into a digital form (Steam, 2020). Both of these debates (GameDaily Connect, 2018; Steam, 2020) allow for some key takeaways about the process of board game transmediation into the video game medium through the lenses of some of the most experienced actors in the field. From the information provided, it is possible to have a better understanding of the challenges that come from adapting a board game into its video game counterpart, some possible benefits the video game version has over the board game version, and some practices that help ensure a good transmediation from board game to video game.

The main challenges mentioned were the following:

- **Screen size constraints** - Dao, Corno, and Sobolewski mention that one of the biggest challenges that come from adapting a board game to the digital medium is the limited size of screens (GameDaily Connect, 2018; Steam, 2020). Complex board games usually offer a big surface area with lots of information, which is hard to translate into screens of smaller dimensions. This becomes an even greater task if the video game is supposed to be playable on different-sized screens, like televisions, computers, and mobile phones.
- **User interface** - Dao states that because of the screen size constraints, the user interface is tricky to achieve (GameDaily Connect, 2018). A board

game that has dice, cards, pawns, and other objects, needs to have all of these items translated into the digital version in a way that is clear and accessible to the player. Sobolewski also mentions that the digital version of the board game needs to show the rules of the board game, which in the original version usually comes in a rule book, in a way that is integrated into the gameplay itself, instead of just listing the rules (Steam, 2020). Additionally. According to Iwanicki, to create the user interface the designers need to learn how to play the original board game to be able to access and prioritize the information that needs to be available to the players at each time. Actions that are more common than others should have better or at least equal accessibility than others. Iwanicki also mentions that this information can be hidden in menus that expand when needed to minimize the space it occupies on the screen (Steam, 2020).

- **Player interaction** - Corno mentions that when playing a board game "things happen on the table", referring to the player's interaction with the game, "and above the table", which refers to players interacting with each other (Steam, 2020). Since to play a board game all players need to be present in the same place, this allows players to better express themselves with each other, whilst simultaneously interacting with the game in a manner that few video game platforms can mimic. Preserving these two types of interactions in the digital medium can prove to be a challenge, especially if the video game adaptation can be played online, where the relay of information between players needs to be mediated through some sort of voice, text, or another type of communication channel.

The main benefits the video game adaptation has over the board game are:

- **Not starting from scratch** - According to Dao, since the video game is an adaptation, the developers already have a reference of what they need to build. There is no production phase, the developers' work is focused on creating a user interface for the video game, not on creating rules, stories, game objects, etc., which are already established in the original version. Additionally, adapting a board game that already has a fanbase means that the digital version will, most likely, appeal to the fans of the original version, which facilitates its market entry (GameDaily Connect, 2018).
- **Data management** - As stated by Sobolewski, since a video game runs on machine software, it is capable of managing data that would usually be managed by the players, such as keeping track of resources (Steam, 2020).
- **Play testing** - Similar to the previous point, the video game software is also capable of storing data pertaining to players' choices and gameplay statistics that can be later reviewed by the developers to improve the game. Stegmaier mentions that play testing for the original version of Scythe (Stonemaier Games, 2016) had yielded results for around one thousand game sessions, whilst with Scythe: Digital Edition (Asmodee & The Knights of Unity, 2018) they were able to get data from tens of thousands of game sessions, which further helped the developers to learn which combinations of actions

in the game proved to be too overpowered and had to be removed (Steam, 2020).

- **Additional media** - Stegmaier also mentions that a video game adaptation allows for new media to be added, such as music and sound effects. This addition, which is hard to achieve in a board game, can help elevate the video game environment and its world (Steam, 2020).

- **Aid players' actions** - The video game can aid the gameplay by showing hints to the players about what actions they can and can not perform. Iwanicki, in particular, gives the example of being able to highlight what the player can do (Steam, 2020). This type of tool can be very helpful to players in cases where they are not sure about all the actions they can perform at a specific point in the game or in cases where they are stuck and do not know what they are supposed to do.

The following are some of the key practices mentioned by Dao that help ensure the quality of a video game adaptation of a board game (GameDaily Connect, 2018):

- **Create exclusive content adapted to digital player's behaviors and needs** - Video games allow for a multitude of content that cannot be found in most other types of medium. Although it is important to retain what makes the original game unique without deviating from it, a video game has the potential to add new content that improves the board game's experience, like background music, sounds, and animations. Additionally, it may be necessary to add content that helps guide the players through the game, like tutorials and hints.

- **Do not copy the analog game** - As previously mentioned, it is important to try and bring to the digital game the interactions and feelings that made the board game unique, however, board games and video games are different mediums that represent content and offer interaction in different ways. The video game should therefore try to offer the same experiences as the board game, but change the way these are achieved by working with the constraints and benefits that the digital medium has over the board game.

- **Do not complexify the flows, the UI, etc.** - As mentioned above, screen size constraints need to be taken into consideration when transmediating the board game. The way information is shown to the players should be as simple and clear as possible. Likewise, players should be able to access all of the game's available functionalities as easily and seamlessly as possible.

- **Do not neglect the Artificial Intelligence (AI) in solo mode** - If the game is a solo mode where players play with or against the computer, its AI needs to be developed enough to offer engaging gameplay. If a player is playing against the AI it can't be too weak nor too strong, otherwise, the challenge it provides can break the player's experience. Likewise, if the player is playing cooperatively with an AI, it can't be too smart to the point of fixing every problem for the player nor too dumb to the point where the player would rather play alone.

- **Do not neglect first-time user experience and tutorials** - The video game should provide easy access for first-time players either by making the gameplay itself simple enough to understand on a first try or by making tutorials where the game teaches the players the fundamentals as the game progresses.

2.2 Game Design

Salen and Zimmerman (2003), defines design as "the process by which a designer creates a context to be encountered by a participant, from which meaning emerges". They believe that one of the most important parts of game design is the understanding and creation of "meaningful play" and it is around this concept that their approach to game design is formed. Salen and Zimmerman (2003) define this concept of meaningful play in two different views:

- **Descriptive meaningful play** - In a game, players interact through actions that, in turn, result in outcomes provided by it. Meaningful play is created through the relationship between the actions of the player and the outcomes of the game. This definition is therefore centered on how meaning is generated through the narrative nexus of causality or player influence over the outcome.
- **Evaluative meaningful play** - Meaningful play arises when the aforementioned action-outcome relationship is "both discernable and integrated into the larger context of the game". Discernable refers to the feedback provided by the game when a player executes an action. For there to be meaning the game needs to show the player that their action did indeed have an outcome, otherwise, the player might think that their action had no real consequence or not be made aware of what the consequence was. Integrated, on the other hand, refers to the fact that the relationships between the actions and outcomes need to affect the game in the long run, in a way that the actions performed by the players not only change the game immediately but also change the way the game will evolve from there on out.

To create meaningful play, it is necessary to understand both of these definitions. Whilst the first one influences how meaning is created, the second one explains how meaning can achieve different degrees of quality, which is what differentiates meaningful play in different games.

This emphasis on meaningful play originates from the need game designers have to understand the concept of meaning, due to their involvement in the creation of systems of interaction - it is the designer's job to create a context that triggers the production of meaning in the players when interacted by them. Because of this, designers have turned to several different fields to seek aid in this task. One of these fields is the field of Semiotics, which is the study of signs and their utilization and interpretation or, in other words, “the study of how meanings are made” (Salen & Zimmerman, 2003).

2.2.1 Signs

By dissecting semiotician Charles Peirce’s definition of sign: "something that stands for something, to somebody, in some respect or capacity.", Salen and Zimmerman (2003) found that it provided four ideas by which this concept is comprised:

- **A sign represents something other than itself** - Games use signs to represent action and outcome as well as elements of the game (Salen & Zimmerman, 2003). As an example of the first case, in the game of Tag, if a catcher touches another player with their hand it results in the outcome of the “tagged” player becoming a catcher. In this case, the simple sign of a catcher touching another player with their hand represents a shift in the roles of the players. Another example is in the game Dark Souls (FromSoftware Inc., 2008), the game uses signs that represent enemies, health, stamina, chests, and many others. As is common to many games, even though signs reference real-life objects, their meaning in the game is derived from their context in the game. A bonfire in real life may mean warmth and light, whilst in Dark Souls it means something that can restore a player’s health or serve as a checkpoint in the case of death.
- **Signs are interpreted** - Since people are the ones giving signs meaning, this meaning is influenced by agreed-upon conventions, whether they come from the person’s cultural background or are made up on the spot. Salen and Zimmerman (2003) give once more the example of the game of Tag. In some derivations of this game, the players agree on a “home base” that serves as an area where players are immune to the catchers’ “tag”. This base can be anything the players agree upon and is interpreted by them as being a safe zone.
- **Meaning results when a sign is interpreted** - For meaning to be taken from a sign by an individual he must first interpret that sign. If in a game of Tag, a catcher touches another player with their feet the player may ask the catcher what they meant by that action. If the catcher says they tagged them the player may either disagree with them by claiming that a tag has to be done by hand and, therefore, the catcher performed a sign with no value or meaning to the game, or the player might accept it, in which case a new meaningful sign is added to the game.
- **Context shapes interpretation** - The same signs can be used in different circumstances and their interpretation may change depending on the context. In League of Legends (Riot Games, 2009), players can use emoticons and pings as a means of communicating with other players. One of these pings is the “Enemy Missing” ping, which is shaped like a question mark and is

often used by players to indicate areas in the map where an enemy player went missing. This ping, however, has also been used by players to indicate confusion or even disdain for another player's actions. A player might decide to use this ping on top of the area where a friendly player is to show that they did not understand why the player acted the way they did. This community practice influences how the ping is therefore interpreted by the player's team, according to the place where it is placed as well as the events that happened recently in that area.

These four concepts have the potential to help designers understand what type of meaning, if any, the players may give to objects when interacting with them and shape the game design accordingly.

2.2.2 Games as a System

Another idea that Salen and Zimmerman (2003) present that may aid in the process of game design is to perceive games as "a set of parts that relate to form a whole": a system. As a system, a game is made up of four core elements: objects - the elements that constitute the system -, their attributes and those of the system itself, the internal relationships between them, and finally, the environment. These elements, however, vary depending on the way the system is framed.

According to Salen and Zimmerman (2003), if a game is viewed as a system it can be framed in three distinct levels. To frame a game as a formal system is to take into account its rules and logic. To frame it as an experiential system is to take into account the previously mentioned factors as well as the interaction between the players and the game. Finally, to frame it as a cultural system is to understand how it integrates into a culture, which takes into account the factors of the two previous system levels. All three of these framings are always present in any game, however, Salen and Zimmerman (2003) state that focusing on one, in particular, may help build certain parts of the design. For this to happen, however, the designers must have a high understanding of how all three of the levels interact with each other.

2.2.3 Interactivity

For players to engage in these systems in a way that allows them to perform actions that result in outcomes that provide meaningful play they need to be able to interact with them. Interactivity is therefore the concept that allows players to play the games and make them move forward. Salen and Zimmerman (2003) present four modes of interactivity that a person might experience in a system. These modes are not mutually exclusive and most systems that focus on interactivity use multiple or even all of these modes simultaneously.

- **Cognitive interactivity** - The cognitive interaction that happens between the player and the representation system.

- **Functional interactivity** - The interaction that allows players to discern the quality of the physical and the virtual functionalities of the computational system.
- **Explicit interactivity** - The direct interaction between the players and the game system. This includes following the rules of the game, making choices during the gameplay, experiencing the game events, etc.
- **Beyond-the-object-interactivity** - Interaction with the culture surrounding the system.

The third mode - explicit interactivity - is the one that better describes interactivity in a game system. However, for an experience to be considered interactive, the choices made by the players need to be designed into the actual computational and representation system. This is what constitutes complete designed interactivity. If a player chooses to do an action that was not expected during the object design then the game might not have a relevant expression or outcome for it and, therefore, the action has no meaning and meaningful play is not achieved.

In games, choices can happen at two levels. Micro level refers to the small-scale choices that the player needs to make during a game, whilst macro level is the way the micro-level choices come together to impact the game in the greater picture. The micro-level choices of deciding what pieces to move and where to move them during Chess will influence the macro-level choice of what strategy to perform once those pieces are in position. Macro-level choices are the ones that dictate if the player wants to play the game, and, if so, how he plays the game.

Furthermore, the process of choice can be subdivided into five stages:

1. **What happened before the player was given the choice?**
2. **How is the possibility of choice conveyed to the player?**
3. **How did the player make the choice?**
4. **What is the result of the choice? How will it affect future choices?**
5. **How is the result of the choice conveyed to the player?**

These five steps are what constitute an action-outcome relationship and knowing how to answer each one of their questions can help designers to understand flaws and improve aspects of their interactivity system.

All of the previous terms and definitions mentioned in this section are what constitute the "space of possibility" (Salen & Zimmerman, 2003), which is all the possible actions and outcomes that might occur in a game thanks to the way it is designed. This term aggregates all that makes up Salen and Zimmerman (2003) approach to game design: "The space of possibility is designed (it is a constructed space, a context), it generates meaning (it is the space of all possible meanings), it is a system (it is a space implied by the way elements of the system can relate to each other), and it is interactive (it is through the interactive functioning of the system that the space is navigated and explored)".

2.2.4 Participation Centered Game Design

Based on ideas that follow Salen and Zimmerman (2003) definition and approach to game design and with the intention to help designers understand how the players interact with their game, Pereira and Roque (2013) developed a model focused on the perspective of players' participation to aid in the design process of a game. This model takes into account six different perspectives on participation to "assist the designer in thinking, in a comprehensive manner, about the range of possibilities at [their] disposal to define or give a certain character to a game." (Pereira & Roque, 2013):

- **Playfulness** - The game provides flexibility to be interpreted by the players in ways that encourage them to explore and improvise. According to Pereira and Roque (2013), this flexibility can be provided by many of the game's elements, like custom character creation and development, world exploration, how to interact with the game resources, etc.
- **Challenge** - The game is a source of incentives to achieve certain goals that promote the creation of strategies and the improvement of skills. The aspects of the game that drive this perspective are the elements that make up a challenge, like the challenges themselves, their difficulty, the frequency in which they occur, their outcomes, etc.
- **Embodiment** - The game serves "as a context for physical performance" (Pereira & Roque, 2013) where the player participates in it through the physical relationship established between the player and the video game.
- **Sociability** - The game is a "context for legitimizing forms of interaction between players" (Pereira & Roque, 2013), enhancing activities related to socialization, such as competition and cooperation.
- **Sensemaking** - The game is a "means of expression" (Pereira & Roque, 2013), where players participate in it by interpreting and acting according to the semantics the game presents.
- **Sensoriality** - The game serves as a "source of stimulation for the senses" (Pereira & Roque, 2013), where the player's participation is formed by their process of engaging, understanding, and acting upon stimuli.

Each one of these participation perspectives happens on three different levels:

- **Intention** - The participation ideal that is suggested by the game.
- **Artifact** - The way the game provides the participation ideal.
- **Participation** - The aspects that describe or quantify the participation ideal.

	Intention	**Artifact**	**Participation**
Playfulness	exploring, discovering, recreating, customizing	the nature of a player's agency, the variety of interactive elements of the game (objects, characters, actions, etc.)	degree, variety, and tendency of exploration
Challenge	overcoming a challenge, creating a strategy, defeating an opponent, mastering a skill	nature of challenges proposed, type of penalties and rewards, intensity and organization of challenges	control, pace, progress, efficiency in performing tasks
Embodiment	physical involvement, physical performance	representation of the physical game world, player's representation on the game world, interpretation of player's movement	control and rhythm of movement, aesthetics of the movement
Sensemaking	interpretation of a role, fantasy, self-expression	theme and underlying narratives, models and representations of phenomena, roles, and motives, significant actions	alignment between actions and roles, understanding and or critique of the represented phenomenon
Sensoriality	contemplation, wonder	style, nature of the stimuli, visual and sonic compositions, synesthetic explorations	degree of exposure and responsiveness to stimuli, interaction or engagement with sources
Sociability	competition, cooperation, friendship, identification, recognition	diversity and nature of social interactions and relationships, models of social structures (team, hierarchy, etc)	the intensity and types of interactions between players, effectiveness bonds

Table 2.1: Participation Centered Game Design Model (Pereira & Roque, 2013)

2.3 Procedural Content Generation

Besides the basic tweaks that come from adapting a board game into a digital format, the major desired difference between the original game and the digital prototype will be the inclusion of procedural content generation in the latest to create different scenarios every time the game is played, with the intent to promote its replayability.

2.3.1 What is Procedural Content Generation?

In the context of games, content can be categorized as every piece of audio-visual information (other than the players themselves) as well as every functionality (rules that dictate how the game operates) that exists in the game.

Procedural Content Generation (PCG), is a technique used by game designers to aid in the creation of content in games. More specifically, "Procedural content generation PCG) is the process of using an AI system to author aspects of a game that a human designer would typically be responsible for creating" (Smith, 2017). This means that an algorithm that is designed to follow some specific rules of content generation is used to replace a human designer, either partially or entirely, in the creation of gaming content, hence the procedural in PCG. This technique can be utilized to create almost every type of said content, "from textures and natural effects to levels and quests, and even to the game rules themselves" (Smith, 2017).

Although mainly used in video games, PCG is also used for other types of media, such as board games. The famous tabletop roleplaying game Dungeons & Dragons (Tactical Studies Rules, 1974), for once, had a PCG program built for it. The AD&D Dungeon master's assistant (Strategic Simulations Inc, 1998), was a program that assisted the Dungeon Master (the person responsible for the flow of the game) with tasks like dice rolling and random enemy encounters.

2.3.2 Applications of Procedural Content Generation

One of the first motivations for the creation of PCG algorithms for video games was data compression. Due to machine memory limitations, it is hard to store the data of large gaming worlds, with this fact being especially true in the 1980s when machines had severely fewer capabilities than the ones available today. One of the solutions found to circumvent this issue was the creation of PCG algorithms that utilized seeds to create gaming worlds. These seeds, which are short numbers or text strings, represent a starting point for the PCG algorithm, forcing him to create the same content every time the same seed is used. By doing this, instead of having to store all the information of the game world, all there is needed to do is to store a seed that represents the said game world. One of the first implementations of this type of algorithm was in the game Elite (Braben & Bell, 1984), which contains 8 different seeds, each one capable of creating a universe with 256 planets. For every planet, its composition, location, and other details are created using procedural content generation, making each galaxy different from the others.

As briefly mentioned above, PCG can be used to create bigger gaming worlds. Since PCG is algorithmic and uses computer power to generate content, it can do so immensely faster than any human designer or even entire teams of designers. The game Elite (Braben & Bell, 1984), as previously mentioned, has 8 galaxies of 256 planets, adding up to a total of 2048 planets, which is unfeasible for any team of designers, regardless of size or qualification, to handcraft. With PCG however, it is possible to create this and even more amounts of content in a timely fashion. The list of games in which PCG was used to create massive worlds is extensive, with one of its most well-known entrances being Minecraft (Mojang Studios, 2011), in which a PCG algorithm is used to create worlds of 60 million meters in diameter. To put this size in perspective, a YouTuber that started a Minecraft world with the sole purpose of walking 30 million meters from the spawn point to the world's edge took 2500 hours to complete this feat (Filby, 2022). Another recent example of PCG being used for this purpose was done by the small team of 26 at Hello Games. This studio created the game No Man's Sky (Hello Games, 2016), in which players enter a game in a Universe with 18 quintillion planets, with the theoretical possibility of being able to visit all of them. The fact that such a small team can build a game that has, for all intents and purposes, infinite content, is a testament to the power of PCG algorithms.

The following image shows a portion of the galaxy generated in the game, where each dot represents an entire planet the players can go to and explore.

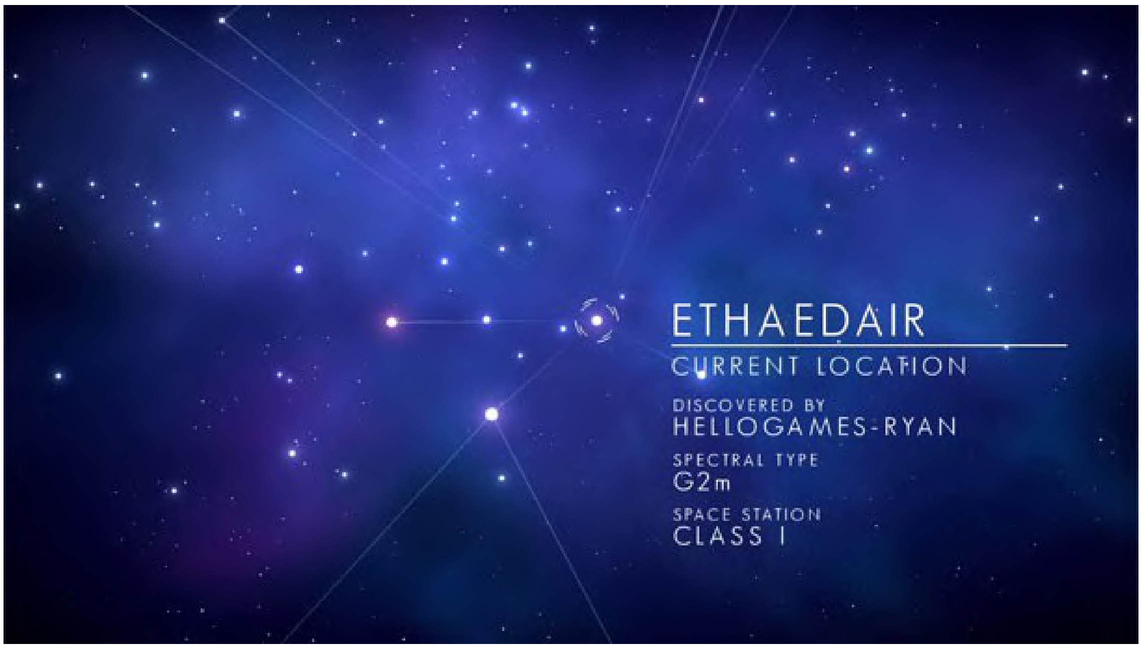


Figure 2.5: No Man's Sky galaxy (*Galactic Map Puts Scale Of No Man's Sky In Perspective*, 2014)

Another aspect that PCG can help with is variety. One of the main concerns when creating a game is ensuring its replayability value. This is done by implementing functionalities that allow the player to have different experiences with the game most or every time they play it. PCG can do this while the game is launching as mentioned in the game Elite (Braben & Bell, 1984), where the game runs a PCG algorithm with a certain seed to create a world. The more seeds the game has the more worlds it can generate, ensuring that every time a player launches a game to create a new world the world is in fact, new to the player. A more extreme example of this happens in the aforementioned Minecraft (Mojang Studios, 2011), in which the 60 million meters in diameter worlds are a result of a PCG algorithm choosing between more than 18 quintillion different number seeds. PCG can also be used whilst the game is being played, like in the case of Left 4 Dead (Valve Corporation, 2008), in which the game has no fixed spawn points for enemies, instead, an AI called The Director, chooses during playtime where to spawn the enemy infected as well as which type of special enemy infected, even deciding what gear the players will find along their gameplay, which makes for a different experience every time the same level is played.

PCG can also be used as a means to reduce the costs of game production. As already stated, PCG algorithms can replace human designers in the process of content creation, this replacement means that game companies can invest less in human labor, using algorithms instead of having to pay for large amounts of designers. This is especially useful in the creation of massive AAA games, where the budgets tend to be in the dozens if not hundreds of millions of euros, and any way of saving money without compromising the game's quality is a welcome one.

The following image depicts how The Director modulates the population in the game.

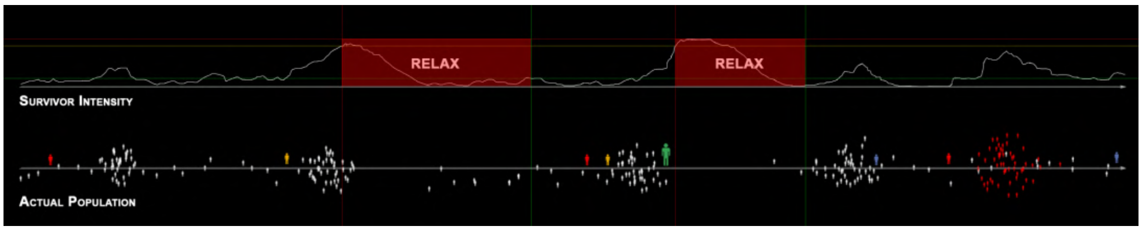


Figure 2.6: Left 4 Dead's The Director modulating population (Booth, 2009)

2.3.3 Challenges of Procedural Content Generation

Although PCG is a powerful tool that can greatly help designers build games, it does not come without its flaws.

Togelius et al. (2013) enumerate eight challenges derived from PCG implementation.

- **Non-generic, Original Content** - Content generated by PCG methods is usually very generic, especially when compared to handcrafted content. This generic content lacks in art and players can frequently tell when something is created by a machine or by hand. However, some exceptions do exist, such as Galactic Arms Race (Evolutionary Games, 2010), which can create unique and varied types of weapons.

"The challenge, then, is to create content generators that can generate content that is purposeful, coherent, original and creative" (Togelius et al., 2013).

- **Representing Style** - The authors mention that developing a PCG method that is able to create content following a certain style is very challenging. The method needs to be fed with enough information describing a specific type of style in order for it to generate content in a similar fashion.
- **General Content Generators** - Most content generators are focused on a single aspect of content generation. The challenge derives from creating a general PCG method that is able to create content of different types in different areas. This is especially important for this project since the goal is to develop a PCG algorithm that can change both the arena configuration and the interactive content hidden in the arena.
- **Search Space Construction** - This challenge derives from defining the search space in which a certain type of content can be modified, i.e., the type of changes that can result in the content when mutating the values in the PCG method.
- **Interfaces and Controllability for PCG Systems** - This challenge refers to how PCG methods interfaces are usually not user-friendly and hard to understand. Additionally, another challenge that is highlighted here is how the PCG method offers players the ability to change its parameters in order to influence content generation.
- **Interaction and Opportunistic Control Flow Between Generators** - If a PCG method creates a certain type of content that influences other content, it is necessary that this other content also works. The challenge derives from creating a generator that is able to create several different pieces of content that correctly interact with each other.
- **Overcoming the Animation Bottleneck** - The authors mention that 3D animation, is an area in which PCG methods are very challenging to implement due to costs, data size, and time.
- **Integrating Music and Other Types of Content** - Finally, and similarly to the previous one, the authors mention that the creation of music through PCG methods is extremely challenging. This, however, is not relevant to this project, since there will be no attempt to generate music.

2.3.4 Taxonomy by Togelius et al.

Togelius, Yannakakis, Stanley, and Browne (2011), realized that there was a lack of formal taxonomies for procedural content generation and therefore sought to build a classification system to better clarify the role of the different applications of PCG and their distinctions. To realize this classification they came up with 5 different dichotomies; however, in a subsequent book, they present an updated categorization that offers an additional 2 dichotomies (Shaker, Togelius, & Nelson, 2016).

- **Online versus offline -** This first categorization evidences the two points in a game cycle where PCG can occur. Offline refers to when PCG is used before a game is released to the public, meaning that content is generated by an algorithm and then revised and edited by human designers before being used in the finalized version of the game. This offline use of PCG is most common (but not exclusive to) in games that have a single iteration of a world generated by PCG. One of the most well-known examples of offline PCG is found in The Elder Scrolls II: Daggerfall (Bethesda Softworks, 1994), in which the game developers utilized PCG to create the baseline for the massive 209,331 square kilometers map, which was then improved with added detail by the designers.

  Online PCG on the other hand, occurs in-game whilst a player is in the middle of a play section. This type of PCG affects the game on the fly, usually adapting it somehow to players' needs and style of play. One example of online PCG is found in the aforementioned AI The Director of Left 4 Dead (Valve Corporation, 2008). This AI can change the game layout, resources, and enemy placement according to the player's gameplay. Besides offline and online PCG, Togelius et al. (2011) also considered the possibility of intermediate PCG, giving the example of "a real-time strategy (RTS) server [that] suggests new maps to a group of players daily based on logs of their recent playing styles".

- **Necessary versus optional -** The second categorization of PCG stems from the degree of necessity of the generated content. If the content is deemed as necessary, meaning that the player needs to experience the content somehow to progress in the game, then there is a need to ensure that said content is optimal. This type of content not only needs to offer the same degree of difficulty as the one the game is supposed to offer, but it also needs to be generated in a way that it is entirely possible to be used or beaten for progression to occur. The game Spelunky (Mossmouth, 2008), which creates its entire layout by connecting random premade dungeon rooms, needs to ensure that said rooms can be connected. One room's exit needs to connect to another room's entrance in the same spot and vice-versa, otherwise, the traversal of the game's world is made impossible or nonsensical. This exact style of PCG is still seen nowadays, with the creators of the more recent Dead Cells (Motion Twin, 2018), taking inspiration from Spelunky's PCG to create their game's levels.

  If the content is considered to be non-essential to the core game, meaning that the player can choose whether to interact with it or not, then its degree of reliability does not have to be as great as the previously mentioned type of content. One example of this is the game Borderlands (Gearbox Software, 2009), which utilizes a PCG algorithm to create millions of different guns out of several different weapon parts combinations. Most of these weapons are commonly found during gameplay with some being borderline useless but, since trying out different guns to choose the best ones is the core of the Borderlands franchise gameplay, these weapons are not a hindrance to the overall experience, as they can be easily disposed of and replaced.

- **Degree and dimensions of control -** The third factor that distinguishes PCG approaches is the amount of parametrization that is provided to the algorithm. Algorithms that use random seeds create worlds in which their layout is defined by the seed itself, with one seed creating the same world in every single iteration. Minecraft (Mojang Studios, 2011), as already stated, stores a large number of different seeds, each one creating a world that only differs in random entity spawning every time they are used by the algorithm. If there is a need for a higher degree of constraints then a PCG algorithm that utilizes parameter vectors to receive a more specific set of parameters should be used.

- **Generic versus adaptive -** The process of content generation through PCG is, most commonly, generic. This type of generation is labeled as generic because it does not consider a player's behavior, it does not change regardless of how the player plays the game and as such does not represent a uniquely tailored experience. On the other hand, content generation can also be labeled as adaptive, which refers to content that is created by a PCG algorithm that analyzes a player's behavior and generates said content to fit that behavior.

  Adaptive PCG, although not as commonly used in games as generic PCG due to requiring a higher degree of specifications to be developed, tends to create a more pleasing and fluid experience, as the game tries to generate content that better reflects the player's choices. Left 4 Dead's (Valve Corporation, 2008) The Director is a great example of adaptive PCG since its purpose is to study the player's most common choice of weapons, style of gameplay, and amount of resources held to mold the game world in a way that always keeps players on their toes whilst feeling rewarding at the same time.

- **Stochastic versus deterministic -** If content generation through a PCG algorithm can be replicated when given the same starting point and parameters to said algorithm then it is considered to be deterministic. It is considered as such because regardless of the complexity of the content generation process, it can always be restored under the same set of circumstances. The PCG algorithm that generates the worlds in Minecraft (Mojang Studios, 2011) is an example of this, as even though the worlds are massive in scale, the same seed will always create a block-by-block replica of the same world.

  When the content generated is usually impossible to replicate even when the algorithm is given the same parameters, then the PCG algorithm is said to be stochastic. One of the most common stochastic approaches in games is wave function collapse, in which given a single or more modules, the algorithm puts other modules adjacent to the original ones based on a neighbor list of possible outcomes (Møller & Billeskov, 2019). The first game to utilize wave function collapse in the creation of its levels was ProcSkater (Parker & Jones, 2016).

- **Constructive versus generate-and-test -** The last distinction between PCG algorithms is whether they are constructive or generate-and-test algorithms. The former refers to algorithms that generate content only once, through a single iteration, without performing any verification on the content quality; instead, the algorithm is designed to only produce content that is guaranteed to not break the game. Generate-and-test algorithms, however, perform a series of iterations between content creation and content verification, applying conformity tests that ensure that the generated content is valid whilst discarding the one that is not.

The following image is a simple graphic representation of Spelunky's level generation process.

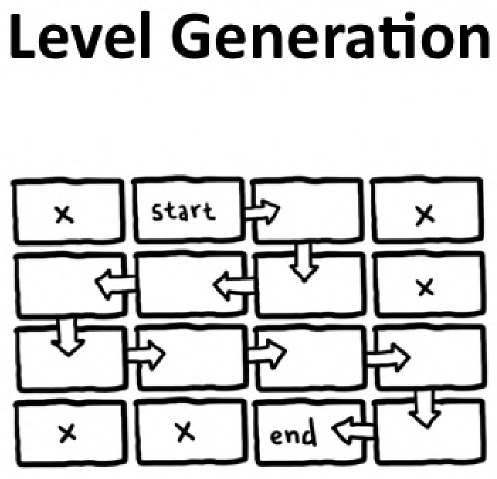


Figure 2.7: Spelunky level generation (*The Full Spelunky on Spelunky*, 2011)

2.3.5 Taxonomy by Smith

Smith (2017) argues that when choosing between PCG approaches one must take into consideration the extent and type of control needed in content generation. To categorize PCG algorithms based on the two previous factors, Smith proposes a set of five different types of approaches to procedural content generation, which can also be combined to achieve greater results.

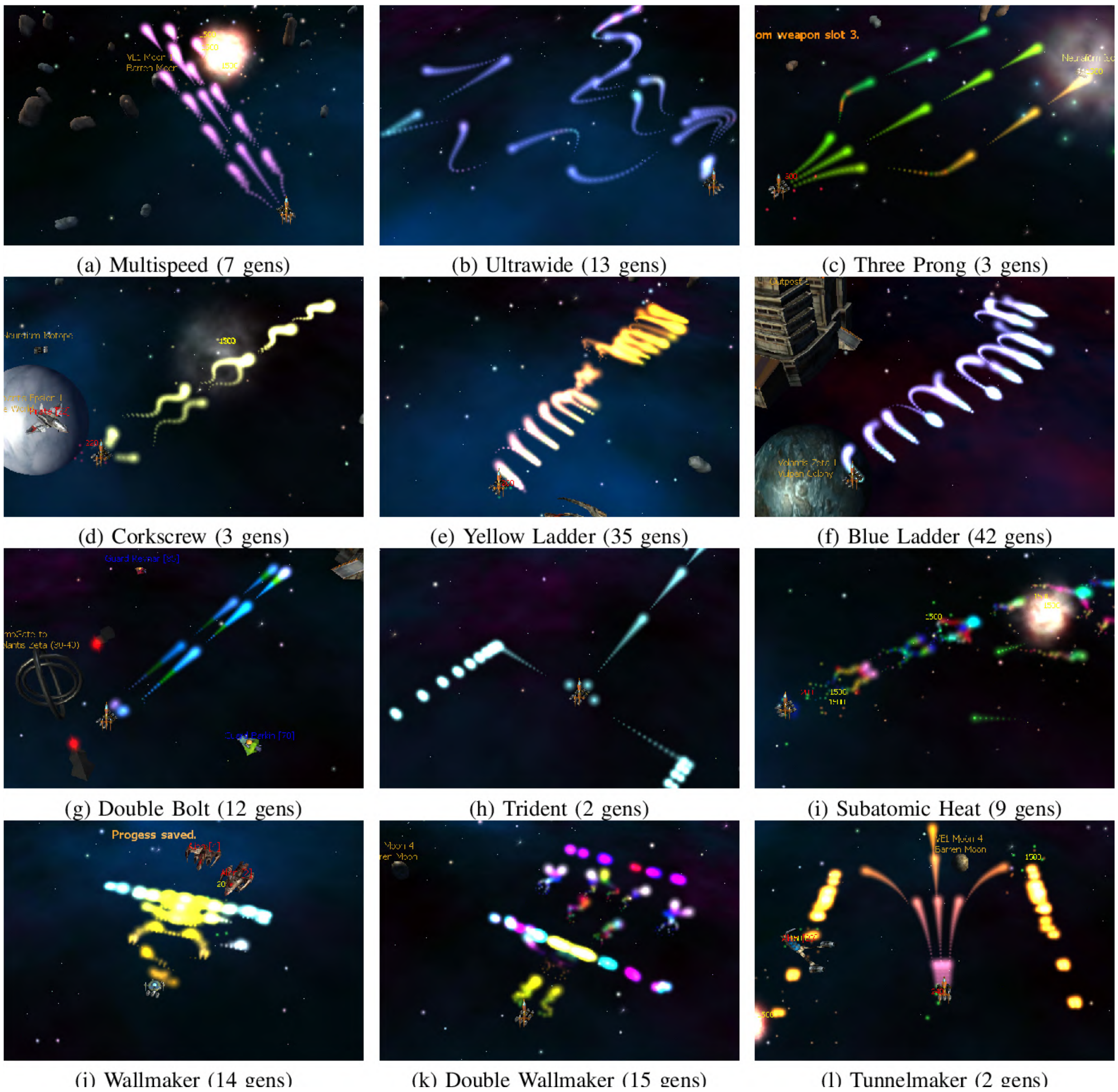
- **Simulation-based systems** - Simulation-based systems are algorithms that, after generating the content, run it through simulations to either check if it is up to standard or evolve the generated content. This type of approach is most commonly used in complex scenarios with several interactions, in which the behavior of the generated content is hard to predict. To be considered a simulation, the verification method has to be entirely conducted by the algorithm, with no human input. An example of this approach can be found in the game Dwarf Fortress (Bay 12 Games, 2006), in which the world structure, animals, and population positions are procedurally generated. After this generation is finished, a simulation is run on all population groups, who eventually start developing their civilization and interacting with each other. Once the simulation is stopped, the player is dropped into that point in time (Adams, 2015).

- **Constructive methods** - A constructive approach utilizes an algorithm "that pieces together premade building blocks" (Smith, 2017), with a larger and more diverse amount of premade building blocks usually dictating a larger amount of content variation and helping fight against human pattern recognition to increase the game's replayability value. Constructive algorithms work as randomness directors, as even though the process of piecing together the building blocks is random, it is done so based on predefined constraints that ensure that the generated content is not faulty. These algorithms, however, are specifically built for the game in which they are being used, meaning that there's little to no reusability beyond those games. This method is very frequently found in rogue-like games, with one of the most well-known examples being Spelunky (Mossmouth, 2008).

- **Grammars** - Grammars can be used in PCG to create large amounts of content at very fast rates. In this type of approach, the rules of content generation are not in the algorithm itself but instead are dictated by a grammar that specifies how the content should be built. This grammar is then parsed through an interpreter that chooses which rules to follow. It is usually recommended that the grammar and the interpreter be separate, as grammars tend to be faulty and are constantly being modified. By doing this, if the need to change the rules of content generation arises, as long as there are no changes to how the grammar must be interpreted, there may be no need to modify the algorithm itself, only the grammar. Additionally, these rules should be such that they are not constrained to the point of not allowing for a lot of diversity but also not lose to the point of creating undesirable content. An example of grammars being used in procedural content generation can be found in Joris Dormans' Zelda-like game (Dormans, 2010), which utilizes graph grammars to generate missions and shape grammars to generate spaces.

- **Optimization-based systems** - In optimization approaches an algorithm tries different content combinations until the most desirable one is achieved. Each combination returns one or more number variables that, when inserted into a specific formula, express the degree of desirability. Because of this, optimization approaches are usually slow, as creating and finding the best candidate outcome tends to take time. Additionally, it is also common to add a human into the equation, by having someone pick what they deem to be the most desirable content; this is known as a human-in-the-loop approach. Galactic Arms Race (Evolutionary Games, 2010) is a game that learns with the player's choices and style of play to generate new weapons that better fit their particular style, effectively using the player as the human-in-the-loop without them even realizing it.

The following image depicts several different weapons generated by the game.



(a) Multispeed (7 gens) (b) Ultrawide (13 gens) (c) Three Prong (3 gens)
(d) Corkscrew (3 gens) (e) Yellow Ladder (35 gens) (f) Blue Ladder (42 gens)
(g) Double Bolt (12 gens) (h) Trident (2 gens) (i) Subatomic Heat (9 gens)
(j) Wallmaker (14 gens) (k) Double Wallmaker (15 gens) (l) Tunnelmaker (2 gens)

Figure 2.8: Weapon evolution in Galactic Arms Race (Hastings et al., 2009)

- **Constraint-based systems** - In this type of approach, an algorithm generates content based on a set of constraints. Unlike optimization approaches that search for the best possible solution, constraint-based approaches define what the best possible solution should be. The speed of this type of approach depends on the size of the game space and the number and type of constraints, although it tends to be very fast in smaller domains, as it immediately discards all of the non-desirable outcomes before they are even generated. Correctly defining the constraints, however, is the most essential part of this approach. When writing all of the constraints there is a need to make sure that all necessary restrictions are being declared, and correctly so, otherwise undesirable content may be generated and desirable content may be blocked from being created. If the constraints are all correctly defined, then it is possible to be assured that desired content will be generated.

2.3.6 Taxonomy by Craveirinha et al.

Craveirinha, Barreto, and Roque (2016) state that Togelius et al. and Smith's taxonomies only implied the role of human actors in PCG approaches. With this knowledge in mind, they sought to create a PCG taxonomy that not only explicitly exhibits the role of human actors but also better distinguishes and clarifies the roles of all different actors; these being the designer, the computer, and the player. The designer refers to anyone who was directly involved in the creation of the procedural content generation system of a game before its release. Computer refers to the algorithms that either prior to a game's release or during gameplay perform procedural content generation. Player refers to anyone who plays a game after its release, regardless of their interaction with the game during gameplay generating changes through PCG.

Additionally, Craveirinha et al. (2016) divide the content generation process into two phases: generation, which refers to the process of creating and iterating content, and evaluation, which refers to the process of assessing the content's quality and desirability.

Finally, Craveirinha et al. (2016) describe the degrees of intervention of each actor in each of the two content generation phases.

The designer's intervention in the evaluation phase can be one of the following types:

- **None** - This refers to cases where the designer does not have any impact on the evaluation process of the generated content. The evaluation process is either done by the computer, the player, or both. For the designer to not have any influence in the evaluation it is also necessary that they do not have any part in the development of the evaluation algorithm, to avoid a biased design.

- **Implicit** - This happens when the designer indirectly influences the evaluation process. This indirect influence can happen in one of two ways: either the designer chooses and switches the procedural content generation technique based on their preferences, or the designer is somehow involved in the creation of the evaluation algorithm in a way that the design itself promotes certain biases.

- **Explicit** - Explicit evaluation by the designers happens when they are directly involved in the evaluation process. This direct influence happens either when the generated content is evaluated by the designer, who picks the most desirable outcomes themselves, or when the designer directly expresses in the algorithm's evaluation process what constitutes good and bad content. Explicit designer evaluation can be found in most games that use PCG, with one example being found in Ghost Recon: Wildlands (Ubisoft, 2017), where the designers iterated between several algorithms to create elements like vegetation and the road system until they found the results that pleased them the most (Seidel et al., 2018).

The designer's intervention in the generation phase can be one of the following types:

- **None** - In some cases, the computer is responsible for generating all of the content without designer input. For this to happen, the computer must be completely independent of the designer in the process of generation, this includes having been created by someone other than the designers of the game.
- **Configuration** - This refers to cases in which the game designer utilizes a pre-programmed procedural content method that offers some sort of customization. This can happen when the PCG method offers the designers the option to choose between different algorithms, when the designer can configure the algorithms to change the procedural process, or when the designer can insert parameters into the algorithm that determine the type of content that is going to be generated.
- **Base-Design** - In base-design PCG method utilizes content created or edited by humans. This category is subdivided into five different types. Four of these types, subcomponents, component patterns, templates, and experiential chunks, were adapted from a categorization done by Smith (2014) and refer to pieces of content with different scales and effects previously built by designers and from which the PCG algorithm can pick to build more detailed content. The fifth type, labeled idea, refers to cases in which the PCG method needs some sort of input that specifies an idea, with this input dictating what the algorithm generates. The artificial game designer ANGELINA (Cook & Colton, 2014), is a system capable of creating entire games based on this concept of idea. It analyzes a word given by the designer and searches for several meanings and thoughts associated with that word, creating content around the results of that search.
- **Co-Design** - This refers to cases in which the designers manually alter the content generated by the PCG algorithm and feed their modified solutions back to the algorithm, with this process being continuously repeated until desirable content is achieved. In this scenario, both the designer and PCG algorithm are responsible for generating the content, with each one relying on the other's work.
- **Meta-Design** - Meta-design refers to scenarios where the designer is the creator of the PCG algorithm being utilized. In this scenario, the designer uses an algorithm with which they are very familiar and that was built according to the specifications that better suit the content and game for which it is going to be used, allowing for a greater degree of control.

The computer's intervention in the evaluation phase can be one of the following types:

- **None** - The computer might not have any impact on the evaluation process. This happens when the computational agent either has no method for content evaluation or does but does not apply it. In these scenarios, the evaluation process is carried out either by the designers or the players.

- **Implicit** - Though neither the authors nor myself could find examples of this, a computer might have, in its generation process, some bias that leads to it favoring some content on top of others, resulting in an implicit evaluation method.

- **Explicit** - This is the most common case of evaluation performed by computers and happens when the computer has a built-in evaluation algorithm that measures the desirability of the generated content. Explicit computer evaluation can be further subdivided into content-based, which refers to cases in which the computer evaluates the content directly by taking into account factors like the number, size, and placement of objects, or player experience, which happens when the computer, instead of evaluating the content itself, evaluates the player's experience with it and translates it into measurable data that can be evaluated, as seen in both Galactic Arms Race (Evolutionary Games, 2010) and Left 4 Dead (Valve Corporation, 2008).

The computer's stakes in the generation phase are the following:

- **Content-type** - This refers to the type of content being generated by the computer agent, which can be divided into five types. Derived content is content that is generated as a consequence of what happens in the game, such as news stations in Mass Effect (Electronic Arts, 2007) broadcasting news related to quests completed by the players. Scenarios refer to the manner and order in which events take place. Space is the environment in which the gameplay is happening. Decorative content is content that increases the appeal of the game and that is usually indirectly experienced by the player, like ominous sounds in Minecraft (Mojang Studios, 2011) that only appear when the player is in a cave. System design refers to the way real-life events translated into the game behave and interact, like the non-playable human characters and animal encounters in Far Cry 3 (Ubisoft Montreal, 2012). Finally, design refers to scenarios in which a PCG method is capable of creating full prototypes instead of just individual content, as is the case of ANGELINA (Cook & Colton, 2014).

- **Strategy** - Strategy is the approach taken by the computer to generate content through procedural generation. The different types of approaches can be found in Smith's taxonomy (Smith, 2017) above.

- **Phase** - Phase is the point in time in a game's life cycle in which the procedural content generation takes place. Taking some concepts described by Smith (2014) and Togelius et al. (2011), Craveirinha et al. (2016) list three possible phases in which PCG can occur. Design-time refers to when the game is still in production and PCG is used to aid in the creation of content, like the world of The Elder Scrolls II: Daggerfall (Bethesda Softworks, 1994). Pre-play refers to the moment after a game is released but before gameplay starts, as seen in Minecraft (Mojang Studios, 2011), which creates the game's world based on a random number seed once the players select the option to start a new game but before they enter the game world itself. Lastly, play-time is the phase in which gameplay is already taking place and PCG shapes the experience the players have in real-time, as happens in Left 4 Dead (Valve Corporation, 2008).

The player's intervention in the evaluation phase can be one of the following types:

- **None** - This scenario occurs when players do not take any part in the evaluation process of a PCG approach, with it being carried out by the computer or designers.
- **Implicit** - Implicit player evaluation happens when player behavior and interactions with the game are analyzed and used as a way to evaluate the quality of the content. This process can happen in two ways: preference-inference and experience model-based. Preference-inference refers to moments where the player's choices during playtime indirectly express their likes and dislikes with the experience, with the generator mapping these preferences into values that can be interpreted and evaluated. In experience model-based, instead of the player's preferences, the player's behavior as a whole is analyzed, with it being mapped into a type of experience, such as, for example, the feelings or actions shown by the players when they are exposed to certain types of content, as performed by the Director AI in Left 4 Dead (Valve Corporation, 2008).
- **Explicit** - This refers to cases in which direct player feedback is used to evaluate the quality of content. Explicit evaluation by the players can be achieved by preference, which happens when they express how much they enjoyed certain content or when they compare it to similar content, or due to experience self-evaluation, which happens when the players evaluate their experience more objectively, without expressing preferences or making comparisons.

The player's intervention in the generation phase can be one of the following types:

- **None** - This refers to scenarios in which players have no impact on the content generation process.
- **Game-play** - Game-play generation occurs when the PCG method generates content based on the player's interactions with the game, dynamically shifting the world to fit the player's behavior. One example of this type of generation can be found in The Elder Scrolls V: Skyrim (Bethesda Softworks, 2011), which generates side quests based on factors like the player's location and choices made throughout the gameplay (Bertz, 2011).

- **Parametrization** - Similar to the designer's configuration described above, some PCG methods allow the players to insert certain parameters that modify some of the content generation processes. Some Minecraft (Mojang Studios, 2011) versions before snapshot 18w06a, allow players to do exactly this, by letting them customize numerous world settings like sea level, ore distribution, creation of certain structures, and even biomes in the game (*Minecraft Interactive Experience*, 2022).
- **Co-Design** - This refers to cases in which the player can directly handle the generation of content with the help of the computer.

2.4 Summary

The research conducted for this section will aid in the making of the video game prototype. Studying the several definitions of play and games allows one to understand what these concepts mean to a player and what type of feeling and interaction is triggered in them. By better understanding of what constitutes both a video game and a board game and the benefits and disadvantages one offers over the other in conjunction with the ideas evoked in the transmediation section, serves as guidelines on what should be taken into consideration in the process of creating the video game prototype based on the board game. In the greater scheme of game design, the approach presented by Salen and Zimmerman (2003) offers the main aspects that should be focused on when designing a game, Meaningful play, signs, games as a system, and interactivity are all concepts that can and will be highly focused on in the creation of the prototype. Additionally, the participation model suggested by Pereira and Roque (2013) will aid in creating a game prototype that can offer interactive experiences that promote participation in several different ways. Finally, since the incorporation of PCG into the prototype is the main goal of this paper, the study previously realized what PCG is, the main reasons for its usage, and the different taxonomies proposed, will aid in deciding what type of PCG should be used in the prototype as well as how it should be used.

Chapter 3

Objectives and Methodology

This chapter concerns the main goals and focus points of this thesis as well as the methods and processes that will be used for the rest of the work, taking into account the research that was performed in the previous section.

3.1 General Objective

The first goal of this thesis is the transmediation of a board game into a digital format. The main goal of this transmediation process is to transform all of the original board game's elements into a digital version without its original purpose and feeling being lost, in particular, its potential for socialization, cooperation, and promotion of self-efficacy. The digital version should therefore aim to allow players to do everything that they are capable of in the original board version. However, since the environment of a video game is vastly different from that of a tabletop game, some changes will have to be made in order for everything to fit and work inside a computer screen. Additionally, in order to take advantage of the digital medium, the prototype should implement some additional features that could not be present in the original version, such as sound and animations.

The second goal of this thesis is the inclusion of a procedural content generation method into the digital version of the board game to promote its replayability value by changing the way elements are presented to the players every time they play it.

3.2 Process

The prototype design process will follow Salen and Zimmerman (2003) iterative design approach.

Iterative design follows the idea that the most efficacious way for a designer to evaluate their design is by interacting with it. It is not, however, a guide on how to manage time or resources during the design process, nor is it a specification on how the design should look like. It is a methodology that allows designers to find and fix errors in the game design in a fast and methodical way.

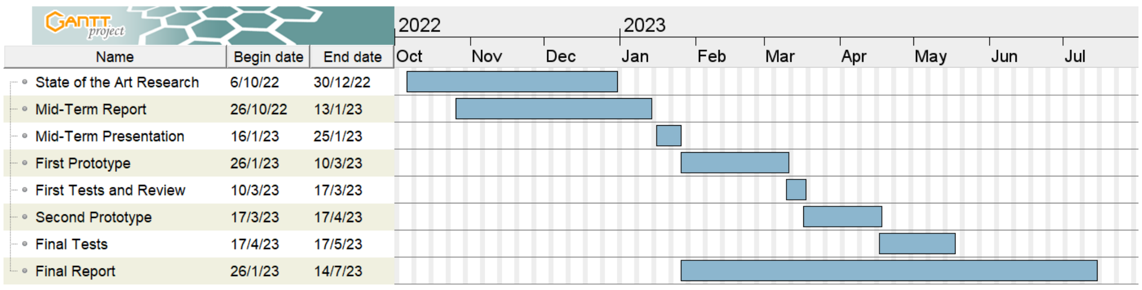
Iterative design is a process in which a prototype of the game is built as soon as possible. This prototype is not supposed to resemble the final product in aesthetics nor should it be focused on it, instead, it is only created with the mechanics and functionalities that the designer wants in the game. After the prototype is created, it should be played, tested, reviewed and finally, modified according to the results found during playtesting. This process is then repeated over and over, with the designers creating several iterations of the prototype where each one improves on the previous version until a desirable outcome is achieved.

Salen and Zimmerman (2003) recommend that the first prototype iteration be created and tested no later than 20 percent through the project schedule, therefore the first prototype of the digital board game should be finished at around that point in time and the subsequent processes of playing, testing, reviewing and prototyping will follow from there.

3.3 Plan and Calendarization

The project's initial work plan was divided into eight steps. In the first semester, the work was focused on the research for the State of the Art section (and, as a consequence, the Objectives and Methodology section), the writing of the mid-term report, and preparation for the mid-term presentation. The second semester was focused on the writing of the final report and the iterative design steps of the design process of the game.

The initial work plan can be observed in the following Gantt diagram.

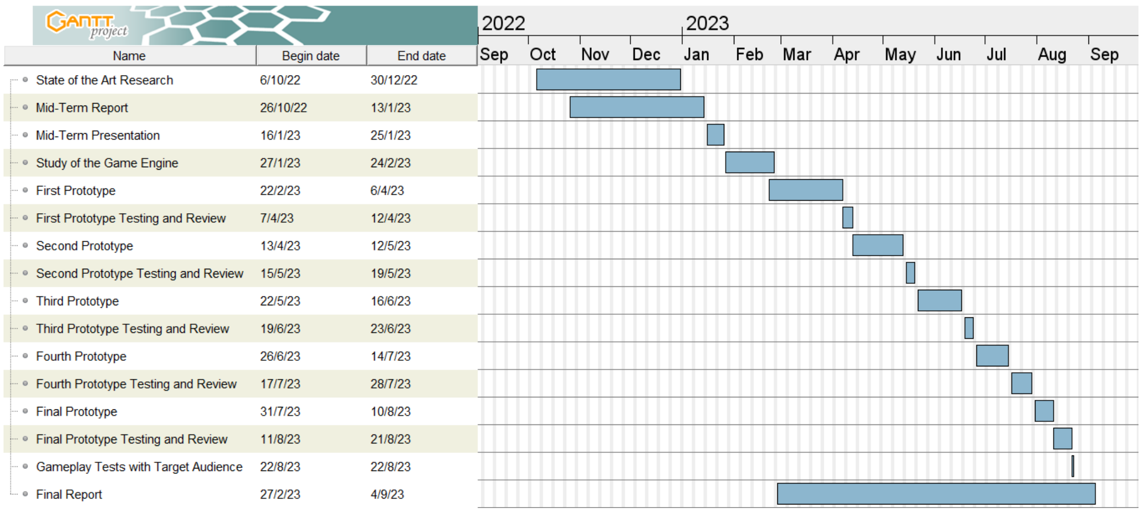


Name	Begin date	End date
State of the Art Research	6/10/22	30/12/22
Mid-Term Report	26/10/22	13/1/23
Mid-Term Presentation	16/1/23	25/1/23
First Prototype	26/1/23	10/3/23
First Tests and Review	10/3/23	17/3/23
Second Prototype	17/3/23	17/4/23
Final Tests	17/4/23	17/5/23
Final Report	26/1/23	14/7/23

Figure 3.1: Work plan Gantt diagram

After active development began, it was clear that the timeline set in the initial plan calendar was not going to be met and that it needed to be revised. Additionally, the total time needed to finish the project ended up being extended as a result.

The way code development was conducted was immediately testing all components as soon as they were created. These were not in-depth tests as the goal was to gauge if the components were functional. Only after several components were added, were longer and more detailed tests performed to verify if the game built thus far was working as a whole and if the components correctly interacted with each other as intended. This resulted in five different prototypes instead of the two initially planned, as can be observed in the following diagram.



Name	Begin date	End date
State of the Art Research	6/10/22	30/12/22
Mid-Term Report	26/10/22	13/1/23
Mid-Term Presentation	16/1/23	25/1/23
Study of the Game Engine	27/1/23	24/2/23
First Prototype	22/2/23	6/4/23
First Prototype Testing and Review	7/4/23	12/4/23
Second Prototype	13/4/23	12/5/23
Second Prototype Testing and Review	15/5/23	19/5/23
Third Prototype	22/5/23	16/6/23
Third Prototype Testing and Review	19/6/23	23/6/23
Fourth Prototype	26/6/23	14/7/23
Fourth Prototype Testing and Review	17/7/23	28/7/23
Final Prototype	31/7/23	10/8/23
Final Prototype Testing and Review	11/8/23	21/8/23
Gameplay Tests with Target Audience	22/8/23	22/8/23
Final Report	27/2/23	4/9/23

Figure 3.2: Revised work plan Gantt diagram

3.4 Possible Risks and Mitigations

Each possible risk has an associated level of impact, probability of occurring, and a mitigation plan.

The impact of a risk can happen at one of three levels:

- **Low** - The paper may suffer small changes that should not interfere with the overall structure and planning.
- **Medium** - The risk will result in changes that might slightly delay the paper and/or change its structure.
- **High** - The impact of the risk is such that both the paper and the prototype may be substantially delayed resulting in considerable changes to planning and structure.

The probability of a risk happening is also divided in three different levels:

- **Low** - 0-40%, unlikely to occur.
- **Medium** - 41-60%, may occur.
- **High** - 61-100%, likely to occur.

The following are possible risks that might occur in the development process of the prototype:

1. Building the prototypes takes longer than expected due to uncertainty or undefined aspects about the game design.

   - **Impact** - High
   - **Probability** - Medium

     Although the probability of this happening was initially thought to be "medium", this risk did, in fact, happen during the development of the game, as the time estimated for the development was underestimated. In order to mitigate this, the time needed to finish the project was reevaluated and ended up being extended. Additionally, some new content that was initially considered for the game ended up being discarded in favor of the content that was already in the original version, so as to not compromise the transmediation process.

2. Prototype testing and review take longer than expected due to too many bugs or difficulties in fixing them.

   - **Impact** - Medium
   - **Probability** - Low

     Although there were several bugs encountered during testing, the amount found and their complexity did not affect the development process. Nevertheless, in the eventuality that this might happen, the mitigation plan would be to create a structured and hierarchized guideline for prototype testing and review based on the features' importance to the game and the likelihood of catastrophic failure.

3. Implementation of the Procedural Content Generation method takes longer than expected due to complexity.

   - **Impact** - High
   - **Probability** - Medium

     As was the case with the previous risk, this one did not occur during development, as the developed algorithm was relatively simple in design. However, were this to occur, a reevaluation of the necessities of the algorithm and the content it should change had to be performed. This reevaluation would hopefully provide further insight into the actual needs of the algorithm in order to find a simpler implementation that was able to achieve the same results.

4. Difficulty reaching testers from the target population resulting in testing and results analysis not being available in time.

   - **Impact** - High

- **Probability** - Medium

  The probability of this risk was initially estimated to be "medium" due to the initially scheduled planning finishing in July. However, because the game development started to backlog and the time needed to finish it was extended, the probability of this happening started to progressively increase every month, since the target audience was composed of children and schools were starting to close. Eventually, once the month of August arrived, schools were already closed, and the probability of the risk increased to high. Thankfully, tests were still made possible and the results analysis can be found further down.

  If tests with the target audience were not possible to be made, in order to still have some sort of results and evaluation performed on the game design, the gameplay test would have been made with adult participants. This would not entirely translate to preferable results, but an evaluation of the game's transmediation and PCG algorithm effectiveness could still be achieved.

3.5 Summary

The two main objectives of this project are to perform a faithful transmediation of a tabletop game into a digital counterpart and to implement a PCG algorithm into this digital version in order to promote replayability.

The development will follow an iterative design inspired by Salen and Zimmerman (2003) concepts and a schedule with the planned prototype iterations and testing was traced.

Finally, several risks that might occur during the development process were enumerated and mitigation plans were set in order to take early precautions if they occur.

Chapter 4

Designing the Game

This chapter serves to highlight the process behind designing the digital game, from the decisions made to how those decisions were validated.

Additionally, a description of what the original game looks like and how it works is also presented at the beginning of the chapter.

4.1 Original Game Concept

The original version of the game which we will attempt to transmediate was developed by Valéria Pinto (Pinto, 2022). Motivated by the goal of "increasing the representation of the female gender in STEM areas, promoting gender balance", the game was created according to the following guidelines:

- "Explore ways to promote the factors that trigger self-efficacy through a collaborative game environment, as a social learning opportunity."
- "Design an experience that incentivizes playfulness in order to promote the development of skills."
- "Design an environment that explores different ways of computational thinking, namely through activities that stimulate the mobilization of competencies for the definition and chaining of an action, its dependencies, and its results, inviting reflection on the space of possibilities."
- "Gather evidence of how the invitation of playfulness and the game environment collaborative work enhance self-efficacy, beliefs in individual capabilities, and consequent consideration of a future in STEM."

As Pinto (2022) summarized it: "[...] this work focuses on the design of a game that seeks to promote, with the support of an attitude of playfulness and collaboration, the manifestation, development, or mobilization of skills that may be relevant to programming, and with this to enhance the sense of self-efficacy in the individual."

In order to find a setting and characters for the game that would go in tandem with the goals of the project, Pinto (2022) researched groups of animals that work as a group, performing collaborative tasks, regardless of their gender, for the well-being of the whole colony. From this research, meerkats were found to meet these requirements and were subsequently chosen as protagonists of the game.

The final version of the prototype, which would later be titled Meerplay, puts a team of players, up to six at a time, in the "shoes" of meerkats who move across a savanna looking for food to bring back to their burrow and grow their colony, all the while trying to avoid or fight off predators as well as other obstacles. Although a digital version was not developed at the time, a paper and cardboard prototype of a board game was built in order to perform gameplay tests with several players to gauge if it had achieved the desired goals.

4.1.1 Game Arena

The board game in question is a six-by-eight matrix where each cell represents an area that players can move to. When players enter a cell, they can choose to lift up the piece of paper that is covering it in order to see what is hidden underneath it and interact with its content.

Each player takes control of a meerkat that starts in the bottom right cell and their goal is to work together to learn how to play the game and expand their meerkat colony, although independent gameplay is also possible.

At the beginning of each turn, every player throws a dice and the number that each one gets represents the number of actions they have in that turn. Players can then choose to either share the number of actions between them, by giving some or all of their actions to other players or keep it for themselves.

The following images show the original game's arena and a gameplay session in that version of the game, respectively.

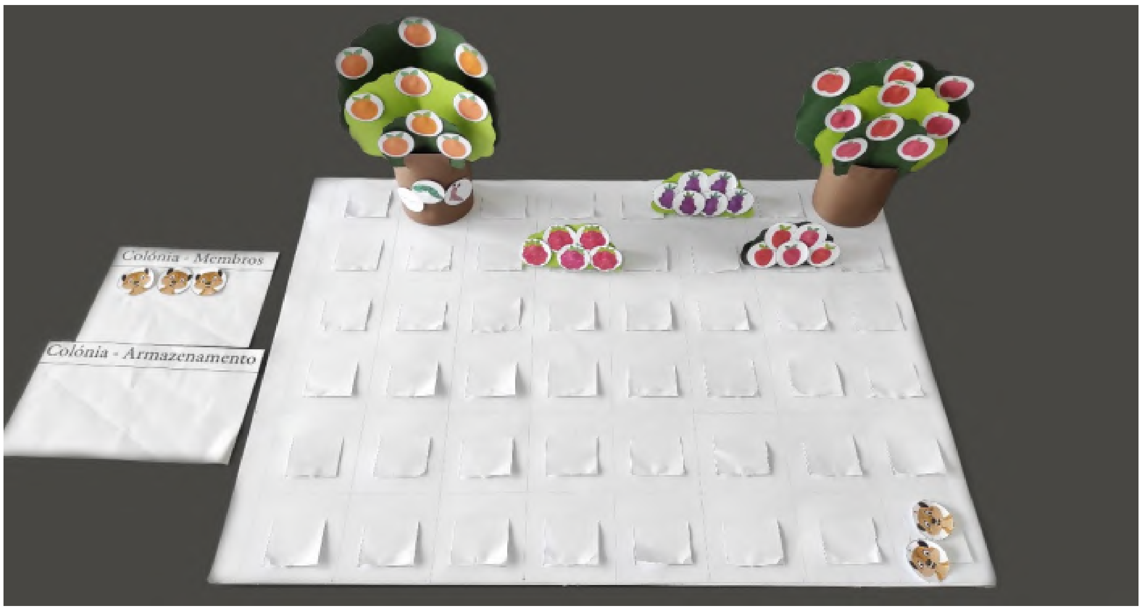


Figure 4.1: Board game arena (Pinto, 2022)

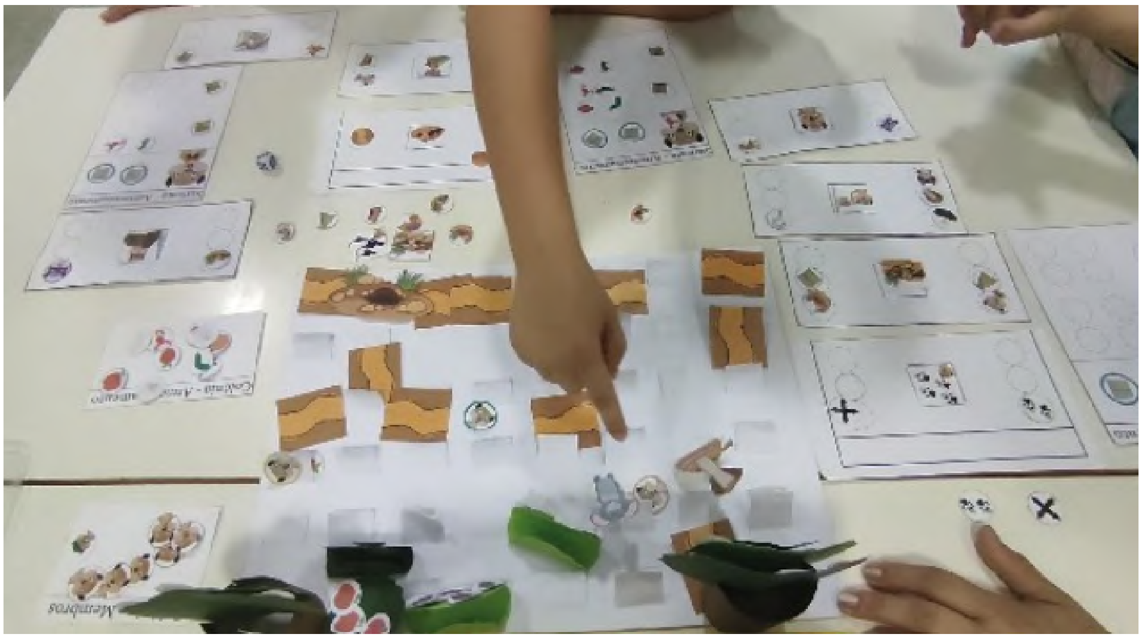


Figure 4.2: Gameplay session of Meerplay (Pinto, 2022)

4.1.2 Player Action

As stated previously, the number a player gets on a throw of dice represents the number of actions they have for that turn. An action is anything the player does that may change the game, such as moving to another cell, picking up food, creating a burrow, storing food in the burrow, etc. Every time they perform one of these actions, their number of actions for that round is decreased. Once the total number of actions is zero, the players throw the dice again and a new round begins.

For a player to perform an action, however, it has to first be defined. The way players define these actions is "based on a Petri Net model" (Pinto, 2022), where they have to define a condition, an action, and an outcome. Each action the players define can be performed when its predefined conditions are met and it will always result in its predefined outcome. Until an action is defined, players will not be able to perform it, even if it is something as simple as moving to another cell.

Furthermore, although the conditions, actions, and outcomes that the players can choose are constrained to a list provided by the game, the number of possible combinations is relatively large, since an action can be tied to many different conditions and outcomes. The validity of these combinations i.e. if a condition, action, and outcome combination is sensical is up to the players to decide and agree upon.

Additionally, the number of combinations the players can have defined at any given time is limited, meaning that they have to evaluate the game environment to assess which actions are more relevant and change them accordingly.

The following image represents an example of how Petri Nets are built and look like in the original game.

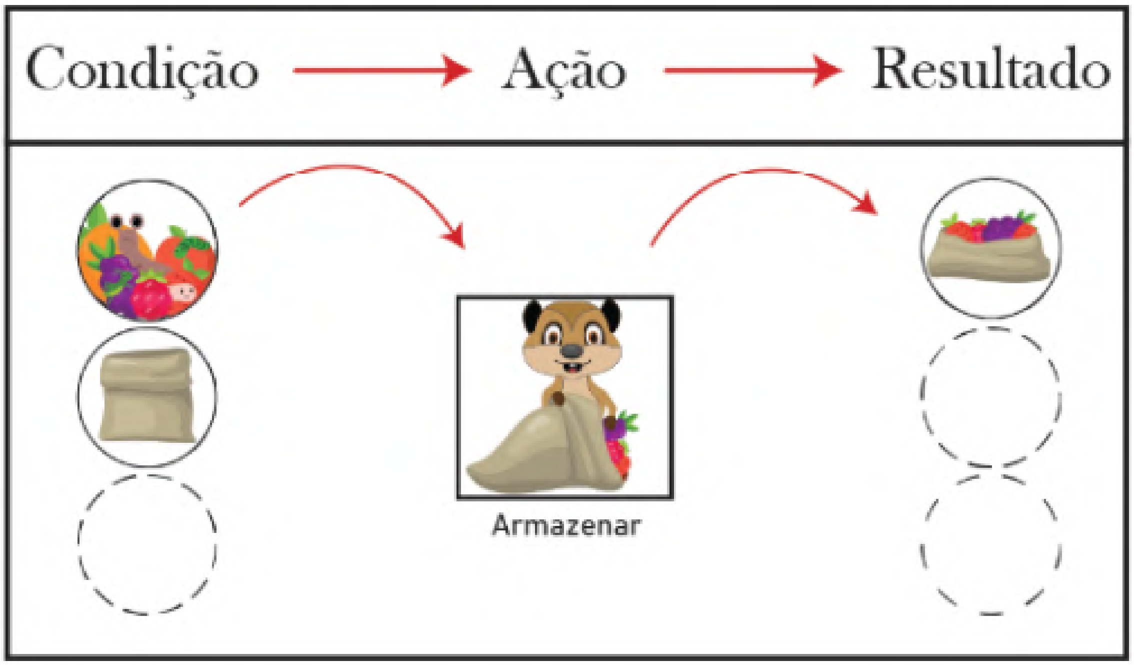


Figure 4.3: Petri Net model example used in the board game (Pinto, 2022)

4.1.3 Objective of the Game

The way the game works is players move their meerkat through the game's cells and choose to either open or let the cell they walk into remain closed. If the cell is opened, the player can see and interact with the elements that were hidden underneath it as long as the player remains in that cell. These elements might be favorable to the players, such as food or backpacks that allow them to carry more food, or adverse, like a raccoon that can steal food that the players have stored in the burrow.

Although the game allows for a great degree of freedom, there are specific goals that the players are expected to achieve. The players' first goal should be to try to create the aforementioned burrow for the meerkats, which will serve as the primary cell for their gameplay. After the burrow is created, players should then try to collect and bring food to the burrow. As food is dispensed into the burrow, baby meerkats start appearing in it and as gameplay proceeds, the baby meerkats grow into adults. These meerkats in the burrow, however, are not meant to be used as pawns like the ones players control, instead, they allow players to perform new actions and also serve as an indicator of how well the players are performing. The more food and meerkats exist in the burrow, the better the players are performing.

It is possible, however, for players to lose some or even all of their food and meerkats living in the burrow if faced by a raccoon or a predator respectively. When a player opens a cell with one of these animals one of two things can happen:

1. **Victory** - If the players have previously defined actions that allow them to engage these animals, a mini-game is played where all of the players throw their dice to try and match or surpass a certain number (in this prototype version the number was revealed by Pinto (2022) to the players). If the total number achieved by all of the players reaches the predetermined number, the players are allowed to perform the action on the animal. If the animal is a predator (eagle, wolf, or snake), it can either be fought or players can flee from it. If fought and defeated the predator disappears and if fled from the player is sent to the burrow (or starting cell if a burrow is not yet created), but the predator remains. The number that players need to reach for fleeing, however, is lower than the one for fighting, so it is up to the players to decide how they want to engage the predator.

   If the animal is a raccoon, it cannot be fled from, so players can only fight it.

2. **Defeat** - Otherwise, if the number of the mini-game is not reached or if the players have not defined actions to flee or fight that particular animal, they lose. If players lose against a predator the player that was in the cell with the predator is sent back to the burrow and a dice must be thrown to check how many meerkats inside the burrow will be lost. If a player loses against a raccoon no player moves, but a dice is equally thrown to check how much food the raccoon steals from the burrow.

Additionally, it is also possible to create tunnels that, when connected, allow the players to move between them at the cost of a single action, effectively reducing the number of actions needed to move around. These tunnels can be destroyed if a player encounters a rat, which is an encounter that works under the same rules as the ones for the raccoon. If a player loses against the rat, a dice is thrown to check how many tunnels are randomly destroyed.

Since there can only be one burrow in the game and it is shared by the players' meerkats, the food and meerkats present in it refer to all of the players' performance as a whole, which encourages them to work as a team.

The game effectively ends when all of the food hidden in the board is stored in the burrow.

4.2 Transmediation: from Tabletop to Digital

The following sections describe the decisions behind the transmediation process, from the type of graphics and visuals to early sketches of the user interface.

4.2.1 Game Graphics and Perspective

It was decided very early on that, due to both time constraints as well as the higher degree of difficulty that comes from creating a 3D game, the digital version of Meerplay was going to be a 2D game. Additionally, since there would be no one with a design background working directly on creating the game's content, a retro style based on pixel art was chosen as the art style for the game due to being easier to draw than more realistic graphics, although still arguably complicated to get right.

Since the decision to make a 2D game was already set in stone the question of how to represent the game arena came down to the position of the board in relation to the viewpoint of the players. The arena could either be viewed directly from above, such as in games like the original Grand Theft Auto (BMG Interactive, 1997), which, since Meerplay has a stationary arena, would provide almost no degree of isometry and all of its elements would be seen solely from above, or it could be viewed from an angle, reminiscent of the viewpoint players had when playing the paper version, which allows for more details and content to be seen.

The choice to pick an isometric point of view was almost immediately agreed upon when the previous question was raised due to allowing to bring as much essence from the original prototype to the digital version as possible, as well as allowing for the exposure of more content at any given time, which both enrich the game environment and help guide the player's choices. It also has the potential to provide a higher sense of scale and tridimensionality to a 2D game, making it seem more detailed and complex.

Additionally, research was made on the visual design of some digital games that were adapted from board games, such as Scythe: Digital Edition (Asmodee & The Knights of Unity, 2018) and Catan (Asmodee & Dovetail Games, 2019), as well as on games that possess a similar arena and gameplay to Meerplay, as seen in both Super Mario Party (Nintendo, 2018) and Monopoly Plus (Ubisoft, 2014), which showed that an isometric view was picked for most cases, further cementing the decision to pick isometry.

4.2.2 Designing the User Interface

The tabletop prototype for Meerplay is made up of four major areas positioned next to each other in which the game takes place and is interacted with. The first challenge in the transmediation process was to iterate through several ways of adapting these areas into the digital medium and determine which solutions provided the most desired results.

The largest and most important area of the game is its arena. This area is where players will be focusing on most of their gameplay and therefore was the first one worked on in an attempt to bring it to the digital medium.

As previously mentioned, Pinto (2022) prototype's arena is a six-by-eight matrix drawn on a large paper sheet where each cell of the matrix represents a location where the players can move to and from. In addition to the cells being drawn on the arena, there are additional elements such as trees and bushes made out of cardboard that are placed on top of some of the cells, providing a sense of tridimensionality or depth.

The initial digital design for this arena originated immediately as the result of the decision to create a 2D isometric game, as mentioned in the previous subsection. Therefore the arena would be a simple rectangle made up of six lines and eight columns forming squares that represented the cells the players could interact with. This rectangle would be seen from an angle slightly above it and next to one of its corners, to mimic the view players have when playing a board game in person.

The other three areas of the game are its menus, which can be directly or indirectly interacted with. These are the Petri Net menu, the inventory each player owns, and the colony menu, composed of two sub-menus, one that shows how many meerkats are in the burrow and one that shows how much food is stored in the burrow.

Although the colony menu is divided into two other menus, these two would always be shown next to each other when the prototype version was set up and worked in tandem with one another. Because of this, when creating design ideas for the menus in the digital version, the colony menu was seen as one single menu divided into two, instead of two separate menus. Hence the reference to only four main areas.

One of the initial challenges was deciding where the menus should be placed on the screen, so they do not overlap most of the game, allowing players to look at both the menus and the arena at the same time since these menus are inventories that show the state of the game. They should also be located in different places on the screen to not generate confusion for the players and to allow more than one menu to be opened at a time if needed. It was decided that due to being similar in shape and size the inventory menu and the colony menu would be placed on the left and right side of the screen respectively, whilst the action definition Petri Nets menu would be placed at the bottom of the screen.

Additionally, since the menus have to be opened and closed in order to not occupy screen space, a tab was drawn on all sides of the screen but the upper one. When clicking on the left tab the food inventory menu will slide to the right showing itself and, when the tab is clicked in this state, the menu will slide to the left, disappearing from the screen. The same logic is implemented in the other two menus: click the right tab to open the colony menu, click it again to hide it on the right; click the bottom tab to make the Petri Nets menu slide up, click it again to make the menu slide down and out of view.

The following is the mockup created for the menu positions.

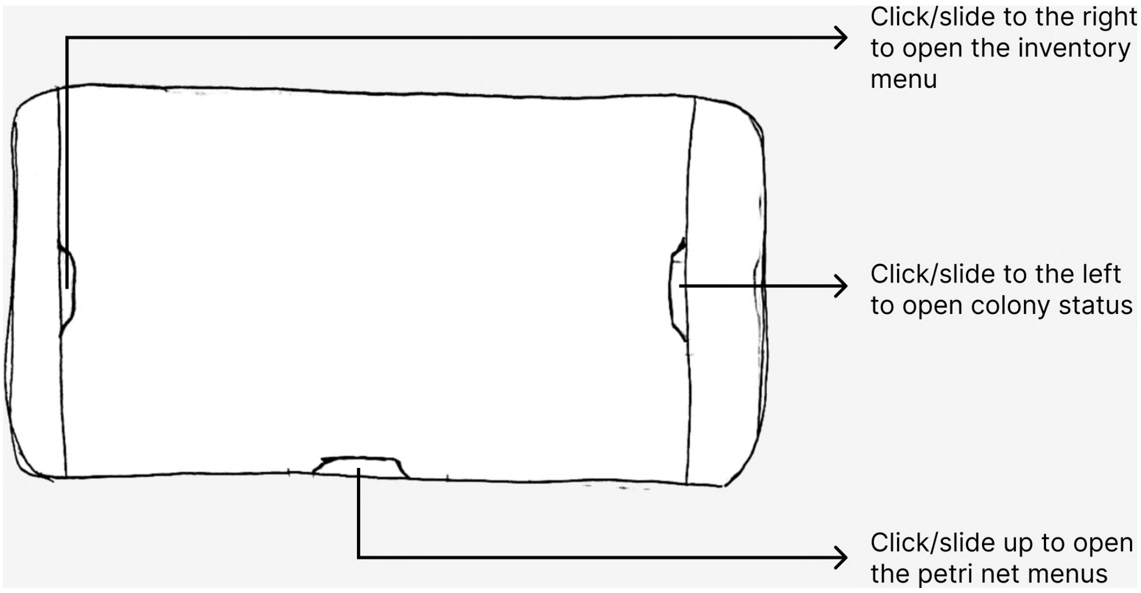

Click/slide to the right to open the inventory menu
Click/slide to the left to open colony status
Click/slide up to open the petri net menus

Figure 4.4: Menu's tabs mockup

When it came to the design of the menus themselves barely any changes were made, since in the prototype version all three menus were already very simplified and effective.

The following two images serve to compare the design of the food inventory menu in the original game and the mockup designed for the digital one.

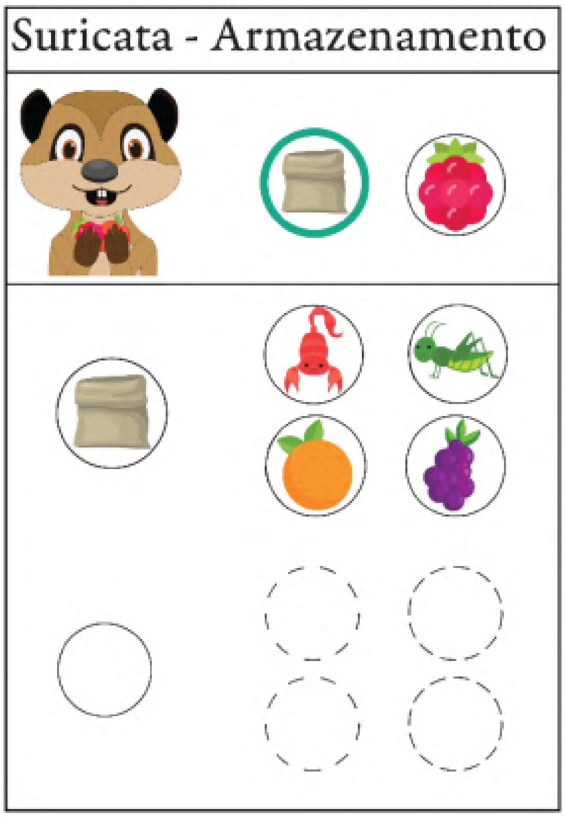


Figure 4.5: Player food menu example (Pinto, 2022)

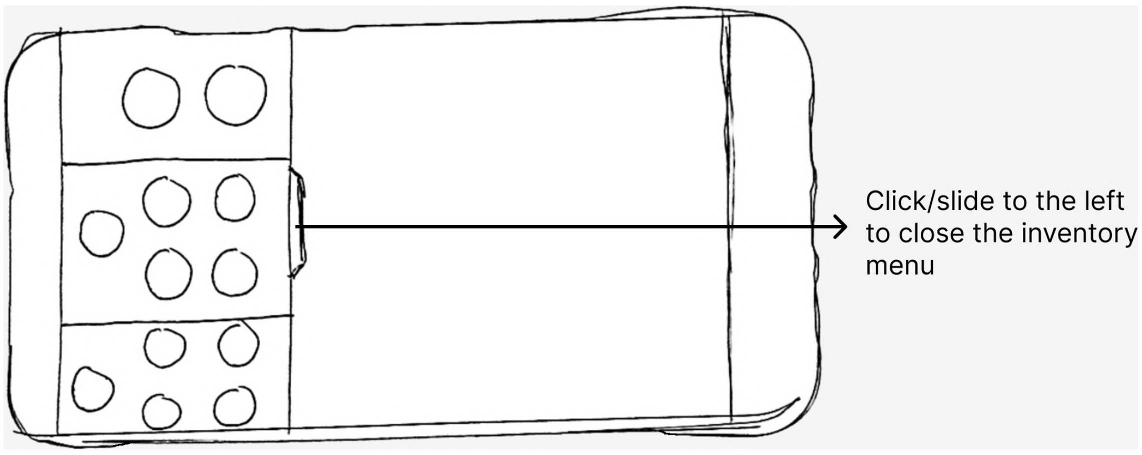


Figure 4.6: Food menu mockup

The following two images serve to compare the design of the meerkat and colony food inventory menus in the original game and the mockup designed for the digital one.

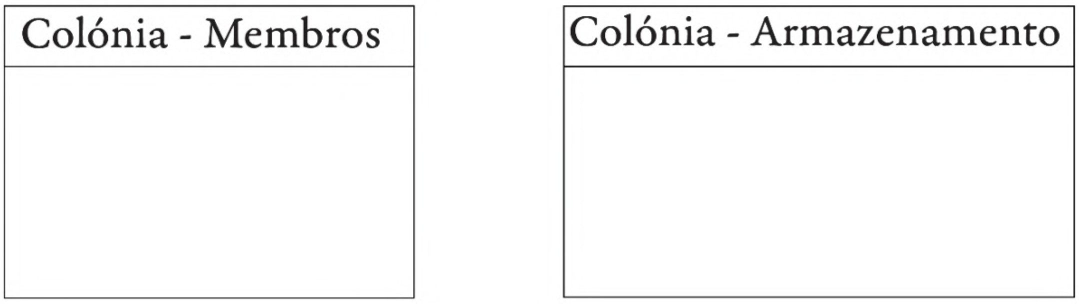


Figure 4.7: Player colony menu (Pinto, 2022)

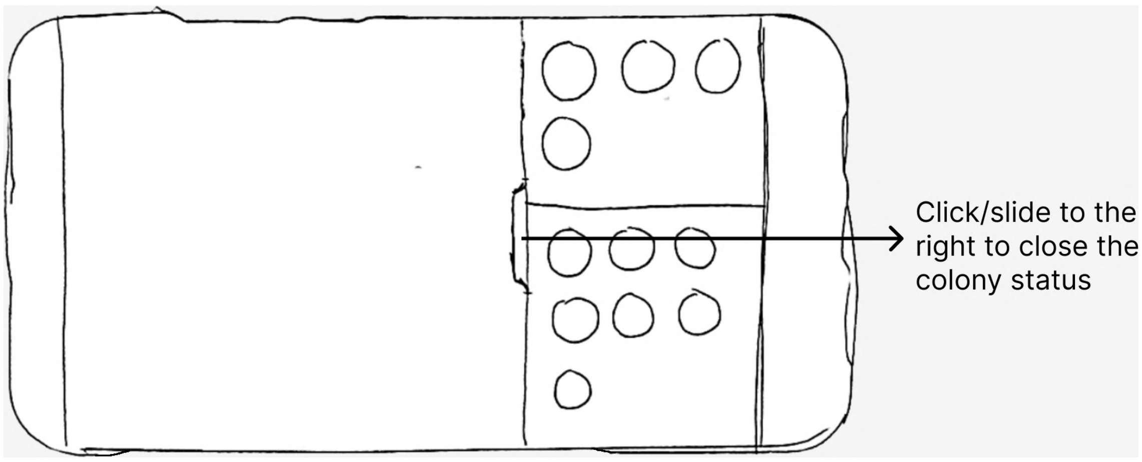


Figure 4.8: Colony menu mockup

The bigger challenge was deciding how to represent all of the Petri Nets due to their size and complexity. Since by the end of the game each player has three Petri Nets and there is a maximum of six players, there could be eighteen Petri Nets in the game simultaneously. If eighteen Petri Nets were shown at the same time either the entire screen would be occupied by them or they would be so small that the players would not be able to see their content, much less interact with them, therefore only a few Petri Nets could be shown at a time.

This resulted in the decision to organize the Petri Nets in a slider, which players would move in order to see other Petri Nets, only being able to see three definitions at a time. The Petri Nets were then packed into one single large menu and placed at the bottom of the screen.

The following image is the mockup created for the look of the Petri Nets menu.

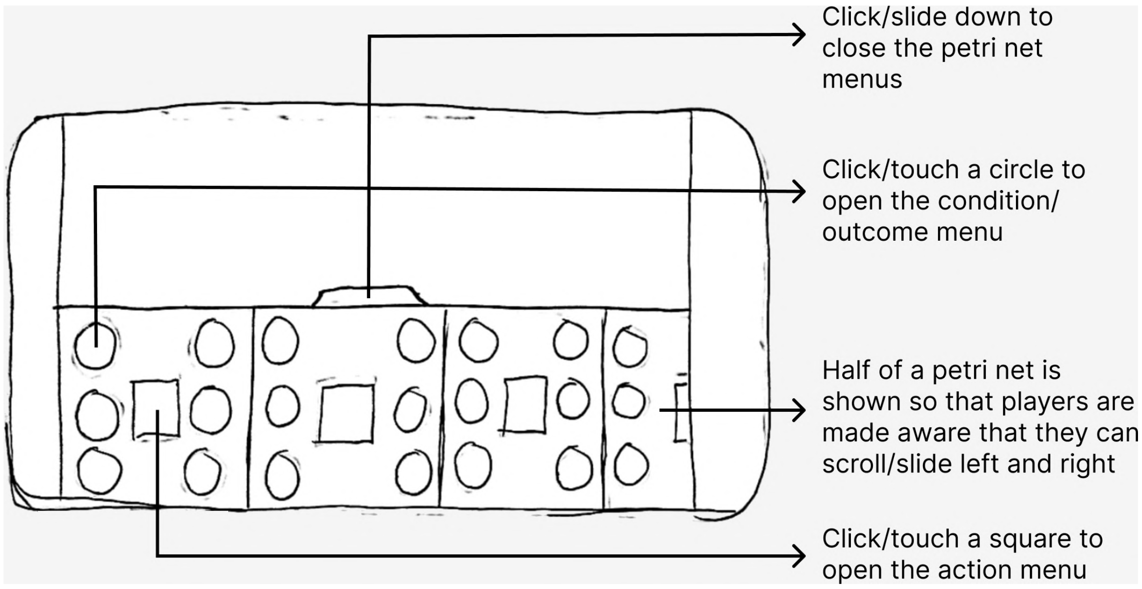

Click/slide down to close the petri net menus
Click/touch a circle to open the condition/ outcome menu
Half of a petri net is shown so that players are made aware that they can scroll/slide left and right
Click/touch a square to open the action menu

Figure 4.9: Petri menu mockup

Although an acceptable design was achieved for all three menus, one final challenge arose. In the prototype, players would create Petri Nets by moving circular pieces of paper with conditions/outcomes, and square pieces of paper with actions drawn in them, into the respective condition/outcome and action areas respectively. The total amount of conditions/outcomes and actions are, like in the case of the Petri Nets, too large to show all at once. Because of this, a similar design to the one created for the Petri Net menu was proposed. In this design, when players clicked on a circle of a Petri Net, a new menu would pop up in the middle of the screen, showing some of the conditions and outcomes, allowing players to slide to the side to see the remaining ones and once one was clicked on, it would appear on the Petri Net circle the player had originally selected.

The following images show the mockups created for both the slider menu for the conditions and outcomes and how the Petri Nets would look with those elements.

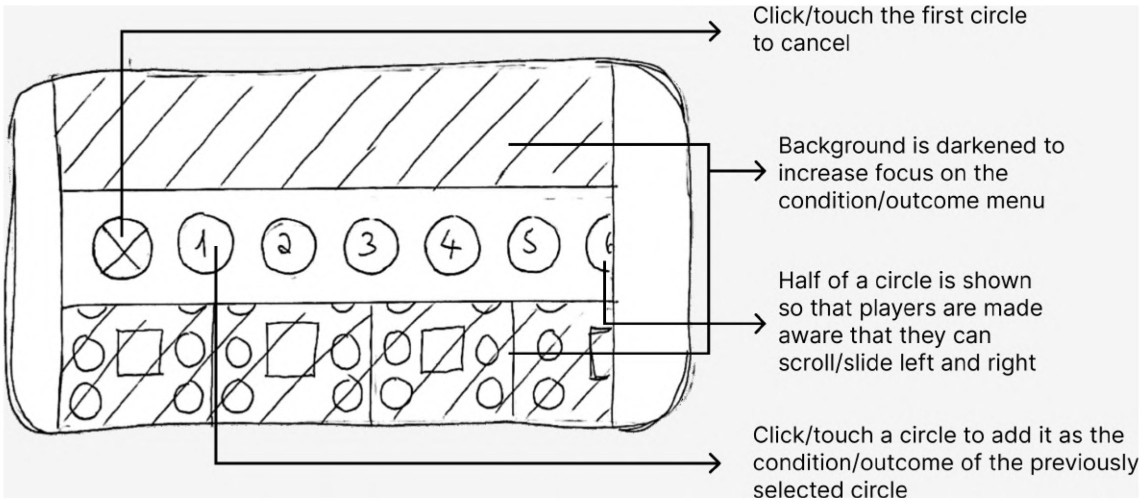

Click/touch the first circle to cancel
Background is darkened to increase focus on the condition/outcome menu
Half of a circle is shown so that players are made aware that they can scroll/slide left and right
Click/touch a circle to add it as the condition/outcome of the previously selected circle
1
2
3
4
5

Figure 4.10: Conditions and outcomes menu mockup

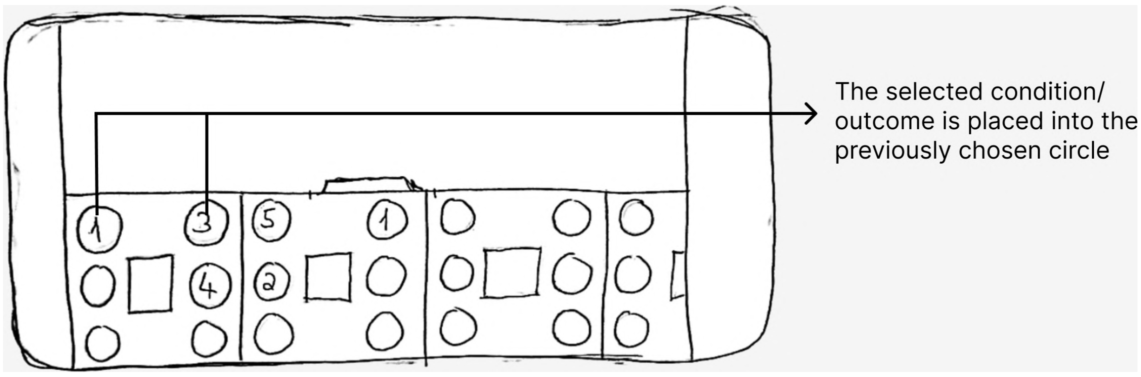

The selected condition/ outcome is placed into the previously chosen circle
1
3
5
1
4
2

Figure 4.11: Petri Net with conditions and outcomes mockup

A similar menu was proposed for the actions, as observed in the two images below.

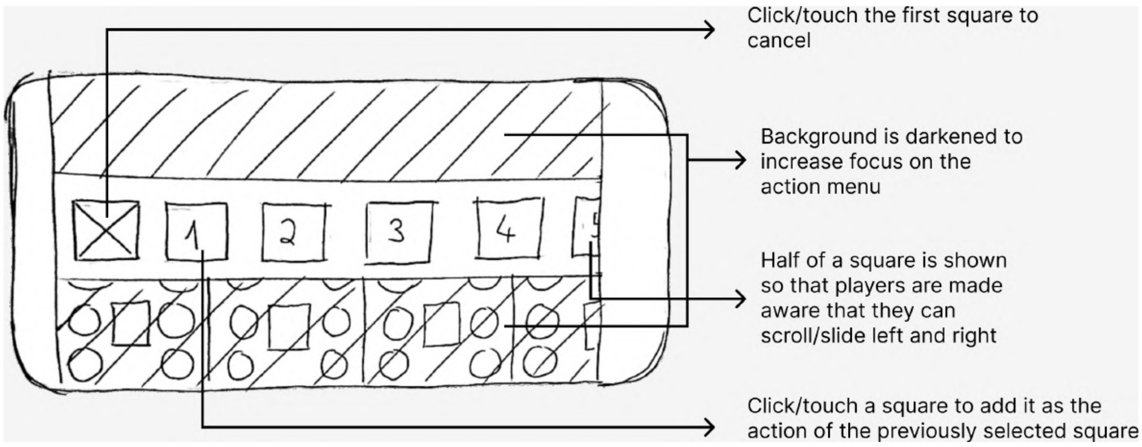

Click/touch the first square to cancel
Background is darkened to increase focus on the action menu
Half of a square is shown so that players are made aware that they can scroll/slide left and right
Click/touch a square to add it as the action of the previously selected square
1
2
3
4
5

Figure 4.12: Actions menu mockup

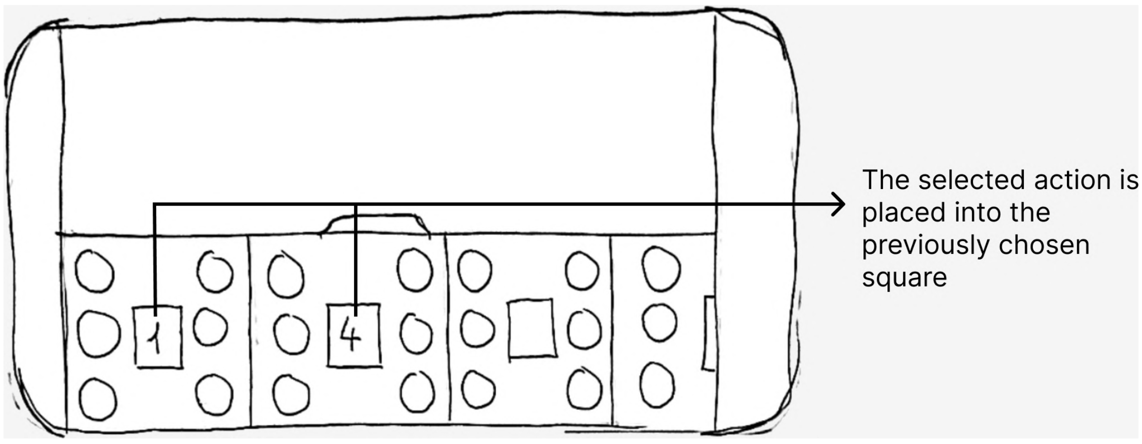

The selected action is placed into the previously chosen square
1
4

Figure 4.13: Petri Net with actions mockup

In addition to the game arena and three main menus, two other components of the game would have to be redesigned in order to work on a digital version.

The simpler one is the Dice. At the start of each round, each player has to throw a dice to get the amount of actions they can use in said round. In order to reflect this, it was decided that in the digital game, every time a new round started the screen would get slightly darker, and a dice would appear in the middle of it to signal the players they should click it. Once clicked, the screen would go back to the game and each player would be given their respective action numbers.

The following is the mockup drawn to represent the dice players have to throw in order to gain actions and defeat predators.

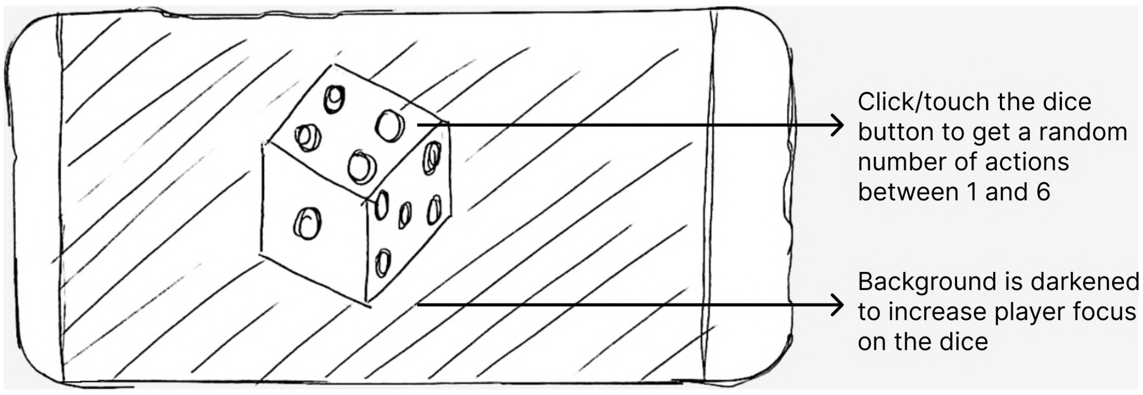

Click/touch the dice button to get a random number of actions between 1 and 6
Background is darkened to increase player focus on the dice

Figure 4.14: Dice mockup

The more complex component is the Cell, which enables players to activate an action when interacting with the content hidden underneath a cell.

In the prototype, when landing on a cell, players would open a piece of paper hiding content underneath it and interact with it by stating which of the defined actions they choose to perform with the content. For the digital version, we proposed that when players landed on a cell, a menu would be opened that showed all the content hidden, including an option to close it. Once one of the pieces of content was selected, a similar menu was opened in place of the first one, showing the outcome of any action the players had created that utilized the content the player clicked on as a condition. For example, if the players have defined three actions that each use the condition labeled "3" and have the outcomes labeled "2", "4" and "5", clicking on the condition labeled "3" in the cell's menu will make the second menu pop up and show the outcomes "2", "4" and "5" so that the player can choose which one to perform. This example can be seen in the following mockups.

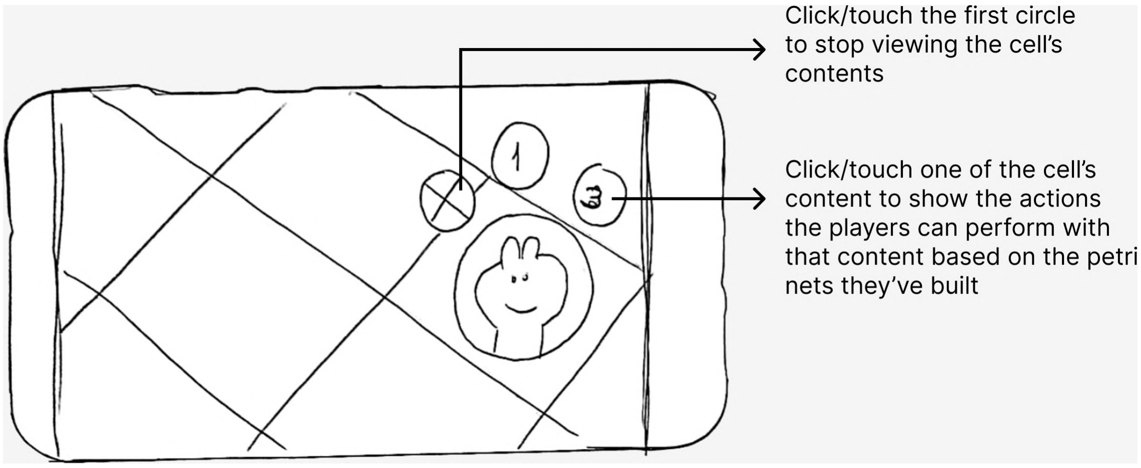

Click/touch the first circle to stop viewing the cell's contents
Click/touch one of the cell's content to show the actions the players can perform with that content based on the petri nets they've built
1
3

Figure 4.15: Cell content menu mockup

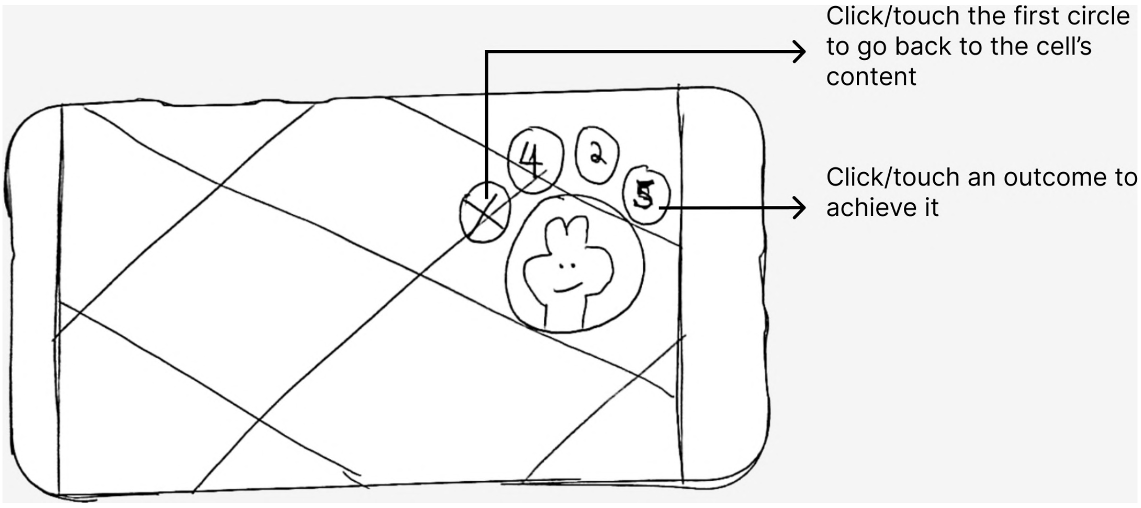

Click/touch the first circle to go back to the cell's content
Click/touch an outcome to achieve it
4
2
5

Figure 4.16: Cell content menu action options mockup

4.3 Design Inspections and Evolutions

To produce and evaluate the design sketches here presented, for the digital version of Meerkating, several meetings were conducted with the game creators, Valéria Pinto, Mariana Seiça, and Licínio Roque. During initial meetings, ideas for the design of the game's core elements such as the arena and menus were suggested by the author. Through systematic usability inspections and discussions, the design propositions that were inevitably deemed too convoluted, nonsensical, or simply unappealing were discarded or reformulated. The promising ones were further detailed to be discussed in the following meetings and eventually prototyped for feasibility.

For large interface components such as the more complex menus, once the idea for a design was considered acceptable, a mockup was drawn to give a better idea of how said design would look once implemented. These mockups were then presented at the meetings to discuss them further and, when needed, modifications or complete overalls were suggested and new mockups were drawn. This iterative process of drawing mockups based on the best design ideas gathered at the meetings and presenting them at future meetings to discuss improvements was made until the mockups were deemed good enough for future implementation.

The final mockups can be found in the previous subsection.

When the content being discussed was of a smaller scale, such as how and where to represent the number of actions each player has, how to skip turns, how to share actions, etc. the ideas were mainly suggested and accepted through verbal discussion or, if needed, rough sketches were drawn on a board or sticky notes to better project the idea in the person's mind.

4.4 Summary

The original prototype developed by Pinto (2022) has several interactive menus that players need to engage with throughout gameplay. In the paper prototype, all of these elements are laid out on a table with enough space for all of them, which lets players see everything at the same time. For the digital version, however, the screen-size constraints imposed by computers and cell phones do not allow all of these menus to be constantly visible for the entire duration of the game, especially not simultaneously.

Changes were proposed for the digital version that allowed the components of the original game to be adapted into this new medium in a way that fits the players' screens without taking too much space. Likewise, new ways of interacting with the game were created in order to mimic the interaction players have with the original prototype, as well as how the game reacts to that interaction.

Chapter 5

Game Development

The development process of the digital version of Meerplay went through several stages of work. The following chapter serves to highlight this development process, from the choice of the game engine, the game architecture, the developed PCG algorithm, the integration with a server to allow online play, the game's current interface and gameplay, and finally, a description of the overall process timeline and planning.

5.1 The Selected Game Engine: Godot

Godot is a cross-platform game engine that allows the creation of both 2D and 3D games. It provides its users with comprehensive and easy-to-use tools that allow them to quickly and more efficiently focus on creating the game instead of spending too much time learning the engine. It is entirely free and open-source under the MIT license, making the creator's games solely theirs. It is also non-profitable and community-driven, incentivizing all its users to help in its improvement.

Due to being community-driven, there is a vast amount of free tutorials online that can and were used to learn its basics, as well as more advanced functionalities.

Since this project is a first-time experience in video game making, a game engine that is comprehensive, easy to learn, has little to no overhead, and has a large number of free tutorials on every functionality as well as on how to create almost anything that can be included in a video game, would be the perfect pick. Godot was found to be the option that better fit all of these criteria and was ultimately chosen as the game engine for this project.

Additionally, the digital version is expected to work on both computers and mobile devices, and Godot allows projects to be exported to these platforms very easily.

The version of Godot used to develop the game was Godot 3.5. since it was the latest most stable version of the engine by the time the game was starting its development. As the game was being developed, Godot 4.0. was released, however, it was decided that the game would not be ported to this version since there were several changes and tutorials for this version were almost nonexistent. The version was very recent and therefore could prove unstable, and the interface and some of its tools changed, which would mean re-learning the engine.

5.2 Game Architecture in Godot

The fundamental building blocks in Godot are called Nodes. Nodes are structures that offer a variety of functions, such as displaying images, playing audio, animation, buttons, etc. Nodes can also have children, effectively becoming a tree. A group of Nodes organized hierarchically is called a scene. Scenes can be used independently or in conjunction with other scenes. Additionally, it is possible to create several instances of the same scene if necessary.

With the Godot game architecture in mind, our game is composed of fifteen different scenes:

- **CellContent** - Scene that shows and manages the button for players to search a cell and the respective cell content.
- **ColonyMenu** - Scene that represents the players' colony menu, showing how many meerkats are in the burrow and how much food is stored.
- **ConfrontPredator** - Scene that allows players to choose between fighting or fleeing from a predator when players have both actions defined. Always appears before DicePredatorControl.
- **DicePredatorControl** - Scene that shows up when players face a predator or thief.
- **FoodMenu** - Scene that represents each player's food menu.
- **GameManager** - Although GameManager is a scene it only serves as a script that is AutoLoaded, meaning that it is accessible throughout the entire game, even when the main scene is changed. Its purpose is to store information created during a scene, such as a player's ID, which is created in the StartMenu scene, so that this information can be used in other scenes, such as Lobby and World. It is also responsible for all of the broadcasts with the server. This scene is never shown to the player and is not directly linked to any other scene.
- **Lobby** - Scene that shows all the players waiting to start a game session together.
- **PetriCircle** - Scene that has the circles shown in the Petri Nets.
- **PetriSquares** - Scene that has the squares shown in the Petri Nets.

- **PetriNet** - Scene that has an entire Petri-net made up of PetriCircle and PetriSquare scenes.
- **PetriMenu** - Scene that represents the players' Petri menu. It is made up of PetriNet scenes.
- **Player** - Scene that represents the playable character.
- **PlayerMenu** - Scene that represents the players' player menu, where they can see every player's actions and bags.
- **StartMenu** - The first scene of the game, allows the players to either create or join a game.
- **World** - Main scene of the game. This is where the arena is built, where the menus are put into place, and where the players are introduced as children of one of its nodes. This scene is where every scene comes together to form a game.

The relations between scenes can be found in the ontology diagram below.

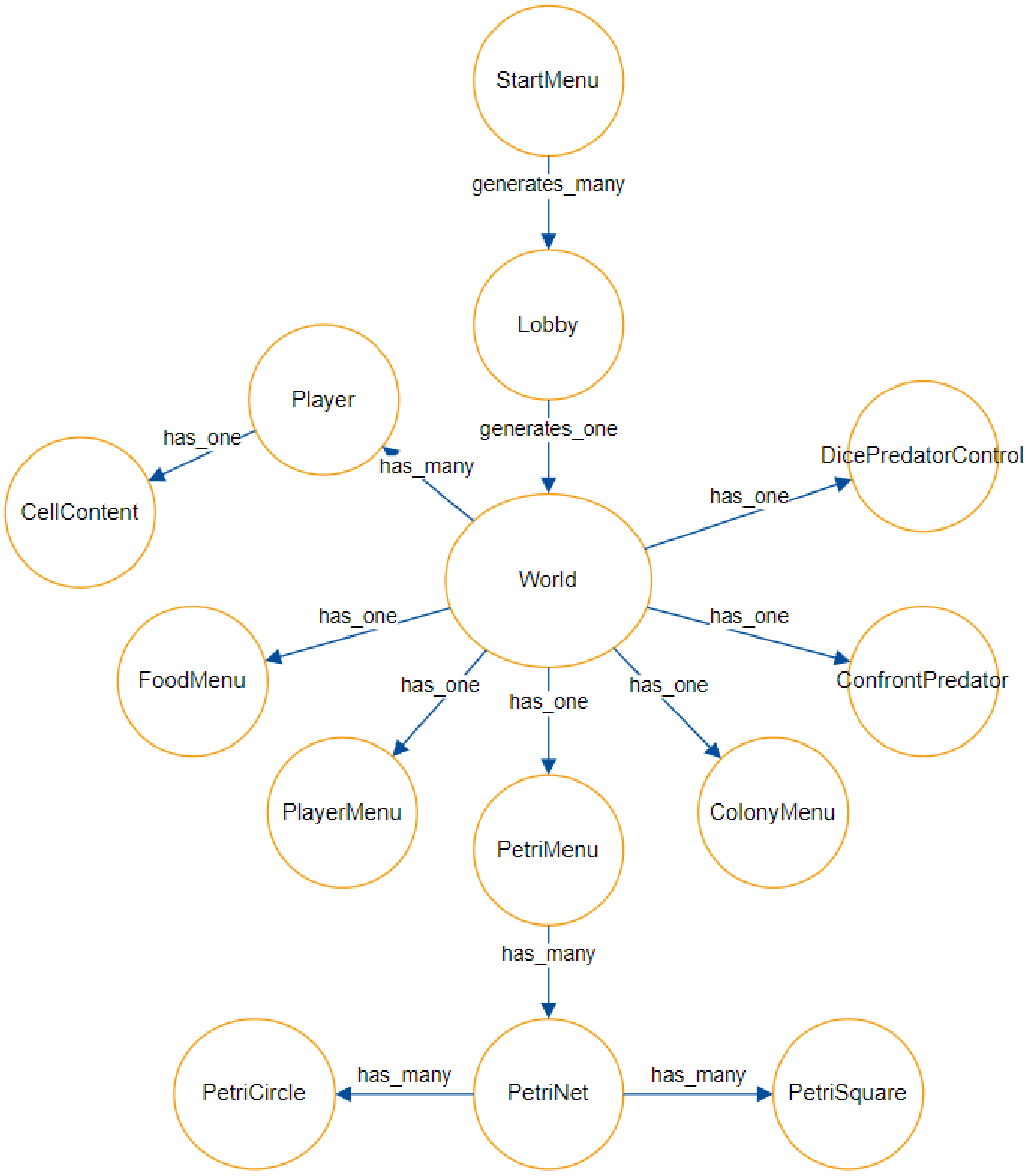


Figure 5.1: Ontology diagram of the game's scenes

Additionally, twenty-eight different scripts are being used to run the game (the vast majority hold the same name as the scene or the node in which they are used). Some of these scripts are very simple and are used for game components that do not require much elaboration, however, some are worth elaborating on:

- **GameManager** - As previously mentioned, GameManager is a script that is autoloaded in order to be used by any scene at any point of the game. Besides allowing the game to store and retrieve information from various scenes it is also responsible for the communication with the server, by sending and receiving the broadcast messages.
- **MQTTClient** - MQTTClient is a script provided by GDMosquitto that allows clients to subscribe to different topics and send and receive messages

from those topics. In the case of this game, the topics are the servers to which the players are joining. Additionally, this script also receives every broadcast message sent by players and calls functions from other scripts in response to the message according to the action code that is sent within the broadcast,

- **Play** - This is the script used in the StartMenu scene. It sends a player to the Lobby scene as either a host if the player created the game or a non-host if the player joined a game. It also creates and gives a unique ID to each player.

- **Dice** - This script is used to show and control the dice that the players throw at the start of every round. It generates a random number between 1 and 6 for each player once they click the dice button and stores the number of actions every player has at any point. This script also works in conjunction with the Player and TurnManager scripts to switch players once a player presses the button to switch turns, as well as hiding and showing the correct components of the game to each player when this is done.

- **DicePredator** - DicePredator is a script used to show and manage the dice thrown by the players when players encounter a predator or a thief and possess at least one action that allows them to confront the animal. Besides the throw of the dice, it calculates the total amount achieved by every player, checks which animal the players are facing, which action they have defined, and if the players achieved a high enough number to confront the animal. It is also used for a second throw of dice done by the player who encountered the animal when players lose the confrontation with a thief, which results in the number of food removed from the burrow by a raccoon or tunnels destroyed by a rat. Finally, it is also responsible for selecting which tunnels are randomly destroyed when players lose to a rat.

- **PetriMenuActions** - This script has two main purposes. The first one is to open the slider which shows all the conditions and outcomes the players have unlocked thus far when the players select a circular element of a Petri-net and, likewise, open the slider with all the unlocked actions when a player selects a square element of a Petri-net. The second and most important function of this script is to check if players are able to perform an action. Every time a script needs to verify if a player can perform a specific action it runs a PetriMenuActions function with a specific number value correspondent to that action, and the function will, based on that number, check if any of the Petri-nets in the entire Petri menu has that action defined.

- **SearchCellButton** - SearchCellButton is one of the most important scripts in the game. It is responsible for showing the content a cell has hidden when a player selects to see it, and it is also in this script that the procedural content generation algorithm developed for the game resides. This algorithm is responsible for creating the game arena by initially defining the content inside each cell. Further elaboration on the algorithm will be seen below.

- **CellContentControl** - This script is used to check when players click on a piece of content shown by a cell. The script checks which content was selected and which type it is. Once the type is known, the script runs through a series of conditions to verify if the player can in fact interact with that content by checking if it is the player's turn, if the player has enough actions, if the action is already defined, etc., and modify the game accordingly.

- **Control** - Control is the script that allows players to move in the arena by creating the animation that slides characters when moving, as well as highlighting the cells to which the players can move.

- **TurnManager** - TurnManager is a script used to switch turns between players. When a player selects to switch turns, the script finishes the current player's turn and signals the next player that it is their turn to start playing. Additionally, all Player scenes are children of the Node TurnManager, which is the Node that utilizes the TurnManager script.

- **Player** - The player script serves several purposes. First, it works with the TurnManager script to start the current player's turn. Second, it aids other scripts in setting the food inventory for each player. Finally, it checks if the player can actually move to a cell when prompted to. While the Control script previously mentioned moves the players from cell to cell, it does not check for any conditions other than if the player has finished moving, all of the verification for player movement is done in the Player script. This verification includes checking if the player is in their turn to play if the players have defined the action to move, if the player has enough actions to move, if any of the player's menus are open, etc.

5.3 The Procedural Content Generation Algorithm

One of the goals of this project was to implement a procedural content generation algorithm to increase the game's replayability and study its effects.

Since the game arena is fairly simple in shape and size and the content that the players can find is already set under certain rules in the original version of Meerplay, the algorithm for content generation would mostly focus on generating interesting and viable initial board configurations. The algorithm developed for this purpose is based on several random choices that are made to respect several game constraints.

The algorithm goes through five initial steps in order to create the board configuration:

1. **Unique Number Generation** - When a new game is started, the host's (the player that created the game) operating system's time in milliseconds is stored and sent to every player in that game. This value is subsequently used as the seed for a random number generator. Since the seed number is almost impossible to replicate, every time a new game is created the content generated will be different, however, all players in each game are sent the

same number and set the same session seed, so the content is the same for every player inside the same game section.

2. **Arena Generation** - The game's arena is created as a seven-by-seven square made up of forty-nine cells. These dimensions are not changed by the algorithm, so every game instance has the exact same arena size.

   The two leftmost columns and the two uppermost lines form the predators' zone. This area was chosen for the predators due to being the farthest from the player's starting area, which is in the bottom right cell of the arena.

   A visual representation of these areas can be seen in the image below.

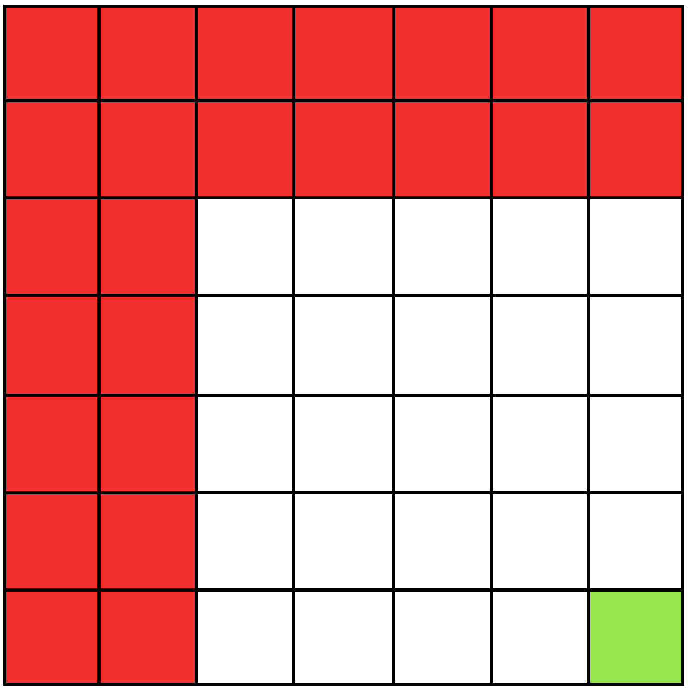


Figure 5.2: Representation of the predator zone (red) and the starting cell (green) in the arena

   The coordinates of these cells are stored in a list and a random number generator is used in order to pick an index, and therefore a cell, from the list. In this area, five different random cells are set with either a tree or a bush. If the cell is at the edge of the arena it will have a tree and if it is in the inside section of the predator zone then it will have a bush. The reason for this distinction is due to the tree tiles being very large and hiding the tiles behind them, so they are set at the edge of the arena in order to not cover anything.

   If a cell is picked by the random number generator more than once then the generator will pick a new number until five different cells have been chosen from the list.

   After the vegetation cells are set, the algorithm chooses two different random cells on any position on the arena, other than the edges, to place a rock. If the chosen cell already has a rock or a bush, then the algorithm generates a new number, and a new cell is chosen until two rocks are placed on the arena.

A visual representation of this area can be seen in the image below.

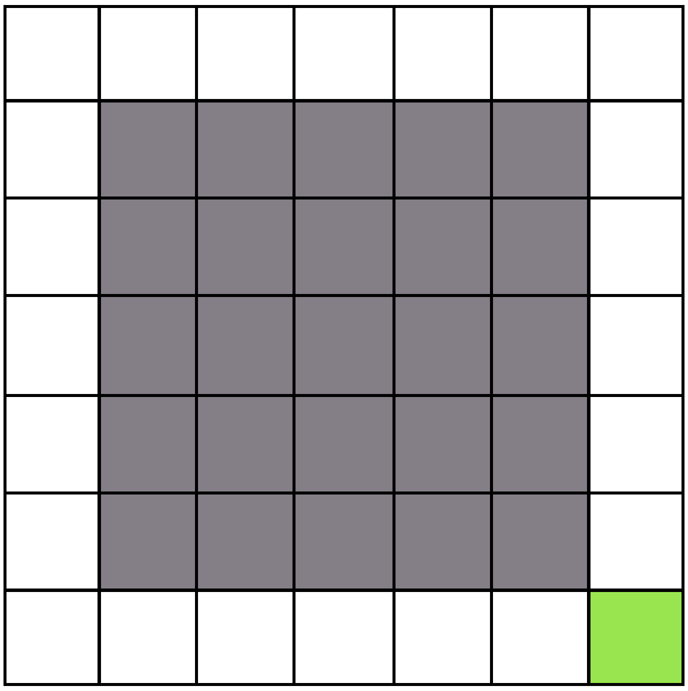


Figure 5.3: Representation of the rock zone (grey) and the starting cell (green) in the arena

Since rocks are cells to which the players can't move, they are only placed on the inside area of the game arena to avoid situations in which the rocks block a section of the arena. A more complex approach was first tried where rocks were placed anywhere on the map and at random quantities, however, the algorithm needed to check if every single cell was connected to the entire arena proved to be time-consuming and the idea was pragmatically discarded.

Finally, once both the five vegetation cells and two rock cells are set, all of the remaining forty-two cells are set with one of three regular different tiles. Each one of these tiles also has three different elevations to give some resemblance to the irregularity of the savanna's ground, which brings the total number of different tiles to nine.

Until this point, the algorithm has only tangled with the visuals of the game, by creating a one-of-a-kind arena based on a unique number. In the following stages, it will start to generate content that greatly changes the game experience in each play section.

3. **Predator Generation** - Once all the arena tiles are set, the vegetation cells, previously created are stored in a list. Since there are three different predators (snake, wolf, and eagle) and each predator appears three different times in a game, the algorithm needs to select nine different cells from the predator zone to place these predators, however, each vegetation cell also needs

to hide a predator, and those cells are already set and stored in a list. With five vegetation cells having already been set, five predator cells are already defined, so the algorithm picks four other cells from the predator zone (taking into account that these cells can't be vegetation cells again or rock cells) and stores these cells in the previously mentioned list.

Finally, with all predator cells defined by the algorithm, three instances of each predator are placed into the cells, with each cell hiding only one predator. Three different lists are also created to store the cells of each predator.

4. **Thief Generation** - Afterwards, the algorithm selects six cells from the nine cells formed by the inside three-by-three area of the arena to place three rats and three raccoons, as seen in the following representation.

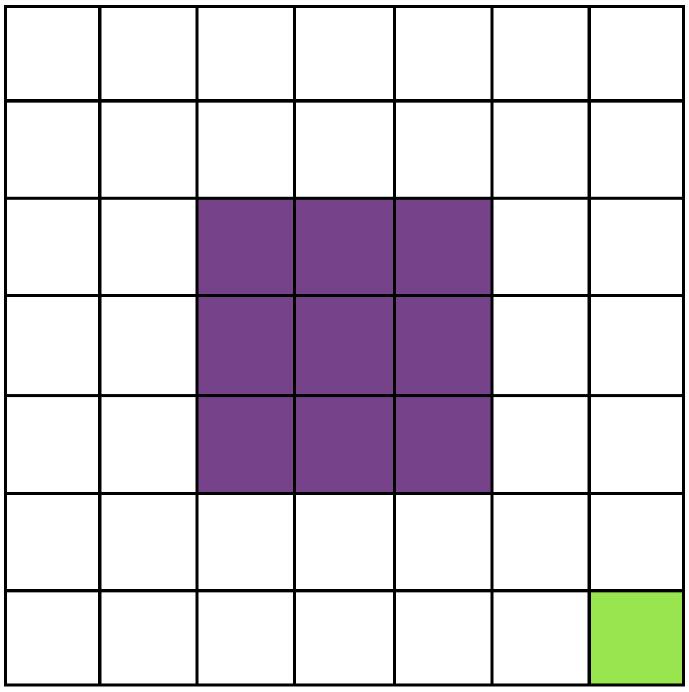


Figure 5.4: Representation of thief zone (purple) and the starting cell (green) in the arena

The middle of the arena was chosen since it gives some space between the player's starting area and the thieves, and doesn't conflict with the predator cells.

Finally, the cells of each thief are stored in two different lists.

5. **Bag Generation** - The final pieces of content that need to be set before the players can interact with the game are the bags. For each player in the game, the algorithm selects two different cells from the entire arena (other than the rock cells and vegetation cells), stores those cells in a list, and hides a bag inside those cells, whilst also making sure that no more than one bag is stored in a single cell.

All of these five steps occur between the host starting a game and the game switching from the lobby to the actual play-time arena. However, two final pieces of content still need to be added to the arena: food and ground.

Instead of defining what type of content is hiding in every cell as soon as the game starts, the algorithm waits for a player to search a cell and only then places the content inside it. When a player decides to search a cell, a new seed is set based on the unique number sent by the host and the coordinates of the cell the player opened.

$$\mathbf{Seed = unique\_number + x * 10 + y},\ x \in [5,11],\ y \in [-3,3]$$

This specific formula was chosen because it allows the use of both the x and y positions of the player's cell without reaching results that could be duplicated with different formulas. This makes the seed for each cell different, which allows for different content to be generated on different cells, but the same cell will always show the same content inside the same game instance.

Afterward, the algorithm will check if the cell the player searched is a nest tile or a tunnel tile. If it is a nest tile the algorithm will show a nest and if it is a tunnel tile the algorithm will show a tunnel, however, if the cell the player is in is neither of these tiles, the algorithm will check if it is hiding one of the three predators, one of the two thieves, a bag, or if it is a vegetation tile (in this order), and show that specific type of content. If none of these previous checks were verified, then the algorithm either shows food or rich ground. Although there are ten different types of food and only one type of rich ground, the algorithm is set to have about a 66% chance of picking food and a 33% chance of picking rich ground.

Once the first piece of content hidden in the cell is shown to the player, the algorithm will check which type of tile the player is in. If the player is in a bush, the algorithm will then generate four more pieces of content. If the player is in a tree, the algorithm will generate eight more pieces of content. Finally, if a player is not in a vegetation tile, the algorithm will generate one or two more pieces of content. Regardless of the type of tile the player is in, the algorithm will repeat the step of checking if it has predators, thieves, bags, or none and generate the appropriate type of content.

The following diagram depicts a visual representation of the algorithm's decision flow.

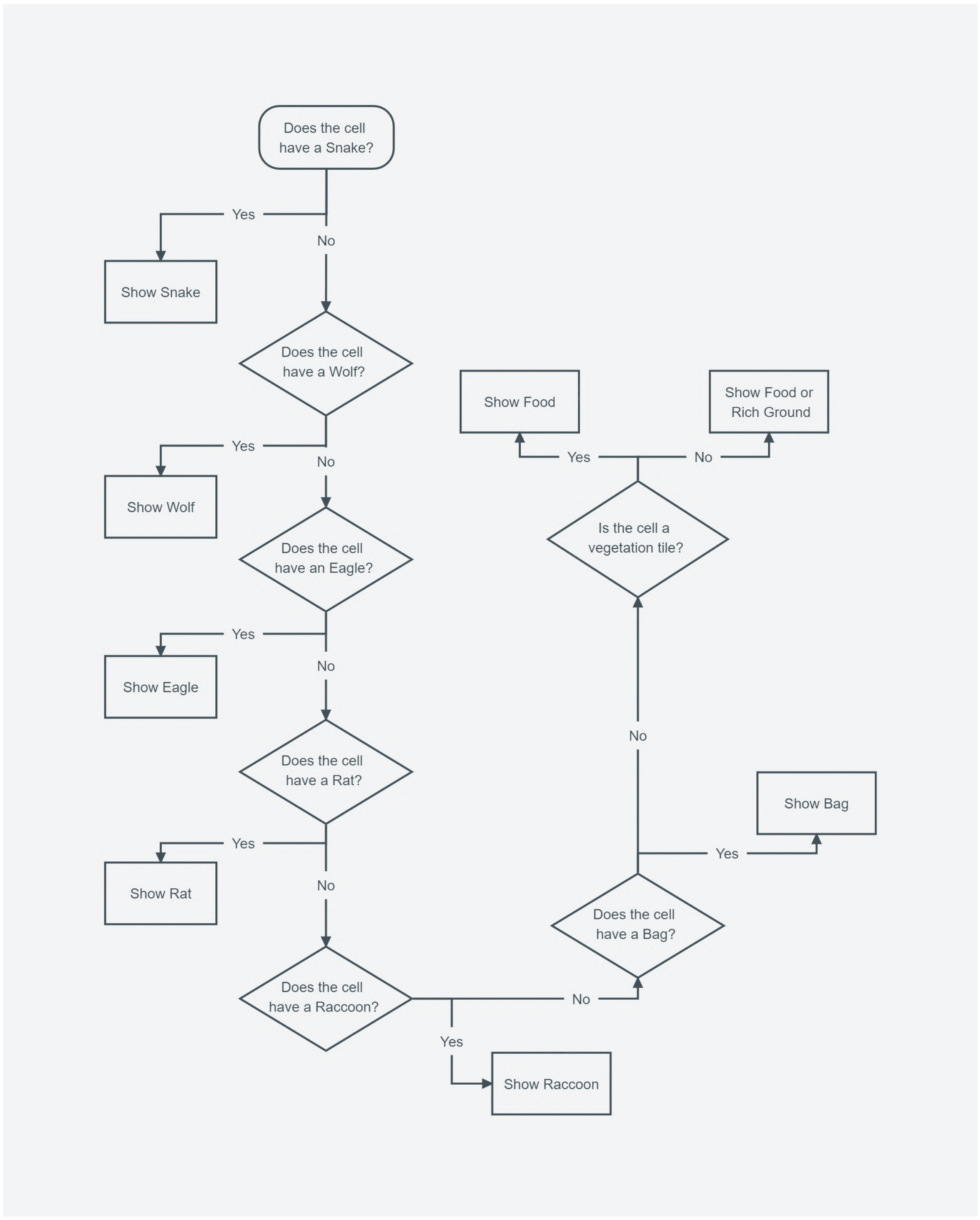

Does the cell have a Snake?
Yes
No
Show Snake
Does the cell have a Wolf?
Yes
No
Show Wolf
Does the cell have an Eagle?
Yes
No
Show Eagle
Does the cell have a Rat?
Yes
No
Show Rat
Does the cell have a Raccoon?
Yes
Show Raccoon
No
Does the cell have a Bag?
Yes
Show Bag
No
Is the cell a vegetation tile?
Yes
Show Food
No
Show Food or Rich Ground

Figure 5.5: Visual representation of the algorithm's flow

5.4 Integration with the Erlang Multiplayer Server

One of the goals of this project was to take advantage of the digital medium and allow players to enjoy the game together from different devices. Such a goal meant that the game needed to be integrated with an online server that allowed communication between several machines and managing the player entry and coordination of the game state.

The server used to achieve this is an Erlang Server developed by João Calhau as part of a separate project taking place at the University of Coimbra. The server was not created specifically for Meerplay, it was created for another game that also needed online multiplayer to be implemented. However, since this server was capable of hosting both games simultaneously on different ports and since it was already built and proven to be functional, it was picked as the server to be reused for Meerplay's online multiplayer support.

The way this Erlang Server works is by receiving messages from a player and sending those messages back to every player, including the sender. This is done for every action a player takes that takes the game into a different state that all players' game instances need to be aware of. Instead of the game immediately performing a function when a player tries to do something, it sends information about that action to the server and the server sends it back to every single player. Only then, after having received the message, does the game perform any function that is required, this ensures that every player's game is doing the same thing simultaneously. For example, if a player tries to move to a different cell, the game won't immediately move their character, instead, it will first broadcast a message to the server with information such as which player is trying to move, which cell they are trying to move, and, most importantly, an action code that denotes that the action the player is trying to perform is moving the character. This message is then sent by the server to all the players, who simultaneously move the current player's character to the same cell.

In order for the players to receive messages from the server, however, an MQTT client had to be created. This was achieved using a Godot plugin called GDMosquitto, which offers the tools necessary to both create an MQTT client as well as subscribe to different topics.

Communication between the players and the server client works by creating an MQTT client for each player as soon as they open the game using the MQTTClient script. Once a player sends a request directly to the server to either create or join a game through the GameManager script, the server will either answer with an error or with a success message. In the case of success, the message also contains the ID of the server ("Server_1", for example) in which the game will take place. Once receiving the name of the server, the MQTT client that was created when the game was first opened will subscribe to a topic that contains the ID of that server. By doing this, every time the server sends a message to that topic, every player that subscribed to that topic will receive that message. This guarantees that, for example, players subscribed to "Server_1" will receive messages sent by other players to the topic "Server_1". Finally, when a client broadcasts a message to the Erlang Server, the server will send that message back to the MQTT client of every player on the same server as the sender. The MQTTClient script will then check the content of the message for its action code and act accordingly.

Although this server is built to answer ten different types of requests, only five were used for the game:

- **/create_game** - This request is sent by a player who wants to create a new game session of Meerplay. The message sent to the server needs to hold the client's ID, which is created by the Play script once the player selects the button to create a new game, the player's avatar, and their nickname, although for Meerplay only the client ID is relevant. Once the Erlang Server receives this request it will reply with either a response code of 400, which means an error occurred, or 200, which means the creation of the server was successful. In the case of success, the Server also sends the ID of the server the game will take place on.

- **/join_game** - The request to join a game is very similar to the previous one, however, the message body also needs to include the ID of the server to which the player is trying to join in addition to their own ID. When the server replies, it can again give a response code of 400, which means an error, or 200, however, two types of messages with a response code of 200 can occur. If the code is a 200 but the message states that the server no longer exists the player does not join the game, but if the code is a 200 and the message states that the server does indeed exist then the player joins that game. If the reply is a 200 positive then the message will also contain the host's ID as well as all the players' IDs (the players' avatars and nicknames are also sent in the message but this is once again not relevant for the game).

- **/start_game** - This request is sent from the waiting lobby of a game by its host in order to start the game. Once more, the Server reply can be a 400, a 200 negative, or a 200 positive. If the reply is a 200 positive the game will start.

- **/heartbeat** - Heartbeat is a request that is sent by each player to the server every five seconds once they either create or join the game. This request serves to notify the Server that the player is still online, otherwise, the Server will kick the player.

It is important to note that no requests were used to signal the Server when players leave a game or get outright disconnected. This was not done since the primary goal of this project was to develop a functional game ready for testing and not yet for official deployment. These requests can be added and the game's code expanded further in the future in order for it to be ready for any situation.

The final request that is sent by the players to the Erlang Server is **/broadcast**. This request is used by the players to signal to others that they are trying to perform something that will affect everyone's game. Processing these messages in order is how the game state is kept consistent across players.

The message sent to the Server needs to include, the server's ID, the player's ID, an entry labeled "region", an entry labeled "simValues", an action code, and finally, a message. The entry "region" is used when players are distributed between regions or sectors, which does not apply to Meerplay, instead, this entry was used to send any necessary data through the broadcast. When the server replies, the MQTT client of each player will receive the broadcast's action code, the data stored in "region", the sender's ID, and the message.

In order for the MQTTClient script to parse different messages accordingly, different action codes were set for every type of message that the players could broadcast during a game, as seen in the following table:

Action Code	**Message**	**"Region Data"**
1	"Update Lobby"	Player ID
2	“Start Game”	“Start Game” Unique number used for random number generation
3	"Set Player's Action Number"	Number of actions the player got from the dice role
4	"Change Player Turn"	
5	"Move Current Player"	Cell to which the player moves and a flag that signals if the character is moving because of the player or a predator encounter
6	"Give Actions"	Number of the player giving actions and number of the selected player
7	"Give Bag"	Number of the player giving a bag and number of the selected player
8	Set Condition/Outcome"	Number of the selected Petri Net, number of the selected circle, and number of the selected condition/outcome
9	"Set Action"	Number of the selected Petri Net and number of the selected action
10	"Prepare Next Turn"	

Table 5.1: Broadcasts list I

Action Code	**Message**	**"Region Data"**
11	"Current Player Collected Bag/Food"	Number of the content button selected, type of food or bag picked up, flag that indicates which storing actions the players' have defined (hands, bag, both, or none)
12	"Current Player Entered Nest"	Number of the content button selected
13	"Current Player Opened/Closed Content"	Bool that indicates if the current player has a cell's content opened or closed
14	"Current Player Opened/Closed Petri Menu"	Bool that indicates if the current player's Petri Menu is opened or closed
15	"Current Player Opened/Closed Food Menu"	Bool that indicates if the current player's Food Menu is opened or closed
16	"Current Player Opened/Closed Colony Menu"	Bool that indicates if the current player's Colony Menu is opened or closed
17	"Show Petri Menu"	
18	"Current Player Created a Nest"	Coordinates of the cell
19	"Current Player Created a Tunnel"	Coordinates of the cell
20	"Current Player Deposited Food In the Nest"	All the food being deposited in the burrow

Table 5.2: Broadcasts list II

Action Code	Message	"Region Data"
21	"Found a Predator"	Type of predator/thief and its location
22	"Pressed Predator Button"	Number the player got from the dice role
23	"Remove Tunnels"	Number of tunnels to be removed and list of all tunnels
24	"Delete Rat" or "Delete Raccoon"	Index of the thief in the content (to be deleted)
25	"Define Cell Content"	Dictionary with all opened cells and excluded values for those cells
26	"Lost Food"	Quantity of food lost
27	"Confront Predator (Choice)"	Predator type and confrontation option
28	"Current Player Opened/Closed Player Menu"	Bool that indicates if the current player's Player Menu is opened or closed
29	"Show New Conditions/Outcomes"	All new conditions and outcomes unlocked by the players
30	"Show New Actions"	All new actions unlocked by the players

Table 5.3: Broadcasts list III

5.5 Game Interface and Gameplay

This section will serve to show and better describe how the game was brought to the digital medium, as well as the added content and the changes that had to be made in order to have a working game prototype in this medium.

The first thing every player sees when opening Meerkat is the start menu. This menu has two buttons, one to create a game and the other to join an existing game.

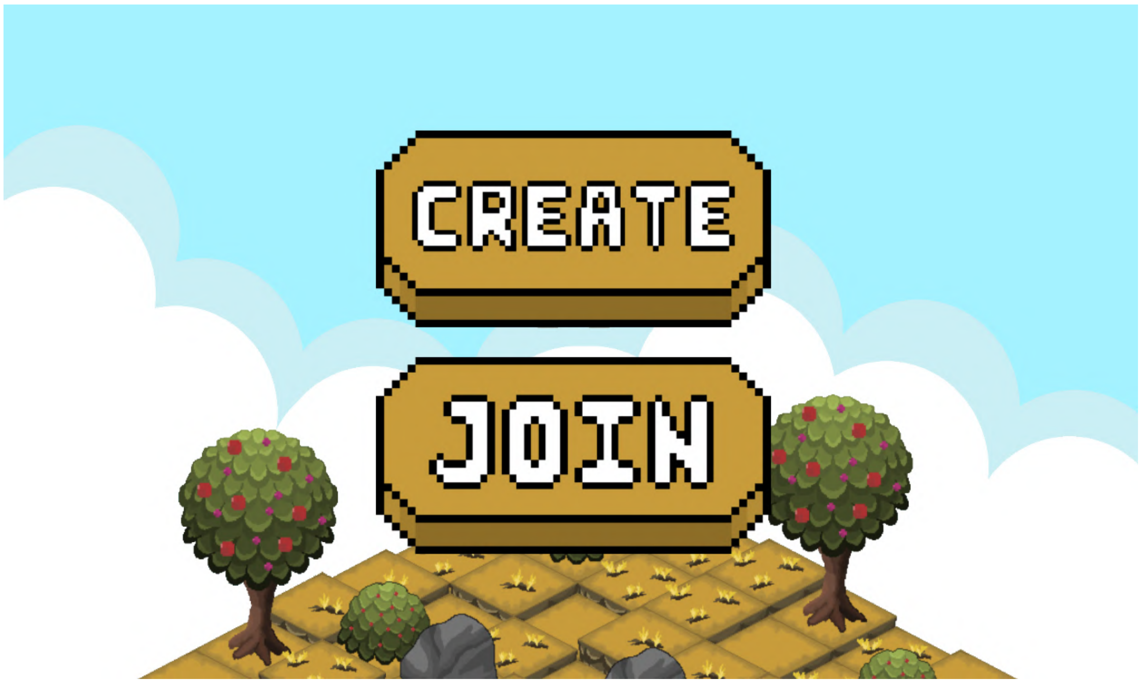


Figure 5.6: Start menu

A player who presses the create game button will be sent to a lobby composed of 6 slots. In the first slot, the face of a meerkat surrounded by the color yellow can be seen, representing the character of the player. There is also a start game button for the host to start the game once he desires. Every time a new player joins that game, the lobby is updated with a new meerkat face with a different color.

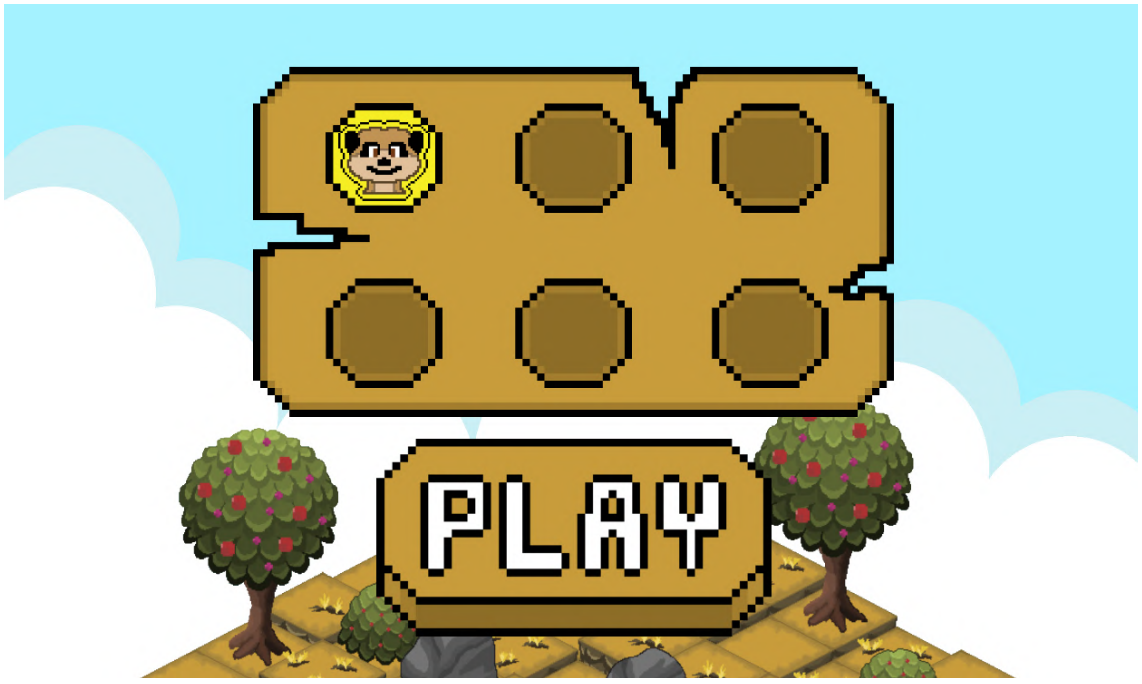


Figure 5.7: Host's lobby

On the other hand, if a player presses the join game in the start menu, they will be sent to an already-existent lobby with at least one other player, the host. This lobby will look exactly the same to all players other than the start game button, which only appears to the host.

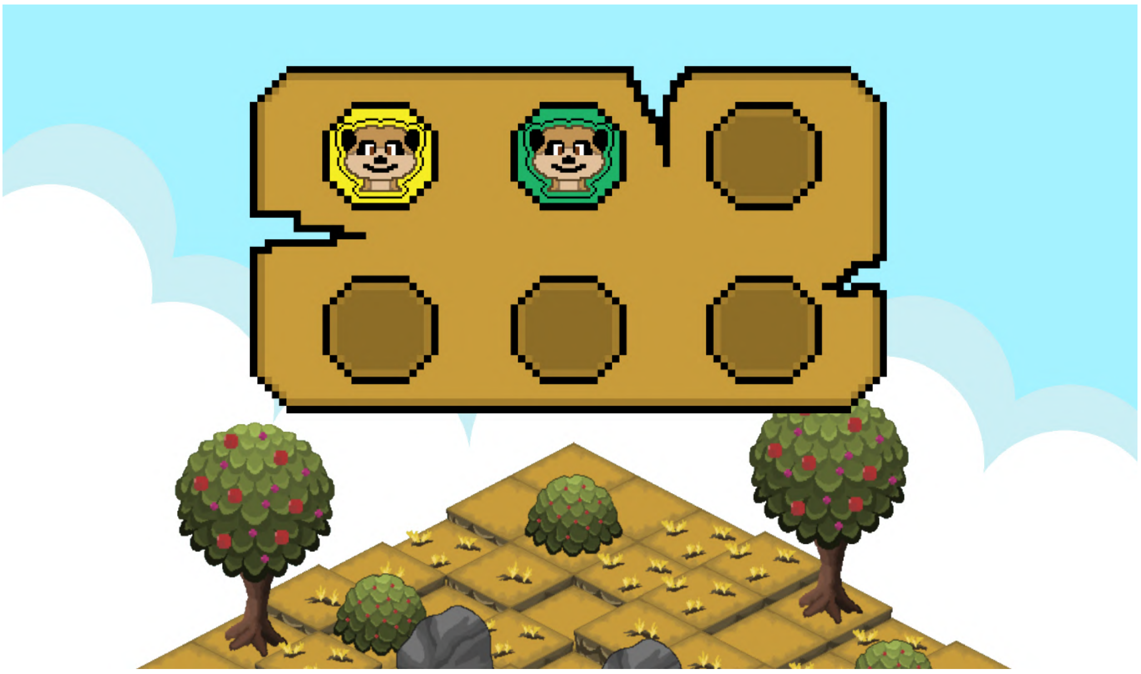


Figure 5.8: Players' lobby

It is technically possible to have several games running simultaneously, since the Erlang Server provides that functionality, however, for the sake of expediting the integration of the Erlang Server with the game and because the gameplay tests will be made one game at a time, the game is currently only able to connect to one instance of online gaming at a time, meaning that as of now, the game cannot be played by several different groups of players, only one. If a player tries to create a game whilst the first gaming section is still running they will be sent to a lobby, but cannot start the game.

When the game starts, all players are sent simultaneously to the game arena, and the Petri menu of each one pops up to signal that something must be made with it. The amount of Petri Nets present in this Petri menu is three times the amount of players present in the game, so in order to show them all in one menu only a few are shown at the same time, and the player needs to slide the menu left and right to see other Petri Nets.

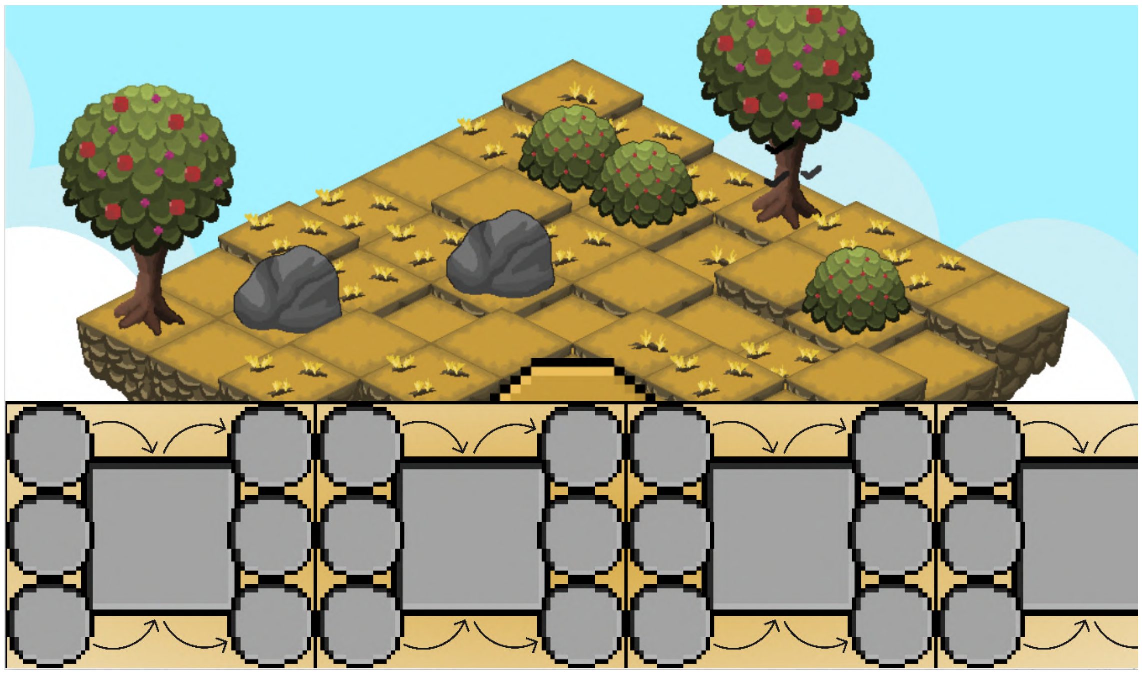


Figure 5.9: Petri Nets menu

Other than the slider menu, there is a key difference here between the video game and the paper game. In the paper game, each player was given one Petri Net at the start and then two more were given to each one throughout the game. This served to limit the number of actions the players could define at once and choose which ones were more important. The reason for all Petri Nets being present at the start of the digital version came from the fact that even though players had access to all Petri Nets from the get-go, they could only use new elements once they were unlocked, which was believed to be enough to limit the possibilities for different actions.

If the players select one of the circles present in one of the many Petri Nets, a new slider menu showing all the unlocked conditions and outcomes will appear, allowing players to select one to be placed in the circle spot they previously selected. A similar menu appears with all the unlocked actions when players select a square from one of the Petri Nets.

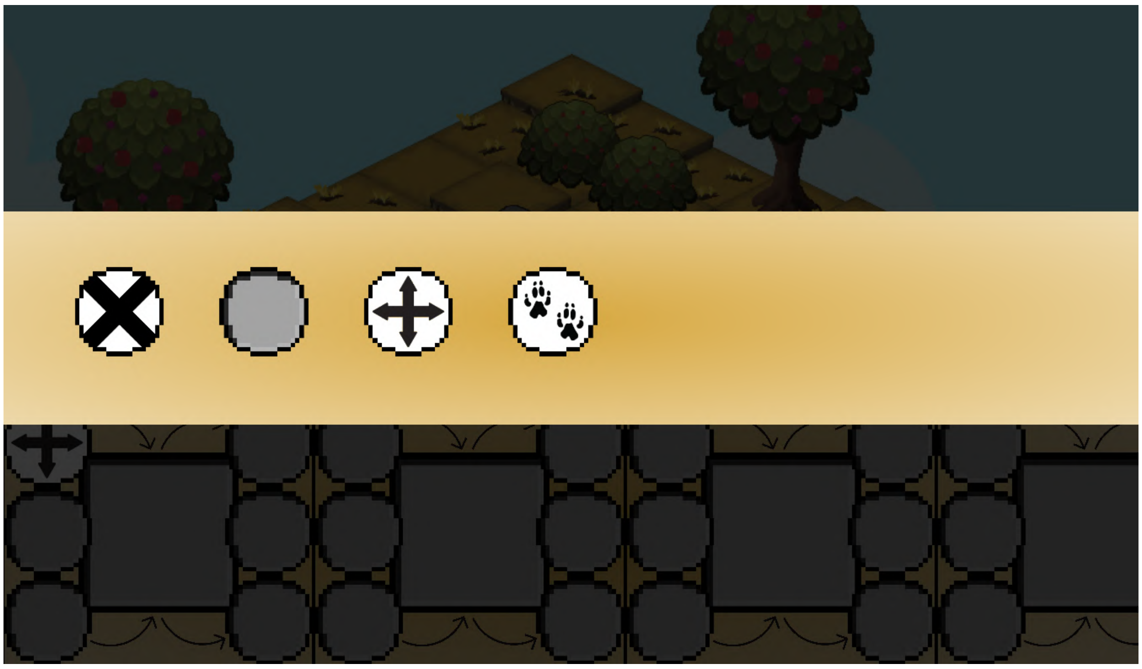


Figure 5.10: Conditions and outcomes menu

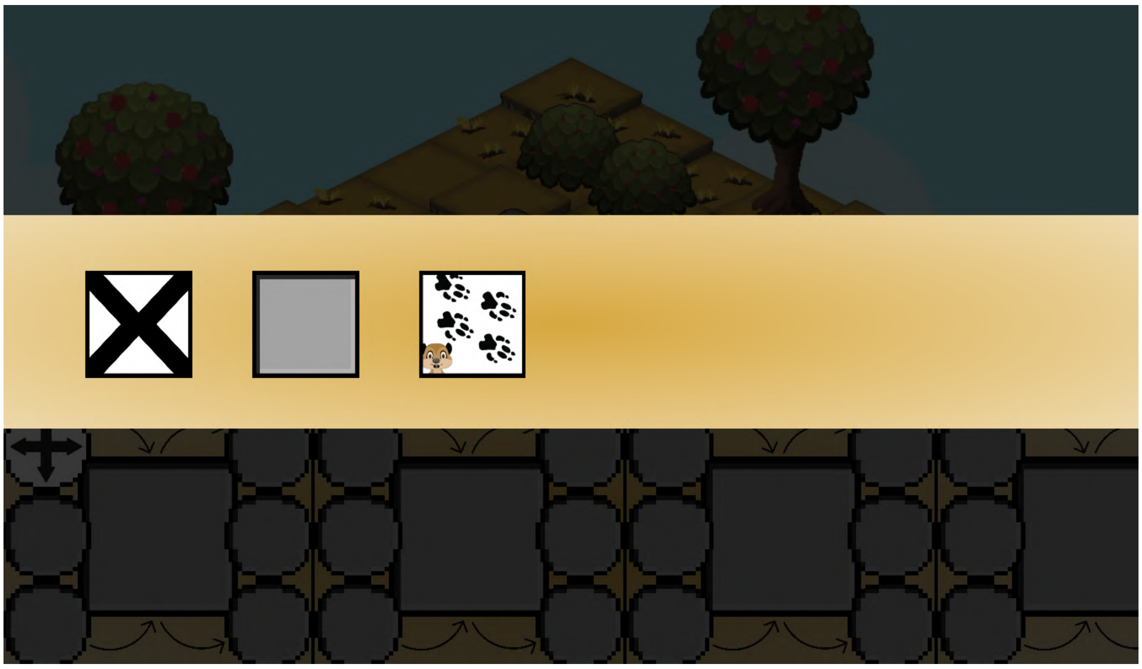


Figure 5.11: Actions menu

At this point in the game, only two conditions/outcomes and one action are unlocked, which are the ones used to allow players to move from cell to cell.

It is also worth noting that all Petri Nets are shared by the players, meaning that when a player changes one element, that element will also be changed in the exact same Petri Net for all players. As previously mentioned, in the paper game each player had their own Petri Nets, which meant that, unless they authorized other players, only they could create actions in those Petri Nets. This was changed for the digital version so that players could more easily see all the actions being created at the same time without plaguing the screen with each player's creations. Furthermore, having the Petri Nets shared by all players could help them to better understand that any action defined in a Petri Net affected all players and not just the player who created it, which could possibly happen if the Petri Nets were separated.

Once a player is satisfied with the actions that were created, they close the Petri menu by pressing the tab on top of it, which reveals a dice on the bottom right of the screen, which when pressed will reveal the number, from one to six, that the player got for that round. The game will remain still until all players "throw" their dice.

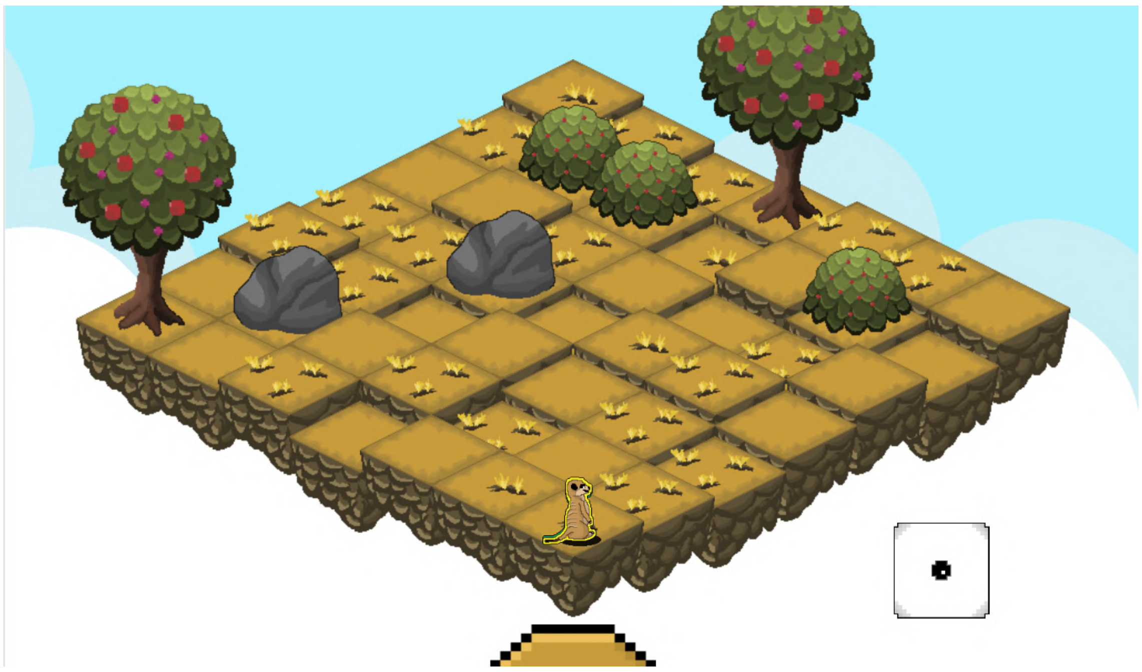


Figure 5.12: Round start dice

Finally, once all players have their number for the following round the game starts, hiding the dice button, keeping the Petri Menu, and revealing an assortment of new elements.

First, on the left side of the screen, players will be able to see a new tab which when pressed, opens their food menu. In this food menu, the players are able to see an image of a meerkat holding food and two empty circles on each side of it, representing the food the player's meerkat is actually carrying. There is also an area below the previous one with even more empty circles, however, this area is darkened to signal to the player that it is not yet unlocked.

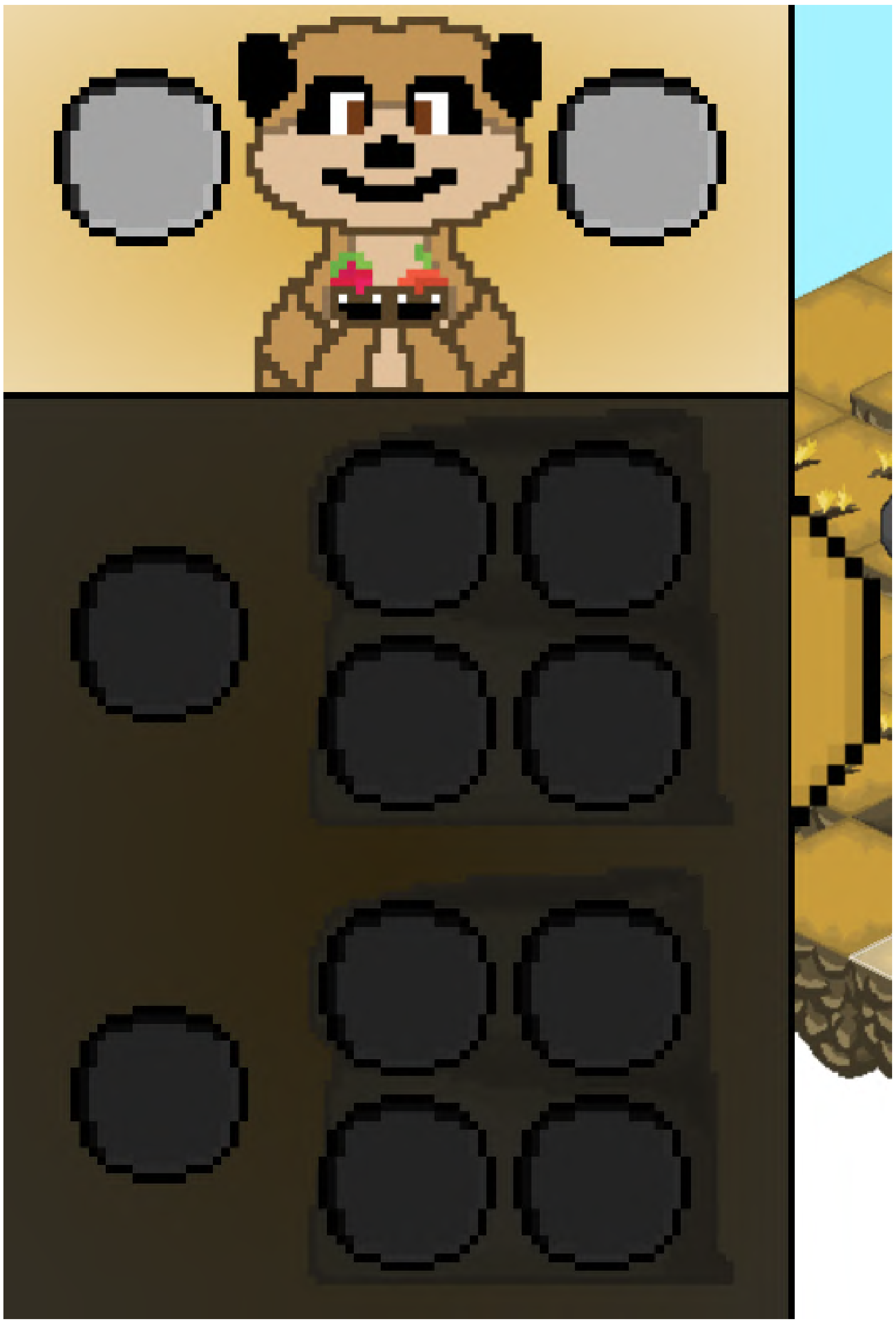


Figure 5.13: Food menu

On the bottom right side of the screen, players can see two buttons. The first one has the exact same meerkat face surrounded by a specific color found in the game lobby to represent the player's meerkat. When this button is pressed, a menu that occupies the entirety of the screen pops up showing all players' meerkats with their distinctive colors. Next to each meerkat is also a number, which represents the number of actions that the players still have for that round, and two bag icons, representing how many bags they have.

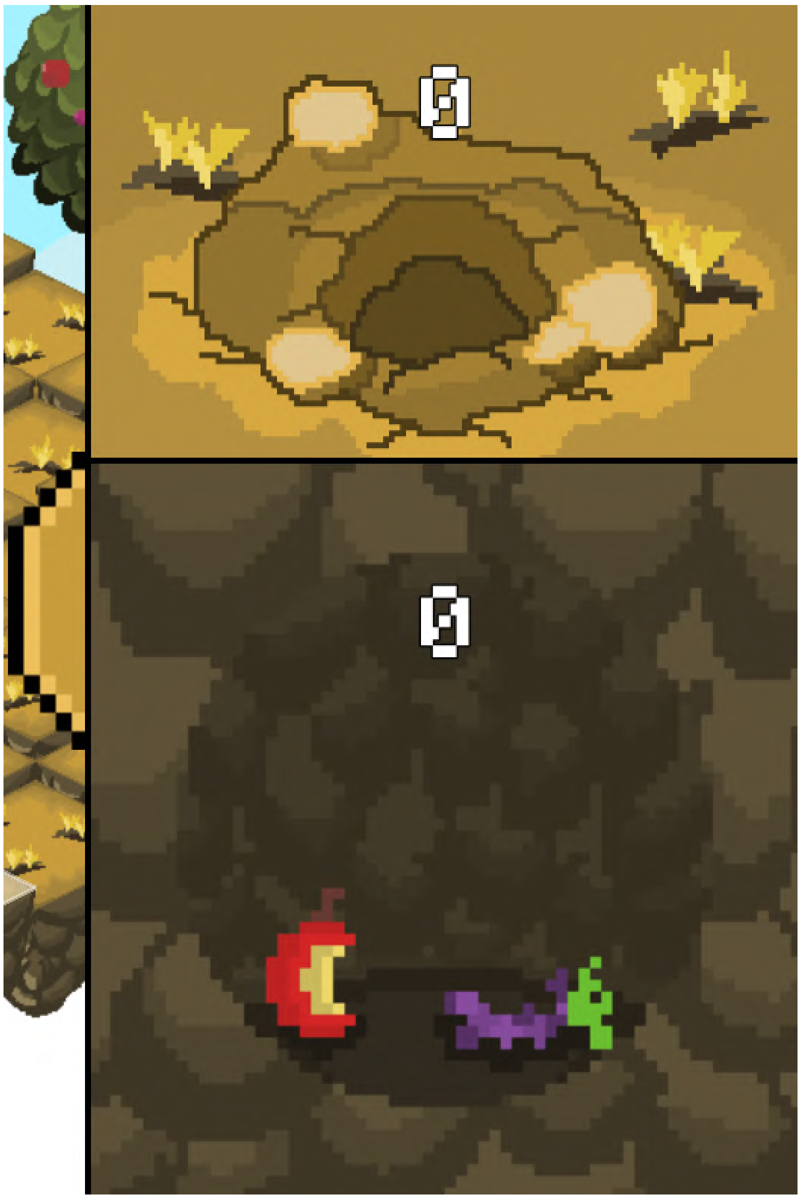


Figure 5.14: Colony menu

On the bottom right side of the screen, players can see two buttons. The first one has the exact same meerkat face surrounded by a specific color found in the game lobby to represent the player's meerkat. When this button is pressed, a menu that occupies the entirety of the screen pops up showing all of the player's meerkats with their distinctive colors. Next to each meerkat is also a number, which represents the number of actions that the players still have for that round, and two bag icons, representing how many bags they have.

The bag's icons are barely visible when the player does not have them to indicate exactly that and become fully colored when the player gets hold of one. Next to both the number of actions and the bags are two buttons with a plus sign.

If a player presses the plus button next to the actions of another player they will give one of their own actions to the selected player, likewise, if a player presses the plus button next to the bags they will give one of their bags to the chosen player. The first one was created to emulate moments where players would share their number of actions in the paper game by simply "giving" some or all of the actions they got for the round. The second one was made so that players could perform the action of sharing a bag.

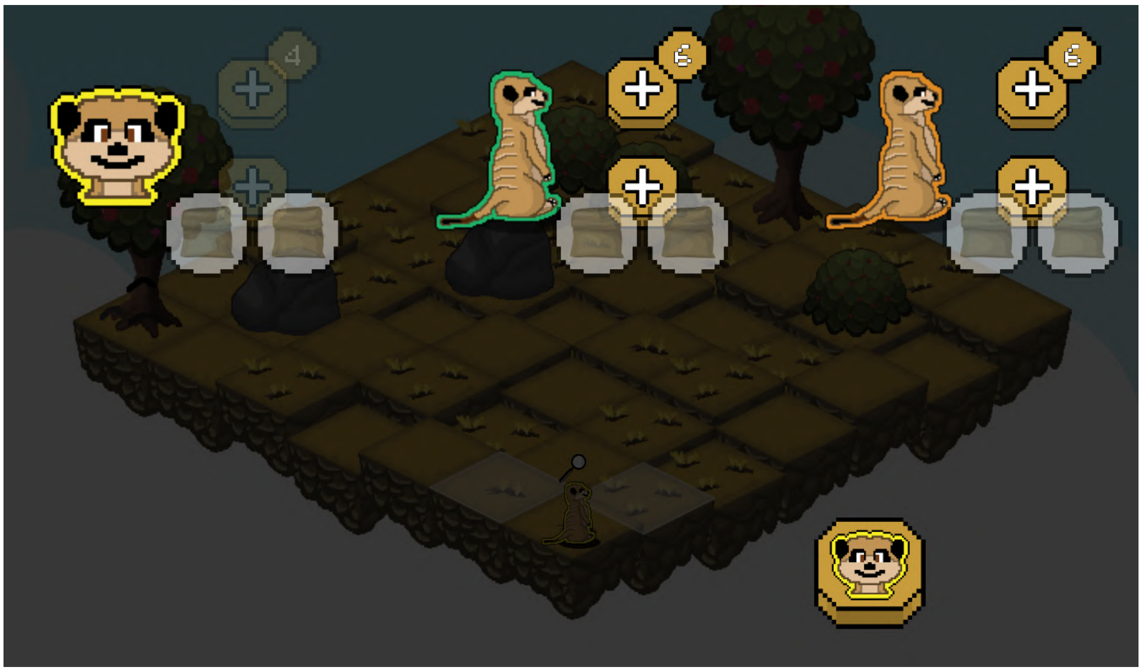


Figure 5.15: Player menu

All of the meerkats in the previous menu are shown in their full body sprite, whilst the player's own meerkat is represented only by its face, to help distinguish it from the others. Additionally, and to further help the players distinguish their character from the others, their plus buttons are set to be barely visible and unresponsive.

In order to close this menu players need to click the button with the meerkat face at the bottom right of the screen again.

Back to the game arena, next to the button with the meerkat face, there is a button with a check sign and a number on top of it. The number represents the number of actions the player still has and clicking on the button itself signals the game that the player is done and will end their turn, starting the next one. When a player has zero actions, the number icon on this button will start to flash to indicate to the player that he cannot perform any more actions and should finish their turn. This button only appears to players when it is their turn to play, as can be observed in the two images below.

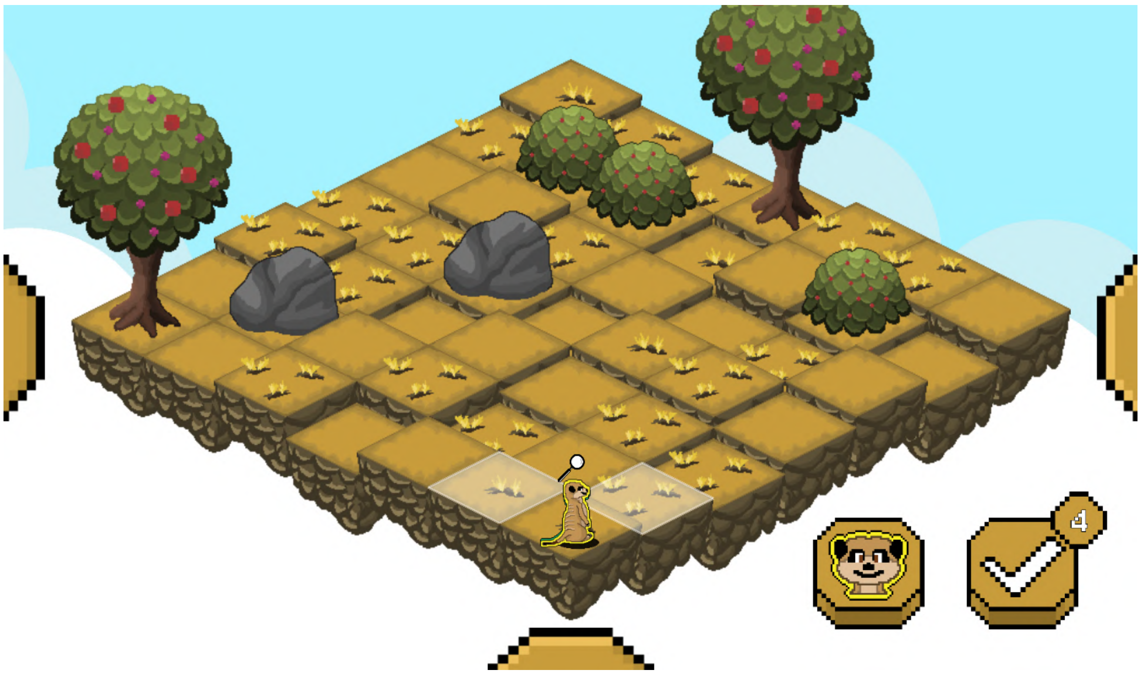


Figure 5.16: Screen during a player turn

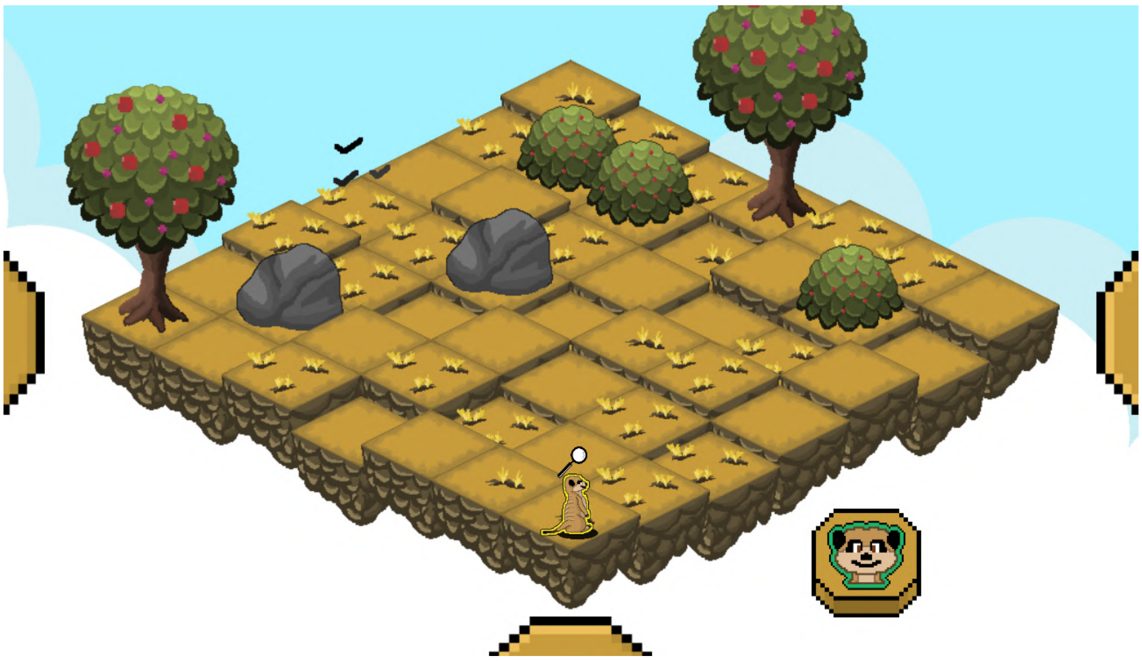


Figure 5.17: Screen when a player is waiting for their turn

Players can select to change their turn to the next player even when they have available actions. This does not end their turn for the whole round, instead, the turn cycle will eventually go back to them as long as they have actions. The turn cycle always skips the players with zero available actions since they cannot perform any actions, however, other players with actions can still give their own to players with zero actions, who will then be set back to the turn cycle rotation. Additionally, this cycle is defined by the order in which players enter the game lobby and a round does not end until all players have expended their own actions, hence why if a player skips to the next player while still holding actions, the turn cycle will eventually go back to them until they have used all of their actions and only then will the round end.

The turn cycle order is always maintained, but at the beginning of every round, it continues from the last active player. Because of this, the first player of each round is always the one next to the previous round's last player. When a round finishes, the screen goes back to the same state as the one seen when the game first started, by popping the Petri menu and hiding all others.

The players are also able to see their characters on the starting cell of the arena, with each meerkat having a different colored outline equal to the one in the meerkat's face button to denote which is which. Since the tiles of the arena are relatively small for the size of the meerkats, when more than one meerkat is in a cell only one can be seen at a time (if they are all standing up or lying down), with the others being hidden behind it.

Other than the very beginning of the first round, which has all the meerkats standing up, when a player finishes their turn their meerkat will change from being upright to lying down on the floor. This further helps determine which character is the one playing.

When a player is in their turn to play, they will find a magnifying lens icon on top of their meerkat character. If the player clicks on this icon, the content hidden in the cell will pop up one by one under the character. This is the equivalent of a player lifting the paper and hiding the cell content to reveal it in the original game. The amount of content hidden in the cell is either two or three for regular cells, one for burrows and tunnels, five for bushes, and nine for trees. Examples of these different number of element representations can be observed below.

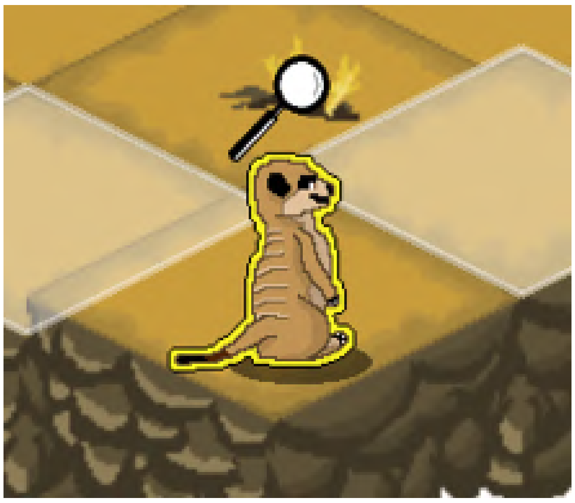


Figure 5.18: Magnifying glass icon to search the cell's content

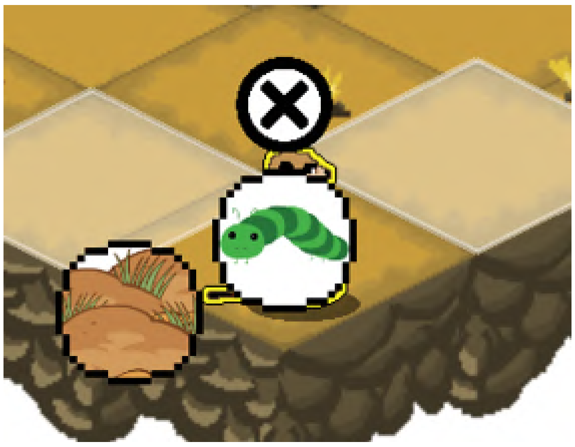


Figure 5.19: Player found two pieces of content in a cell

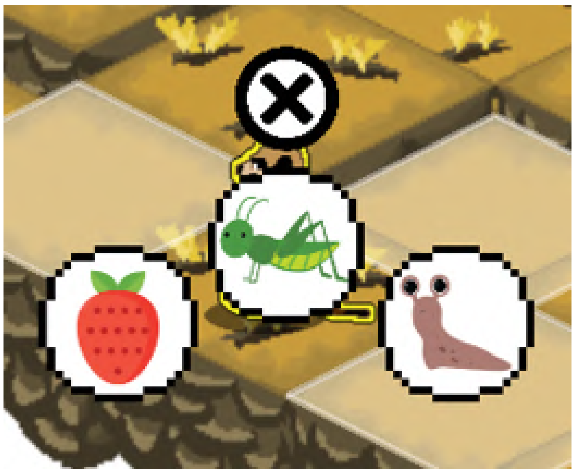


Figure 5.20: Player found three pieces of content in a cell

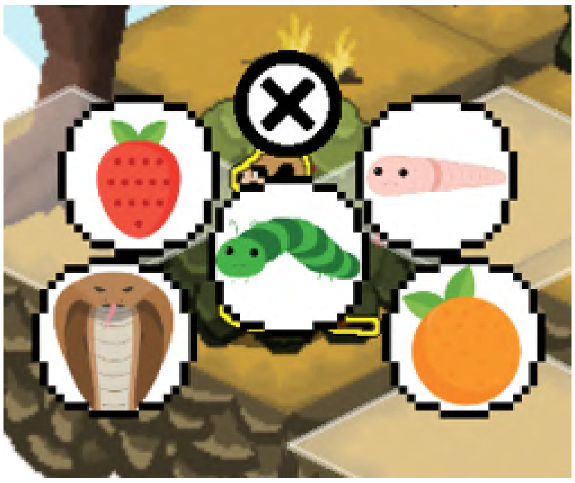


Figure 5.21: Player searched for content in a bush

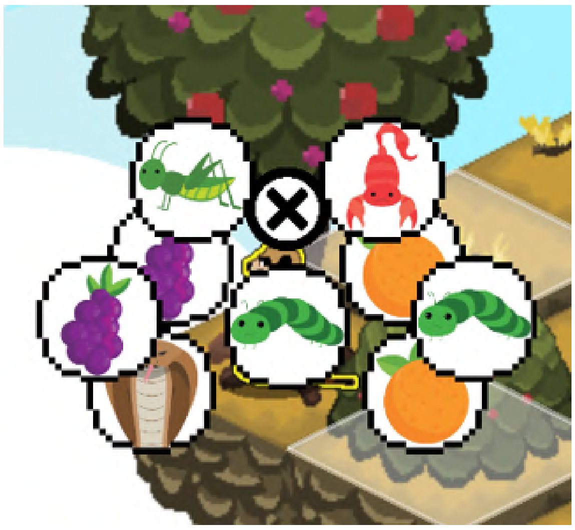


Figure 5.22: Player searched for content in a tree

Players who are waiting on their turn can also click the magnifying glass above the character of the player who is playing, however, the content will only appear until the player who is actually playing opens the cell to reveal it at least once. Until then, nothing will pop up. This was done in order for players to be able to see exactly what the active player is seeing and not more.

Every time a new piece of content is discovered, new conditions, outcomes, and actions that involve said piece of content are unlocked and shown to the player one by one. First, the conditions and outcomes are shown, then the actions. This serves to mimic the part in the paper gamer where Pinto would give the players new conditions, outcomes, and actions made out of paper to the players when they found new content.

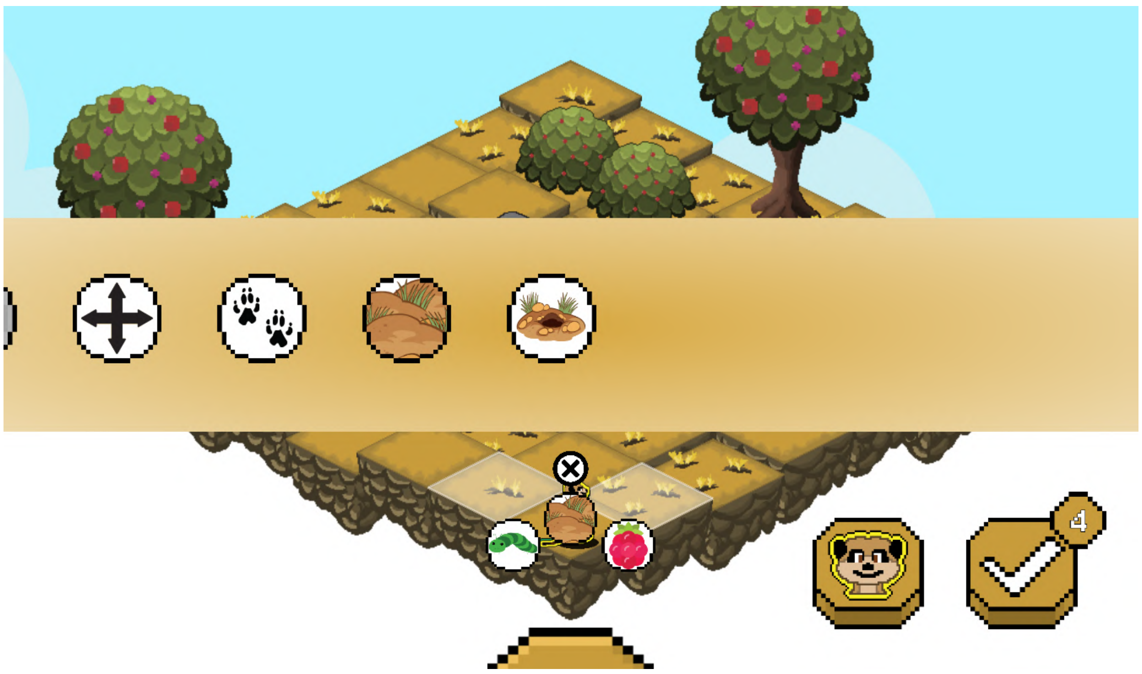


Figure 5.23: New conditions and outcomes are unlocked

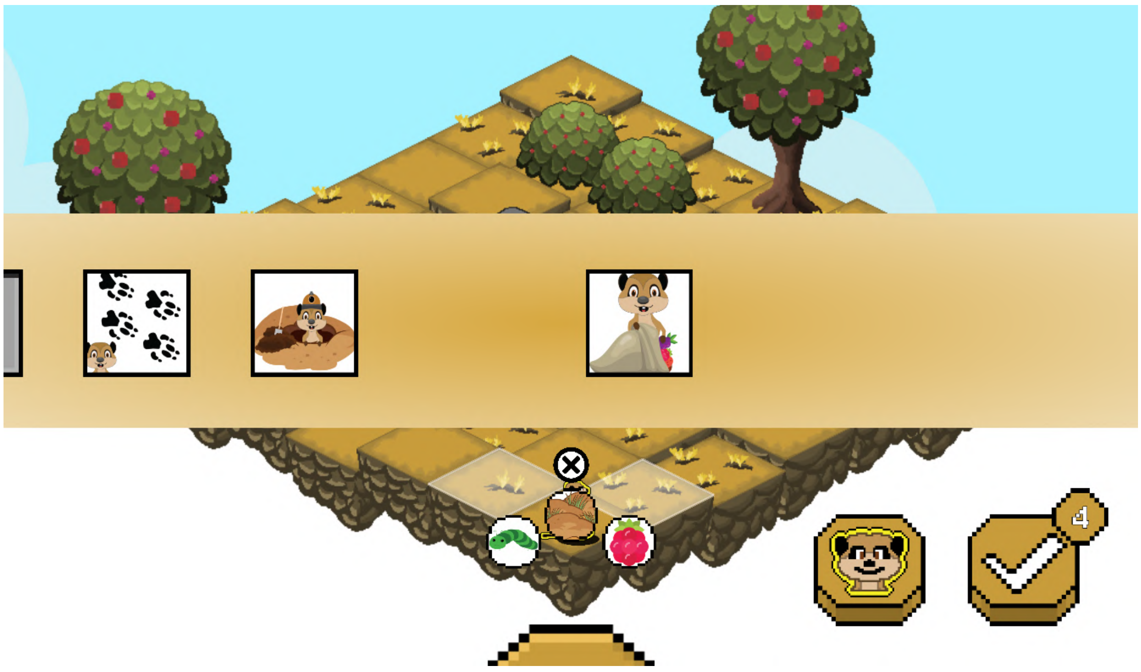


Figure 5.24: New actions are unlocked

When players open the Petri menu after finding new content they will find that they can use the newly acquired elements to create new Petri Nets. However, even though players can see their Petri Nets and unlocked elements, they cannot create new Petri Nets during a round. Petri Nets can only be changed at the start of every round before every player throws their dice. This is similar to the paper version since giving players the freedom to create Petri Nets whenever they wanted would diminish the stakes of the game. In this way, it also promotes a moment of debate and cooperation between them.

The last thing players can see in the arena are indications in the cells to which the characters can move. These highlighted cells only appear to the player when it is their turn and serve simultaneously as an indicator of where the players are able to move and when their turn starts. Highlighting these cells is a good example of the digital medium allowing the game to further aid players' actions, as previously mentioned by Iwanicki (Steam, 2020).

5.5.1 Defining and Processing Acceptable Player Actions

At the start of the game, players have the ability to move around and search cells, so they need to keep exploring, finding new content, unlocking new Petri Net elements, and spending their number of actions until all players have a total of 0 actions and a new round can start where they define new actions on the Petri Nets. The only thing left for players to learn is which new actions they can create and how to perform them. This is where one of the biggest changes had to be made in order for this system to work in the video game.

In the tabletop version, players were allowed to create any combination of conditions, actions, and outcomes as long as it made sense. The only entities responsible for evaluating if these actions made sense were the players themselves and the game master or facilitator. A player could create a Petri Net that was not accepted by the rest because the logic behind was nonsensical, but another player could come up with the exact same Petri Net and give an explanation for the connections he made that was so clear, that it is was actually accepted and used. This type of system allowed for the creation of dozens of different Petri Nets, some of which were not even considered by Pinto when creating the game. For the digital version, however, this could not be done, since the game needs to know a priori which should be validated for the player to perform actions that were known to make sense. To solve this a significant number of possible and sensical combinations of conditions, actions, and outcomes were coded into the game. This way the game not only knows what the players are trying to do, allowing them to do so, but it is also able to check if the Petri Nets are correctly built and calculate the outcome. In total, there are thirty-seven implemented combinations the players can create to allow the meerkats to perform different actions. The elements used for this are the exact same used in the paper version.

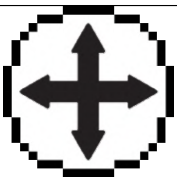
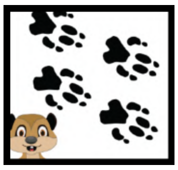
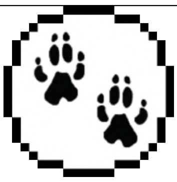
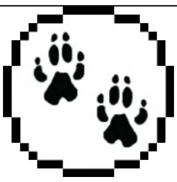
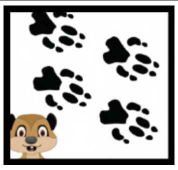
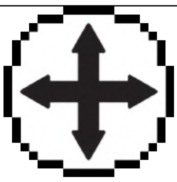
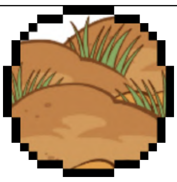
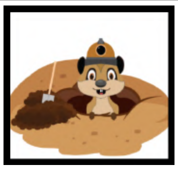
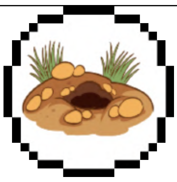
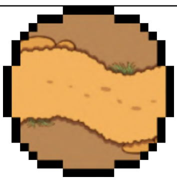
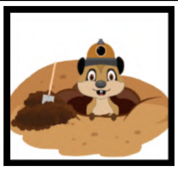
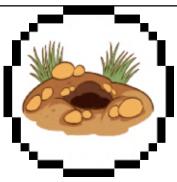
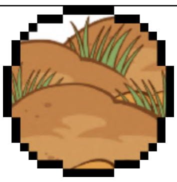
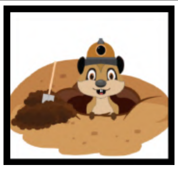
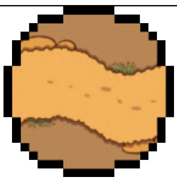
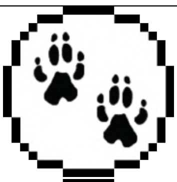
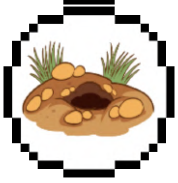
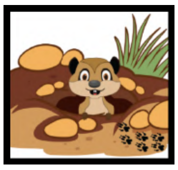
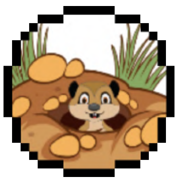
Action Number	Conditions	Actions	Outcomes	Legend
0				Move Meerkat
1				Move Meerkat
2				Create the burrow in rich ground
3				Create the burrow in a tunnel
4				Create a tunnel in rich ground
5	 			Enter the burrow

Table 5.4: Petri Net builds list I

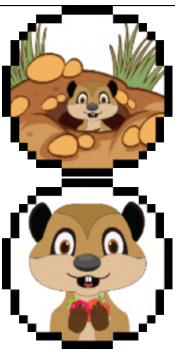
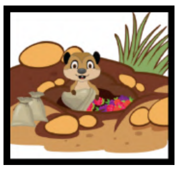
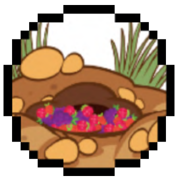
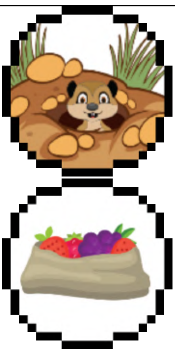
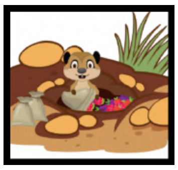
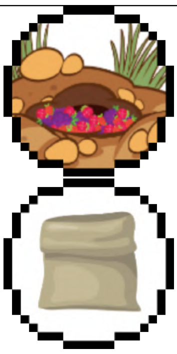
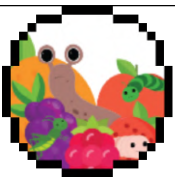
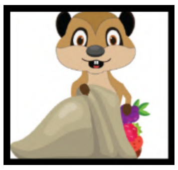
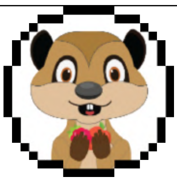
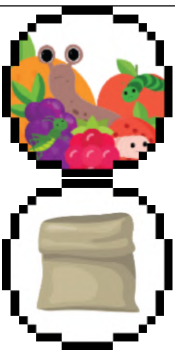
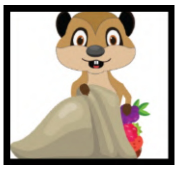
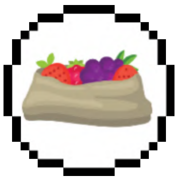
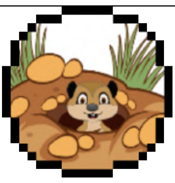
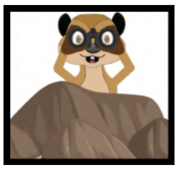
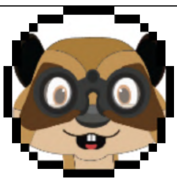
Action Number	Conditions	Actions	Outcomes	Legend
6				Deposit food from the hands in the burrow
7				Deposit food from the bags in the burrow
8				Collect food with hands
9				Collect food with the bags
10				Meerkats on the lookout

Table 5.5: Petri Net builds list II

Action Number	**Conditions**	**Actions**	**Outcomes**	**Legend**
11				Warn about the presence of a snake
12				Warn about the presence of a wolf
13				Warn about the presence of an eagle
14				Warn about the presence of a raccoon

Table 5.6: Petri Net builds list III

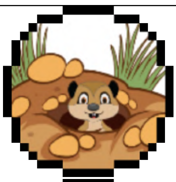
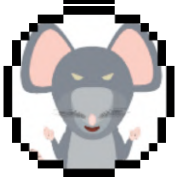
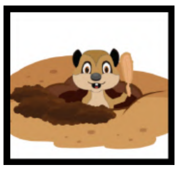
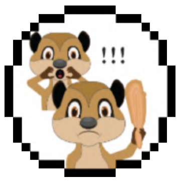
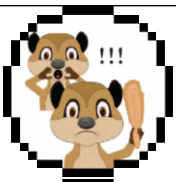
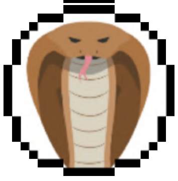
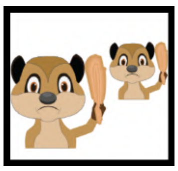
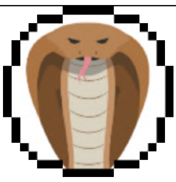
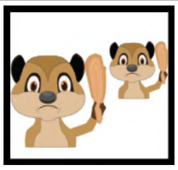
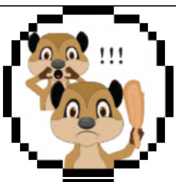
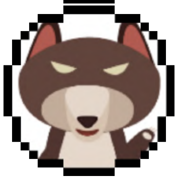
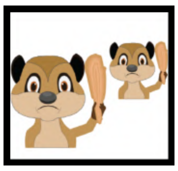
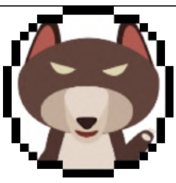
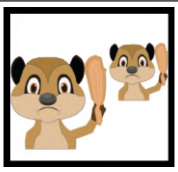
Action Number	Conditions	Actions	Outcomes	Legend
15	 			Warn about the presence of a rat
16	 			Fight snake with a warning
17				Fight snake without a warning
18	 			Fight wolf with a warning
19				Fight wolf without a warning

Table 5.7: Petri Net builds list IV

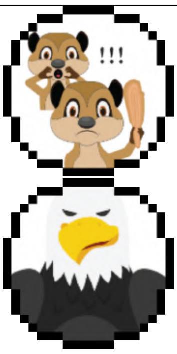
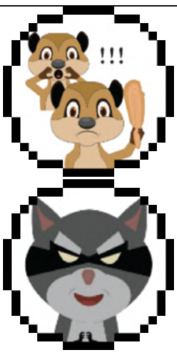
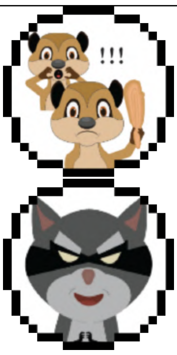
Action Number	Conditions	Actions	Outcomes	Legend
20			?	Fight eagle with a warning
21			?	Fight eagle without a warning
22			?	Fight raccoon with a warning
23			?	Fight raccoon without a warning
24			?	Fight raccoon with a warning

Table 5.8: Petri Net builds list V

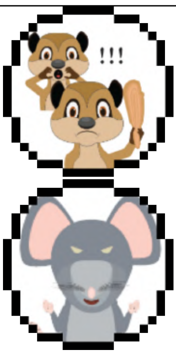
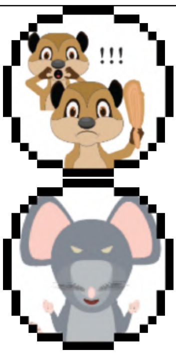
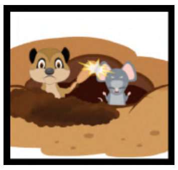
Action Number	Conditions	Actions	Outcomes	Legend
25			?	Fight raccoon without a warning
26			?	Fight rat with a warning
27			?	Fight rat without a warning
28			?	Fight rat with a warning
29			?	Fight rat without a warning

Table 5.9: Petri Net builds list VI

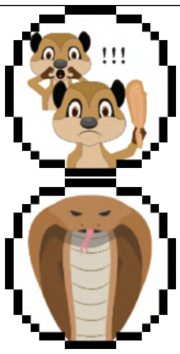
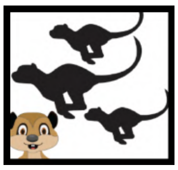
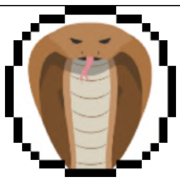
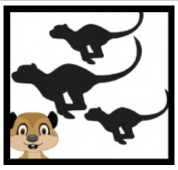
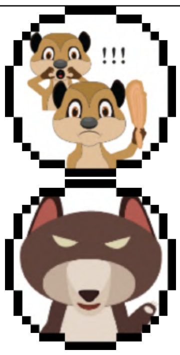
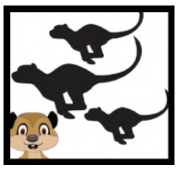
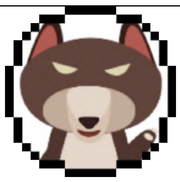
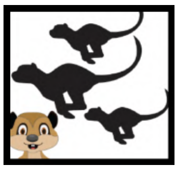
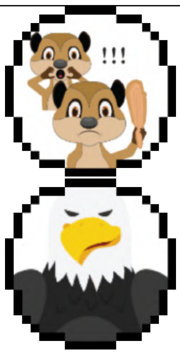
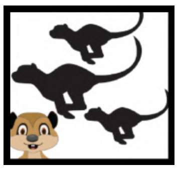
Action Number	Conditions	Actions	Outcomes	Legend
30				Flee from the snake with a warning
31				Flee from snake without a warning
32				Flee from the wolf with a warning
33				Flee from wolf without a warning
34				Flee from eagle with a warning

Table 5.10: Petri Net builds list VII

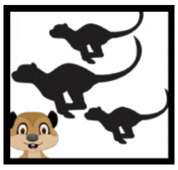
Action Number	Conditions	Actions	Outcomes	Legend
35				Flee from eagle without a warning
36				Give Bag

Table 5.11: Petri Net builds list VIII

Every time any of these actions is performed by one of the players, the number of actions they have (number given by the dice and other players) is decreased by one.

5.5.2 Inventory, Bags, Tunnels, and Burrow

If a player has enough space in their inventory, once they pick up a piece of food that food will disappear from the cell and appear in their food menu. Likewise, if a player picks up a bag (which is the only action in the entire game that does not need to be defined in a Petri Net) and granted that they do not already have two bags, the bag will appear in their food menu and vanishes from the cell. Once a player has bags in their inventory the food he collects can now be stored in them, with each bag taking one of the hand slots in the food menu but being able to carry four different pieces of food. Additionally, players can give bags to one another, as long as the bags are empty.

The following two images depict a food inventory menu with an empty bag and another one with a bag carrying food.

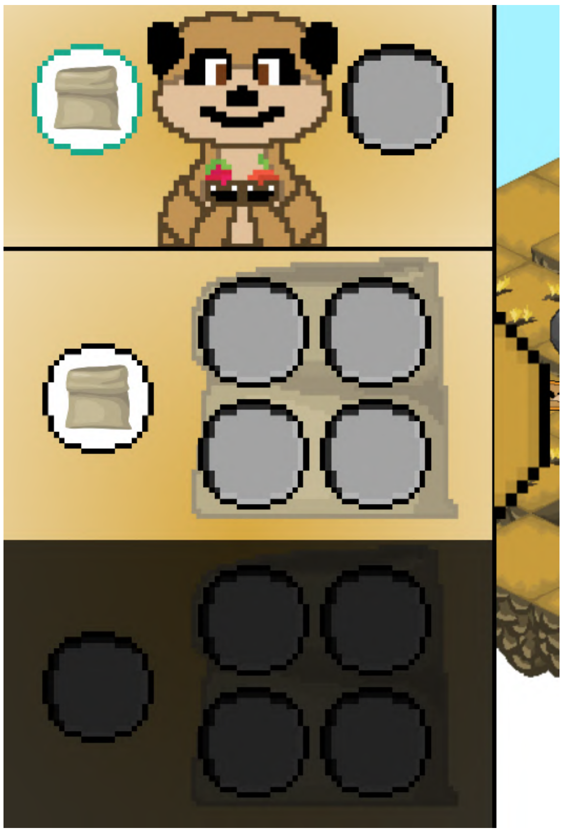


Figure 5.25: Food menu with an empty bag

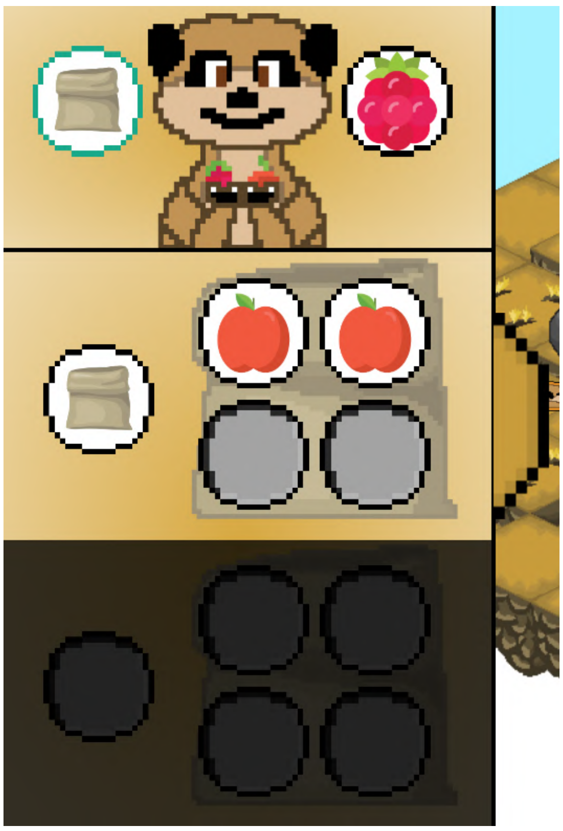


Figure 5.26: Food menu with a bag carrying food

The following image shows the player's bag icon turning more visible to indicate that the player is carrying a bag.

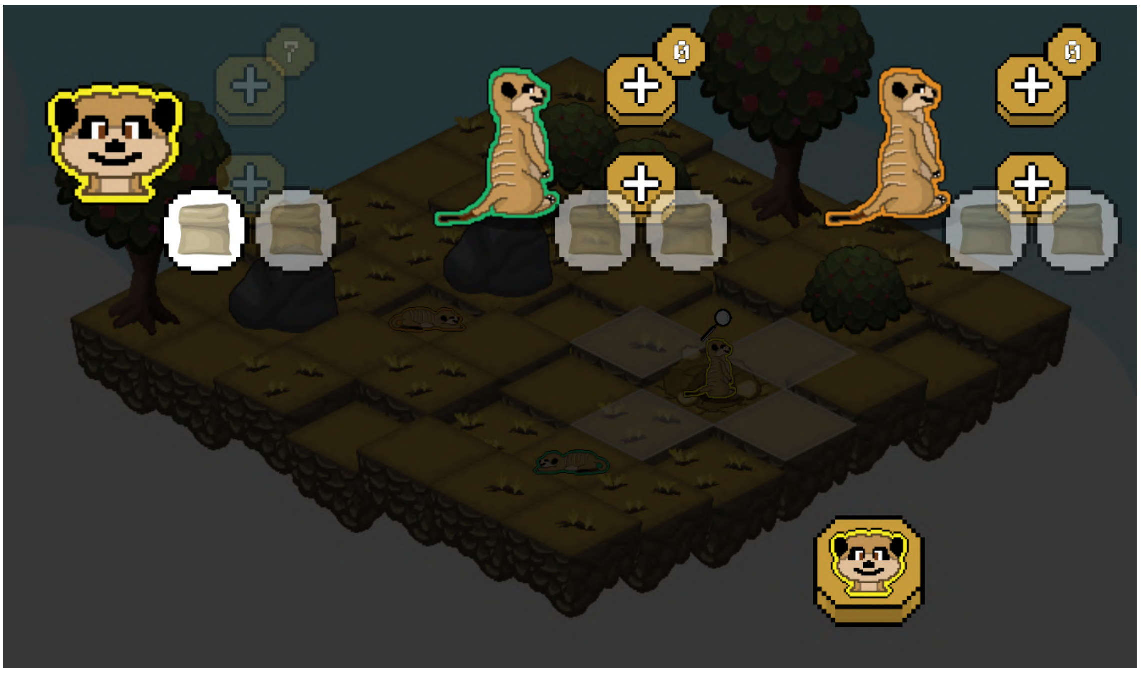


Figure 5.27: Player menu showing that player one has a bag

Players can also interact with the rich ground that can sometimes be found when searching a cell. Rich ground allows players to perform two different actions: dig a tunnel or create a burrow.

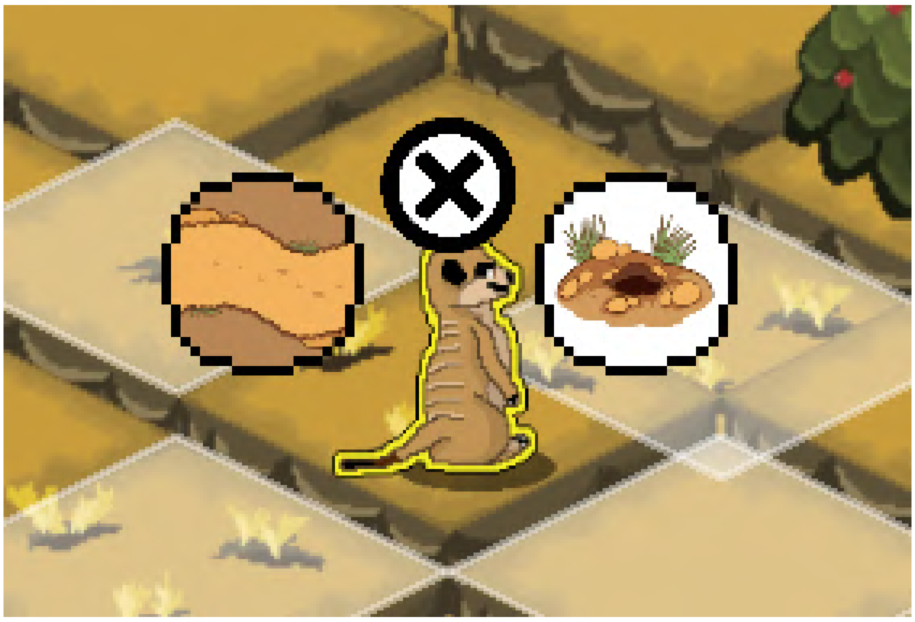


Figure 5.28: Options of interaction with rich ground

When tunnels are connected, they allow players to travel between them at the cost of a single action. If the players have five different tunnels connected, they can move to one of them and move all the way to any one of the other four and it will cost them only one action.

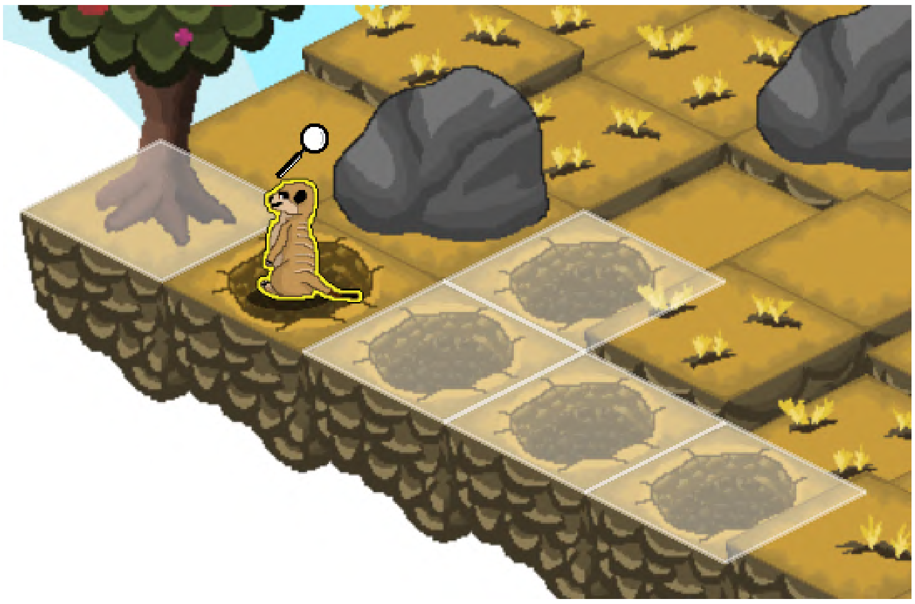


Figure 5.29: Five connected tunnels

The burrow has several functions and once created, is the most important cell in the entire game. Burrows are where players can move to and enter in order to deposit the food they are carrying in their inventory.

The following two images depict a meerkat on a cell with a burrow and a meerkat inside the burrow, respectively.

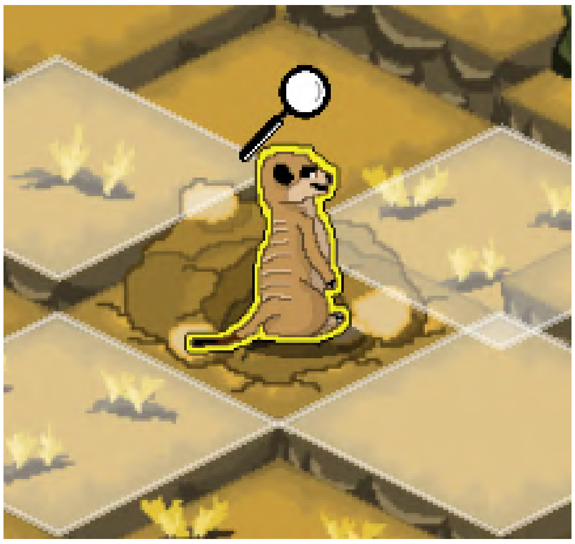


Figure 5.30: Player on the cell with a burrow

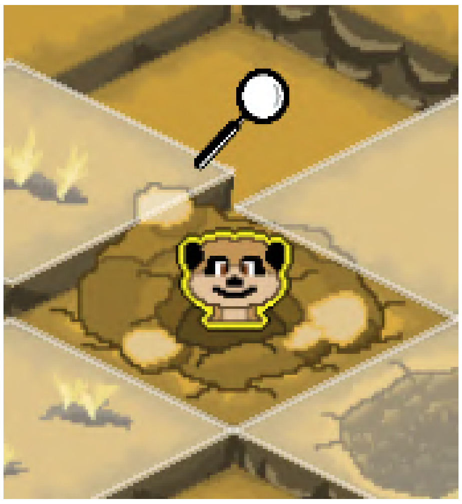


Figure 5.31: Player inside the burrow

By storing food in the borrow the bottom section of the colony menu will receive that food and the counter will go up, representing how much food is stored. At the same time, for every two food elements stored in the burrow, 1 baby meerkat appears in the top section of the menu, and its counter will also go up. After two elements of food are stored after a baby meerkat is born, the meerkat will grow into an adult.

By having meerkats in the burrow players will be able to use them in order to create an action that puts these meerkats on the lookout for predators and thieves, which helps players fight off the animals more easily.

The following image depicts a colony menu with food, a baby meerkat, and an adult meerkat.

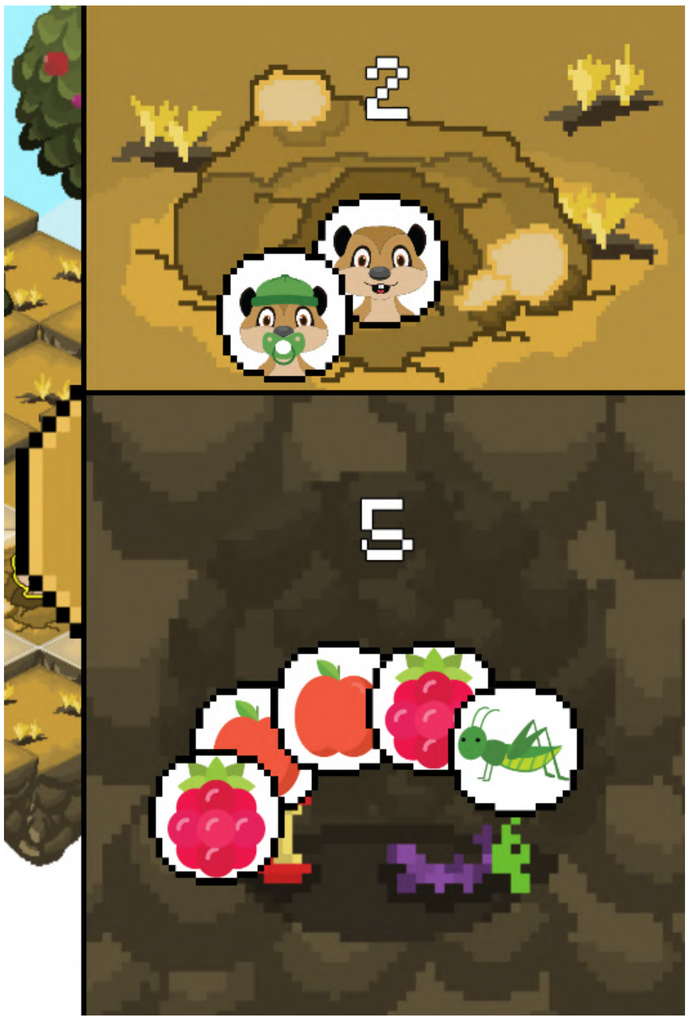


Figure 5.32: Colony menu with food and meerkats

Another function of the burrow has to do with predator confrontation. When players find a snake, wolf, or eagle and have no actions that allow them to confront the predators or if they have the action to flee from these animals, they will be sent all the way back to the starting cell. However, if the players have created a burrow, they will be sent there instead. This and the fact that players will be trying to store food in the burrow for the entirety of the game makes choosing the place to create the burrow a strategic decision.

Additionally, the burrow can also be created from a tunnel instead of rich ground, but regardless of where it is created, there can only be one single burrow in the game.

5.5.3 Predator Interactions

When players find a predator, one of three things can happen. The first scenario happens when players have no way to confront the predator and immediately lose and are sent back to either the starting cell or the burrow. If the players lose against a predator, one meerkat is removed from the colony menu.

In the second scenario, players have either the ability to fight the predator or flee from it. When this is the case, a dice will appear on each player's screen and after all of them have clicked their respective dice, a bar will start to fill up with the total amount that the players achieved. If the bar is not totally filled up, the players lose, but if they are able to fill it up they win and perform the action they have defined. The total amount needed to fill the bar is less when trying to flee than when trying to fight the predator, however, fleeing sends the meerkat back to the starting cell or burrow and the predator remains, whilst successfully fighting the predator will remove the predator from the cell.

The third scenario happens when players have both fighting and fleeing from the predator defined. When this happens, both options will appear on the screen of the player who found the predator, and it is up to them to decide which one to perform, as seen below.

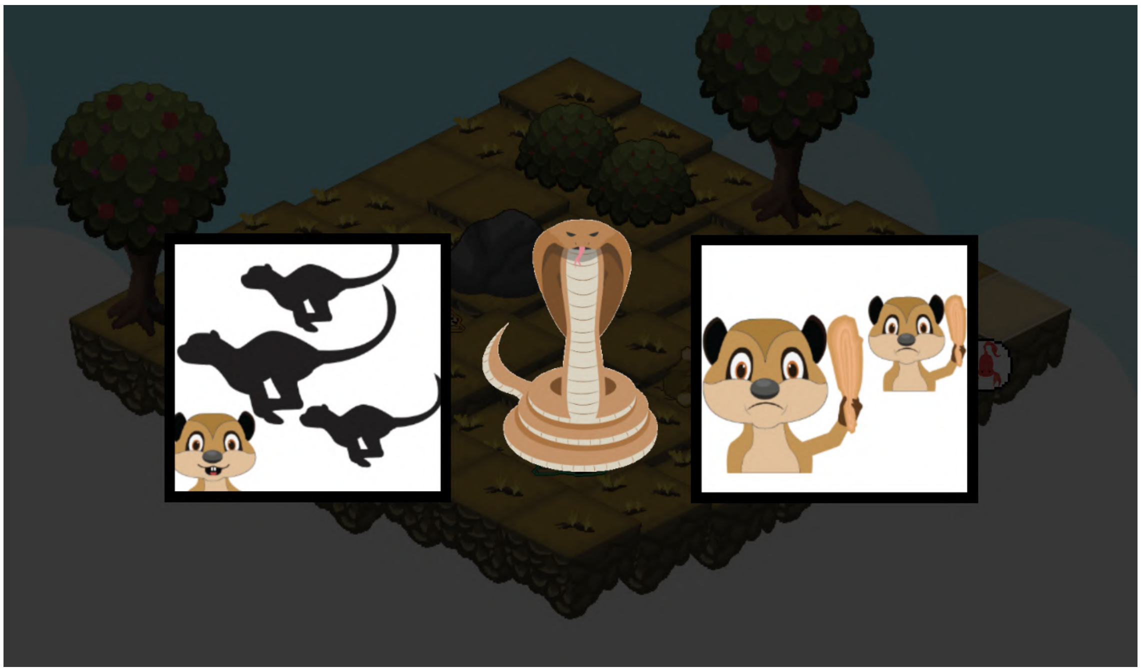


Figure 5.33: Options to confront predator. Flee on the left and fight on the right

Once players decide which option to take, or if they only have one action, the following screen will be shown.

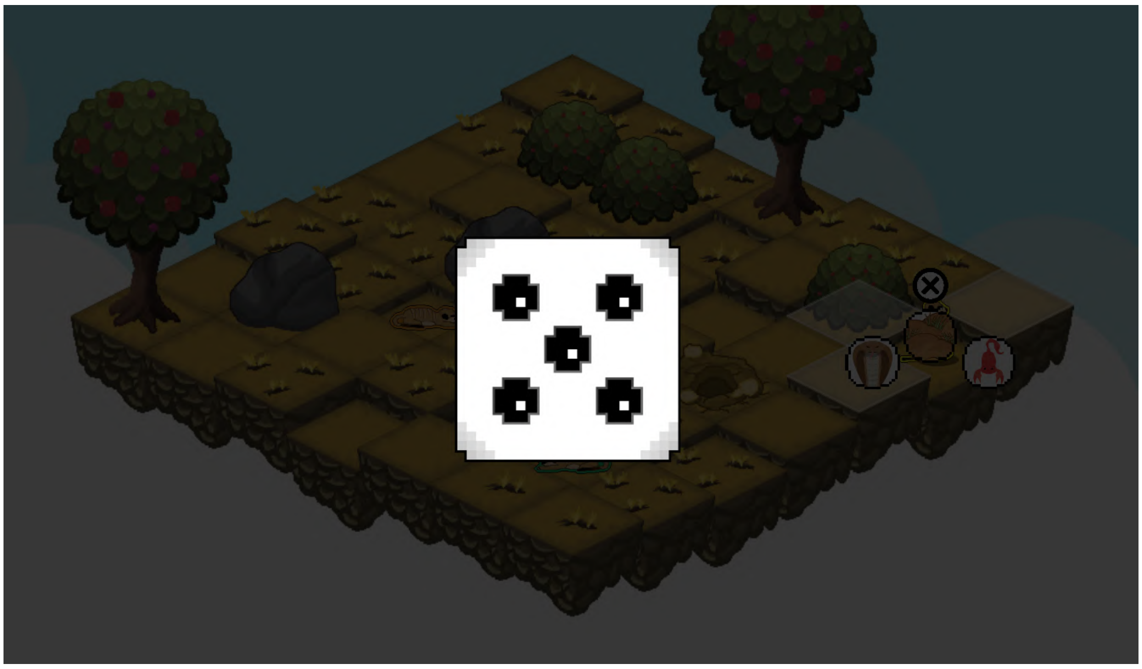


Figure 5.34: Dice used in predator and thief mini-game

Afterward, a bar will start to fill with the total accumulated points.

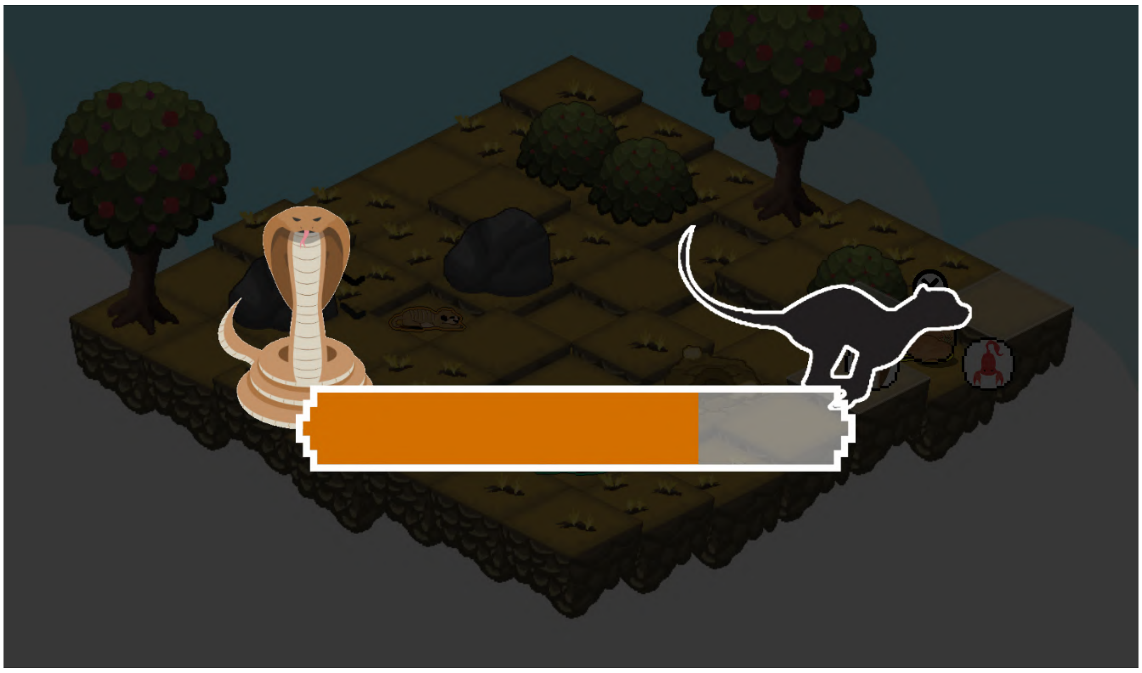


Figure 5.35: Predator bar filling up

When players find a rat and have no way to fight it off, the screen of the player that found it will show a dice that when pressed will reveal a number between one and six. This number represents how many tunnels the rat will destroy. However, players can fight the rat just like they do with the predators. In this scenario, all players throw a dice and the bar fills up. If the bar is not totally filled up, the player who found the rat will throw a second dice to see how many tunnels are destroyed. If it is filled up, however, the rat will be removed.

The raccoon encounter works exactly the same as the rat. The major difference, other than the elements needed to create the Petri Nets to confront it, is that instead of destroying tunnels, the raccoons will steal food from the burrow.

It is not possible to flee from the rat or raccoon, players either lose or fight them off.

The reason the game uses a bar to show how close players are to achieving their action against the predators or the thieves is that in the original version when players found these animals they were told that all of them had to throw a die and the total amount had to be equal or greater than a set number. The bar replaces that verbal warning with a graphic one.

5.6 Development Process

Initially, the idea for the development process of Meerkplay was to follow Salen and Zimmerman (2003) iterative design approach and build two functional prototypes for testing, one in the middle of March and another in the middle of May.

A meeting was held where all the components needed for the game were discussed and how they could or should be implemented. Additionally, a weekly plan was formulated that described which components were going to be built each week until the very final testing week.

Very early on the actual development, however, it was made very clear that these predictions were highly underestimating the difficulty of the whole process. The whole month of February alone was spent learning the basics and intermediate functionalities of Godot, as well as creating the most basic components of the game, such as the board and the characters. The time required to finish each component of the game proved to be way longer than expected and so the initial calendarization ended up not being followed.

Eventually, the game development conducted resulted in five different prototype stages. The game would reach a new prototype stage once several components were added that deemed it necessary to test the game created thus far as a whole. However, brief tests were also performed on these components as soon as they were created, to verify if they were indeed functional.

The first prototype was tested once the arena, Petri menu slider, characters, character movement, turn change, and throw of dice were implemented.

The second prototype was tested once the conditions and outcomes slider, action slider menus, character redesign, player's action number representation, arena redesign, content pop-up, and food storage in the food menu were implemented.

The third prototype was tested once the predator encounter, thieves encounter, Petri Net logic, food storage in the colony menu, and the birth of meerkats were implemented.

The fourth prototype was tested once the integration with the Erlang Server was finalized.

Finally, the last prototype was tested once the PCG algorithm was implemented.

All of the necessary game components that had to be created for the game, regardless of complexity or importance, were kept in backlog lists in order to serve as a constant reminder of what still needed to be done and what had been done.

Similar lists were made for bugs. Every time a bug was encountered during testing it was added to a list and would only be removed once it was fixed. This proved especially useful when certain components of the game proved to be buggy when testing was being done on something else.

5.7 Summary

The development of the game proved to have some hindrances and the initial planned timeline had to be remade. Learning the basics and intermediate aspects of Godot took longer than expected and the process started getting backlogged because of this.

Once this hurdle was surpassed, however, development started taking shape. Every single major component of the original version was transmediated into the digital version and the PCG algorithm developed, although simple, is capable of performing the tasks that are required of it.

The biggest problems originated with the integration of the game with the Erlang Server, which not only forced a large amount of code to be changed, it also introduced a large amount of new bugs, with some being unfortunately still present.

Chapter 6

Game Evaluation

In this chapter, the goals of the evaluation process will be clarified as well as the protocol and materials used for the tests. The target audience of the project and the profiles of the participants in the gameplay tests for evaluation are also expressed further down. Finally, a resume of all game sessions is performed, followed by the results found in them, as well as an analysis of said results.

6.1 Evaluation Goals

The evaluation's main purpose was to test if and how well the main goals of this project were achieved. These goals, as previously mentioned, are the transmediation of an analog game to a digital medium and the inclusion of a PCG algorithm in the digital version that changes the game's content in order to grant a higher level of replayability.

With this being said, there were several aspects of each of these two goals that were evaluated during the gameplay testing sessions.

For the goal of media transmediation, the purpose of the evaluation was to answer several questions related to the changes that were made between the tabletop paper version and the digital version. Extensive gameplay tests were made by Pinto (2022) on the original prototype, which sought to not only evaluate if the game achieved the game's goals of "self-efficacy" and "collaborative work", but to gauge if players understood the components of the game and how to interact with them. The cooperative gameplay was found to be promising in the way it promoted self-efficacy toward developing computational thinking competencies.

One of the goals of the transmediation process was to bring as much of what made the original game itself to the digital version, adding new content that can only be achieved in this computational medium, but only enact changes when necessary, so as to not deviate too much from the gameplay qualities achieved with the source material. However, since several components were changed or added in order for the current version to work, these changes had to be evaluated on how clear they were to the players and the resulting gameplay.

The evaluation performed on the transmediation process during and after gameplay sought to answer the following:

1. Do players understand the purpose of the Petri Nets?
2. Can players create Petri Nets by themselves?
3. Do players understand the purpose of the number given by the dice?
4. Can players understand how to search for content?
5. Can players understand how to interact with a cell's content?
6. Can players understand how to give actions and bags to others?
7. Can players understand how to finish their turn?
8. Do players understand that predators are hiding behind every bush and tree?
9. Do players understand how to win against a predator or thief?
10. Do players understand what happens when they collect food?
11. Do players understand what happens when they deposit food in the burrow?
12. Are players able to play the game by themselves without help?

For the PCG algorithm, since it was built in order to change the game arena and its content, both in quantity and type, for every different game, the evaluation process served to give an answer to the following questions:

1. Do players notice that the arena configuration changes between games?
2. Do players notice that the content in the cells changes between games?
3. Do the changes from one game to another affect how players engage the arena?

6.2 Evaluation Protocol

For the evaluation process, three different sessions were performed in person with three different groups.

In each session, two or three players played the game simultaneously on different machines for a total of one hour and thirty minutes. Each group played two instances of the game, in order to play in different arenas created by the PCG algorithm.

Group	First Session	Second Session
Group 1	Started at 11:00; ended at 11:39	Started at 11:50; ended at 12:35
Group 2	Started at 14:00; ended at 14:42	Started at 14:50; ended at 15:35
Group 3	Started at 15:50; ended at 16:35	Started at 16:40; ended at 17:15

Table 6.1: Session times

Whilst the game sessions were occurring, an audio recording of the player's conversations was made for anonymous transcription and future study, and that will be erased to respect the privacy of the participants. Anonymous logs of the gameplay with all of the broadcasts performed in each game were also created for the same purpose.

During gameplay, notes were taken on how players were reacting to each new element of the game. The objective here was to record how players understood the purpose of those new elements, how well they understood how to interact with them, and what their feelings towards those elements seemed to be.

Once players played the game for about forty-five minutes, a new game would be started to check if players showed more autonomy the second time, by being more familiar with its rules and if they noticed the changes to the arena and its content.

Finally, once both gameplay sessions were done and the players were off the computers, they were verbally asked to give one short answer about what part or component of the game they liked the most as well as what part or component they disliked most about the game, as well as if they would play the game again.

6.3 Target Audience

Pinto (2022) claims that their paper prototype had as its target audience "children from ages 7 to 10". Since this project is essentially a port of Pinto's prototype that tries to be as faithful to the source material as possible, the target audience of the digital version of Meerplay is also directed to, and tested with children around that age group.

6.4 Participants

The following is a table that shows how many participants were present in each group, as well as their gender, age, and education level.

Group	Player	Gender	Age	Education Level
Group 1	P1	Male	9	3rd Grade
	P2	Female	8	3rd Grade
	P3	Male	8	3rd Grade
Group 2	P1	Female	6	1st Grade
	P2	Male	7	1st Grade
Group 3	P1	Female	12	6th Grade
	P2	Female	7	1st Grade
	P3	Female	11	5th Grade

Table 6.2: Participants' profile

As can be observed, most participants fall in the target audience's age group, whilst the ones that do not, deviate very slightly.

6.5 Materials Used

Three computers were used simultaneously to perform the gameplay tests. One of these computers was a notepad, which also served to test how well the game works on touch screens.

In order to have all machines running the same game instance, a shared internet connection was set up to access the multiplayer server.

In addition to the computers, a cell phone was utilized to record the players' conversations.

6.6 Results and Analysis

This section's objective is to describe how the participants behaved during the gameplay sessions, as well as report and analyze the results which allowed answering the questions previously formulated.

6.6.1 Sessions' Resumes

Performing three different sessions with three groups of different ages and personalities resulted in very distinct gameplay sessions. The following are brief resumes of what happened during each group's sessions to both show how the players interacted with the game and give some context to the results discussed further down.

Group	Player	Gender	Age	Education Level
Group 1	P1	Male	9	3rd Grade
	P2	Female	8	3rd Grade
	P3	Male	8	3rd Grade

Table 6.3: Group 1 participants' profile

This group was especially interesting because all three participants had also been part of the gameplay tests for the original game, which meant all three were at least somewhat familiar with the game.

P2 was the calmest and most focused player of the three. She tried to be strategic and asked other players for help doing so. She tried to guide them into accomplishing objectives and overall, tried to play the game the way it should be played.

P3 was the most introverted of the group. He talked very few times throughout the session even when directly asked questions. He was also slower than the other two when building Petri Nets, which caused him to be confused and just watch as the other two built them. It is possible, however, that if the players were located in different rooms without direct access to the others' screens, this player would have tried harder to create Petri Nets on his own.

P1 however, was extremely extroverted compared to the rest and very early on in the game, took charge of the whole session. He would tell other players to hurry up during their turns, would tell, and almost force them, to perform the actions he wanted, and would ignore anything the other players told him when it was his turn to play, especially to P2, who was actually trying to reach the games' goals. Most egregious of all, however, was a moment where he picked up P2's mouse and clicked to skip turn because he believed she was "taking too long". At this point, the other players, who were already frustrated by the way he was behaving, started to get irritated by him, to the point that P2 started shouting at him. P1 had to be called out to calm down and let the others play.

Additionally, because this group was already familiar with the game, P1 knew that predators were hiding in the bushes and trees and made it their mission to defeat one of these predators. Because of this, while the other two were trying to explore the cells, collect food, build a burrow, and store food, P1 was constantly trying to reach a cell that contained a snake to defeat it.

Eventually, a bug occurred where one of the players had a negative amount of actions and the game wouldn't progress to the next turn, which forced a restart. Thankfully, this restart was very close to the forty-five-minute mark, which was the end of the first session.

In the second session, the players were able to explore the arena even further since they were now more accustomed to the game and knew how to build the Petri Nets to define actions. P1 was even finally able to defeat a predator by listening to the instruction that in order to do so, they had to build an action to confront it. When the predator was defeated, P1 jumped from his seat running around and screaming in happiness. Unfortunately, after this, he appeared to lose almost all interest in the game since all he wanted to do was defeat a predator.

Group	Player	Gender	Age	Education Level
Group 2	P1	Female	6	1st Grade
	P2	Male	7	1st Grade

Table 6.4: Group 2 participants' profile

Both P1 and P2 showed very similar personalities when playing, to the point that both seemed to almost think in unison. The only major difference between the two was that P2 was a little bit less shy to express his ideas when building Petri Nets.

Both sessions went very similar. They both played together to create burrows and tunnels, collect food, store food, defeat predators, etc. They showed to be so interested in the game that when they were watching the other play they would lean over to the other's computer and almost cover their screen with their head. In fact, in order to get them to stop playing after the end of the second session, they had to be told that they could only play one more round, and then it was over.

One thing worth mentioning is that the second game of this group had to end a bit earlier than expected since a different bug from the previous group occurred, where each player's game showed players to have different amounts of actions, which led to the impossibility of finishing a round since one client still counted an action left while the other was at zero.

Group	Player	Gender	Age	Education Level
Group 3	P1	Female	12	6th Grade
	P2	Female	7	1st Grade
	P3	Female	11	5th Grade

Table 6.5: Group 3 participants' profile

The third group proved to have a different dynamic than the previous two. Two of the participants were older than the rest, in fact, they were older than the ideal target audience for this game. The other, P2, was almost half their age.

P1 and P3 understood Petri Nets relatively quickly.

P2 however, may be one of the participants who least practiced how to create Petri Nets by the end of their session. This is due to the other two being able to very quickly come up with ideas and implement them, whilst P2 remained in a passive listening stance that may have inhibited trying to understand their logic. Even though P1 and P3 still tried to explain to P2 what they were doing, P2 seemed not to keep up with everything. However, despite not understanding the Petri Nets the other two participants were building, P2 still managed to understand how to play the game and what was expected of them to do.

For some unknown reason, when players were about to start their second session, one of the computers, which had been used on all previous game sessions, stopped connecting to the Erlang Server. The computer was able to run the game, but nothing would happen when either the create game or join game buttons were pressed, even after relaunching the game and resetting the computer. Because of this, P2 and P3 had to play the second session together on the same computer.

All of this led to two game sessions similar to the second group's, although slower since the older participants had to explain to P2 what they were creating between rounds. Just like the other groups, once the participants showed that they knew how to play the game by themselves, they were left on their own without external help, other than when necessary, and kept playing until the time to end the session eventually came.

6.6.2 Sessions' Analysis and Discussion

Although Meerplay is a game designed for kids, it can be somewhat confusing when played for the first time. The gimmick of creating Petri Nets to allow the characters to perform certain actions is not only unusual for a game, it can be borderline hard to understand without help, especially for children.

One thing that the digital version developed for this project has in common with Pinto's prototype and that is lacking in both, is the fact that there is no manual, guide, or indication of what the players should do with the Petri Nets in the game itself. Because of this, exterior help had to be given to guide the players in their first action definition, as done in Pinto (2022) gameplay evaluation tests.

With this startup guidance, the players' behaviors described above, and the answers given by them at the end of the sessions, it is possible to get an idea as to how well the game was received and interpreted and answer the main evaluation questions.

1. **Do players understand the purpose of the Petri Nets?**

   As previously stated, the Petri Nets were what gave the players the hardest time, which was expected. The concept of a Petri Net in itself, although fairly simple, was completely novel to most of them, and took getting used to it by building several actions to finally understand it.

   The way Petri Nets work had to be explained to every single group at the beginning of the first gameplay session and it was visibly clear by both their expressions, statements, and overall way of interacting with the game that they did not fully understand what to did at first.

   Group 1, who had already come in contact with Petri Nets during Pinto's gameplay tests, had to be reminded of how they work, as all three players could not quite remember it. Eventually, after receiving some help to build three different Petri Nets, both P1 and P2 started to verbally express how to build some of the next ones, but P3 however, remained silent throughout most of the sessions, preferring to observe what the other two were building.

   For the second group, Pinto helped in explaining to them the concept of the game and Petri Nets. This explanation was so simple and clear, that even though both players were relatively young, they both started to show they understood how Petri Nets worked as soon as the one to move the meerkats was built.

   The third group also grasped the concepts of the Petri Nets relatively quickly.

   With what was observed and recorded during these sessions, it is possible to state with a high degree of confidence that most players understood Petri

Nets by the end of the gameplay. Only the first group's P3 and the third group's P2 seemed to not fully understand it since they practiced less or preferred to watch the other two players of their group build the Petri Nets, and very rarely tried to build on their own.

Regardless, the initial reason players were able to understand the Petri Nets was because it was explained to them. Without this explanation, players would not know what to do or have to persist in trial and error, which indicates a necessity to revise the design to include a tutorial demonstration at the beginning.

2. **Can players create Petri Nets by themselves?**

   The only thing harder for players to grasp than how Petri Nets worked, was how to actually create one and for what purposes.

   Throughout all three groups' first sessions, there wasn't a single Petri Net built by the players without some exterior help. Players simply did not know what to create, because they did not know their goals unless they were told. Additionally, in order to build the Petri Nets even after being given a goal players found trouble in coming up with sets of conditions, actions, and outcomes that worked in the game.

   Even when they knew that they could only use the elements that were presented to them in the game, players tried to suggest ideas with other elements. Group 2 P1 for example, when asked "What does a meerkat need to build a hole?" would answer "A shovel!" or "Hands!", which, while somewhat correct answers, especially because the action used for digging a hole shows a meerkat with a shovel, were not viable elements present in the game, and so her attention had to be redirected to the existent elements until she finally said "Earth!" when referring to rich ground.

   For the second set of gameplay sessions, however, players showed to remember some of the Petri Nets previously built and were able to create them on their own.

   Unfortunately, only one type of Petri Net was built by the players without any sort of exterior help, which was the Petri Net to fight certain predators. This only happened once players were helped in creating a Petri Net to confront one predator. They would then realize, after finding a new one, that they could create a similar Petri Net to defeat this new enemy. This occurred in all three group sessions and for every other Petri Net the players decided to build, they needed some sort of push to finally get them right.

   If the game was finished sooner and longer gameplay sessions were performed, the players would have had more time to go through the process of trial and error until they finally created working Petri Nets, but due to the eventual time constraints, there was a need to push the players in order for them to experience more components of the game. This push, however, never meant explicitly telling players what to do, they were merely subtle hints. Furthermore, sometimes all the help players required was a hint for a single element of the Petri Net they were trying to create.

In conclusion, the gameplay sessions showed that players were not able to build entirely new Petri Nets by themselves other than very similar ones. They only built Petri Nets that they were shown to work in previous games. However, the reason for them not being able to create Petri Nets by themselves did not stem from the digital version's design, as all eight participants showed to be able to open the Petri menu, scroll through this menu, select the Petri Net elements in order to pop-up the elements' menus, scroll through these menus, and place elements in all positions of the Petri Net. Their difficulty stemmed from the creation of Petri Nets in of and of itself and was also a result of the aforementioned lack of time to go through the process of trial and error.

3. **Do players understand the purpose of the number given by the dice?**

   All players immediately understood that they had to throw a die at the start of each round to start it. They also understood that the number each one got represented the number of actions they could perform in a round. This was even more evident when the groups were let to play by themselves and still managed to play dozens of rounds.

4. **Can players understand how to search for content?**

   Just like the dice, the players understood very easily how to search for content in the cells. The magnifying glass icon proved to be very clear on its purpose and players very quickly figured out what it was used for.

5. **Can players understand how to interact with a cell's content?**

   At the beginning of the first session of each group, all of the players tried to click the content in the cell once it popped up. However, they had to be reminded that in order to interact with the elements they had to create Petri Nets that allowed them to do so.

   Once this was clarified once more, players realized that when they selected the content but had no action defined that allowed them to interact with it, the content remained in the cell, but if they defined an action that allowed them to do so, the content would react.

   All of them also quickly noticed that when either a predator or a thief was found, an instant mini-game was started, even without them directly interacting with the content.

   All groups were able to collect food, deposit food in the burrow and confront predators without any extra help other than the one given at the start, which shows that they were indeed aware of how to interact with the cells' content.

6. **Can players understand how to give actions and bags to others?**

   Players had no problem understanding how to give actions to one another. They showed to be able to always open this menu when needed and knew which meerkat was who's, including their own. Since giving actions is a simple press of a button and players can see their actions decreasing and the other player's actions increasing when doing so, the system to share actions was very clear to them.

Group 1 and group 2 especially used this a lot. Group 1's players always tried to share the number of actions between them in order for all of them to have around the same amount, regardless of where they were in the arena or how much food they were carrying.

Group 2 also shared actions almost every round, however, this sharing of actions was purely strategic and they would only give actions if doing so proved to benefit the whole group. There were even moments where one player gave all of their actions to the other because the other one was closer to an objective that both agreed on.

Group 3 also understood this process, however, they shared actions very few times between them. The sharing of actions in this group was mainly done when one player had very few actions given to them at the start of the round and asked the others for more.

Bag sharing was a little bit more complicated because of the conditions needed to do so. Players tried to share bags once they picked them up and could not, so they had to be reminded that they probably did not create a Petri Net that allowed them to do that action. Both group 2 and group 3 gave up on sharing bags once reminded of this as they only wanted to share bags out of curiosity to try them out. Group 1 however, was adamant on sharing a bag and tried to do it again a few rounds after having collected the first one in the first session. When P2 tried to give her bag to P1 the bag did not move from one player to another even though they had created the correct Petri Net. At this point, players were told that bags could only be given to other players when they were empty and P2 had already collected food inside her bag. Eventually, P2 deposited the food in her bag in the burrow and was finally able to give her bag to P1.

Because group 2 and group 3 gave up on sharing a bag, they were never made aware of the conditions that had to be met in order to do so. With this being said, it is reasonable to assume that only group 1 finished their session knowing how to share their bags, but that both group 2 and group 3 would also understand how to do so if it were explained to them in the same manner.

7. **Can players understand how to finish their turn?**

   Almost every player understood how to finish their turn in the very first round of the very first session. Only group 1's P3 and group 3's P2 showed to be confused about what to do once they had no actions and the other players had to aid them. This only occurred in the first few rounds, however, after which both players fully grasped how to finish their turn.

   Other than two players having shown trouble with this component, the only thing worth noting is that players sometimes forgot they had to actively finish their turn, as sometimes they would stare at the screen or look at the other players, waiting for them to play, at which point the other players had to remind them to press the finish turn button.

   Since all three groups were able to play by themselves for several minutes, it is certain that everyone understood this element, since it is necessary to progress in every single round.

8. **Do players understand that predators are hiding behind every bush and tree?**

   Group 1 was aware of this from the start since they remembered it from the time they play-tested Pinto's prototype. P1 in particular had the single goal of reaching the closest bushes in both game sessions in order to defeat the predator hidden in them.

   Group 2 and group 3 started associating bushes and trees with predators in the same way. Once one of the group's participants found a predator in one of these tiles, they realized that these specific tiles stored large amounts of food but hid a predator in them, almost as if it were guarding them. This realization was evident since after the first predator encounter players in both groups started avoiding both bushes and trees, even telling each other to not search for content in those cells. Group 2's P2 in particular, was so keen on avoiding these that he would not even move to those cells and would ask P1 to do the same.

9. **Do players understand how to win against a predator or thief?**

   After every group had their first encounter with a predator or a thief, they were aided in creating Petri Nets that allowed them to confront these animals. Once these Petri Nets were built and they reached the same predator or thief again, they realized that a new mini-game would occur where all of them had to throw a new dice and a bar would start to fill up with the total amount the group achieved. The bar and its consequences were also very clear to the players, as even before it stopped filling up, players knew that if it were not totally full, they would lose.

   Every time a bar started filling up, players in group 1 and group 2 would get ecstatic, yelling to their screens hoping that the bar would reach its end.

   Additionally, in all three groups, the players inferred that the way to confront different predators and thieves was similar and that all they needed to change was the predator or thief element in the Petri Nets. This conclusion came up naturally to both group 1 and group 3, whilst group 2 had to get a suggestion in order to arrive at the same conclusion.

10. **Do players understand what happens when they collect food?**

    Players were told what each menu represented even before they started moving their characters in the first round, however, the amount of information they received regarding all of the game's components proved to confuse them at the beginning. Collecting food was the very first action players in all three groups defined and performed other than moving and yet, in every single group, once the first player collected food, everyone forgot for a brief moment where the food should end up.

    However, after a few seconds, the players remembered to check the side menus for the one that held the food. Once the players who picked up the food in each group opened their menu and everyone saw that the food they had collected was on that menu, everyone understood where the collected food would go. It was also at this point that players realized that the food menus were not shared between them, as only the players who had collected food had food in their inventory.

11. **Do players understand what happens when they deposit food in the burrow?**

    This situation was similar to the previous one. Once players finally had food in their inventory, a burrow, moved inside the burrow, and deposited food they forgot for a few seconds what would happen to it.

    Eventually, when players remembered to check the colony menu, they figured that the food in their food menu had moved to this other menu and once enough food was stored they also realized that baby meerkats appeared in it. It was at this point that the groups were told that the more food and meerkats they had in their burrow, the better they were performing.

    The baby meerkats were especially interesting to the players as both players in group 2 and the younger player in group 3 rejoiced when they saw the babies in the colony menu.

12. **Are players able to play the game by themselves without exterior help?**

    As previously stated, all three groups were in fact able to play the game for upwards of thirty minutes by themselves in their second session.

    Without the help given to them to explain what Petri Nets were as well as why and how to use them in the time they had to play the first session, players would not have been able to play the game autonomously. Even in the second session, where they already understood how to play the game, the only Petri Nets they built were the ones they remembered from the previous session and still needed some help in doing so at the beginning.

    Once again, it is worth noting that if the players were given more time to experiment with the Petri Nets, they would almost certainly have reached some if not all of their goals through trial and error, since they understood how to build them.

In relation to the PCG algorithm, these were the following results:

1. **Do players notice that the arena configuration changes between games?**

   Players did not actively state or show to have noticed any changes in the arena between games. Not a single player commented on the fact that the ground was different, that the rocks were in different places, or that bushes and trees showed up in different places and numbers.

   Players not noticing the changes stems from the fact that they only had time to play two sessions and none allowed them to explore every single cell and collect every single food. Were the players given more time to experience an arena in order to get more familiar with it and were they able to play more than two games, they would have most likely noticed the differences as soon as new games started.

However, even if players did not consciously realize that the arenas were different, the way they traversed the arenas clearly changed in order to navigate the new elements, which indicates that they were naturally influenced by the changes. This was more apparent in the first group's sessions, as P1 was fiercely trying to reach the closest bush or tree to the starting area in order to defeat its predator in both sessions, which had these cells in different positions. Even if none of the group's players realized it, P1 was trying to conquer the same objective with the same conditions in the same game, but because the cells had changed, he was forced to move to a different place in each session.

Although players did not notice the changes in the arena configuration, the changes provided by the PCG algorithm made them naturally behave differently between sessions, which proves that the algorithm did in fact offer come degree of replayability as intended.

2. **Do players notice that the content in the cells changes between games?**

   As was the case with the arena, players did not show any signs that they realized the content was different between games. However, much like with the game's arena, if players were given more time and more game sessions, they would probably start to actively figure out that the game changes every time a new arena is built.

   Additionally, although players did not state that the content was different in the second session, they also did not state the opposite. There was not a single occasion where a player said they knew what a cell was hiding or where they seemed to move to a specific cell (other than bushes and trees) because they thought cells in the same position as the last game would contain the same content. Likewise, there was not a single moment where a player stated, after searching a cell, that they expected there to be something else.

   Much like the arena, even if players did not visibly notice any change in the second session, they still played the game with an entirely new approach, by exploring the cells and their new content instead of trying to repeat what they had done in the previous game, which once more proves that the PCG algorithm has effectively reached its goal.

3. **Do the changes from one game to another affect how players engage the arena?**

   As stated in the previous couple of questions, players did indeed engage in the game differently between sessions.

   Even if the players did not actively notice the changes to both the arena and the content inside the arena's cells, the way the new arena was built forced players to navigate it in different ways in both games. Similarly, because the content had changed, players accomplished tasks at different rates.

   In Group 2's second session, for example, the players found so many cells containing rich ground next to the starting cell that they were able to create several tunnels connecting to each other fairly early on. The same did not happen in the first session, where instead of rich ground, players found larger amounts of food.

What influenced the players' way of playing the most from the first session to the second was the knowledge acquired about what the cells were hiding and how players interacted with them. Players knew going from one session to the other that rocks could not be traversed and that they blocked their path, that bushes and trees hid predators and lots of food, that the middle area of the arena housed rats and raccoons, that other than predators and thieves, cells only hid bags, food, or rich ground, and all groups also noticed that predators were hiding in regular cells ate the edge of the arena.

Players did not engage the arena in their second session as a repeat of the first. They analyzed the arena and interacted with what they found one cell at a time, as was the goal with the incorporation of a PCG algorithm

Finally, when players were verbally asked at the end of both game sessions to give short succinct answers about the component or part of the game they liked the most they gave the following answers:

1. **Group 1**:
   (a) **P1** - Defeating a predator.
   (b) **P2** - Collecting food.
   (c) **P3** - Collecting food.
2. **Group 2**:
   (a) **P1** - The baby meerkats.
   (b) **P2** - Defeating a predator and the baby meerkats.
3. **Group 3**:
   (a) **P1** - Exploring the arena.
   (b) **P2** - Exploring the arena.
   (c) **P3** - Exploring the arena.

All groups had different answers to what their favorite part was, from something as simple as watching baby meerkats appear in the colony menu, to exploring the game arena.

The groups' answers were very similar inside each one because the players probably influenced the others' answers. In group 3 for example, when asked what their favorite part of the game was, P2, the youngest, said that her favorite part was exploring the arena and finding new things, the other two players then immediately said that their favorite part of the game was also exploring.

Nevertheless, there were four different answers to this question, which shows that the game was able to capture the players' attention in different ways.

When it came to stating what part of the game the players disliked the most, however, the answers were not so varied, as there was a common answer that every single one of the eight players gave.

Every player claimed that the Petri Nets were very hard to understand and that it was confusing for them. As previously stated, even after playing two games and getting a verbal explanation, with the aid of images, of why and how to build Petri Nets, players still had trouble coming up with sensical solutions. Regardless of this, it was possible to see that all players other than groups 1's P3 and group 3's P2 were progressively understanding better and better how Petri Nets worked and how to interact with them throughout the game sessions. Even if they were not able to build Petri Nets that worked in the game, they still managed to try and build them based on the idea of a condition leading to an action which leads to an outcome. This means that although players did not create viable Petri Nets, the game did effectively promote their self-efficacy towards computational thinking, and once again, were they given more time, they would have probably created viable Petri Nets through trial and error.

Another answer players gave in more than one occasion was that they did not like to lose to predators and thieves, as that would remove some of their hard-earned content. This discontent, however, was not directed towards the game design, it was merely the players expressing their frustration when losing.

6.7 Functional Flaw Corrections

The spontaneous error that occurred at the start of the third's group second session was not the first time it had happened.

When testing the game prior to the gameplay tests with participants in the target audience's age range, it was found that when launching the game on Android, the game would open, and the buttons to create and join games would react when pressed, however, no connection was made with the Erlang Server. However, this was consistent in Android devices, so it was believed that there was either an error in Godot's importing system or with the server since this never occurred on a Windows import. This happening on a Windows import seemingly at random just adds to the confusion and mystery of what is the cause for this.

It is still unknown what causes mobile devices to not connect with the Erlang Server or why a computer that had run the game for over three hours on the evaluation day, and even more during prior testing, started to experience the same issue. It is highly unlikely, however, that the error resides in the game's code since a Windows version was launched and played hundreds of different times without this ever happening during development and testing.

One final note worth pointing out is that the computer that stopped connecting to the Erlang Server was relatively old and was starting to heat up to very high temperatures by the end of the sessions, which might have led to it not being entirely functional and possibly not having a good connection with the Erlang

Server.

This functional flaw should be corrected if the game is to be played on Android devices and to prevent it from stopping working on a Windows device again.

6.8 Design Implications

During the gameplay tests, several different bugs and errors occurred that need to be fixed if the game is to ever be officially used in any capacity.

As stated in the game evaluation section, the game suffers from a bug where players lose more actions than they should, which leaves them with a negative number of actions. Because the game only checks if players have zero or more actions to skip their turn or finish a round, if a player has less than zero actions the round will not finish and the game cannot progress.

A simple workaround would be to simply check if players have either zero or fewer actions, but this does not correct the issue, only circumvents it. The cause for this bug is still not yet known since players were able to play for upwards of thirty minutes without encountering it, and neither the logs from the game nor Godot's debug were able to give any insight into what caused it.

Further testing has to be made into every single action that reduces the players' total number of actions, to try and understand which ones are faulty. It is suspected that since players only faced this bug after playing for a while, these actions have to do with either confronting predators or thieves or with sharing bags under certain conditions that were not accounted for in the code, as both these types of actions were mostly performed by players in the late game.

Another bug players encountered that also forced a restart, was a bug found during Group 2's first session, where each participant's player menu showed them to have different actions in each one. P1 menu showed that they had zero actions and that P2 had one action, whilst P2 menu showed the opposite. Just like the previous one, it is still unknown what caused this bug, as it occurred after more than forty minutes of gameplay, and neither the logs nor debugging showed anything useful. This bug most likely has to do with one or more broadcasts that are transmitting erroneous information under specific conditions, which causes the games to be out of sync.

Both previous bugs only started occurring once the Erlang Sever was integrated with the game, further leading to the suspicion that these bugs are somewhere to be found within the broadcasts.

6.9 summary

The evaluation of the digital version of Meerplay would not have been possible were it not for Pinto's help in finding both a place of testing and, more importantly, the participants. The gameplay sessions conducted, although fairly small in both scale and time, allowed for several aspects of the game to be evaluated.

The sessions revealed that the game, like the original version, lacks in giving a clear explanation of how Petri Nets work and what they are used for, to allow first-time players to enjoy the game without another person's explanation of it. However, they also showed that the design of the digital game was good enough for players to understand it and be able to play autonomously, just like in Pinto (2022) version. Some players who had played the original version even recognized elements they still remembered from their previous sessions. All this allows one to conclude that the digital version's design is very similar to the original design, which was one of the main goals of this project.

Additionally, the other main goal of this was also met. The two gameplay sessions performed in all three groups revealed that even though players did not consciously notice the differences in both arena configuration and content quantity and placement, they naturally shifted their behavior from one session to the other in order to adapt to the game that was presented to them. The PCG algorithm has, therefore, increased the replayability value of this game.

Finally, some flaws and game-breaking bugs were also found during the gameplay sessions, which should be fixed in order to have a fully functional game.

Chapter 7

Conclusions

This final chapter serves to encapsulate what was achieved with the making of this project, future work that can be done in order to improve it, and finally, what knowledge was gained throughout the whole process behind the project's inception.

7.1 Results Achieved

With the development of this project, several results were achieved.

Firstly, a study was made on the concept and design of a game previously developed with the intent to promote the development of computation thinking skills in its players by encouraging self-efficacy and collaboration. This study furthered the understanding of both the game itself as well as how it helped players develop the previously mentioned attributes. This comprehension was necessary in order to be able to bring the game to a different medium in the form of a tabletop to digital transmediation.

Secondly, the aforementioned transmediation was performed, which resulted in a working digital version of the original tabletop game that attempted to be faithful to both its design and purpose. To make use of the possibilities the digital medium offers over the tabletop version of the game, several components such as interactive menus and animations were implemented. Furthermore, the game was integrated with a multiplayer server to allow players in different machines and locations to play together, effectively transforming the game from a solely local experience to an online one that can be experienced anywhere at any time.

Thirdly, a Procedural Content Generation algorithm was developed and incorporated into the digital version constructed in the transmediation process. This algorithm was developed in order to change the game's arena and content generation to provide different gaming scenarios every time the game is played, increasing its replayability value.

Fourthly, evaluation tests were made during gameplay sessions with participants in the target audience age range which provided a better insight into the game's design. These sessions showed that some elements of the game needed to be corrected or improved, but it also revealed that the game's transmediation was in fact faithful to the source material and that the PCG algorithm was capable of influencing player behavior by changing the game.

Lastly, the whole process required to create this digital game provided several lessons in game design that can be of help for further work in this project or other projects of similar foundations or goals.

7.2 Future Work

Despite the game proving to have accomplished its goals, several aspects of can be improved and expanded upon in order to increase its replayability and overall appeal.

- **Clearer player aid** - If the game is to be played without exterior participants helping new players understand its rules, it needs to do a better job of conveying to the players what each element is used for and how to use it. This is especially true when talking about the Petri Nets which, as shown above, were the game's component that gave players the most trouble.

  One thing that was avoided during the game design was to use any sort of text, as the game is intended to be for children of every background. The only place in which text was used was in the start menu and lobby buttons since these had to be as clear as possible. However, text might have to be used in order to help players during gameplay. The Petri Net elements, for example, could use some sort of label to describe what they represent to eliminate any sort of confusion from the players' minds.

  Additionally, short replayable videos could be used to visually explain to the players what the game elements are used for, such as a video showing the creation of a Petri Net and the player performing the action described by that Petri Net.

  Finally, something more interactive could be used to help players create their first Petri Net, such as arrows guiding the players to what they should click on whilst blocking any other game element, as is done at the start of a multitude of mobile games.

- **Zoom and drag** - One component that was to be incorporated in the game but showed to have several associated problems and was inevitably dismissed, was the ability to zoom in and out of the arena and drag the screen around. This was initially designed with mobile gaming in mind since the screen size of cell phones is smaller than that of computers. By zooming in, players would be able to have a better look at the arena and would also be able to drag the screen around in order to move their field of view. They would also have been able to zoom out in order to see the whole arena at once, as it is shown in the current version.

Although this was not implemented, it would be interesting to do so in the future. Especially if the error preventing the game from working on mobile devices is fixed.

- **Difficulty settings and modifiers** - Since the implemented PCG algorithm is entirely based on choices pulled from a pool of options, it is possible to influence these in order to change some aspects of the game.

  Prior to starting a game, players could be presented with a menu where they could change the odds of finding certain types of content, increase or decrease the number of predators or thieves, increase or decrease the number of bushes and trees, expand and choose the area where predators can spawn in, etc.

  Additionally, difficulty options could also be presented, where instead of the players having to manually select each component, predefined changes would be set, effectively changing the difficulty of the game.

- **Player communication** - In order for players who are not present in the same room to be able to communicate with each other, a voice or text chat could be implemented to allow them to engage in conversations. Alternatively, one other solution that was considered for this during development was to allow players to express their thoughts through emotes that could be universally interpreted, allowing players who speak different languages to communicate with each other.

  Additionally, one other possible addition to the game would be to show other players' mouses or screen presses on each player's screen. This could be useful for players to indicate to others where something is or what they should interact with.

  All of these components should also offer the possibility of being muted, should a player desire to do so.

- **Energy consumption** - One idea for the game developed by Pinto (2022) during the paper prototype development but that was eventually scrapped due to adding a higher degree of difficulty to the game was to have energy levels associated with actions.

  Every time a player performed an action it would not only reduce their total amount of actions, it would also decrease their energy level. Once a player's character was out of energy they had to go to sleep and wake up to restore their energy, otherwise, they were unable to perform anything due to being tired.

  This idea was too much for the paper version of the game since it added one more menu to which players had to pay attention as well as having to manually interact with it. The digital version, however, could add this component more seamlessly by automatically managing the energy of each player.

- **Day and night cycles** - One final idea that could be incorporated into the digital game is a day and night cycle. This concept could work in tandem with the previous idea of having meerkats go to sleep to restore energy, however, the main idea behind day and night cycles would be to modify the game's difficulty during gameplay. During the day players would face fewer predators whilst during the night the predators would be more common encounters.

Additionally, it could be further expanded to have thieves more common during the day and predators more common during the night or vice-versa, forcing players to constantly adapt to the game throughout the game session.

7.3 Acquired Knowledge

The development of this project allowed for a plenitude of knowledge to be gained from different areas.

The research performed for State of the Art provided a greater degree of awareness about the history of games. From their beginning with games like Senet, to the first video games in Tennis for Two (Higinbotham, 1958) and Spacewar! (Russell et al., 1962), to the first big commercial success in video games in the form of Pong (Atari, 1972), and finally, to the massive systems that comprise the games of today.

The study of the concepts of play and games also offered a higher degree of comprehension of what some experts believe defines each one of these terms and all their different approaches to it.

Game design is another area that was further understood via Salen and Zimmerman (2003) approach to it. Their fundamental ideas of game design, signs, and player choice helped build the digital version this project sought to develop.

The knowledge in the field of procedural content generation was also vastly increased with the study of not only what entails procedural content generation and what it is used for but also the study of different taxonomies describing the different types of PCG.

Finally, the creation of the digital prototype itself not only granted further comprehension of what game development entails, but the use of Godot for several months also provided a high degree of proficiency with the game engine.

References

Aarseth, E., & Calleja, G. (2015). *The word game: The ontology of an indefinable object.*

Adams, T. (2015, 4). Simulation Principles from Dwarf Fortress. *Game AI Pro 2*, 519–522. Retrieved from `http://dx.doi.org/10.1201/b18373-49` doi: 10.1201/b18373-49

Asmodee. (2022a). Retrieved 2022-12-10, from `https://corporate.asmodee.com/`

Asmodee. (2022b). *A game of thrones: The board game digital edition.*

Asmodee, & Dovetail Games. (2019). *Catan.*

Asmodee, & The Knights of Unity. (2018). *Scythe: Digital edition.*

Atari. (1972). *Pong.*

Barbara, J. (2015, 6). Measuring User Experience in Multiplayer Board Games. *Games and Culture*, *12*(7-8), 623–649. Retrieved from `http://dx.doi.org/10.1177/1555412015593419` doi: 10.1177/1555412015593419

Bay 12 Games. (2006). *Dwarf fortress.*

Bertz, M. (2011, 1). *The Elder Scrolls V: Skyrim Preview - The Technology Behind The Elder Scrolls V: Skyrim.* Retrieved from `https://www.gameinformer.com/games/the_elder_scrolls_v_skyrim/b/xbox360/archive/2011/01/17/the-technology-behind-elder-scrolls-v-skyrim.aspx?PostPageIndex=2`

Bethesda Softworks. (1994). *The elder scrolls ii: Daggerfall.*

Bethesda Softworks. (2011). *The elder scrolls v: Skyrim.*

BMG Interactive. (1997). *Grand theft auto.*

Booth, M. (2009). The ai systems of left 4 dead. In *Artificial intelligence and interactive digital entertainment conference at stanford, 2009.*

Braben, D., & Bell, I. (1984). *Elite.*

Caillois, R. (2001). *Man, Play and Games* (Reprint ed.). University of Illinois Press.

Cook, M., & Colton, S. (2014, 1). Ludus Ex Machina: Building A 3D Game Designer That Competes Alongside Humans. *ICCC*, 54–62. Retrieved from `http://computationalcreativity.net/iccc2014/wp-content/uploads/2014/06/4.2_Cook.pdf`

Craveirinha, R., Barreto, N., & Roque, L. (2016). Towards a taxonomy for the clarification of pcg actors' roles. In *Proceedings of the 2016 annual symposium on computer-human interaction in play* (p. 244–253). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2967934.2968086` doi: 10.1145/2967934.2968086

Crawford, C. (2011). *The Art of Computer Game Design.*

Crist, W. (2020, 01). Passing from the middle to the new kingdom: A senet board in the rosicrucian museum. *The Journal of Egyptian Archaeology*, *105*, 030751331989628. doi: 10.1177/0307513319896288

Dormans, J. (2010, 6). Adventures in level design. *Proceedings of the 2010 Workshop on Procedural Content Generation in Games.* Retrieved from http://dx.doi.org/10.1145/1814256.1814257 doi: 10.1145/1814256.1814257

Ekdahl, D. (2021, 05). Mechanical keyboards and crystal arrows: Incorporation in esports. *Journal of Consciousness Studies*, *28*, 30-57.

Electronic Arts. (2007). *Mass effect.*

Electronic Arts. (2013). *Battlefield 4.*

Electronic Arts. (2014). *Titanfall.*

Engström, H. (2020). *Game development research.*

Evolutionary Games. (2010). *Galactic arms race.*

Filby, J. (2022, 8). *Minecraft Twitch streamer breaks record by walking to the end of the game.* Retrieved from https://www.dexerto.com/minecraft/minecraft-twitch-streamer-breaks-record-by-walking-to-the-end-of-the-game-1913065/

FromSoftware Inc. (2008). *Dark souls.*

The Full Spelunky on Spelunky. (2011, 3). Retrieved from https://makegames.tumblr.com/post/4061040007/the-full-spelunky-on-spelunky

Galactic Map Puts Scale Of No Man's Sky In Perspective. (2014, 12). Retrieved from https://www.gameinformer.com/b/features/archive/2014/12/08/galactic-map-puts-scale-of-no-man-s-sky-in-perspective.aspx

GameDaily Connect. (2018, 9). *The Challenge of Adapting a Board Game to Digital | Philippe Dao.* Retrieved from https://www.youtube.com/watch?v=0aIye5EU9ko

Garvey, C., & Lloyd, B. (1990). *Play: Enlarged edition (the developing child).* Harvard University Press.

Gearbox Software. (2009). *Borderlands.*

Graetz, J. M. (1981). The origin of spacewar. *Creative Computing*, *7*(8).

Hastings, E. J., Guha, R. K., & Stanley, K. O. (2009). Evolving content in the galactic arms race video game. *2009 IEEE Symposium on Computational Intelligence and Games*, 241-248. Retrieved from https://api.semanticscholar.org/CorpusID:16598064

Hello Games. (2016). *No man's sky.*

Higinbotham, W. (1958). *Tennis for two.*

Huizinga, J. (1938). *Homo ludens: A study of the PlayElement in culture.* Taylor & Francis.

Juul, J. (2005). *Half-Real: Video games between real rules and fictional worlds.* The MIT Press.

Kraaijeveld, A. R. (2000). Origin of chess–a phylogenetic perspective. *Board Games Studies*, *3*, 39–50.

Lowood, H. (2009). Videogames in computer space: The complex history of pong. *IEEE Annals of the History of Computing*, *31*(3), 5-19. doi: 10.1109/MAHC.2009.53

Minecraft Interactive Experience. (2022). Retrieved 2022-3-12, from https://minecraft.fandom.com/wiki/Old_Customized

Mojang Studios. (2011). *Minecraft.*

Monnens, D., & Goldberg, M. (2015). Space odyssey: The long journey of spacewar! from mit to computer labs around the world. *Kinephanos: Journal of*

Media Studies and Popular Culture.

Mossmouth. (2008). *Spelunky.*

Motion Twin. (2018). *Dead cells.*

Møller, T., & Billeskov, J. (2019). *Expanding wave function collapse with growing grids for procedural content generation.* (Doctoral dissertation). doi: 10.13140/RG.2.2.23494.01607

Naughty Dog. (2013). *The last of us.*

Nintendo. (2018). *Super mario party.*

Park, W. (2021). *The ancient invention that ignited game play.* Retrieved from `https://www.bbc.com/future/article/20210318-the-ancient-invention-that-ignited-game-play`

Parker, J., & Jones, R. (2016). *Procskater.*

Pereira, L. L., & Roque, L. (2013, 1). Understanding the Videogame Medium through Perspectives of Participation. *Digital Games Research Association Conference, 7*. Retrieved from `http://www.digra.org/wp-content/uploads/digital-library/paper_326.pdf`

Pinto, V. A. M. V. (2022). *Design e estudo de uma experiência de jogo para promoção da diversidade de género nas stem* (Unpublished master's thesis).

Pivec, M., Dziabenko, O., & Schinnerl, I. (2003). Aspects of game-based learning. In *3rd international conference on knowledge management, graz, austria* (Vol. 304).

Riot Games. (2009). *League of legends.*

Russell, R., Graetz, M., Saunders, B., & Piner, S. (1962). *Spacewar!*

Salen, K., & Zimmerman, E. (2003). *Rules of play: Game design fundamentals*. The MIT Press.

Seidel, S., Berente, N., Martinez, B., Lindberg, A., Lyytinen, K., & Nickerson, J. V. (2018, 10). Autonomous Tools in System Design: Reflective Practice in Ubisofts Ghost Recon Wildlands Project. *Computer*, *51*(10), 16–23. Retrieved from `http://dx.doi.org/10.1109/mc.2018.3971341` doi: 10.1109/mc.2018.3971341

Shaker, N., Togelius, J., & Nelson, M. (2016). *Procedural Content Generation in Games*. New York, United States: Springer Publishing.

Shotwell, P. (1994). The game of go: Speculations on its origins and symbolism in ancient china. *Changes*, *2008*, 1–62.

Smith, G. (2014, 4). Understanding procedural content generation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Retrieved from `http://dx.doi.org/10.1145/2556288.2557341` doi: 10.1145/2556288.2557341

Smith, G. (2017). Procedural content generation: An overview. *Level Design Processes and Experiences*, 159–183.

Solly, M. (2020, 2). *The Best Board Games of the Ancient World.* Retrieved from `https://www.smithsonianmag.com/science-nature/best-board-games-ancient-world-180974094/`

Stahlhacke, P. (2003). *The game of lasker morris* (Tech. Rep.). Retrieved from `http://www.althofer.de/stahlhacke-lasker-morris-2003.pdf`

Steam. (2020, 10). *How to adapt a board game into its digital version: Scythe | Steam Digital Tabletop Fest.* Retrieved from `https://www.youtube.com/watch?v=3ByiCX86GZM`

Stonemaier Games. (2016). *Scythe.*
Stony Brook University. (2013, 5). *When Games Went Click: The Story of Tennis for Two.* Retrieved from `https://www.youtube.com/watch?v=6QSHZ20MQfE`
Strategic Simulations Inc. (1998). *Ad&d dungeon master's assistant.*
Sweidan, S., & Darabkh, K. (2018, 01). Vreg: A virtual reality educational game with arabic content using android smart phone. *Journal of Software Engineering and Applications*, *11*, 500-520. doi: 10.4236/jsea.2018.1110030
Tactical Studies Rules. (1974). *Dungeons & dragons.*
Togelius, J., Champandard, A., Lanzi, P. L., Mateas, M., Paiva, A., Preuss, M., & Stanley, K. (2013, 01). Procedural content generation: goals, challenges and actionable steps. *Dagstuhl Follow-Ups*, *6*, 61-75.
Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011, 9). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172–186. Retrieved from `http://dx.doi.org/10.1109/tciaig.2011.2148116` doi: 10.1109/tciaig.2011.2148116
Ubisoft. (2014). *Monopoly plus.*
Ubisoft. (2017). *Ghost recon: Wildlands.*
Ubisoft Montreal. (2012). *Far cry 3.*
Valve Corporation. (2008). *Left 4 dead.*