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

GAME WITH HAPTIC FEEDBACK

HRA S HAPTICKOU ZPĚTNOU VAZBOU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **OLEKSANDR PROKOFIEV**
AUTOR PRÁCE

SUPERVISOR **Ing. TOMÁŠ POLÁŠEK**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

156986

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Prokofiev Oleksandr**
Programme:	Information Technology
Title:	**Game with Haptic Feedback**
Category:	Computer Graphics
Academic year:	2023/24

Assignment:

1. Survey the state of alternative approaches to user interface and their implementation in game controllers.
2. Choose a target game controller, design a game using its feedback options and create a Game Design Document.
3. Implement the game by means of your choice.
4. Iterate implementation with continuous testing and feedback integration.
5. Evaluate your game in a user study.
6. Present your results using a poster and a short video.

Literature:

- Koster, Raph. Theory of fun for game design. O'Reilly Media, Inc., 2013.
- Schell, Jesse. The Art of Game Design: A book of lenses. CRC press, 2008.
- Yao, Richard et al. Oculus VR Best Practices Guide. Online, 2014.
- Leap Motion, VR Best Practices Guidelines. Online, 2015
- Unity Learn. Unity, https://learn.unity.com/.
- Further sources according to the supervisor.

Requirements for the semestral defence:
Goals 1, 2 and a working game prototype.

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

Supervisor:	**Polášek Tomáš, Ing.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	9.11.2023

Abstract

This bachelor's thesis details the creation of a digital game called "Cut the Red!" that integrates haptic feedback technology. The game is a 3D puzzle game developed using Unity, and its objective is to defuse a bomb by resolving various mini-games. The thesis includes an introduction to the gaming industry and its history, an exploration of modern alternative approaches to user interface, and their application in game controllers. Additionally, it details the process of creating a game design document and explains the implementation of various game elements. The game is then tested in a short user study to gather feedback on its mechanics and overall player experience.

Abstrakt

Tato bakalářská práce podrobně popisuje tvorbu digitální hry s názvem "Cut the Red!" která integruje technologii haptické zpětné vazby. Hra je 3D logická hra vyvinutá pomocí Unity a jejím cílem je zneškodnit bombu vyřešením různých miniher. Práce obsahuje úvod do herního průmyslu a jeho historii, zkoumání moderních alternativních přístupů k uživatelskému rozhraní a jejich aplikaci v herních ovladačích. Kromě toho podrobně popisuje proces vytváření dokumentu o designu hry a vysvětluje implementaci různých herních prvků. Hra je poté testována v krátkém uživatelském studiu, aby se získala zpětná vazba ohledně herních mechanik a celkového zážitku hráče.

Keywords

game, game design, Unity, haptic feedback, PlayStation, consoles, C#, UI, VR

Klíčová slova

hra, herní design, Unity, haptická zpětná vazba, PlayStation, konzole, C#, UI, VR

Reference

PROKOFIEV, Oleksandr. *Game with Haptic Feedback*. Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Tomáš Polášek

Rozšířený abstrakt

Videoherní průmysl za posledních několik desetiletí rychle rostl a stal se jednou z nejpopulárnějších forem zábavy. Díky moderním technologiím a internetu se hry staly médiem pro vyjádření uměleckých vizí, sociálních sdělení a interaktivních vyprávění. Videohry se neustále vyvíjejí a přizpůsobují novým technologiím a preferencím hráčů, od jednoduchých arkádových her až po složité a rozsáhlé světy s ohromující grafikou a příběhy.

Tato práce si klade za cíl vytvořit hru, která využívá technologii haptické zpětné vazby. K dosažení tohoto cíle byl proveden výzkum historie digitálních her, různých typů uživatelského rozhraní (UI) a konceptu haptické zpětné vazby a jejího potenciálního využití ve hrách. Závěrečná část je zaměřena na herní konzole relevantní pro diplomovou práci, představení konzolí se zajímavými metodami zpracování vstupu/výstupu a využití haptické zpětné vazby.

Po výzkumu byla vytvořena počítačová hra, která je kompatibilní s ovladačem PlayStation 5 DualSense, inovativním ovladačem, který přináší nový rozměr herního zážitku. Díky pokročilé haptické technologii a dynamickým adaptivním spouštěčům mohou hráči cítit hru intenzivněji a reagovat na každou herní situaci s proměnlivou intenzitou a napětím. Ovladač DualSense má také pohybové senzory, které hráčům umožňují využívat ovládání pohybu ve vybraných hrách.

Vytvořená hra s názvem "Cut the Red!" je 3D logická hra, kde hráči přebírají roli specialistů na zneškodňování bomb, kteří mají za úkol odzbrojovat složitá zařízení pomocí jedinečných funkcí ovladače DualSense. Bomba se skládá z různých modulů, které představují různé výzvy od navigace v labyrintu po zodpovězení načasovaných otázek a přesné řezání drátů. Každý slot bomby vyžaduje použití různých senzorických vstupů: haptická zpětná vazba, adaptivní spouště a snímání pohybu. Hra je navržena tak, aby otestovala koordinaci hráčů, dovednosti při řešení problémů a schopnost zvládat stres pod tlakem, aby bylo možné bombu úspěšně zneškodnit.

Hra byla implementována v herním enginu Unity s využitím jeho možností a konceptů. Všechny skripty pro správu hry byly napsány v jazyce C#. Pro využití funkcí řadiče DualSense na PC byl použit balíček UniSense.

Závěrečná část práce je zaměřena na testování hry. Během procesu vývoje bylo provedeno manuální testování, které zkoumalo a testovalo všechny aspekty herních mechanismů, prvků uživatelského rozhraní, vstupních metod a hmatové zpětné vazby, aby bylo zajištěno, že hra bude vysoce kvalitní, stabilní a hratelná. Kromě toho bylo testování hry prováděno s hráči s různými herními zkušenostmi, aby bylo možné získat zpětnou vazbu o hratelnosti hry, designu úrovní, obtížnosti a celkové hráčské zkušenosti. Kombinací těchto testovacích metodologií by mohla být hra iterativně vylepšována a vylepšována, což hráčům poskytuje vybroušený a poutavý herní zážitek.

Cíl práce považuji za splněný, hra využívá haptických prvků DualSense. Testy hráčů ukazují, že hra je docela zábavná a dostatečně náročná a hráči si herní zážitek užívají. To považuji za hlavní úkol vývoje her.

Game with Haptic Feedback

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Tomáš Polášek I have listed all the literary sources, publications, and other sources, that were used during the preparation of this thesis.

.....
Oleksandr Prokofiev
May 6, 2024

Acknowledgements

In this section, I would like to express my gratitude towards those who have made this journey less daunting. I would like to extend my heartfelt thanks to my supervisor, Ing. Tomáš Polášek, whose unwavering support has been indispensable in the completion of this thesis. I am also deeply indebted to my family, whose support played an essential role in my success throughout my studies. Additionally, I would like to thank all my friends who helped in testing the game.

Contents

List of Figures

Chapter 1

Introduction

Over the past decades, the video game industry has become one of the most dynamically growing entertainment industry sectors [12]. With the advent of modern technologies and the widespread availability of the internet, games have become not only a popular form of entertainment but also a significant medium for expressing artistic visions, social messages, and interactive narratives. From the early simple arcade games to today's expansive worlds with stunning graphics and stories, video games continue to evolve and adapt to new technologies and the tastes of players.

Various gaming concepts emerge in this dynamic and innovative atmosphere, reflecting gamers' diversity and preferences. While some games attract a sizeable audience due to their simplicity and entertainment value, others focus on deeper storytelling and complex gameplay mechanics.

We have decided to venture into the world of game development by creating a PC game that will be compatible with the PlayStation 5 controller. As one of the top gaming consoles available, the PlayStation 5 boasts various features and technologies that can significantly enhance the gaming experience and provide exciting new opportunities for developers.

The thesis is divided into five chapters, each contributing to the overall goal of this thesis. The **Research** 2 chapter will explore the history and development of games. It will also cover User Interface and haptic feedback concepts and their potential use in games. The **Game design** 3 chapter will describe the process of game design document creation. The **Technologies** 4 will focus on the game development process, technologies, and the chosen platform for the development of the game. In the **Implementation** 5 part of this thesis, the implementation phases of game development will be detailed, including the development process's peculiarities and the outcome's presentation, which will be the main and most extensive part of the work. Finally, the **Testing** 6 chapter will include feedback from users who tested this game and evaluate possible improvements, positive aspects, and successful implementation of concepts.

This thesis will provide a comprehensive view of the game development process for PC using the PlayStation 5 controller and contribute to understanding the specifics of this platform and its impact on the overall gaming experience.

Chapter 2

Research

This chapter is divided into three parts. In the first part, a brief history of digital games is presented, ranging from the very first game created to modern-day games. The second part describes various types of user interface (UI) and explains the concept of haptic feedback and its potential use in games. The third part focuses on gaming consoles that are relevant to my bachelor's thesis on haptic feedback and introduces consoles with interesting methods of input/output processing related to this work.

2.1 History of videogames

Video games have become an integral part of modern entertainment, capturing the hearts and minds of millions of people worldwide. From simple pixelated graphics to lifelike virtual reality experiences, the journey of video games has been nothing short of remarkable. The origin story of video games can be traced back to the 1950s when engineers began crafting rudimentary games on early computers. A pivotal moment arrived in 1962 with MIT student Steve Russell's creation of „**Spacewar!**“, featuring controllable spaceships engaging in battles amidst simulated gravity. Today, video games have evolved into a **multi-billion-dollar** [12] tech enterprise (fig. 2.7), offering a diverse array of immersive experiences ranging from simple pixelated graphics to lifelike virtual reality simulations. This journey from humble beginnings to the forefront of entertainment underscores video games' remarkable evolution and enduring appeal in shaping contemporary culture.

2.1.1 Pioneering Origins and 1960s

The genesis of video games can be traced back to the nascent days of computing in the mid-20th century when visionary minds embarked on a quest to harness the potential of emerging technologies for recreational purposes. In the 1940s and 1950s, scientists and engineers laid the groundwork for the medium, experimenting with primitive machines to craft rudimentary gaming experiences. A seminal moment in gaming history occurred in 1958 when physicist William Higinbotham created „**Tennis for Two**“[1], a primitive tennis simulation displayed on an **oscilloscope**, allowing two players to engage in virtual gameplay. This groundbreaking endeavor predates the iconic „**Spacewar!**“[2] developed by MIT students in 1962, which featured controllable spaceships engaged in interstellar battles, heralding a **new era** in interactive entertainment.

[1] The history of Tennis for two https://www.bnl.gov/about/history/firstvideo.php
[2] The history of Spacewar! https://www.thoughtco.com/history-of-spacewar-1992412

Despite its limited commercial success, „**Tennis for Two**“ played a significant role in shaping the trajectory of video game development. Further are images of these games (fig. 2.1).

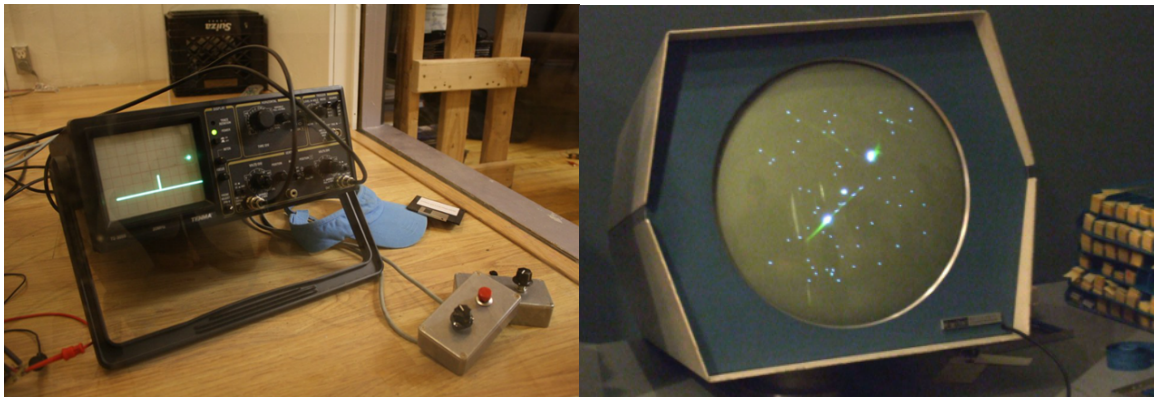

Figure 2.1: Tennis for two (left) and Spacewar! (right)[3].

2.1.2 The Arcade Renaissance 1970s

The 1970s witnessed the meteoric rise of video games with the advent of **arcade halls**, where patrons flocked to experience the latest gaming sensations. The watershed moment came in 1972 with the release of „**Pong**“ by Atari, a simplistic yet mesmerizing table tennis simulation that captivated audiences worldwide. As quarters clinked into arcade cabinets, a cultural phenomenon was born, marking the inception of the arcade renaissance. Buoyed by the success of „**Pong**“, a tidal wave of iconic titles inundated the market, including „**Space Invaders**“, „**Pac-Man**“, and „**Donkey Kong**“[4]. These games captivated players with their immersive gameplay mechanics and vibrant visual aesthetics, transforming arcades into bustling hubs of social interaction (fig. 2.2).

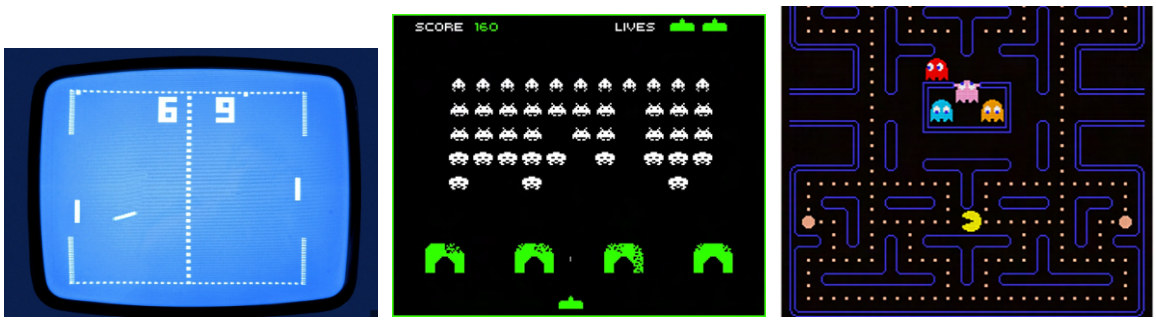

Figure 2.2: Pong (left), Space Invaders (middle), and Pac-Man (right)[5].

[3]Source: Tennis for two - HeatSync Labs Flickr and History of Spacewar for Spacewar!

[4]Donkey Kong for Nintendo Switch https://www.nintendo.com/us/store/products/arcade-archives-donkey-kong-switch/

[5]Source: Pong - Wired journal, Smithorian magazine for Space Invaders, and Pac-Man - CNN

2.1.3 The Revolution of Home Consoles 1980s

While arcades reigned supreme, the advent of **home consoles** precipitated a paradigm shift in the gaming landscape, democratizing access to interactive experiences and ushering in a new era of gaming convenience. In 1977, Atari revolutionized the industry by introducing the Atari 2600, bringing arcade-like thrills into the living room and paving the way for a burgeoning market of home gaming enthusiasts. The early 1980s witnessed a surge of innovation with the emergence of iconic consoles such as the Nintendo Entertainment System (**NES**) and the **Sega Genesis**, igniting a fierce rivalry that fueled the evolution of gaming technology. Classic titles such as „**Super Mario Bros.**“ and „**The Legend of Zelda**“[6] (fig. 2.3) captivated audiences. At the same time, advancements in graphics, sound, and gameplay propelled the medium to new heights of popularity and sophistication.

Figure 2.3: Super Mario Bros. (left) and The Legend of Zelda (right)[7].

2.1.4 The Rise of PC Gaming 1990s

As home consoles gained traction, **personal computers** emerged as a formidable platform for gaming, offering a diverse and immersive gaming experience for enthusiasts. The 1990s heralded a golden era of PC gaming, characterized by groundbreaking titles such as „**Doom**“ and „**Quake**“[8], which pushed the boundaries of graphics and multiplayer capabilities. The proliferation of online multiplayer games like „**World of Warcraft**“[9] and „**Counter-Strike**“[10] (fig. 2.4) transformed PCs into vibrant hubs of virtual connectivity, fostering global communities of players and revolutionizing the concept of online gaming. With each technological leap, PC gaming solidified its status as a legitimate and influential force in the gaming industry, offering unparalleled versatility and customization options for players.

[6]The Legend of Zelda official page https://www.nintendo.com/en-gb/Games/NES/The-Legend-of-Zelda-796345.html

[7]Source: Super Mario Bros. - Nintendo, The Legend of Zelda - Nintendo

[8]Quake at RetroGames https://www.retrogames.cz/play_467-DOS.php

[9]World of Warcraft official page https://worldofwarcraft.blizzard.com/en-us/

[10]Counter-Strike page at Steam https://store.steampowered.com/app/10/CounterStrike/

[11]Source: Quake - Retrogames, and Researchgate for World of Warcraft

Figure 2.4: Quake (left) and World of Warcraft (right)[11].

2.1.5 The Mobile Gaming Revolution 2000s

As technological innovation continued to advance, the gaming landscape underwent a seismic shift with the emergence of **mobile gaming** in the late 2000s. The all-presence of smartphones and tablets democratized access to gaming, enabling millions to experience interactive entertainment on the go. Cultural phenomena such as „**Angry Birds**“[12] and „**Candy Crush Saga**“[13] (fig. 2.5) captivated audiences with their accessible gameplay mechanics and addictive design, transcending traditional gaming demographics and redefining gaming as an omnipresent form of entertainment.

Figure 2.5: Angry Birds (left) and Candy Crush Saga (right)[14].

[12] Angry Birds official page https://www.angrybirds.com/
[13] Candy Crush Saga official page https://www.king.com/game/candycrush

2.1.6 Embracing the future

As we look ahead into the future of gaming, we are met with a vast array of possibilities, which are being made possible by advancements in virtual reality (**VR**), augmented reality (**AR**), and artificial intelligence (**AI**). VR technologies like the Oculus Rift and the HTC Vive provide an unprecedented level of immersion, taking players to captivating virtual worlds, and redefining the limits of interactive storytelling.

Moreover, progress in graphics, processing power, and AI is allowing developers to create hyper-realistic and dynamic gaming experiences, blurring the lines between reality and fiction. Titles such as „**The Last of Us Part II**“[15] and „**Red Dead Redemption 2**“[16] showcase (fig. 2.6) how technology and artistry have converged to deliver compelling narratives and immersive gameplay experiences that resonate with players on a profound level.

Figure 2.6: The Last of Us Part II (left) and The Last of Us Part II (right)[17].

Additionally, the gaming industry is committed to diversity and inclusivity, which is leading to a more representative and welcoming gaming community. Video games feature a diverse array of characters and narratives that reflect the rich tapestry of human experience. This commitment to inclusivity is driving innovation and creativity, enriching the gaming landscape, and ensuring that the medium remains vibrant and relevant in an ever-evolving world.

[14]Source: Angry Birds - Angry Birds page, Candy Crush Saga - Google Play

[15]The Last of Us Part II on PlayStation https://www.playstation.com/en-cz/games/the-last-of-us-part-ii-remastered/

[16]Red Dead Redemption 2 official page https://www.rockstargames.com/reddeadredemption2

[17]Source: The Last of Us Part II - PlayStation, Red Dead Redemption 2 - Rockstar Games

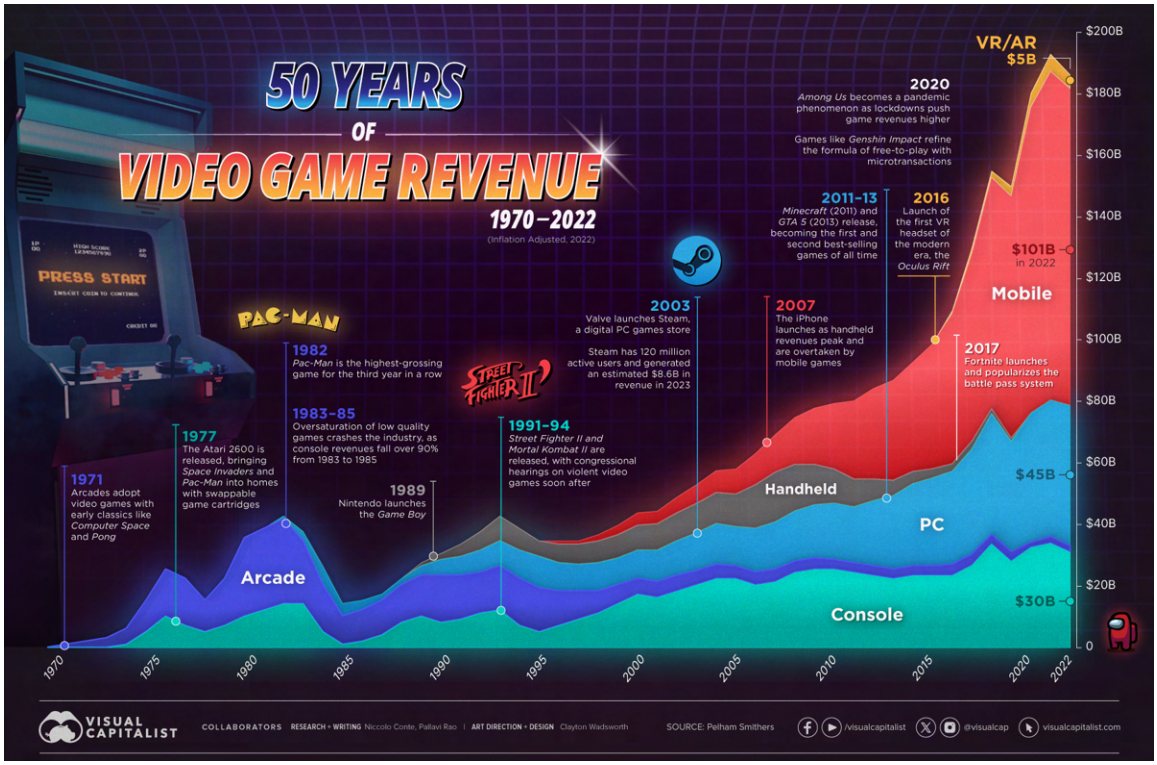

Figure 2.7: Brief history and capitalization of the game industry for 50 years. Source: [12]

In conclusion, the evolution of video games is a testament to human creativity, ingenuity, and technological prowess. We can see (fig. 2.7) that from the early experiments to the awe-inspiring marvels of the contemporary era, video games have transcended mere entertainment to become a cultural phenomenon that resonates with audiences of all ages and backgrounds.

2.2 UI in Games

As part of this thesis, we explored how gamers and games communicate with each other. Gamers use various input devices, including keyboards, mice, controllers, and Virtual Reality systems. But in this section, we will focus on how the User Interface (UI) helps players overcome the fourth wall, which is an imaginary barrier that separates the game from the players [2]. The primary function of UI is to guide players through the game, provide them with the necessary information, and help them achieve their goals.

The user interface (UI) of a game serves a vital role in enhancing the player's experience. It is not designed to grab the player's attention but rather to improve the game's playability and create a seamless, immersive experience that characterizes the best games.

The game's UI accomplishes this by providing cues to guide players in the right direction, whether these cues are subtle or obvious. These cues help players make informed decisions and feel more confident in their progress through the game.

2.2.1 Types of UI

In order to classify UI (user interface) components in games [2, 5], we will utilize two key concepts: narrative, which refers to the **story** that the game tells to the player [16]; and fourth wall, which is the invisible barrier that separates the player from the game **space**. The classification of UI components is based on the answers to two questions:

1. Does the component exist in the game's **story**?
2. Does the component exist in the game's **world space**?

Four categories then emerge from these answers:

- Diegetic
- Non-Diegetic
- Spatial
- Meta

The next table illustrates this categorization: (fig. 2.8)

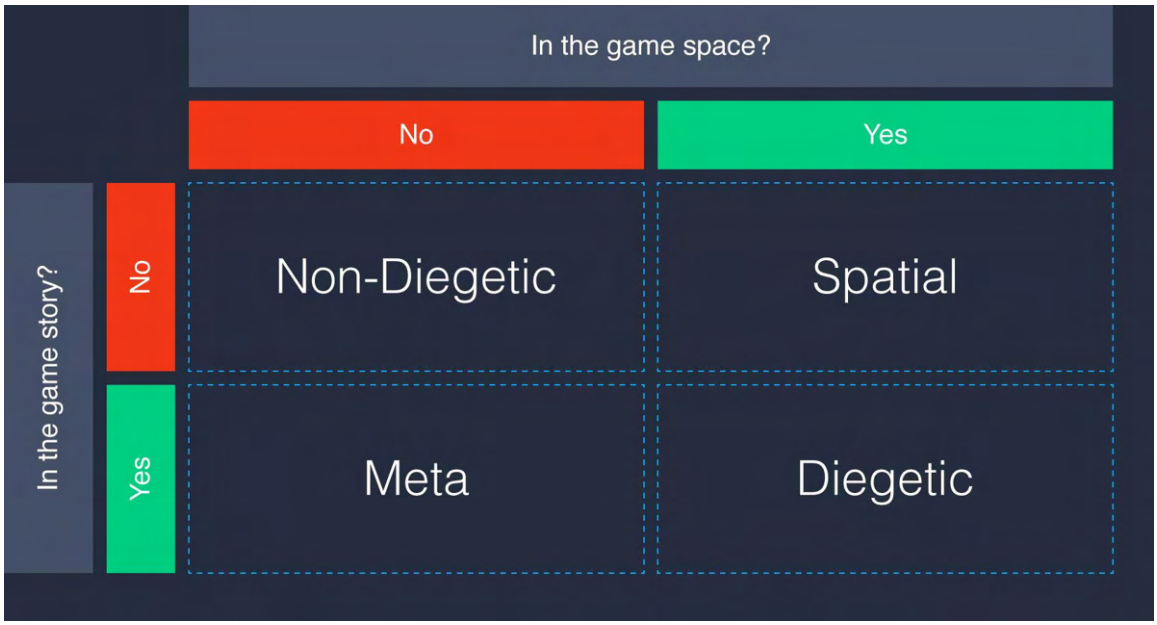

Figure 2.8: Types of UI in games [1]

Diegetic

- Does the component exist in the game's story? **Yes**
- Does the component exist in the game's world space? **Yes**

Diegetic user interface (UI) components are elements that exist within a game's story and space. These components can be seen or heard by characters in the game and serve a purpose in the narrative. Examples of such components include holograms, speedometers, timers, phones with visible screens, in-game gadgets used by the player's avatar, or futuristic UI overlay inside of the player's character helmet.

However, using these components can be challenging as they are often small in size and scale within the game's world, making them difficult for the player to view. To address this issue, some diegetic components can be toggled into a full-screen view, making them non-diegetic, because they do not occupy space in the game and are seen only by the player.

Non-Diegetic

- Does the component exist in the game's story? **No**
- Does the component exist in the game's world space? **No**

Non-diegetic user interface (UI) elements are essential elements that exist outside of a game's narrative and world. These elements are invisible to the characters within the game but provide players with crucial information about their performance and progress, such as point tallies, stat meters, health bars, and level maps. It is crucial to design and place these elements carefully to maintain gameplay flow and immersion, especially in fast-paced games where interruptions can disrupt player engagement.

Spatial

- Does the component exist in the game's story? **No**
- Does the component exist in the game's world space? **Yes**

In video games, spatial UI components are present in the game's world, however, they are not a part of the game's story. These components are visible to the player, but the character in the game is unaware of them. Spatial components play a crucial role in directing the player toward their goal or desired action and highlighting important landmarks or objects.

Examples of spatial UI components include hovering text labels near in-game objects, distinctive auras or outlines surrounding selected objects, or a racing line displaying the correct path for the player to follow.

Meta

- Does the component exist in the game's story? **Yes**
- Does the component exist in the game's world space? **No**

Meta UI components are elements that exist within the story of a game, but not within the game world itself. These components are designed to indicate events or conditions that affect either the player or the game world. For instance, they may represent damage through the accumulation of dirt or blood on the screen, or change the player's perception, such as blurring vision in response to in-game actions. Another example can be the scrolling dialogue text or a color filter that overtakes a player's field of view to indicate changing health levels.

2.2.2 Haptic feedback

Haptic feedback is an important aspect of UI/UX design that adds a new layer of interaction by involving the user's sense of touch [4]. It is also known as kinesthetic communication or 3D touch and comprises technologies that create a sense of touch by applying force, vibrations, or movements to the user [15].

This technology has various applications, such as creating virtual objects in computer modeling, controlling virtual entities, and enhancing remote control of machines and devices. Devices equipped with tactile sensors can measure the force exerted by the user on the interface. Haptic feedback technology enables players to perceive physical touches, vibrations, and movements through gaming controllers, vests, and other devices [20].

Initially, tactile feedback in games was limited to simple vibrations, referred to as the „rumble feature." However, over time, this technology has improved, offering increasingly realistic and enjoyable experiences. For example, Sony PlayStation consoles and controllers with rumble pack extensions allowed players to feel vibrations during in-game actions, enhancing immersion in the virtual world.

In recent years, there has been an increase in innovative haptic feedback technologies, such as VR haptic gloves, wristbands, and vests. These technologies provide players with even greater immersion in the virtual world by simulating various physical sensations.

2.3 Interesting consoles

This section will describe consoles that have interesting ways of interaction, usage of haptic feedback, and processing input and output.

2.3.1 Magnavox Odyssey

Let's start with the first existing console. The Magnavox Odyssey [14] was the first commercially available gaming console developed by Magnavox. It was released in 1972. This console featured eight simple games presented through a black-and-white television display signal. Among these games were tennis, hockey, and target shooting. The Magnavox Odyssey is considered a pioneer of the video game industry and paved the way for subsequent generations of gaming consoles.

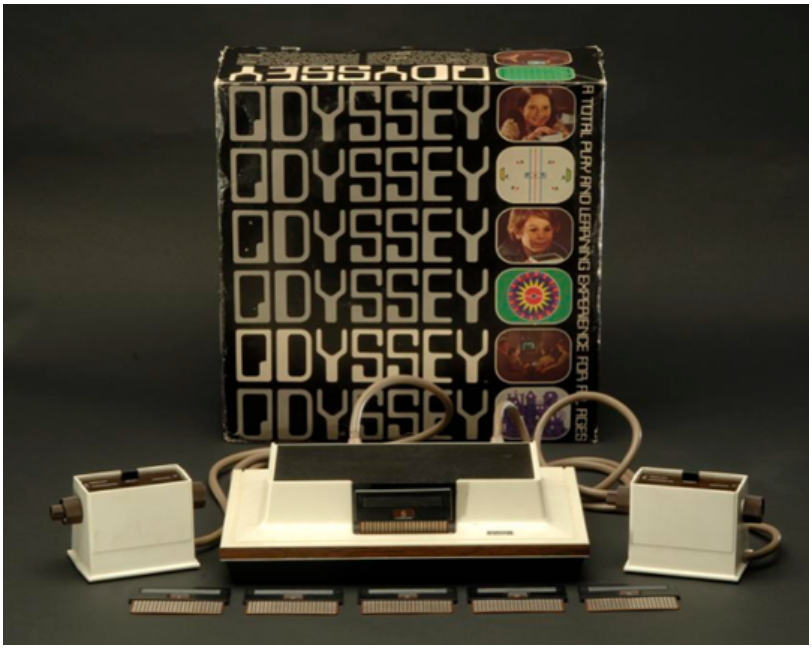

Figure 2.9: Magnavox Odyssey console[18]

The Magnavox Odyssey operated using simple electronic circuits that generated the video signal and allowed interaction with the player through basic controllers. Games were loaded using interchangeable cards that defined the rules and progression of each game. This method provided a foundational gaming experience that gradually became the basis for future gaming consoles.

2.3.2 Play Date

Playdate by Panic Inc.[8] introduces an innovative gaming console that offers users entertainment through new games and a unique design. Its main characteristic feature is a special black-and-white screen that is not backlit but highly reflective, providing a unique visual aesthetic. This device is equipped with a powerful processor, the ability to connect to Wi-Fi, and a surprisingly loud speaker, ensuring a comfortable gaming experience.

[18]Source: National Museum of American History

An intriguing element of the console is also the mechanical clicking „crank“, which integrates haptic feedback. The crank functions as an analog controller that can be flipped out from the side of the console. This controller allows users to control gameplay actions more precisely in certain games. While not every game utilizes the crank, some provide a fun and interactive experience thanks to its use. This crank serves as both a control input and a tactile feedback device. Players can physically turn the crank to interact with games, and the console provides haptic feedback in the form of resistance and vibrations, enhancing the overall gaming experience. Additionally, developers can creatively leverage the crank's haptic capabilities in their games.

Figure 2.10: Playdate console[19]

Playdate also offers the possibility for anyone to create their own game. Its development tools are free to download and do not require any special hardware, enabling any programmers, including unknown ones, to create and share their game titles. The company collaborates with a variety of talented and independent programmers to support diversity and creativity in game development for Playdate.

[19]Source: Kottke journal

2.3.3 Nintendo Switch

Another interesting example is the Nintendo Switch console. The Nintendo Switch is a gaming console that offers a unique combination of home and mobile gaming. Its main unit is a tablet monitor with a 6.2-inch LCD screen, which supports up to ten-point multi-touch capacitive sensing and integrates haptic technology from Immersion Corporation. This technology allows players to feel real tactile feedback when interacting with game elements, such as vibrations and resistance, enhancing overall gaming realism.

The Nintendo Switch [7] console also features stereo speakers located on the bottom of the unit beneath the screen, providing high-quality sound. Additionally, there are slots for game cards and a microSD card slot for memory expansion. The console includes volume control and power buttons, all located on the top of the device.

Figure 2.11: Nintendo Switch console with Nintendo Labo Toy-Cons[20]

In January 2018, Nintendo introduced Nintendo Labo, a new concept for an interactive experience that combines games with DIY cardboard kits called Toy-Con (fig. 2.11). Inspired by various gaming concepts, these kits allow players to create physical objects such as pianos, cars, or robots, which then connect to the Nintendo Switch console and Joy-Con controllers. While assembling the kits, players can discover how the technology works and create their own gaming experiences with different forms of Toy-Con creations. Nintendo Labo offers new opportunities for entertainment and learning for players of all ages and builds on Nintendo's long tradition of innovation.

[20]Source: Nintendo official news page

2.3.4 PS 5 and DualSense

The PlayStation 5 [9], or PS5, is Sony's latest gaming console, released in November 2020. It represents a significant leap in performance and features over its predecessor, the PS4. With powerful hardware, including fast SSD storage and advanced graphics, the PS5 delivers the most modern gaming experiences with stunning visual effects and smooth gameplay. With its innovative design and technologies, it has become a leading player in the gaming console market.

The DualSense controller is a key element of the PlayStation 5 console, bringing a new dimension to the gaming experience. With advanced haptic technology and dynamic adaptive triggers, players can feel the game more intensely and react to every gaming situation with variable intensity and tension. The built-in microphone also offers new communication and interaction possibilities in the online environment.

Figure 2.12: PlayStation5 console with DualSense controller[21]

The DualSense controller also has motion sensors, allowing players to utilize motion controls in selected games. This innovative technology brings new control and exploration possibilities to the virtual world, elevating the gaming experience to a higher level.

The decision to focus on creating a game specifically for the DualSense controller became the main goal of my bachelor's thesis. This work will lead to exploring the possibilities offered by this innovative controller and how it can influence players' interaction with the gaming environment.

[21]Source: Polygon journal

2.3.5 Virtual Reality

Headset

In the concluding section of this chapter, we will concentrate on virtual reality headsets that offer captivating gaming experiences to users. There are a number of notable devices that deserve attention and introduce innovative technologies to the world of VR (Virtual Reality). To learn more about Virtual Reality, please visit [13].

The Meta Quest 3, created by Meta, is one of the most remarkable examples of virtual reality headsets[22]. These goggles are equipped with advanced haptic feedback technology that allows users to experience a more immersive virtual world. The haptic feedback feature makes it possible for players to feel various touches, vibrations, and movements in space, thereby enhancing their emotional involvement in the game. With the Meta Quest 3, you can enjoy truly interactive and realistic gameplay, making virtual reality all the more appealing.

Figure 2.13: Meta Quest 3 VR goggles[23]

Another significant VR headset is the PS VR2[24] for the PlayStation gaming console. These goggles feature an advanced haptic feedback system that provides users with immersive sensations, such as the feeling of a character's heartbeat or the speed of objects nearby. In addition to this, they offer other technological innovations, such as dual OLED panels with 4K resolution and Fresnel lenses to reduce ghosting. The PS VR2 delivers users a genuinely immersive and realistic gaming experience, enhancing gameplay and increasing the attractiveness of the virtual environment.

Vest

In addition to headsets, there are virtual reality vests that enhance the user experience by conveying physical sensations into the virtual environment. These vests are equipped with vibration motors or pressure balloons that react to in-game events, such as shootings,

[22]Meta Quest 3 page https://www.meta.com/quest/quest-3/
[23]Source: Gameshub journal
[24]PS VR2 page https://www.playstation.com/cs-cz/ps-vr2/

impacts, or touches. This allows users to experience sensations like impact, pressure, or vibrations, contributing to greater realism and immersion in the game. Some vests even allow synchronization with haptic effects in other virtual reality devices, such as VR headsets or controllers.

A VR vest worth mentioning is the Tactsuit X40 by bHaptics[25] (fig. 2.14), which is a haptic vest with 40 motors that deliver precise feedback in line with actions in VR games.

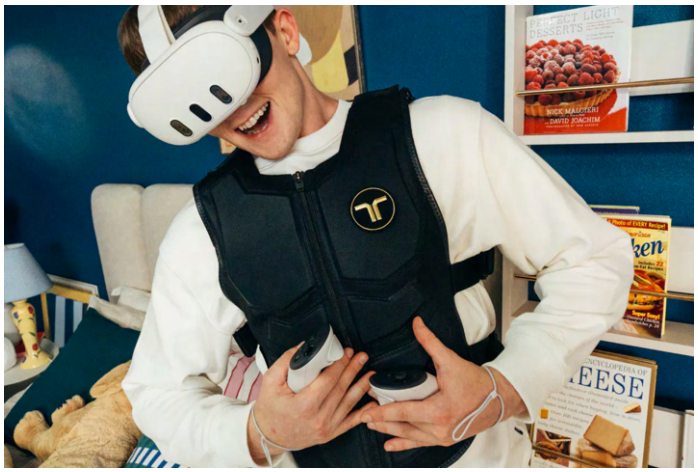

Figure 2.14: bHaptics Tactsuit X40[26]

An excellent example of technological innovation in this domain is the Teslasuit[27] (fig. 2.15), which is not just a vest but a full-body suit that utilizes electro-muscle stimulation (EMS) and transcutaneous electrical nerve stimulation (TENS) to recreate a range of real-life sensations. The suit also features 14 built-in sensors that offer full-body motion capture to monitor the movements and positioning of users. This advanced technology has applications beyond gaming and can be used in various simulations, training, and educational scenarios.

Figure 2.15: Teslasuit image (left), and a scheme of haptic stimulations (right)[28].

[25] bHaptics Tactsuit X40 https://www.bhaptics.com/tactsuit/tactsuit-x40
[26] Source: bHaptics
[27] Teslasuit https://teslasuit.io/products/teslasuit-4/

Gloves

Special gloves are an accessory that can be used in conjunction with VR headsets, serving as an alternative to VR controllers. By eliminating the need for controllers, a greater level of immersion can be achieved. The combination of finger-tracking gloves with haptic feedback technology is a powerful one, allowing the user to accurately feel 3D objects in their VR environments, providing vibrations to fingers to simulate the presence of an object.

There are some noteworthy gloves available in the market, such as the ones made by Manus[29] Prime 3 Haptic XR[30]. These gloves are lightweight and very easy to integrate and use. Another option is the Quantum XR[31], which is a more complex model. It provides better high-fidelity finger tracking by using drift-free fingertip tracking sensors.

HaptX Gloves G1[32] are one of the most advanced gloves at the moment. The construction of these gloves is larger and less compact than other comparable VR gloves, but they incorporate more haptic motors and bring high-precision motion capture, that gives a more realistic haptic experience in return.

Figure 2.16: Manus Prime 3 Haptic XR (left), Manus Quantum XR (middle), and HaptX Gloves G1 (right)[34].

[28]Source: Teslasuit - Teslasuit
[29]Manus official page https://www.manus-meta.com/vr-gloves
[30]Prime 3 Haptic XR https://www.manus-meta.com/products/prime-3-haptic-xr
[31]Quantum XR https://www.manus-meta.com/products/quantum-xr-metagloves
[32]HaptX Gloves G1 https://g1.haptx.com/learnabout
[34]Source: Manus Prime 3 Haptic XR and Quantum XR - Manus, HaptX for HaptX Gloves G1

Chapter 3

Game design

This chapter will explore the inspirations behind the game concept and the development of the game design document. We will discuss the games that served as inspiration for the project and how these influences shaped the design and vision of our own game. Additionally, the chapter will detail the process of crafting the game design document, outlining key elements and features based on the identified inspirations.

3.1 Inspiration

Keep Talking and Nobody Explode

When considering creating my own game for the bachelor's thesis, I used two games as a reference. The first one was Keep Talking and Nobody Explodes (KTANE)[1]. The game focuses on cooperation and communication among players and requires the participation of at least two players. One player takes on the role of the „defuser," actively playing the game on a device supporting various control methods, while the other players, known as „experts," have access to a manual with instructions for defusing the bomb.

Figure 3.1: Example of level in Keep Talking and Nobody Explodes (KTANE)

[1]Keep Talking and Nobody Explodes game page https://keeptalkinggame.com/

The bomb consists of several modules, each of which can be deactivated independently of the others. The defuser must describe each module of the bomb, and then the experts use the manual to determine the procedure for its defusal. There are also unstable modules that require regular attention. The bomb has a countdown timer and a maximum allowable number of errors. Each mistake accelerates the bomb's timer.

Modules contain complex puzzles and tasks, such as mazes or word puzzles. Some modules have multiple stages, with each subsequent stage depending on the previous ones. Deactivating some modules may depend on the bomb's status, such as the number of errors, the bomb's serial number, or the battery status.

The game includes various difficulty levels with different numbers of modules, time limits, and maximum error counts. Each game procedurally generates the bomb and its modules. Players can also create their challenges by specifying the number of modules, time limits, and error counts.

Astro's Playroom

The second game was Astro's Playroom[2], an exclusive title for PlayStation 5. It serves as a sequel to the successful Astro Bot Rescue Mission and acts as a great showcase of the capabilities of the new DualSense controller. Players take on the role of Astro Bot and explore three-dimensional worlds filled with platforming levels, puzzles, and unique challenges.

Each world in Astro's Playroom represents different components of the PlayStation console and has its environment and characteristic features. For example, GPU Jungle represents the graphics processor, Cooling Springs embodies the cooling system, SSD Speedway refers to the storage, and Memory Meadow symbolizes the memory modules. Players explore these worlds, complete tasks, and collect various collectible items.

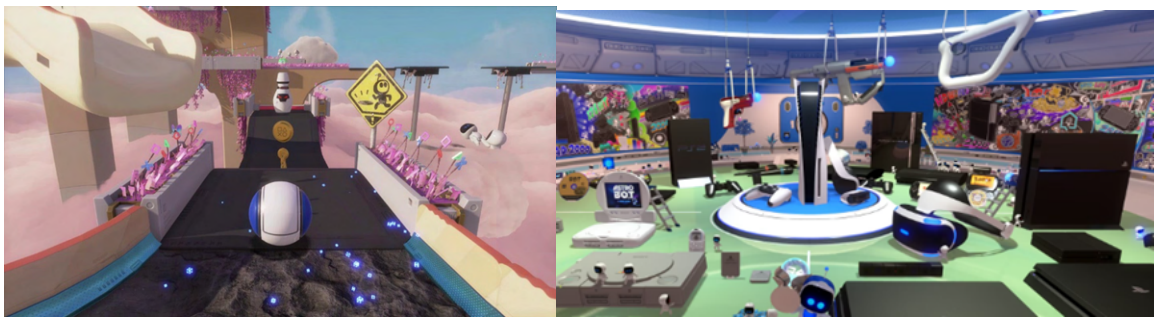

Figure 3.2: One of the levels on the left and central hub on the right in Astro's Playroom

The game also offers enjoyable mini-games and competitions. At the end of each world, players face a boss inspired by PlayStation history, adding another dimension and challenge to the game. Upon completing all worlds, players unlock a secret fifth world where they face additional challenges and discover additional secrets.

Overall, Astro's Playroom is not only an entertaining and engaging game but also an excellent way to explore the innovative features of the gaming controller.

[2]Astro's Playroom game page https://www.playstation.com/en-cz/games/astros-playroom/

3.2 Game Concept

Cut the red! is the name of our 3D puzzle game designed exclusively for the PlayStation 5's DualSense controller. Players take on the role of bomb defusal specialists tasked with disarming intricate devices using the unique features of the DualSense controller. The bomb consists of different modules (fig. 3.3), presenting a distinct challenge, ranging from navigating a labyrinth to answering timed questions and cutting wires precisely. Each bomb slot requires the use of different sensory inputs: haptic feedback, adaptive triggers, and motion sensing. The game is designed to test players' coordination, problem-solving skills, and ability to manage stress under pressure in order to defuse the bomb successfully.

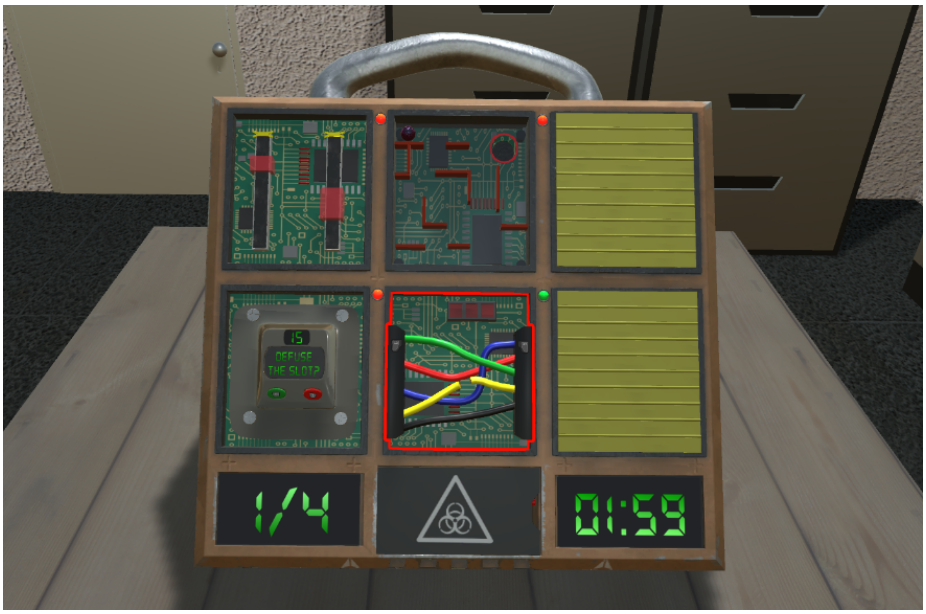

Figure 3.3: The bomb in the game

Modules

The game comprises several modules, each representing a different bomb scenario. Some examples include:

- **Haptic Labyrinth:** Players navigate a maze by feeling vibrations that guide them and avoid walls by sensing vibrations intensify as they get closer to obstacles.
- **Adaptive Wire Cutting:** Players must cut the correct wire based on haptic feedback and trigger resistance to prevent detonation.
- **Balance Challenge:** Utilizing the built-in motion sensor, players must maintain balance while performing precise tasks.
- **Quick questions:** Time-sensitive questions require quick responses to defuse or detonate the bomb.

DualSense Controller Integration

The game leverages the unique features of the DualSense controller:

- **Adaptive Triggers:** Mimic the resistance of wire cutting, or provide tactile hints, adding a tactile dimension to gameplay.
- **Haptic Feedback:** Provides tactile cues during different slot defusion to guide player actions.
- **Gyroscope:** Controls ball movement in one of the modules.

Gameplay Flow:

- Players start with an overview of the bomb scenario, highlighting key components and challenges.
- They select a module and engage with the bomb using the DualSense controller's specific functionalities.
- Each module has unique objectives and failure conditions, such as detonation upon incorrect actions or exceeding time limits.
- Successful completion of modules progresses the game towards the final bomb defusal.

Target audience

Cut the red! is designed for players seeking a unique and immersive experience that showcases the capabilities of the DualSense controller. It appeals to gamers who enjoy puzzle-solving, coordination, and tactile interactions.

Platforms

The best platform for the game is PlayStation 5, which utilizes the advanced features of the DualSense controller and provides a smooth user experience. However, within the framework of this project, this is not possible to use PS5, because it requires a PlayStation Partnership[3], which is not available for us at this moment.

The target platform for this project is a Personal computer (PC). Using specific packages and tools it is possible to use DualSense on a PC, but the range of possibilities is not as vast as on PlayStation. However, some basic ideas are quite realizable.

Summary

Cut the red! offers a thrilling and engaging gameplay experience combining puzzle-solving and innovative controller mechanics. By integrating the DualSense's unique features, including haptic feedback, adaptive triggers, and motion sensing. The game challenges players to think critically and act decisively in high-pressure bomb defusal scenarios. Each module is designed to make the player feel as if they are physically interacting with the bomb, enhancing immersion.

[3]PlayStation Partnership https://partners.playstation.net/

Chapter 4

Technologies

In the upcoming chapter, we will delve into the game development process, focusing on Unity as the primary platform. We will discuss the Unity game engine, and other used tools, highlighting their roles and functionalities within the context of the project. This chapter will provide insights into the technologies employed to bring the game concept to life and enhance the overall user experience.

4.1 Game development

Game development is the process of creating video games. It includes designing, developing, and producing interactive entertainment software. The development of games requires a multidisciplinary approach that combines various elements such as software engineering, game design, visual arts, audio production, and storytelling to create immersive experiences for players. The process of game development starts with a concept or idea, which is then refined and expanded through various stages of planning and production. This process involves translating creative vision into technical implementation, ensuring the game is enjoyable and functional across different platforms and devices. The most common description of game development stages: [10]

- **Planning** begins the development cycle, where brainstorming sessions and market research are conducted to define the game's concept, style, and vision. This phase also involves creating a preliminary budget, assembling the development team, and outlining technical requirements, setting the groundwork for subsequent stages.
- **Pre-production** is a phase focused on detailed documentation and initial creative work. A comprehensive Game Design Document (GDD) detailing the game's mechanics, characters, and storyline is created here. Concept art, prototypes, and technical requirements are also defined, laying the foundation for the game's production phase.
- **Production** is where the game truly takes shape, with coding, asset creation, level design, and integration of audio-visual elements. This stage emphasizes continuous optimization to ensure smooth performance across platforms and devices.
- **Testing** follows production, involving QA testing, debugging, performance optimization, compatibility testing, and regulatory compliance to identify and address any issues before launch.

- **Pre-launch** activities involve final testing, marketing, setting up support systems, and community engagement to build anticipation and awareness around the game.
- **Launch** marks the public release of the game, involving distribution, monitoring, addressing immediate issues, and analyzing user feedback.
- **Post-launch** focuses on ongoing support, updates, and improvements, including monitoring player feedback, addressing bugs, releasing patches and updates, and planning additional content to sustain player engagement.

The game development process described reflects a structured approach that balances creativity with technical execution, emphasizing collaboration, thorough testing, and continuous improvement to deliver high-quality gaming experiences that resonate with players and meet business objectives. Each stage contributes to the game's overall success, from initial ideation to post-launch support and beyond. More about the game development life cycle can be learned here [11]

4.2 Unity

Unity is a versatile cross-platform tool used by game developers to create interactive 2D and 3D content. It is a popular choice among creators due to its powerful options and ease of use. Unity comes with a wide range of features, including a robust game engine, physics simulation, audio management, user input processing, scripting languages, artificial intelligence, and network capabilities. It is also considered one of the most beginner-friendly game engines, which is why it was chosen for this project. In this section, we will explain the basic concepts of Unity that are necessary to understand the thesis. All needed information can be found in the online Unity manual [19].

Scene

A Scene in Unity refers to a specific environment or level in a game or application. It's made up of game objects such as characters, environments, cameras, lights, and other elements that define a particular section of the game world. The purpose of scenes is to organize and manage different parts of a project. Unity developers can dynamically load and unload scenes during gameplay, creating seamless transitions between different levels, menus, or cutscenes. Each Scene has its own hierarchy of game objects, which can be viewed and edited in the Unity Editor. This hierarchy enables you to organize, manipulate, and structure the objects within a scene.

Game Object

GameObject is a fundamental building block used to represent entities or objects within a scene. It serves as a container for components that define its behavior, appearance, and functionality. Game objects can represent characters, items, obstacles, cameras, lights, and more. Every element of the game is a GameObject, rather it is some object in the game world that a player can see, or a camera, or even a script that runs in the background. GameObjects are defined and controlled by Components. Components are attached to GameObjects to give them specific attributes or behaviors, providing a flexible and modular approach to game development.

Component

Components are essential elements of GameObject that define its functionality within a game. Each GameObject has at least one component by default and it is Transform. This component defines the position, rotation, and scale of the object in the game scene. However, there are a lot of different components that can be combined to create complex interactions. Here are a few of them used the most frequently.

Renderer

There are a few different types of Renderers in Unity, but their main purpose is to define how objects should appear visually, based on their geometry and materials. It is a fundamental component used to display graphics and meshes on the screen. For 2D graphics, a SpriteRenderer is usually used. It uses a sprite to visually represent an object. For rendering 3D models usually used MeshRenderer (fig. 4.1).

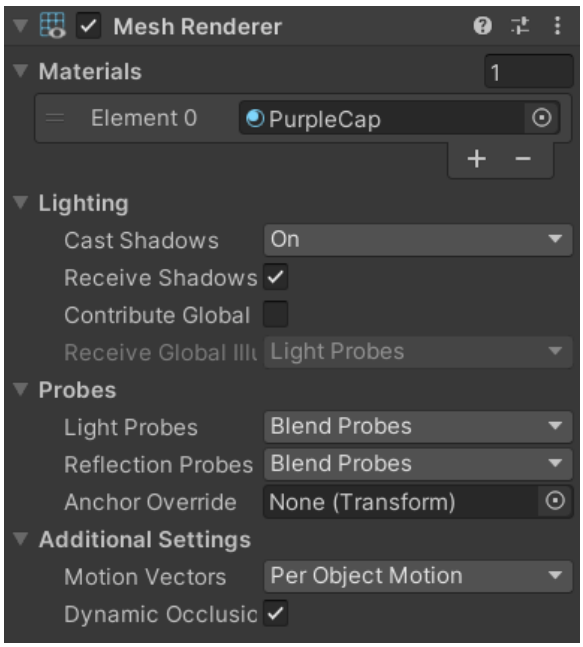

Figure 4.1: Example of a Mesh Renderer Component

Collider

The Collider component is used to define the physical shape and size of GameObjects for the purposes of collision detection and physics interactions. Colliders can be attached to GameObjects to create boundaries that interact with other objects in the scene. There are various types of colliders available in Unity (fig. 4.2), such as BoxCollider, SphereCollider, CapsuleCollider, and MeshCollider, each suited for different shapes and forms of collision detection.

Rigidbody

The Rigidbody component is used to simulate physics interactions for GameObjects by applying forces and responding to collisions. Rigidbody enables realistic physics behavior,

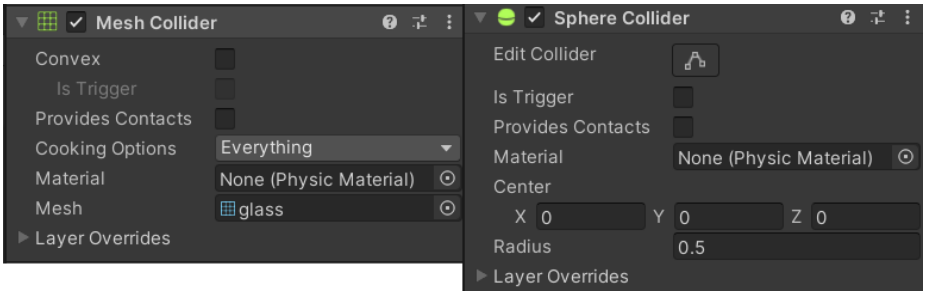

Figure 4.2: Example of a Mesh and Sphere Collider Component

such as gravity, velocity, and mass. This allows GameObjects to move and interact with other physics-enabled objects in the scene. Rigidbody (fig. 4.3) properties like mass, drag, and angular drag can be adjusted to control how GameObjects respond to forces like gravity or applied impulses.

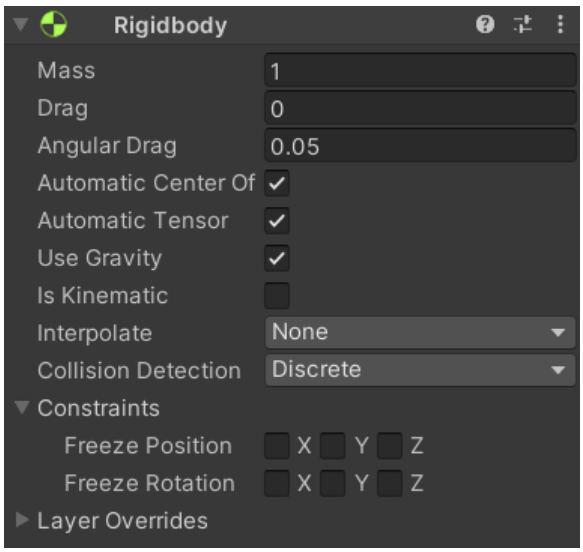

Figure 4.3: Example of a Rigidbody Component

Script

The Script component is a fundamental tool in Unity for implementing game logic. Scripts in Unity are written in C# and provide the ability to control how GameObjects behave and interact within the game environment. By attaching a Script component to a GameObject, we can define functions, variables, and event handlers that dictate the GameObject's behavior, including movement, input handling, animations, and more. Unity scripting involves several important concepts, and here are some of them.

MonoBehaviour is a base class in Unity that allows you to create scripts to control GameObjects. Scripts derived from MonoBehaviour can interact with GameObjects, respond to events, and execute custom behaviors. MonoBehaviour provides a range of methods that Unity calls automatically in response to various game events. The most commonly used of them:

- **Awake():**This method is called when an enabled script instance is being loaded and is used to initialize variables or states before the application starts.
- **Start():**This method is called on the first frame when a script is enabled. It's typically used for initialization tasks.
- **Update():**The Update method is called every frame and is used for regular updates or computations that need to run continuously.
- **FixedUpdate():**This method is used for physics-related updates and is called at fixed time intervals, making it suitable for physics calculations.

Another base class in Unity is ScriptableObject, which is a serializable data container that allows developers to store and manipulate data in a way that can be reused and shared across different GameObjects and scenes. They are often used to create custom asset types such as settings, configurations, or game data that can be modified in the Unity Editor without needing to attach them to specific GameObjects.

4.3 Input System

The Input System (fig. 4.4) in Unity is a powerful and flexible framework designed to handle player input more efficiently and with greater flexibility compared to the legacy input system. It provides a unified way to manage input from various devices such as keyboards, gamepads, touchscreens, and more, allowing developers to define and manage actions and control schemes in a centralized manner.

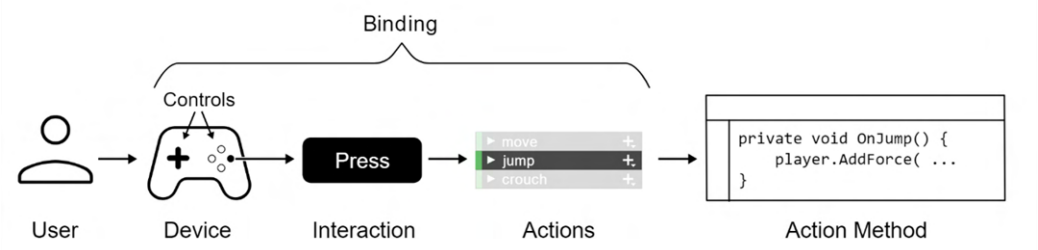

Figure 4.4: Input System concept

The Input System introduces the concept of Input Actions, which represent specific actions that players can perform in the game, such as „Jump," „Fire," „Move," etc. Control Schemes define mappings of input bindings (e.g., keys, buttons, touch controls) to input actions for different input devices or scenarios.

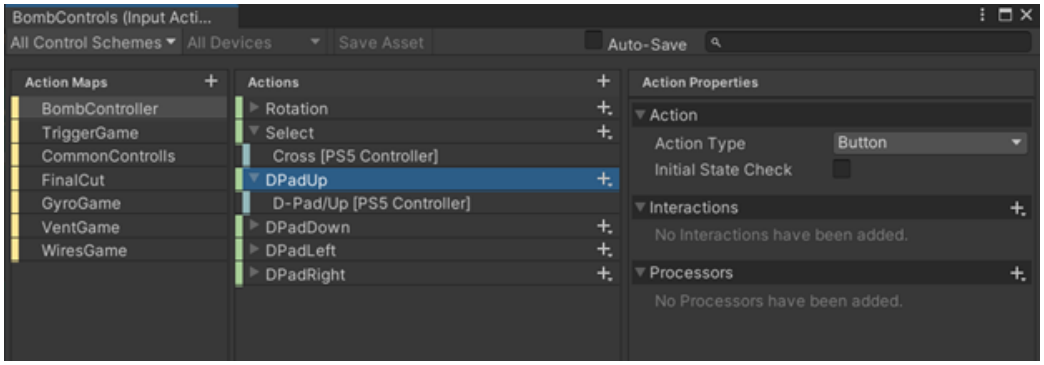

Figure 4.5: Example of Input Action file with Action Maps from the game

The Action Map concept is especially useful for our project. Action Maps group related input actions together based on specific game contexts or modes. For example, we used it to separate actions for menu navigation and different gameplay scenarios and modes (fig. 4.5). Switching between action maps allows the game to activate different sets of input actions based on the current game state or player interaction. This provides greater flexibility in managing input behavior.

4.4 UniSense

UniSense[1] is an open-source package that allows developers to use the DualSense controller for PS5 in Unity apps on Windows. While the Input System already supports the controller, it does not support all of its features, such as the gyroscope or different trigger modes and settings. UniSense defines a PS5 template for InputSystem, making it easier and more seamless to interact with the controller. The tool also provides direct access to features like adaptive trigger settings, haptics, and trigger resistances, which are important for fluent game development and can greatly enhance the user experience.

4.5 Graphics

All 3D objects were created by Arsenii Petrov[2] exclusively for this game. I think that visuals unique and original are a crucial part of any game, and it is especially important to catch the attention of players in the stream of game diversity nowadays. In a crowded market, distinctive visuals help indie games stand out from the competition. With so many games available across various platforms, visually striking graphics can catch the attention of players browsing through digital storefronts or social media feeds. This uniqueness can spark curiosity and prompt players to explore the game further.

One more important aspect of indie game development is creating original graphics that contribute to the game's overall identity and branding. A strong visual identity helps create a memorable and cohesive experience for players, especially when the graphics match the game's theme, tone, and mechanics. This enhances immersion and strengthens the game's storytelling capabilities, which can be harder to achieve by using pre-made assets available on online marketplaces or asset stores.

[1] UniSense https://github.com/nullkal/UniSense
[2] Arsenii Petrov LinkedIn https://cz.linkedin.com/in/arsenii-petrov-908790151

In the development process, a combination of tools was utilized to create models, textures, and assets for the game. Cinema 4D and Blender were employed for modeling tasks, offering versatile options for sculpting and shaping various in-game elements. To bring these models to life with vibrant textures and details, Substance 3D Painter proved instrumental. Following the creation and texturing of models, all assets were integrated into Unity.

When it comes to the User Interface, the default Unity components were used, in combination with Electronic Highway Sign font and DS-Digital font taken from the Dafont page[3].

[3]DS-Digital on the Dafont page https://www.adobe.com/cz/products/substance3d-painter.html

Chapter 5

Implementation

The **Bomb** 5.1 is a crucial **GameObject** within the game scene, acting as the game's centerpiece. It consists of six module slots, a timer, a slot completion counter, and the final wire slot, which opens only when all modules are defused. All these components will be described further in this chapter. To ensure a clear separation between logical and visual components, the Bomb is simply an object that consists of two attached scripts that manage its overall logic. Its other components are its children. **BombInput** and **BombManager** are essential scripts that control the bomb logic. **BombInput** manages user inputs while **BombManager** manages game completion, timer, counters, and other logic.

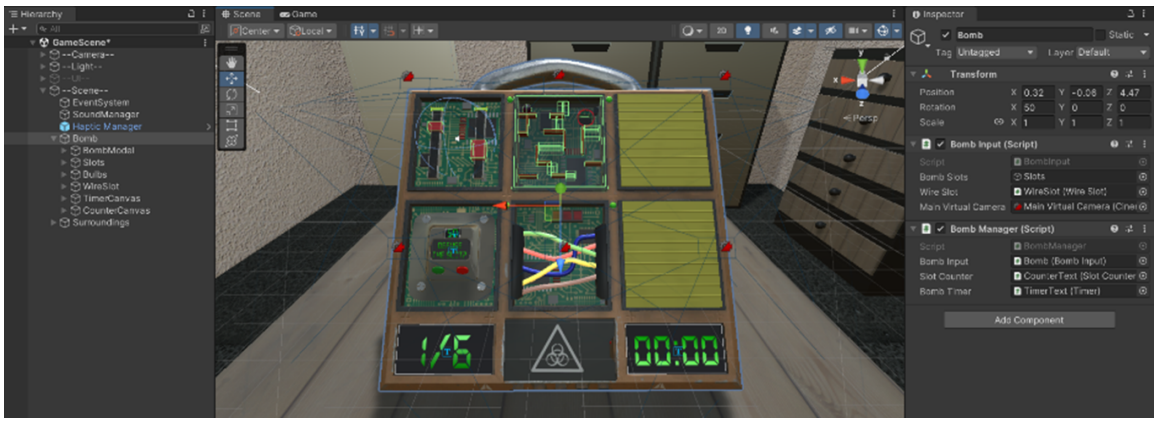

Figure 5.1: The Bomb GameObject in the game scene

5.1 Bomb Input

When it comes to managing user inputs, we utilized the Input System and created an Input Actions File named **BombController** . This file stores all the Action Maps and Input Actions required to effectively control the bomb and its various slots.

To navigate the bomb and select different slots, we created an action map with the same **BombController** name. This action map is specifically designed to be used only in the default bomb state.

To listen for commonly used Input Actions that are relevant to all states and slots, we created a **CommonControls** Action Map. This Action Map is responsible for enabling the return from a slot to the default bomb state and toggling the pause functionality.

The **BombInput** script is a fundamental component of the game, responsible for managing and processing player inputs to facilitate intuitive interaction with game elements. This script is intricately designed to integrate with the DualSense controller, leveraging Unity's Input System to translate controller inputs into meaningful actions within the game.

Upon initialization, the **BombInput** script configures **BombControls** to handle various input actions, including directional navigation using the controller's DPad inputs. This feature allows players to move a cursor across bomb slots represented by game objects in the scene. The script meticulously manages lists of bomb slots and associated **Outline** components, enabling dynamic visual feedback by highlighting selected slots as players navigate through the game environment.

The **BombInput** script also implements event-driven programming techniques, triggering events (**OnPause**, **OnNavigate**) to communicate critical game actions to other systems. For instance, the PausePerformed method invokes the **OnPause** event to pause the game when prompted by player input.

Furthermore, the script handles specific input actions like wire cutting (**OnNavigate**), ensuring that actions are executed only when conditions are met within the game logic. This approach not only enhances gameplay responsiveness but also maintains clarity and immersion by providing players with clear visual cues and responsive controls during the bomb defusal process.

5.2 BombManager

The **BombManager** script serves as the central orchestrator of the game, responsible for managing the game's progression, completion, and pause/resume functionality.

The script tracks the completion status of bomb slots, updating the progress displayed by the slot counter based on the number of completed and selectable slots. It subscribes to events triggered by bomb slots (**OnSlotStateChanged**) and wire cutting (**BombWireCut**), responding accordingly to progress updates and game outcomes.

During gameplay, the **BombManager** continuously checks the bomb timer's remaining time. If the timer reaches **zero** without the bomb being defused (**TriggerBombExplosion**), the game concludes with a detonation scenario, invoking the **GameOver** method to handle the game-over state appropriately.

The **GameOver** method disables player controls (**BombControls**) upon game conclusion, triggering events (**OnBombDefused**, **OnBombDetonated**) and displaying corresponding UI screens (**GameOverUI**) based on the game outcome (defusal or detonation). This method effectively manages game states and ensures a seamless transition between gameplay phases.

The script also implements pause/resume functionality (**HandlePause**) to allow players to pause and resume the game at any time. When paused, the game disables player controls and triggers the **OnGamePaused** event, while resuming re-enables controls and triggers **OnGameResumed**. These methods adjust the game's time scale (**PauseGame**, **ResumeGame**) to halt or resume game logic execution accordingly.

5.3 Haptic Manager

The haptic manager is the next crucial element of the game, controlling all haptic effects within it. Implementation is based on the online tutorial of **Iain McManus** [6]. The script stores a list of active haptic effects and provides static methods, including **PlayEffect** and **StopEffect**, for triggering and stopping haptic feedback effects that are implemented as scriptable objects. The **PlayEffect** method creates a specified haptic effect at a given location in the game world, while the **StopEffect** method removes an active haptic effect from the list of currently playing effects.

During the game loop, the **HapticManager** processes active haptic effects by updating the controller motor speeds. The low frequency is used for the left motor, and the high frequency is used for the right motor based on the effect's behavior, type (continuous or one-shot), intensity, and form. Active effects are updated in each frame, and if an effect is finished, it is removed from the list of active effects.

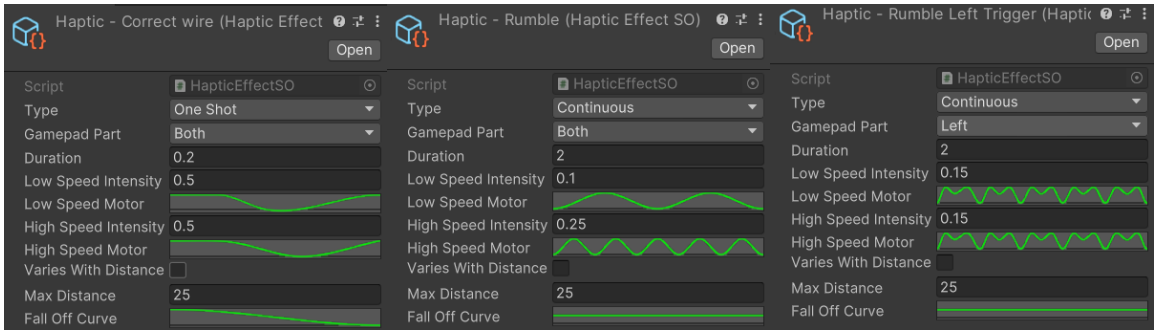

Figure 5.2: Examples of haptic effect ScriptableObject

Haptic feedback effects are implemented as **ScriptableObjects** (fig. 5.2) that allow haptic effects to be defined and configured in a data-driven manner using serialized fields. This makes it easy to adjust parameters such as intensity, duration, vibration patterns, and impact side on the controller directly within the Unity Editor without modifying the code. Once defined, this **scriptable objects** can be reused across different game elements or scenarios, promoting efficient asset management and reducing redundancy.

By combining these effects with the haptic manager, we can create unique game mechanics and interactions. For instance, creating a rumble zone that activates continuous rumble feedback when a player enters it or one-shot feedback when a specific event occurs.

5.4 Bomb slots

The upcoming chapter discusses individual **bomb modules**, which are different minigames that a player must solve to defuse the bomb. Although each module has different mechanics and represents different ideas, they all inherit the same base class, called **BombSlot**. This is done to simplify the code and to benefit from code reusability. The parent class stores common information for all slots, such as whether a slot is empty or not, whether it is selected, sets the slot's completion (which triggers events for other systems and managers), enables slot-specific controls, and handles slot selection and camera.

5.4.1 Trigger minigame

The **Trigger Game** (fig. 5.3) is the first bomb module that was created. Its idea is to utilize the adaptive triggers of the **DualSense** controller and provide a gameplay scenario where players must manipulate the left and right **triggers** to achieve specific positional goals within defined thresholds. The gamepad assists players in achieving this goal by applying force back to triggers and rumbling when players get closer to the desired zone.

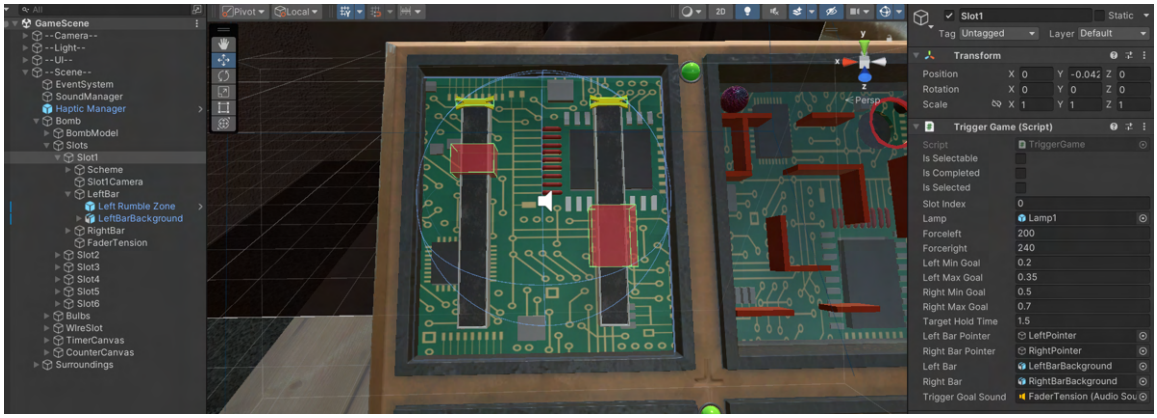

Figure 5.3: The Trigger minigame in the game scene

To achieve this, the module utilizes the **UniSense** library to control the resistance of triggers and the Haptic manager to generate rumble zones. When the trigger position corresponds to the designated area, the rumble is triggered. To represent the trigger level visually, two vertical bars with pointers were created, along with goal areas corresponding to the rumble zones.

When the minigame is initialized (**Awake**), critical parameters such as trigger forces and goal zone positions are set up for each trigger separately. These parameters determine the feedback force and target ranges for the trigger-based gameplay challenge. Whenever the slot is selected (**HandleSlotControls**), the HandleSlotControls method enables interaction with the player's DualSense controller, specific only for this slot. The script then interfaces with the UniSense library to dynamically adjust the resistance and position of the left and right triggers based on player input before starting the main coroutine for this slot.

The **UpdateTriggerGame** coroutine is a key feature of the **TriggerGame** script. It continuously runs in the background while the gameplay scenario is active. The script monitors the current positions of the left and right triggers and maps them to graphical pointers within visual bars displayed in the game interface. The coroutine evaluates whether the trigger positions fall within specified goal ranges and provides haptic feedback by providing determined force back to triggers and enabling the rumble based on specifications of the rumble zone (fig 5.4) while tracking the player's progress toward completing the module task.

When both trigger positions align with their respective goal thresholds within a defined hold time, the gameplay challenge is considered completed. This triggers positive audio-visual feedback, changes the light bulb indicator color to green, and progresses the game state accordingly.

Furthermore, the **HandleReturn** method manages player actions related to exiting the gameplay scenario by returning to the default bomb state. It resets the trigger states

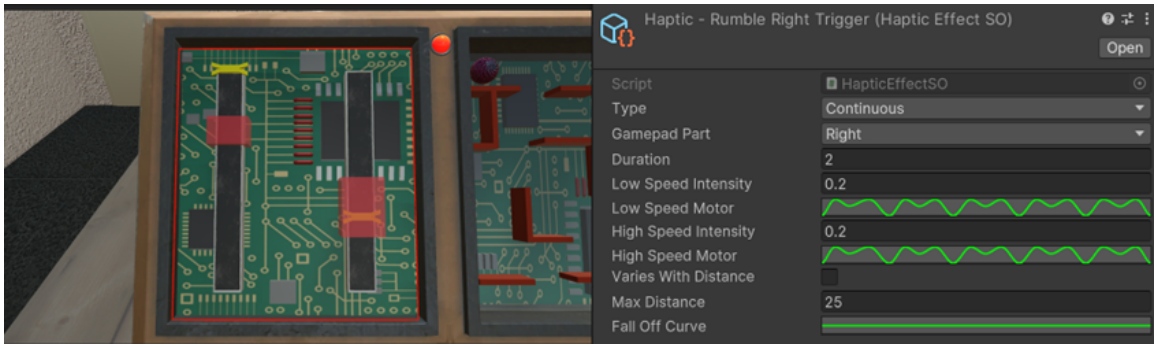

Figure 5.4: Rumble effect for right trigger activates when gets into the desired zone

to default (no resistance) and terminates the **UpdateTriggerGame** coroutine, effectively concluding the interaction sequence.

5.4.2 Gyro minigame

The next module created was the **Gyro Game** (fig. 5.5). It's a fun labyrinth minigame that utilizes the gyroscope and haptics of the DualSense controller. The goal is to get the ball to the end of the labyrinth, but the maze is built of both visible and invisible walls. To achieve this, players must navigate with help from the controller, which gives a haptic response when a player gets near an invisible wall.

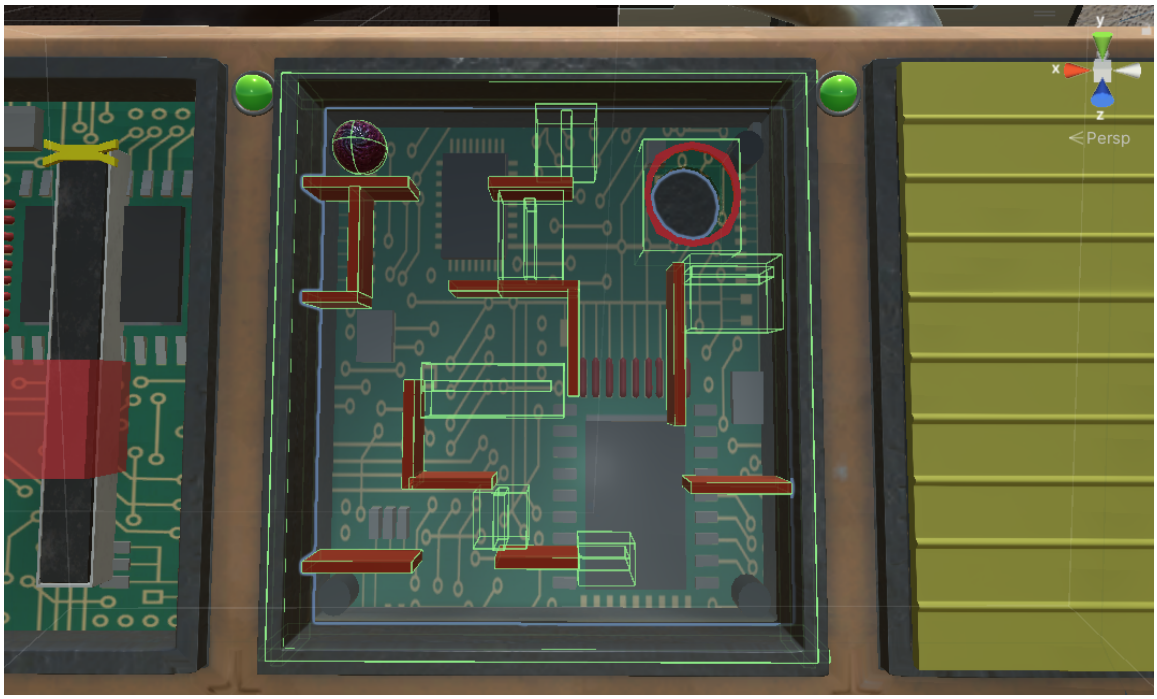

Figure 5.5: The Gyro minigame in the game scene

The key elements of the **Gyro Game** script include the integration of gyroscope input from the DualSense controller and the manipulation of physics-based interactions within Unity's physics engine. To read the controller's gyroscope direction, we created the **Gy-**

roGame action map, which reads the value of every axis of the controller orientation. Then, the script dynamically adjusts these values, translating gyroscopic movements into directional forces that influence the ball's motion.

To isolate the ball within the slot borders, we surrounded the playing field with box colliders that collide with the ball, not allowing it to fall out of the boundaries. Labyrinth walls also have colliders, whether visible or not, but invisible ones have a rumble zone surrounding them to trigger haptics (fig. 5.6) when the ball gets nearby.

Another task was creating local gravitation for the ball. We did this to eliminate global gravitation and ensure correct behavior in any position of the ball. To achieve this, we dynamically set the gravitational direction by calculating a plane normal based on the orientation of a specified game field. This plane normal, determined by the cross product of the field's right and forward vectors and then normalized, defines the direction perpendicular to the field's surface. By adjusting **Physics.gravity** to be the negative of this normalized plane normal scaled by a gravitational constant (9.81f), a unique local gravity effect is applied specifically to objects within this game scenario, allowing for dynamic and responsive ball movement.

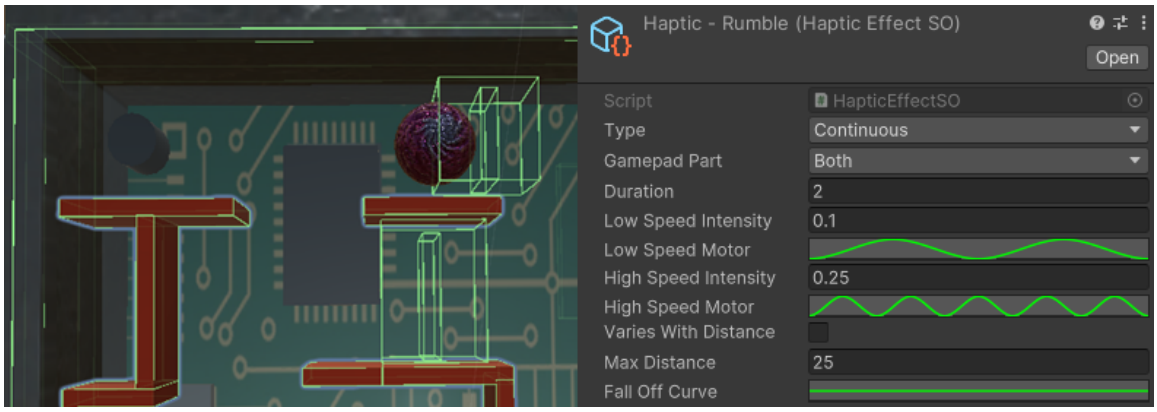

Figure 5.6: Rumble effect activates when the ball gets into rumble zone near invisible walls

Moreover, the script transforms the gyroscopic input from local space (relative to the field's plane) into world space to ensure it aligns correctly with the game's coordinate system, by adjusting the movement vector based on the field's rotation using **Quaternion.Euler**, the script ensures that the ball's motion corresponds accurately to the orientation of the game field.

The main aspects and tasks of the game are described, and the flow of the script is explained next. The entry point is slot selection (**HandleSlotControls**), where gyroscope input controls (defined in the Action map) specific to the Gyro Game scenario are enabled, allowing real-time capture of controller movements. Event listeners are established to detect when the ball enters a designated exit area to destroy the ball and trigger game completion. The main slot coroutine starts at this point.

The **UpdateGyroGame** coroutine continuously captures gyroscopic input while the slot is selected. This input is transformed into a local movement vector, providing responsive control over the ball's movement dynamics.

In the **FixedUpdate** method, synchronized with physics updates, the script calculates the movement direction based on the transformed gyroscopic input. This is where movement

direction is adjusted to align with the orientation of a local field's plane, applying local gravitation, and ensuring consistent gameplay dynamics within a defined spatial context.

The **HandleReturn** method then manages player actions related to exiting the gameplay scenario and returning to the default bomb state. It resets the ball's physics state (its velocity and direction) to freeze it in the place the player left it, disables gyroscope input controls, and cleans up event listeners associated with the module. This method ensures proper game state restoration and resource management upon player interaction.

5.4.3 Vent minigame

The **Vent Game** (fig. 5.7) is the third module created. It is a mini-game that does not use the unique features of the DualSense controller but is designed to vary the gameplay and test players' attention and stress control. This slot consists of a timer, a digital screen with a question, and two buttons: „*yes*“ and „*no*“.

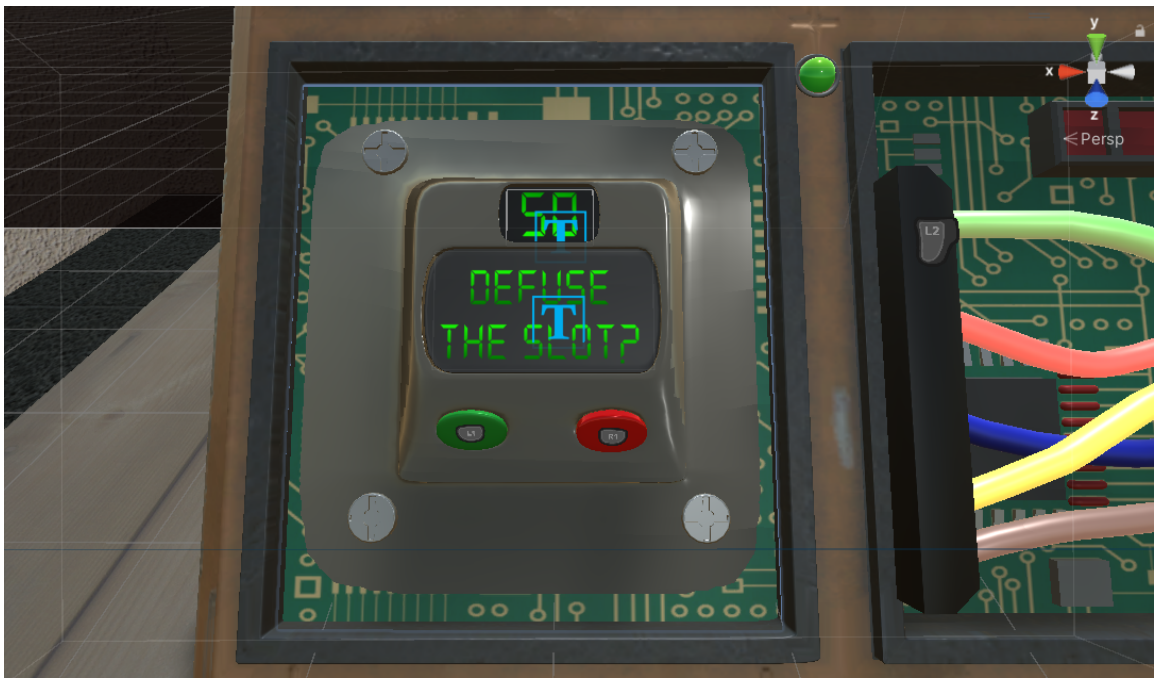

Figure 5.7: The Vent minigame in the game scene

This is the first module where the player's actions, or lack thereof, can lead to the bomb detonation. The game randomly asks two questions: „*Defuse the slot?*“ or „*Detonate the bomb?*“. The player has a minimal amount of time, much less than the overall bomb timer to answer these questions. If the time runs out, the bomb explodes. It also explodes if the answer is „*yes*“ to „*Detonate the bomb?*“ or „*no*“ to „*Defuse the slot?*“. However, if the player answers correctly, the slot ventilates the bomb preventing it from detonating, and the slot is successfully defused. The diagram scheme for this slot describes it better (fig. 5.8).

To achieve this, the Vent Game script configures essential components upon initialization (**Awake**), including the UI components representing digital screens for the question and timer and the Timer script itself to manage countdown mechanics. Additionally, the random question selection happens on initialization. When the slot is selected

(**HandleSlotControls**), input controls specific to the Vent Game scenario are enabled. To control inputs, a **VentGame** action map was created. Event listeners are established to handle player responses („*Yes*“ or „*No*“) via button inputs. The “*L1*” button stands for “*Yes*”, and “*R1*” stands for “*No*”.

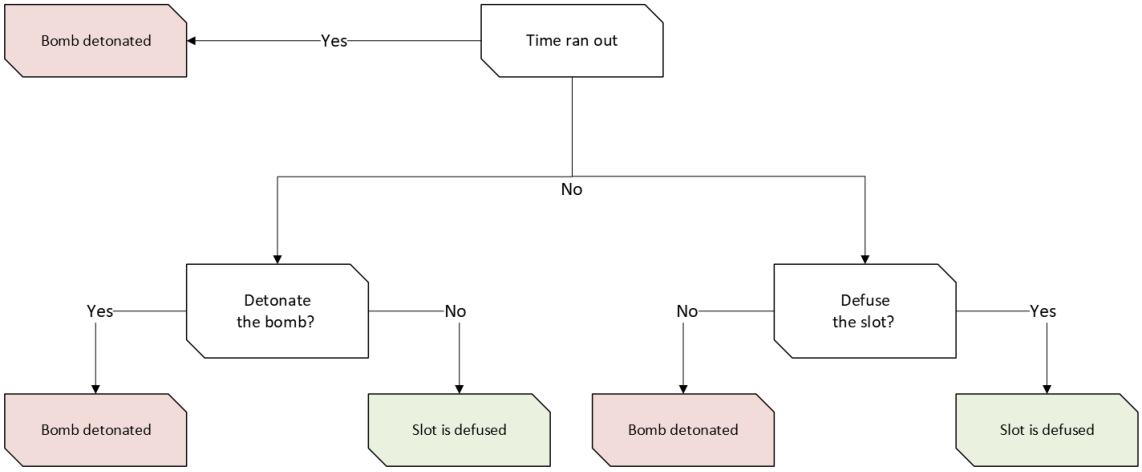

Figure 5.8: Diagram of Vent slot defuse conditions

The core gameplay logic revolves around the HandleButtonSelect method, triggered upon player input. Depending on the randomly selected text option, the method either defuses or detonates the bomb based on the player's choice. This method directly interacts with the Bomb Manager to initiate bomb defusal or detonation.

During gameplay, the script continuously monitors the slot countdown timer. If the timer reaches **zero** , the script triggers the bomb detonation with the help of **BombManager**, ending the interaction sequence.

Additionally, the script includes functionality to handle players exiting the gameplay scenario and returning to the default bomb state. This method disables input controls specific to the Vent Game and cleans up associated event listeners, ensuring proper resource management upon interaction conclusion.

5.4.4 Wires minigame

The fourth and last bomb module created is the **Wires Game**. It is a mini-game that utilizes haptic feedback and adaptive triggers of DualSense to help players navigate and defuse the slot. The goal is to choose one of five wires and cut it, but which wire to cut can hint only controller.

On the navigation through the wires controller gives a one-shot rumble effect and the correct one has a different pattern and intensity than the others. One more hint is the wire-cutting process itself which is done with controller triggers and has a specific resistance simulating the actual wire-cutting. The correct one has another required force to cut through than the others, which are slightly easier to cut (fig. 5.9).

This is one more module where the player's actions can lead to the bomb detonation. When players cut the wrong wire, they get a strike. If three wrong wires are cut the bomb detonates, but a player still can make one or two mistakes and then defuse the slot by choosing the correct wire.

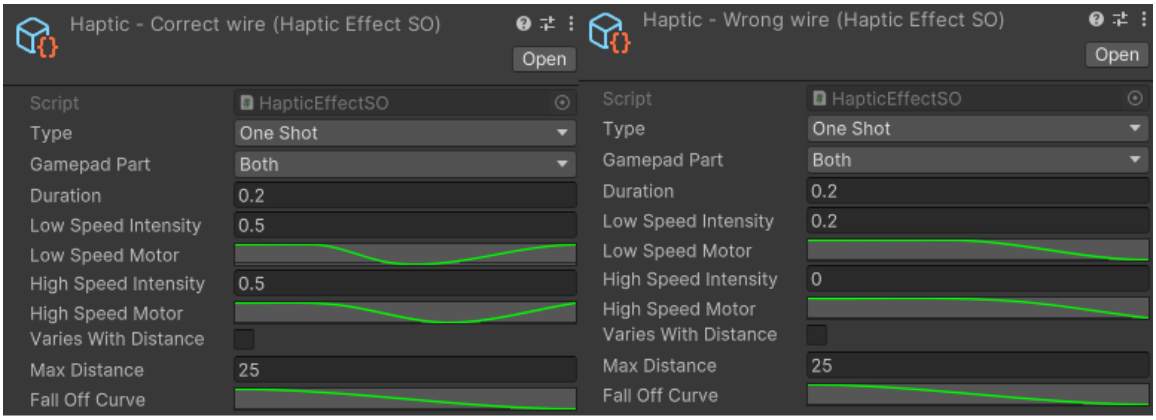

Figure 5.9: Example of different rumble effects for different wires

To achieve this, upon initialization (**Awake**), essential components and configurations are set up. The script identifies and configures wire game objects, and strike indicators, and integrates with the **BombManager** for bomb-related interactions. Additionally, wire models are configured with specific materials for highlighting and cutting effects.

The core gameplay revolves around wire navigation and trigger interaction. Players use directional inputs (**DPadUp**, **DPadDown**) to navigate between wires, with each wire having unique rumble and trigger resistance settings. Navigation also changes wire materials to be highlighted when selected, providing visual feedback to the player.

The script manages wire model states (**SwitchWireModel**) based on player interactions, dynamically switching between uncut and cut wire representations with appropriate model and material changes.

Gameplay interactions (**HandleTriggerPull**) monitor trigger inputs to determine wire-cutting outcomes and change the wire model to cut. Correct wire cut triggers slot completion events, while incorrect wire cuts trigger failure, incrementing strike indicators and potentially leading to the bomb detonation (fig. 5.10).

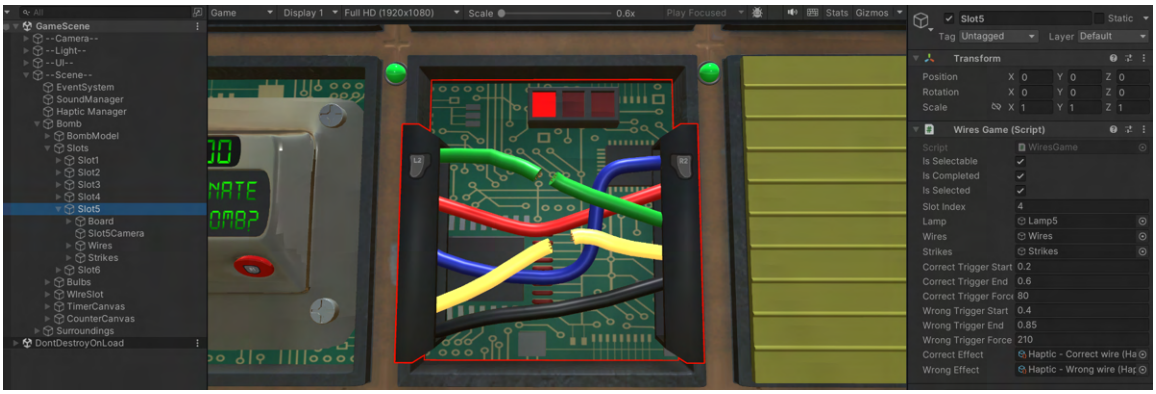

Figure 5.10: During the gameplay showcase, the strike counter increased when the wrong wire was cut and the correct wire was cut to complete the slot.

Whenever the slot is selected (**HandleSlotControls**), the script configures DualSense trigger states (**correctTrigger**, **wrongTrigger**, **noResistance**) based on wire-specific parameters like its type, resistance force, and zones. These states control trigger resistance

feedback during gameplay, enhancing the tactile experience for players. Enables the specific Wires Game slot controls. To control inputs, a **VentGame** action map was created. Event listeners are established to handle player inputs.

The update coroutine (**UpdateCoroutine**) continuously monitors player interactions and trigger inputs, ensuring real-time responsiveness and accurate wire-cutting mechanics throughout gameplay when the slot is selected.

The script also handles input disablement and resource cleanup (**HandleReturn**), guaranteeing proper state management and interaction conclusion, and cleans up associated event listeners on players exiting the gameplay scenario and returning to the default bomb state.

5.4.5 Final red wire

The **final wire** slot can only be unlocked once all modules have been defused (fig. 5.11), allowing access to the last red wire that needs to be cut to completely defuse the bomb. The goal is to give players the opportunity to make a final effort to defuse the bomb and experience its tactile feedback using the DualSense controller.

During initialization (**Awake**), the script configures essential components such as the camera controller for scene management and game objects related to wires: slot covering cup, and wire models. During the **Update** loop, the script manages the opening and closing animation of the covering cup using quaternion interpolation (**Quaternion.Lerp**), resulting in a smooth transition between specified opened and closed states.

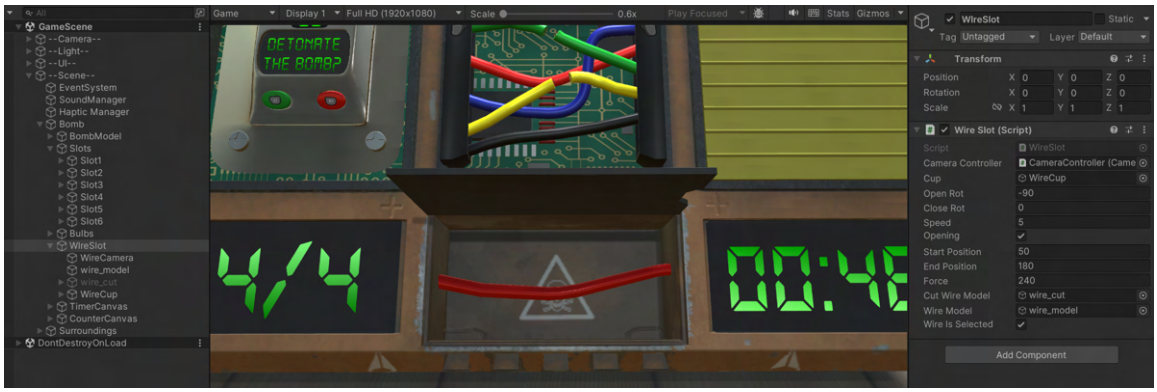

Figure 5.11: The final wire is unlocked when all slots are defused

Wire interaction is facilitated through the monitoring of DualSense controller inputs. The script reads left and right trigger values and checks for specific trigger thresholds. If both triggers exceed the threshold, the wire model is visually updated to simulate a successful cut, triggering a bomb defusal event and leading to successful game completion.

The **HandleWireControls** method manages wire slot interactions upon selection. It enables wire-specific controls while disabling broader bomb controls and activates the camera controller's wire view. Additionally, the method configures DualSense trigger states to provide tactile feedback during wire manipulation.

The **HandleWireReturn** method handles player actions related to exiting the wire interaction. It triggers the covering cup closing animation, re-enables bomb controls, and transitions the camera controller back to the default view, ensuring a seamless return to the main bomb defusal scenario.

5.5 Camera controller

In the game, there are eight virtual cameras available: one for each bomb slot, one more is the final wire slot, and the final is the default bomb state (fig 5.12). For better camera control in the game, the **Cinemachine**[1] package was used and the **CameraController** script was created. The main advantage of **Cinemachine** for this game is a priority-based camera system, allowing us to easily manage multiple cameras and determine which one should be active at any given time. In the script, we used the Priority property of **CinemachineVirtualCamera** instances to control camera switching. This simplifies camera management and ensures that the correct camera is active based on game events or states.

The **Cinemachine** enables smooth transitions between different camera shots and perspectives. By adjusting the priorities of virtual cameras, as demonstrated in the script, we can seamlessly switch between default bomb view, wire view, or slot-specific views without abrupt cuts or jarring movements. This enhances the overall visual experience for players.

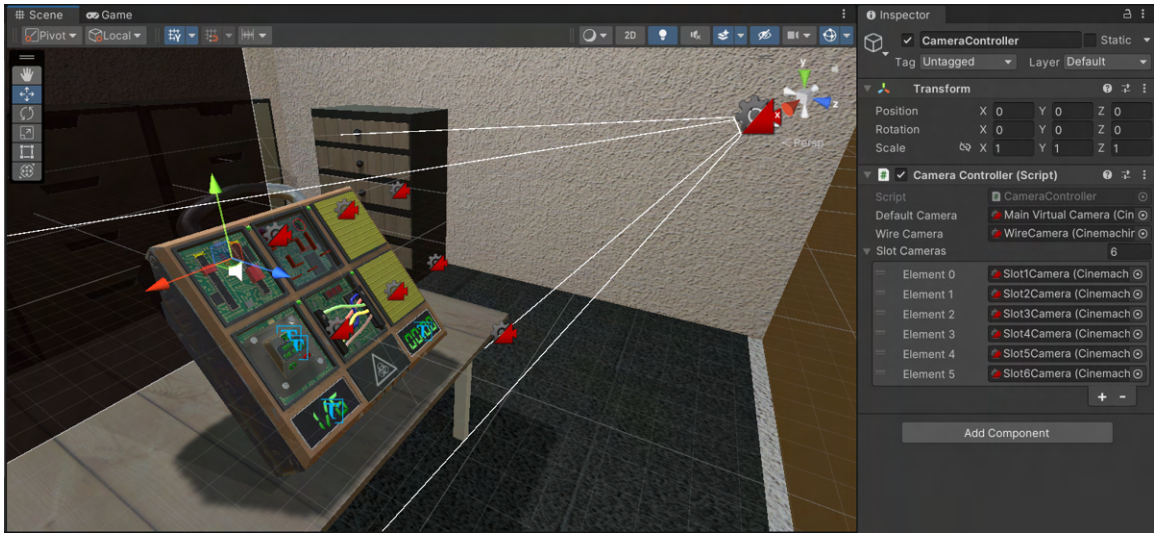

Figure 5.12: Showcase of all cameras in the game scene

[1]Cinemachine page https://unity.com/unity/features/editor/art-and-design/cinemachine

5.6 User interface

The primary concept behind the user interface was to create a realistic UI, that exists in the game space known as a **diegetic** 2.2.1 **UI** (fig. 5.13), which would have minimal overlays. To achieve this, all the information that could be displayed on the bomb was displayed on it. For instance, the timer and slot progress counter were integrated into the bomb, and each slot had its completion indicator. Additionally, some control hints were also displayed on the bomb.

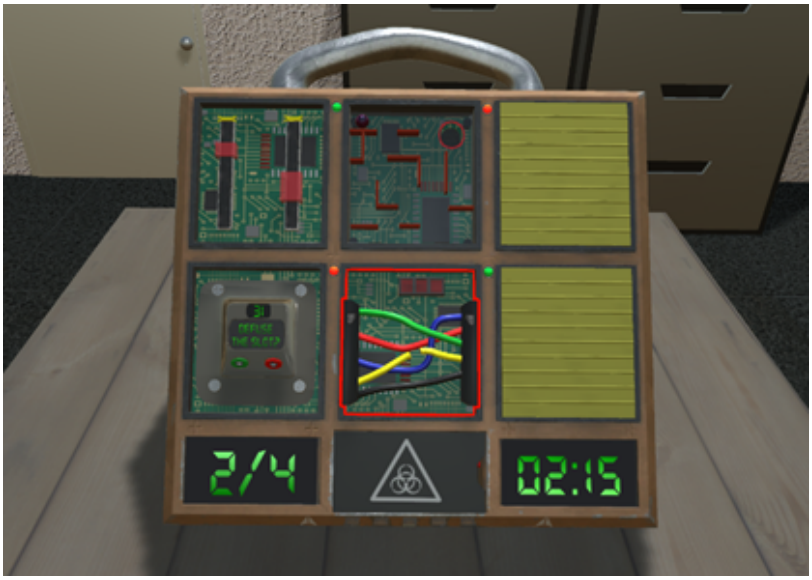

Figure 5.13: Diegetic UI in the game

The game pause and game-over overlay screens are non-diegetic UI aspects of the game. To create a UI overlay in Unity, we followed the tutorial of **CodeMonkey** [3] and will explain the process here. Firstly, we created a canvas component that acts as a container for UI elements and provides settings for screen space (overlay, camera, or world space). Then, we created two separate views in this canvas: one for the pause screen and another for the game-over screen (fig. 5.14).

Each screen has attached scripts to control and manage them. In the game-over scenario, when the **BombManager** calls a **ShowDetonateScreen** or **ShowDefuseScreen** method of the **GameOverUI** script, it configures the required elements and sets the screen active to be shown. The script also handles button navigation using Unity's **EventSystem** to ensure that the correct button is selected and enables highlighting when it is chosen. The game-over screen provides information on whether the bomb was defused and how much time was left and allows the player to restart the level or return to the main menu.

The pause screen logic is similar to the game-over screen but with some differences. It also handles button navigation using **EventSystem** and has listeners for the resume and main menu buttons. Additionally, it listens to the **BombManager** , which invokes **OnGamePaused** and **OnGameResumed** events to notify the **GamePauseUI** when the game is paused or resumed. Upon receiving these events, the **GamePauseUI** responds by showing or hiding the pause UI.

Figure 5.14: The game-over screen in the game scene

5.7 Audio

All the sound effects in the game are created by Daniel Bogdanov[2]. Daniel produced several audio clips exclusively for the different aspects of this game. Implementation of the sound manager is also based on the tutorial of **CodeMonkey** [3].

The **SoundManager** script was created to handle the sounds in the game. It is responsible for managing and playing various audio clips that correspond to specific game events triggered by different components and systems. To achieve this, the script utilizes Unity's **AudioSource.PlayClipAtPoint** method, which plays audio clips at specified positions in the game world. The aim was to have all the audio clips in one place for easy management.

To accomplish this, a **ScriptableObject** was created to hold references to all the audio clips used in the game. These clips are assigned in the Unity Editor and accessed through the **SoundManager** script to trigger the appropriate sounds based on game events.

To play sounds, event handlers were used. These events are triggered by different game components such as **BombInput**, **WireSlot**, **BombManager**, and others. The **PlaySound** method is then triggered, which plays a specific audio clip at a specified position and volume.

Overall, the **SoundManager** script centralizes audio management in the game, ensuring that audio cues enhance gameplay by providing feedback for critical game events such as bomb defusal, wire cutting, slot completion, and UI interaction. This approach contributes to a more immersive gaming experience by engaging players through auditory feedback aligned with their actions and progress in the game.

[2]Daniel Bogdanov LinkedIn https://www.linkedin.com/in/daniel-bogdanov-4a7113208/

Chapter 6

Testing

In the development of the **Cut the red!**, both **manual** [17] testing and **playtesting** [18] were utilized to ensure its quality and success. Manual testing involves rigorous testing of the game's features, mechanics, and performance focusing on identifying bugs, glitches, and usability issues. Additionally, playtesting sessions were conducted with players with different gaming experiences to gather feedback on the game's gameplay, level design, difficulty, and overall player experience. By combining these testing methodologies, we could iteratively refine and improve the game, and gather valuable feedback to deliver a polished and engaging gaming experience to players.

6.1 Manual testing

During the development process, I conducted **manual testing** after implementing each module or a significant part of the game. This involved examining and testing all aspects of the gameplay mechanics, user interface elements, input methods, and haptic feedback to ensure that the game was of high quality, stable, and playable.

Manual testing was particularly crucial for this project since it aimed to use various haptic effects that could not be tested automatically. Therefore, it was a significant part of the development process. Every single mechanic was tested and adjusted manually to enhance user experience and allow them to really feel the game.

6.2 User testing

Playtesting refers to the process of observing real players as they interact with a game to understand their preferences, behaviors, and reactions. The main goal of playtesting is to evaluate the **fun** factor, **engagement**, and overall satisfaction of the players with the game. It is an exciting phase for developers because it helps them understand the potential of the game and whether or not users find it interesting, fun, and emotionally engaging.

In this user test, we had a sample size of **6 people**. Although this is a small sample size, it included **3 people** with low or no gaming experience and **3 more experienced** gamers. This allowed us to analyze the level of accessibility, measure how user-friendly the game is for different types of players, and compare their gameplay experience. The objective of the test was to try out the puzzles and mechanics implemented, so we gave the participants the minimum amount of information before starting the test.

All participants were provided with the following:

- A **brief description** of the game, including its ideas, mechanics, and goals.
- The basic **control scheme** for the bomb.
- A clear understanding of the **motivation and tasks** involved in the testing process. Users were asked to share their thoughts, feelings, and actions during gameplay.
- A list of **bugs** and **not implemented** features.

After this brief introduction, the users played the game with no further instructions to gain a genuine impression of the game and identify any issues with playability or bugs. The entire process was recorded, and feedback was obtained through reports later.

6.2.1 Bug Reports

During the early stages of video game development, user testing plays a crucial role in the identification and fixing of bugs. When only one person, particularly a developer, is testing the game, they may become too focused on certain mechanics or areas where they anticipate the most bugs. This can result in other parts of the game being overlooked and not thoroughly tested, leading to potential issues being missed. Therefore, user testing by a diverse group of individuals can help identify bugs across the entire game and ensure that it is functioning smoothly.

The group of experienced gamers had an edge in finding and identifying bugs because of their in-depth knowledge of game mechanics, tricks, and solutions. This enabled them to complete the game quickly and concentrate on uncovering bugs.

Below are the most frequently reported bugs in the game:

- There was a bug with the slot completion in the **gyro** minigame. After the ball falls into the finish zone, nothing happens, the responsible event is not triggered.
- There are various issues with **inputs** after **pausing** the game. After pausing the game, the controls are redefined, allowing a user to get to the final wire without solving all the slots. Additionally, triggers are working in the pause state.
- There are some problems with controller **connectivity**.

6.2.2 User impressions

In addition to bug reports, gathering user feedback on their overall experience with the game was crucial. This included their impressions on the game's difficulty level, whether it was easy or hard enough if the objectives were clear, and how the haptic feedback felt. Less experienced players took more time to solve the puzzles and defuse the bomb, but they found every mechanic to be highly interesting and challenging. On the other hand, more experienced players emphasized the importance of haptic feedback implementation throughout the game.

Some of the most common recommendations are the next:

- Add the control scheme **hint** at the beginning of the game.
- Make the ball movement with the **gyroscope** more responsive.
- Provide information to the player whether the **controller** is connected or not.

Chapter 7

Conclusion

In conclusion, I have successfully achieved the goal of creating a game that utilizes haptic feedback technology by developing „Cut the red!“ - a 3D puzzle game in Unity. This bomb defusal game uses the advanced features of the DualSense controller, including adaptive triggers and haptic feedback, to provide players with an immersive gaming experience.

I researched the game development industry, its history, techniques, and technologies to accomplish this. I also learned how haptic feedback is implemented in games and which tools, accessories, and consoles use this technology.

During the development process, I faced several challenges due to the limited range of DualSense controller possibilities, available on Personal Computers. However, I overcame these challenges by utilizing available tools, and successfully implementing the core mechanics and haptic feedback. Nevertheless, there is always room for improvement.

Looking toward the future, I plan to continue improving and expanding the game by addressing any issues identified by testers, implementing their feedback, and introducing additional puzzles and mini-games. My ultimate goal is to become a PlayStation Partner, which would provide me with access to the necessary tools, resources, and opportunities to bring the game to the PlayStation store and reach a wider audience.

In summary, I am satisfied with the result, this game development project has given me valuable experience and insights into the intricacies of game development. It has also provided me with a solid foundation for future game development projects, and I am excited to continue learning and growing in this field.

Bibliography

[1] Bowers, M. Level Up: A Guide to Game UI (with Infographic). *Toptal Design Blog* online. Toptal, september 2019. Available at: https://www.toptal.com/designers/gui/game-ui. [cit. 2024-05-04].

[2] Cheung, J. Game UI Design: Everything You Need To Know. *CareerFoundry* online, may 2023. Available at: https://careerfoundry.com/en/blog/ui-design/game-ui-design/#the-game-ui-design-market. [cit. 2024-05-03].

[3] CodeMonkey. *Learn Unity Beginner/Intermediate 2023 (FREE COMPLETE Course - Unity Tutorial)* online. Youtube, january 2023. Available at: https://youtu.be/AmGSEH7QcDg?si=EOEVE8smHUOt7MiP. [cit. 2024-04-30].

[4] Flanders, C. The Impact of Haptic Feedback in UI/UX: Engaging Users through Touch Sensations. *Medium* online. Bootcamp, december 2023. Available at: https://bootcamp.uxdesign.cc/the-impact-of-haptic-feedback-in-ui-ux-engaging-users-through-touch-sensations-843e46d0a884. [cit. 2024-05-03].

[5] History, E. *Diegetic UI - Realistic, or Distracting? - Extra Credits* online. Youtube, august 2019. Available at: https://www.youtube.com/watch?v=-auyB19Mc6U. [cit. 2024-05-03].

[6] McManus, I. *Unity Tutorial: Adding Haptic Feedback* online. Youtube, november 2021. Available at: https://youtu.be/6YMcoVkAOBA?si=P7DXgM8alUpdhzt7. [cit. 2024-04-30].

[7] *Nintendo Switch - Official Site* online. April 2024. Available at: https://www.nintendo.com/us/switch. [cit. 2024-04-29].

[8] *Playdate - Official Site* online. April 2024. Available at: https://play.date. [cit. 2024-04-29].

[9] *PlayStation 5 - Official Site* online. November 2023. Available at: https://www.playstation.com/cs-cz/ps5. [cit. 2024-04-29].

[10] Prots, P. *7 stages of game development process* online. December 2023. Available at: https://stepico.com/blog/game-development-stages. [cit. 2024-04-29].

[11] Ramadan, R. and Widyani, Y. Game development life cycle guidelines. In: *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)* online. 2013. Available at: https://doi.org/10.1109/ICACSIS.2013.6761558. [cit. 2024-04-29].

[12] Rao, P. 50 Years of Video Game Industry Revenues, by Platform. *Visual Capitalist* online, december 2023. Available at: https://www.visualcapitalist.com/video-game-industry-revenues-by-platform. [cit. 2024-04-29].

[13] Reer, F.; Wehden, L.-O.; Janzik, R.; Tang, W. Y. and Quandt, T. Virtual reality technology and game enjoyment: The contributions of natural mapping and need satisfaction. *Computers in Human Behavior* online, 2022, vol. 132, p. 107242. ISSN 0747-5632. Available at: https://doi.org/https://doi.org/10.1016/j.chb.2022.107242. [cit. 2024-04-29].

[14] Robarge, D. *The Magnavox Odyssey predicted the future of video games* online. September 2022. Available at: https://americanhistory.si.edu/explore/stories/magnavox-odyssey-predicted-future-video-games. [cit. 2024-04-29].

[15] See, A. R.; Choco, J. A. G. and Chandramohan, K. Touch, Texture and Haptic Feedback: A Review on How We Feel the World around Us. *Applied Sciences* online, 2022, vol. 12, no. 9. ISSN 2076-3417. Available at: https://doi.org/10.3390/app12094686. [cit. 2024-04-29].

[16] Simons, J. Game Studies - Narrative, Games, and Theory. *Game Studies* online, 2007. Available at: https://gamestudies.org/0701/articles/simons. [cit. 2024-05-04].

[17] Solutions, P. Game Testing Methodology: A Comprehensive Guide. *Blog* online, december 2023. Available at: https://prometteursolutions.com/blog/a-guide-on-game-testing-methodology-best-practices-techniques-and-tools/#manual-game-testing-methodology. [cit. 2024-04-30].

[18] Solutions, P. What is PlayTesting? How to Get Good Feedback For Your Game? *Blog* online, may 2023. Available at: https://www.testbytes.net/blog/what-is-play-testing. [cit. 2024-04-30].

[19] Technologies, U. *Unity - Manual: Unity User Manual 2022.3 (LTS)* online. April 2024. Available at: https://docs.unity3d.com/2022.3/Documentation/Manual/index.html. [cit. 2024-04-29].

[20] Xu, T. What Is Haptic Feedback? *Built In* online, june 2023. Available at: https://builtin.com/hardware/haptic-technology. [cit. 2024-04-29].

Appendix A

Contents of the SD card

The SD card attached to the work contains a **README** file with instructions, along with the program source files of the Unity project in version 2022.3.1f1. The Unity project files can be found in the folder named **UnityProject**. In the **Build** folder, there is an executable build of the game. Additionally, the thesis text and LaTeX source files are located in the **Docs** folder. As part of the thesis, a poster and a short video were created to present the game, and these files are in the **Extra** folder.

- `README.txt`
- `UnityProject` ..... Source files folder
  - `Assets`
  - `Library`
  - `Logs`
  - `Packages`
  - `...`
- `Build` ..... Build of the game folder
  - `Cut the red!_Data`
  - `Cut the red!.exe` ..... Executable game file
  - `UnityPlayer.dll`
  - `...`
- `Docs`
  - `xproko40.pdf` ..... Text of the thesis
  - `TextSource` ..... LaTeX source files
- `Extra`
  - `video.mp4` ..... Game promo video
  - `poster.pdf` ..... Poster for thesis