CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Video Game Genre Blending: A Prototype Study of the Combination of Extraction and Deck-builder Mechanics

Master's thesis in Computer Science and Engineering

Levente Varga

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Video Game Genre Blending: A Prototype Study of the Combination of Extraction and Deck-builder Mechanics

Levente Varga

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Video Game Genre Blending: A Prototype Study of the Combination of Extraction and Deck-builder Mechanics

Levente Varga

Supervisor: Natasha Mangan, Department of Computer Science and Engineering
Examiner: Michael Heron, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Video Game Genre Blending: A Prototype Study of the Combination of Extraction and Deck-builder Mechanics

Levente Varga
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis presents the design and development of an experimental video game prototype that merges the mechanics of extraction and deck-builder games, incorporating procedural generation and dungeon-crawler elements with a rogue-like meta-progression system. Developed using the Godot engine with C#, the project explores this novel genre fusion and focuses on creating engaging player experiences through carefully balanced game mechanics. The results demonstrate the feasibility of combining deck-building and extraction elements, providing insight into the potential of hybrid game genres.

Keywords: Computer science, video game, card game, extraction, dungeon.

Acknowledgements

I would like to thank my supervisor, Natasha, for being supportive, patient and always available throughout this one-and-a-half-year journey. I also want to thank my friends and family who helped me stay on track during the design and development phases of the project. Finally, I would like to thank everyone who participated in the playtests.

Levente Varga, Gothenburg, 2025-06-18

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

In the second half of the twentieth century, often referred to as the dawn of the digital age or the prehistory of the computer game[1], video games started to emerge as a new leading medium. Unlike today's blockbusters, which are often the result of the collaboration between multiple international studios, early video games were typically developed by just a handful of people and featured limited and basic gameplay elements and mechanics.

Over time, genre boundaries began to blur, while entirely new genres started to form. At the same time, the gaming industry has undergone a rapid evolution in both production scale, quantity, and variety, and introduced new hybrid game genres through a process known as genre blending[2], which combines key concepts from existing genres. This process has led to increasing commercial success, as genre-blended games usually stand out with their unique mix of game mechanics and diverse gameplay, appealing to a broader audience.

1.1.1 A brief history of video games

It is challenging to tell exactly when the video game era began, as there is ongoing debate over what constitutes the first video game. Some consider it to be Tennis for Two (1958)[3], an abstract 2D simulation of tennis played on an oscilloscope. However, its creator, Willy Higinbotham, developed it solely to demonstrate the capabilities of the hardware it was running on. As a result, there was no scoring system, nor any other feature that would resemble a modern video game[1].

A stronger candidate for being the first true video game is Spacewar (1962) [4], developed by Steve Russell during his time in college. Unlike Tennis for Two, Spacewar was intended to be a way of entertainment for two players. It was programmed on PDP-1 mainframe computers and featured a simplified representation of space with two controllable spaceships, a black hole located in the middle of the screen, and a few blinking stars for decoration. Players competed to destroy each other's spaceship using torpedoes while avoiding crashing into the black hole. Since Spacewar was a computer program intentionally developed for entertainment, it is widely regarded as the first true video game. Furthermore, it laid the groundwork for the action and simulation genres that emerged shortly after[1].

Interestingly, most people would mistakenly believe that the first video game was Pong[5], probably due to its massive popularity. However, it was released a full decade after Spacewar, in 1972.

1.1.2 Genre combinations

In 1993, Doom[6] was released, a prime example of a genre being born. Even though Doom was not the first of its kind, and just like other games, it was a combination of already existing game types like action and shooter, it became so successful and popular that the common term to describe similar games became "doom-clone". Only a couple of years later did a proper name "first-person shooter" (FPS) for these types of games appear and gain mass acceptance[2], thus giving birth to a new video game genre.

Soulslike games share a similar story, although 'Soulslike' is not considered a genre of its own, but rather a subgenre. Its name comes from games developed by From-Software, a game development studio which is mainly famous for its games of high difficulty, namely Demon's Souls and the Dark Souls trilogy, all built around a very similar theme and game aesthetics, dynamics and mechanics[7]. The oldest entry of the series, Demon's Souls, was released in 2009, so they are considered relatively new, therefore a better and more descriptive name for this sub-genre has not been found yet.

1.2 Thesis

1.2.1 Motivation

Although numerous genre combinations already exist, there are still many unexplored possibilities that could lead to the creation of engaging and innovative games. By examining existing combinations and identifying gaps in the market, this research aims to propose new genre fusions that could offer new gameplay experiences. The ultimate goal is to develop a game based on one of these novel combinations.

Due to the limited time available for the thesis, the development of a complete game is unlikely. Instead, the feasibility of the proposed genre blend will be demonstrated by a game prototype that will retain all the characteristics and core concepts of each chosen genre, with limited visual appeal and polish. More about limitations will be discussed in Section 7.2.

1.2.2 Research questions

The core motivation driving this research is to find out whether it is feasible to blend elements of a specific genre combination into a single, functional gameplay experience that has not been done before. The first part of this study is to discover these combinations and analyze their feasibility, then, as the second part, test it using a game prototype and gain feedback from playtesting sessions to examine the practical implications and player perceptions of such a hybrid game.

The following research questions will guide this research.

RQ1: *What is one promising genre combination for the creation of a new game type?*

The first part of the thesis will answer RQ1, observing which video game types exist in the smallest number, and which two forms a combination that results in a new game type. Finally, a combination will be chosen to be implemented into a game prototype to help answer RQ2, which is:

RQ2: *What implications can be derived by implementing a game based on a combination of specific game genres?*

Since RQ1 serves as the base for the development of the game prototype, it must be answered earlier than RQ2. For that, Chapter 4 was introduced. Thus, Chapter 5 will present only the implementation of the game prototype.

1.2.3 Goals and Challenges

The main goal is to come up with a new video game genre combination and develop a functional game prototype that successfully combines all the elements and mechanics of the selected genres. Even if the resulting gaming experience turns out to be unenjoyable, it could still serve as a valuable lesson and a good indicator that these genres might not fit well together.

The core challenge - as with most games - is to achieve player satisfaction. Players who become frustrated and do not enjoy playing the game could indicate a potential design flaw and the wrong combination of genres. It could also mean that not the right players were selected for testing; therefore, it is crucial to conduct a brief 'pre-screening' of playtesters to ensure that the potential target audience are the ones testing the game. This is an attempt to avoid unnecessary dissatisfaction and thus unwanted outliers in the feedback data.

2

Theory

This chapter is about the literature review that will serve as a valuable background for future research.

2.1 Defining video game genres

2.1.1 Chris Crawford

The first attempts to classify video games occurred years after the first video game was released. In his book published in 1984, Chris Crawford listed games in two major categories, namely skill and action games (S&A) which rely heavily on hand-eye coordination and motor skills and are usually fast-paced, and strategy games that emphasize more decision-making, and due to their nature, they generally take longer than S&A games[8]. Within these main categories, Crawford defined multiple subdivisions for finer categorization. Some of these categories have evolved into modern game genres and are still present today, like sports and adventure games, while others, including paddle games, D&D games and games of chance have completely merged into other genres and do not exist on their own. It is important to note that Crawford did not refer to these categories as genres.

2.1.2 Mark J. P. Wolf

Many of the first definitions of true video game genres were formulated by Mark J. P. Wolf, who also compared video games with the other existing media types. Still, mainly cinema [9]. He concluded that while many similarities exist, video games and movies cannot be categorized in the same way, due to their differences in interactivity, that is, video games require active participation from the audience. Thus, the 41 genres and subgenres explained in his paper[9] are based on interactivity aspects, instead of iconography. The border between some of these genres is blurry. Racing and Driving, Catching and Capturing, Obstacle Course, and Platform all have a surprisingly similar definition to each other, differing only in a few parameters; however, these similarities are explicitly stated by the author.

2.1.3 Thomas H. Apperley

Another approach was made in 2006 by Thomas H. Apperley, who agreed with Wolf that games cannot be categorized like other media types. He also criticized earlier game categorization attempts, mainly because they are "loose aesthetic clusters based around video games' aesthetic linkages to prior media forms"[10]. Instead, his proposed framework tries to shape a new set of genres around the style of "ergodic interaction" that takes place within the games. In his paper, he defined four distinct video game genres. *Simulation* games mimic the intricate processes of real-life systems, such as cars or cities. *Strategy* games, including real-time strategy (RTS) and turn-based strategy (TBS), are outliers due to their similarity to board games. These games usually have a god's-eye view and information-heavy interfaces. *Action* games include first-person shooters (FPS) and third-person games and focus more on the player's performance in combat manoeuvres and aiming with weapons, and unlike strategy games, they do not contain complex decision-making processes. And finally, *role-playing* games (RPG), often compared to pencil-and-paper RPGs such as Dungeons and Dragons, usually revolve around fantasy themes and focus on some form of character development, such as the collection of experience or skill points.

2.1.4 Ludwig Wittgenstein

Another perspective for understanding game genres comes from the work of philosopher Ludwig Wittgenstein who introduced the concept of family resemblance[11] to explain how categories often lack clear definitions, yet still form recognizable groups based on overlapping similarities. He argued that, just as members of a family may have some form of physical resemblance, concepts and categories can also be defined by a set of similar features. This idea of family resemblance can also be applied to video game genres.

While early attempts at classifying games, such as those by Crawford, Wolf, and Apperley, outlined specific criteria, modern genres rarely fit into rigid definitions. Instead, many games share overlapping mechanics, aesthetics, and dynamics, creating unclear genre boundaries. For example, in the case of genres defined by Wolf, Racing and Driving, Catching and Capturing, Obstacle Course, and Platform games all share similar gameplay elements, differing only in minor parameters. Similarly, hybrid games that combine mechanics, such as Action-RPGs or Tactical Shooters, demonstrate that genre boundaries are not strict taxonomies, but flexible groupings based on recognizable patterns.

2.1.5 Conclusion

The early video game taxonomies were not applicable to the modern genres that followed[12]. Such complex genres as Massively Multiplayer Online Role-Playing Games (MMORPG) or Multiplayer Online Battle Arenas (MOBA) could not fit into the existing categories that were shaped, as they are a mix of several already defined game types (Multiplayer and Role-Playing).

Overall, it can be observed that the video game industry is and has always been

struggling to come up with a universal organization system for games. Existing categorization methods, such as those used for books or movies, could not be applied to games due to the differences between the mediums[13].

Instead of sticking to one taxonomy, Steam[14], an online video game store, uses a list of predefined, as well as user-defined tags that consist of one or a few words, to categorize games published on the platform. Tags are similar, and sometimes identical, to the name of genres discussed in the above-mentioned papers, however, there is no definition linked to them. What each tag means is determined by the list of games assigned to them on the platform. This enables greater freedom in video game categorization than sicking to a chosen taxonomy would, since tags can expand and adapt to new game types, while taxonomies remain the same.

2.2 Finding genre combinations

Today's most popular game store on PC, Steam[14], which, at the time of writing, has a library of over 204.000 apps, including the majority of modern video games. There are some exceptions, however, as some games are platform-specific and thus have never been released on Steam, but the number of such games can only be measured in the hundreds, so they will be considered insignificant for this research.

Steam uses two systems to categorize this massive amount of content. First, developers and publishers pick from a list of pre-defined genres to assign to their apps during upload. Second, users who download these apps can assign them user-defined tags that can indicate genre, theme, aesthetics, or even difficulty, and help other users find new apps that are to their liking. From the user's point of view, only the tags are visible on the apps' dedicated pages while genres are only used as filtering in the store's search field.

Steam has a public API (Application Programming Interface) that can be used to query data about their games and apps. This API, however, is not well-documented, thus it would take a considerable amount of trial and error to use it properly. Luckily, a relatively fresh dataset[15] can be found on GitHub that, according to the author, was acquired in October 2024 using the Steam API and contains all the data necessary to find existing genre combinations on Steam. To check exactly how up-to-date this data set is, I checked the number of games under some tags on Steam and compared them with the numbers found in the data set, and the differences were around 40%, which is unexpectedly high. However, games released in October 2024 can be found in the data set as expected with the correct release dates, indicating the data was truly acquired in or after October 2024. The explanation for such a large difference in the numbers could be explained by the ever-rising number of new games published each month on the platform and the new tags applied to games during this two-month period. However, the ratio between the tag occurrences could not change significantly in such a short time. These data are in CSV format which is mostly used by spreadsheet editors such as Excel or Numbers, but can easily be read from script too.

Going through the data set and individually counting the times each tag occurs, it is

easy to determine that Steam currently uses 447 tags and 121 genres. It can also be observed that tag occurrences, as seen in Figure 2.1, like many other data sets, seem to follow Zipf's law loosely[16]. Interestingly, some of the tags make little sense, like "Intentionally Awkward Controls" or "TrackIR", and do not represent a genre, but rather some other aspect of the game. There are a few tags like "Warhammer 40K" that solely indicate which franchise the game belongs to.

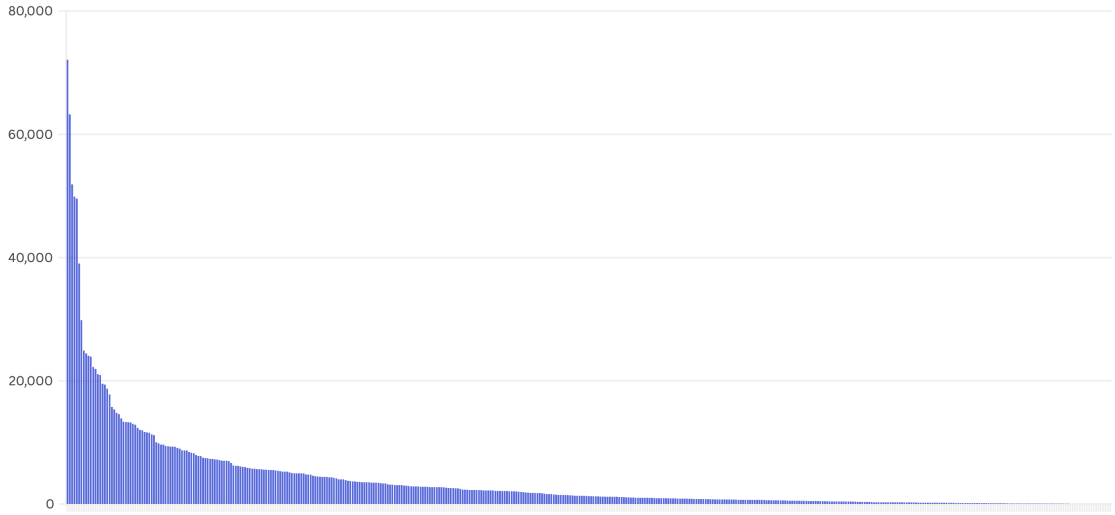

Figure 2.1: Tag occurrences

On the other hand, the genre distribution in Figure 2.2 seems unbalanced. The reason for this is that for some reason, 84 of the 121 genres are localized versions of others, and each appears less than 30 times in the entire data set. These genres can be safely discarded, as their occurrence times would not contribute too much to the other numbers.

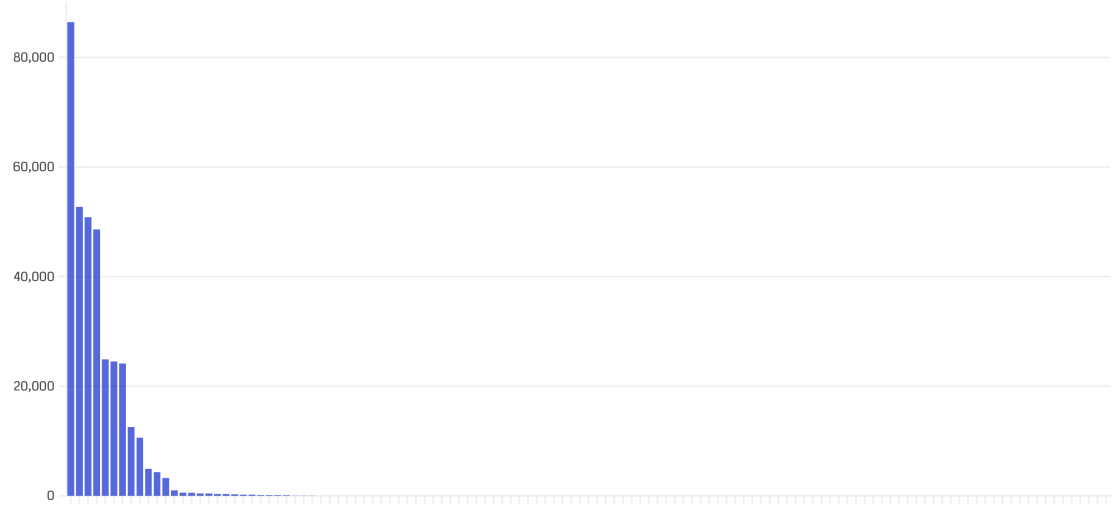

Figure 2.2: Genre occurrences

Steam uses only a handful of genres to categorize games, and therefore these genres

Genre	Occurrence
Indide	86425
Action	52753
Adventure	50835
Casual	48633
Simulation	24916
RPG	24510
Strategy	24146
Early Access	12568
Free To Play	10639
Sports	4922
Racing	4319
Massively Multiplayer	3258

Table 2.1: Top 12 genres and their occurrences on Steam

are vague. As visible in Figure 2.2, only 12 genres occur more than 1000 times, these genres and their number of occurrences can be seen in Table 2.1.

After these, the list of meaningful genres practically ends. Although I could start combining these genres to come up with possibly new combinations, they are too ambiguous. Instead, for this thesis, I will use Steam tags, as they provide a much finer way to describe what genres the developed game prototype should belong to and what types of game mechanics it uses.

The author of the data set also looked for genre combinations, as seen in Figure 2.3, but only used combinations of two genres, while I was more interested in more complex mixtures. Furthermore, they only included tags assigned to at least 900 games in the Steam library, and that criteria takes away almost half of the tags. For this thesis, I was interested in the other half of the tags, those that are used rarely but at the same time clearly represent a genre or sub-genre. To extract exactly the data I needed, I had to write my own script which will be discussed in Chapter 4.

2.3 Game engines

This thesis will include the development of a game prototype, and to reduce development time, a game engine will be used. There are many freely available game engines on the market at the time this thesis is being written, so to pick one, a short comparison needs to be made.

But first of all, what is a game engine? To answer this question, it would be a good idea to look back at the very first game engine. Doom Engine was developed for Doom (1993) by id Software, it was the first of its kind [17], the very first piece of software that had its components well-defined and separated, allowing developers to make different games with minimal modifications made to the game engine itself.

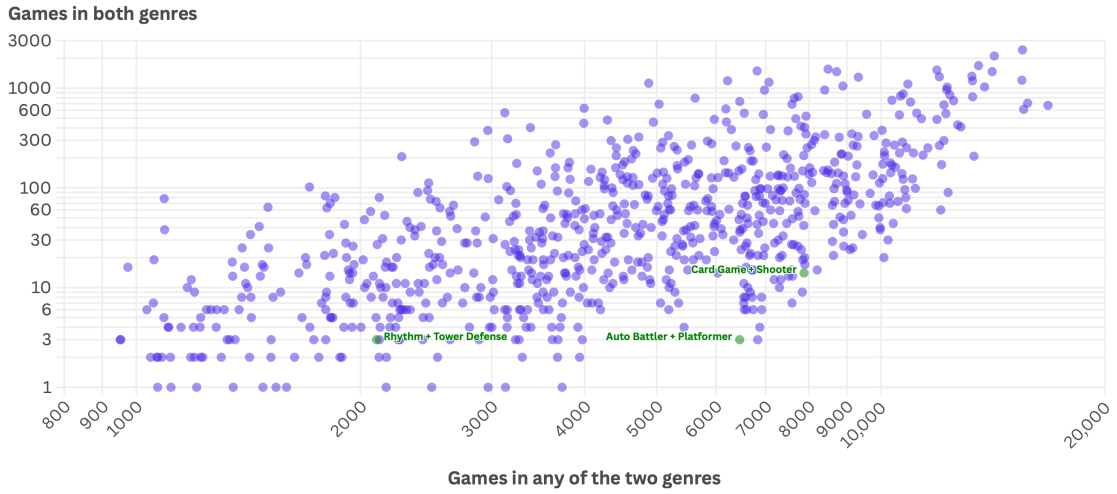

Figure 2.3: Most frequent genre pairs

Soon many major game development studios started to develop their own game engines, while licensing other's already working engines started to become a norm.

Developers today can also choose to use free-to-use game engines such as Unreal Engine [18] or Unity [19]. These engines are usually for general purposes, meaning that they include a handful of features that make it possible to develop any kind of video game with them, with the side-effect of being hard to learn on a professional level. Additionally, they are not entirely free, as there is usually a limit above which developers have to start paying a share of their total revenue to the engine.

There are more compact alternatives, such as Godot[20] or LÖVE[21], which are completely free and open-source, meaning that developers can actively contribute to their code base to implement new features. However, they often lack features found in more robust engines.

2.4 Game development

Creating video games is a combination of several professions. It includes programming, game design, asset creation such as sprites, 3D models, music and sound effects, and even marketing. Dedicated tools can be used for each of these areas; some of them are discussed in Axel Ljungdahl's master's thesis [22] on individual game development. Axel's work is considered extremely valuable regarding this thesis, as it provides important tips and guidelines for how a game prototype should be developed individually, using only affordable tools.

For programming, code editors, or IDEs (Integrated Development Environments) can be used to help with coding syntax, formatting, and debugging, as well as code completion.[22] Popular options include Visual Studio[23], Visual Studio Code[24] and the JetBrains[25] suite. Some game engines, such as Godot, even have their own integrated code editors.

For creating graphical assets, like sprites or textures, one can choose between raster graphics, storing images as a grid of rectangular pixels, and vector graphics, storing images as mathematical models that can be scaled without using detail.[22] Respectively, free tools such as GIMP [26] and Figma [27] can be used for these approaches. If the game is three-dimensional, it usually requires 3D model assets, which can be created using Blender[28].

Music and sound effects can be created with free tools such as LLMS. Existing audio files can be edited using Audacity. Audio can also be recorded using a regular microphone found inside an average smartphone or laptop. Some games, like Samorost 3[29], use sound effects produced solely by mouth and real instruments.

3

Methodology

This chapter describes the core problem and how this research plans to find answers to it.

3.1 Approach

Planning: To find the answer to RQ1, the thesis will begin with a short research and experimentation where the genres that are used the least frequently will be explored to determine whether any two of them could function well together. To help with this research, Steam's[14] video game database will be used, as it contains most modern video games and their necessary details. As a result of this part, a genre combination will be chosen with which the thesis can continue to its second part.

Implementation: To answer RQ2, the project will involve the design and iterative implementation of a game prototype that combines all the chosen game genres from Planning and their corresponding elements and game design patterns.

During the project, between iterative cycles, the prototype will be regularly tested on small groups of players to gain instant feedback that could help the development stay on track.

At the end of the project, a formal game testing session will be conducted that will include presenting the game prototype to potential players and letting them play with it for a short time while being guided and supervised. At the end of the session, using a feedback form and verbal questions[30], anonymous data will be collected that will show how the combined gameplay design patterns work together and what the players think about their experience. Additionally, the game will be internally evaluated using the extended version of the GameFlow model[31].

Finally, the feedback collected during the tests will be reviewed and reflected on. This involves data collected from surveys, but also statistical data from the game itself that can be visualized using various methods[32].

Furthermore, during the thesis, regular meetings will be conducted with the project supervisor Natasha Mangan to discuss progress, deal with any problems at hand, and plan the next steps.

Task	Start	End	Length
Detailed literature review	December 1	January 31	9 weeks
Exploring genre combinations	January 1	January 31	4 weeks
Developing the game prototype	January 15	April 1	10 weeks
Development testing	February 1	April 1	8 weeks
Game testing	April 1	April 15	2 weeks
Reacting to feedback	April 15	April 26	2 weeks
Finishing up the Final Report	April 29	May 31	5 weeks

Table 3.1: The list of planned deadlines for the project

3.2 Tools

To create the prototype, preferably an existing game engine is used. As discussed in Section 2.4, there are several free-to-use game engines available, such as Unity, Unreal Engine, and Godot, just to name some of the largest. There are many tutorials, guides, and documentation available online for all three game engines. For this thesis, Godot will be used, because it is completely free, easy to learn, lightweight, and user-friendly, and it packs the necessary features for the intended game prototype.

Godot uses its own scripting language called GDScript, which is close to Python. However, in this thesis, Godot Mono will be used, which works with C# instead, a much more widely used language.

To keep the code organized and secure, GitHub, a Git version tracking system, will be used. This way changes in the code can be tracked, and if anything goes wrong, it is possible to revert back to a previous version.

3.3 Time plan

Table 3.1 contains the planned deadlines. Although this is a 30-credit master's thesis project, the time span is longer than the course schedule would suggest, because the work was started earlier.

3.3.1 Expected limitations

To allow more time for the development of the prototype, the research was affected by the following limitations.

- An existing data set of games was used, which was not up-to-date. Getting the most recent data would have taken significant extra work while providing minimal impact on the result.
- Given the vast amount of possible combinations of video game genres, the complete list was not discussed. Instead, a few promising genres were pre-selected and tested to see if their combinations exist.

- Steam tags were used to find existing genre combinations. Although tags do not necessarily represent genres, Steam has an extensive database of its library, and tags are the most useful way of categorization found during research, making them a viable choice.

The game prototype was struck by the following limitations due to the limited amount of time:

- The graphics, audio and other assets used were primitive to save time.
- The focus was on developing the core gameplay, instead of decorative features like a detailed settings page, achievements, or advanced accessibility features.
- The prototype was primarily developed for Windows, without any planned support for other platforms.
- The game only supports the 16:9 aspect ratio, since this is the most popular. Taking care of other aspect ratios would have been a waste of time with respect to the goal of the thesis.
- There is no tutorial or any proper introduction to the game and its mechanics, since during playtests, players were given guidance and instructions when it seemed necessary.

3.4 Ethical considerations

Anonymous feedback data will be collected during playtests. All playtesters will be informed about this in advance. Participation in the playtests will be voluntary and the participants can opt out at any time if they do not want to continue the playtest.

Player accessibility must also be taken into consideration. This means using a color palette that can be easily distinguished by players with color blindness, supporting multiple controller types, and preferably re-mapping buttons. However, this thesis can focus only on so many things. Paying attention to every small detail, including accessibility, is not the top priority right now, as the main goal is to check whether the new game type works or not. If it does work, then the game can be extended with various accessibility features in the future.

4

Planning

This chapter will explain the process behind finding an answer to the first research question, including the selection of a genre combination to work with, and then explores ideas for possible game mechanics to use in the prototype developed later in Chapter 5.

4.1 Coming up with a genre combination

Before real work on RQ1 started, during a few brainstorming sessions, a few ideas for genre combinations that intuitively felt like they were still unexplored had already been written down for later validation. The most promising among these were the "Souls-like deck-builder" and "extraction deck-builder" genre pairs. Coincidentally, on 20 January, a few days after writing my idea down, the demo for Death Howl[33], the first ever Souls-like deck-builder on Steam, was released, essentially rendering that genre combination obsolete. So I was left with the extraction deck-builder combination, which is the combination of the "Extraction Shooter" - without the shooter aspect - and "Deck-building" concepts. Extraction on its own does not exist as a separate tag on Steam yet. This mix proved to be unique after conducting the following research.

Initially, a script was written that generated a list of all possible combinations of any two tags found in the dataset and sorted it in descending order based on the number of games on Steam that included both tags in the pair. In this way, an overwhelming large amount of pairs were generated of which the data set contained zero games, most of them made up by tags that do not represent genres, but rather a specific concept, element, or aesthetic of the game, such as "Fox", "Birds", "Elf", and many more.

After reviewing the data set discussed in Chapter 2 and going through the less popular tags while filtering out those that do not represent a clear genre or game mechanic, still, a lot of tags remained. There were 195 tags that appeared less than 1000 times in total. For comparison, the most used tag, "Singleplayer", appeared 72120 times. Furthermore, there were only 38 tags with which fewer than a hundred games were tagged. To limit the number of genre combinations to choose from, it seemed ideal to include at least one tag that had such a low occurrence. "Extraction Shooter" is one of the least used tags, the 33rd in the list, counting from the back, with an occurrence of 84, which confirmed my intuition of this genre being

rarely used. Interestingly, at the time of writing, the number of games tagged with "Extraction Shooter" has already grown to 212, which is a 152% increase compared to the data set, indicating that it is a trending tag. For comparison, the occurrence of "Singleplayer" has only increased by 59%.

Thus, using Extraction Shooter as a starting point seemed like a good idea for the following three reasons:

- There are only a handful of games that are tagged with it.
- It is popular based on the increase in its occurrence over just a couple of months.
- I personally like the high-risk, high-reward concept, which extraction shooters are known for.

When combining the Extraction Shooter tag with Deck-building, which was also chosen based on personal preference, there is not a single game in the data set that matches this combination. At the time of writing, however, there are two games on Steam called Echo Chambers and Build and Extract, but both of them are upcoming games without a set release date, and based on their description and screenshots found on Steam, they severely differ from the type of game this thesis aims to explore. Therefore, the combination of extraction and deck-building mechanics can be considered a truly new mix.

4.2 Defining Extraction Deck-builder

To better understand what exactly the term Extraction Builder means, each genre of which it consists of will be defined using definitions of gameplay design patterns [34].

4.2.1 Deck-building

Magic: The Gathering[35], published in 1993, is the first collectable card game (CCG); it laid the foundations of the genre and inspired countless other games. In CCGs, players collect cards by purchasing them with real money and assemble their decks before the game starts. There is usually some limitation to the deck construction process, for example, in Hearthstone[36], the player must have exactly 30 cards in their deck, and each unique card can only appear a maximum of 2 times, not more, while in Gwent[37], the number of cards in a deck must stay between 25 and 40.

Dominion[38] from 2008 is considered the first deck-builder game to define the genre. Contrary to collectable card games, in deck-builders players build their deck during the game, not before. Starting with a small set of cards, players collect a currency that they can spend on better cards that they add to their deck, becoming stronger. The genre itself does not define the rules that imply exactly how the cards work and interact with each other; they are unique to each game. Occasionally, in some deck-builders the player has the ability to alter some card properties. For example,

in Balatro[39], a modern deck-builder video game from 2024 that uses traditional playing cards, the player can add special abilities to the cards that drastically change the gameplay, and even modify their suit or value. These mechanics can also be observed in some CCGs, as in Hearthstone[36], some cards modify their effects depending on the cards the player has previously played, holds in their hands, or has in their deck.

The closest genre to deck-building from Mark J. P, Wolf's taxonomy[9] is Card Games, Strategy from Apperley's[10] more ambiguous categories, and Games of Chance in Crawford's earlier list[8].

The gameplay design patterns used in deck-builders are the following:

- **Cards**: the main **Resource**, with unique **Abilities** or some other game-altering effects.
- **Card building**: the modification of cards in various ways. Although not all deck-builders utilize this pattern, it can be observed in Hearthstone[36] and Balatro[39].
- **Decks**: a collection of cards that through shuffling, introduces the **Randomness** and **Luck** patterns.
- **Deck Building**: putting newly acquired cards in the player's deck for it to be drawn later. The core gameplay pattern of deck-builders.

4.2.2 Extraction Shooter

This type of game is a recent alteration of the Tactical Shooter genre. Tactical Shooters, and Shooters in general, focus on using firearms as the main tool for destroying enemies or other players. One of the most popular Shooter sub-genres is FPS (First Person Shooter), where players control the game from their character's point of view. The Extraction Shooter genre first appeared as a PvE (Player versus Environment) game mode in Tom Clancy's The Division[40] in 2016, called the Dark Zone, and one year later Escape from Tarkov[41] became the first shooter game that was built around the extraction mechanic. This kind of game is played in iterations called runs and features an open world that the player enters with some equipment they have gathered in previous runs. During a run, the player has to explore and collect loot while fending off other players or NPCs (non-playable characters). At the end of a run, they have to reach an extraction point where they can safely leave the match with all the collected loot, adding it to their inventory outside of the match. However, if the player dies during a run, all the loot that they have collected, as well as the equipment they have brought in, will be lost forever. However, there are usually some twists, such as a safe pocket that keeps its content upon death. This type of game falls under the Combat Games category in Crawford's list[8], Action in Apperley's, and Shoot 'Em Up in Wolf's taxonomy, while also borrowing elements from the Collecting genre. Interestingly, Extraction Shooters do not fit Wolf's Escape genre, as it does not allow the player to fight back.

The recently increased number of upcoming extraction shooter games shows that

this novel genre is gaining popularity. Huge names in the industry like Bungie and Embark Studios are actively developing new extraction shooter titles called Marathon[42] and Arc Raiders[43] respectively.

4.2.3 Extraction

The term "Extraction" has not been used officially as a genre on its own yet, it is a made-up genre for this thesis only. To put it simply, it contains every game element Extraction Shooter does, except those related to the Shooter part, making it a more general category. It could be combined with other major genres, like Platformer or Strategy, but in this thesis, it will be mixed with Deck-building.

The definition of extraction games is formulated as the following:

Run-based games where players collect resources which they must transport to extraction points to retain them.

Furthermore, players manage two inventories: an inner inventory used during runs and an outer inventory for permanent progression. Players can transfer loot between inventories outside runs. A run ends either by reaching an extraction point or by dying, which results in the permanent loss of loot in the inner inventory.

There are a few existing video games that fit into the extraction category, but are not extraction shooters. Deep Rock Galactic[44] for example, while containing some shooter elements, focuses more on collecting loot by mining and safely returning it to the extraction point. In Lethal Company[45] players have to explore a procedurally generated facility, safely extracting items, while surviving against enemies, without a serious focus on combat. Even Soulslike games can be considered closely related to extraction games, where bonfires serve as extraction points and the soul is the loot, but the loot is not lost permanently if the player manages to retrieve it before dying for a second time. However, while, for example, Dome Keeper[46] definitely contains some extraction elements, it completely lacks the option of permanent loss of loot.

The core gameplay design patterns include the following:

- **Loot**: The collectible valuables that need to be extracted. Usually some kind of **Resource**, **Equipment** or currency.
- **Inventories**: The storage in which the player keeps the collected loot.
- **Rescue**: The closest pattern to extraction, rescuing someone or something like loot that is otherwise not free to move by its own will.
- **Enemies**: NPCs that are trying to stop the player from extracting loot, usually by attempting to kill them.
- **Replayability**: Due to the run-based nature of extraction games, they are highly replayable, and each run usually differs from the last.
- **Death Consequences**: Dying results in the collected loot being lost.

4.2.4 Mixing Deck-building with Extraction

The resources in extraction games do not have a specific type; however, since Deck-builders already work with cards, which can be considered a resource, it makes sense to use cards as the obtainable loot in Extraction Deck-builders. Card loots become even more interesting with the fact that cards are the key gameplay elements of Deck-builders, so gaining new cards can drastically change the gameplay while the run is in progress by allowing the player to immediately put the newly collected cards in use. On the other hand, making such a vital game element permanently loseable is a risky move, since players can easily get frustrated. But keep in mind that this is exactly what Extraction Shooters do, just instead of cards, those games have weapons and gear as the key resource. To allow the player to secure at least a small portion of the gathered loot, some options need to be provided, like the already mentioned safe pocket, or a system that in exchange for some in-game currency, ensures that an item will be kept if the player dies.

In order to make an extraction game, the player needs to extract loot from somewhere. In general, this 'somewhere' could be a large open-world map similar to Tarkov[41], but for this thesis that would set the scope extremely high. A more reasonable option is to use an abstract 2D grid-like world that can still be large, but it is much easier to maintain due to its simplicity. To achieve the feel of a large-scale map, a procedural generation algorithm[47] could be used to dynamically create the map as the player progresses. To keep things contained and manageable, the algorithm could place predesigned rooms next to each other, connecting them with doors, just like in Enter the Gungeon[48] or The Binding of Isaac[49], greatly saving development time.

In Extraction Deck-builders, the gameplay design patterns of Deck-builders and Extraction games get mixed too, resulting in the following list:

- **Cards** become the **Loot** and **Resource**.
- **Inventories** become **Decks**.

4.2.5 Further ideas

Apart from the essential game elements of Extraction Deck-builders, a currency system could be used in the upcoming game prototype to make room for some permanent progression in between runs. The currency could be a type of resource that can be spent on upgrades such as the safe pocket, and is always kept, i.e., at least partially, regardless of whether the player completed the run successfully. Otherwise, if the player lost a run and had no means to keep part of the loot, they would also lose the currency, thereby losing the motivation to continue.

Using cards for as many actions as possible could be another interesting twist. This would include movement, opening chests or doors, picking up items, etc. If the player cannot play a movement card for any reason, they are simply unable to move. However, this rule could greatly limit the player and might turn out to be annoying, so some basic actions should not be tied to playing the right cards. One way of

dealing with this is to use multiple hands and decks, one for each category of action: Movement, Combat, and Interaction. Each deck could have their own cards and rule sets. For example, the Movement deck can not be depleted, while the Combat deck needs a rest to reshuffle, and the cards from the Interaction deck are permanently discarded upon use. However, it should be kept in mind that multiple decks might easily multiply the complexity of the game, a scenario which, in the case of this thesis, should be avoided.

Cards could be played relatively free, as in Balatro[39] or Gwent[37], or for a specific cost, usually called mana, like in Hearthstone[36] or Marvel Snap[50]. For Extraction Deck-builder, the use of mana is a better fit because the game prototype will likely include a turn-based combat system, and mana is perfect for limiting the number of actions the player can take per turn. So a system needs to be implemented that determines how much mana the player has each turn, as well as how much mana each card costs.

The most important type of card will be combat cards, as fighting enemies is the main way to get loot in many games. For an engaging combat system, there should be a variety of actions that a player can take during a fight. These could include swinging a sword, holding a shield, casting a spell, drinking potions, and maneuvering around the enemy. Furthermore, the player should have basic stats such as health, armor, and attack strength, and some more complex ones such as accuracy, critical strike chance, and dodge chance. Stat values could be altered by various factors including their initial value, modified by equipment, cast spells, used potions, and other buffs and debuffs.

4.3 Prioritization methods

Developing software in general requires serious planning and task prioritization to ensure that available resources and time are properly used. Doing so increases the likelihood of the project being finished in time and that its core components are developed first.

4.3.1 Moscow prioritization

The tasks are divided into four categories using the Moscow prioritization method. Each category indicates the importance of its tasks. Tasks in the **Must have** category need to be implemented for the game to be considered finished. The **Should have** category contains tasks that are intended to be part of the finished product, but are not essential. In **Could have**, the likelihood that the tasks are included is low, they would add to the experience but are far from necessary. And finally, **Won't have** is for the tasks that are no way to be implemented, at least for this iteration of the project.

The following is a list of the most important features that come or do not come with the prototype, categorized using the Moscow method.

4.3.1.1 Must have

- **Game world**: A set of 2D tiles representing a top-down world where the player and other game objects can exist and perform actions. Contains the extraction locations, too.
- **Cards**: Collectible cards with abilities and other properties such as cost. Different cards can be played differently; some will require a target location or enemy to be picked in the game world, while others will be playable regardless of a target.
- **Set of rules**: Rules that dictate how the game works, when the player draws cards, how the cards interact, and how enemies behave.
- **Playable character**: The character that the player controls in the game world.
- **Enemies**: Non-playable characters with unique behaviors, trying to kill the player, controlled by a simple algorithm.
- **Inventory or Deck**: Some way of storing and organizing cards. An outer and inner inventory are needed.
- **Combat system**: Some sort of logic for the player to damage enemies and for enemies to damage the player.

4.3.1.2 Should have

- **Menu**: The first thing players see when launching the game should be a main menu, where they can enter the game world, adjust some settings, and exit.
- **Sprites**: 2D image files to represent game objects. Makes the game more pretty, but does not add functionality.
- **User interface**: Some basic way to show the stats of the player and the enemies they are currently fighting.
- **Fog of war**: The entire game world should not be shown to the player initially. Instead, it should be revealed as they move around.
- **More interactable objects**: Doors, chests, and other things that the player can interact with. Extraction locations are one such object, and unlike the others, it is a must-have.

4.3.1.3 Could have

- **Sound effects**: Most interactions should have their own sound. Improve immersion, give audio feedback to actions, and make it easier to distinguish between events.
- **Status effects**: Additional game mechanics seen in many games, would include poison, fire, frost, etc. They would be applied to enemies using special

cards, and the player could also receive these effects from enemies. Status effects could interact with each other.

- **Large variety**: An increased variety of enemies, cards, and loot. Would improve replayability, but it is not necessary to showcase the novel game mechanics.
- **Minimap**: A miniature representation of the game world to aid player coordination.
- **Procedurally generated world**: Generating the world on the go is a great way to make it feel large-scale. However, it would most likely require too much effort and would not fit the needs of the thesis.
- **Animations**: Sprites, some text, and the movement of game objects could be animated if time allows.
- **Save**: The genre is heavily based on long-term progression; therefore, the game should save and load the game files. However, implementing it might not be necessary to test the core mechanics of the game during a short testing session.

4.3.1.4 Won't have

- **Performance optimizations**: This would include dynamic loading of the game world, enemies, objects, and items based on the position of the player. It would only be necessary if scalability were a key goal. However, for the prototype, it is negligible. Instead, every game object will be loaded and handled together.
- **Tutorial**: A shorter version of the main gameplay loop with explanations. It would require implementing features that could be told verbally for game testers.
- **Music**: While it would greatly improve immersion, for the game prototype it is unnecessary.
- **Story**: To show how the game mechanics work, a story is not required.
- **Key mapping**: An accessibility setting that would allow the player to freely change the keyboard layout.
- **Multi resolution support**: The game will be only designed for the standard 16:9 aspect ratio, which usually means 1920 by 1080 screen resolution.
- **Cross platform support**: The target platform will be Windows. Other platforms will not be supported at this time.
- **Achievements**: This would be a nice way to motivate players to explore every corner of the game. However, implementing them takes time and should be omitted from the prototype.
- **Localization**: The game will be developed only in English.

4.4 Concept

The initial design question before starting the development of the prototype is what kind of game is the best to test the mix of the extraction and deck-builder genres. Extraction shooter games are usually First Person Shooter games that use 3D graphics, while most card games are 2D with a top-down view. After careful consideration, I chose a top-down approach with a grid-based 2D world to make development as easy as possible. Using 2D removes the need for 3D assets that tend to be more tedious to work with and also makes it easy to implement the game logic.

The open world where each run takes place is a procedurally generated dungeon filled with rooms and corridors connecting them. Each room can contain enemies, chests, and extraction points. To make the game

Although the main progression factor in extraction games comes from the collected and extracted loot, some permanent progression between runs, i.e., meta progression, is also crucial. The exact method chosen for this initially was meant to be the introduction of card types that can be spent on player stat upgrades, such as max health boost, increased hand size, and extended vision, similarly to how players can spend gold in Vampire Survivors to unlock permanent bonuses. However, this type of progression does not fit the deck-builder theme of the game, i.e., spendable cards are really just a currency in disguise. Instead, the chosen meta-progression method became card upgrades: Three identical cards can be combined into a more powerful variant. Upgraded cards can also be upgraded further, up to a set limit.

To prevent players from losing everything when they die during a run, Escape from Tarkov [41] uses two mechanics: safe pouches whose contents can be recovered, and insurance, which can be applied to any item in the player's inventory for a price. I will use the latter approach, but instead of putting a price tag on this feature, i.e., I want to prevent introducing a currency just for this purpose, I want to make insured cards extremely rare. Every card dropped from slain enemies and opened chests will have a small chance of being insured. Although this technically makes it possible for the player eventually to enter a run with only protected cards in their deck, the cards collected during the run can still be lost from the inventory.

For the game to have an ending, there will be a final boss that the player must defeat to finish the game. Exactly one boss will be generated inside each dungeon. Although this makes it possible to encounter it on the first run, it will be so powerful that the player has virtually no chance of defeating it early on.

5

Implementation

5.1 Used software

GitHub Projects[51] was used as a Kanban board to keep track of the tasks defined in Section 4.3.1. The development was not split into multiple sprints, but it was rather following the waterfall method with the addition of continuous adaptation to changing requirements.

The latest version of Godot .NET, which at the time of starting the project was 4.3, was chosen as the game engine, so the game was written in C#. Rider, an IDE (Integrated Development Environment) made by JetBrains exclusively for .NET, was used as an external code editor.

Three different software products were used to create the graphic assets. The most frequently used was Procreate running on an iPad Pro 2020, which was perfect for pixelart as it supports importing and creating custom pixel brushes. On desktop, GIMP and Affinity Photo were used simultaneously for more complex tasks such as editing tile maps.

5.2 General process

5.2.1 Tile map layers

One of the fundamental parts of any game is the world in which the game objects exist. In a 2D game, especially with a top-down view, this world is often abstracted into a grid of rectangular tiles. To make it easier to implement tile-based games, Godot comes with a built-in `TileMapLayer` node that displays a grid of sprites defined in a common `TileSet` resource. The `TileSet` contains a list of source texture files that include tile sprites. It also comes with definitions for each tile, including its texture region within the source texture, its collision shape, navigation properties, and other metadata. Additionally, each tile can be accessed using its atlas coordinates, a 2D vector representing the tile's position within a source texture. When placing tiles programmatically using the `TileMapLayer.SetCell()` method, tiles are referenced using their atlas coordinates and source ID, which tells which source texture they come from.

The prototype uses three layers of `TileMapLayer` nodes: one for the walls, another

one for the ground decorations, and finally the last one is for the fog of war effect. Although technically the first two layers could be combined into a single one, since a ground decoration and a wall sprite are never displayed at the same position, but as an optimization step, separating them seemed more logical. This is because `TileMapLayer` nodes can be queried to check whether a specific tile cell is occupied, allowing for an elegant way of detecting walls without using additional resources. Additionally, this is a future proof solution in case later versions allow walls and ground decor to overlap.

Although tile map layers were perfect for quickly designing test levels within the editor, they did have some limitations. For example, I originally used a `TileMapLayer` to display both enemies and objects, but I also wanted to animate their movement by interpolating between their old and new position; however, tile map layers can display sprites exactly at grid cells, not between, therefore it was impossible to achieve the effect of enemies moving between tiles. So, instead, regular `Sprite2D` nodes had to be used for each enemy that can be positioned freely on the screen.

Before the procedural generation was implemented, a method was used that populated the game world with instances of the `Enemy` class, based on the enemy layer's cell data: wherever a certain atlas coordinate was used for a tile, the method would place an enemy of the corresponding type to that tile's world-space position. The objects and the player were generated using a similar method. This allowed for creating levels manually inside the editor, which was convenient for testing but was later replaced by the automated dungeon generator.

5.2.2 Pathfinding

For an enemy to be able to get closer to the player, it would have to know in which direction to go, how to avoid walls, objects, and other enemies. Pathfinding algorithms, such as the A* (A-star) algorithm, overcome this problem. Given a graph consisting of a list of nodes (tiles) and edges connecting them (between neighboring tiles), A* returns the shortest path between two nodes. Godot has an `AStarGrid2D` class with a built-in implementation of the algorithm. The parameters of the grid on which it operates, such as its region, cell size, and offset, can be customized using its fields and methods, then calling the `GetPointPath()` method with a starting A and ending B node returns a list of nodes that must be traversed to get from A to B. With the `SetPointSolid()` method, a cell can be set to solid (uncrossable) or not solid, which can be used for walls. A* is an expensive algorithm, so it must be used sparsely and preferably on a small grid. A possible optimization would be to use spatial partitioning to reduce the grid size, but that was outside the scope of this project. Instead, there is a fixed map size of 99x99 cells.

Enemy logic heavily relies on pathfinding, as they first try to move towards the player until they are in range for an attack. However, each time an enemy or the player moves, `AStarGrid2D` needs to be updated with the changed solidity of the cells entities have moved between. Some objects, such as doors, also have varying solidity depending on whether they have been opened or not.

The Ranger is a special ranged enemy with a unique logic that allows it to stay just in range to the player. This means that if the player moves closer to the Ranger, it backs away if possible. This is achieved by finding all the cells around the player that are as far away from them as the Ranger's attack range using Euclidean distance to reduce the computational complexity, finding the closest one of these cells to the Ranger, again, using Euclidean distance, and then pathfinding towards that one using A*.

5.2.3 Packed scenes

Sometimes, games need multiple instances of the same game object, such as bullets, cards, and enemies. For example, in a card game, the player's hand might consist of several cards, each displaying unique text, sprites, and values. Creating each card manually would be inefficient and error-prone.

Godot addresses this problem using `PackedScene` resources. A packed scene encapsulates a separate node tree that can be instantiated at run-time from script, adding it to the current scene. This allows the developer to design a single card scene with placeholder content and then dynamically create multiple instances of it, each with different data.

Using packed scenes is straightforward; they are first loaded as a resource using the `Load()` method of the `ResourceLoader` class; then later they can be instantiated to create one object that can be inserted into the node tree:

```
var scene = ResourceLoader.Load<PackedScene>("res://card.tscn");
var card = scene.Instantiate<CardView>();
AddChild(card);
```

The prototype utilizes packed scenes mainly for cards, providing a name and description label, as well as textures for the card art and other indicators in the corners as seen in Figure 5.1. A script is attached to the root node with customizable parameters and an overridden `Ready()` method that, based on the parameters, sets up the node subtree, hiding certain nodes if necessary, dynamically generating new ones, and assigning resources to textures.

Although enemies do not use packed scenes, as it would have required a complete rework of the game's structure, as most game objects, including Enemies, Interactables and the Player, inherit from a base `TileAlignedGameObject` class, which made it complicated to replace enemies with packed scenes. Instead, they are implemented using a script attached to a single `AnimatedSprite2D` node; however, in their `Ready()` method, they add an instance of a `StatusBar` packed scene as a child to themselves. The status bar contains the health and shield bars, plus additional damage, health points, and armour indicators.

5.2.4 Turn-based combat

Initially, the game was intended to be turn-based and would implement a simplified version of the combat system that Baldur's Gate 3[52] uses. This would have meant

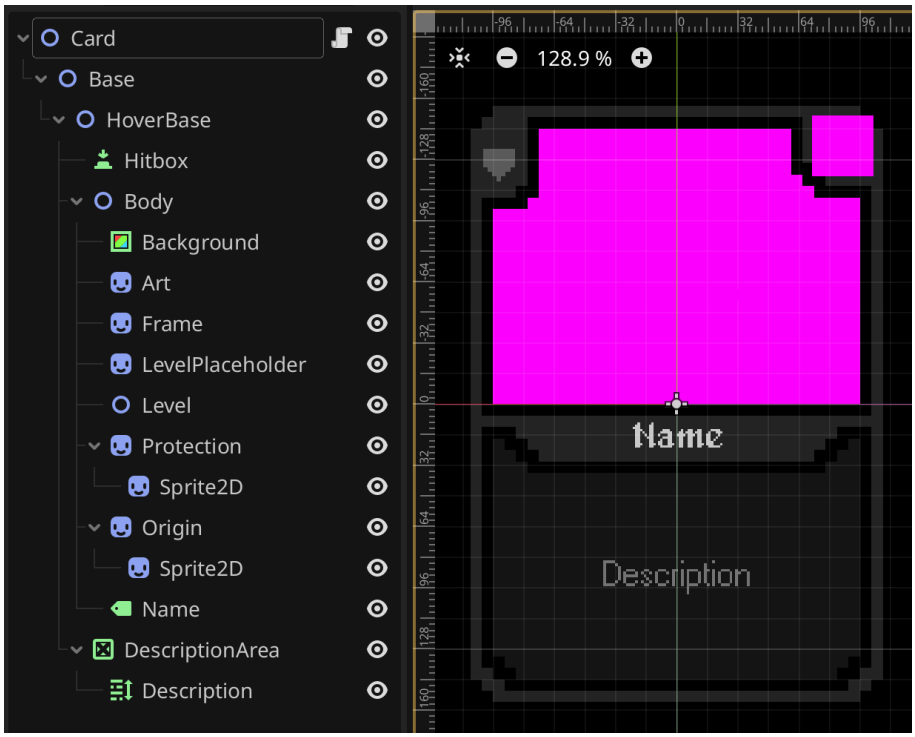

Figure 5.1: The packed scene to be used for cards

that enemies would enter a combat state whenever they detect the presence of the player by either noticing them or receiving damage or a status effect from them. Then, based on a defined turn order and each character's own energy points, the player and enemies would take turns in which they can move and use spells, until their energy is depleted. At the beginning of each turn, energy would be replenished.

An initial version of this system was successfully implemented in an early version of the prototype; however, enemy turns occurred in an instant, without a clear visual representation of each individual action, which made it extremely hard to understand what actually happened between the player's two turns. To make enemy turns visually appealing and easy-to-follow, I wanted to add small delays between each action they take, for example, they would wait for half a second before every step they take. However, in order not to freeze the whole game while enemy turns last, their animations needed to run on separate threads using asynchronous programming, which added so much unnecessary complexity to the logic of the game, resulting in numerous bugs that were hard to debug due to their multithreaded nature.

Ultimately, for the sake of simplicity, I decided to remove the turn-based combat system in its former state and proceed with a much simpler approach, inspired by the Crypt of the Necrodancer[53], a 2D top-down rhythm dungeon crawler. Practically, turns got reduced down to a bare minimum: Each player action, such as movement, interaction, playing, or discarding a card, is considered a whole turn, and thus performing one results in all enemies taking a turn as well. To match the player's reduced flexibility in their turn, the complexity of the enemy logic was also minimized

in the same fashion. Although enemies previously had an energy bar that allowed them to take a few steps on their turn, with the new system in place, most of them only step one every other turn.

5.2.5 User interface

Figure 5.2: The in-game user interface

The game has a simple user interface during the runs, without much text on it. It has three parts:

- The discard pile, located in the bottom left corner of the screen, where a face-down card is shown with a random orientation for each discarded card. On top of the pile, a number shows the exact number of cards.
- The hand in the bottom middle with cards face up, organized in a fan shape, just as someone is holding the cards in their hand.
- The deck in the bottom right corner, which is also a set of face-down cards, but with defined orientations, making it resemble a squared-up pile of cards.

5.2.6 Controls

To reduce the scope, the game was designed solely for keyboard and mouse. The movement of the character is controlled with the arrow or the W, A, S, and D keys, as this is standard in any 2D game. For the cards, dragging them with a mouse feels more natural than just selecting one with the arrow keys and then playing it on a key press. Some cards also require the selection of a target, which is also easier to do with a mouse by hovering over and then clicking the target tile, so for these actions there were no alternative controls implemented. The cards can be inspected

using a method inspired by Hearthstone[36]: When the player hovers one card in the hand, it slides upward to indicate that it is being hovered as seen in Figure 5.2, and to make itself more visible. After the player starts dragging one card, they can end the drag over two areas: the play area covers the upper two-thirds of the screen, and the discard area is located in the bottom left corner.

5.2.6.1 Menus

The game has a pause menu that can be opened by pressing the Escape key. It has four menu items, one for resuming the game, one for quitting it and returning to the desktop, one for opening the help menu, and the last one, which can only be clicked during a run, is for immediately returning to home leaving unprotected cards behind. The help menu includes an explanation of the most important rules of the game and a list of useful tips. Before releasing the game for testing, an additional label was added to the main menu that encourages the tester to open this help menu for further information.

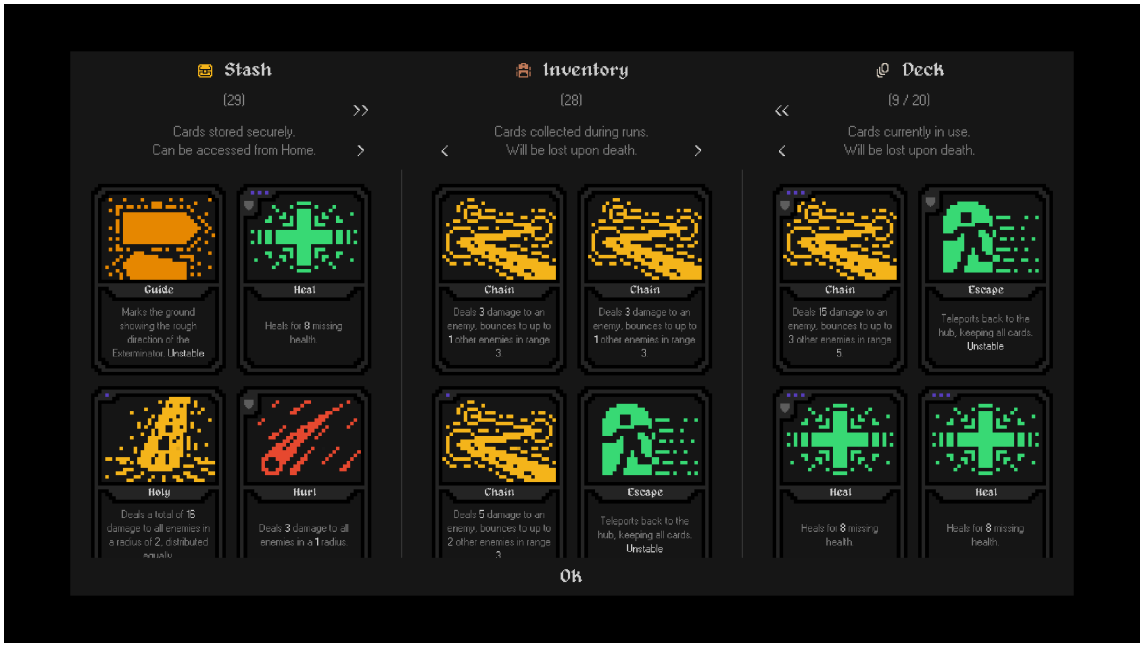

Figure 5.3: The user interface of the inventory menu

In the inventory menu, shown in Figure 5.3, players are presented with panels containing scrollable sorted lists of cards, one container for cards in the stash, inventory, and deck each. They can move cards between these containers using the mouse, or the arrow buttons located on the top of the panel that allows moving all cards from a container to another one at once.

The workbench, which can be seen in Figure 5.4, also shows a scrollable list of cards on the left side, but this list is a combination of all the cards the player possesses, with each card having an indicator in its upper right corner showing its original location. On the right side, there is one output slot and three input slots in which the player can drag the cards to upgrade them. In this menu, cards can also be

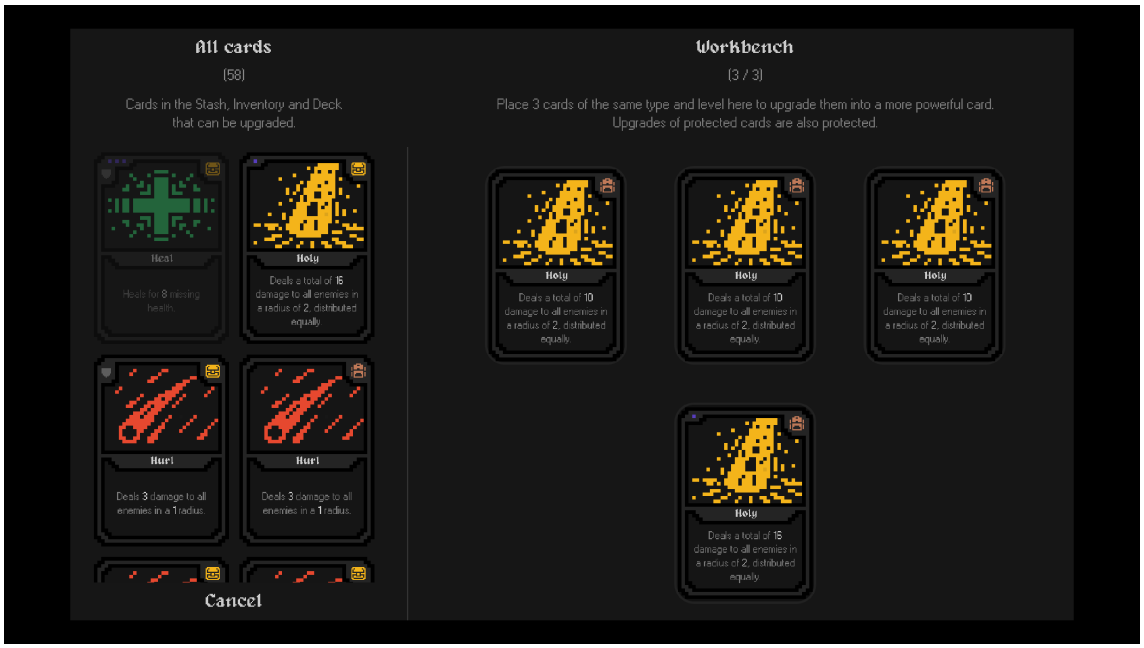

Figure 5.4: The user interface of workbench menu

clicked to instantly put them into the first available slot or to put them back into the list, making the upgrade process drastically faster.

5.2.6.2 Interaction

The game has several objects with which the player can interact. In early versions, for the interaction to happen, the player had to press a specific key, and then the game checked if the player was standing directly next to any interactable object. If so, then the first of these objects is interacted with. Since the order in which the objects are interacted was not defined, this control method was not straightforward, so it had to be improved.

A method called *nudging* was implemented, which is triggered when the player tries to move into an occupied place, such as a wall or a tile with an enemy or interactable on it. Nudging, just as any other action, also takes one turn, so bumping into walls can even be used to skip turns without moving, allowing enemies to get closer to the player. Nudging an interactable triggers an interaction, and since nudging can only target a single tile, the player has total control over which object they want to interact with.

However, some special objects cannot be interacted with using nudge, as chests, for example, require a key to be opened. Using a *Golden Key* card on a chest instantly opens it, requiring no nudge at all.

5.2.6.3 Continuous movement

Since the dungeon can have long straight corridors, navigating them requires repeatedly pressing the same arrow key many times, which quickly becomes tiring. An

alternative movement method was added that allows the player to just hold down an arrow key to keep stepping in that direction at regular intervals. The length of these intervals gradually decreases the longer the key is held down, allowing faster movement while still keeping the turn-based pacing of the game.

5.3 Cards

The most important part of the prototype was to develop a card system with a set of rules. Cards are the only resource in the game, they can be collected during runs and due to the extraction genre, they can also be lost when the player dies, if left unprotected.

In the final design, cards were separated into three containers: the deck contains cards that are in use during a run, the inventory is where the picked up cards go, and the stash is the safe storage where cards remain even after the player dies. Generally, cards cannot be transferred between the three containers, except at home or near a bonfire.

The cards in the deck are divided into three groups, similarly to board games that utilize playing cards.

1. **Hand**: Cards with their face up, that the player is currently holding and is able to play or discard.
2. **Draw pile**: A stack of cards face-down from which the player draws after playing a card from the hand.
3. **Discard pile**: Also a stack of cards facing down, where played or discarded cards go.

There is a maximum hand size of 4 and a deck size of 20, and there is no limit on the discard pile. However, the deck size initially was set to 30, which during internal tests felt too much, so it was reduced.

5.3.1 Protection

As mentioned in Section 4.4, cards can be protected to prevent such edge cases when the player loses all their cards, and thus can no longer do anything in the game. To assist in that and to give the player an opportunity to learn by failing, the starting cards are all protected.

Protection is indicated with a shield symbol in the upper left corner of a protected card. Protection can be carried over to upgraded cards if at least one of the used cards was protected, allowing the player to safely upgrade their starting deck.

5.3.2 Single-use cards

Some cards are considered *utility* cards, most of which can be used once, and then they are lost forever. Such cards are marked with the keyword *Unstable* in their

descriptions. These cards, when played, do not go to the discard pile, even when protected; instead, they are lost forever. This feature prevents the exploitation of some powerful cards, such as golden keys, or the later discussed Escape card.

5.3.3 Types

The game in its final state supports four types of cards:

1. Targets a single enemy
2. Targets a single interactable object
3. Targets a location
4. Targets the player

Except for the cards that target the player, every card is played by first selecting a target tile. Then, based on the tile's content and the card's type, the game determines whether it is a valid target, and then executes the card's ability, or else it cancels the card play.

The game started off with a minimal set of cards: one that heals the player by restoring an amount of missing health points (Heal, type 4), one that damages a single enemy (Smite, type 1), and one that deals damage to all enemies over an area defined by a range (Hurl, type 2). Over time, additional cards were added to enhance the experience. Some new cards were introduced simply to improve variety, like Holy that damages enemies in an area, similarly to Hurl, but divides the damage equally between all the enemies hit, or Chain that deals damage to a single enemy, like Smite, but then it can jump to another target in range a fixed number of times. Other cards, like Rest and Shuffle, were added for balancing reasons.

5.3.3.1 Rest

Rest is a card that shuffles the discard pile into the deck, except for already discarded Rest cards. This allows the player to last significantly longer between bonfire uses or extractions. The introduction of this card was also the main reason the deck size limit was reduced, as putting 15 Rest cards into the deck would allow the player to use the other 15 cards 15 times each, which results in 15 x 15 = 225 card uses in total before needing a bonfire. With a size limit of 20, this mechanic can still be exploited, but the total card uses can only reach a maximum of 100. Additionally, Rest is one of the rarest cards in the game, found only sometimes in chests that require golden key cards, which is also hard to find. So, it is unlikely that the player will ever have 10 Rest cards in their deck.

5.3.3.2 Shuffle

Perhaps the most controversial card, based on feedback from internal playtests, as it sometimes felt useless to the testers. It is similar to Rest, except that this time the hand gets shuffled into the deck, and then new cards are drawn. Its intended use is to prevent the player from discarding momentarily undesired cards from the

hand, and instead put them away for later use. However, a valid criticism is that the current hand limit, which is four, is too small for this card to play a meaningful role in card management. If the hand capacity were to increase over time, as originally planned, Shuffle would receive more love.

5.3.3.3 Guide

When the final enemy, the Exterminator, was introduced, its location was entirely random, which turned out to be extremely frustrating when the player was actively trying to finish the game by defeating the boss. To resolve this issue, I added an additional, rare utility card called Guide that would mark the ground beneath the player, creating an arrow that points in the approximate direction of the boss, drastically reducing the time spent looking for it. Since this card only becomes useful in the late game, its low drop rate is acceptable early on, as players will likely have acquired at least one by the time they are strong enough to face the boss.

5.3.3.4 Teleport

As the name suggests, this card allows the player to instantly jump to an unoccupied target location. It can be used to bypass closed doors or escape from enemies.

5.3.3.5 Shield

A more powerful alternative to Heal is Shield, which adds extra health points on top of the player's maximum allowed health. When the player receives damage, the shield points are depleted before the health. Shield points have their own bar below the health bar.

5.3.3.6 Escape

The most rare card in the game, Escape allows the player to return home without a ladder, from anywhere. It was only added to have a card that truly feels like a legendary class loot. To prevent it from being too powerful though, it is unstable, so the player cannot reuse it for every run once they find one.

5.3.4 Upgrades

In the original plans, collecting cards was the only meta-progression to be included in the prototype. Getting stronger would have been achieved by adding more and more cards to the deck until it is at full capacity, thus the player reaching their full potential. This quickly turned out not to be enough. Getting more and more of the same cards would just mean that the player can repeat the same tasks more during the run without actually feeling stronger.

Card upgrades were implemented as a solution. To upgrade one, three identical cards with the same type and upgrade level are required at the workbench, a new object that can be found at home. Each card got a maximum upgrade level with predefined values, such as damage, range, healing amount, depending on level. However, some

cards, such as keys and other utility cards, cannot be upgraded as they could not make any significant improvements.

5.4 Procedural generation

Initially, the game world was meant to be hand-made and quite small. However, I quickly realized that an extraction game needs a large, open map that allows exploration and long run times with sparse extraction points; otherwise, the game might become too easy and repetitive. To accomplish this, hand-crafting the entire map would have been a waste of time, and with fixed extraction point locations, the players could memorize the map and get in and out effortlessly. Instead, I decided to use a procedural dungeon generation algorithm following the method demonstrated by Bob Nystrom[54], extended with the placement of decorations, enemies, and objects. The algorithm consists of six stages:

1. First, a number of randomly sized rectangular rooms are placed on the map. If a room overlaps with an already placed one, the new one is discarded. This process is repeated until the predetermined maximum number of attempts is reached.

2. After the rooms are placed, a random maze generation algorithm is run that carves paths through all the areas that were left empty. As a result, the entire map is now filled with rooms or corridors.

3. Then hallways are carved out of the walls between the rooms and the maze to make the whole map a single, connected region. Some hallways are generated with a locked door that requires a key to open, somewhat limiting the player's freedom.

4. Dead ends are removed from the maze to improve exploration. This way every corridor leads to another room and the player will not feel frustrated or misled by paths that go nowhere.

5. The ground and the walls are decorated based on their surroundings: the adequate wall sprite is selected from a tile map based on its eight neighboring tiles, and the ground is populated with foliage based on its position inside a room or corridor: denser foliage grows along the wall and inside corners.

6. Finally, rooms are filled with extraction points, bonfires, chests, and enemies based on their size. One exit and one bonfire are always guaranteed, as there are at least two rooms generated exclusively for them. Similarly, a room is reserved for the player to spawn in. With the introduction of the Exterminator, the algorithm was extended with another signature room that is always reserved for the boss.

In later stages of the development, randomized loot generation, along with randomized enemy levels and object density - according to the selected level of difficulty - was also added to the algorithm. With this update, different types of enemies started to drop different loot, defined by a list of probabilities. Additionally, with a

higher level of difficulty came more and tougher enemies that yield more loot when defeated; however, extraction points and bonfires became more rare.

5.4.1 Pretty walls

In a tile-based game, walls are usually stored as a two-dimensional array of Boolean variables, where each true value means that the player cannot pass through the corresponding tile. There are multiple ways to illustrate this towards the player, the simplest and probably most boring solution being to use squares of uniform color for each wall tile. Alternatively, walls can be visualized by continuous lines along the connected edges of the wall tiles. However, a more advanced and elegant solution is to create a tile map containing all possible wall shapes and then use the appropriate sprite for each wall tile.

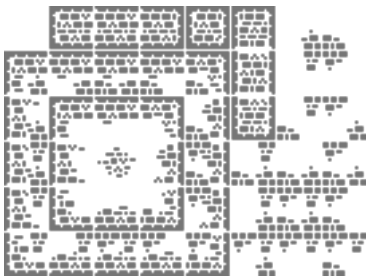

Figure 5.5: All possible wall textures used in the prototype

I decided to use a style that, in addition to visualizing the edge, also includes unique details inside the wall, depending on orientation, shape, and width. To achieve this style, there are exactly 48 possible wall shapes and orientations to choose from, shown in Figure 5.5.

To retrieve the appropriate sprite from the tile map, I first represented a tile's neighboring eight tiles as a bitmask, the greatest bit storing the value for the top-left neighbor, and following a reading order to store the others, the bottom-right neighbor being the last. Then I create a dictionary object, containing all the possible 256 in total bitmasks as keys, and a two-dimensional vector representing their corresponding sprites' position inside the tile set, also called atlas coordinates. To make the process of filling up the dictionary slightly easier, I created patterns, similar to a regular expression or RegEx, consisting of eight nullable booleans, describing the requirements for the bitmasks that correspond to the same sprite in the following way:

1. If a bit must be 1 (wall), its value in the pattern is true
2. If a bit must be 0 (empty), its value in the pattern is false
3. If it does not matter whether a bit is 1 or 0, its value in the pattern is null

Then the patterns and the atlas coordinates of the desired sprite are passed to a method that generates all possible bitmasks that match the pattern and adds them to the dictionary with the bitmask as the key and the provided coordinate pair as the value.

After the dungeon generation is complete, an additional pass is performed that is responsible for filling the `TileMapLayer` object with the appropriate wall tiles, using the bitmask dictionary.

5.4.2 Extraction points

A defining mechanic of extraction games is the use of extraction points with which players can leave the battlefield and return to a safe zone where they can manage the collected loot and have access to meta-progression systems. In the prototype, ladders were used for such a purpose, which were placed randomly in smaller rooms, specifically in those sized 3 by 3 or 3 by 5 tiles. If a room was chosen to contain a ladder, enemies were not allowed to be generated inside it.

To exit, players need to approach a ladder and then bump into it by trying to move onto its occupied tile. As a safety measure, I added a popup text that asks if the player really wants to exit, and to confirm, they are required an extra bump, so that players don't accidentally leave the dungeon.

To exit the dungeon, players must approach and nudge a ladder, see Section 5.2.6.2 for more details. As a safeguard against accidental exits, the first interaction only triggers a pop-up text label informing the player that they are about to leave the dungeon. To confirm, the player must bump into the ladder a second time, ensuring they genuinely intend to leave the dungeon.

5.4.3 Bonfires

Bonfires started as a purely aesthetic asset as a reference to the Dark Souls[55] series and did not play a role in the gameplay. However, quickly after the development had started, I decided to turn bonfires into an object with which the player can interact by nudging it to heal to maximum health.

After implementing a few cards and the game's core rules, it became clear that players needed a way to reset their deck by shuffling the discard pile back into it. This would allow for longer runs and potentially justify a lower maximum deck size. The initial solution was to let players trigger a reload manually, similar to reloading a weapon in shooter games, by pressing a button. Like other actions, this took one turn to perform. However, this approach quickly proved too powerful, as it placed no meaningful constraints on how often the player could use it. For instance, if a player wanted to draw a specific card, they could cycle through the entire deck by repeatedly discarding until the desired card appeared, then reload. Although this cost one action per discard, doing so outside combat had no real downside, making the mechanic easily abusable.

To limit the usage of reload, the most logical solution appeared to be to tie its use to bonfires, placing them similarly to how ladders were already distributed throughout the map. In addition to reloading the deck, bonfires were also a perfect mechanic to allow the player to move cards between their deck and inventory, so they could gain access to the cards collected during the run without having to return home.

During internal tests, it was common for multiple bonfires, or ladders, as they were placed similarly, to be generated close to each other. Finding such a cluster was a huge advantage, as the player was able to reset their deck multiple times without any need for further exploration. Still, on the other hand, the rest of the map had a reduced amount of bonfires, which resulted in some runs without finding any. A minimum distance criterion was introduced to prevent this: if a newly generated bonfire or ladder was too close to any existing one of the same type, it was discarded. This process - similar to the process of how rooms are placed - was repeated until the maximum number of attempts was reached.

When the difficulty parameter was introduced, the minimum distance between bonfires and ladders became tied to it. On higher difficulties, the distance increased, resulting in fewer generated objects and longer exploration sessions.

5.5 Saving

Since the game is a couple of hours long, saving would prevent the loss of progression in case the tester cannot play the game for hours straight. However, saving should prevent players from escaping a foreseeable deadly situation by simply closing and reopening the game and loading an earlier save. To accomplish this, the save file contained a flag about whether the last run started was formerly finished using a Ladder or Escape card, or by player death. The next time the game is launched, it checks for this flag, and if its value is false, the last run is treated as if it was ended by player death, thus all the unprotected cards are erased from the deck and inventory.

Saving and loading are implemented with Godot's native `Json` class using its `Stringify()` and `Parse()` functions. To save data, first, a `Dictionary` object is instantiated from the `Godot.Collections` package, and it is filled with the content of the player's three card containers (stash, inventory, and deck), statistics, and settings, represented as string-value pairs.

6

Results

6.1 Playtests

6.1.1 Showcases

During the early stages of development, I conducted gameplay presentations, where I sat down and showed live gameplay to others and asked for feedback and ideas. These sessions were especially helpful, guiding me in the right direction regarding design decisions in gameplay. These presentations showed the game in a super early state, so the goal was simply to get an idea of whether the project was heading in the right direction.

6.1.2 Early in-house tests

When the game reached a state in which it was considered playable, having a fully functional gameplay loop, I started conducting in-house tests. I let them play the existing part of the game without any prior explanations and watched how they explored the controls and tasks. Occasionally, I added comments on a certain feature missing when the player was approaching its intended place, i.e., extraction points were already present in the generated map, but were not functional. The first of these tests was using a version of the game with limited card and enemy variation, terrible balancing, and the complete lack of meaningful meta-progression. Since the game was far from its intended state, the true goal of these tests was only to gain more early feedback about whether the game's core concept is functional and can be built upon. Most of the reactions I received were about the lack of variety in the game, which was naturally expected. Suggestions were made to add more content and variety to the dungeon generation method. Still, in general, everyone thought that the combination of the extraction and deck-builder genres was a unique and viable idea.

6.1.3 Advanced in-house tests

After responding to the feedback and finishing all the remaining tasks, a series of more serious tests was started, during which players were presented with a finished, feature-complete game. The test consisted of a complete playthrough of the game, as well as an extensive discussion afterwards, during which I took notes, guided by

an early version of the feedback form used later in the public playtest. Three of these tests were conducted, with a gap of a few days between them that allowed the implementation of improvements and requested features.

During the first test, it became clear that the way the inventory and card upgrade menu was designed made it extremely slow and tedious to upgrade a large number of cards at once. In that version, the player had to first move all cards they were planning to upgrade from the inventory or deck to the stash by dragging them one by one from one container to the other. Furthermore, the cards were not sorted alphabetically or in any other way; their position in the list was determined by the order in which they were added to the container, so the player had to constantly search for the desired cards. At the workbench, they also needed to look through the whole list of cards placed in the stash to find the ones they wanted to upgrade, then drag them one by one to one of the upgrade slots, and then finally drag the upgraded version back to the stash container. Additionally, cards that could not be upgraded were also listed and could be dragged onto the upgrade area; only when all three upgrade slots were filled with cards would the game alert the player that the current card is already at its maximum level. Considering that most cards could be upgraded four times back then, to achieve the maximum upgrade for a single card, a total of $3 + 9 + 27 + 81 = 120$ upgrade cycles were required, which is a ridiculous amount, especially if we take into account the time it takes to go through just a single cycle.

It was also pointed out that the game, especially the randomly generated maps that resulted from the procedural dungeon generation algorithm, felt boring and "repetitive, going through the same map over and over". As soon as the player had met all the enemies and found all the card types, there was nothing new or exciting to discover that would keep the player playing. The best way to solve the lack of this mystery factor would be a complete rework of the map generation method, introducing biomes, i.e., distinct regions of the map, and secrets, i.e., rare rooms and enemies, areas locked behind special doors, etc. However, all of these features would not fit the scope of this thesis.

The second test went much more smoothly, since most of the features requested from the first test were implemented by the time it was conducted, making the game much more enjoyable. All menus featured a sorted list of cards, and the cards could be dragged to the upgrade area from any container, as all of their contained cards were combined into a single list. The cards got an indicator in their top right corner showing which container they originate from, helping the player keep track of what cards they are using for the upgrade. However, regardless of where its components originated, the upgraded card variant was always put in the stash container. A good suggestion to solve this was made during the internal tests: if at least one card originates from the deck, then the upgraded card should go to the deck instead. Similarly with the inventory: if at least one card originates from there and there are no cards from the deck, the upgraded card goes to the inventory; otherwise, the stash. This simple logic "feels the most natural" according to multiple playtesters, so it became the permanent solution. Additionally, even though the upgrade process has been dramatically sped up since the last test and the maximum level of cards

was reduced to three, based on feedback, it still did not meet the expectations and was slow to use. However, making it even faster was no longer a high priority.

Since the first test, a new difficulty system has also been implemented that largely increased the feel of progression. Higher difficulty levels meant fewer extraction points, fewer bonfires, a higher number of enemies, and an even higher yield of loot from all sources.

Another requested feature was that the game should save occasionally. Originally, this feature was categorized as a could have in section 4.3.1, however, after multiple suggestions, it was implemented anyway.

Aside from additional balancing and detecting small bugs, the third and last test served as a confirmation that everything was working as intended and that the game could enter the public playtest state.

6.1.4 Public playtest

After making sure that every part of the game was functional and thoroughly tested, I set up a project page on itch.io that contained a brief description of the game and instructions for the playtesters, created a feedback form using Google Forms, and advertised the game among university groups and friends. A deadline of one week counted from posting was set to conclude the test in time; however, this later was extended due to the surprisingly low number of downloads and form responses.

To make the test more accessible, testers were not required to complete a full playthrough of the game, as it would likely have taken multiple hours. Instead, they were asked to try to play the game as long as they could and then fill out the feedback form. During internal tests, the average time it takes to beat the game was estimated to be between two and three hours, but this result was achieved by continuously guiding the players when they felt stuck or did not understand a rule of the game.

6.1.4.1 Hotfixes

Three days after the original demo version was released, a quality-of-life feature was added to the workbench menu as the 1.1 hotfix patch, which made upgrading cards slightly faster. Instead of dragging cards from the list container over to the upgrade slots, they could simply click on a card to make it instantly jump to the first free slot. Similarly with the upgraded card: clicking on it puts it back into the list. This small enhancement made the upgrade process significantly faster, and thus likely reduced the total time it takes to beat the game, as it was shown during internal tests that card upgrades take up a considerable part of the gameplay loop.

Another correction was made in the 1.2 patch. Previously, the card sorting algorithm did not take into account whether a card was protected or not, and as a result, the order of protected and unprotected cards, which were otherwise identical, was undefined. This issue was caused due to a missing comparison between the protected flag of the card class. After the update, the protected cards were sorted ahead of

others. Additionally, the previous patch was not thoroughly tested due to its quick release, so it shipped with a bug that could be exploited to duplicate cards using the workbench. This issue was also resolved.

Finally, a fix was implemented in the 1.3 version that solved a critical issue introduced also with the 1.1 patch. When the player were to click on a card in the hand during a run without starting to drag it, the game would enter an unresponsive state due to a wrongly set Boolean flag, and would stay that way until a card is properly played or discarded. Due to a direct request from a playtester, this version was also released for MacOS.

6.2 Feedback

During the public playtest, the game reached 36 downloads on itch.io, and 16 testers filled out the feedback form. Furthermore, three testers agreed to play the game supervised by me, allowing the observation of their learning curve and every minor issue they encountered. At the same time, they had the luxury of asking questions when they did not understand something. These three tests provided invaluable information that could not have been collected solely from the feedback form.

6.2.1 General gaming background

The form started with a couple of questions about the testers' general gaming background, including how frequently they play video games and how they consider their skills. More than half of the testers play games on a daily basis; however, 20% play just occasionally.

Most of the respondents indicated that they tend to stick with a small number of games rather than playing a wide variety of titles.

Almost all participants stated that they were familiar with the concept of game genres, some even attempting to articulate their own definition. When later asked which genres they play the most, they were able to provide valid examples.

6.2.2 Rating the prototype

The game received an average rating of 8 out of 10, a relatively high score considering that many features were still missing. The testers generally rated the difficulty of the game between moderate and hard, which aligned with the expectations due to the extraction-based mechanics.

Surprisingly, only six out of the 16 players completed the game by defeating the final boss. When asked why they stopped playing before reaching the end, the most common answer was a lack of available time to test, which is completely acceptable given the game takes more than 2 hours to finish. Others mentioned that the game felt too repetitive and they lost interest. Two testers noted that they were unaware of the main objective of the game until they were directly asked, which highlights

a serious design issue, likely caused by the lack of a tutorial and the fact that the help menu is hidden within the pause menu.

6.2.3 Punishment

The players were also asked how they felt after dying for the first time. Most reported feeling sad or surprised, not expecting the game to be that challenging. However, their responses indicated that they were generally not discouraged from starting over. This is likely due to the fact that the starting deck contains only protected cards, meaning that the first death, before any significant progress is typically made, does not result in major losses.

The majority of respondents agreed that losing cards as a punishment for dying is a fair mechanic. One tester noted that, although they did not feel like they had lost anything valuable, they still wanted to avoid losing more cards in the future. Another tester found the mechanic frustrating, stating that collecting the lost cards had taken a significant amount of effort. A third player suggested adding a zen difficulty mode in which players cannot lose any cards at all.

6.2.4 Meta-progression

In the following section, players were asked about the current meta-progression system. The first question focused on how they felt when they returned to the dungeon with their first upgraded cards. The most common responses described feelings of satisfaction, increased power, and confidence. Although the upgrades made the game noticeably easier, players also reported feeling more cautious, as the potential loss of upgraded cards upon death became more significant.

More than half of the respondents agreed that the current system provides a satisfactory level of progression. However, many suggested expanding it to include permanent upgrades to core stats, such as hand size, deck size, maximum health, starting shield, base damage, or dodge chance, possibly through a skill tree or a level-based progression system. Others expressed a desire to revisit previously generated maps after leaving them, to avoid restarting exploration from scratch with each run.

6.2.5 Cards

The next section focused on card usage, starting with the cards players used most frequently. The overwhelming majority mentioned Smite, due to its high effectiveness on lower difficulties where area damage is rarely useful, as most rooms contain only a single enemy. Additionally, Smite was one of the most common drops, making it easy to upgrade early in the game. Several respondents also noted that Smite felt even more powerful than rarer cards like Chain or Holy, suggesting a balancing issue in the current design.

Other frequently mentioned cards included Hurl, Holy, and Heal. In contrast, the least liked cards were Wooden Key, Chain, Shuffle, and again Heal. Chain was

especially criticized for dealing too little damage despite being rare, even though it has already received a significant buff during internal tests. Wooden Keys were rarely used, as doors did not follow a meaningful generation logic. Several players even reported starting the game in areas locked behind doors, happening because doors were entirely randomly placed, without having any Wooden Keys in their deck, forcing them to use the flee. There were single mentions of the Guide, Shield and Teleport cards as well. However, two testers said they have found all cards equally useful.

The testers were also asked to suggest ideas for new cards that could be added in future updates. This resulted in several specific and creative proposals, listed below.

- A card that instantly kills a single enemy, excluding the final boss.
- A variant of the *Teleport* card that moves the player to a random location anywhere on the map.
- A utility card that, similar to *Guide*, that points towards or reveals the exact location of the nearest bonfire or ladder.
- A close-range attack card with exceptionally high damage with the trade-off of needing to get close to enemies.
- A card that allows cards from the inventory to be moved directly into the hand.
- Cards that apply status effects lasting several turns, such as poison, invisibility, or increased vision range.
- A high-risk-high-reward card that randomly damages either an enemy or the player.

6.2.6 Genres

The players were asked whether they believed that the combination of game mechanics could work well in a full game, and the unanimous response was yes. Although they expressed interest in seeing more variety and a wider selection of mechanics and progression options in a complete version, they agreed that the core foundation is already solid.

The testers were not informed of the intended genre classification of the game, but were asked with which genres they would associate it. The most common responses were dungeon crawler and roguelike, followed by mentions of the survival, deck-builder, and extraction genres.

In another question which asked what existing games this prototype resembles, respondents frequently compared the aesthetics and mechanics of the prototype to titles such as Pixel Dungeon[56], Escape from Tarkov[41], Slay the Spire, Pac-Man, and Hearthstone[36]. Baba Is You[57] was also referenced two times, which actually served as a visual inspiration for the prototype. Some testers also mentioned Minecraft[58], Soul Knight[59], and Loop Hero[60] as games with similar elements.

In conclusion, no single example was provided that fully matched the prototype, suggesting that, based on the knowledge of the testers, the combination of extraction and deck-building genres resulted in an entirely new game type. However, it is possible that such a game exists and the testers simply did not know about it, especially considering that the total number of testers was only 16.

6.2.7 Mechanics

The final mandatory section included two questions about missing or potentially beneficial game mechanics. The first question asked whether players felt that any important mechanics were missing. The responses varied widely. Several testers suggested the inclusion of traps, either randomly generated throughout the dungeon or manually placed by enemies or the player. One player proposed cards that automatically activate under predetermined conditions. Others recommended features such as multiple dungeon floors, a map, or a card that allows the player to zoom out and view a larger portion of the map. The lack of a proper tutorial or guide was also mentioned again. One of the most intriguing suggestions was to allow players to combine any two cards, which could result in unexpected effects.

The second question asked about any additional mechanics that could improve the overall experience. Most of the answers repeated ideas from the previous question, but there were a few new suggestions, such as adding story elements and including more interactive objects in the dungeon.

6.2.8 Other questions

An additional section of the form, which was optional to complete, included several open questions. The average time it took players to defeat the final boss was 155 minutes, almost exactly as expected.

Two questions asked players to identify the prototype's best and worst aspects. For the best features, players highlighted the unique interaction system using nudging, the fast-paced gameplay, the ability to upgrade cards, and the overall satisfaction of using fully upgraded cards.

In contrast, the most common complaints included the difficulty of locating the final boss, the tedious process of upgrading a large number of cards at once, getting trapped between locked doors or cornered by enemies with no means of escape, and the Voidling enemy, which could turn invisible and was confusing to some players.

In general, players responded positively to the game's look and feel, and many highlighted the lobby screen as the best part.

6.2.9 Bugs

Several players encountered bugs during testing, but fortunately, none were game-breaking. The most notable issues are listed below:

- **Locked behind doors**: Sometimes, players were placed in a starting area completely surrounded by closed doors. If they did not have a *Wooden Key* or a *Teleport* card to bypass walls, they could not proceed with the exploration and were forced to flee, losing all unprotected cards.
- **Lagging menus**: In the late game, when players typically accumulate hundreds of cards, the inventory and upgrade menus would experience severe lag whenever a card was moved between containers. This issue was especially annoying while upgrading cards, sometimes breaking the user interface completely and making further upgrades impossible until the game was restarted. The lag is caused by the way the menus are redrawn after each change, which could likely be resolved by optimization.
- **Floating cards**: Some testers reported that when a card was played and then canceled with a right click, it sometimes did not return to the hand but remained floating on the screen at a random position. The issue resolved itself when the player hovered the mouse over the card. The bug is purely visual and relatively small, so it was not prioritized for fixing.
- **Card duplication (fixed in 1.2)**: As described in Section 6.1.4.1, a bug introduced in 1.1 allowed players to still claim an upgraded card from the workbench after removing all cards from the input slots. This issue was quickly reported and resolved within minutes in version 1.2.
- **Unresponsive state (fixed in 1.3)**: Also noted in Section 6.1.4.1, this bug caused the game to become unresponsive until the player either played or discarded a card from hand. Two testers encountered this issue before it was patched, but both were able to find a workaround on their own.

In addition to these bugs, there was some confusion around the *Shuffle* and *Rest* cards, based on both feedback form responses and observations during the playtests. Some players believed that *Shuffle* did not actually draw new cards and considered it useless. Others did not understand the purpose of *Rest*, mistakenly thinking they had run out of cards when they could have used it to shuffle the discard pile back into the deck.

The players were also observed to use bonfires and the *Rest* card inefficiently. The optimal strategy is to use all available cards before resetting the deck, but many players used these mechanics prematurely, while their deck was still nearly full, thus wasting some card usages in the process.

6.3 Summary

Overall, the playtest is considered a success, as almost all the testers who provided feedback had a positive experience with the game and the responses to each question were constructive and meaningful.

Based on accumulated feedback, the project successfully demonstrated the viability of combining the extraction and deck-builder genres, creating an enjoyable game

prototype that can later be expanded into a full-fledged game.

7

Discussion

7.1 Answering the second research question

After the playtests had been concluded and sufficient feedback was collected from the testers, the second research question can now be addressed:

What implications can be derived by implementing a game based on a combination of specific game genres?

Even during the early design phase, the game genres, chosen while answering **RQ1**, dictated several key decisions, as both genres' core mechanics had to be meaningfully integrated into the prototype. In this specific case, combining deck-building and extraction games proved to be a natural fit, as reflected in the overwhelming positive feedback from the playtests. The majority of criticism pointed towards the lack of content, as expected due to the tight time constraints, and not the mechanics, which ultimately indicates that the two genre structures support each other well.

This implementation shows that genre combinations can indeed work. However, it is also important to note that although the deck-builder with extraction mix worked effectively, it cannot be consequently assumed that all genre combinations would result in a similarly coherent experience. For example, blending genres with conflicting gameplay types, such as puzzle games and fast-paced multiplayer shooters, might lead to friction and frustration.

Since only one specific genre combination was implemented and tested, no general claims can be made about genre compatibility as a whole. However, the results indicate that, at least in this case, the chosen genres complemented each other well. This suggests that with careful design, some genre combinations can produce coherent and engaging gameplay experiences. At the same time, it highlights the value of early prototyping and playtesting when exploring hybrid games, especially when working with untested combinations.

An already released demo of an upcoming title called Death Howl[33], similarly to this thesis, explores the unprecedented mix of the deck-building and souls-like genres, setting yet another example of how genre blending can lead to completely new and functional games. Notable similarities include a grid-based open world, card collection, and a turn-based combat system, all contributing towards a similar, strategical gameplay.

7.2 Limitations

This section lists several features that were cut from the final prototype to reduce scope and save development time. Most of these are expected to be implemented in future versions. Although a list of features not planned for the finished prototype was defined in Section 4.3.1, some changes were made during development. A few features originally marked as out of scope were eventually implemented to improve the playtesting experience. These include cross-platform support for both Windows and macOS, basic animations, and a save system.

7.2.1 State machines

A popular approach to designing complex enemy behavior is to use state machines. Most games, such as Elden Ring[61] and Hearthstone[36], use something close to state machines to keep track of the current state of every game object. State machines make it easier to implement different behaviors based on the state in which a game object is currently in. For example, a boss in Elden Ring[61] usually has many repeating attack patterns that, without a state machine, would require tons of if statements, checking for the current player distance, what the previous attack was, or what phase the boss is in. However, with a state machine, different behaviors are well divided and do not rely on each other, as the logic of state transitions is managed separately. This results in more maintainable and scalable code in the long run.

This game could have also used such a system, but time constraints did not allow it to. Additionally, the extra complexity by implementing state machines was not necessary, since the logic of most enemies, except the Ranger and the Exterminator, consists only of these three states:

- Standing still, continuously checking if the player is in view
- Following player while in view
- If close enough, attacking the player

7.2.2 Enemy intel

In the early version of the prototype, cards that target an area or an enemy were not using any indicator as to where their casting range or damaging area were. These numbers were only shown in their description and nowhere else, so some playtesters were rightfully confused when they did not hit the enemy they were aiming at. As a quality of life improvement, colored area and range markers were implemented that are displayed when aiming with a selected card.

A similar feature was planned that would have shown where enemy attacks will land in the enemy's next turn, giving the player a clear warning about what tiles are considered dangerous. This would have also made it easier to understand how certain enemies work, as during the playtests the Ranger was especially pointed out to have confusing logic. However, due to time limitations, this idea was scrapped

from the prototype, as it would also have required a complete rework of enemy behaviors to allow for a way to predict their next attack. Instead, players are left to discover each enemy's attack patterns.

A feature that would have allowed the player to inspect an enemy by clicking on it, gaining a detailed overview of its logic, stats, and drop rates, was also removed from the scope.

7.2.3 Variety

Having a large selection of cards is essential for deck-builder games. Collecting all the different cards can be a great motivator for players, making the game replayable. However, with great car variety comes a huge number of tasks: designing each card's logic, implementing, and then balancing it. To keep the scope of the prototype appropriate, the variety was classified as low priority.

Dungeon crawlers are also known to have numerous enemy types to keep traversing the dungeon exciting. For similar reasons, the enemy variety was also reduced.

The dungeon generator algorithm is also considerably simple, as it only generates rectangular rooms with a tight maze connecting them. There are no special rules for rooms with unique shapes, closed regions, or areas with a different style.

There were plans to extend the algorithm so that it would generate rooms without any entrance, making it only accessible using Teleport. Furthermore, rooms containing bonfires or ladders would have been more likely to be closed behind doors on higher difficulties, making the Wooden Key card viable in the late game.

7.2.4 Tutorial

Having a tutorial was ruled out during the process of writing down the concept for the game. However, back then the exact method used for the test was not yet selected. Later, it turned out that the testing will take place entirely online, and players would need to learn the rules of the game on their own. For that purpose, a short tutorial would have been the best option, but there was not enough time left to start working on a proper introductory level. The only feature resembling a tutorial that was implemented was an indicator that only shows up when the player first starts dragging a card from hand during a run, telling the player where to drag-and-drop a card to play or discard it.

As a replacement, a help panel, accessible from the pause menu, was included that describes the basic rules and concepts of the game with additional helpful tips. Furthermore, a text box was added to the main menu screen with information about the test and how to access the help menu. Most of this information was also included on the game's itch.io page.

7.2.5 Controller support

Such a game would be natural to play with a controller, but its card-game aspect makes it a bit harder to design intuitive controls for a gamepad. So, it was chosen not to support controllers in the prototype, and put this feature away for a later release, as it was much easier to implement mouse and keyboard controls both for testing and production.

Similarly, most controls do not have any alternative other than the default keys, and there is no settings menu to customize key bindings.

7.2.6 Accessibility

As stated in Section 4.4, accessibility was given low priority throughout the project in order to save time and focus on making the genre combination work effectively. However, some accessibility-friendly aspects were inherent.

The game uses visual assets, such as card art, tile maps, symbols, and enemy and object sprites, that are distinguishable based on their shapes. Although this was not explicitly tested, the design may allow colorblind players to fully experience the game. Furthermore, since the game does not rely on audio cues, players with hearing impairments are not at a disadvantage.

With that said, several accessibility features are missing, primarily due to the project's limited scope.

- **Audio**: The game has no sound, making it difficult for visually impaired players to play the game.
- **Tutorial**: There is only a minimal introduction to the controls and mechanics. Players are expected to discover most of the rules on their own, which can be frustrating or inaccessible to some.
- **Control remapping**: The game does not allow players to rebind controls. Users must adapt to the default key mappings, which may not suit everyone and can create unnecessary barriers for players with different input preferences or needs.
- **Text readability**: Several playtesters reported that in-game text appeared pixelated and hard to read. This is a known issue caused by the chosen upscaling method in the project settings, which works well for 2D sprites but significantly reduces text clarity.
- **Localization**: The in-game texts are all written in English, and there is no option to change the language, making the game hard to understand for those who don't speak this language on at least a basic level.
- **Settings**: There is no settings menu, so players cannot customize how the game looks or performs.
- **Performance**: During the public playtest, one tester was unable to start the game due to their computer's low specifications. Since no minimum system

requirements were defined for the prototype, hardware compatibility issues could not be anticipated.

Future versions of the game could address these issues, making the game accessible to a broader audience.

7.2.7 Playtester groups

No particular strategy was used when selecting playtesters; therefore participants with a wide range of gaming preferences were included. As a result, the feedback reflected diverse perspectives, and some testers expressed their dissatisfaction with certain mechanics. For example, several participants did not resonate well with the turn-based combat, stating that they "don't play these types of games." Others felt that the penalty for player death was too punishing, especially those who primarily play more casual or low-stakes games.

It is likely that selecting testers based on their familiarity with the relevant genres would have yielded significantly different feedback, since such players may have been more open to certain design choices and better able to judge the experience as it was meant to be played.

7.2.8 Demographic data of playtesters

Unfortunately, no demographic data were collected during the public playtests, as this information was not considered relevant at that time. However, later during the evaluation of the collected feedback, it would have proven valuable.

However, since the test was mainly advertised in international master-level university groups, the average age of the testers can be estimated to be between 20 and 30 years.

7.3 Ethical considerations

Playtesters were informed on the game's download page prior to the test exactly what kind of game they were about to play and that they were not required to play it in its entirety. However, the estimated play time was stated, which reportedly discouraged some potential testers from participating. It was also noted that participation is entirely voluntary during the test and in the feedback form. In addition, no demographic or personal data was collected and their contribution remained anonymous. In the first question of the feedback form, the participants also had to confirm that they were at least 18 years old to ensure compliance with GDPR and other country-specific laws, as involving minors would have required parental consent, which would have complicated the research process. In retrospect, this may have been unnecessary since no personal data was collected. However, without this confirmation, the participants could not proceed.

The game does not contain any intentional offensive or violent content, aside from the fact that the player is required to defeat enemy creatures. However, a tester

pointed out that the game's dark setting had made them uncomfortable.

Two external assets were used for tilemaps, namely Kenney 1-bit pack[62] and Hexany's Roguelike Tiles[63], both licensed under Creative Commons, making them essentially free to use in any project. Some of these assets were altered to better fit the intended look of the game, while other assets, mainly card art, animations, and menus, were newly created.

Lastly, as mentioned in Section 7.2.6, the game lacks many accessibility features, making it harder to play for some groups of players. Future iterations should focus on solving these issues.

7.4 Future work

Most of the features already mentioned are missing, including accessibility and quality-of-life features, variety, a more complex meta-progression system, and more. There are ways the prototype can be improved in the future. More detailed plans are listed below.

7.4.1 Improved meta-progression

In addition to upgrading cards, another system was planned, which was eventually abandoned during development. With this system, the player could improve their core stats, such as max health, armor, hand size, and others. The reason for abandonment was the intention not to use any currency mechanics, since such a system would require the collection of ability points. Another approach could have been to use unneeded cards as a form of payment for these stat improvements, which is the likely method that will be used in future versions.

A new approach to the current progression system could allow for the combination of any card types at the workbench, making it possible to come up with an almost unlimited variety of cards. However, to make such a feature work and be well-balanced, almost all cards would need to be reworked.

Story elements could also serve as a way of progression: when the player reaches a point in the story, it triggers some effects that later have consequences and alter the gameplay in some way.

7.4.2 Increased variety

During development and testing, many new card ideas appeared, but they did not make it to the demo version due to time constraints. These cards include those with directional effects, such as hitting enemies in a cone-shaped area or shooting an arrow in a straight line, hitting all enemies in its path, or cards applying status effects, such as poison, freeze, and other common effects.

Due to not using state machines, as discussed in Section 7.2.1, implementing enemy logic was extremely cumbersome and time-consuming; therefore, most enemies in

the prototype used the same basic behavior. Only the Ranger, Voidling, and Exterminator enemies had unique actions. With the potential introduction of state machines in future versions, implementing diverse enemy behaviors would become significantly easier and more scalable, allowing for richer combat and more distinct enemy types.

A more complex algorithm would be the most likely solution to improve the variety of generated dungeons, as the current one is limited in layout diversity. Implementing features such as special rooms, areas intentionally separated by doors, or any structure that differs from the general look of the current dungeon would make exploration more exciting and add a mystery factor to the game.

7.4.3 Additional accessibility and quality of life features

The highest priority task currently is to add a tutorial or a better explanation to the game. Most playtesters had a hard time and some even gave up, figuring out the controls and rules of the game, as the current help menu was hard to access and not engaging enough to make the players read it.

In addition, additional settings should be added to support a greater variety of devices, screen resolutions, input methods, and accessibility options.

Improvements need to be made to the in-game user interface, the basic controls, as some players unintentionally discarded cards because the discard and play area were too close to each other, and the use of colors, since some cards with different effects use the same color.

7.5 Project Drift

This section looks back at the start of the master's thesis and explains how this project was born.

7.5.1 The original thesis topic

This project is the result of my second masters thesis topic, which is drastically different from the one on which I originally started working. The story behind my decision to change direction deserves an explanation.

In 2024, I was looking for thesis ideas and approached Box Dragon, a small game development studio based in Gothenburg. They were already working on their first game, As We Descend, which has since been released on Steam in early access. Box Dragon proposed exploring a novel global illumination technique called radiance cascades, based on a research paper draft published about half a year earlier. Originally developed for Path of Exile 2, this technique utilizes a hierarchy of light probes, each with varying angular and linear resolutions, depending on its layer. This enables realistic, real-time global illumination rendering. My long-standing interest in computer graphics, combined with the opportunity to collaborate with a real game studio, motivated me to undertake this project as my thesis.

Box Dragon used a custom version of Unreal Engine 4, so the project was set to be implemented using that. However, I had never worked with Unreal Engine, so the thesis included learning to use the engine itself and implementing a shader prototype as well. Since the technique was almost brand new, very few resources were available at the time. Additionally, just setting up a simple shader in Unreal Engine proved to be quite a challenge on its own, even with the help of Box Dragon. I often had to delve into the engine's complex internal code, which lacked proper documentation, and sometimes took days to decipher. The combination of this struggle and the fact that shader code itself is quite cumbersome to debug resulted in constant errors and bugs that required countless work hours to solve, and I could not be sure that, during the process of fixing them, I would not introduce even worse problems.

This project was originally scheduled to be completed in May 2024, but it was extended by three months until late summer. However, due to the ongoing challenges and frustrations, I became burnt out and found it increasingly difficult to stay motivated. Starting over became a real alternative in my head, as the idea of working on an entirely different topic with familiar tools felt refreshing, promising, and motivating.

So, after playing with the thought for the entire summer, I finally informed Box Dragon about my decision in September.

I still feel guilty for leaving that project, especially since the team at Box Dragon invested so much time helping me overcome the obstacles, and they were always understanding and supportive. Furthermore, successfully implementing a novel rendering technique for a company would have been a prestigious achievement for me. Sadly, I had to accept that moving on was the best choice.

7.5.2 From shaders to genres

The reason for such a significant shift in direction can be attributed to my long-standing interest in developing my own game. In the past, I have initiated the development process on various ideas several times in my free time, but none of them have turned into completed projects. Developing a game as a master's thesis seemed like the perfect opportunity to finally realize one idea.

During one of the meetings with my supervisor, Natasha, she mentioned the idea of a master's thesis focused on combining game mechanics and conducting research to test whether they make a functioning game. I loved the topic, so I decided to pursue it.

8

Conclusion

The main goal of this thesis was to find unexplored combinations of video game genres and to select one such pair to be implemented in a prototype.

The research began with an investigation of existing genres to identify combinations that are commonly used, as well as those that appear to be rarely or never explored. Through manual analysis of a dataset that contained extensive data on Steam[14] tags, several genre pairs were found to be uncommon or completely absent, making them ideal candidates for the second part of the project. As a result, the deck-builder and newly defined extraction genres were selected for further exploration.

In the second phase, a prototype was developed using Godot, incorporating the core mechanics of both selected genres. The gameplay loop revolved around players using a deck of cards while navigating a procedurally generated dungeon, fighting with enemies in a turn-based combat system, and attempting to locate extraction points to secure their collected loot. In addition to card collection, a card upgrading system was implemented to provide an additional layer of meta-progression.

Playtesting provided valuable feedback throughout and after development. Players responded positively to the hybrid gameplay, and most of the criticism focused on the limited content rather than the mechanics themselves. This suggests that the combination of deck-builder and extraction mechanics worked well within the prototype.

Based on this project, genre compatibility is speculated to have a direct impact on how viable a genre combination can be. Since only a single mix was explored, this statement cannot be generalized. However, this work, along with industry examples such as Death Howl[33], demonstrates that certain genre combinations can produce new and functional games that players enjoy.

In conclusion, this thesis shows that there are numerous unexplored genre pairs, some of which look promising for future development. The project provides an example of how such experimentation can be carried out through research and prototyping. Future work could involve systematically testing other combinations to better understand what factors contribute to successful genre blending and how players respond to different types of hybrid games.

Bibliography

[1] S. Malliet and G. De Meyer, "The history of the video game," *Handbook of computer game studies*, pp. 23–45, 2005.

[2] D. Arsenault, "Video game genre, evolution and innovation," *Eludamos: Journal for computer game culture*, vol. 3, no. 2, 2009. DOI: 10.7557/23.6003.

[3] W. Higinbotham, *Tennis for Two*, Brookhaven National Laboratory, Oscilloscope, 1958.

[4] S. Russell, M. Graetz, and W. Wiitanen, *Spacewar!* DEC PDP-1 Computer, 1962.

[5] Atari, Inc., *Pong*, Arcade, 1972.

[6] J. Romero, J. Carmack, *et al.*, *Doom*, PC, 1993.

[7] R. Hunicke, M. LeBlanc, R. Zubek, *et al.*, "Mda: A formal approach to game design and game research," in *Proceedings of the AAAI Workshop on Challenges in Game AI*, San Jose, CA, vol. 4, 2004, p. 1722.

[8] C. Crawford, *The Art of Computer Game Design*. Berkeley, CA: McGraw-Hill Osborne Media, 1984, First game genre definitions.

[9] M. J. Wolf, "Genre and the video game," in *The medium of the video game*, University of Texas Press, 2002, pp. 113–134.

[10] T. H. Apperley, "Genre and game studies: Toward a critical approach to video game genres," *Simulation & gaming*, vol. 37, no. 1, pp. 6–23, 2006.

[11] L. Wittgenstein, *Philosophical Investigations*. Blackwell, 1953.

[12] J. Starosta, P. Kiszka, P. D. Szyszka, S. Starzec, and P. Strojny, "The tangled ways to classify games: A systematic review of how games are classified in psychological research," *PLoS one*, vol. 19, no. 6, e0299819, 2024.

[13] J. H. Lee, N. Karlova, R. I. Clarke, and Thornton, "Facet analysis of video game genres," *IConference 2014 Proceedings*, 2014, Proposes a new framework for improved intellectual access to video games along multiple dimensions.

[14] Valve Corporation, *Steam*, Digital distribution platform for video games, 2003. [Online]. Available: https://store.steampowered.com/.

[15] N. I. G. Dev, *Steam catalog insights*. [Online]. Available: %5Curl%7Bhttps://github.com/NewbieIndieGameDev/steam-insights%7D.

[16] W. Li, "Zipf's law everywhere.," *Glottometrics*, vol. 5, no. 2002, pp. 14–21, 2002.

[17] J. Gregory, *Game engine architecture*. AK Peters/CRC Press, 2018.

[18] Epic Games, Inc., *Unreal engine*, https://www.unrealengine.com/.

[19] Unity Technologies, *Unity*, https://unity.com/.

[20] Godot Engine Development Team, *Godot engine*, Game Engine, 2014.

[21] LÖVE Development Team, *LÖVE*, Game Framework, 2008.
[22] A. Ljungdahl, *Individual game development with open-source software - a case study with guidelines for programmers and game designers*, 2020.
[23] Microsoft Corporation, *Microsoft Visual Studio*, Software, 1997.
[24] Microsoft Corporation, *Visual Studio Code*, Software, 2015.
[25] JetBrains s.r.o., *JetBrains*, Software Company, 2000.
[26] The GIMP Development Team, *GIMP*, Software, 1996.
[27] Figma, Inc., *Figma*, Software, 2016.
[28] Blender Foundation, *Blender*, Software, 1994.
[29] Amanita Design, *Samorost 3*, PC, Mobile, 2016.
[30] P. Lankoski, S. Björk, *et al.*, "Game research methods: An overview," pp. 93–116, 2015.
[31] P. Sweetser, D. Johnson, P. Wyeth, A. Anwar, Y. Meng, and A. Ozdowska, *Gameflow in different game genres and platforms*, eng, New York, 2017.
[32] P. Lankoski, S. Björk, *et al.*, "Game research methods: An overview," pp. 231–250, 2015.
[33] The Outer Zone and 11 bit studios, *Death Howl*, In development, 2025.
[34] J. H. Staffan Björk, *Gameplay design patterns*, http://virt10.itu.chalmers.se, Acessed: 2024-10-22.
[35] R. Garfield, *Magic: The Gathering*, 1993.
[36] Blizzard Entertainment, *Hearthstone*, PC, Mobile, 2014.
[37] CD Projekt Red, *Gwent: The Witcher Card Game*, PC, Mobile, 2018.
[38] D. X. Vaccarino, *Dominion*, 2008.
[39] LocalThunk, *Balatro*, PC, PlayStation, Xbox, Switch, 2024.
[40] Massive Entertainment, *Tom Clancy's The Division*, 2016.
[41] Battlestate Games, *Escape from Tarkov*, PC, 2017.
[42] Bungie, *Marathon*, PC, In development, 2025.
[43] Embark Studios, *ARC Raiders*, PC, In development, 2025.
[44] Ghost Ship Games, *Deep Rock Galactic*, PC, Xbox, PlayStation, 2018.
[45] Zeekerss, *Lethal Company*, PC, 2023.
[46] Bippinbits, *Dome Keeper*, PC, 2022.
[47] R. Van Der Linden, R. Lopes, and R. Bidarra, "Procedural generation of dungeons," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 1, pp. 78–89, 2013.
[48] Dodge Roll, *Enter the Gungeon*, PC, PlayStation, Xbox, Switch, 2016.
[49] McMillen, Edmund and Himsl, Florian, *The Binding of Isaac: Rebirth*, PC, PlayStation, Xbox, Switch, 2014.
[50] Second Dinner, *Marvel Snap*, PC, Mobile, 2022.
[51] GitHub Inc, *Github*, `https://github.com/`.
[52] Larian Studios, *Baldur's Gate 3*, PC, 2023.
[53] Brace Yourself Games, *Crypt of the NecroDancer*, 2015.
[54] B. Nystrom, *Rooms and mazes: A procedural dungeon generator*, 2014. [Online]. Available: `https://journal.stuffwithstuff.com/2014/12/21/rooms-and-mazes/`.
[55] FromSoftware, *Dark Souls (franchise)*, 20112016.
[56] Watabou, *Pixel Dungeon*, 2012.

[57] A. Teikari, *Baba Is You*, 2019.
[58] M. Persson, *Minecraft*, 2009.
[59] ChillyRoom, *Soul Knight*, 2017.
[60] Four Quarters, *Loop Hero*, 2021.
[61] FromSoftware, *Elden Ring*, 2022.
[62] Kenney, *1-Bit Pack*, `https://kenney-assets.itch.io/1-bit-pack`, Game asset pack, licensed under CC0, 2020.
[63] H. Ives, *Hexany's Roguelike Tiles*, `https://hexany-ives.itch.io/hexanys-roguelike-tiles`, Game asset pack, licensed under CC0, 2023.

A

Feedback form

This chapter lists the questions that have been asked in the feedback form as part of the public playtest. The form had 4 pages; each of the are discussed below separated into their own sections. There were a total of 16 respondents.

A.1 Welcome page

The testers who opened the feedback form were presented by a simple welcome screen that included a brief description of the upcoming questions, information about the form being anonymous, and a confirmation that the tester was at least 18 years old, to ensure that the collection of information was in compliance with legal requirements.

A.2 Second page

The first page of the form contained questions regarding the player's gaming background. As mentioned in Section 7.2, no questions were asked about the player's demographic background. The following is a list of the questions with their exact wording.

- **"How would you rate your general gaming skills?"** (from 1 - rookie to 5 - veteran): The average response was 3.5.
- **"How often do you play video games?"** (multiple choice):
  - **"2+ hours daily"**: 6
  - **"1-2 hours daily"**: 4
  - **"Few times a week"**: 3
  - **"More rarely"**: 3
- **"Do you know what a video game genre is?"** (short text answer): Most respondents just replied "yes" or "kind of", only a few of them tried to actually form a definition. This indicated that most of them at least had an idea about what a game genre is.

- **What video game genres do you enjoy playing the most?** short answer: The main purpose of this question was to confirm that the respondents really knew what a game genre is.

A.3 Third page

This was the main content of the form containing questions about the prototype, the mechanics, and how players felt during the test.

- **"Overall, how would you rate the game?"** (from 1 to 10): The average response was 7.9. which is considered positive.
- **"How would you rate the difficulty of the game?"** (from 1 - Easy to 5 - Dark Souls): The average rating was 3.3, which is around the expected value based on the in-house playtests.
- **"Have you beaten the game by slaying the boss (the Exterminator)?"** (yes or no): Only 6 players (37.5%) have defeated the final boss.
- **If you haven't beaten the game, what caused you to stop playing?** short answer: The most common answer was "lack of time" and that the game was "too long". Others found the game too "repetitive", one tester even got stuck and could not continue the test.

The next questions were themed around the death mechanic, including the permanent loss of loot.

- **"Briefly describe what you felt when you died in the game for the first time."** (short answer): Most of the respondents described feelings of sadness and anger. Some of them were "not surprised", while others were confused and did not fully understand how they died.
- **"How many times did you die in the game?"** (short answer): The average death count was around 6, although two players did not specify an exact amount, and one player entered 36 which seems unnaturally high, since most people wrote numbers between 1 and 10.
- **"Have you beaten the game by slaying the boss How did you feel about the way the game punishes you for dying (losing cards)?"** (multiple choice):
  - **"Way too harsh"**: 1
  - **"Fair"**: 10
  - **"Too forgiving"**: 1
  - **"Other"**: 3 people felt it was frustrating to lose cards as "it was no easy task to collect them". One tester "did not feel they lost anything important, but they did not want to lose more".

These questions are about the card upgrades, the main way meta-progression in the game.

- **"How did you feel after you have upgraded some cards and headed back into the dungeon?"** (short answer): All testers felt satisfied, stronger, and excited. One of them even found the game to be too easy.
- **"Do you feel upgrading cards is enough meta-progression? (meta-progression: permanent progression that persists between runs)"** (multiple choice):
  - **"Yes"**: 10
  - **"Other"**: 5 people suggested a skill-tree-like progression system for increasing the character;s basis stats like maximum hand size or maximum health. One tester also mentioned the possibility for combining any three cards.

The following questions are about the cards themselves.

- **"Which cards did you use more often than others?"** (short answer): Smite and Hurl were mentioned most frequently. There were also 3 mentions of Heal, 2 mentions of Holy and 1 mention of Golden Key.
- **"Was there any card you did not like and rarely used?"** (short answer): Most testers wrote Wooden Key, Chain, Heal and Shuffle. Teleport and Guide were also mentioned.
- **"Is there any card you think should be in the game?"** (sort answer): Many great ideas were listed here, such as status effect cards (stun, poison, invisibility), compass that shows the direction towards nearest bonfire/ladder and a card that instantly kills any non-boss enemy.

Lastly, there were a couple of miscellaneous questions.

- **"Do you think this prototype proves that this combination of game mechanics is functional and could work well in a full game?"** (short answer): All testers replied yes. Some of them criticized the current lack of content and variety.
- **"What video game genre(s) do you think the game belongs to?** (short answer): The most common answer was rogue like, dungeon crawler, and card game. Only 3 respondents mentioned extraction shooters.
- **"Which existing games does this prototype remind you of?"** (sort answer): This question aimed to verify that the prototype is indeed a new type of game, so the tester were not expected to come up with an exact match. There were 3 mentions of Pixel Dungeon, 2 mentions of Slay the Spire, Escape from Tarkov, Pac-Man and Baba is You. 2 testers did not specify.
- **"Did you feel there were any mechanics missing from the game?"** (short answer): The testers mainly suggested the inclusion of a map or min-

imap. The lack of a proper tutorial was also mentioned several times. Improving the map generation with "traps/bombs" also came up.

- **"What other mechanics would make the game better, if any?"** (short answer): A similar question to the previous one, therefore, many aswers overlap. Those that do not are just repetitions of earlier suggestions, such as a skill tree or new cards.

A.4 Fourth page

The last page only contained optional questions as they were not considered crucial for the project, therefore not every respondent answered them.

- **"If you have beaten the game, how many minutes did it take? (exact time is shown on the victory screen)"** (short answer): Only 5 respondents wrote an exact amount of time, and the average was 155 minutes.
- **"What did you love the most in the game?** (short answer): many players found nudging exceptionally satisfying. The upgrading of cards was also highlighted multiple times.
- **"What did you hate the most?"** (short answer): Several testers reported getting stuck due to getting surrounded by monsters. Some players even encountered maps that were generated with a completely locked starting area, rendering the player unable to explore the dungeon and forcing them to flee.
- **"What did you think about the look of the game? (card art, tiles, etc.)"** (short answer): This was an objective question, so naturally the answers varied. Some liked the simple pixelated art style, some found it too minimalist.
- **"Did you encounter any bugs?"** (short answer): Luckily, there were only a few bugs throughout the entire testing period, and none of them were truly game breaking. Getting stuck in the starting area was one of them, which is due to the generating method not checking for such cases. Other bugs included purely visual ones such as cards not returning to the hand. There was one serious bug, however, that allowed the player to essentially duplicate any card, exploiting an issue with the workbench, but this error was patched in an update.
- **"Please add any further remark or critique. Any feedback is welcome!** (short answer): Some extra ideas included adding sound to the game, and "undo feature" which would cancel the player's last action, and making upgrading cards less cumbersome. Overall the feedback was overwhelmingly positive.