The Effects of Levelling System in Video Games

Hu Erzhizhi

BEng (Hons.) Robotics
Honours Dissertation

Supervised by Dr. Adam Sampson

Heriot-Watt University
School of Mathematical and Computer Sciences

March 2025

DECLARATION

I, Hu Erzhizhi, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed:
Hu Erzhizhi

Date:
13th November 2024

ABSTRACT

As one of the most popular genres in video games, role-playing games (RPGs) have been widely embraced by players since their inception. Among their many features, progression mechanics—particularly levelling systems—are highly valued. Due to their well-designed structure and ability to enhance player engagement, levelling systems have been widely adopted in many video games. However, the complexity of these systems varies across different games, with some offering intricate progression paths while others adopt a more straightforward approach. This project employs a questionnaire-based approach to examine whether levelling systems of varying complexity—complex, normal, and simple—can coexist within the same game framework. Additionally, it evaluates the playability of each system and its impact on players' psychological satisfaction. By analyzing these factors, the study aims to explore how different levels of complexity in levelling systems influence both gameplay experience and overall player engagement.

ACKNOWLEDGEMENTS

I'd like to express my heartfelt gratitude to my parents for their unwavering support of my dreams, they may not align with conventional paths.

I also wish to thank my supervisor Adam, for supporting my ideas about game making and providing valuable feedback that greatly improved this project.

Finally, I would like to extend my gratitude to all the participants who contributed to this project. Their involvement and support were invaluable in bringing this work to completion.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

Since the rise of role-playing games (RPGs), the levelling system has become a widely adopted mechanic in many video games, including titles such as Fallout [Bethesda 1997], The Elder Scrolls [Bethesda 1994], and Mass Effect [BioWare 2007]. This system, commonly referred to as the levelling system or progression system [Zagal and Altizer 2014], will be further discussed in detail in Section 2.1. In general, it rewards players with experience points (XP) for completing specific actions, such as finishing a task. Once players accumulate enough XP, they level up [Cook and Williams 2003], leading to enhanced attributes, such as increased hit points (HP) and access to new perks. Simply put, higher levels make the player character stronger.

However, different video games will have different levelling system designs. In this project, we roughly divide them into 3 different versions. Version 1 features a strict levelling system, requiring players to select attributes and other factors, such as aptitude, when creating their characters at the start of the game. Each time players level up, the system prompts them to choose which skill (or attribute, as seen in games like Cyberpunk 2077) to enhance. Additionally, upon reaching certain levels or meeting specific attribute requirements, players earn points that can be used to unlock perks. Games with this system include The Outer Worlds [Obsidian 2019], Fallout: New Vagas [Obsidian 2010] and Cyberpunk 2077 [CD Projekt Red 2020]. Version 2 features a simpler levelling system. At the beginning, players are asked to select their attributes. When players level up, the system does not prompt them to enhance attributes or skills. Instead, players choose from a selection of perks, some of which can improve their attributes and skills. This version includes Baldur's gate 3 [Larian Studios 2023], Disco Elysium [ZA/UM 2019] and Borderlands 3 [Gearbox Software 2019]. Version 3 has the simplest system, literally, a non-stratified system, players can earn XP or money during the game, but they won't be able to levels up, the XP or money they earned can be used to purchase or upgrade some new abilities and gears. Call of Duty's [Treyarch 2008] Zombie mode and Bioshock Infinite [Irrational Games 2013] are included in this version.

Different levelling system designs can have a significant impact on the same game, due to each system bringing its own balance, affecting the game's playability. The more complex levelling system such as version 1 and version 2 we mentioned above can make combat in the game more diverse, offering various types of combat. For instance, players can choose to be a gun specialist by enhancing the skills or perks for shooting, alternatively, they can allocate points to melee skills or perks if they want to become a berserker. Moreover, some systems integrate traditionally non-combat skills into combat, e.g., using "skills like science, persuasion, and hacking actually somewhat valuable in a fight by passively causing enemies to flee or cower"[Stapleton 2019], which adds an extra layer of interest and variety to the gameplay. In comparison, the levelling system in version 3 may seem somewhat dull or lacking in excitement. However, it's not entirely like that, In the early stages, players must carefully decide how to allocate their limited XP or money when upgrading abilities. As they progress, they will "accrue in-game currency to obtain upgrades, including better weapons and stronger

defenses."[Francisco 2023] This creates a strategic balance, where players must prioritize the most crucial upgrades to ensure their progression while managing scarce resources, which also increase the interesting of the gameplay.

Players also have varied tastes—some may enjoy complex systems, while others, may prefer a simpler approach. For the systems in version 1 and version 2, some people feel they are too complex and difficult, which makes people tired, while someone think the system makes difficulty of the game quite easy in the later game since "it is far too easy to have plenty of high-leveled skills that can end up trivializing most encounters." They also think that in some games, the perks players can gain "mostly offer either flat bonuses or slightly better situational bonuses"[Webster 2020], lacking creativity or appeal. However, systems in version 3 might also have the same problem in the difficulty of the game. As players progress, they have unlocked almost every ability and upgraded all their gears, rendering enemies less threatening. Another aspect frequently criticized in version 3 is the need for players to be overly cautious with their limited XP or money when deciding which skills or items to upgrade or purchase. This focus on resource management can detract from the overall enjoyment, as players may feel pressured to optimize every choice rather than exploring the game freely. For more specific and detailed player reviews for each system, we will talk about it in Section 2.2.4.

The aim of this project is to examine the design concepts behind different levelling systems and to analyze how these systems influence the gameplay experience in video games. By comparing the 3 versions mentioned above, we hope to identify the strengths and weaknesses of each system. In this project, we will:

- Design a game with 3 different versions.
- Recruit volunteers to play some of those versions and provide feedback.
- Analysis the feedback.

By doing so, we can explore whether different types of levelling systems can be integrated into a unified game framework and examine how they impact player engagement, character development, and overall game balance. Ultimately, this contributes to a deeper understanding of why levelling systems are widely embraced by players and what makes them such a popular game mechanic.

2 BACKGROUND

This section provides an overview of existing research and understanding of levelling systems in video games, including their mechanics and impact on gameplay. Specific examples from established games will be used to illustrate these systems in practice. Additionally, foundational concepts in general game design will be introduced. Finally, the section outlines the approach to experimental design and analysis, detailing the methods used for evaluating and interpreting results.

2.1 Levelling System

In Playing at the World [Peterson 2012], Peterson discusses the iconic tabletop game Dungeons and Dragons(DnD), highlighting the development of its mechanisms. One such mechanism is the character progression system, or levelling system, where "player characters improve and grow stronger the longer they play."[Zagal and Altizer 2014] The creator of DnD described the system as allowing "adventurers to advance in ability through hard fighting and clever deeds, ultimately becoming forces to be reckoned with". In simple terms, the levelling system makes players' characters become progressively stronger as they continue through the game.

The levelling system can be categorized into different types. Zagal and Altizer [2014] classified them into positive progression, negative progression, and borderline progression. Positive progression, the most common type, enhances the player's character, making them stronger as the game progresses. In contrast, negative progression, the opposite of positive progression, weakens the character or makes them less effective as the game advances. For borderline progression, they described them as "not quite fitting the sense of character progression we have described." These systems involve more ambiguous or situational changes to a character's abilities that do not strictly align with the idea of continuous improvement or decline.

In video games, the levelling system is often "narrowed down to points being accumulated upon successful completion of a quest."[Pav 2020] Players' characters gain experience points (XP) by winning battles, completing tasks, or defeating enemies. In some RPGs, characters can even earn XP through unique actions such as successfully persuading others, picking locks, or hacking devices. As characters accumulate enough XP, they reach a new "level"—a numerical representation of their overall power, which is often linked to various statistics and their class. Each level reflects a meaningful stage in the character's growth, granting improved powers, enhanced stats, or access to new abilities that align with their journey's progress. For example, a novice mage at level 1 is relatively weak, while a high-level wizard embodies the strength and experience of countless battles and quests. As the game progresses, each new level generally requires more XP, creating a cycle of challenge and reward that fuels the character's advancement and deepens player engagement. Additionally, some video games apply a non-stratified levelling system, which operates without traditional "levels." We consider this a variation of levelling systems, as characters can still earn money or XP throughout

the game and use these resources to purchase or upgrade equipment and skills, allowing for character growth and progression outside of fixed levels.

2.2 Levelling System in Existing Games

In this section, we will provide more detailed examples of the three versions of levelling systems mentioned earlier, as implemented in existing games.

2.2.1 Levelling System in The Outer Worlds.

The Outer Worlds, a recent game developed by Obsidian Entertainment, inherits the progression system mechanics of Fallout: New Vegas [Obsidian 2010]. Like its predecessor, The Outer Worlds has a quite complex levelling system. Players are asked to allocate limited points across attributes. Each attribute not only influences the starting values of specific character skills but also affects the character's overall performance in various gameplay aspects. For example, increasing Intelligence will boost the initial values of skills like "Hacking" and "Medicine", while also enhancing the character's critical hit damage. Players are required to choose an aptitude for their character as well, which provides a slight buff to certain skills. For instance, selecting the" Sub Sous Chef" aptitude will give a +1 boost to the character's skill "One-handed melee." However, the more significant impact of aptitudes is seen in the game's dialogue system. Certain aptitudes unlock additional dialogue options, allowing players to use their character's unique traits to influence conversations and unlock new responses.

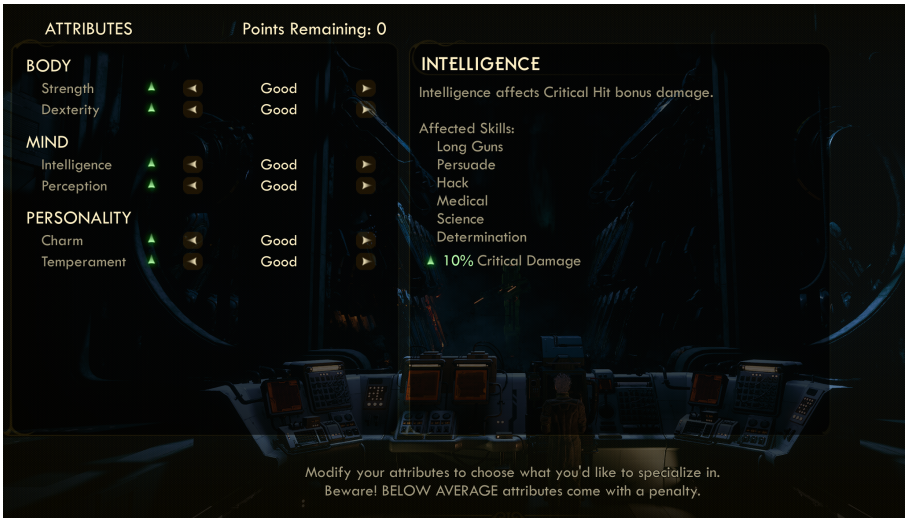


Fig. 1. Attributes in The Outer Worlds

Each time characters levels up in the game, they receive 10 points to allocate toward enhancing various skills. Players can "dump all 10 into the same skill every time (but that's probably not the smartest way to do it)" [Parkin 2020], or they can focus on boosting skills they use

frequently or want to improve. This flexibility allows players to “make up for anything they lost out on during the attributes phase of character creation.” The skills are organized into seven broad categories, each containing specific skills within it. Before any skill within a category reaches level 50, players can add points to the entire category, thereby boosting all skills within that group. For example, spending a point on the "Dialog" category will simultaneously increase "Persuade", "Lie", and "Intimidate". However, once a skill surpasses level 50, players must assign points to it individually to have a further enhancement. Perks are special abilities that players can unlock every few levels, enhancing their performance and making them more effective in various ways. These perks can boost core attributes such as health, speed, and carrying capacity, or amplify the damage dealt with specific weapons. Tier 1 perks are accessible from the start, but unlocking higher-tier perks—such as Tier 2 and Tier 3—requires progression and investment, adding depth and strategy to character development. Equipment in The Outer Worlds also features a levelling system. A higher level means better attribute for the equipment, such as more damage for the weapon. Players can improve their gear in two main ways: by spending money at a workbench to directly upgrade their existing equipment or by picking up higher-level gear dropped by enemies. Notably, the equipment found from defeated enemies is dynamically scaled to match the player’s current level, ensuring that newly acquired gear aligns with the character’s progression.

Fig. 2. Skills and Perks in The Outer Worlds

2.2.2 *Levelling System in Baldur's Gate 3.*

Baldur's Gate 3, game of the year at TGA 2023, inherits the traditional tabletop RPG (TRPG) levelling system, blending elements of classic Dungeons & Dragons mechanics with a digital format. Players are prompted at the start to allocate points to character abilities, similar to attributes in The Outer Worlds, but even more integral to gameplay. The character's abilities determine their success in various actions; for example, attempting to persuade others triggers an ability check [Cook and Williams 2003] on Charisma, and a higher Charisma score makes it easier to pass this check, increasing the likelihood of a successful outcome. Additionally, players can select two skills and one background for their character, each of which provides specific bonuses during ability checks throughout the game.

Fig. 3. Abilities in Baldur's Gate 3

Unlike in The Outer Worlds, each time the character levels up in Baldur's Gate 3, players only need to choose new feats or combat actions (such as spells and actions) rather than directly adding points to the character's ability. Some feats, however, can increase the scores of specific abilities, for instance, the "Durable" feat can boost ability "Constitution," allowing for gradual character improvement without manually assigning ability scores. There is no levelling system

for equipment in Baldur's Gate 3, so players don't need to spend time or effort upgrading their gear.

Fig. 4. Feats in Baldur's Gate 3

2.2.3 Levelling system in Bioshock Infinite.

The levelling system in Bioshock Infinite is what we commonly refer to as a non-stratified system—one where characters improve in a piecemeal fashion rather than progressing through discrete, uniform steps such as levels or ranks. Compared to traditional levelling systems, the progression elements in Bioshock Infinite are significantly streamlined. Players are no longer required to allocate points at the start of the game, nor do they level up in the conventional sense. Instead, character enhancement is achieved with vending machines, where players can improve their character's abilities in various ways. For instance, they can purchase new special abilities or upgrade existing ones by reducing cooldown times or lowering resource consumption. In addition to enhancing their abilities, players can improve their weapons through various modifications. Beyond these upgrades, vending machines offer gear with unique bonuses, such as a hat that allows players to summon a legion of the undead. These enhancements come with no level restrictions; if players have sufficient in-game currency and access to a nearby vending machine, they can be unlocked and applied at any time.

Fig. 5. Vending machines in Bioshock Infinite

Although weapons can be upgraded at vending machines, this process is fundamentally distinct from a traditional levelling system. It does not simply enhance the weapon's overall tier or base attributes; rather, it focuses on specific modifications. For example, upgrades may increase the weapon's firepower, improve accuracy, reduce reload time, or expand ammunition capacity.

2.2.4 Reviews from players.

The different levelling systems in the games above have garnered mixed feedback from players. To gather insights on this feedback, we reviewed discussions on various forums, such as Reddit [Reddit 2005], and compiled a table summarizing different opinions expressed by players. For a comprehensive view of the comments, see the complete images in Appendix B. All users' private information has been anonymized to ensure ethical standards are met. From the table, it's clear that nearly all negative feedback for The Outer Worlds centers around the simplicity of the perks rather than the system's overall complexity; many players expressed disappointment with the lack of exciting abilities that would make level-ups more anticipated. In contrast, positive comments focus on the unique weapon customization and the convenience of the perks. Compared to The Outer Worlds, discussions about the levelling system in Baldur's Gate 3 primarily revolve around Larian's decision to set a level cap, with less focus on the system's complexity. However, feedback suggests that many players find Baldur's Gate 3's levelling system overly complex, requiring a certain level of familiarity with DnD rules. At the same time, others appreciate this complexity, noting that it adds depth and enjoyment to the combat system.

In the case of BioShock Infinite, which features minimal progression elements, feedback on its levelling system is notably sparse. However, the popularity of a similar, low-progression element levelling system in the Call of Duty's Zombie mode suggests that players are generally

receptive to simpler levelling mechanics in games with a focus on action rather than deep role-playing systems.

2.3 The Concepts of Game Design

Different game designs have a significant impact on the gaming experience, shaping not only whether players are willing to engage with a game but also the type of players it attracts. The following section will explore how game design influences the overall gaming experience.

2.3.1 Immersive.

In the book A Theory of Fun for Game Design, Koster [2013] stated that different people will have different ideal games. Some players are particularly drawn to RPGs because they enjoy immersing themselves in the role of a virtual character they have crafted within the game. In his research, Klimmt et al. [2009] found that players like to merge their own identity with their game characters, shaping the characters in the way they want, which makes experiences of the game even more immersive and interesting. Similarly, Godtfredsen [2016] observed that players often create characters like a better version of themselves, or their ideal selves, living in a safe place to explore and embody their aspirations or personality traits. This can also enhance the sense of immersion.

This might explain why games like Fallout: New Vegas and Baldur's Gate 3 are so popular. These games offer players a quite safe and free world, and in there, players can explore things they cannot do in the real world. The games allow players to be the one they want to become, no matter a good person or a bad guy, while the progression system that can give players new abilities when they levels up, further deepens the immersive feelings. Each level-up further strengthens the player's connection to the world inside the game. This combination of freedom and meaningful progression is a key reason why these games resonate so strongly with players.

2.3.2 Challengeable.

How to properly design game difficulty is also a significant question of game design. Koster argues that the goal of a game is to challenge players, and poor design may allow players to circumvent these challenges by exploiting the rules. The research of Vahlo and Karhulahti [2020] supports this by saying that it can improve players' engagement and interest when a well-designed challenge matches their skills. Their study also highlights that different types of challenges can meet varied player preferences, broadening a game's appeal. This is agreed by Koster's view that games can exist without prescribed solutions, as such designs allow players to solve problems in multiple approaches. However, Koster cautions that games without structured challenges will possibly become "rather limited" and break the essence of gaming. But the shrine challenges in The Legend of Zelda: Tears of the Kingdom [Nintendo 2023] deny this idea. These challenges avoid a single solution, encouraging players to use different tools, strategies, and creative approaches to overcome the difficulties. This open-ended design not

only preserves the essence of gaming but also ensures that all the challenges are meaningful, requiring players to think critically and adapt without a defined framework.

Koster also mentioned a behaviour like "bottom-feeding" in his book, this behaviour describes players who prefer to take on weaker opponents because they think that although a series of guaranteed wins are boring, it is better than risking everything on a high risk, and uncertain battle. Games like Dark Souls [FromSoftware 2011] and Elden Ring [FromSoftware 2022], however, cleverly use this thought by subverting player expectations. While players may initially look for easier paths or weaker enemies, these games punch them with unpredictable difficulty Bosses and encounters. This design emphasizes using challengeable gameplay as the driving force, making failure and death the paths that players must pass and learn from as a part of their journey toward strong and success. This also challenges Koster's opinion that games are destined to become boring, giving examples of how sustained challenges can keep the experience engaging and rewarding.

2.3.3 Creative.

A great game has creativity, both in itself and in how it encourages players to express their own creativity. Zackariasson and Wilson [2009] found that creativity is the bedrock of the video game industry, driving innovation and shaping every aspect of game development. Throughout history, great games have shown their own creativity. For instance, by integrating the narrative into the FPS game, Half-Life [Valve 1998] illuminated the direction of the later development of the game. Similarly, Mass Effect takes the story-driven genre game to new heights, allowing players to influence the entire story of the trilogy by their own choices.

As mentioned above, a great game also encourages players to show their own creativity, allowing them to shape their own experiences inside the game world. As Wright [2006] described, video games often function as a "dream machines," it can offer open-ended "possibility spaces" for players to face challenges and solve problems by using their imaginary. Gee [2009] emphasized this in his research, well-designed games transform players into creators, giving them permission to modify the game and design new levels, as seen in LittleBigPlanet [Molecule 2008] and Portal 2 [Valve 2011]. In their research, Kim and Shute [2015] pointed out, great games allow creativity to flow between developers and players. For example, community-driven modifications or "mods", something made by players, in games like The Sims 4 [Maxis 2014] illustrate how players can expand and rebuild game content, demonstrating the collaborative potential of video games as platforms for user-generated content.

2.4 Experiment and Evaluation

Designing an experiment to evaluate player experience can provide valuable information about the impact of levelling system changes in video games. Questionnaires are quite important in this process, they offer a structured way to gather both quantitative and qualitative feedback from players. Krosnick [2018] did many research on questionnaire design, in his opinion, the key to obtaining better answers from respondents is to make sure they clearly understand

the intent of the question. To achieve this, he advocated for the use of precise, concise, and familiar language, avoiding ambiguity and ensuring the question adheres to principles of clarity, meaning uniformity, and univocality. Although closed-ended questions are more time-efficient and generally preferred by respondents for their simplicity, Krosnick highlighted that open-ended questions can provide more reliable and valid data, particularly in exploratory research or when the range of possible responses is unclear. For the rating scale, Krosnick suggested that the optimal length for bipolar scales (e.g., "like" to "dislike") is 7 points, since it strikes a balance between granularity and respondent ease of use, while for unipolar scales (e.g., "not at all important" to "extremely important"), the ideal length is 5 points, as it reduces cognitive load while maintaining measurement accuracy.

Other studies have also used effective methodologies for evaluations. For instance, in Kao's research on the effects of "juiciness" in RPGs [Kao 2020], he recruited volunteers to participate in testing different versions of the game, then he collected players' feedback by using the Player Experience of Need Satisfaction (PENS) scale [Ryan et al. 2006] and use the Intrinsic Motivation Inventory (IMI) to collected the gameplay data such as time spent playing, character level, and task completion rates. After that, Kao analyzed the data by using Multivariate Analysis of Variance (MANOVA) to evaluate various dimensions of player engagement across the different levels of "juiciness".

3 METHODOLOGY

3.1 Game Design

For the purposes of this study, we are going to design and implement a game prototype that focuses on three different versions of levelling systems. The game will utilize standard components for other gameplay aspects, with modifications where necessary, to ensure consistency across the different levelling mechanics:

1. The complex levelling system
2. The normal levelling system
3. The simple levelling system

These versions will be designed to be above the average level for video game ("The complex"), average ("The normal") and below the average ("The simple").

3.1.1 Detail of the systems.

For the complex system, the game prompts players to allocate a limited number of points to their character's attributes during the initial setup. Players are also required to choose a background for their character, each of which provides unique attribute bonuses. Finally, players are required to allocate points to their character's skills, each linked to a specific attribute and offering unique advantages. Each time the character levels up, players gain priority points to enhance skills and unlock corresponding perks. Each perk has specific requirements for attributes, skills, and levels.

To prevent imbalance in the late-game, characters have a level cap, and enemy levels scale accordingly with the character's progression.

The normal system shares a similar character creation process with the complex version, requiring players to allocate attributes and select a background. However, it does not involve assigning initial skills. Upon levelling up, players can choose from perks available at their current level but must decide between several options rather than unlocking all of them. Players have a level cap, and enemy levels are dynamically scale with the player's progression.

The simple system only requires players to select a background during character creation. As players accumulate enough money, they can purchase new perks and equipment or upgrade them at vending machines. Weapons do not have a levelling system. The difficulty of the game increases progressively as the game advances.

3.1.2 Detail of the implementation.

We have considered many options for game engines and conducted a thorough comparison among them (Table 1).

Name	Advantages	Disadvantages
Unreal[2024a]	Most advanced game engine today, delivers high-quality graphics	High hardware requirements, steep learning curve for beginners
Unity[2024]	Versatile, suitable for various types of game development	Some features require a paid subscription, not particularly beginner-friendly
GameMaker[2024b]	Ideal for prototyping or small-scale indie projects, offers good performance optimization	Smaller community with fewer development resources and plugins
Panda 3D[2024]	Python support for easy and rapid development, ideal for prototyping	Smaller community with fewer development resources and plugins
Godot[2024]	Completely free, allows developers to modify code freely; suitable for both 2D and 3D games, with excellent performance optimization	Limited third-party resources

Table 1. Game Engines

We finally chose Godot to develop the game in this project as Godot is more suitable for indie developers and beginners compared to the others. Godot employs a node-based structure to connect scenes and content within the game. For instance, when creating a “scene” with an "enemy," developers can first create a root node for the "scene" and another root node for the "enemy." After programming the enemy’s behavior using GDScript, the "enemy" node can be directly imported as a child node into the "scene." This approach makes the development process highly convenient and easy to learn.

For the level design, we plan to draw inspiration from Call of Duty’s Zombie mode and create a Metroidvania-style map. Players will start in an initial area with limited space. During the game, players must defeat enemies to earn money and experience points, then they can use money to unlock new areas, open shortcuts, and expand their movement range.

3.2 Experiment

In order to evaluate the effects of each system in video games, we will use a mixed-method approach that combines quantitative and qualitative data through a questionnaire survey. Participants will be recruited to play the game, after which they will provide feedback through both closed-ended and open-ended questions in the survey. Finally, the collected feedback will be systematically measured and analyzed to draw conclusions about the impact and effectiveness of each system. Based on these objectives, the following research questions were

formulated:

- **RQ1**: Does the complexity of the levelling system impact the game playability?
- **RQ2**: How does the levelling system impact the players' overall game experience?

RQ1 examines whether the complexity of the levelling system introduces convenience or inconvenience for players during gameplay, thereby influencing the playability of the game itself. While RQ2 delves deeper by focusing on whether the levelling system affects players' psychological factors, such as satisfaction and engagement, ultimately having effects their overall gaming experience.

3.2.1 Pre-experiment Preparation.
For the experiment, we will recruit a group of volunteers to test the game. Eight experienced video game players, aged between 18 and 40, will be selected, with all participants sourced from the campus to ensure both data reliability and convenience. They will be recruited through face-to-face inquiries on campus. The participants will be given an information sheet about the purpose of the experiment and the things they are supposed to do. Once they agree, they will need to sign a consent form which covers the purpose of the study, experimental procedures, data collection and processing methods, and compliance with GDPR guidelines, ensuring full protection of participants' privacy. And their contact details such as e-mail will be recorded for further file sending.

3.2.2 Experiment Procedure.
During the experiment, participants will receive the game file, a questionnaire and instructions for some more details (e.g. the time limitation, how to return the questionnaire). Then, they will be asked to play two randomly assigned versions of the game on their own devices. Each gameplay session will end once their character defeats a specific enemy and will take approximately 15 minutes or longer. Upon completing each session, the game will create a file with game statistics, participants can get a summary of their gameplay for that specific game version in this file, including data such as playtime duration, the number of enemies defeated, the amount of money accumulated, the experience points gained and the current level of the character. Based on the file and their personal feelings, participants will then complete a questionnaire for each version. After participants finish the questionnaires, they can return them and the files by following the instructions. In this period, participation is totally voluntary, and participants can withdraw at any time.

In the course of the trial, we will encrypt the data upon storage and separate email addresses from any other identifying information to ensure that individuals cannot be directly linked to their data. Furthermore, we will retain this data only for as long as necessary and promptly delete it since it is no longer required.

3.3 Questionnaire Design

The questionnaire design for this project will follow the theory mentioned in Section 2.4. In order to prevent respondents from being tired or annoyed by overly lengthy phrasing, the questions will be clear, which means it will use the fewest words to express their intent. Additionally, each question will maintain its meaning uniformity, only focusing on the concept that being measured and avoiding extraneous information that could confuse respondents then mislead to their answers. Finally, the questions will ensure univocality, with wording carefully designed to have the same meaning for all respondents, regardless of their background or comprehension level, thus reducing potential misunderstanding.

3.3.1 Design Philosophy.

To ensure the scientific validity and accuracy of the questionnaire, we will refer to the experiment mentioned in Section 2.4. Our project integrates elements from two classic user experience assessment tools: the System Usability Scale (SUS) [Brooke 1996] and the Player Experience of Need Satisfaction (PENS) [Ryan et al. 2006].

SUS includes 10 Likert-scale questions and will calculate answers to these questions to get a score for usability. Incorporating SUS into the questionnaire can evaluate participants' perceptions of the usability and ease of learning associated with different versions of the levelling system. For instance, asking participants about "the complexity of the levelling system" and "their willingness to choose this levelling system in future games" can provide useful information about the system's playability and its potential acceptance among players.

PENS is based on Self-Determination Theory (SDT) and often uses five dimensions—Competence, Autonomy, Relatedness, Presence/Immersion, and Intuitive Controls—to evaluate how much a game satisfies players' psychological needs. Integrating PENS into the questionnaire can give us a deeper understanding of how the levelling system impacts players' engagement with the game and their overall gaming experience. Considering that this project only focuses on the effects of the levelling system, the questionnaire will be designed using three selected dimensions of PENS: Competence, Autonomy, and Presence/Immersion. For example, a question like, "This levelling system allows me to play the game in the way I want" can evaluate the system's impact on players' sense of autonomy. By focusing on these dimensions, the questionnaire can precisely capture the impact of the levelling system on player experience.

3.3.2 Questionnaire Content.

The questionnaire for this project will be divided into four main sections with 17 questions and incorporates both 5-point Likert scale and open-ended question to ensure a comprehensive evaluation of the feedback. The complete questionnaire can be seen in the Appendix C.

The first section primarily focuses on measuring participants' intuitive perceptions of the levelling system during gameplay, providing feedback on the game's playability (or usability). The next section will assess players' satisfaction, sense of freedom, immersion and expectation during gameplay to help evaluate the system's impact on the overall gaming experience by.

Finally, open-ended questions are used to gather players' suggestions for system improvements, providing valuable information about the strengths and weaknesses of each system.

3.4 Data Analysis

For RQ1, we will first evaluate the Playability section of the questionnaire to calculate an overall usability score for each levelling system. These scores will then serve as the dependent variable in a one-way ANOVA - a method that will be used to analyze the differences across levelling systems of varying complexity (e.g., simple, normal, complex).

To use ANOVA, we first need to calculate the descriptive statistics for each system, which include mean and standard deviation. Then we can use ANOVA to check whether there are significant differences in player responses across the three levels of complexity by determining the p-value. If $p<0.05$, indicating significant differences are found, a post-hoc analysis, such as Tukey's Honestly Significant Difference (HSD) test, will be performed to identify which specific groups show significant differences.

As for RQ2, we can use MANOVA to analysis the Play Experience Feedback section as it examines multiple interrelated psychological factors. Like ANOVA, MANOVA takes the complexity of the levelling system as the independent variable and selects multiple dimensions of player experience as the dependent variables (e.g., competence, immersion, and autonomy scores from PENS). By calculating the p-value, we can evaluate whether there are statistically significant differences among the groups. If significant differences are identified ($p <0.05$), post-hoc tests will be conducted to determine which specific dependent variables or group comparisons contribute to the overall significance.

Finally, for the game statistics file, these data will be presented in the form of bar charts to visually illustrate the varying impacts of different levelling systems on gameplay. The open-ended questions in the final section will be analyzed qualitatively to identify recurring themes, gather in-depth information, and uncover potential areas for improvement in the levelling systems.

These methods, combining both qualitative and quantitative analyses, have the potential to provide a deeper insight into the impact of levelling systems on video games. By incorporating participant feedback, they also contribute to a further improvement in the design and functionality of levelling systems.

4 IMPLEMENTATION

This section explains how we implement our experiment, which includes two parts: Game development and Evaluation experiment.

4.1 Game Development

4.1.1 Basic design.

We divided the basic design into two parts.

The first part focused on implementing the character's basic mechanics. As we were complete beginners in using Godot, we searched for tutorials on YouTube related to character movement control and found a GitHub repository [Majikayo Games 2024b] containing the accompanying code. Of course, there were other resources available, such as Brackeys' "How to Make a Video Game - Godot Beginner Tutorial"[Brackeys 2024] and GDQuest's "3D TUTORIAL: Make Smooth 3D Movement in Godot 4."[GDQuest 2023] However, these tutorials were relatively fragmented and lacked a well-structured learning path. Therefore, we chose Majikayo Games' "Godot FPS Tutorial,"[Majikayo Games 2024a] which provided a more comprehensive and organized approach. The tutorial covered everything from basic character movement to environment interaction, weapon system implementation, and finally character animation integration. Additionally, its code flexibility far exceeded that of other resources. For example, its weapon system allows for the creation of new weapons by leveraging Godot's Resource system—instead of rewriting code, new files can be generated with modified parameters, making it easy to introduce new weapons without extensive reprogramming. We believed this resource would enable us to quickly learn and establish the foundational game mechanics, while also making future modifications and adjustments more efficient and manageable.

Building upon this foundation, we introduced an experience and currency system tied to the character's progression and implemented various skills along with their corresponding unlock requirements. This groundwork significantly streamlined the later development of our levelling system, providing a solid base for further enhancements and refinements.

The second part focused on level design. As mentioned in Section 3.1.1, we aimed to draw inspiration from Call of Duty's zombie mode [Treyarch 2008] and Metroidvania-style level structures. Because modifying an existing map allowed us to focus on the key aspects of implementation while also providing a practical starting point. Rather than building a map from scratch, we adapted a completed GitHub map to suit our needs, therefore, we selected this particular map [Legion Games 2024] because it was relatively simple, featuring a single flat plane, which made integration into our project more straightforward compared to other available options. This significantly simplifies AI pathfinding design, making it easier to implement and optimize enemy navigation in later stages of development.

To enhance gameplay progression, we divided the map into multiple smaller areas that players must gradually unlock as they advance, we also placed vending machines and ammo

boxes strategically throughout different sections, allowing players to restock supplies and purchase new weapons when needed.

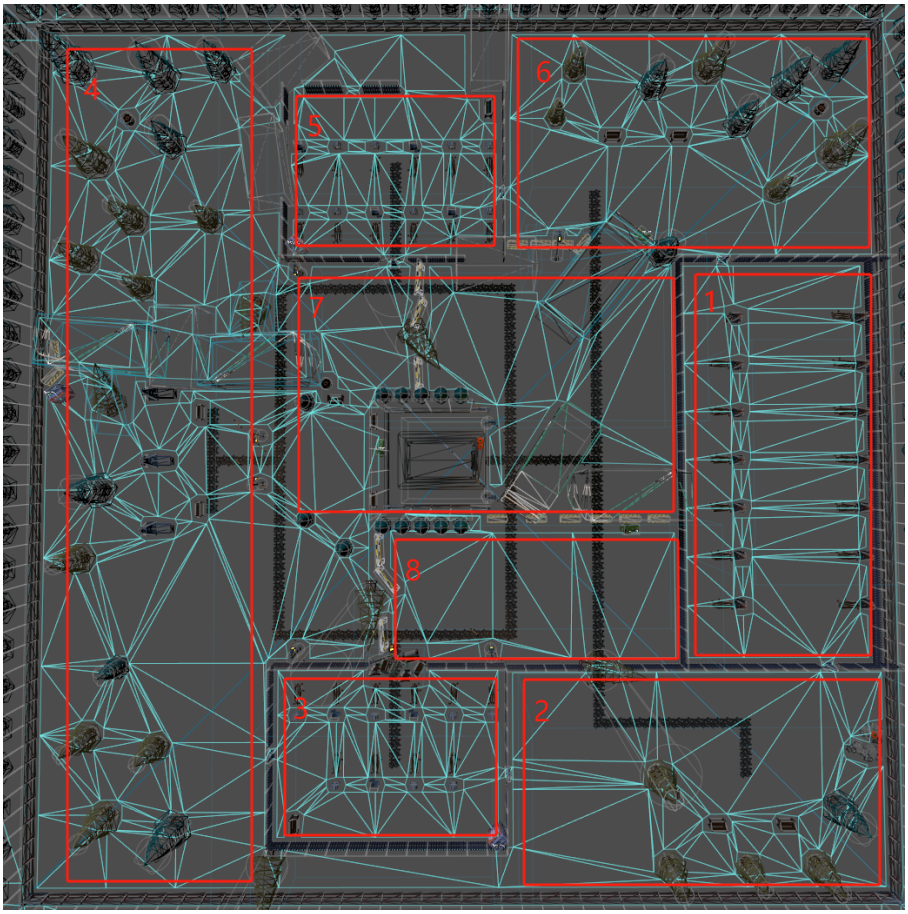


Fig. 6. The revised map

Additionally, we carefully designed the enemy spawn system to prevent enemies from appearing in predictable locations. Instead of using a single spawn point, we implemented multiple spawn zones, each containing several spawn points, as players navigate through different areas of the map, the corresponding spawn zones are activated via signals, triggering enemy spawns from the designated points within that zone. This dynamic system significantly increases both the challenge and randomness of enemy encounters, making the gameplay experience more engaging and unpredictable.

4.1.2 Levelling system.
Following the approach outlined in Section 3.1.1, we implemented an attribute-skill-perk

mechanism inspired by The Outer Worlds [Obsidian 2019] for our complex version. At the start of the game, players must choose their attributes and an aptitude, which influences the selected attributes. The chosen attributes determine the initial values of skills, and in turn, skills impact specific character properties. For example, every point invested in "Resilience" increases the character's damage reduction against enemies by 0.003. Unlike The Outer Worlds, we designed the perk unlock system following the approach used in Fallout: New Vegas [Obsidian 2010], where both attributes and skills collectively determine perk availability. This structure requires players to carefully consider their attribute and skill investments to optimize their character progression. For a detailed breakdown of attributes, skills, aptitudes, and perks, please refer to the Appendix F.

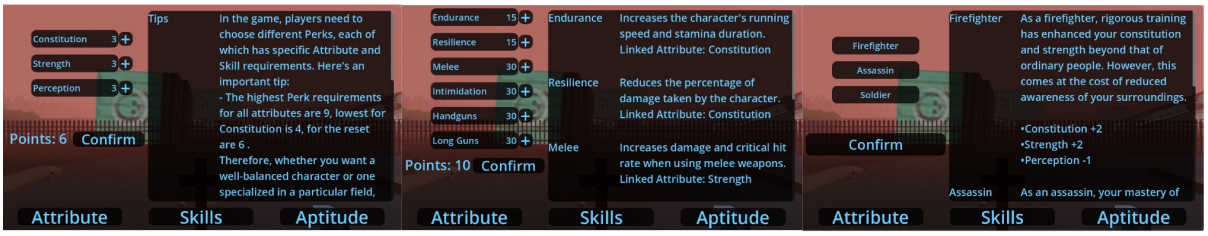


Fig. 7. The Attribute Allocate Menu (left), Skill Allocate Menu (mid) and Aptitude Choose Menu (right)

Following the method outlined in the Godot tutorial[Godot 2024], we implemented an experience requirement system where the amount of experience needed for each level increases exponentially:

$$Exp\ Requirement = round\left(200 \times (level^2 + 4 \times level)\right) \tag{1}$$

upon levelling up, players receive 10 skill points, which can be allocated to modify skills. Additionally, at every multiple of level 2, players gain extra 10 skill points and an extra perk point to unlock new perks. Furthermore, levelling up does not only affect the character but also dynamically adjusts enemy difficulty: As the character's level increases, enemy health and damage scale accordingly, ensuring a consistent level of challenge throughout the game.

The normal version of our levelling system is a simplified adaptation of the complex version. In this version, we removed the skill system, meaning that all character attributes directly determine the character's properties. As a result, when character level up, they receive attribute points instead of skill points. Additionally, perk unlocks no longer depend on specific skill or attribute requirements; instead, they are tied solely to character level. Similar to the complex version, players earn perk points at every multiple of level 2, which can be used to unlock perks. The enemy scaling mechanics remain the same as in the complex version, ensuring that as players progress, enemy health and damage increase accordingly to maintain the game's challenge.

The simplified version completely removes the character levelling system. Although in Section 3.1.1 we initially proposed allowing players to choose an aptitude, we ultimately abandoned this feature for game balance and technical reasons. As a result, players cannot manually adjust their character's properties in this version. However, despite the removal of aptitude selection, players can still purchase perks from vending machines. Additionally, players can upgrade their weapons on the upgrade machine, enhancing various attributes such as damage and recoil. In this version, enemy attributes scale with game progression—as the game progresses, enemies become increasingly stronger, with higher health and damage output. This mechanic forces players to upgrade their weapons to keep up with the growing challenge and effectively combat the increasingly powerful enemies.

Fig. 8. The Upgrade Machine (left) and Vending Machine(right)

4.1.3 Balancing.

To enhance the credibility of our experiment, we implemented balance adjustments across all versions of the game. For all the versions, players start with only a pistol and a knife. To obtain more powerful weapons, they must purchase them from vending machines. Additionally, to encourage exploration and area unlocking, stronger weapons in vending machines are locked behind a progression system, requiring players to unlock a certain number of areas before these weapons become available for purchase.

The enemy spawning system follows a standardized wave-based structure, inspired by Call of Duty: Zombies mode. In each wave, a set number of enemies spawn, and players must eliminate all enemies in the current wave before advancing to the next. As the waves progress, the game's difficulty increases, starting with 20 enemies in the first wave. Each following wave spawns 10 additional enemies, and their attributes scale over time, which will be explained in detail later, making them progressively stronger. This ensures a gradual difficulty curve that challenges players to adapt their strategies, manage resources, and upgrade their characters or weapons accordingly.

Fig. 9. The Block

The enemy AI design is standardized across all versions using Godot's NavigationAgent [Godot 2024]. This system utilizes the A-Star algorithm to pre-bake navigation paths for the map, allowing enemies to quickly and efficiently determine the shortest route between two positions in real time. As a result, enemies can dynamically navigate the map using the shortest path to reach the player's current location. Once within attack range, enemies will engage in melee attacks. By leveraging real-time pathfinding, enemies can avoid obstacles and adjust their movement based on the player's position, creating a more responsive and challenging AI behavior. This ensures that players must stay mobile, strategically position themselves, and utilize the environment to survive against increasingly aggressive enemies.

4.1.4 Structure and codes.

Nodes are the fundamental building blocks of Godot's scene system, serving as modular components that can be arranged hierarchically to form a tree structure. Each node can have child nodes, and all sibling nodes (direct children of the same parent) must have unique names to maintain clarity within the hierarchy [Godot 2024]. Unlike traditional object-oriented programming (OOP), where objects are instantiated from classes and rely heavily on inheritance, Godot's node system emphasizes composition—instead of creating deep inheritance trees, functionality is built by combining different nodes within a scene. This scene-based approach allows for better modularity and reusability, as entire node structures can be saved as scenes and instantiated multiple times, similar to class instances in OOP but with a more visual and structured workflow. Additionally, while OOP typically uses methods and properties within objects to define behaviors, Godot nodes function independently and can be individually scripted using GDScript, C#, or other supported languages [Godot 2024], allowing for greater flexibility in game development. This hybrid approach combines the hierarchical structure of a scene graph with object-oriented principles, making it easier to manage complex game logic while maintaining an intuitive, modular, and scalable design.

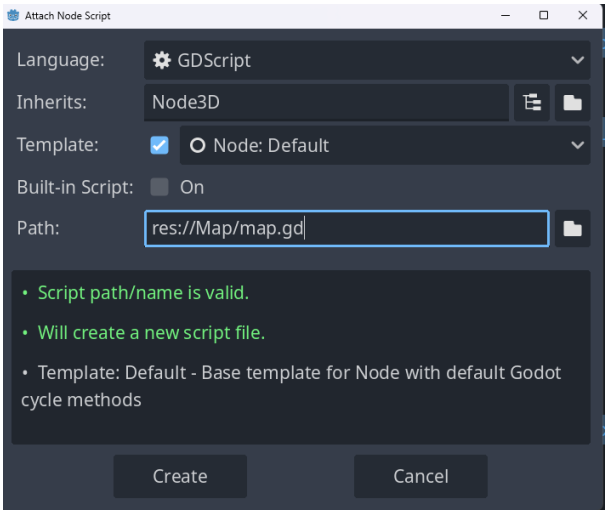


Fig. 10. Adding Scripts for Node

The overall structure of the game is roughly shown in the Figure 11. World serves as the parent node of the entire project, with all game-related nodes organized as its child nodes. Among them, the Map node contains various game scenes, including player scene, enemy scenes, and other environmental elements such as stores and blocks.

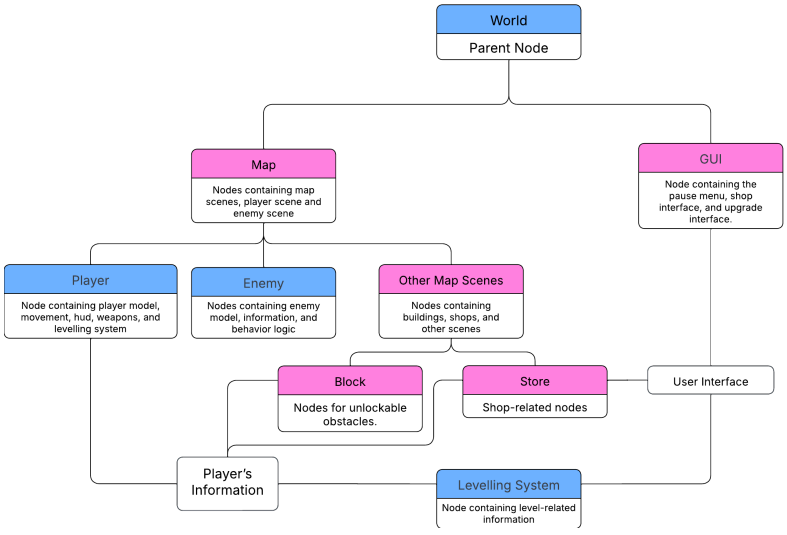


Fig. 11. The structure of the project

The Levelling System is divided into two parts: the User Interface and the Player's Information. The User Interface is created as a child node of GUI, while Player's Information is stored inside Player node and its child nodes.

As mentioned in Section 4.1.1, the project was developed based on YouTube tutorials, incorporating improvements to the provided codes. This section focuses on detailing these enhancements.

a) Player node enhancement.

We introduced a child node named LevellingSystem under the Player node. This node handles experience point acquisition and calculates the experience required for each level-up, a feature exclusive to the Complex and Normal versions. Additionally, it defines character Perks, including their descriptions, unlock conditions, and functionality implementation. The check_perk_requirements method, located in the Character_Sheet node (or the Store node in the Simple version), verifies these unlock conditions and returns relevant information.

Fig. 12. Methods of gaining experience and levelling up (left) and the execution method of perks and the dictionary storing perks (right)

Furthermore, in the Complex and Normal versions, the LevellingSystem is responsible for computing the impact of attributes and skills on the character's overall performance. For instance, increasing Melee by one point enhances melee damage by 1%.

```
func update_influence_from_skills():
    player.skills_influence["endurance"] = 1 + max(-0.5, player.skills["endurance"] -15) * 0.005
    player.skills_influence["resilience"] = max(-0.5, player.skills["resilience"] -15) * 0.003
    player.skills_influence["melee"] = 1 + max(-0.5, player.skills["melee"] -15) * 0.01
    player.skills_influence["intimidation"] = max(-0.5, player.skills["intimidation"] -15) * 0.005
    player.skills_influence["handguns"] = max(-0.5, player.skills["handguns"] -15) * 0.006
    player.skills_influence["longguns"] = max(-0.5, player.skills["longguns"] -15) * 0.006
    for key in player.skills_influence:
        if player.skills_influence[key] < 0 :
            #print(player.skills_influence[key])
            player.skills_influence[key] *= 100
```

Fig. 13. The method of calculating the impact of skills on player stats.

We can also observe in Figure 13 that in both the Normal and Complex versions, attributes and skills influence the character through the same underlying mechanism—they are ultimately converted into skills for calculation. However, this process for Normal version happens behind the scenes, giving players the illusion of an easier system than the latter. A detailed explanation of this implementation will be provided in the Character_Sheet node.

In addition to adding a new child node to the Player, we also modified the code of the Player node, its child nodes, and related classes to accommodate various functionalities and integrate the Levelling System's impact on character stats.

We introduced a dictionary in the Player node to store information about the player's weapons and gear. For instance, when a player attempts to purchase a weapon from the shop, a method in the Store node retrieves its purchase cost from the dictionary and compares it with the player's available funds. If the player has enough money, the required amount is deducted, and the weapon's "owned" key is set to true, preventing duplicate purchases. A detailed explanation of this process will be provided in the Store node section.

```
"axe" :    {
        "name" : "axe",
        "owned" : false,
        "price" : 1000,
        "type": "weapon",
        "description" : "Stolen straight from Kratos' hands—the Leviathan Axe! Just kidding...
                          If that were true, I'd have lost my head already.",
        "file_path": "res://FpsControllor/weapon_manager/axe/axe.tres"
        },
```

Fig. 14. The dictionary storing weapons

Additionally, we refined existing methods in the Player node, such as take_damage and add_recoil, along with the WeaponResource class, which manages weapon attributes, to integrate Perk functionalities and ensure that attributes and skills effectively influence key mechanics, including damage, spread, recoil, and other combat-related effects.

To further enhance gameplay, we introduced new methods in the Player node, such as fulfill_ammo and use_medic, to improve game mechanics, expand player interactions, and enrich the overall game experience. We also modified the WeaponManager node, a child node of the Player, by adding an add_weapon method, enabling players to correctly equip weapons after purchase.

```gdscript
func fullfill_ammo():
    var current_weapon = $WeaponManager.current_weapon
    if (current_weapon.current_ammo and
        current_weapon.reserve_ammo == INF) or (current_weapon.current_ammo == current_weapon.magazine_capacity and
            current_weapon.reserve_ammo == current_weapon.max_reserve_ammo):
        return
    if currency < 1000:
        $PlayerHUD.get_node("InteractiveWarning").set_visible(true)
        $PlayerHUD.get_node("InteractiveWarning").text = "You need more money!"
        await get_tree().create_timer(3).timeout
        $PlayerHUD.get_node("InteractiveWarning").set_visible(false)
        return
    else:
        current_weapon.fullfill_ammo()
        currency -= 1000
```

Fig. 15. The method of refilling ammo

b) User-interactable system.

The user interaction system is primarily divided into two categories: the levelling interaction system for the Complex and Normal versions, and the store interaction system.

The levelling interaction system is activated when the player allocates points or selects Perks during character creation or levelling up. This system consists of a single node, which is instantiated as a child node under the GUI node when called.

Upon activation, it retrieves relevant data from the Player node, such as level, skill and attribute values, and available points, then updates this information on the interface. Once opened, players can allocate points to attributes or skills, a process managed through a series of coordinated methods.

Initially, both increase and decrease buttons are disabled. When available points are detected, increase buttons become enabled, allowing players to allocate points. Temporary values (e.g., strength_add, melee_add) track changes before confirmation. Players can increase or decrease attribute or skill values temporarily, with the UI dynamically updating to reflect the changes. Increasing a value reduces the available points, enables the decrease button, and disables further increases if no points remain.

Fig. 16. Methods of increasing/ decreasing skills

Finally, players confirm their allocation by clicking the Confirm button. In the Normal version, attribute values are proportionally converted into corresponding skill values, representing the

"hidden conversion" mentioned earlier. The final skill values are then returned to the Player node, where they take effect and influence gameplay accordingly.

```
func _on_attribute_confirm_pressed() -> void:
    if strength_add + constitution_add + perception_add == 0:
        print("Nothing changed")
    else :
        character.attribute_available_points = attribute_available_points
        character.attributes["constitution"] += constitution_add
        character.attributes["strength"] += strength_add
        character.attributes["perception"] += perception_add
        character.skills["endurance"] += constitution_add * 5
        character.skills["resilience"] += constitution_add * 5
        character.skills["melee"] += strength_add * 5
        character.skills["intimidation"] += strength_add * 5
        character.skills["handguns"] += perception_add * 5
        character.skills["longguns"] += perception_add * 5

        strength_add = 0
        constitution_add = 0
        perception_add = 0
        load_stats()
        for button in get_tree().get_nodes_in_group("AttributeMinusButtons"):
            button.set_visible(false)
        for label in get_tree().get_nodes_in_group("AttributeChangeLabels"):
            label.set_text(" ")
        if attribute_available_points == 0:
            $HBoxContainer/VBoxContainer/Attributes/AttributeName/AttributePoints/AttributeConfirm.set_visible(false)
```

Fig. 17. The method of detecting the Confirm button press

Perk selection first checks requirements through check_perk_requirements, which verifies whether the player's stats meet the conditions stored in the perk dictionary. If the requirements are met, the corresponding perk button becomes available. When the player clicks the button, the perk's value is set to true, and the button's border remains highlighted, serving as a visual indicator that the perk has been acquired.

```
func check_perk_requirements(perk_id: String) -> bool:
    if not levelling_sys.perk_requirement.has(perk_id):
        return false

    var req_dict =  levelling_sys.perk_requirement[perk_id]

    if req_dict.has("attribute"):
        for attr_name in req_dict["attribute"]:
            var required_val = req_dict["attribute"][attr_name]
            var current_val = character.attributes[attr_name] if character.attributes.has(attr_name) else 0
            if current_val < required_val:
                return false
    if req_dict.has("skill"):
        for skill_name in req_dict["skill"]:
            var required_val = req_dict["skill"][skill_name]
            var current_val = character.skills[skill_name] if character.skills.has(skill_name) else 0
            if current_val < required_val:
                return false

    if req_dict.has("points"):
        var required_val =  req_dict["points"]
        if character.perk_available_points < required_val:
            return false

    return true
```

Fig. 18. The method of checking requirement

The store interaction system functions similarly to the levelling interaction system, retrieving relevant player data and updating the interface upon activation. When a player attempts to purchase an item, the system accesses the corresponding dictionary in the Player node to verify whether the player has sufficient funds. If the required amount is available, the purchase is completed; otherwise, a "Not enough money" message is displayed.

```
func _on_purchase_button_pressed() -> void:
    if selected_item_id == "":
        %Warning.text = "No weapon selected."
        return

    if not character.weapons.has(selected_item_id):
        $HBoxContainer/VBoxContainer/Weapons/WeaponInfo/Panel/MarginContainer/Info/WeaponInfo/WarningLabel.text = "Invalid weapon."
        return

    var w_info = character.weapons[selected_item_id]
    var cost = w_info["price"]


    if currency < cost:
        %Warning.text = "Not enough money!"
        return
    match w_info["type"]:
        "weapon":
            _buy_weapon_logic(w_info)
        "medkit":
            _buy_medkit_logic(w_info)
        _:

            $ItenInfo/WarningLabel.text = "Unknown item type!"
```

Fig. 19. The method of detecting the Purchase button press

Once a weapon is purchased, its “owned” key in the dictionary is set to true. During store interface updates, the system checks purchased weapons and hides their corresponding buttons, preventing duplicate purchases.

```
func _buy_weapon_logic(info: Dictionary) -> void:
    if info["owned"] == true:
        %Warning.text = "You already own this weapon!"
        return
    currency -= info["price"]
    info["owned"] = true
    _update_weapon_buttons()
    var weapon_file_path = info["file_path"]
    var weapon_resource = load(weapon_file_path)
    if weapon_resource == null:
        $HBoxContainer/VBoxContainer/Weapons/WeaponInfo/Panel/MarginContainer/Info/WeaponInfo/WarningLabel.text = "Weapon resource not found: " + weapon_f
        return
    var weapon_manager = get_node("../../Map/Player/WeaponManager")
    if weapon_manager and weapon_manager.has_method("add_weapon"):
        weapon_manager.add_weapon(weapon_resource)
    else:
        $HBoxContainer/VBoxContainer/Weapons/WeaponInfo/Panel/MarginContainer/Info/WeaponInfo/WarningLabel.text = "Can't add weapon. WeaponManager not fou


    %Warning.text = "Purchase success!"
```

Fig. 20. The method of purchasing weapons

As previously mentioned, the availability of weapons in the store is tied to the number of unlocked areas, ensuring that weapon options expand progressively as the player advances. The system begins with a default number of visible weapon slots, which increases as the player unlocks new areas by opening doors. As more areas become accessible, additional weapons become available for purchase. During the update process, the system iterates through all weapon buttons, verifying whether each weapon falls within the current visibility range. If a weapon is within the allowed range, its button remains visible; otherwise, it is hidden.

```gdscript
func _update_weapon_buttons() -> void:
    %Currency.text = "$ " + str(currency)
    var parent_node = $HBoxContainer/VBoxContainer/Weapons/WeaponName
    var child_count = parent_node.get_child_count()
    var default_visible = 4
    var extra_for_doors = floor(door_opened / 2)
    var total_visible = default_visible + extra_for_doors

    for i in range(child_count):
        var wbutton = parent_node.get_child(i+1)
        #print(i)
        var w_id = wbutton.name.to_lower()
        #print(w_id)
        if not character.weapons.has(w_id):
            #wbutton.visible = false
            continue
        var w_info = character.weapons[w_id]
        match w_info["type"]:
            "weapon":
                if w_info.has("owned") and w_info["owned"] == true:
                    wbutton.visible = false
                else:
                    if i+1 < total_visible:
                        wbutton.visible = true
                    else:
                        wbutton.visible = false
            "medkit":
                if w_info.has("owned_quantity") and w_info.has("max_quantity"):
                    if w_info["owned_quantity"] < w_info["max_quantity"]:
                        wbutton.visible = true
                        return
                    else:
                        wbutton.visible = false
                        return
                else:
                    wbutton.visible = true
                    return
```

Fig. 21. The method of updating the weapon button

c) Map and Enemies.

We utilize Block nodes to divide the game map into multiple distinct areas, each requiring money to unlock, similar to the Call of Duty: Zombies mode. When a Block is within the player's interaction area, it retrieves relevant player data, such as their current money. Upon pressing the interaction key, the system checks whether the player has sufficient funds. If the condition is met, the Block is removed, and the corresponding amount is deducted from the player's balance.

To maintain game balance and progression pacing, all Blocks are grouped together. Each time a Block is removed, the unlock cost for the remaining Blocks in the group increases, preventing players from unlocking all areas too quickly.

```
func _on_interactable_component_interacted() -> void:
    if has_used:
        return
    if has_node(player_path):
        var local_player = get_node(player_path)
        if local_player.currency < cost:
            return
        local_player.currency -= cost
        world.door_unlocked += 1
        get_tree().call_group("Doors", "new_door_opened")
        has_used = true
        $AnimationPlayer.play("Open")

        #door_opend += 1
        #map.door_opened = door_opend
        await  get_tree().create_timer(3).timeout
        queue_free()
    else:
        print("no player found")
```

Fig. 22. The method of block interaction

Additionally, we have placed interactive objects on the map, such as Stores, Ammo Crafts, and Upgrade Machines. These objects follow a similar interaction mechanism to Blocks, retrieving player data upon interaction and executing corresponding actions based on the retrieved information.

For enemies, we utilized free online resources [Mixamo 2025] to create models and animations, integrating them with behavioral logic to enhance gameplay dynamics. For instance, enemies are programmed to continuously move toward the player's position using the navigation agent in Godot, and when within range, they trigger attack logics.

Additionally, we implemented a hitbox-based detection system, where attacking different parts of an enemy's hitbox results in varying damage and monetary rewards. Enemy health and attacking damage scales as the game progresses, but the scaling mechanism differs across versions. In the Complex and Normal versions, enemy attributes adjust based on the player's level—higher player levels result in stronger enemies with increased health and damage. However, defeating stronger enemies also grants more experience points, aligning with the concept of "enemies also having levels," as previously mentioned.

```gdscript
func _on_area_3d_body_part_hit(dmg, critical_multi) -> void:
    #print("hit")
    if is_dead:
        return
    health -= dmg * critical_multi
    player.currency += 10 * critical_multi
    player.currency_earned += 10 * critical_multi
    $Hp.text = str(health)
    #print(health)
    if health <= 0:
        if not is_dead:
            is_dead = true
        $Hp.visible = false
        var level = player.curr_level
        player.currency += 100
        player.currency_earned += 100
        var exp = get_experience_per_zombie(level)
        player.gain_exp(exp)
        if $CollisionShape3D:
            $CollisionShape3D.queue_free()
        anim_tree.set("parameters/conditions/die", true)
        emit_signal("zombie_died")
        await get_tree().create_timer(6).timeout
        queue_free()
```

Fig. 23. The method of handling enemy hit detection

In contrast, in the Simple version, enemy attributes scale based on wave progression instead of player level. As the wave count increases, enemies become stronger, with higher health and attack power, ensuring a gradually intensifying challenge.

```gdscript
func get_zombie_hp(waves: int) -> int:
    return round(base_health * 1.1 ** (waves-1))

func get_zombie_damage(waves: int) -> int:
    return round(min(max_damage, base_dmg +
                (round_modifier * ((waves - 1) / 5))))
```

```gdscript
func get_zombie_hp(player_level: int) -> int:
    return round(base_health + 30 * player_level + 2 * pow(player_level, 2))

func get_zombie_damage(player_level: int) -> int:
    return round(30 + 3 * player_level)
```

Fig. 24. Methods of calculating enemy-related attributes: based on wave number (left) and based on player level (right)

To ensure the game functions properly, we implemented specific logic within the World node. First, we designed a timer that controls the spawning frequency of enemies, ensuring that they appear at a consistent interval. This also provides players with a brief window between waves to restock supplies before the next encounter. Following this, we introduced a wave controller that continuously monitors the remaining enemies in the current wave, the total number of enemies present on the map, and dynamically calculates the number of enemies to be spawned in the next wave.

```
func _set_wave(wave_num: int) -> void:
    wave_zombies_spawned = 0
    wave_zombies_alive = 0
    wave_zombies_killed = 0
    if wave_num == 1:
        wave_zombies_to_spawn = first_wave_count
    else:
        wave_zombies_to_spawn = first_wave_count + wave_increment * (wave_num - 1)
    print("== Start Wave ", wave_num, " : Need to spawn ", wave_zombies_to_spawn)
```

Fig. 25. The method of handling waves

Additionally, we assigned designated spawn points to different areas of the map. When the system detects that the player has entered a new area, the corresponding spawn points are activated, triggering enemy spawns in that region.

```
func _get_random_spawn_position() -> Vector3:
    if not spawns or spawns.get_node(area).get_child_count() == 0:
        return Vector3.ZERO

    var child_count = spawns.get_node(area).get_child_count()
    var random_id = randi() % child_count


    if child_count > 1:
        while random_id == last_position:
            random_id = randi() % child_count

    last_position = random_id

    var spawn_node = spawns.get_node(area).get_child(random_id)
    if spawn_node:
        return spawn_node.global_position
    return Vector3.ZERO
```

Fig. 26. The method of handling respawn points

4.2 Evaluation Experiment

To evaluate our project, we recruited 20 volunteers from the campus and carried out a two-week testing period. After filtering out low-quality responses—such as patterned responses

where participants selected the same option for all questions, contradictory answers within the same questionnaire, and responses that demonstrated a clear lack of engagement, such as completing the survey in an unrealistically short time—we obtained 14 valid questionnaire feedback for each version.

The questionnaire used a rating system to evaluate different aspects of the game. Questions 1 to 6 assessed playability, where questions 1, 3, and 5 represented negative feedback questions, while questions 2, 4, and 6 represented positive feedback questions. Additionally, questions 7 to 15 evaluated three key factors: Competence (questions 7-9), Autonomy (questions 10-12), and Immersion (questions 13-15), all of which were positive feedback questions.

During data analysis, we performed a one-way analysis of variance (ANOVA) on questions 1 to 6 of the questionnaire. If the results showed $p < 0.05$, indicating a statistically significant difference in mean values among different versions, we then applied Tukey HSD to conduct pairwise comparisons to identify which specific versions exhibited statistically significant differences.

For questions 7 to 15, we first categorized them based on predefined or validated dimensions, grouping multiple questions under the same psychological or experiential factor.

Subsequently, we conducted a multivariate analysis of variance (MANOVA), using Version as the independent variable and the average scores of each factor as multivariate dependent variables to examine the overall differences across versions in these experiential or psychological dimensions. If the results showed $p < 0.05$, indicating that the MANOVA results were statistically significant, Tukey HSD would be applied, just as in the previous analysis.

For questions 16-17 and game-related statistical data that participants will submit, we treated them as supplementary data to provide additional context for player performance and subjective evaluations.

5 RESULTS AND ANALYSIS

In this section, we present the results of the questionnaire survey and the analysis.

5.1 Game Playability

The results of the one-way analysis of variance (ANOVA) indicate that the complexity of the levelling system has a significant effect on playability scores ($F = 5.73$, $p = 0.0066$). Since $p < 0.05$, this suggests that at least two versions exhibit statistically significant differences. Therefore, we conducted a Tukey HSD post-hoc test to identify specific differences between versions. The results show:

- A significant difference between the Complex and Normal versions ($p = 0.0383 < 0.05$).
- A significant difference between the Complex and Simple versions ($p = 0.0075 < 0.05$).
- No significant difference between the Normal and Simple versions ($p = 0.7932 > 0.05$).

These findings indicate that the Complex version has significantly lower playability scores compared to both the Normal and Simple versions, suggesting that an overly complex levelling system negatively impacts the game's playability. Some players also mentioned in question 16 that they struggled to distinguish between attributes and skills while playing the Complex version, which we believe further contributed to its lower playability score. However, the lack of a significant difference between the Normal and Simple versions suggests that if the levelling system is not overly complex, proper adjustments to its complexity do not have a significant impact on playability. We believe this indicates that if the complexity of the levelling system remains within a range that players find acceptable, changes to its complexity will not significantly affect their perception of the game's playability.

We can roughly summarize that while excessive complexity reduces playability, a levelling system with moderate complexity remains unaffected by appropriate adjustments.

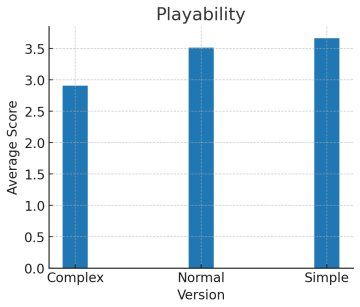


Fig. 27. Statistical results on game playability

5.2 Player Experience

The results of the one-way multivariate analysis of variance (MANOVA) indicate that levelling system complexity has no significant impact on the player's gaming experience at the multivariate level:

- Wilks' λ = 0.9010, F = 0.6603, p = 0.6818
- Pillai's trace = 0.1001, F = 0.6674, p = 0.6761
- Hotelling-Lawley trace = 0.1088, F = 0.6623, p = 0.6803
- Roy's greatest root = 0.0965, F = 1.2229, p = 0.3146

This finding is further supported by the one-way ANOVA results, which also show no significant differences across versions in any of the individual dimensions: Competence (F = 0.2656, p = 0.7681), Immersion (F = 0.6686, p = 0.5182), and Autonomy (F = 1.5147, p = 0.2325). As none of these dimensions reached significance, post-hoc tests were not necessary. Furthermore, we can observe that all three versions received scores above the median score of 3 across all dimensions. This suggests that the levelling system provided a certain degree of positive reinforcement for the player's overall gaming experience.

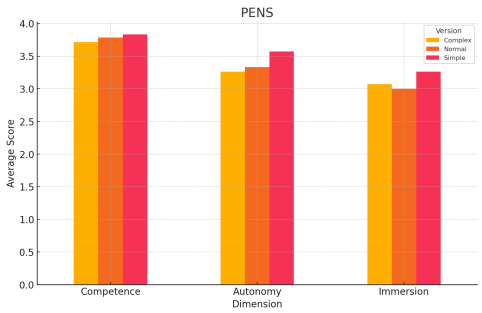


Fig. 28. Statistical results on the players' overall gaming experience

Additionally, we conducted further analysis incorporating other questionnaire items to gain a deeper understanding of how the levelling system impacts the player's experience. However, since most participants did not submit game-related statistical data, we only integrated questions 16 and 17 into our analysis.

As shown in the Figure 28, there is no significant difference in Competence scores across different levels of levelling system complexity, and all scores are above the median value of 3. This suggests that the complexity of the levelling system does not impact players' sense of Competence during gameplay and that all versions of the levelling system provide a positive reinforcement effect on players' competence.

In terms of Autonomy, the Simple version received significantly higher scores than the other two versions. We believe this is because, although players in the Complex and Normal

versions can customize their character stats, their ability to unlock Perks is strictly limited by factors such as level requirements, attributes, and skills. In contrast, while the Simple version does not allow direct modification of character stats, it grants complete freedom when upgrading weapons or unlocking Perks—as long as the player has enough money. Compared to the structured customization of character stats in the other versions, this unrestricted choice may have provided an even stronger sense of freedom and control, making players feel more autonomous in their decision-making.

For Immersion, the Normal version received the lowest score, which was somewhat unexpected. However, upon further reflection, we believe we have identified the reason behind this phenomenon. In the Normal version, the levelling system determines Perk unlock paths based on character level, meaning that players can only unlock certain Perks once they reach a specific level. Compared to the strictly regulated system in the Complex version and the completely unrestricted system in the Simple version, this approach may have felt too rigid and uninspiring, limiting players' sense of progression and agency, thereby weakening their engagement with the game and reducing their overall sense of immersion. Additionally, some players mentioned in question 20 that character progression felt too slow, which we believe further contributed to their loss of immersion: The prolonged levelling process may have made progression feel unrewarding, reducing players' motivation to stay engaged with the game.

In questions 16 and 17, several participants mentioned that some Perks in the game felt useless. We believe this may be one of the reasons why Immersion scores were slightly lower compared to the other two dimensions. The inability of certain unlocked Perks to meet players' expectations for meaningful progression could have diminished their motivation to continue playing, thereby leading to a lower Immersion score. We also believe that the high difficulty level of the game may have contributed to this outcome. Apart from some participants explicitly mentioning the game's difficulty in the questionnaire, several others provided similar feedback informally. However, since difficulty balance is not directly related to the levelling system, we acknowledge this factor but do not conduct further analysis on it.

Overall, while the complexity of the levelling system does not have a statistically significant impact on the player's overall gaming experience, and all versions provide a certain degree of positive reinforcement, different levelling systems shape player experience in distinct ways: The Simple version enhances Autonomy by offering unrestricted Perk purchases and weapon upgrades, while the Normal version received the lowest Immersion score, likely due to its rigid Perk unlocking style and slow character progression.

6 CONCLUSION

In this section, we will discuss the achievements of our project, its contributions to the gaming industry, its limitations and future works.

6.1 Achievements

Our project broadly categorizes levelling systems into three distinct versions and uses Godot to develop a game that incorporates these three different levelling systems within a unified framework. Through a questionnaire-based approach, we explored two key research questions:

- Does the complexity of the levelling system impact the game playability?
- How does the levelling system impact the players' overall game experience?

From our analysis, we arrived at the following conclusions:

- Overly complex levelling systems have a significant negative impact on playability, while adjusting the complexity within a reasonable range does not lead to a significant difference in playability.
- Different levelling systems influence various aspects of the player's experience, mostly in a positive way, but do not result in statistically significant differences.

By combining our statistical analysis of the questionnaire responses with the conclusions drawn earlier, we have addressed the impact of levelling systems on both game playability and players' game experience. Furthermore, our findings suggest that the idea of integrating different types of levelling systems within a unified game framework is feasible to a certain extent.

6.2 Contributions

In today's gaming landscape, many players are discouraged from trying certain games due to their overly intricate levelling systems and steep learning curves. For example, in Baldur's Gate 3, only 23% of Steam players have completed the game [Steam Community 2023], illustrating how complex mechanics can alienate potential players and limit a game's reach.

Our project provides game developers with a broader perspective by demonstrating the feasibility of integrating levelling systems with varying complexity within a unified game framework. This approach allows developers to cater to different player preferences, making games more accessible while preserving depth for those who seek it. If a game like Baldur's Gate 3 implemented such an approach, more players might be able to experience its intricate gameplay design and rich narrative, leading to a win-win scenario where developers expand their audience while players of varying skill levels can engage with the game in a way that best suits their preferences.

6.3 Limitations

Despite successfully addressing our research questions, our project has several limitations.

First, although we integrated three different levelling systems into a unified game framework, we essentially developed three separate games rather than implementing a single game with multiple selectable levelling systems. As a result, we lack a comprehensive understanding of the technical complexity and workload required to implement multiple levelling systems within the same game. This is why we can only conclude that this approach is feasible to a certain extent, rather than making a definitive claim about its practicality.

Second, there are several shortcomings in our game design:

(1) Code Structure – The game's codebase has low usability, making further research and development challenging. If future studies aim to expand on this project, a major code overhaul would likely be necessary to improve its structure, stability, and performance.
(2) Difficulty Balancing – Influenced by the classic Call of Duty: Zombies mode, we set enemy attack damage relatively high and did not include difficulty adjustment options. This led to many participants struggling with the game, which in turn negatively impacted their experience. As a result, some players did not engage deeply enough with the levelling system, potentially introducing bias into our analysis.
(3) Perk Design – As noted earlier, many participants reported that some Perks felt useless, which may have further contributed to analytical discrepancies.

Lastly, our evaluation methodology also has limitations. The number of participants was relatively small, and given the time constraints of our participants, we did not conduct a focus group evaluation. Instead, we relied on ANOVA and MANOVA for analysis, which may have impacted the robustness of our findings. Additionally, some participants did not fully complete the testing process with diligence, further affecting the credibility of our results.

These limitations primarily stem from the prototype nature of our project, as this was our first attempt at developing such a system. While the design is not perfect, the development process has provided valuable insights into potential improvements for future iterations.

6.4 Future Works

This project has several potential directions for expansion, including refining the classification of levelling systems, expanding to different game genres, analyzing long-term player engagement, and exploring the structure and reusability of the levelling systems developed in this study.

First, further refinement of levelling system complexity could provide deeper insights. Games like Mass Effect [BioWare 2007] and Kingdom Come: Deliverance 2 [Warhorse Studios 2025] do not fit neatly into any of the three categories we defined. Instead, they represent a middle ground between different complexity levels. Future research could introduce a more granular

classification, helping to understand the nuanced impact of levelling systems on gameplay and player experience.

Second, application to different game genres could broaden the scope of this study. Our project integrates different levelling systems within a Roguelike-inspired framework, but future research could explore how different levelling systems interact with gameplay mechanics in RPGs, FPS games, or strategy games. Investigating genre-specific challenges and adaptations could provide valuable insights into how levelling systems influence player experience across a wider range of game designs.

Third, long-term player engagement analysis could offer a more comprehensive perspective. Our study captures short-term player feedback, but it remains unclear how different levelling system complexities affect player retention over extended play sessions. Future research could explore whether certain systems encourage long-term engagement more effectively than others, potentially offering guidance for developers in designing more compelling progression mechanics. To support such analysis, future studies could also make greater use of in-game metrics, enabling the game engine to collect more detailed data on player behavior and interactions.

Finally, the structure and reusability of the levelling systems developed in this project could be further explored. While our implementation successfully integrates multiple levelling systems into a unified framework, reusability and modularity remain key concerns. Future research could investigate how adaptable our framework is to different game engines or development environments and whether a more flexible, scalable approach could be implemented. This could lead to a standardized method for integrating multiple levelling systems, making it easier for developers to implement customizable progression mechanics in their games.

REFERENCES

ACM. 2018. ACM Code of Ethics and Professional Conduct. https://www.acm.org/code-of-ethics. Accessed: 20-Nov-2024.

Bethesda. 1994. The Elder Scrolls series. [Video game series].

Bethesda. 1997. Fallout series. [Video game].

BioWare. 2007. Mass Effect. [Video game].

Brackeys. 2024. How to make a Video Game - Godot Beginner Tutorial. https://www.youtube.com/watch?v=LOhfqjmasi0. Accessed: 15-Mar-2025.

British Computer Society (BCS). 2021. BCS Code of Conduct. https://www.bcs.org/more/code-of-conduct/. Accessed: 20-Nov-2024.

John Brooke. 1996. SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*, Patrick W. Jordan, Bruce Thomas, Bernard Weerdmeester, and Ian L. McClelland (Eds.). Taylor & Francis, London, 189–194.

CD Projekt Red. 2020. Cyberpunk 2077. [Video game].

Carnegie Mellon University Entertainment Technology Center. 2024. Panda 3D. https://www.panda3d.org. Accessed: 20-Nov-2024.

Monte Cook and Skip Williams. 2003. *Player's Handbook: Core Rulebook I V. 3.5.* Wizards of the Coast.

Eric Francisco. 2023. How Zombies Became Call of Duty's Biggest Mode. https://www.ign.com/articles/how-zombies-became-call-of-dutys-biggest-mode.

FromSoftware. 2011. Dark Souls. [Video game].

FromSoftware. 2022. Elden Ring. [Video game].

Epic Games. 2024a. Unreal Engine. https://www.unrealengine.com. Accessed: 20-Nov-2024.

YoYo Games. 2024b. GameMaker Studio. https://www.yoyogames.com. Accessed: 20-Nov-2024.

GDQuest. 2023. 3D TUTORIAL: Make Smooth 3D Movement in Godot 4. https://www.youtube.com/watch?v=JlgZtOFMdfc. Accessed: 15-Mar-2025.

Gearbox Software. 2019. Borderlands 3. [Video game].

James Paul Gee. 2009. Good video games and good learning. *University of Wisconsin–Madison. Recuperado* (2009).

Godot. 2024. Godot Engine: Free and Open Source Game Engine. https://godotengine.org.

Dennis Godtfredsen. 2016. Enhancing role-play in games with a redesigned leveling and skill system without class restrictions. (2016).

Irrational Games. 2013. BioShock: Infinite. [Video game].

Dominic Kao. 2020. The effects of juiciness in an action RPG. *Entertainment Computing* 34 (2020), 100359.

Yoon J Kim and Valerie J Shute. 2015. Opportunities and challenges in assessing and supporting creativity in video games. *Video games and creativity* (2015), 99–117.

Christoph Klimmt, Dorothée Hefner, and Peter Vorderer. 2009. The video game experience as "true" identification: A theory of enjoyable alterations of players' self-perception. *Communication theory* 19, 4 (2009), 351–373.

Raph Koster. 2013. *Theory of fun for game design.* " O'Reilly Media, Inc.".

Jon A Krosnick. 2018. Questionnaire design. *The Palgrave handbook of survey research* (2018), 439–455.

Larian Studios. 2023. Baldur's Gate 3. [Video game].

Legion Games. 2024. Zombie Level: Complete Godot 4 Cemetery Level with a Character Controller. https://github.com/LegionGames/ZombieLevel. Accessed: 20-Feb-2025.

Majikayo Games. 2024a. Godot 4 Sourcelike FPS Character Controller Tutorial | Smooth Movement, Bhop, Surf, And More. https://www.youtube.com/watch?v=ZJr2qUrzEqg. Accessed: 15-Mar-2025.

Majikayo Games. 2024b. Simple FPS Controller. https://github.com/majikayogames/SimpleFPSController. Accessed: 20-Feb-2025.

Maxis. 2014. The Sims 4. [Video game].

Mixamo. 2025. Mixamo Zombie Character Collection. https://www.mixamo.com/#/?page=1&query=zombie&type=Character. Accessed: 15-Mar-2025.
Media Molecule. 2008. LittleBigPlanet. [Video game].
Nintendo. 2023. The Legend of Zelda: Tears of the Kingdom. [Video game].
Obsidian. 2010. Fallout: New Vegas. [Video game].
Obsidian. 2019. The Outer Worlds. [Video game].
Jeffrey Parkin. 2020. The Outer Worlds guide. https://www.polygon.com/outer-worlds-guide-walkthrough/21280947/attributes-stats-skills-perks-flaws-build-character-creation.
Pav. 2020. Level systems and character growth in RPG games. https://pavcreations.com/level-systems-and-character-growth-in-rpg-games/.
Jon Peterson. 2012. *Playing at the world: A history of simulating wars, people and fantastic adventures, from chess to role-playing games*. Unreason Press San Diego, CA.
Reddit. 2005. Reddit - Dive into anything. https://www.reddit.com.
Richard M Ryan, C Scott Rigby, and Andrew Przybylski. 2006. The motivational pull of video games: A self-determination theory approach. *Motivation and emotion* 30 (2006), 344–360.
Dan Stapleton. 2019. The Outer Worlds Review. https://www.ign.com/articles/2019/10/22/the-outer-worlds-review.
Steam Community. 2023. Achievements for Baldur's Gate 3. https://steamcommunity.com/stats/1086940/achievements/. Accessed: 15-Mar-2025.
Unity Technologies. 2024. Unity Game Engine. https://unity.com. Accessed: 20-Nov-2024.
Treyarch. 2008. Call of Duty. [Video game series].
Jukka Vahlo and Veli-Matti Karhulahti. 2020. Challenge types in gaming validation of video game challenge inventory (CHA). *International Journal of Human-Computer Studies* 143 (2020), 102473.
Valve. 1998. Half-Life. [Video game].
Valve. 2011. Portal 2. [Video game].
Warhorse Studios. 2025. Kingdom Come: Deliverance II. https://store.steampowered.com/app/1771300/Kingdom_Come_Deliverance_II/. Accessed: 16-Mar-2025.
Zack Webster. 2020. The Outer Worlds Review. https://rpgamer.com/review/the-outer-worlds-review/.
Will Wright. 2006. Dream machines. *Wired Magazine* 14, 4 (2006), 110–112.
Peter Zackariasson and Timothy L Wilson. 2009. Creativity in the video game industry. *Creativity: Fostering, measuring and contexts* (2009).
José P Zagal and Roger Altizer. 2014. Examining'RPG elements': Systems of character progression.. In *FDG*.
ZA/UM. 2019. Disco Elysium. [Video game].

A PLES

A.1 Professional Issue

This project adheres to the ACM Code of Ethics and Professional Conduct [2018] and the BCS Code of Conduct [2021] in terms of professional factors. The project ensures compliance with applicable legislation, all data collected is used transparently and ethically, and any references to others' work or research are properly cited to avoid plagiarism or other forms of academic misconduct. The project avoids falsely claiming any level of competence, if we cannot achieve some goal during the experiment, we will never say we can do it. The project also strives to maintain high-quality work, professional tools have been employed, while professional evaluation models were selected for data analysis, and it follows a rigorous timeline, with weekly summaries of completed tasks and planning for upcoming work. We respect and accept diverse perspectives, regular meetings with the supervisor are conducted to exchange ideas, receive feedback, and identify innovative approaches to optimize the project. Moreover, we will continuously evaluate and report potential risks, every time a task is completed, backups are made to prevent data loss due to unforeseen circumstances.

A.2 Legal Issue

This project strictly adheres to legal standards and regulations. All personal information collected from participants during the experiment will be handled in strict compliance with GDPR. The data will be anonymized and securely stored, and any unnecessary information will be promptly destroyed after its use. Participants will be protected throughout the experiment. In the software development process, all code and resources will be used in full compliance with intellectual property laws, and the use of libraries and repositories will strictly adhere to their licensing terms. No pirated or unauthorized materials will be used. Furthermore, all software and tools used in the project will adhere to their respective End-User License Agreements (EULA). Additionally, the final software product will also comply with all relevant laws and regulations, ensuring its lawful use.

A.3 Ethical Issues

In this project, all participants will receive an information sheet detailing all aspects of the experiment and will sign an informed consent form. All personal information collected from participants will be securely stored and handled. Participants have the right to withdraw from the experiment at any time according to their wishes. During the questionnaire survey, respect for participants will be prioritized, and sensitive questions will be avoided. The software design will ensure the exclusion of discriminatory, unlawful, or harmful elements, and it will not

negatively affect the mental or physical well-being of others. This project has received approval from the ethics committee, ensuring compliance with ethical standards in all aspects.

A.4 Social Issues

This project strictly adheres to the ACM Code of Ethics and Professional Conduct and the BCS Code of Conduct in terms of social factors. The project ensures respect for others' privacy, we will not talk about any privacy-related issues during the experiment. During the project's execution, it is committed to ensuring that neither the project itself nor its processes will discriminate against or harm others for any reason. Accordingly the game will not contain any stereotypes or derogatory depictions of anyone, and it will not include any content that may cause physical or psychological harm to others. Furthermore, the project is dedicated to contributing to the fair and inclusive advancement of the IT field, we undertake this project to promote progress in the gaming industry.

B COMMENTS FROM REDDIT

Game	Opinion
The Outer Worlds	The perks themselves are generally fine, but none of them offer particularly exciting abilities, which means there's little sense of anticipation for the next level-up.
	Upgrading the initial assault rifle proves more effective than investing points in skills for science weapons.
	The perks for certain useful skills, such as bullet time, are too few.
	Frequent level-ups and excessive points make progression too fast, making abilities like conversation and hacking feel too easy.
	The levelling system is good for its detailed, old-school RPG feel, resembling a pen-and-paper game. However, the upgrades don't have a significant impact on gameplay, and unlike other RPGs, there's no exciting new ability to unlock. This leads to a sense of pure progression, but without engaging worlds to explore, the excitement starts to fade.
	Perks are boring and uninteresting, offering only basic upgrades like damage buffs, increased carry weight, and companion cooldown reductions.
	The game doesn't have a "best weapon," and players must continuously modify and upgrade their gear, which enhances the game's replayability. Hope other games can follow this approach.
	The levelling system is well-designed, with each attribute benefiting multiple skills across different areas. Even a pure melee build gains bonuses in other skills, providing more options for character development.

Game	Opinion
Baldur's Gate 3	The levelling system quite unfriendly for those players who know nothing about DnD.
	The levelling system is a bit boring, as it lacks a sense of meaningful character progression.
	The levelling system in *Baldur's Gate 3* is criticized for its level cap of 12, which halts progression too early and diminishes the sense of growth in the latter half of the game. This limitation leads to repetitive combat and reduces the motivation for exploration or completing side content, as progression is a key element of engagement.
	Larian has perfectly balanced the levelling system, making it more accessible than traditional DnD tabletop games while avoiding spells that would overly impact the gameplay experience. Even after reaching the level cap, the game's difficulty remains well-balanced, and players are encouraged to experiment with various class synergies and gear combinations to enhance their combat experience.
	The levelling system in *Baldur's Gate 3* with an early level cap allows players to enjoy max-level gameplay for much of the game. Unlike most RPGs where levelling peaks just before the final fight, this system emphasizes progression through diverse strategies, gear combinations, and combat approaches, enhancing replayability.
	The levelling system creates a balance between intrinsic and extrinsic motivations, shifting the player's focus from exploring solely for levelling up to engaging with the game out of genuine curiosity and interest. This design enhances the player's immersion in the world and story while encouraging the discovery of hidden details and experiences, adding greater depth and enjoyment to the game.
	The levelling system heavily relies on the player's understanding of DnD 5e rules, which can lead to issues if players are unfamiliar with the system. They may make suboptimal choices during levelling, potentially compromising their character's effectiveness later in the game.

Game	Opinion
Bioshock Infinite	The levelling system is designed to encourage players to prioritize upgrading their favorite abilities or traits rather than improving everything in one playthrough. This approach involves focusing on maxing one preferred trait first, then moving on to the next.

C QUESTIONNAIRE

Playability

1. How much confusion did you experience while playing this levelling system?

Very Little	Little	Moderate	Much	Very Much

2. The character's development path in the game is very clear.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

3. How difficult was it to get familiar with this levelling system?

Very Easy	Easy	Neutral	Difficult	Very Difficult

4. How likely are you to choose a similar levelling system in future games?

Strongly Dislike	Dislike	Neutral	like	Strongly like

5. How much pressure did you feel when choosing skills?

Very Little	Little	Moderate	Much	Very Much

6. The design of this levelling system made the game more enjoyable.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

Play Experience Feedback

7. The levelling system made me feel a strong sense of accomplishment as my character became stronger.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

8. I felt satisfied when defeating enemies using new perks.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

9. The levelling system made me feel that my choices and strategies had a direct positive impact on the game's progress.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

10. The levelling system made me feel I could explore different strategies or combat styles.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

11. The levelling system allowed me to freely allocate resources.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

12. The levelling system lets me try different directions in character growth without permanent restrictions.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

13. Playing this game made me lose track of time.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

14. I felt excited and eager to explore when unlocking new areas.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

15. I was very excited about unlocking new perks.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

Improving suggestion

16. Were there any features of the levelling system that felt confusing or overwhelming? Please share your suggestions for making it more intuitive or streamlined. (optional)

17. If you could change this levelling system to improve it, what would you like to do? Feel free to share any suggestions, whether it's adding new features, improving existing ones, or addressing other areas. (optional)

D CONSENT FORM

Consent Form: Take a Part in the Video Game Evaluation
Principal Investigator: Hu Erzhizhi
Contact: eh2020@hw.ac.uk
Objective of the experiment:
This experiment aims to evaluate how will the levelling system in a Role-playing game effect the playability of the game and how will the system influence player's overall game experience.

Procedure:

- Play 2 or 3 different versions of the game.
- Complete the questionnaire.
- Allocate approximately 45 minutes for the entire session.

Voluntary Participation:
Participation is entirely voluntary. You have the right to withdraw at any point without penalty or consequences.

Confidentiality and Data Use:
All collected data will be anonymized to ensure the protection of your privacy. The data will be used solely for academic purposes, such as publications and presentations, with no personally identifiable information disclosed.

Potential Benefits and Risks:
The direct benefit for you is you can play a game in two different versions, and totally free. The risks for this experiment are you might harm your eyes due to improper posture while playing games, or you could also lose track of time because you become too engrossed in the game.

Consent:
By signing below, you confirm that:

(1) You have read and understood this consent form.
(2) You voluntarily agree to participate in this experiment.
(3) You understand your right to withdraw at any time without consequences.
(4) You consent to the anonymized use of your data for academic purposes.

Name: **Signature:**

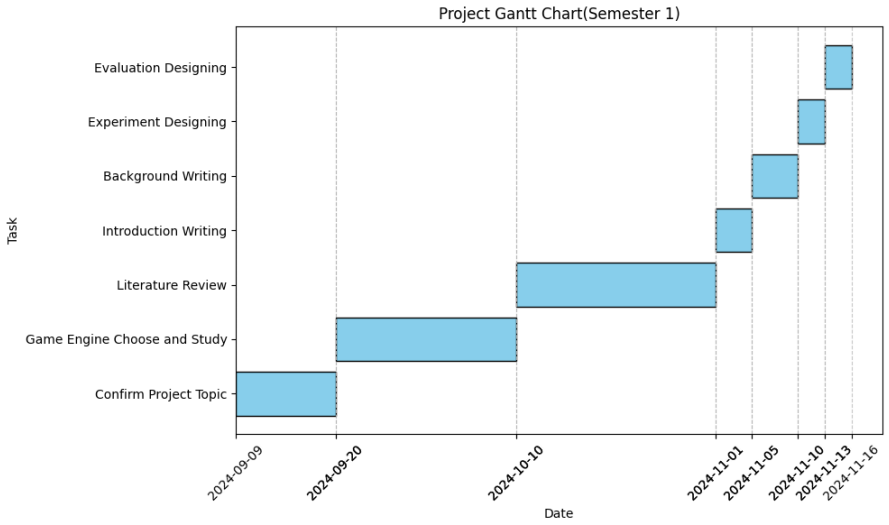
Date: **E-mail address:**

E PROJECT MANAGEMENT

E.1 Project Plan for Semester 1

In the table below, I've detailed my time plan for the first semester. I believe I spent too much time researching game engines, which left insufficient time for subsequent tasks. Additionally, as it was my first time conducting literature reviews, I struggled to efficiently determine the right direction for my search, which resulted in wasted time.

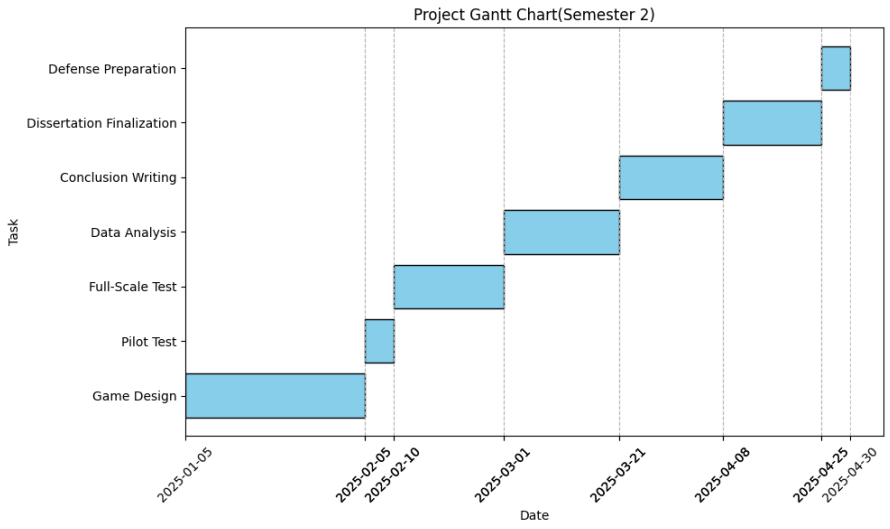
Name	Start Date	End Date	Duration
Confirm Project Topic	2024-9-09	2024-9-20	11 days
Game Engine Choose	2024-9-20	2024-9-24	4 days
Game Engine Study	2024-9-24	2024-10-10	14 days
Literature Review	2024-10-10	2024-11-01	21 days
Introduction Writing	2024-11-01	2024-11-05	4 days
Background Writing	2024-11-05	2024-11-10	5 days
Experiment Designing	2024-11-10	2024-11-13	3 days
Evaluation Designing	2024-11-13	2024-11-16	3 days
Deliverable 1 Deadline	2024-11-21	2024-11-21	-



E.2 Project Plan for Semester 2

The table below outlines my plan for the second semester. Initially, I will use the vacation period and the first few weeks with fewer classes to design the game. After that, I will conduct a pilot study with 1-2 individuals to test the survey, followed by a full-scale experiment. Considering the time needed to recruit participants and the variability in their schedules, I have allocated a generous 18 days for the experiment. Once the data is collected, I will proceed with the data analysis. After completing the analysis, I will begin working on the remaining sections of the thesis, allowing ample time for revisions and improvements after finishing this part.

Name	**Start Date**	**End Date**	**Duration**
Game Design	2025-1-5	2025-2-5	31 days
Pilot Test	2025-2-5	2025-2-10	5 days
Full-Scale Test	2025-2-10	2025-3-1	18 days
Data Analysis	2025-3-1	2025-3-21	20 days
Conclusion and Future Work Writing	2025-3-21	2025-4-8	17 days
Dissertation Revision and Finalization	2025-4-8	2025-4-25	17 days
Defense Preparation	2025-4-25	2025-4-30	5 days



E.3 Risk Analysis

The table below presents a risk analysis. In this table, I have compiled various potential risks that may arise during the experiment, assessed their likelihood, and further explored their potential impact. Finally, I proposed relevant mitigation strategies to address these risks.

Risk Description	**Probability**	**Impact**	**Mitigation Strategy**
Unable to find enough suitable participants	Possible	High – Data from participants is the most important part of this project.	Allocate sufficient time for recruitment. Provide incentive programs.
Hardware breaks or malfunction	Unlikely	High – Hardware is vital for software design, data analysis, and data storage.	Regularly back up hardware data, and be prepared to quickly obtain replacement hardware and retrieve backups to minimize disruption to the experiment as much as possible.
Data breach event of participant personal information	Unlikely	High – The disclosure of participants' personal information would constitute a serious violation of GDPR.	Immediately destroy any information that is no longer needed, and strictly safeguard participants' personal information.
The software used in the experiment encounters errors, crashes, or compatibility problems	Possible	High – Directly impacts the quality of data collection and may lead to unreliable experimental results.	Have alternative software options available in case the primary software fails or becomes incompatible.
The experimental results do not align with the hypothesis	Likely	Medium – May require additional time and resources for corrections and retesting.	Conduct pilot studies to identify potential issues before the full experiment, allowing adjustments to the methodology if needed.
The project may not be completed by the set deadline	Unlikely	High – Missing the deadline means low or no grades.	Create a comprehensive project plan with clear milestones and deadlines for each phase.

F MECHANICS IN THE GAME

Attribute	**Constitution**	Influences the character's health and endurance.	Complex Version	Each point affects **Endurance** and **Resilience** by ±5.
			Normal Version	Each point investment affects the character's sprint speed and sprint duration by ± 2.5% , and affects damage taken by ± 1.5%.
	Strength	Influences melee weapon capabilities.	Complex Version	Each point affects **Melee** and **Intimidation** by ±5.
			Normal Version	Each point investment affects melee damage by ±5% and affects critical hit chance for melee attacks by ± 2.5%.
	Perception	Influences ranged weapon capabilities.	Complex Version	Each point affects **Handguns** and **Longguns** by ±5.
			Normal Version	Each point investment affects ranged weapon damage by ±6%, and affects recoil, bullet spread and reload speed by ±3%.
Skill	**Endurance**	Each point investment affects the character's sprint speed and sprint duration by ±0.5%		
	Resilience	Each point investment affects damage taken by ±0.3%.		
	Melee	Each point investment affects melee damage by ±1%		
	Intimidation	Each point investment affects critical hit chance for melee attacks by ± 0.5%.		
	Handguns	Each point investment affects handguns damage by ±1.2% recoil, and affects bullet spread and reload speed by ±0.6%.		
	Longguns	Each point investment affects long guns damage by ±6% recoil, and affects bullet spread and reload speed by ±0.6%.		
Aptitude	**Firefighter**	Constitution+2, Strength+2, Perception-1		
	Assassin	Constitution-2, Strength-1, Perception+5		
	Soldier	Constitution+1, Strength+1, Perception+2		

Perk	**Deserter**	Greatly increases the character's movement speed in seconds after being attacked (Requirements: Constitution: 4, Endurance: 45 / Level: 3 / $5000)
	Pack Rat	Double the character's ammunition reserves and magazine capabilities (Requirements: Constitution: 4, Endurance: 45 / Level: 3 / $2500)
	Flak Jacket	Prevents the character from taking damage from explosions (Requirements: Constitution: 4, Resilience: 45 / Level: 9 / $2500)
	Qinggong	Enables double jumping and significantly enhances aerial maneuverability (Requirements: Constitution: 6, Endurance: 60 / Level: 7 / $7500)
	Bullet Time	Slow down time (Requirements: Perception: 6, Longguns: 60 / Level: 3 / $7500)
	Vampire	Restores a small amount of health after attacking an enemy in melee combat (Requirements: Strength: 6, Melee: 60 / Level: 7 / $7500)
	Tough Skin	Significantly reduces damage from enemy attacks (Requirements: Constitution: 9, Endurance: 100 / Level: 11 / $15000)
	Cowboy	Greatly improves weapon damage, accuracy while moving and reduces recoil (Requirements: Perception: 9, Handguns: 100 / Level: 11 / $15000)
	Die Hard	Grants 2 seconds of invincibility when taking a fatal hit, with a cooldown of 4 minutes (Requirements: Strength 9 Intimidation: 100 / Level: 11 / $15000)