BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

A 3D GAME IN UNITY WITH FOCUS ON A BLAST

3D HRA V UNITY SE ZAMĚŘENÍM NA TLAKOVOU VLNU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ŠTĚPÁN BÁRTA**
AUTOR PRÁCE

SUPERVISOR **Ing. ONDŘEJ KLÍMA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2025

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

164361

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Bárta Štěpán**
Programme:	Information Technology
Title:	**A 3D game in Unity with focus on a blast**
Category:	Computer Graphics
Academic year:	2024/25

Assignment:

1. Learn about 3D game development techniques from a first-person perspective in Unity.
2. Design a game inspired by a real-life scenario for an improvised shelter inside a building in crisis situations, focus especially on a blast scenario.
3. Implement the designed game using the provided models of public or private buildings.
4. Evaluate the results achieved and publish the game online.
5. Present the achieved results and create a demonstration video.

Literature:

• Gregory, Jason. Game engine architecture. crc Press, 2018. ISBN 1351974289, 9781351974288
• Unity Learn. Unity, https://learn.unity.com/

Requirements for the semestral defence:
Points 1, 2, and partially 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Klíma Ondřej, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2024
Submission deadline:	14.5.2025
Approval date:	12.11.2024

Abstract

This bachelor's thesis focuses on developing a 3D educational game designed to teach players how to behave during a bombing or a nuclear attack. In the game, the player completes tasks for which they receive points. These time-limited tasks represent real procedures that should be followed in an emergency. The game was developed using Unity game engine and the C# programming language, focusing on serious game design principles, interaction mechanics, and informative visual elements. The implementation includes a task system, inventory, cutscenes, and animations. This thesis demonstrates how interactive simulations can be an engaging and memorable tool for effectively educating the public in emergency preparedness.

Abstrakt

Tato bakalářská práce se zaměřuje na vývoj 3D výukové hry, která má hráče naučit, jak se chovat během bombardování nebo jaderného útoku. Ve hře hráč plní úkoly, za které získává body. Tyto časově omezené úkoly představují skutečné postupy, které by se měly dodržovat v případě mimořádné události. Hra byla vyvinuta pomocí herního enginu Unity a programovacího jazyka C# se zaměřením na principy designu seriózních her, mechaniky interakce a informativní vizuální prvky. Implementace zahrnuje systém úkolů, inventář, cutscény a animace. Tato práce ukazuje, jak mohou být interaktivní simulace poutavým a zapamatovatelným nástrojem pro efektivní vzdělávání veřejnosti v oblasti připravenosti na mimořádné události.

Keywords

Educational game, Unity, 3D game, Emergency preparedness, Pressure wave, Survival simulation, First-person gameplay, Game development

Klíčová slova

Edukativní hra, Unity, 3D hra, Krizová připravenost, Tlaková vlna, Simulace přežití, Hra z pohledu první osoby, Vývoj hry

Reference

BÁRTA, Štěpán. *A 3D game in Unity with focus on a blast.* Brno, 2025. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Ondřej Klíma, Ph.D.

Rozšířený abstrakt

Tato bakalářská práce se zabývá vývojem edukativní 3D hry, jejímž cílem je naučit hráče, jak se správně zachovat v případě bombardování nebo výbuchu atomové bomby. Díky své interaktivitě a schopnosti poutavě zprostředkovávat informace představují videohry vhodný nástroj pro vzdělávání široké veřejnosti. Práce se zaměřuje na návrh a realizaci vážné hry, která simuluje krizovou situaci a hráčům přístupnou formou předává důležité poznatky o postupech přežití.

Hra byla navržena a implementována v herním enginu Unity s využitím programovacího jazyka C#. Obsahuje systém úkolů s časovým omezením, jejichž plněním hráč získává body. Tyto úkoly vycházejí z reálně doporučených kroků, které je vhodné provést v případě mimořádné události. Herní mechaniky byly navrženy tak, aby podporovaly edukativní charakter hry. Její součástí jsou animované cutscény, interaktivní prostředí umožňující manipulaci s objekty a infrastrukturou, inventář s předměty důležitými pro přežití a výstavbu úkrytu a také postava NPC představující kamarádku, kterou je nutné během hry zachránit.

Vývoj probíhal ve spolupráci se zástupci Hasičského záchranného sboru, kteří poskytli cennou zpětnou vazbu k realistickému pojetí herních scénářů a obsahu. Díky jejich podnětům bylo možné zaměřit se na klíčové aspekty chování obyvatelstva při jaderném útoku nebo běžném vojenském ohrožení.

Výsledná hra obsahuje dva odlišné herní scénáře (levely). První scénář se odehrává v běžné obytné budově a trvá přibližně 10–20 minut v závislosti na zvolené obtížnosti. Úkolem hráče je během stanoveného času splnit co nejvíce z celkem jedenácti úkolů, mezi něž patří například sběr jídla, pití a léků, získávání materiálu na výstavbu úkrytu, hledání bezpečné místnosti, která je následně upravena pro lepší ochranu a přežití, například přidáním ventilace a podpěrného sloupu, vypnutí infrastruktury (vody, plynu, elektřiny) a záchrana kamarádky. Po vypršení časového limitu následuje cutscéna zachycující výbuch a příjezd hasičů, kteří hráče zachrání.

Druhý scénář simuluje okamžité ohrožení a trvá přibližně 90 až 150 sekund. Hráč nejprve sleduje 3 přilétající stíhačky směřujících k útoku na jeho dům a musí se co nejrychleji ukrýt v nejbezpečnější místnosti. Po nalezení úkrytu je spuštěna cutscéna, ve které každá stíhačka shodí bombu na budovu. Následuje exploze a zásah záchranných složek, které hráče zachrání. Oba scénáře kladou důraz na odlišné herní dovednosti a rozhodování, přičemž společně poskytují praktický návod na správné chování v krizových situacích.

Na základě uživatelského testování, do něhož se zapojilo 11 respondentů včetně profesionálního herního QA testera, byla hra dále upravena tak, aby lépe naplňovala svůj edukativní účel. Dotazník ukázal, že většina hráčů považovala úkoly za srozumitelné a úroveň obtížnosti za přiměřenou. Zároveň se však ukázalo, že původní časový limit byl příliš krátký, což vedlo ke zvýšení dostupného času pro dokončení prvního scénáře. Na základě připomínek byla také vylepšena fyzika pohybu postavy, která měla tendenci zadrhávat se o stěny. Celkově byla hra hodnocena pozitivně z hlediska zábavnosti, vizuální stránky i celkového dojmu. Většina hráčů rovněž uvedla, že jim hra poskytla nové informace a zlepšila jejich připravenost na krizové situace.

Projekt ukazuje, že i závažná témata lze účinně přiblížit prostřednictvím interaktivních herních prvků. Výsledná hra může sloužit jako podpůrný nástroj při vzdělávacích aktivitách zaměřených na krizovou připravenost, například při výuce na školách nebo jako součást interaktivních vzdělávacích programů pro veřejnost. Zároveň však může oslovit i širší publikum jako zábavná hra, která nenásilnou formou předává užitečné informace o přežití v krizových situacích.

A 3D game in Unity with focus on a blast

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Ondřej Klíma, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis. Language and stylistic editing was partially supported by artificial intelligence tools, specifically Grammarly and ChatGPT. In addition, ChatGPT was used as a programming assistant during the development of the game. These tools were used exclusively to support the author's own work. All content, code, and ideas presented in this thesis are the original work and responsibility of the author.

.....
Štěpán Bárta
May 13, 2025

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Ing. Ondřej Klíma, Ph.D., for his guidance, support, and valuable feedback throughout the development of this thesis. I would also like to thank Ing. Jakub Rak, Ph.D., for mediating communication with the fire and rescue services and for providing practical recommendations that helped shape the educational content of the game. Finally, I would like to thank my family and Matilde for their continuous encouragement and support throughout my studies.

Contents

Chapter 1

Introduction

Nowadays, almost everyone, regardless of age, has heard of video games, seen them, or played them. Thanks to their entertainment value, visual appeal, and interactivity, video games are capable of capturing the attention of a broad audience. This makes them not only a means of spending leisure time, but also a tool that can be used for learning and conveying important information.

This bachelor's thesis focuses on the development of an educational video game aimed at teaching players how to respond appropriately in the event of a nuclear attack or bombing. Given the growing global tensions and the increasing number of armed conflicts, the topic of civil protection is highly relevant. The goal of the game is to present this serious issue in a comprehensible way and, through an interactive experience, deliver information that could save lives in a crisis situation.

Serious games are used in fields such as healthcare, the military, and education, where they simulate real-life situations and support the learning process. They are effective because they actively engage players, making the learning experience more memorable. In the field of emergency preparedness, several games already exist, but few focus specifically on the threat of nuclear attacks or bombings. Games on this topic are mostly entertainment-focused and lack a significant educational component.

I chose this topic because I have long been interested in the issue of armed conflicts and their impact on civilian populations. At the same time, I have always wanted to create my own video game and experience the complete game development process. The opportunity to combine game development with a critical issue of civil protection particularly appealed to me.

Chapter 2 describes milestone video games, the process of game development and its phases, and provides an overview of the most commonly used game engines. Chapter 3 focuses on the design of the game, including its concept, gameplay structure, player tasks, and the basic user interface design. Chapter 4 deals with the implementation of the game in the Unity engine and describes the technical tools used. Chapter 5 presents the results of user testing and evaluates the feedback received. Chapter 6 summarises the thesis results, assesses the achievement of the stated goal, and suggests possible directions for further development.

Chapter 2

History and Development of Video Games

This chapter introduces some of the earliest video games that laid the foundation for the gaming industry, as well as several historically significant titles that had a substantial impact on the development of game design and game culture, as discussed in Section 2.1. It then continues with an overview of the game development process, describing its key stages and required roles in Section 2.2. The chapter concludes with a presentation of two of the most widely used game engines today—Unity and Unreal Engine—which help simplify and accelerate the creation of modern games, as detailed in Section 2.3.

2.1 Video Game History

This section presents several historically significant games that have had a major impact on the development of the games industry. It focuses on early video game titles, technological innovations, and games that have had a broad impact on the further development of game design and game culture [12].

Tennis for Two

One of the first video games to be rolled out was Tennis for Two [1], introduced in 1958 by nuclear physicist William Higinbotham, who advocated for nuclear disarmament while serving as the inaugural chair of the Federation of American Scientists. The game was created at Brookhaven National Laboratory, displayed on an oscilloscope screen (see Figure 2.1), and powered by an analog computer as its processing unit. During its exhibition, Tennis for Two drew large crowds, with visitors waiting in long lines for a chance to play this innovative electronic game.

Spacewar!

In 1962, Steve Russell and his team at MIT created Spacewar [3], one of the first widely recognized computer games. The game was developed on a PDP-1, an early interactive mini-computer donated to MIT by Digital Equipment Corporation (DEC). Writing the initial version took approximately 200 person-hours. Spacewar was designed for two players, allowing them to navigate and control spaceships in combat, maneuvering away from the sun's gravitational force while firing photon torpedoes at their opponent (see Figure 2.1).

The game quickly spread to research computers nationwide and became a widely played early computer game. Although Russell never profited from Spacewar, its influence was significant, inspiring future game developers, including Nolan Bushnell, who later founded Atari.

Figure 2.1: Tennis for Two[1] (left) and Spacewar![2] (right).

Pong

In 1972, Nolan Bushnell and Ted Dabney established Atari. They brought in engineer Allan Alcorn, initially assigning him a warmup project to develop an improved arcade version of a ping-pong-inspired game Bushnell had seen at a trade show. Within weeks, Alcorn designed Pong [4, 7], refining its mechanics to make the gameplay more engaging and adding features like a scoreboard. The prototype was installed at Andy Capp's Tavern in Sunnyvale, California (see Figure 2.2), where it quickly gained popularity, earning four times more than other arcade machines in the venue. Atari soon began mass-producing Pong, which sparked a wave of imitators and helped establish the video game industry. Reflecting on its unexpected success, Alcorn stated, "Just keeping the game simple and fun to play is the secret. I had no idea that it would have such an impact..."

Pac-Man

Pac-Man [14, 20] was released in Japan in 1980 and was designed by Toru Iwatani. In the same year, it was released in the USA. At this time, there was a market for violent shooter games, so Toru Iwatani thought he could make a game that would focus on food and would, therefore, attract many more people, including women. The inspiration for creating the main character was a pizza that was missing one slice. This main character has to eat colored dots in a maze, and there are also four ghosts that he has to avoid (see Figure 2.2). Pac-Man can also eat energy pellets that allow him to eat ghosts for a short period. The name Pac-Man comes from the Japanese slang paku paku, which describes the sound of munching. The game was a massive success in the United States, selling over 100,000 consoles. In 1981, approximately 250 million Pac-Man games were played in the

[1] Image taken from [1].
[2] Image taken from [3].

U.S. weekly. The game was so popular in the early 1980s that some U.S. cities, concerned about arcade culture, passed laws restricting or regulating video games.

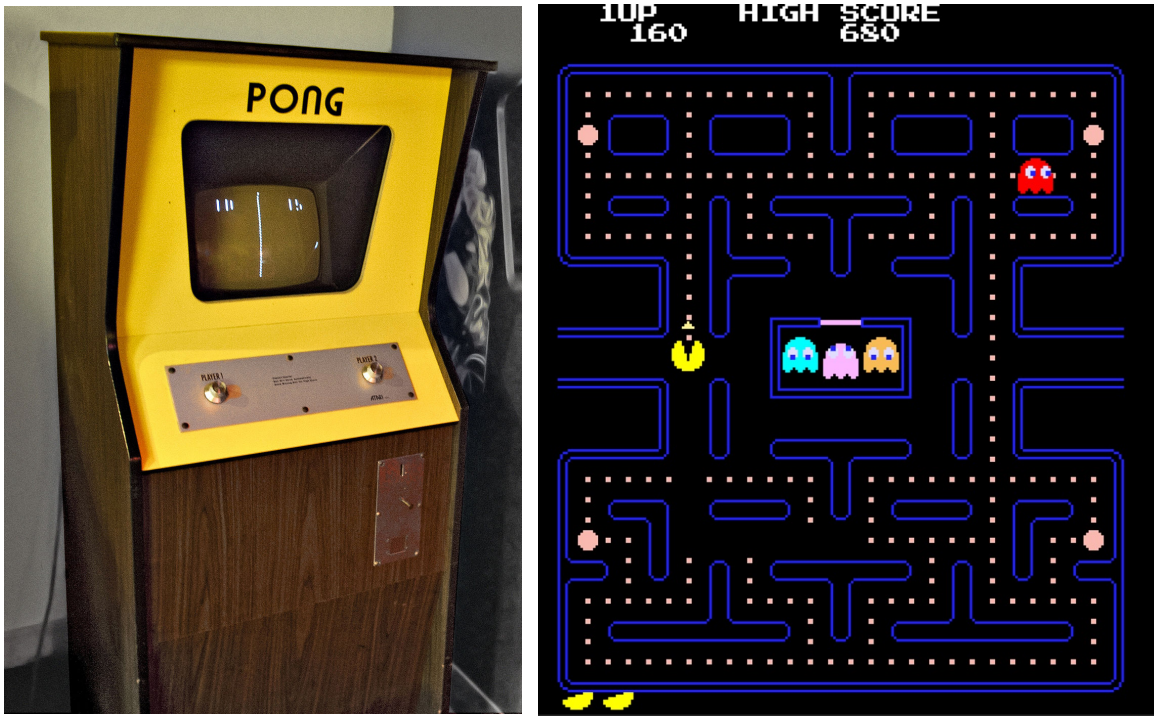

Figure 2.2: Pong[3] (left) and Pac-Man[4] (right).

Doom 1993

Doom [15, 2] was a groundbreaking first-person shooter game released in December 1993 by ID Software under the direction of John Carmack and John Romero. The game introduced groundbreaking graphics, multiplayer, game design, and distribution advances. It influenced almost every aspect of computer gaming. The game's development used a new game engine that separated artwork from game logic, which inspired many game designers in subsequent game development. In the game, users play as space marines fighting demonic forces on the Martian moon, Phobos (see Figure 2.3). Beyond entertainment, Doom had real-world applications. In 1997, the U.S. Marine Corps adapted the game into Marine Doom, converting its monsters into opposition forces to train soldiers in teamwork, tactics, and communication.

[3]Image taken from: https://commons.wikimedia.org/wiki/File:Atari_Pong_arcade_game_cabinet.jpg.
[4]Screenshot taken by the author during gameplay of Pac-Man.

Minecraft

Created by Markus Persson in 2009 and developed by Mojang Studios, Minecraft [22, 21] was officially released in 2011 and has since expanded to various platforms, including mobile devices and gaming consoles. Minecraft is an open-world game where players manipulate a 3D block-based environment. Known for its simple graphics and accessibility, it became especially popular among children and teenagers, with women making up nearly half its player base. By 2023, it was the second best-selling video game in history after Tetris. In 2014, Microsoft acquired Mojang for $2.5 billion. Two years later, it introduced Minecraft: Education Edition. Microsoft created this version to help in the classroom. It includes lessons on mathematics, coding, history, and geography. The aim is to encourage creativity in children, problem-solving, and teamwork. This educational version of the game includes building historical structures such as the Egyptian pyramids (see Figure 2.3), solving math puzzles, and space colonization. The educational version of the game has been incorporated into the curriculum by schools around the world, including the United States, Germany, France, Australia, and the United Kingdom.

Figure 2.3: Doom 1993[5] (left) and Minecraft[6] (right).

[5]Screenshot taken from: https://youtu.be/Q4GiCg_m8wA.
[6]Screenshot taken from: https://youtu.be/hyXciy8UM4k.

2.2 Game Development

Game development is a complicated process aiming to create the highest quality and highest-selling game while keeping costs low. Most developers employ a combination of Agile methodologies, particularly Scrum or its variations, which emphasize flexibility and iterative progress in development. In daily or weekly Scrum meetings led by a "Scrum Master", team members discuss progress and set future short-term goals called Sprints [8]. Game development involves working in a team to make software, for which creation is needed by many individuals from different professions, from art, programming, design, marketing, sales, and many others, to make a game that may not even be profitable for example in 2001, developers released over 3,000 games for PC. However, only 100 of those games made a profit [6]. The next part will walk through the stages of game development and their goals [18, 8].

Stages defined

Figure 2.4 illustrates each phase of development. On the left side, the stages have labels, while the right side details the activities performed in each stage. The right side forms a "V" shape and features a darker blue color, indicating that you can make more significant changes during the early stages without complicating things. That means that in the early stages of game development, more significant changes can be made without too many complications and with little financial consequence. However, making more significant changes in later stages can be dangerous and financially harmful. As development progresses, changes in ideas have to be made less and less significant [8].

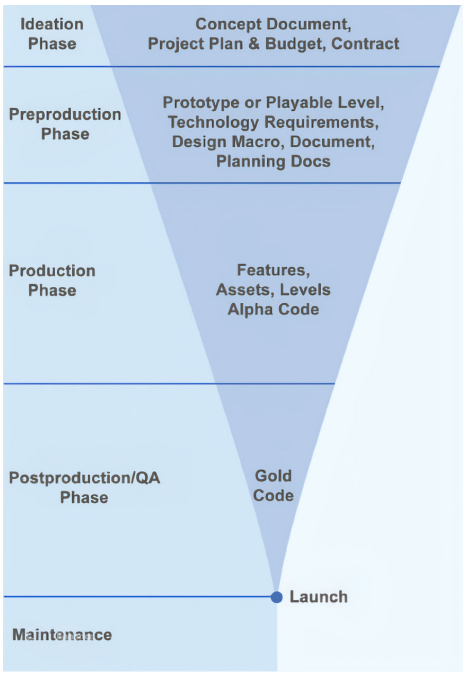

Figure 2.4: Stages of game development[7].

[7]Image taken from [8].

1. Pre-production

In the first phase of game development, it is important to specify what the game will be about, who the target audience is, how it will be published (on what platform, whether it will be sold or offered for free with in-game microtransactions), whether there are sufficient staff and resources, the estimated development timeline, and the likelihood of success. This phase usually takes up to 20% of the total production time. However, it is essential not to underestimate this part because changes in the later stages of development could delay and complicate the whole process. The **Game Design Document (GDD)** contains this information [18].

Game Design Document (GDD)

The development team uses it as the central reference point and continuously updates it throughout development. It is updated continuously to reflect newly identified gaps and changes during development. The Game Design Document (GDD) is a structured overview of all key aspects of the game, including its concept, genre, characters, and stories. It also focuses on game mechanics, gameplay, level design, and game world design. It often includes visual designs and funding or monetization strategies. Developers can use the GDD to present their game to investors and improve their chances of securing funding [18].

Prototyping

During this phase, developers rapidly create and test a game prototype to evaluate its functionality and user-friendliness. This process helps determine whether the game concept meets initial expectations or needs modification or a complete redesign. People who did not create the prototype should test it to uncover any ambiguities the developers might have missed [18].

2. Production

During the production phase, the team implements the visions and plans established in pre-production. This phase is typically the longest and most resource-intensive. As development progresses, it often becomes necessary to revise or scale back some of the initial ideas due to technical or time constraints.

Programmers write the game code, while artists produce 2D and 3D assets, including models, textures, and animations. Writers develop and refine dialogues, and sound designers create music and sound effects. Testers begin analyzing the game and devising testing methods to ensure functionality and playability. Throughout this phase, maintaining effective communication between team members is crucial, a responsibility usually overseen by the project manager. The project manager ensures that tasks remain on schedule and helps prevent development delays.

All the individual components—visual assets, animations, sound, dialogue, and others—are integrated into the game, gradually bringing it to life. According to Steve Ackrich, Head of Production at Activision-Blizzard, it is advisable to implement the game's initial segments last. As the team gains experience throughout development, the quality of these early gameplay sections benefits from improved understanding and refined processes, thereby increasing the likelihood that players will remain engaged.

The primary objective of the production phase is to reach the Alpha milestone, meaning the game is feature-complete. To meet this deadline, some features may need to be cut [8].

Development Milestones

Developers must reach several milestones during the development process [18].

- **First playable**: The game's first version has basic mechanics and a basic look. It gives the developers an idea of what the game will look like and how it will play.
- **Vertical slice**: A short section of the game that is fully functional and looks like the final product. Developers use it to present the game to investors and the public.
- **Pre-alpha**: In this phase, developers create most of the game, but it does not yet contain all significant features. In this phase, developers make big decisions about adding or removing new features to improve gameplay.
- **Alpha**: The game contains all significant features and is playable from start to finish. Some graphical and audio details may still be missing. Testers are testing the game and reporting bugs and errors.
- **Beta**: The game is complete and contains all game mechanics, graphics, and sounds. The game needs to be optimized.
- **Gold Master**: Final finished version ready for release to the public.

3. Post-production

After the game is released, the post-production phase begins. Some team members remain on the project to maintain the game by fixing bugs reported by players—issues that were not identified during testing. Additionally, the team may develop downloadable content (DLC) to keep players engaged. This phase also offers an opportunity for the team to reflect on the development process, assessing what worked well and what could be improved to ensure greater success in future projects [18].

2.3 Game engines

The concept of a "game engine" occurred in the 1990s. One of the first games that used a game engine was the popular Doom by ID Software, a first-person shooter game. Doom architecture was very well structured because it separated core software components like rules of play, 3D graphics rendering system, game worlds, art assets, collision detection system, and audio system. This division proved highly significant, giving rise to the "mod community" a collective of gamers and small studios that create new games by altering existing ones with the original developers' toolkits, by creating new arts, rules, game worlds, stories, weapons that they added to the original game with minimal changes to the game engine. At the end of the 1990s, developers created many games focusing on reuse, which marked the beginning of engine licensing. This development allowed people to earn money from making games and developing game engines. Today, game developers license game engines, which makes their development faster because they can use existing engines with many components that will be needed and do not need to develop their own [10]. Using a game engine to prototype and develop a game can save time. Choosing the wrong game engine can complicate development, especially if it lacks the necessary support for implementing features. Some game engines are open source, and access to engine code allows developers to modify them to suit the needs of their project better. One of the most popular game engines for developers is Unity and Unreal Engine. They enable even starting developers to create sophisticated games ranging from 2D to 3D. Additionally, these game engines allow developers to deploy their games on various platforms, including Mac OS, Windows, Android, Linux, iOS, web-based systems, and VR platforms. Figure 2.5 shows the top 250 popular Steam games by the engine and the total games released on Steam by the game engine. Also, they both have free versions for students and small teams of developers to start making their games without much money needed for software purchasing. Other significant easy-to-use game engines for beginners are Godot, GameMaker Studio 2, Construct 2, Buidbox, Gamesalad, and RPG Maker [8].

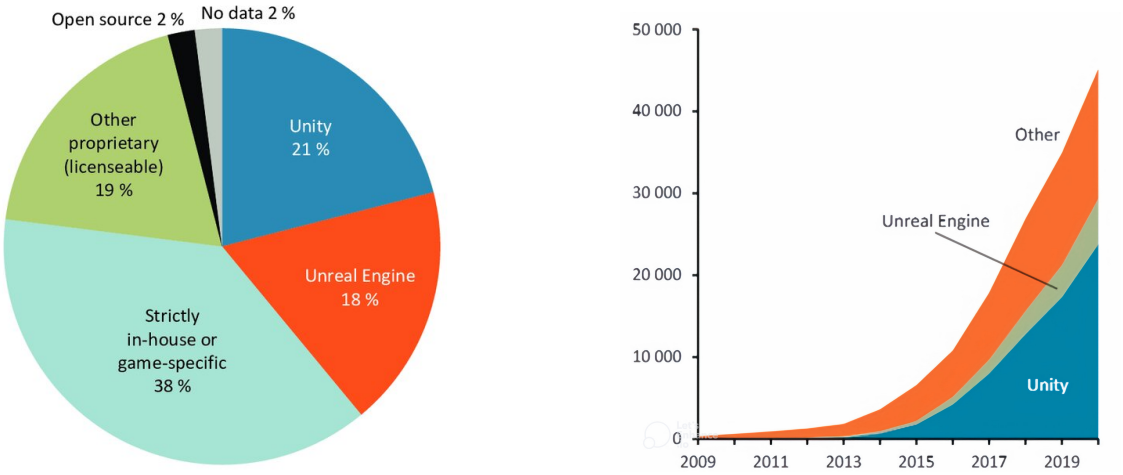

Figure 2.5: Top 250 most popular Steam games by engine (left) and Total games released on Steam by game engine (right)[8].

[8]Images taken from: [16].

Selecting the right engine is one of the most important decisions developers must make in the early stages of development. If developers do not consider which game engine to use and, after months of development, realise that the game engine they chose is not right for them or their game, it can result in a significant setback. This situation can quickly kill their motivation [19]. Over the last decades, programmers have developed many tools that game developers need for game development. Some of these tools are:

- Graphic engines
- Physics engines
- Fluid simulation
- Particle simulation
- Path finding
- Auto-rigging and animations

Most of these tools are what every game developer needs. Instead of being separated into different software, they are all combined into one game engine. All game engines offer similar set features with different implementations and enhancements. Nevertheless, that does not mean everything needed to develop a game is a game engine; the developer may need help from other external tools.

Choosing the Right Engine

Before starting development, it is important to choose the right game engine. This decision is crucial, and developers can easily make mistakes here. Each task requires specific tools that are used in different ways. Understanding the desired outcome and selecting the appropriate tool is essential. There are many game engines to use, and their base is very similar, but they are different from each other, and that difference can be an important thing that will decide if it is possible to reach the goal. When choosing a game engine, developers can make a few mistakes [17]:

1. A beginner picked a random game engine and started developing a game. After finishing the game, they realised they must pay a licensing fee to publish it.
2. Developers want to create a photorealistic 3D game, and after playing with the engine for a while, they realise that this engine does not support photorealistic 3D games.
3. Realising that the engine is incomplete and lacks many features during implementation.
4. After developing much of their game, they realised that the community of this engine was tiny. As a result, they could not find many resources.
5. Finding out that the game engine is not flexible enough to make generic ideas.

The following section presents an overview of the most widely used game engines.

Figure 2.6: The Unity logo[9].

Unity

The Unity game engine's creators first announced it at Apple's Worldwide Developers Conference in 2005 (see Figure 2.6), and since that time, it has had a significant impact on the video game industry. First, it has made significant efforts to expand the range of its target platforms. By 2015, it could export to 15 platforms with a unified code base [5]. Using Unity, developers can deploy their games on a variety of platforms, including [10]:

- Mobile operating systems (such as Apple iOS and Google Android),
- Gaming consoles (such as the Microsoft Xbox 360, Xbox One, Sony PlayStation 3, PlayStation 4, along with the Nintendo Wii and Wii U),
- Portable gaming devices (e.g., Nintendo Switch, PlayStation Vita),
- Personal computers (for instance, Microsoft Windows, Apple Macintosh, and Linux),
- Streaming devices (such as Android TV and tvOS),
- Virtual reality (VR) platforms (such as Oculus Rift, Steam VR, and Gear VR).

Unity is the most popular choice among indie developers. One reason for Unity's popularity is that it was among the first free engines available. Most of the good engines at the time of the Unity announcement were paid. Currently, there is a massive rise in indie games, and Unity has a significant impact on that. Unity benefits significantly from its large community, which has developed due to its vast popularity. Unity has an asset store, which is a place where people can download assets for free or for a fee, which can speed up their development. People can also make money by uploading their made assets into an asset store to help people with development [17]. Unity's main design objectives are simplifying development and enabling game deployment across multiple platforms. Unity provides a user-friendly integrated editor environment that allows developers to create and manage the assets and entities that make up their game world. They can quickly preview their game in action within the editor or directly on the target hardware. Unity also offers a comprehensive set of tools for analyzing and optimizing games on each target platform. The ability to effectively manage the balance between performance and quality for each deployment platform. In Unity, developers use the C# programming language to perform scripting. Unity supports multiplayer games and features a robust animation system [10]. Unity is a highly capable and powerful game engine. A wide range of people and studios can utilize Unity. Unity is a valuable game engine suitable for beginners and indie studios, as well as larger enterprise studios. It enables developers to create both small independent

[9]Image taken from: https://unity.com/.

games and large AAA titles produced by enterprise studios. Developing large AAA games is typically more suitable for Unreal Engine than Unity, as many features in Unity require configuration for desired quality. When developing games in Unity, purchasing assets from the Asset Store to speed up development is sometimes necessary. Unity offers excellent licensing options for developers. The platform does not require royalties from the games developed using its services. Developers fully own the games they create [17]. Currently, no fees are required if a game made with the Unity game engine does not exceed $200,000 within 12 months. If a game's revenue surpasses this threshold, the developer must upgrade from Unity Personal to Unity Pro, which costs $2200 per year for one developer; however, developers can retain all profits from game revenue [23].

Here is an overview of Unity's advantages and disadvantages [17].

Pros:

- A large community of support.
- Various rendering pipelines tailored for different platforms.
- No cost for the license.
- Complete ownership of the final game product.

Cons:

- Developing AAA games requires significant initial setup time.
- Need to purchase some tools from the asset store.

Unity has been used by numerous developers to create a wide range of successful games. The following section highlights several notable examples developed with this engine [17].

Ori and the Blind Forest

This game was very famous in 2015. It is a 2D action platformer highlighting why video games are considered the most excellent art form (see Figure 2.7). This game was created by an independent studio, Moon Studios, which was later purchased by Microsoft Studios. When creating a 2D game during that period (and still today), Unity had virtually no competitors in terms of the features it provided. The developers swiftly ported and released the game across various stores thanks to Unity's build support.

Cuphead

In 2017, the small independent game studio known as Studio MDHR developed Cuphead using Unity. Its distinctive aesthetics gained widespread recognition, which was made possible through Unity's support for sprite sheet animation (see Figure 2.7). The entire game utilised hand-drawn animations for its visuals.

HearthStone

This game comes from one of the industry leaders, Blizzard. They typically create their own game engine for their titles, but for this game, they used Unity. They recognised that the Unity game engine accelerated their process. They could complete the project without needing to employ many staff members. Only 15 people were involved in the creation of the game. Typically, their teams were two to three times bigger than that (see Figure 2.7).

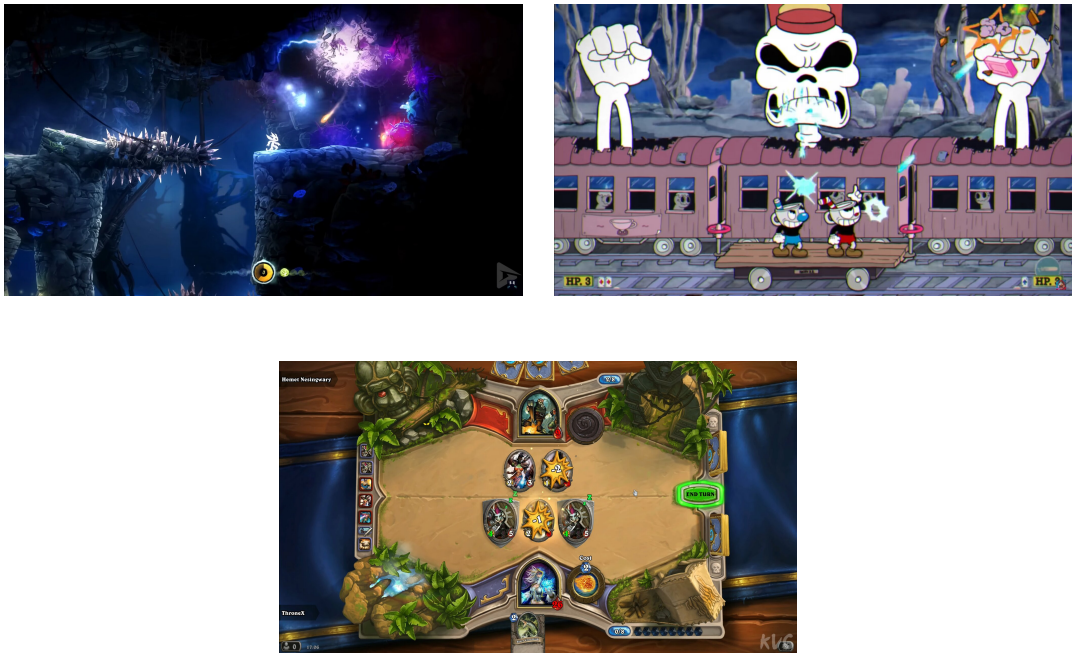

Figure 2.7: Ori and the Blind Forest[10] (top left), Cuphead[11] (top right), HearthStone[12] (bottom).

[14]Screenshot taken from: https://youtu.be/ey4Dk0dZD0o.
[15]Screenshot taken from: https://youtu.be/lV5Ailg7tf0.
[16]Screenshot taken from: https://youtu.be/hUi0eFuTi-g.

Figure 2.8: The Unreal Engine logo.[13]

Unreal Engine

One of the most well-known game engines, developed by Epic Games, made its mark in the FPS genre in 1998 with the iconic game Unreal. Since then, the Unreal Engine (see Figure 2.8) has become a major competitor to Quake technology in the FPS space, a family of engines made by ID Software, a company that created the legendary Doom in 1993 [10]. Developers created the Unreal Engine using the C++ programming language. Unreal Engine is open source and is known for its high speed and excellent portability thanks to the chosen programming language [9]. Many people regard C++ as one of the most powerful and user-friendly languages for game development. Although Unity uses C#, which is more user-friendly, it lacks crucial memory management control for game development. Choosing C++ over C# as the primary programming language for Unreal Engine is a logical decision. Unreal Engine should be the top choice for developers aiming to create a high-budget AAA title. It provides an out-of-the-box solution for those looking to achieve photorealistic graphics. With all the essential features included, developers can create a game from scratch without purchasing additional third-party libraries to support the development process. Similar to other leading game engines, it offers the ability to develop for various platforms such as mobile, web, desktop, PlayStation, XBOX, and Nintendo. It is important to note that, unlike other game engines, this one is designed specifically for high-end mobile devices. This choice may not be ideal for developers aiming to reach a broader audience. Additionally, starting development with Unreal Engine requires decent hardware equipped with a capable GPU. This requirement can lead to significant expenses, especially for educational purposes [17]. Unreal Engine 4 version introduced innovative designated visual scripting called Blueprint. These Blueprints feature a node-based interface that allows designers to create gameplay components directly within the editor. According to the documentation, this method is designed to empower designers to prototype and explore concepts that are usually limited to programmers. This tool is exceptionally powerful, enabling the development of entire games using it alone [5]. The Unreal Engine offers many features and integrated, user-friendly tools. The Unreal Engine has flaws, and many developers adjust it differently to ensure their game runs efficiently on specific hardware. Unreal is a powerful tool for prototyping and a comprehensive platform for commercial game development, capable of creating nearly any 3D game from first-person and third-person perspectives and games across various other genres [10]. Licensing can be complex. Developers do not need to pay fees if they create a free game. However, if they intend to publish their game, they must pay a royalty fee of 5% if their gross revenue exceeds a specific threshold each quarter. The charges depend on the product's success and may become unacceptable later [17].

[13]Image taken from: https://www.unrealengine.com/en-US.

Here is a summary of the pros and cons of Unreal Engine [17].

Pros:

- Projects that are free come with no cost at all.
- A licensing model based on royalties is available for games that generate revenue.
- Cutting-edge graphics that are ready to use are provided.
- Every imaginable tool is included as part of the package.
- Advanced visual scripting tools are available.

Cons:

- Does not accommodate low-end devices.
- The community is smaller compared to that of Unity.
- Only 64-bit development is supported.
- Development requires costly hardware.

Various developers have created numerous exciting games in different genres using Unreal Engine, including Rime by Tequila Works, Genesis: Alpha One by Radiation Blue, A Way Out from Hazelight Studios, and Crackdown 3 from Microsoft Studios [10]. One of the most notable and influential games developed using Unreal Engine is Fortnite, which has had a significant impact on the gaming industry.

Fortnite

Despite Unreal Engine's reputation for stunning photorealistic visuals, the team decided to create a low-poly battle royale game. This was primarily to demonstrate that Unreal Engine can also be utilized for low-poly games [17]. Fortnite (see Figure 2.9) has over 650 million registered players and over 110 million monthly active users. The game has generated over $40 billion in revenue, with more than $20 billion coming in just the year 2022. It is particularly popular among younger players, with 62.7% of Fortnite users aged between 18 and 24 [13].

Final Fantasy VII Remake

In 2020, many developers remade games, but none were as highly anticipated as Final Fantasy VII. The original version debuted on the PlayStation, marking the beginning of a new era. When Square Enix opted to recreate the game for PS4 with a fresh approach, they selected Unreal Engine 4 to deliver breathtaking graphics (see Figure 2.9). Their efforts truly captivated millions once again. In 2020, they left fans speechless, just as they had done back in 1997 with the original PlayStation release [17].

Figure 2.9: Fortnite[14] (left) and Final Fantasy VII Remake[15] (right).

[18]Screenshot taken from: https://youtu.be/hAlIsXT_MkQ.
[19]Screenshot taken from: https://youtu.be/KxGQ7DHDtTQ.

Chapter 3

Game Design

This chapter outlines the design of the educational game, focusing on its concept, structure, and key gameplay elements. It begins by presenting the purpose and goals of the game, followed by a detailed description of both playable levels. The first level simulates a scenario with enough time to prepare for an imminent explosion, allowing players to gather supplies and construct a shelter. The second level represents an urgent situation that requires quick decision-making under time pressure. The chapter also covers major gameplay features, including the task system, inventory and hotbar mechanics, and interaction with non-playable characters. All design decisions are based on real-world recommendations, some of which were consulted with professionals from the Fire and Rescue Service of the Czech Republic.

3.1 Game Concept

This project aims to develop an educational game that would teach players what to do in a crisis, such as an atomic bomb explosion or a bombing. The game would be intended for use by rescue service firefighters to help teach school children about safety procedures. They could visit schools to educate kids on what to do in emergencies. Then, the children could play this game to test what they have learned or even while they are learning, aiming to add entertainment value to the education process. The game would not be designed solely for children. It could be played by anyone who wants to playfully better prepare for unexpected military threats and thus increase the chance of survival in real life or even play this game just for fun. The game is being developed in collaboration with firefighters, who provided input on what features should be included. The game would be designed as a first-person 3D game to resemble the real world as closely as possible. It would be developed in the Unity game engine due to its considerable community support, an asset store that accelerates extensive game development, and its beginner-friendliness. The game would include two levels, each simulating different crisis scenarios. Both levels would ideally be developed in partnership with experienced firefighters, who could provide guidance throughout the development process. As a result, the game would include genuine recommendations and best practices for crisis situations. Collaborating with professionals ensures that the game's content is accurate and aligned with current safety protocols. Additionally, the level design would consider information from official resources, including the website of the Fire and Rescue Service of the Czech Republic, particularly the section focused on population sheltering [11].

3.2 Level 1—Improvised Shelter

The first level of the game is designed to simulate a scenario in which the player has sufficient time to prepare for an impending crisis. The gameplay, objectives, and supporting systems are described in the following subsections.

Story and Objective

This level would simulate a scenario where the threat is known well in advance, giving the player time to prepare for the attack. The goal would be to create a safe improvised shelter within an ordinary house (see Figure 3.1) and take all necessary steps to increase the chances of survival. During this time, the player would gather essential supplies for survival, such as food, water, and pills, as well as supplies for building a safe shelter, such as a pickaxe, bricks, wood, and much more. In addition to building the shelter, the player would also need to turn off critical infrastructure, including water, electricity, and gas, to prevent secondary dangers caused by potential system failures or damage from the blast. Finally, the player would have to assist a friend by guiding her to the shelter, ensuring both remain safe during the impending crisis. In this scenario, the bomb would eventually hit and explode after the player has done everything possible to survive. Following the explosion, firefighters would arrive to rescue the player and their friend from the shelter.

Gameplay Flow

This level would simulate a scenario where the player has approximately 20 minutes to prepare for a bomb attack. In reality, such preparation would take significantly longer. At the beginning of the level, the player would find themselves in front of a house situated in a natural setting, and immediately, a timer would start counting down, indicating how much time is left before the bomb detonates. During this time, the player would need to complete as many game tasks as possible. There would be a total of 11 visible tasks in the game, each accompanied by a hint that would guide the player on how to complete it. For every task completed, the player would earn points. Completed tasks would be marked as finished, and the points earned for each task would be displayed. The game would also include an inventory where collected items are stored and a hotbar where the player can place items from the inventory for quick access. The hotbar would be used for items that the player intends to utilize. For example, if the player wanted to dig through a wall, they would need to find a pickaxe, pick it up, add it to the hotbar, and then use it to dig. The player's progress would be determined by the tasks they complete and the points they earn. The total score, the hotbar, and the timer would all be visible throughout the game. When sprinting, an energy indicator would be displayed; once this energy runs out, the player would no longer be able to sprint. When the time runs out, a bird's-eye view scene will showcase a large explosion with flames spreading towards the player's hiding place. The screen would fade to black and then brighten again, revealing firefighters driving through the flames toward the house to rescue the player. Following this, the total score and the list of all tasks from the game would be displayed, allowing the player to see which tasks were completed and which ones were missed.

Figure 3.1: This 3D house model serves as the central location for all gameplay on the first level, both inside and in the surrounding area. The model was provided to the author for the purposes of this project. It was selected because it contains a basement with multiple rooms, and its relatively small size prevents the player from getting lost while exploring.

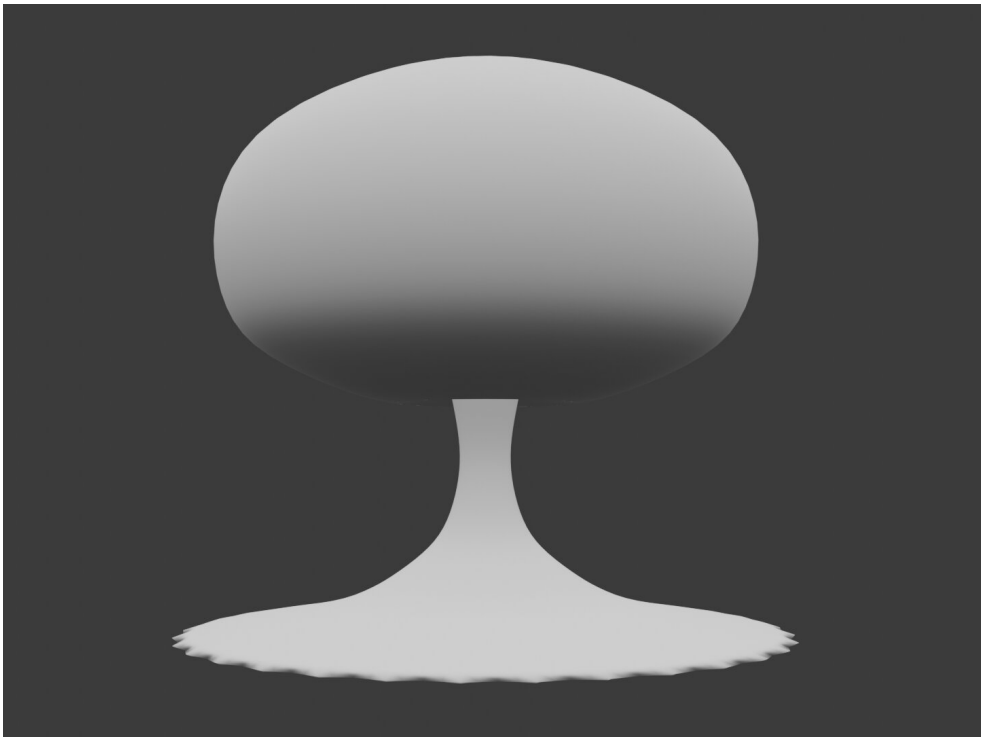

Figure 3.2: A 3D explosion model created by the author using Blender. This model would later be used in the game implementation, where it would be animated to rise upward after a bomb hits the ground.

Inventory

The inventory would function in the game as a backpack where players could store collected items. The item collection process would be straightforward: players would have to look directly at an object they could pick up. When targeting such an item, an on-screen prompt would appear: "Pick up [item name]". Players could then press the designated button to pick up the item, which would be added to their inventory. The inventory would be designed to accommodate many items, featuring 15 slots. If players collected multiple items of the same type, those items would automatically stack in a single slot, displaying the total quantity as a counter. Players could open the inventory at any time to view the collected items. From there, they could transfer selected items to a hotbar used for item usage. For example, after picking up a pickaxe, the player could find it in the inventory, move it to the hotbar, and then use it in the game.

Hotbar

Unlike the inventory, the hotbar would always be visible during gameplay, enhancing quick access to essential items. It would have six slots, but only one slot could be active at a time. The active slot would indicate which item is currently selected and available for the player to use. Players would be able to transfer items from the inventory to the hotbar by either dragging and dropping them or simply clicking on them. When clicked, the item would automatically occupy the first available slot in the hotbar. Similarly, items could be moved back to the inventory from the hotbar using the same methods. This system would enable players to efficiently manage and utilize important tools or supplies during critical moments in the game.

Game Tasks

As illustrated in Figure 3.3, players could view the game tasks by pressing the TAB key, which would open both the inventory and the task list. The inventory would be displayed on the right side of the screen, while the game tasks would be shown on the left. Players would also have the option to turn off the task list in the game settings. This feature would be added based on feedback from firefighters, who requested a game version without task hints. Such a version could be used during school visits, where firefighters would first explain the correct procedures for emergencies and then allow the children to play the game without in-game guidance—relying only on what they had learned during the session. By default, however, the game tasks would be enabled. This would be because an average player without professional instructions would likely not know what to do or how to act in crisis scenarios. The tasks would represent actions that should be taken during an emergency, and players would earn points for completing them. They would be designed to be both educational and entertaining. The number of points a player earns during the game would not be important because, in the end, they would always "win". In the final scene, firefighters would come to rescue both the player and their friend. This design choice would ensure that school children feel encouraged to replay the game and improve their responses to crises. Additionally, it would help create a more peaceful atmosphere in the game. A negative ending—such as a character dying due to not earning enough points—could have a very detrimental emotional impact on children. Through these tasks, players could learn how to respond to dangerous situations in a safe, game-based environment where no real harm can occur. The game would have 11 main tasks, each comprising several subtasks

that players would need to complete. Players would earn points for each subtask, offering more specific guidance on accomplishing the main task. While most subtasks could be completed in any order, a few exceptions would exist. For instance, tasks such as creating a ventilation system in a shelter, building a wooden support column in a shelter, and finding and rescuing a friend could only be undertaken after the player had finished locating a safe hiding place (see Figure 3.4). At the beginning of the game, each task would start with a score of 0 points. Once a task was completed, the player would receive feedback showing the points earned and a green checkmark indicating successful completion.

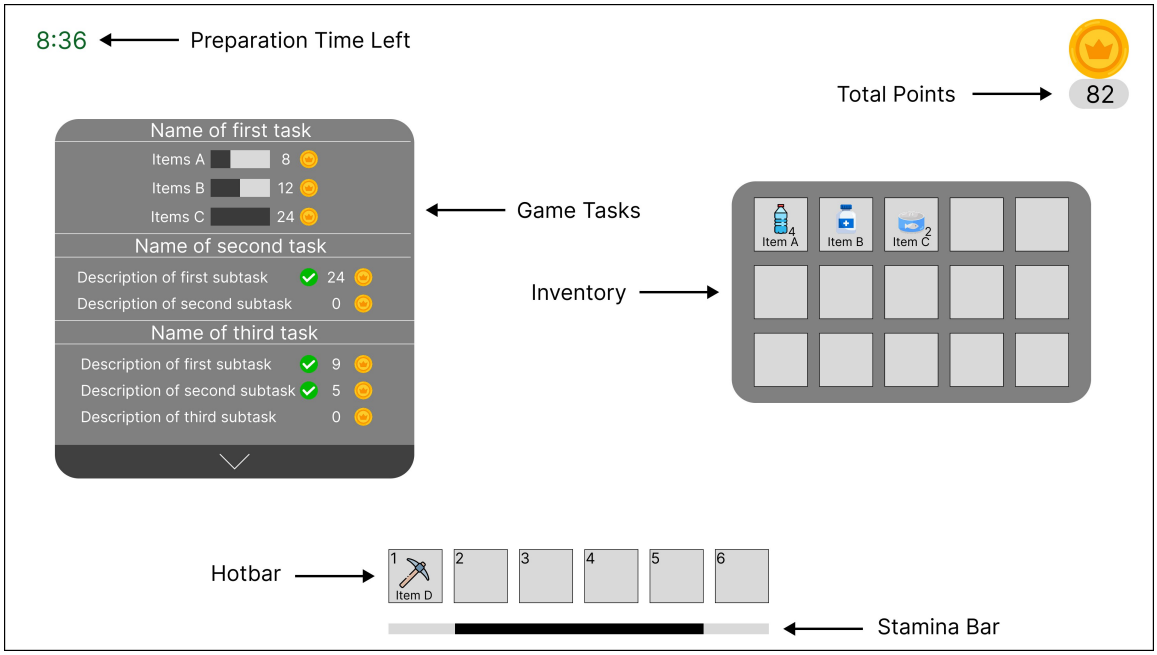

Figure 3.3: The user interface as it would appear during gameplay. The inventory and game tasks panels can be toggled using the TAB key. All other elements remain visible throughout the game, except for the stamina bar, which automatically hides once it is fully replenished after the player stops sprinting.

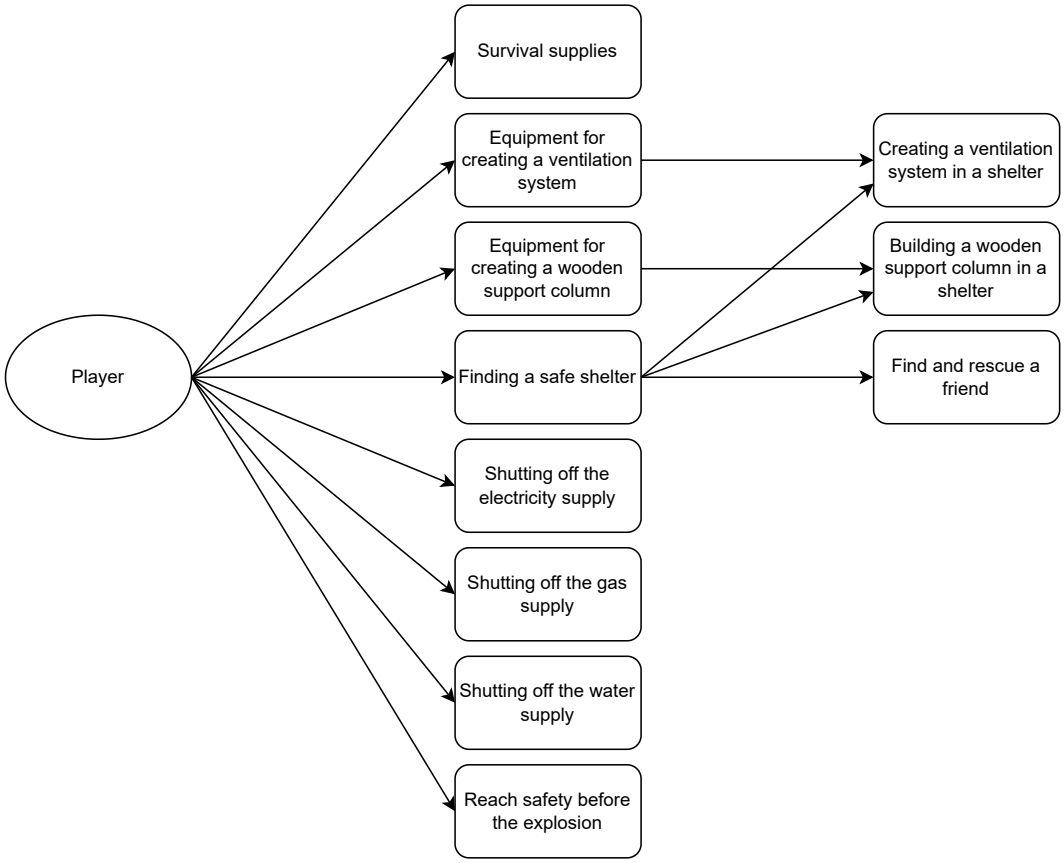

Figure 3.4: This diagram outlines the tasks that could be completed in the game. Most tasks may be completed in any order. However, three specific tasks—creating a ventilation system in a shelter, building a wooden support column in a shelter, and finding and rescuing a friend—can only be completed after the player has found a safe shelter. Additionally, creating a ventilation system and building a wooden support column both require the player to obtain the appropriate equipment before they can be executed.

Survival supplies

In this task, the player would need to collect essential supplies for surviving a crisis—food, water, and medicine. These items would be scattered both inside the house and in the surrounding environment. The player would be required to find and store the following items in their inventory: 24 food items (1 point each) 12 water items (2 points each) 3 medical items (8 points each) The total number of points that could be earned from this task would be 72 points. The Survival Supplies task would be designed to be completed progressively as the player explores the environment and works on other tasks. The supplies would be intentionally placed in various locations to be discovered naturally during gameplay. A key feature of this task would be the randomized placement of the supplies each time the game is started. As a result, players would not be able to rely on memorizing item locations from previous playthroughs. While some items might occasionally appear in similar spots, their positions would generally be unpredictable, significantly enhancing the game's replayability and keeping the experience fresh with each new attempt. Players would need to pick up the required items and store them in their inventory to complete the task. Points would be awarded instantly upon collecting each item. There would be no need to use or process the items in any specific way—collecting them would be sufficient for progress.

Equipment for creating a ventilation system

In this task, the player would need to collect the equipment required to build a ventilation system inside the safe room. The items that would need to be collected would include a pickaxe, ventilation units, and bricks. The pickaxe could be found in the basement, but its exact position would change with each new game, making its discovery less predictable. In contrast, the ventilation units and bricks would always be located in front of the house and remain in the same place in every playthrough. To complete this task, the player would need to collect: 1 pickaxe (10 points) 4 ventilation units (2 points each) 5 bricks (2 points each) The total number of points that could be earned from this task would be 28 points. These items would later be used in the ventilation system building process, which would become available only after the player had completed the Finding a Safe Shelter task.

Equipment for creating a wooden support column

In this task, the player would need to collect the necessary equipment to build a wooden support column inside the safe room. The required items would include a hammer, wood, and nails. All these items would be in the same place in every playthrough—on a wooden table behind the fence next to a campfire. Their positions would be fixed, so players could reliably find them there each time they play. To complete this task, the player would need to collect: 1 hammer (10 points) 3 pieces of wood (3 points each) 3 nails (3 points each) The total number of points that could be earned from this task would be 28 points. These items would later be used in the task Building a Wooden Support Column in a Shelter, which would become available after the player had found a safe hiding place. The wooden support column would reinforce the shelter structure and increase the chances of survival during the explosion.

Finding a safe shelter

Completing this task would unlock three additional tasks that the player could then complete, as illustrated in Figure 3.4. The goal of this task would be to choose a suitable room for survival, which the player would later upgrade in subsequent tasks to make it as safe as possible. The first step would be to locate and pick up a sleeping bag placed outside in front of the house near the campfire, next to the player's friend. For finding and picking up the sleeping bag, the player would be awarded 9 points. After collecting the sleeping bag, the player would need to place it in a safe room inside the house. Placing the sleeping bag in a specific room would signify that the player had selected that room as their safe shelter, where they intended to survive and later improve its conditions. Successfully placing the sleeping bag in one of the valid safe rooms would grant the player an additional 30 points. Although the house would contain many rooms, the sleeping bag could only be placed in a limited number of them. Six designated rooms in the basement are suitable for survival. These basement rooms would be chosen because underground areas offer the best protection during a bomb explosion. The player would need to place the sleeping bag in one of these approved basement rooms to complete the task. If the player attempted to place the sleeping bag in a room unsuitable for survival, an on-screen message would appear, explaining why that specific room was unsafe. Each unsuitable room in the house would have a custom explanation text assigned to it, helping the player understand the logic behind the decision and guiding them to make a better choice.

Creating a Ventilation System in the Shelter

This task would only become available after completing two prerequisites: Equipment for Creating a Ventilation System and Finding a Safe Shelter. To proceed, the player would need to have a pickaxe, four ventilation units, and five bricks stored in their inventory. Additionally, a valid safe room would need to be selected and marked by placing a sleeping bag inside it. Each safe room would feature either a window or a specially coloured section of the wall, which would visually distinguish it from the rest of the room. Once the player had completed the Finding a Safe Shelter task, this wall or window would become interactive. When the player looked at it, a prompt would appear: "You need a pickaxe in your hand to break this". If the pickaxe was placed in the hotbar and selected as the active item by pressing the corresponding key (1–6), the player could initiate the action by pressing the E key or the left mouse button. A progress bar would appear, indicating the ongoing action of breaking the wall or window. Once the progress bar was filled, the section would break open, creating an opening in the shelter. Next, the player would see a prompt when looking at the newly created hole: "You need a ventilation unit in your hand to install ventilation". The player could begin construction after selecting a ventilation unit from the hotbar. The building process would again display a progress bar, taking a total of four seconds to complete, with one-quarter of the ventilation structure appearing each second. This visual feedback would enhance immersion and communicate the ongoing construction clearly. After installing the ventilation system, the player would need to reinforce the surrounding area with bricks. Looking at the unfilled space next to the ventilation unit would trigger a message: "You need bricks in your hand to build the wall". After selecting bricks from the hotbar, the player would see a new prompt: "Build the wall". Initiating this action would display a five-second progress bar, during which the wall would be constructed in visible stages, piece by piece. Completing the task would reward the player with a total of 21 points: 7 points for breaking the wall or window, 7 points for installing the

ventilation system, and 7 points for completing the wall around the ventilation. This task would emphasize tool usage and realistic step-by-step construction, helping players better understand the practical aspects of shelter ventilation in emergencies.

Building a wooden support column in a shelter

This task would become available only after completing the Equipment for Creating a Wooden Support Column and Finding a Safe Shelter. To proceed, the player would need to have a hammer, three wooden planks, and three nails in their inventory. Additionally, a valid shelter room would need to be selected and marked by placing a sleeping bag inside it. Once the sleeping bag was placed, a wooden beam would appear on the safe room's ceiling, indicating where the support column should be built. This beam would become interactive. When the player looked at it, a prompt would appear: "You need wood in your hand to start building". After selecting wood from the hotbar, a new prompt—"Place wood"—would be displayed. Upon initiating the action, a progress bar would appear and fill over one second to simulate the placement of the wooden plank. After this step, the player would be prompted: "You need nails in your hand to continue building". Once the nails were selected, the prompt would change to "Insert nails", a one-second progress bar would appear to visualize the nail insertion process. The player would then see the message: "You need a hammer in your hand to continue building". After selecting the hammer from the hotbar, the player would receive the final prompt in the sequence: "Use a hammer". Pressing the interaction button would then initiate another progress bar representing the action of hammering the structure, with visual feedback indicating progress. Completing this step would result in the construction of the first third of the support column. This sequence—placing wood, inserting nails, and using the hammer—would need to be repeated twice. Each repetition would add another segment to the column. The support column would be fully built upon completing the final cycle, and the task would be marked as finished. This column would strengthen the shelter's ceiling, increasing its structural integrity and improving the chances of surviving the explosion. The player could earn a total of 11 points for completing this task: 1 point for identifying the initial wooden beam in the ceiling, 3 points for completing the first third of the column using wood, nails, and a hammer, and 7 points for constructing the full support column.

Find and rescue a friend

The objective of this task would be to locate and rescue the player's friend. The friend would lie next to the campfire in front of the house throughout the game. If the player approached her before completing the Finding a Safe Shelter task, she would respond with a message such as, "You need to find a safe shelter first", and no further interaction would be possible at that time. However, once the player had completed the Finding a Safe Shelter task—identifying and marking a suitable room for survival—they would unlock a new dialogue option when returning to the friend. The player would be able to inform her that a safe place had been secured and offer to lead her there. Upon doing so, the friend would get up and begin following the player. This action would reward the player with 7 points. The player would also be able to instruct the friend to either wait at her current location or resume following. To complete the task, the player would need to guide the friend to the selected safe room inside the house. Once inside, the player could instruct her to stay. Upon doing so, the friend would react with a joyful animation or dialogue line, and she would remain in the shelter for the rest of the game. Successfully bringing the friend

to the shelter and instructing her to stay would grant the player an additional 7 points, bringing the total reward for the task to 14 points. This task would emphasize individual survival and the importance of helping others during emergencies.

Shutting off the electricity supply

The objective of this task would be to disconnect the electricity in the house before the bomb explodes. In real-life emergency situations, turning off the electrical supply is an essential step in preventing secondary hazards, such as fires caused by damaged wiring. This task would be designed to educate players about the importance of securing household infrastructure during a crisis. To complete the task, the player would first need to find a document containing instructions for shutting off the electricity. This document would be located in the kitchen on the second floor. After retrieving it, the player would then need to locate and open the electrical distribution box, which would be placed on the same floor. Inside the box, the player would see two groups of switches: five top switches that could be set to either "On" or "Off" and four bottom switches with three possible positions—upper, middle, and lower. The player would have to hold the document in hand and set all switches according to the configuration described in the instructions. Once all switches were correctly set, the player would be able to pull the main lever to cut off the power supply to the house. While this method would not fully replicate real-world electrical systems, it would be intentionally simplified and gamified to enhance player engagement. At the same time, the task would maintain its educational value by emphasizing the importance of cutting off electricity during emergencies—a key safety step players would likely remember.

Shutting off the gas supply

The objective of this task would be to shut off the gas supply to the house before the bomb explodes. In real-life emergency situations, turning off the gas is a critical step in preventing dangerous leaks, fires, or even secondary explosions. This task would aim to educate players about the risks associated with damaged gas infrastructure and emphasize the importance of preventive safety measures during crisis scenarios. The player would first need to locate the gas distribution unit to complete the task. This unit would be positioned near a set of benches next to the house. Its location would remain the same in every playthrough to ensure that players could reliably find it. After locating the gas unit, the player would need to interact with the ball valve. Upon looking at it, an on-screen prompt would appear: "Shut off the main gas supply". When the player pressed the interaction key (such as E or the left mouse button), the valve would rotate visibly, and the gas supply would be cut off. The task would then be marked as completed. Completing this task would reward the player with 10 points.

Shutting off the water supply

The objective of this task would be to turn off the main water supply to the house before the bomb detonates. In real-life emergencies, shutting off the water can help prevent flooding caused by damaged pipes or plumbing failures resulting from the shockwave. This task would aim to teach players that water infrastructure—just like electricity and gas—should be secured during a crisis. To complete this task, the player would first need to locate the water distribution unit, which would be situated in one of the rooms in the basement. After finding the unit, the player would have to identify the ball valve. When looking directly at

it, an on-screen prompt would appear: “Shut off the main water supply”. By interacting with the valve (using the E key or the left mouse button), the player would rotate it into the closed position, successfully cutting off the water supply. This action would mark the task as completed, and the player would be rewarded with 10 points.

Reach safety before the explosion

This task would play a crucial role in the game, as it would have the potential to either secure the player’s final score or nullify all previously earned points. The goal of this task would be to take shelter in the safest room the player had been building and preparing throughout the level. A countdown timer would run during gameplay, representing the time remaining before the bomb detonates. To successfully complete the task, the player would need to be inside their designated shelter at the exact moment the timer reached zero. The final score awarded for this task would depend on the player’s location at the time of the explosion: If the player were inside their fully prepared shelter, they would receive 20 points. If the player were in the basement but not in their designated safe room, they would receive 10 points. If the player were elsewhere inside the house, they would be awarded 3 points. All previously earned points would be lost if the player was on the balcony or outside the house. This task would highlight a key lesson: even if a person completed all necessary preparations, failing to be in the right place at the right time could render all efforts meaningless.

3.3 Level 2—Emergency Shelter

The second level is intended to simulate a situation of immediate danger in which the player must react quickly and effectively. The main objective will be to locate and secure oneself in the safest room of the house, situated in the basement. This level is designed to emphasize time pressure and the ability to make fast decisions without access to the inventory or hotbar present in the previous level.

The level is planned to begin with a cutscene. In this sequence, a girl character will approach a window and notice three fighter jets flying directly toward the house. After the cutscene ends, control will be passed to the player, and the game will switch to a first-person perspective.

The gameplay will consist of a single straightforward task: within a time limit of 90 to 150 seconds (depending on the selected difficulty), the player will need to explore the house, identify the safest room, enter it, and close its door to take shelter. No additional gameplay mechanics will be active during this segment to allow the player to fully focus on completing the core objective.

If the player completes the task, a final cutscene is planned. This sequence will be shown from a bird’s-eye perspective, depicting three jets flying over the house and each dropping a bomb. After the explosions, firefighters will arrive on the scene to extinguish the fire and rescue the player.

No final cutscene will be shown if the player fails to complete the objective in time. However, the game is intentionally designed without a traditional “bad ending” to avoid causing unnecessary stress or negative emotions. The overall aim of the game is to support an educational and simulation-based experience rather than to frighten the player.

Chapter 4

Implementation

This chapter describes the technical implementation of the game, including its structure, systems, and components used to realize the game design. It begins by introducing the scenes and environment setup, continues with the implementation of player controls, inventory and hotbar systems, and outlines how game tasks and room detection work. The chapter also explains the creation and integration of in-game cutscenes to enhance immersion and narrative flow.

4.1 Scenes

The game utilizes three main scenes: MainMenu, Level1, and Level2. When the game launches, the MainMenu scene is the first scene the player sees by default. As the name suggests, this scene serves as the game's main menu. From this scene, the player can choose to exit the game, adjust the game settings, or start playing. The MainMenu scene is based on the 3D Modern Menu UI asset[1], which was subsequently customized to fit better the specific needs and visual style of the game. The game offers two playable levels: Improvised Shelter, which is scene Level1, and Emergency Shelter, which is scene Level2. When the player hovers over the name of each level in the menu, a brief description appears, providing context and outlining the level's objectives (see Figure 4.1). After the player selects a level, a loading screen is displayed while the game prepares the scene. This process is implemented using the LoadSceneAsync function, accompanied by a loading screen that informs the player that the chosen level is currently loading. During gameplay, the player can press the ESC key to pause the game. This action opens a pause menu (see Figure 4.1) with the following options: Continue (resumes the current game), Restart (restarts the current level), Menu (returns to the MainMenu scene), and Quit (exits the game entirely). Upon completing a level, the player has a results screen showing the score achieved during the session. Additionally, the screen provides buttons to play the other level, restart the completed level, or return to the main menu (see Figure 4.2).

[1] 3D Modern Menu UI. Unity Asset Store. Available at: https://assetstore.unity.com/packages/tools/gui/3d-modern-menu-ui-116144.

Figure 4.1: MainMenu (left) and PauseMenu (right).

Figure 4.2: Results after completing Level1 (left) and Level2 (right).

4.2 Environment

It was first necessary to create a terrain to create the game environment. The terrain was then shaped with hills to make the environment appear more natural. For this purpose, the *Terrain Sample Asset Pack*[2] provided by Unity was used, which enhances terrain editing capabilities with a set of brushes, textures, and materials, allowing for a more realistic appearance.

After shaping the terrain, a grass texture was applied, and paths leading through the environment were created using the *Outdoor Ground Textures*[3] asset pack, which offers a collection of realistic outdoor surface textures.

Subsequently, trees were added to the environment using the *Conifers*[4] asset pack models, including various tree models. Finally, grass and small vegetation were placed using the *Grass and Flowers Pack 1*[5], providing additional detail and natural diversity to the scene.

After the terrain was completed, the main building where the game takes place was added (see Figure 4.3). The building was organized based on materials, meaning that

[2]Terrain Sample Asset Pack, available at https://assetstore.unity.com/packages/3d/environments/landscapes/terrain-sample-asset-pack-145808.

[3]Outdoor Ground Textures, available at https://assetstore.unity.com/packages/2d/textures-materials/floors/outdoor-ground-textures-12555.

[4]Conifers, available at https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076.

[5]Grass and Flowers Pack 1, available at https://assetstore.unity.com/packages/2d/textures-materials/nature/grass-and-flowers-pack-1-17100.

all parts sharing the same material were combined into a single object. This approach is important for optimization, as grouping objects with the same material into one object significantly improves performance.

However, to create functional doors that could be opened during gameplay, it was necessary to import the building model into Blender and separate all doors within the building into individual objects. Since the doors consisted of multiple materials, they were each combined into a single object to allow proper manipulation. For each door, a separate animation was created in the game, which is triggered when the player attempts to open or close the door during gameplay.

Each building object has a mesh collider, ensuring accurate collision detection between the player and the building structures. A mesh collider was chosen instead of a box collider because the individual parts of the building are often irregularly shaped or positioned far apart from each other, making simple box colliders unsuitable. Although box colliders are generally more efficient for performance, mesh colliders were necessary in this case to provide precise collision boundaries.

Additionally, invisible walls were placed around the building to prevent the player from leaving the intended gameplay area.

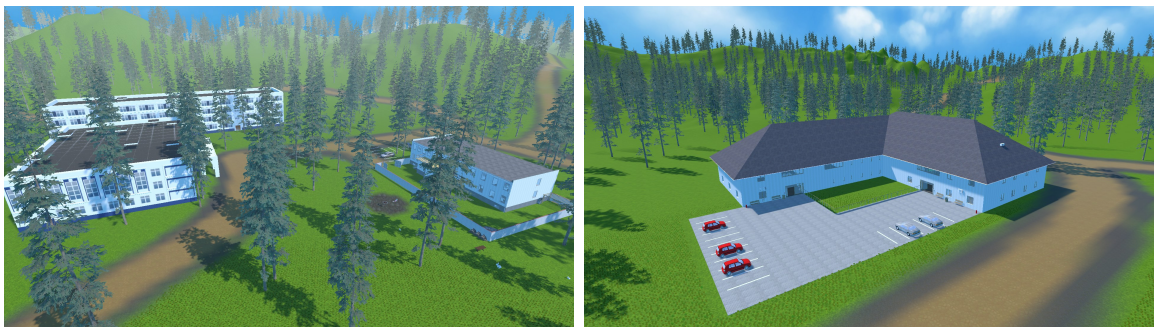

Figure 4.3: The left image shows the environment in which the first level takes place. A large school building is visible on the left side of the image; however, this structure is not accessible to the player and is included purely for visual enhancement, contributing to a more immersive and less empty scene. The right image shows the environment used in the second level.

4.3 First-person controller

The game is designed from a first-person perspective to immerse the player in the simulated emergency situation as much as possible. The player can move, rotate the camera, jump, sprint, and crouch. A stamina system limits sprinting—once the stamina is depleted, the player must walk to allow it to regenerate. The player also has the ability to zoom their view, which can be used when completing specific tasks that require attention to smaller objects or distant elements in the environment.

For implementing the player's movement, the Modular First Person Controller asset[6] was used as a foundation, which was subsequently modified and extended to better fit the needs of the game. The character controller was resized, as the default size was too large for the player to pass through doors and navigate the environment properly. Movement speeds

[6]Modular First Person Controller. Unity Asset Store. Available at: https://assetstore.unity.com/packages/tools/physics/modular-first-person-controller-233881.

were adjusted to create a balance between realistic movement and comfortable gameplay. In contrast, the player can modify the camera rotation speed in the game settings accessible from the MainMenu.

Several UI elements from the controller asset were customized to match the game's visual style better. Moreover, rotation input is disabled when the player opens the inventory or game tasks, ensuring that the player cannot unintentionally rotate the view while managing items or tasks.

The player controls a capsule object that holds the primary camera for first-person viewing. A secondary camera was also introduced, moving and rotating exactly like the primary camera. This secondary camera is used exclusively to render items the player holds. Setting up two synchronized cameras—one rendering the world and the other rendering only the held objects—ensures that items in the player's hands remain visible at all times. Without this solution, objects could visually clip into walls or disappear when the player approaches obstacles closely, negatively impacting realism and gameplay clarity.

A hand model visible from the first-person perspective was incorporated to ensure the player could see the objects they held. For this purpose, a character model was downloaded from Mixamo[7], a platform providing free 3D character models and animations. After downloading, the model was imported into Blender, where it was edited to retain only the hands. These cropped hands were then attached to the secondary camera setup, allowing them to move and rotate consistently with the player's perspective throughout the game.

Additionally, a custom 3D model of a document was created to serve as an interactive object within the game. The process involved designing a document in Microsoft Word, exporting it as a PNG image, and using Blender to model a thin plane resembling a sheet of paper. The document image was then applied as a texture to the 3D model, creating a realistic in-game representation of a document. In Unity, the document model was positioned relative to the hand models to give the appearance that the player was holding the document naturally. The hand model was adjusted to match the grip needed to hold the paper correctly. In gameplay, the document and corresponding hand pose are activated whenever the player picks up the document item, ensuring the interaction looks visually consistent and maintains immersion.

In addition to basic motion functions, the First Person Controller also includes an object interaction system. A dedicated interaction script continuously casts rays from the player's camera to detect whether the player targets an interactable object within a defined range. Interactable objects include collectable items, doors that can be opened and closed, in-game characters such as the friend who must be rescued and with whom the player can interact, and construction points for building better shelters.

In the case of construction, interaction requires not only aiming at the correct spot but also having the appropriate item equipped in the active hotbar slot. This ensures that players must first locate and equip the necessary building materials before they can proceed with improving their shelter. This system promotes immersion and reinforces gameplay mechanics by linking physical actions—such as holding the correct item—to the player's ability to interact with the environment.

[7]Mixamo. Available at: https://www.mixamo.com/.

Figure 4.4: The left image presents the player's first-person perspective while holding an interactive document. The document is highlighted in green within the hotbar, indicating that it is currently selected and in use. The right image illustrates the use of cropped 3D hand models that are attached to a secondary camera and synchronized with the player's movements. These hands, together with the document, become visible whenever the document is activated through the hotbar, enabling the player to view and read its contents within the game environment.

4.4 Inventory

To implement the inventory, creating items in the game that could be collected and stored was necessary. Each such object is represented as a 3D model located in the game world that the player can access, target and then collect.

Most of the 3D models used in the game come from *Sketchfab*[8]. These models were first imported into *Blender*, where they were modified to suit the project's needs, such as resizing them, trimming unnecessary parts, or splitting the models into multiple objects. Once the modifications were complete, the models were exported in `.fbx` format and imported into the project in Unity.

Once imported into Unity, it was necessary to extract textures from the models and then assign these textures to the materials used by the 3D objects. This step was necessary to avoid problems where models would be displayed in the game without textures (i.e. all white). A faithful visual representation of each object was achieved by applying the correct textures to the materials.

Each collectable item is represented in the inventory system by its own `ItemData` data object. This object contains basic information about the object, such as name, icon and description. The icons of each item were mainly obtained from the *Flaticon*[9].

Each collectable object in the game has an associated `BoxCollider` component that allows the detection of interaction between the player and the object. In addition, an `ItemInteraction` script is attached to the objects, which inherits from its own `Interactable` base class. This makes it so that when the player's gaze is focused on an object, a text prompt to pick it up appears on the screen, and the player can add the object to his inventory.

Collected items can be displayed in the player's inventory at any time. The inventory comprises 15 slots instantiated as a prefab object named `Item`. This prefab contains a `Button` component that serves as the fundamental graphical element representing each slot in the inventory.

[8]Sketchfab, available at https://sketchfab.com/feed.
[9]Flaticon, available at https://www.flaticon.com.

Inside the `Item` object is the `DraggableItem` child. This `DraggableItem` object also contains three children:

- `ItemIcon` (the `Image` component) used to display the icon of the item,
- `ItemName` (the `TextMeshProUGUI` component) to display the name of the item,
- `ItemQuantity` (the `TextMeshProUGUI` component) indicates the item's quantity.

When an item is picked up, the `Add` method in the `InventoryManager` singleton is called. This method first checks whether the collected object is already in the inventory or the hotbar. If so, it will only increase its `ItemQuantity`. If the item is not in any slot, the method will search for the first free slot in the inventory and insert the item into it. If there is no free slot in the inventory, the item is inserted into the first free slot in the hotbar.

This structure lets the player view all collected items in the inventory. Placing the `ItemIcon`, `ItemName` and `ItemQuantity` under the `DraggableItem` object also allows the implementation of the drag and drop feature for items in the inventory. The drag-and-drop operation moves the entire `DraggableItem` object, including all its child items, allowing easy items to move between slots.

The inventory can be opened by pressing the `Tab` key (see Figure 4.5). When opened, the player is presented with an overview of all collected items and the mouse cursor is released to interact with the user interface. Closing the inventory again hides its display and locks the cursor for smooth first-person character control.

4.5 Hotbar

A hotbar is used in the game to use items during gameplay quickly. This hotbar is visible throughout the gameplay at the bottom of the screen and contains six slots in which the player can place any items from the inventory.

One of the slots can always be active. Slots can be activated using the number keys `1` to `6`, which are located above the standard game controls `WSAD`, or by scrolling with the mouse wheel. The active slot is graphically highlighted by changing the background colour to green. Activating a slot means that the item placed in that slot is ready for use in the game. For example, when building a wall, it is necessary to have bricks placed in the active slot in order to perform a building action (see Figure 4.6).

The slots in the hotbar are structured similarly to those in the inventory, allowing full compatibility when managing items. Items can be moved between the inventory and the hotbar in two ways:

- **Using drag-and-drop**: the player grabs a `DraggableItem` object and moves it between slots.
- **By clicking on the item**: the item is automatically moved to the first available slot in the hotbar or back to the inventory.

Figure 4.5: An example of the in-game user interface when pressing the `Tab` key. The inventory and hotbar display various items that can be collected in the game.

Figure 4.6: An example of the wall-building process. The player first had to break a window and install ventilation. The image shows the final stage, where the ventilation is being sealed with bricks.

4.6 Game Tasks

The `GameTasks` system in the game is a mechanism for assigning tasks to the player. Tasks guide further progress in the game—they inform the player what needs to be done, what the goal is and how to achieve it. The player receives points for completing them, motivating him to progress through the game actively (see Figure 4.7).

The task display is closely linked to the inventory—the list of current tasks is shown or hidden together with the inventory using the `TAB` key. Each main task may contain several

sub-tasks, for which the player earns points. The current total score is displayed at all times in the top right corner of the screen, allowing the player to keep track of their progress.

Tasks can be divided into three basic categories:

1. **Collecting items**
   This task requires the player to collect a certain number of items. A corresponding *progress bar* is displayed on the screen to visually indicate progress – showing how many items the player has collected out of the total required amount. Next to this indicator is text indicating the number of points the player has earned for the task. Once the progress bar is filled, the player knows the task has been completed successfully.

2. **One-off tasks without a progress bar**
   These tasks do not contain a progress bar. The player is only provided with a text description that explains what needs to be done. Completion of these tasks is usually related to one-time actions or interactions that cannot be quantified by a progress bar. Once completed, the task is marked as completed, and the player receives points.

3. **Delayed task**
   This type of task is only used once in the game and is similar to the second type – the player is given a text description of the task, but its completion is only evaluated at the end of the game after the bomb explodes. Thus, the player does not have immediate feedback and only learns whether the task has been completed after the end of the game. This approach serves to encourage long-term planning and adds an element of uncertainty to the game.

Figure 4.7: Screenshots captured during gameplay illustrating all available tasks in the first level of the game. The images show a combination of completed and ongoing tasks, demonstrating how the task system is visually presented to the player within the game interface.

4.7 Room detection

In the game, each room and outdoor area has a floor consisting of an invisible object with an associated `Layer Floor`. These objects serve as target surfaces for the Raycast, which is used when placing the sleeping bag. When the player has the `Sleeping Bag` item active, a preview of the sleeping bag is displayed, which follows the cursor and is placed based on the impact of the Raycast from the camera on the objects with `Layer Floor`.

When the sleeping bag is attempted to be placed (by clicking the mouse), another Raycast is performed—this time pointing downward from the sleeping bag's intended placement–to verify that the location is part of one of the rooms. Detection is based on objects with the `RoomTrigger` tag. If a room is detected, its name is passed to the `SleepingBagRoomValidator` system, verifying whether the room is survivable.

If the room is judged unsuitable (see Figure 4.8), the player loses one of five attempts, and a warning message appears on the screen explaining why the chosen location would be unsafe (e.g. due to too many windows or location near the edge of a building). If, on the other hand, the room is judged suitable, the sleeping bag is placed, the item is removed from the inventory, and the room is marked as a safe shelter for the next part of the game.

These room floors and outdoor areas are further used in the *Reach safety before explosion* task. Throughout the game, the player has a `PlayerRoomTracker` script attached that

continuously monitors which room or zone the player is currently in. When the time limit expires and the bomb explodes, the game determines whether the player was inside a building or outside at that moment. Based on this information, the player is awarded a score—if they were inside a shelter, they receive points, but if they were outside during the explosion, all points are deducted.

Figure 4.8: The images show the feedback the player receives when attempting to place a sleeping bag. If the player tries to place it in a room that is not suitable for survival, a warning message appears on the screen explaining why the location is unsafe. Each room has its own specific explanation to inform the player about the potential dangers.

4.8 Game Cutscenes

Both game levels include cutscenes displayed at the end of the level and depict an explosion. In the first level, the final cutscene is always triggered after the time limit for completing the tasks expires. In the second level, the cutscene is only shown if the player completes the single objective—to find a secure shelter and close themselves inside quickly. If the player fails to complete this task, a prompt encourages them to try finding shelter again. Additionally, the second level begins with an introductory cutscene in which a female character approaches a window, looks outside, and sees incoming fighter jets heading toward the house she is currently in. After this scene, gameplay begins.

Level 1—Improvised Shelter

The cutscene at the end of the first level is automatically triggered once the time limit for completing the tasks runs out. Regardless of how well the player prepared during the game, they are always directed to a predefined animated sequence once the time is up. This sequence is intended to provide a dramatic conclusion to the level and enhance the game's overall atmosphere.

Once the cutscene is triggered, all interactive game elements are gradually deactivated—the player loses control, and the crosshair, inventory, hotbar, timer, and other UI elements are hidden. The player is thus transitioned into a passive viewing mode.

A smooth transition occurs between the primary gameplay camera and the cutscene camera. This transition is implemented using the `ActivateCutscene` script, which interpolates the position and rotation of the primary camera to match the position of the designated cutscene camera over a predefined duration. After the transition is completed, a `PlayableDirector` component is activated, which plays the animation defined in the Unity Timeline.

The Timeline controls the entire progression of the cutscene and contains several tracks that manage various objects. First, a fireball is activated and falls to the ground. This is followed by the detonation of a nuclear bomb—the author created both of these elements. Simultaneously, the `Fire` component is triggered, which represents spreading flames[10] igniting the surrounding environment (see Figure 4.9).

After the explosion, the screen fades to black, and the message "Several hours later..." appears. Before the scene fades back in, the `TerrainChanger` object is activated. It darkens the terrain texture and replaces large trees with smaller ones to simulate the visual impact of fire damage to the environment.

Once the screen fades back in, only the burning trees remain. At this point, the `firetruck` object is animated to arrive and park in front of the house where the entire level took place. The cutscene then ends, concluding the level (see Figure 4.2).

At the end of the cutscene, the player is presented with a performance evaluation message based on the score achieved during gameplay.

Figure 4.9: These images are screenshots from the the cutscene in the first level. The first image (left) captures the moment shortly after the explosion. Just before this, a fireball had struck the ground, releasing sparks and flames upon impact and eventually forming a characteristic mushroom-shaped cloud. The second image (right) shows the aftermath of the explosion—the surrounding trees are completely burned, the terrain appears scorched, and a fire truck is approaching the house to rescue the player and their friend.

Level 2—Emergency Shelter

The second level features two cutscenes—an introductory and a final one.

Introductory Cutscene

The introductory cutscene is triggered immediately after the level is loaded and establishes a sense of imminent danger. In this scene, the player observes a girl who automatically walks toward a window. After a brief delay, an animation is played in which the character looks outside and sees incoming fighter jets approaching the house she is currently in.

The entire sequence is controlled by the `CharacterMover` script, which handles the character's movement, triggers animations, and switches the camera to a view of the flying jets. These jets are controlled by the `JetMover` script, which enables them to move smoothly toward the house.

[10] Free Fire VFX – URP, available at https://assetstore.unity.com/packages/vfx/particles/fire-explosions/free-fire-vfx-urp-266226.

After a few seconds, the jets stop, the camera switches back, and the player's first-person controller is activated. At the same time, the game timer starts, marking the beginning of active gameplay. During the cutscene, the player also has the option to skip the sequence using a Skip button.

The script includes the `SkipCutscene` method, which allows the entire sequence to be skipped immediately. In that case, all animations and cameras are deactivated, and the game transitions directly into active gameplay mode.

Final Cutscene

The final cutscene in the second level is only triggered if the player successfully completes the objective—to find a secure room and close themselves inside. The game timer is paused at this moment, and the screen gradually begins to fade to black. If the player stays inside the room long enough without opening the door, the game evaluates the objective as completed and starts the final sequence.

This entire process is managed by the `SafeRoomController` script, which monitors the player's position and the door's status. Once the required time has passed, the script calls the `ActivateFinalCutSceneWithFade` method from the `FinalCutsceneManager` class, which switches the camera, turns off the user interface and player control, and plays the Timeline with the prepared animation.

The cutscene begins with a dark screen and a transition to a camera showing the scene from an external perspective. In the Timeline, three fighter jets approach the house and drop bombs on the building where the game took place. This is followed by three large explosions[11] around the house, which then catches fire (see Figure 4.10).

Shortly after, a fire truck arrives and stops in the parking lot in front of the house. The screen fades to black again, displaying "One hour later...". When the scene fades back in, the fire has been extinguished, and the fire truck leaves. On the parking lot, the girl from the introductory cutscene is shown happily jumping—symbolizing survival and the successful handling of a crisis.

If the player fails to complete the objective (i.e., is not in the room or leaves the door open), an alternative sequence is triggered via `ActivateCameraWithFadeOnly`. This version shows only the camera transition without any explosions. The player is then informed that the objective was not completed and is offered the chance to try again.

[11] Cinematic Explosions, available at https://assetstore.unity.com/packages/vfx/particles/fire-explosions/cinematic-explosions-free-257086.

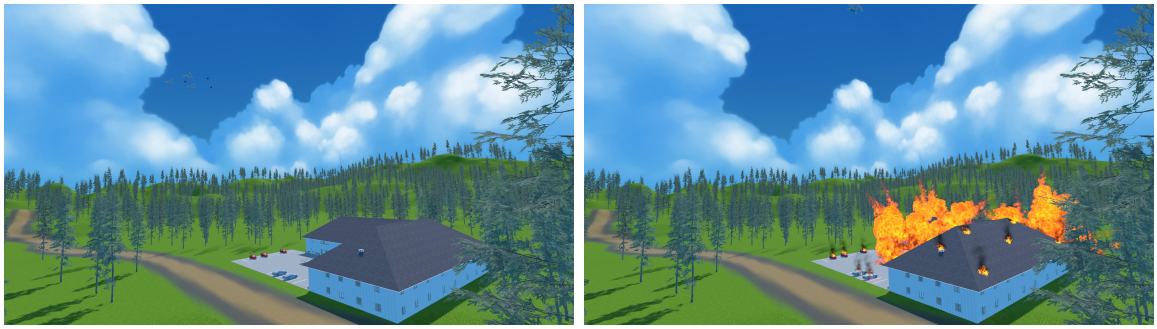

Figure 4.10: Screenshots from the game showing the final cutscene in the second level. The left image depicts fighter jets flying over the house and dropping bombs. The right image captures the explosion after the bombs hit the house, where the player is hiding in the basement. Following the explosion, a fire truck arrives to extinguish the fire and then departs. Meanwhile, the player's character celebrates in the parking lot, symbolically thanking the firefighters for the rescue.

Chapter 5

Game Testing and User Feedback

After the game was fully completed, a questionnaire was created using Google Forms to collect feedback from players. The questionnaire, along with the game and an attached README file that described the game's concept and controls in detail, was distributed among acquaintances and friends. It was also sent to a family member working as a QA tester at Snapshot Games. This respondent provided very detailed feedback focused on functionality, playability, and the overall user experience. A total of 11 people completed the questionnaire. The testing aimed to identify problematic elements and areas with potential for improvement so that the game's final version would be as high-quality as possible and best fulfil its educational purpose.

5.1 Participant Information

The questionnaire began with basic demographic questions to better understand the profile of the respondents who participated in the testing (see Figure 5.1). A total of 11 individuals completed the questionnaire, all of whom identified their gender as **male**.

The age distribution showed that the majority of participants (81.8%) fell into the **19–25** age category, while the remaining 18.2% were in the **26–35** age group.

Responses to the question *"How often do you play video games?"* varied significantly. The results were evenly split among those who play **daily** (27.3%), **several times a week** (27.3%), and **rarely** (27.3%). The remaining 18.2% indicated that they play video games **occasionally**. This variety in responses ensured that the game was tested by users with a broad range of gaming experience—from regular players to those who play only rarely or not at all.

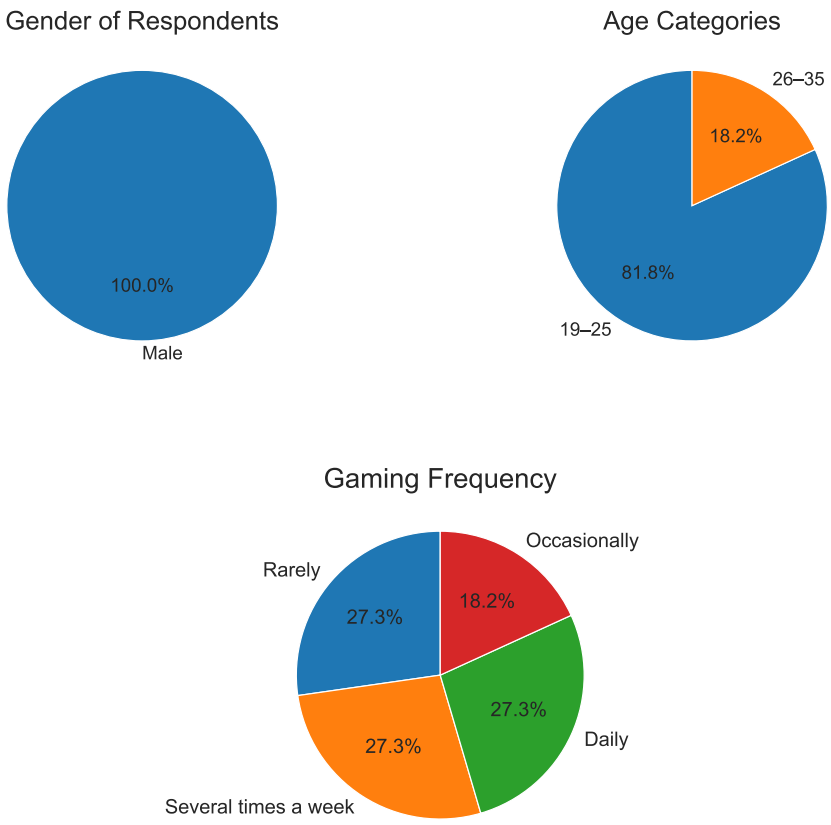

Figure 5.1: Results from the questionnaire: gender (top left), age categories (top right), and gaming frequency (bottom).

5.2 Level 1—Improvised Shelter

Another part of the questionnaire evaluated individual tasks that respondents completed in the first level. Each task was rated based on its difficulty (see Figure 5.2). The scale included the following levels: **very easy**, **easy**, **medium**, **hard**, **very hard**, and **did not manage to start**. This difficulty rating was used to identify which tasks were too challenging for players. According to the results, all tasks were generally considered easy, with some possibly being more difficult but nothing overly problematic. Therefore, it was decided to keep all the tasks as they are.

Respondents were then asked how many tasks they completed during their first playthrough of this level (see Figure 5.3). More than half of the participants stated that they completed **less than half** of the tasks. This suggests that the time limit for completing the level might have been too short.

Based on these results, it was decided to adjust the time limit—increasing it from the original 8–12 minutes to 10–20 minutes. This gives players more room to complete the tasks while maintaining a sense of time pressure.

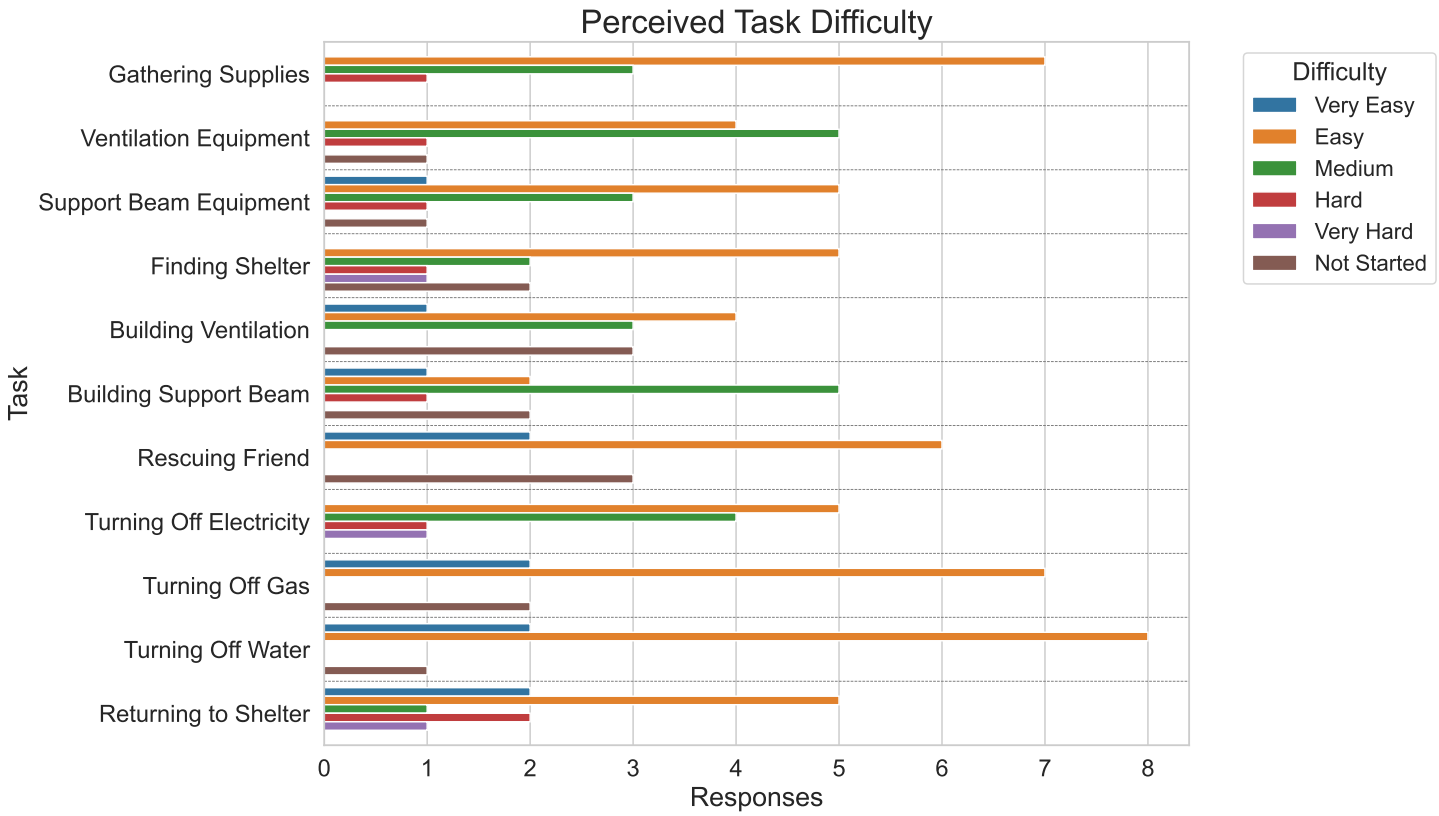

Figure 5.2: Difficulty ratings of individual tasks in Level 1 (Improvised Shelter).

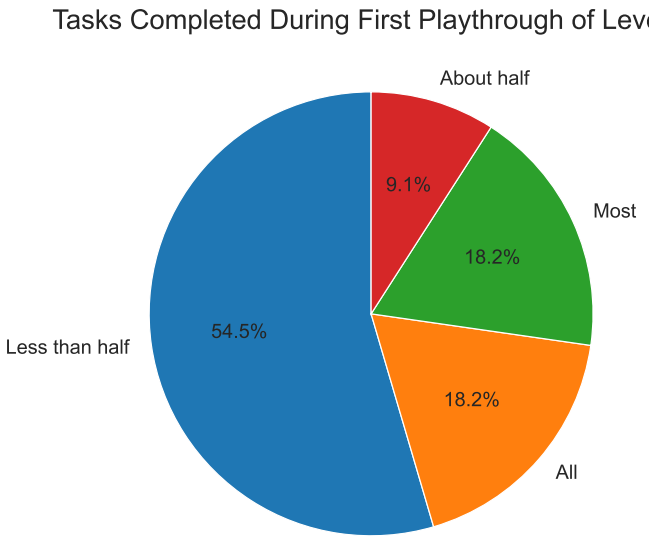

Figure 5.3: Number of tasks completed by players during their first playthrough of Level 1.

5.3 Level 2—Emergency Shelter

The next part of the questionnaire focused on the second game level, whose main objective is to find the safest room and seal oneself inside. Respondents were asked to rate the difficulty of this task using the following scale: **very easy**, **easy**, **medium**, **hard**, **very hard**, and **I was unable to find the room**.

The results show that most respondents rated the task as **easy** (54.5 %) or **very easy** (27.3 %). One respondent rated the difficulty as **medium** (9.1 %) and one stated they **were unable to find the room** (9.1 %). The options *hard* and *very hard* were not selected by any participant.

In the next question, participants were asked whether they managed to find the safest room during their first playthrough. 72.7 % answered **yes**, while 27.3 % **did not find the room on their first attempt**.

These results indicate that the task was generally perceived as simple, and most players could complete it successfully without significant difficulties (see Figure 5.4).

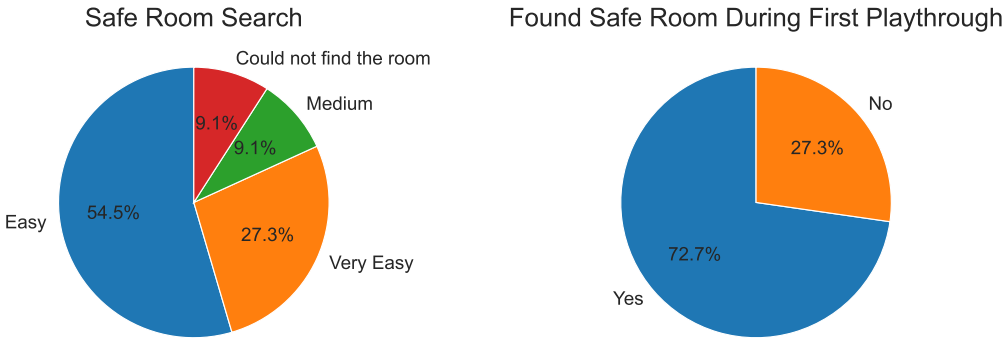

Figure 5.4: Results from Level 2: difficulty of finding the safest room (left) and whether it was found on the first try (right).

5.4 Overall Evaluation

In the next part of the questionnaire, respondents were asked to rate three key aspects of the game: **fun**, **visual quality**, and **overall impression**. The evaluation used a 5-point scale, where **1 represented the lowest rating** and **5 the highest** (see Figure 5.5).

Fun Most players evaluated the game positively. The most common rating was **4** (6 responses), followed by the highest rating **5** (3 responses). Only **2 respondents** gave the game a rating of **3**, which indicates that the game was generally enjoyable, although some players saw room for improvement.

Visual Quality The visual aspect of the game received similar ratings as the fun factor. Again, the most frequent rating was **4** (6 responses), followed by **5** (3 responses). Two respondents gave a score of **3**. None of the respondents chose the lowest rating (1 or 2), suggesting that the game's graphics were at least satisfactory for all participants.

Overall Impression The overall rating of the game followed the same trend. The most selected score was **4** (6 responses), followed by **5** (3 responses) and **3** (2 responses). No one gave a rating of 1 or 2 in this question either. Players thus perceived the game as a solid and functional product that met their expectations.

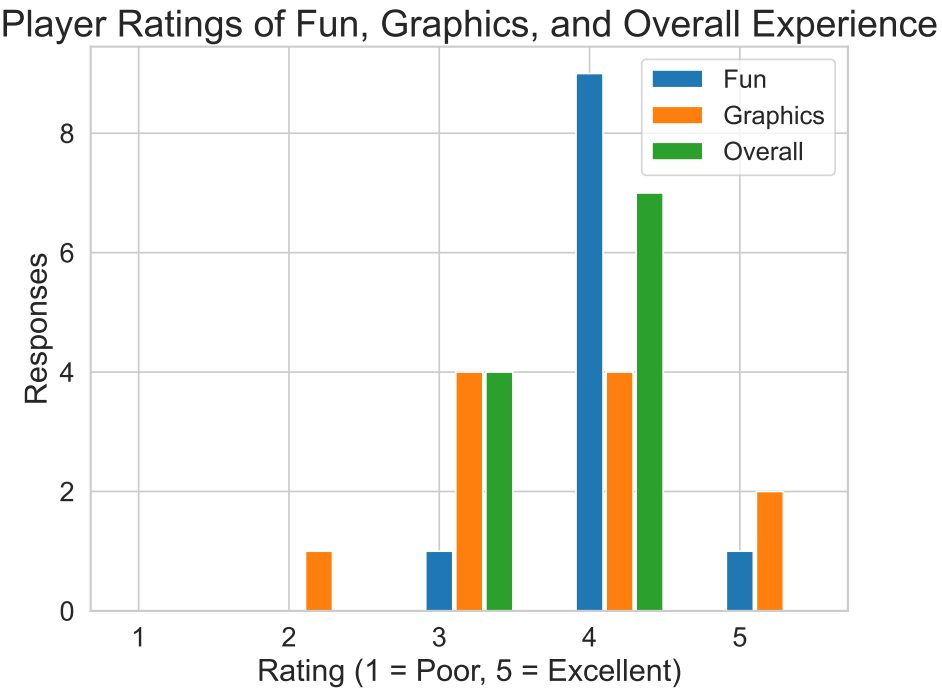

Figure 5.5: Distribution of player ratings for three aspects of the game: fun, graphics, and overall experience. Ratings were given on a scale from 1 (poor) to 5 (excellent). Most players rated all aspects positively, with the majority of responses centered around 4.

5.5 Clarity of Rules and Control Intuitiveness

The questionnaire included two key questions designed to understand better the user experience: the **clarity of the rules and task descriptions** and the **intuitiveness of the game controls** (see Figure 5.6).

In the question related to the clarity of the task instructions, 72.7 % of respondents stated that the rules were **mostly clear**, while 9.1 % rated them as **very clear**. Only 18.2 % of participants considered the rules to be **somewhat unclear**. This suggests that most players did not encounter major issues in understanding the objectives. The results also imply that some players might benefit from slight refinements or clarifications of specific instructions.

The game controls were generally perceived as intuitive—54.5 % of respondents said they had **no problems**, while the remaining 45.5 % admitted that they **occasionally struggled** during gameplay. This indicates that while the controls are generally well-designed, some players could benefit from a brief introductory tutorial or highlighting of key controls at the beginning of the game.

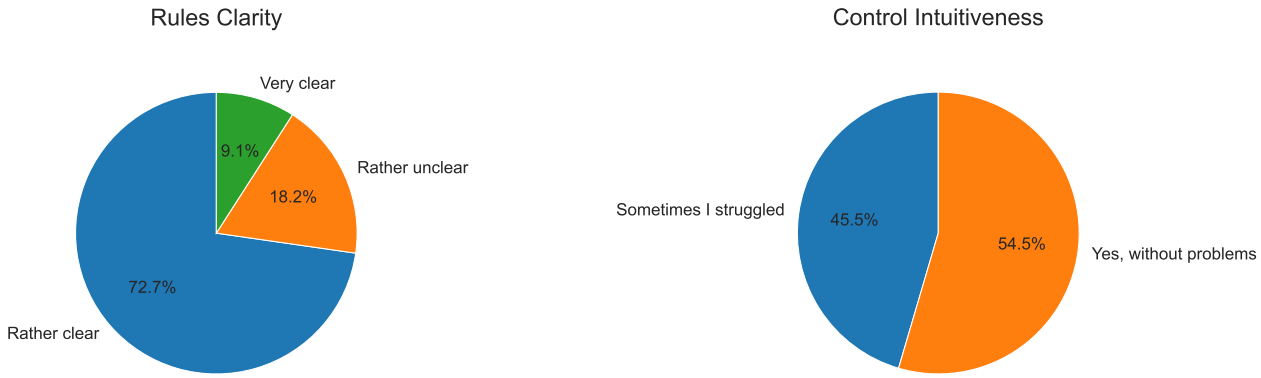

Figure 5.6: Player responses regarding the clarity of the rules (left) and the intuitiveness of the game controls (right). Most players found the rules to be mostly or very clear, and over half reported no issues with the controls, while a smaller portion experienced occasional difficulties.

5.6 Educational Value of the Game

Since the primary goal of the game was to **raise awareness about survival during military threats**, the questionnaire also examined whether players perceived the game as educational and whether they felt better prepared for a crisis after completing it (see Figure 5.7).

In response to the question *"Did you find the game educational in terms of survival during a military threat?"*, the majority of respondents answered positively:

- 54.5 % said **partially**,
- 36.4 % answered **yes, I learned something new**,
- only 9.1 % selected **I don't know**.

These results suggest that the game was able to convey helpful information to almost all players.

To the follow-up question *"Did you feel better prepared for a crisis after playing the game?"*, the responses were:

- 27.3 % answered **yes**,
- 36.4 % said **rather yes**,
- 36.4 % chose **rather no**.

The responses indicate that approximately two-thirds of participants felt more prepared after completing the game, while the rest either noticed no significant change or did not feel better prepared. Nevertheless, the game fulfilled its educational purpose at least partially, which is a positive outcome given the project's goals.

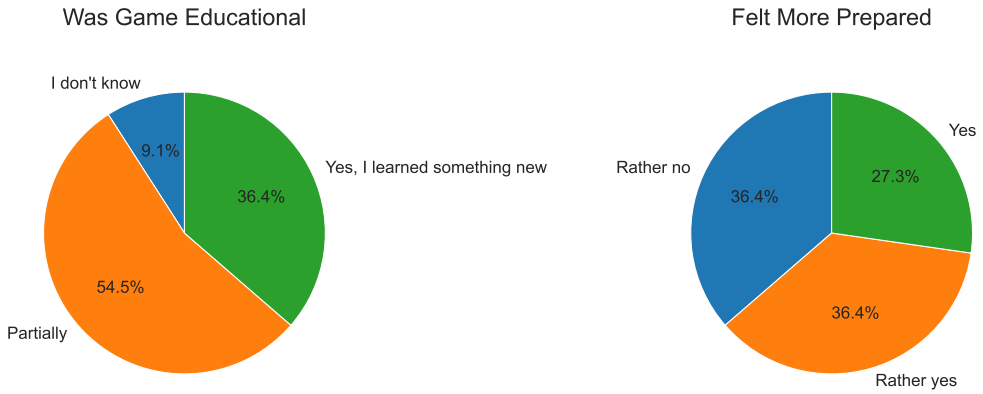

Figure 5.7: Participant responses to two questionnaire items assessing the educational effectiveness of the game: perceived informativeness (left) and self-assessed preparedness for crisis situations after gameplay (right). The results indicate that a majority of players considered the game at least partially educational, and approximately two-thirds reported an increased sense of preparedness.

5.7 Time Limit

The questionnaire also included how players perceived the time limit for completing tasks. The evaluation was based on a five-point scale, where 1 indicated **too short** and 5 **too long** (see Figure 5.8).

The results show that most respondents found the time limit **adequate**—6 out of 11 players selected the neutral midpoint 3. Four other participants considered the time to be **too short** (2 responses for 1 and 2 for 2), while one respondent rated the time as **long** (4). No one selected the option **too long** (5).

Despite the generally neutral evaluations, open-ended feedback revealed that some players needed more time to orient themselves within the game. For this reason, the time limit was adjusted—from the original **8–12 minutes** to **10–20 minutes**—to accommodate a broader range of players while maintaining moderate time pressure to encourage efficient task completion.

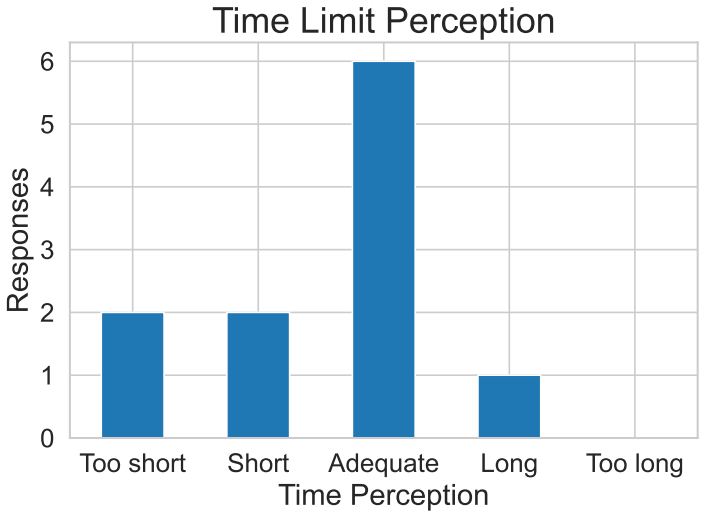

Figure 5.8: Player responses regarding the perceived time limit for completing tasks. The majority rated it as **adequate**, several considered it **too short**, and one respondent found it **long**. No participants selected **too long**.

5.8 Interest in the Theme and Possible Continuation

The questionnaire also explored how engaging players found the game's theme and whether they would be interested in continuing or expanding the concept (see Figure 5.9).

In response to the question *"Did the theme of the game interest you?"*, the majority of respondents answered positively:

- 54.5 % said the theme **somewhat interested** them,
- 27.3 % responded that it **interested them very much**,
- only 18.2 % answered **not really**.

These responses suggest that the theme of surviving an atomic bomb explosion or bombing was attractive to most participants and has the potential to appeal to a broader gaming audience.

Players were also asked whether they would welcome a **continuation or expansion** of the game. The results again showed a positive trend:

- 45.5 % stated they would **definitely** be interested,
- 54.5 % answered **maybe**.

None of the respondents selected the option **no**, indicating that the game sparked interest in additional content. For example, this could motivate future development in the form of new levels, alternative scenarios, or a deeper storyline.

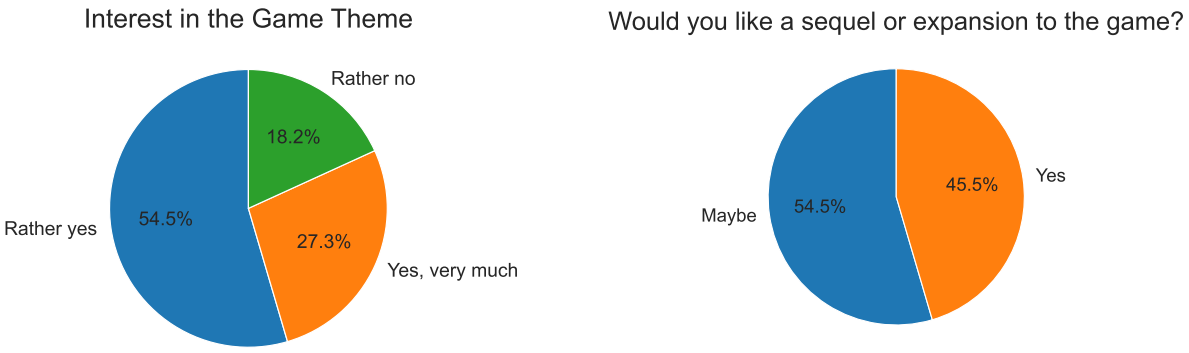

Figure 5.9: Responses to questions regarding the game's theme (left) and interest in a sequel or expansion (right). Most players expressed a clear or moderate interest in the theme, and all respondents showed at least some willingness to see the game further developed.

5.9 Open-Ended Responses and Technical Issues

At the end of the questionnaire, respondents were allowed to provide open-ended feedback in response to the following questions:

- *What did you like most about the game?*
- *Did you encounter any bugs or crashes while playing? If so, please describe what happened.*
- *What would you improve or change in the game?*

From these responses, it became clear that the most significant issue reported was related to **player movement inside the house**, where several players complained about frequently **getting stuck on walls and objects**. This problem negatively affected the fluidity of movement and the overall gameplay experience.

Based on this feedback, a **physics-based adjustment in the form of a slippery material** was added to the player character. This change prevented the character from snagging on collision objects and enabled smoother movement, resulting in a noticeably improved gameplay experience.

Chapter 6

Conclusion

This bachelor's thesis aimed to design and implement an educational 3D game in the Unity engine that teaches players how to behave appropriately in the event of a nuclear explosion or bombing. The game was conceived as a serious game, with an emphasis on educational value and interactive player engagement.

The development process took place in cooperation with representatives of fire and rescue services, who provided valuable feedback on the content and scenarios that should be included in the game to enhance its realism and educational impact. Their expertise made it possible to focus on key survival aspects that reflect real-world emergency procedures.

The result is a functional game with two levels: an improvised shelter scenario, where the player builds a safe hiding place in a regular residential building, and an emergency shelter scenario, where the player must quickly move to a designated safe room.

The game combines various gameplay elements—such as an interactive environment, time-limited tasks for which the user receives points, an inventory system, cutscenes, and animations—that create an immersive experience while fulfilling an educational purpose. These features allow players to actively engage in a simulated crisis situation and learn proper behaviour principles.

This thesis represented my first practical experience with digital game development. Although I had been interested in this field for a long time, it was only through this project that I had the opportunity to go through the entire process, from design to implementation. The experience confirmed that this is an area I would like to continue exploring professionally.

In the future, the game could be expanded to include an English-language version, music and sound effects to enhance the atmosphere and overall gameplay, and a player evaluation system in the form of a leaderboard. Other possible extensions include new levels focused on different types of crisis situations, such as fires, floods, or terrorist attacks. The game could also find application in educational contexts—for example, as a support tool for firefighters and other emergency services during school events and training sessions aimed at helping children understand how to behave during emergencies.

The project can also serve as inspiration or a proof of concept that even serious topics can be effectively and clearly communicated through a game environment, particularly for younger audiences.

Bibliography

[1] *The First Video Game?* Brookhaven National Laboratory, 2008. Available at: https://www.bnl.gov/about/history/firstvideo.php. [online]. [cit. 2025-02-05].

[2] *Doom*. The Strong National Museum of Play, 2015. Available at: https://www.museumofplay.org/games/doom. [online]. [cit. 2025-02-05].

[3] *Spacewar!* The Strong National Museum of Play, 2018. Available at: https://www.museumofplay.org/games/spacewar/. [online]. [cit. 2025-02-05].

[4] *PONG released*. History.com Editors, 2024. Available at: https://www.history.com/this-day-in-history/pong-released. [online]. [cit. 2025-02-05].

[5] Andrade, A. Game engines: a survey. *EAI Endorsed Transactions on Game-Based Learning*, november 2015, vol. 2. Available at: https://www.researchgate.net/publication/283657797_Game_engines_a_survey.

[6] Bethke, E. *Game Development and Production*. Wordware Pub., 2003. Wordware game developer's library. ISBN 9781556229510. Available at: https://books.google.cz/books?id=m5exIODbtqkC.

[7] Dag Spicer. *50 Years of Fun with Pong*. Computer History Museum, 2022. Available at: https://computerhistory.org/blog/50-years-of-fun-with-pong/. [online]. [cit. 2025-02-05].

[8] Fullerton, T. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. CRC Press, 2024. ISBN 9781003858447. Available at: https://books.google.cz/books?id=BgH0EAAAQBAJ.

[9] Gazis, A. and Katsiri, E. Serious Games in Digital Gaming: A Comprehensive Review of Applications, Game Engines and Advancements. *WSEAS TRANSACTIONS ON COMPUTER RESEARCH*, march 2023, vol. 11, p. 10–22.

[10] Gregory, J. *Game Engine Architecture, Third Edition*. CRC Press, 2018. ISBN 9781351974288. Available at: https://books.google.cz/books?id=1g1mDwAAQBAJ.

[11] Hasičský záchranný sbor České republiky. *Ukrytí obyvatelstva v České republice*. 2025. Available at: https://hzscr.gov.cz/clanek/ukryti-obyvatelstva-v-ceske-republice.aspx. Accessed: 2025-04-16.

[12] Kent, S. L. *The Ultimate History of Video Games: From Pong to Pokemon and Beyond : the Story Behind the Craze that Touched Our Lives and Changed the World*. 1st ed. Three Rivers Press, 2001. ISBN 9780761536437. Available at: https://www.book-info.com/isbn/0-7615-3643-4.htm.

[13] Kumar, N. *Fortnite Statistics 2025: Revenue & Active Users Data*. 2025. Available at: https://www.demandsage.com/fortnite-statistics/. Accessed: 2025-03-02.

[14] Lowood Henry E.. *Pac-Man*. Encyclopedia Britannica, 2024. Available at: https://www.britannica.com/topic/Pac-Man-1688279. [online]. [cit. 2025-02-05].

[15] Lowood, Henry E.. *Doom*. Encyclopedia Britannica, 2025. Available at: https://www.britannica.com/topic/Doom. [online]. [cit. 2025-02-05].

[16] Mattila, J.; Seppälä, T. and Salakka, K. The Little Engines That Could – Game Industry Platforms and the New Drivers of Digitalization. *SSRN Electronic Journal*, november 2021, vol. 101.

[17] Minhaz Us Salakeen Fahme, T. H. K. *How to Make a Game: Go From Idea to Publication Avoiding the Common Pitfalls Along the Way*. New York, NY: Apress, 2021. ISBN 978-1-4842-6916-9. Available at: https://link.springer.com/book/10.1007/978-1-4842-6917-6.

[18] Nadia Stefyn. *How Video Games Are Made : The Game Development Process*. CG Spectrum Institute, 2022. Available at: https://www.cgspectrum.com/blog/game-development-process. [online]. [cit. 2025-02-27].

[19] Penazzo, D. *2D Game Development: From Zero To Hero (Python Edition, Version 0.7.11-r1)*. 2024. Available at: https://therealpenaz91.itch.io/2dgd-f0th.

[20] Rosenberg Jennifer. *Pac-Man Video Game History and Background*. ThoughtCo, 2021. Available at: https://www.thoughtco.com/pac-man-game-1779412. [online]. [cit. 2025-02-05].

[21] Samuel Heaney. *The History and Evolution of Minecraft*. IGN, 2023. Available at: https://www.britannica.com/topic/Minecraft-electronic-game. [online]. [cit. 2025-02-16].

[22] The Editors of Encyclopaedia Britannica. *Minecraft*. Encyclopedia Britannica, 2025. Available at: https://www.britannica.com/topic/Minecraft-electronic-game. [online]. [cit. 2025-02-16].

[23] Unity Technologies. *Cancellation of the Runtime Fee and Pricing Changes*. 2024. Available at: https://support.unity.com/hc/en-us/articles/30322080156692-Cancellation-of-the-Runtime-Fee-and-Pricing-Changes. Accessed: 2025-03-01.