UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 648

DEVELOPMENT OF A MULTIPLAYER HIDE-AND-SEEK VIRTUAL REALITY GAME

Emilia Haramina

Zagreb, June 2022

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 648

DEVELOPMENT OF A MULTIPLAYER HIDE-AND-SEEK VIRTUAL REALITY GAME

Emilia Haramina

Zagreb, June 2022

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 11 March 2022

BACHELOR THESIS ASSIGNMENT No. 648

Student:	**Emilia Haramina (0036526081)**
Study:	Electrical Engineering and Information Technology and Computing
Module:	Computing
Mentor:	prof. Lea Skorin-Kapov

Title: **Development of a Multiplayer Hide-and-Seek Virtual Reality Game**

Description:

Modern commercial Virtual Reality (VR) systems enable the tracking of user movements in six degrees of freedom, providing an incentive for application and game developers to develop creative interaction methods with virtual environments. Digital games developed for VR have become a significant part of the gaming market in recent years, primarily due to the development of VR systems that are financially acceptable to the wider market. Although they share many features with games on other platforms, VR games require specific design and development approaches. In the scope of this thesis, the task is to implement a multiplayer networked game using the Unity game engine, which would be based on searching for hidden objects in a virtual environment. Furthermore, the game should be implemented in two versions: a competitive version where players play against each other, and a collaborative version where players collaborate to achieve a common goal. Finally, a user study should be conducted to investigate the impact of the chosen variant on the user experience. All necessary literature and working conditions will be provided by the Department of Telecommunications.

Submission date: 10 June 2022

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 648

Pristupnica:	**Emilia Haramina (0036526081)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Lea Skorin-Kapov

Zadatak: **Razvoj višekorisničke igre skrivača uporabom tehnologije virtualne stvarnosti**

Opis zadatka:

Suvremeni komercijalni sustavi za virtualnu stvarnost (engl. Virtual Reality, VR) zasnovani su na tehnologiji koja omogućava praćenje korisnikovih pokreta u šest stupnjeva slobode te služe kao poticaj razvijateljima usluga i igara za razvoj kreativnih metoda interakcije s virtualnim svijetom. Digitalne igre razvijene za virtualnu stvarnost su u posljednjih nekoliko godina postale značajan dio tržišta igara, prvenstveno zbog razvoja VR sustava koji su financijski prihvatljivi širem tržištu. Iako dijele mnoge značajke s igrama na ostalim platformama, igre u virtualnoj stvarnosti zahtijevaju poseban pristup prilikom dizajna i razvoja. U ovom radu potrebno je u razvojnoj okolini Unity implementirati višekorisničku umreženu igru koja bi se temeljila na traženju skrivenih objekata u virtualnom prostoru. Nadalje, potrebno je igru implementirati u dvije verzije: kompetitivna verzija gdje igrači igraju jedan protiv drugog, te kolaborativna gdje igrači surađuju kako bi ostvarili zajednički cilj. Na kraju potrebno je provesti korisničku studiju s ciljem ispitivanja utjecaja odabrane verzije igre na korisničko iskustvo. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 10. lipnja 2022.

I would like to thank my parents, Lara and Valerio and my boyfriend, Edi, for their immense support during my whole college education and my life as a whole. Thank you for always being here for me. I would also like to thank my mentor, Lea Skorin-Kapov, and the teaching assistants in the Department of Telecommunications, especially Ivan Bartolec, for always helping me with any technical issues and guiding me through the creation of this thesis.

CONTENTS

1. Introduction

Virtual reality (VR) is a rapidly developing technology. With the use of VR headsets, controllers, and other accessories, it allows the user to immerse themselves in a totally virtual environment. Unlike other kinds of extended reality (XR) technology, VR completely obscures the user's vision of the actual world. Because VR is still a relatively new technology, assessing Quality of Experience (QoE) is critical, as it may lead to enormous improvement of this technology in a very short amount of time. Despite the fact that gaming was not the cause for the invention of VR, it is now one of the most prevalent applications of this technology, and it has given VR products a lot of popularity [14].

The goals of this thesis are as follows:

- explore the history and present situation of the VR industry,
- describe VR technology and explain its key characteristics,
- choose and describe examples of VR games,
- discuss the development of a multiplayer VR game,
- comment on the challenges and potential expansions of the developed game and
- perform and thoroughly analyze a user study to investigate the impact of the chosen multiplayer version on player QoE.

All of these goals are addressed in the six major chapters of the thesis. The first chapter offers a short introduction to the thesis. Following that, the second chapter provides a brief history of VR, outlining its characteristics and comparing it to other XR technologies, as well as covering current hardware and software VR support and some of the applications of VR. The third chapter explains VR gaming and provides two examples of VR games, one that popularized VR technology and one that utilizes multiplayer mode. Additionally, it briefly discusses certain elements necessary for measuring the QoE in VR games. In the fourth chapter, a multiplayer VR game, CATch

Cafe, is developed, along with stating its hindrances and potential expansions in the future. In the fifth chapter, a user study that was conducted using CATch Cafe is analyzed. Finally, in the sixth chapter, a conclusion summarizing and assessing all of the attained outcomes is presented. A list of references, a list of figures, a list of abbreviations and the user study are also included in additional chapters.

2. Virtual Reality

According to the Merriam-Webster dictionary, virtual reality is an artificial environment which is experienced through sensory stimuli provided by a computer and in which one's actions partially determine what happens in the environment [34]. By examining the definition more closely, a few crucial elements, which are all necessary for understanding VR, become apparent. The first one is *artificial environment* — the virtual world presented to the user is made up entirely of virtual items, none of which come from the actual world. Secondly, *experienced through sensory stimuli provided by a computer* — VR entails interaction across many sensory channels, primarily vision and sound, but also less typically touch, smell, and taste. Finally, *one's actions partially determine what happens in the environment* — the computer can recognize the user's input and immediately alter the virtual environment, indicating that the user has complete freedom to roam around the virtual world, interacting with the environment and the objects inside of it, see it from various perspectives and reshape the reality they are viewing [59].

As depicted in Fig. 2.1, Burdea and Coiffet defined VR as having three fundamental components [12]. *Immersion* refers to the creation of a realistic-looking environment utilizing a computer graphics simulation in which the user may interact and participate in the activity happening on the screen. *Interaction* indicates that as the user performs various actions, the synthesized environment changes [13], which are enabled through input from various devices, gestures, and spoken orders. *Imagination* references that the use of human imagination is employed in the design and creation of the virtual environment as well as in the development of problem-solving applications. All of these elements and components, as well as the history and the extent of VR's existence and application today, will be examined in greater depth in the following subchapters.

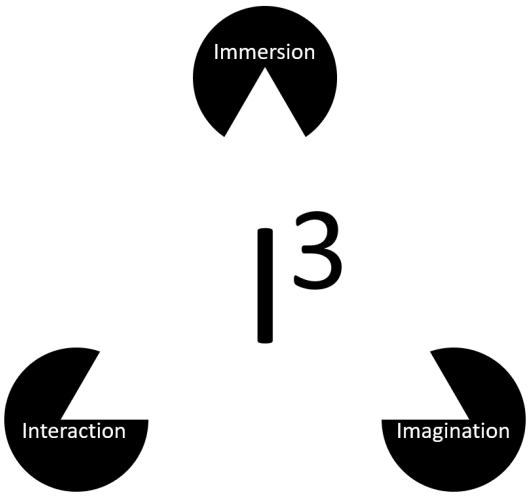

Figure 2.1: The three I's of virtual reality, immersion-interaction-imagination. Adapted from [12]

2.1. Brief History of Virtual Reality

In 1838, Sir Charles Wheatstone was the first to explain stereopsis, and in 1840 was awarded the Royal Medal of the Royal Society for his explanation of binocular vision, which led to the development of the stereoscope [15]. The idea of the stereoscope is that the viewer is looking at two photographs of the same item or location taken from different angles, and their brain integrates the two, giving the image they are viewing an impression of depth and immersion. A sketch of the stereoscope is seen in Fig. 2.2. Today, the popular *Google Cardboard* and low-cost VR head-mounted display (HMD) for mobile phones incorporate the design concepts of the stereoscope.

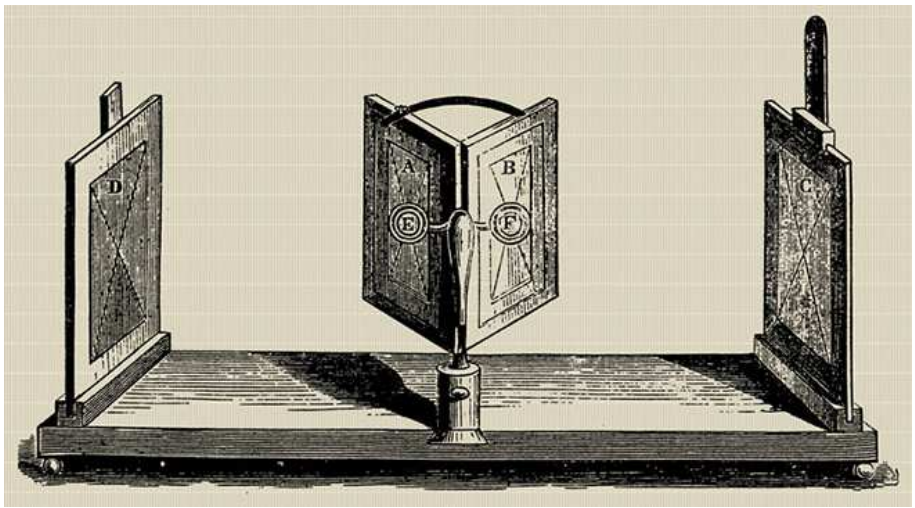

Figure 2.2: The Wheatstone mirror stereoscope [15]

Pygmalion's Spectacles, shown in Fig. 2.3, was initially published in 1935 in the *Wonder Stories* magazine. The notion of VR is introduced for the first time in this story [29]. The tale centers around a pair of goggles which allow the user to immerse himself/herself in a fictitious world using holographics, smell, taste and touch.


PYGMALION'S SPECTACLES

By STANLEY G. WEINBAUM

Author of "The Black Flame," "A Martian Odyssey," etc.

Figure 2.3: Pygmalion's Spectacles short story [29]

Sensorama, the first VR system (patented in 1962), was invented by cinematographer Morton Heilig in 1956 and merged numerous technologies to stimulate all of the senses: full color three-dimensional (3D) video, music, vibrations, smell, and atmospheric effects like wind [15]. The goal was to completely immerse the viewer in the movie. Fig. 2.4 depicts how Sensorama was used.

Figure 2.4: The Sensorama machine [28]

Heilig also patented the *Telesphere Mask*, depicted in Fig. 2.5, the first HMD which provided stereoscopic 3D images with wide vision and stereo sound [15]. Without motion tracking, the HDM remained in a non-interactive cinematic format.

Figure 2.5: The Telesphere Mask [23]

None of the early VR innovations, up to this point in history, allowed for interaction with the virtual environment, which is an essential component in the definition of VR. In 1965, computer scientist Ivan Sutherland unveiled his idea of the *Ultimate Display*, shown in Fig. 2.6. The idea was to create a virtual environment that could be viewed through an HMD and that was so realistic that the user would not be able to tell the difference between it and real life. This provided the ability for the user to interact with objects. This idea included computer gear to create the virtual environment and maintain it running in real time [15].

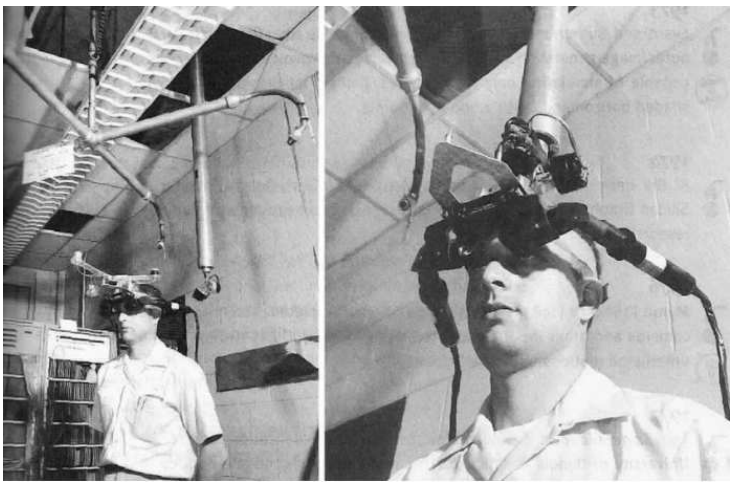

Figure 2.6: The Ultimate Display [4]

2.2. VR Principles and Characteristics

2.2.1. Field of View

The Field of View (FOV) refers to how far the display can accommodate eye and head movement [19]. The monocular FOV of an average human is a 200 to 220-degree arc in which they can observe the environment around them without moving their head, whereas a VR headset can show only roughly 180 degrees. Additionally, the human's left and right eye vision overlap in a roughly 114-degree arc, known as the binocular FOV, which gives their surroundings a 3D appearance [7]. Both the monocular and binocular FOV are depicted in Fig. 2.7. Currently, no VR headset can accommodate the average human's entire natural FOV, and if the presented FOV is too narrow, the user may suffer from nausea as a result of a misunderstanding of information in the brain [19]. Today's VR hardware manufacturers are striving for devices with a 180-degree FOV, which is regarded optimal for a high-performance VR simulation [7].

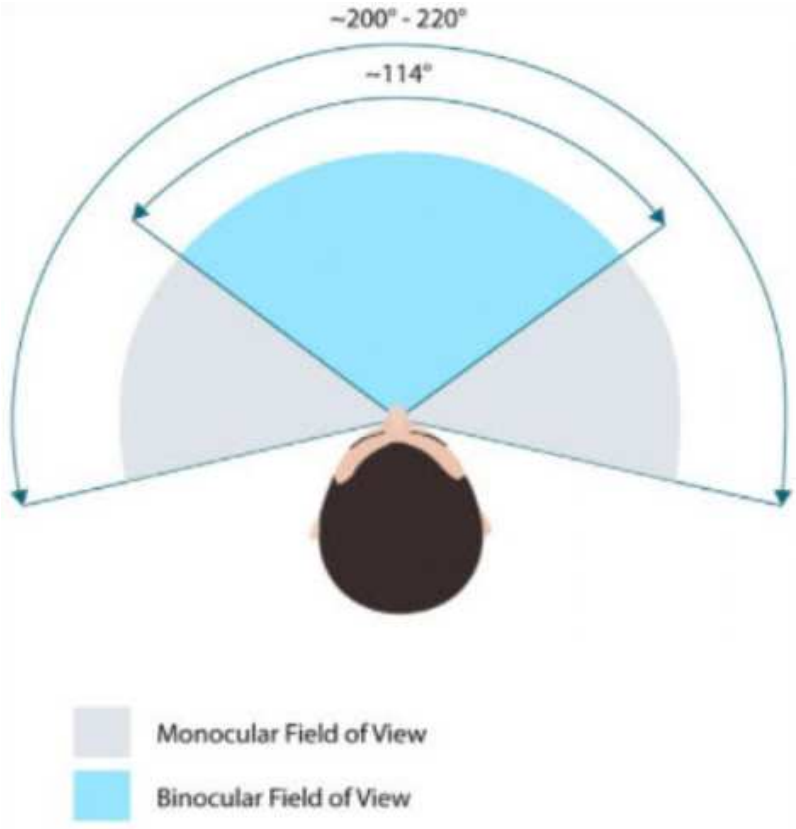

Figure 2.7: Monocular and Binocular FOV of a human [9]

2.2.2. Frame Rate

Frame rate is the number of pictures processed per second by the Graphics Processing Unit (GPU) [19]. In order to properly simulate what people perceive while not wearing a VR headset, frames seen through a VR headset must move at an astounding speed. Experts claim that the human eye can perceive up to 1000 frames per second (FPS), but that level of detail is never received by the human brain via the optic nerve. According to certain studies, humans can distinguish frame rate up to 150 FPS, but above that, the information is lost in translation on the way to the brain [7]. Most VR developers have discovered that anything less than 60 FPS causes the user to experience disorientation, migraines and nausea [30]. As a result, they aim for VR content with a frame rate of around 90 FPS, while some companies, such as *Sony*, refuse to certify software that dips below 60 FPS at any time during use [45]. In order to provide a more realistic experience for most VR applications, VR developers are striving towards 120 FPS or higher.

2.2.3. Spatial Audio and Sound Effects

Binaural or spatial audio is a simulation of the auditory sense that provides a more realistic multi-dimensional experience. Spatial audio or 360-degree sound is required for VR to fully immerse the user into a different space, since it creates a simulated audio landscape that complements the 3D sights created by VR. The user can tell the direction the sound is originating from with spatial audio. Furthermore, auditory cues are essential for leading users through their VR experience in the manner intended by the developers [26]. Some headsets come with built-in audio headsets, while others allow for the use of headphones as an add-on. The use of positional, multi-speaker audio - commonly referred to as positional audio - creates a 3D illusion to the ear in VR [19]. To account for the user's movement, spatial audio for a VR headset must be calculated in real-time. There are a number of existing implementations for spatial audio which all have certain common features, such as:

- volume control,
- direction is conveyed by using a left/right delay,
- head tracking is used to map auditory space and
- simulating environmental elements by manipulating reverberation and echo [7].

2.2.4. Head and Position Tracking

In VR, the virtual world adapts to the user's position. Degrees of freedom (DoF) are used to measure head and position tracking features, allowing the user to explore either 3DoF or 6DoF. With 3DoF, the user can look around the virtual space since the headset is capable of rotational tracking only, which means it can detect when the user turns their head left and right, looks up or down and tilts their head to one side or the other. On the other hand, the headset is incapable of detecting whole-body movement. *Samsung Gear VR*, *Google's Daydream* and the *Oculus Go* are examples of 3DoF headsets [7].

6DoF headsets can track a user's position in a room and display the direction in which their head is pointing. This enables total autonomy in a virtual space, resulting in a considerably more immersive VR experience. Furthermore, sensors on the exterior of the VR headset assist the user in remaining safe while moving around a room [42]. Gyroscopes, accelerometers and magnetometers are the sensors utilized for 6DoF. To calculate a very precise location of the user's head in a virtual environment, the *Sony Playstation VR* headset employs external LEDs that are monitored with an external camera [53].

Eye-tracking technology is becoming increasingly remarkable as it may aid in improving attention in VR experiences and lessen discomfort experienced by some users while using a VR headset. The integration of haptic feedback sensors and other tracking technologies to include controller options in VR may also make the landscape more immersive [42]. The head tracking movement must be less than 50 milliseconds in order to prevent causing nausea to the user [53].

2.3. Comparison to Other Extended Reality Technologies

XR is a catch-all term for any technology that modifies reality by adding digital aspects to the real-world environment [32], enhancing or replacing the user's view of the world to some extent. It includes technologies such as VR, augmented reality (AR) and mixed reality (MR). Despite the fact that they share some traits and criteria, each of the three technologies has its own set of goals and underlying technologies. The capacity to traverse the world and display context-sensitive information through visual input methods such as object, gesture and gaze tracking is a basic feature of all XR devices, and the distinctions between VR and other XR technologies will be examined more

thoroughly in the following subchapters [44].

2.3.1. Augmented Reality

AR is a fully immersive and interactive first-person experience that is spatially registered to the actual world and provides the user with a strong sense of presence within a combination of the real world and a virtual environment [31]. It is the addition of digital elements to a real-world environment in many ways, such as photos, videos or interactive data. This technology is now mostly used in smartphone AR applications that require the user to hold their phone in front of them [44]. The application may show contextual information or display gaming and social experiences that appear to be based in the real world by capturing the image from the camera and analyzing it in real-time.

Snapchat filters and *Pokémon Go*, a game in which digital characters are superimposed on the user's actual FOV, as seen on Fig. 2.8, are two examples of AR experiences [22]. AR applications may also allow users to project date, time, weather and other information into their FOV, see a product in the location they wish or modify their appearance. Many industries, including gaming, education, healthcare and manufacturing are benefiting from the possibility to overlay digital items over the real-world environment [32].

Figure 2.8: *Pokémon Go*, screenshot from [38]

VR systems replace everything in the user's FOV with a simulated environment, whereas AR systems augment the real world surrounding the user with artificial components. Because VR is all about immersing the user in a totally virtual environment viewed through a screen in a headset, the actual world isn not part of the experience. AR, on the other hand, is about enhancing or adding to the current reality. The user may be looking at something real, but it could also have digital characters and content overlaid on it. The main difference between AR and VR is that AR is still focused on the physical environment around the user, but it incorporates data overlays that can give additional insights [52]. Although augmented content does not identify or interact with physical objects in the real world, it does improve the user's experience. Furthermore, AR is regarded as less restrictive than VR since it does not separate the user from the actual world. Because it does not need to generate a fully new world, AR often uses less computing power than VR. The pricing of AR glass devices remains high when compared to VR since it is primarily aimed at enterprise-level clients and because of the vast amount of technology packed into a relatively small wearable device [54].

As depicted in Fig. 2.9, all technologies that make up XR are part of a continuum, according to Milgram and Kishino [35]. VR is on the far right end of the continuum because it immerses the user into a completely new, fully digital, virtual environment, whereas AR is closer to the left end since it incorporates virtual elements into an entirely physical real-world environment while allowing the user to see everything that happens in the actual world around them.

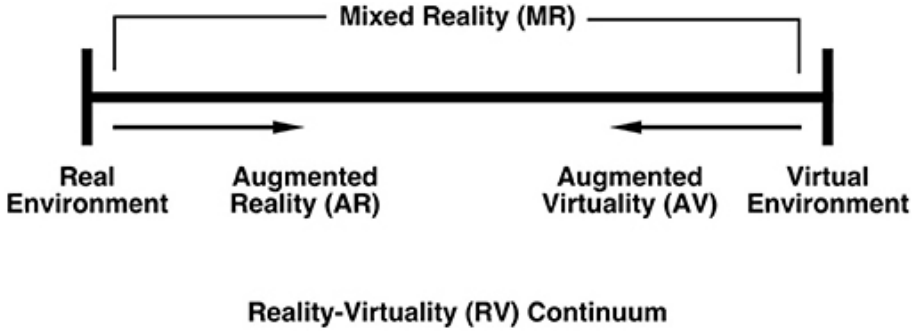

Figure 2.9: Milgram and Kishino's reality-virtuality continuum. Adapted from [35]

2.3.2. Mixed Reality

In MR, also known as hybrid reality, virtual content is not only superimposed on the world, as in AR, but it is also attached to and interacts with the environment. This enables virtual content to integrate, enrich and interact with the user's real-world sur-

roundings. Another version of MR involves people interacting with a completely virtual environment that is superimposed on the real world around them. This type of MR is closer to VR than AR and some VR headsets have sensors that track the physical environment as well [10]. Because MR allows users to observe and interact with both digital and real components, MR experiences get input from the environment and adapt accordingly [32]. MR uses occlusion to blur the line between the real and virtual worlds: from the user's perspective, computer-generated objects can be concealed by objects in the physical environment. Most MR devices are still in the research and development stage, however MR is generally viewed through transparent wearable glasses [54].

Immersive communication experiences enabled by MR technology aid in more efficient collaboration. Users can put on headsets to collaborate while remaining connected to the real world. MR is useful in teaching, much like AR and VR. MR delivers immersive learning experiences by letting students see both the actual world and holograms. Manufacturing operations are aided by MR, which provides workers with real-time guidance. Holographic instructions show right in front of their eyes, ensuring that they always know what to do at any given time or situation. This lowers the risk of human error and enhances quality significantly. Maintenance and repairs can also benefit from MR, particularly in areas that demand expertise and accuracy [10]. One of the first companies to use MR technology was *Microsoft* with their *HoloLens*. Some of the more notable examples include a *Skype* app that allows users to draw and diagram while speaking to one another remotely, allowing them to interact with each other's creation, and *HoloStudio*, which allows you to create virtual objects using intuitive hand gestures, as depicted in Fig. 2.10, and then print them out using a 3D printer [52].

Figure 2.10: *HoloStudio*, screenshot from [20]

Virtual objects may be seen in MR, just as they can in AR, but they can also interact with the real world surrounding the user, which makes MR more immersive and interactive than AR. MR creates a true hybrid by fusing reality and the virtual world together. The software updates and responds automatically when the physical world around the user changes as a result of his/her actions, offering fresh and updated information [52]. Users can interact with both physical and digital aspects in MR experiences. MR differs from AR, in which digital and physical elements do not interact, and VR, in which the real world is fully blocked out of the user's FOV [32]. MR is often thought to be the furthest from realization of the three XR technologies mentioned, as it exerts the highest demand on hardware and software capabilities.

As seen in Fig. 2.9, the center of the reality-virtuality continuum is covered by MR. MR encompasses AR because of its combination of the real world with superimposed digital objects. It also includes augmented virtuality, a form of MR in which the user is mainly immersed into a virtual world while still see and interact with real-world objects. Because they do not merge the real and virtual world, the continuum's extremities, the real and virtual environments, are not included in MR. The former has no digital elements, whilst the latter fully excludes the real world.

2.4. Hardware Virtual Reality Support

To manipulate the VR user's senses, VR hardware is utilized to provide stimuli. These can be worn on the user's body or utilized independently. Sensors in VR hardware monitor motions like button pushes and controller movements such as hands, head and eyes. The sensor is outfitted with receptors that capture mechanical energy from the user's body. The sensors in the hardware convert the energy received from a hand movement or button push into an electrical signal that is sent into a computer or device for processing [19].

VR devices include consoles, cellphones, and a Personal Computer (PC) that processes user inputs and outputs. For the optimal quality and experience, the computer must be able to generate high-quality visuals and commonly utilizes GPUs. Output devices include visual, auditory, and haptic displays that stimulate a sense organ and present users with VR content or environments in order to provoke a feeling. Examples of input devices which aid users in navigating 3D environments are:

- VR controllers,
- balls or tracking balls,

– controller wands,
– data gloves, such as the Manus Prime II depicted in Fig. 2.11,
– trackpads,
– on-device control buttons,
– motion trackers,
– bodysuits,
– treadmills,
– 3D mouses,
– smelling devices,
– and motion platforms, such as the Virtuix Omni shown in Fig. 2.12 [19].

Figure 2.11: *Manus Prime II* VR data gloves, image from [11]

VR headsets are a type of HMD. They enable users to immerse themselves in fictitious surroundings by using a First-Person View (FPV) [8]. It may also have integrated or linked controllers for navigating VR content. VR headsets are classified into three types: PC-based VR headsets, standalone VR headsets and mobile headsets. Because they provide the most immersive experiences, PC-based VR headsets are typically the most expensive. These headsets are often cable-connected and powered by external hardware. High-quality sound and image, as well as head tracking are provided via the dedicated display, built-in motion sensors, and an external camera tracker for increased realism. All-in-one or standalone VR headsets are wireless, integrated pieces of hardware that are similar to tablets or phones. However, not all wireless VR headsets are

Figure 2.12: *Virtuix Omni* VR motion platform, image from [39]

standalone. Some systems employ wireless transmission from nearby consoles or PCs, while others require wired packs carried in a pocket or attached to clothing. Mobile headsets are shell devices that cover a smartphone using lenses. The lenses divide the screen to provide a stereoscopic view, converting a smartphone into a VR device. Because phones do not provide the finest visual experiences and are outperformed by gaming console or PC-based VR, mobile headsets are comparatively affordable. Because the phone does the processing, no wires are required. Because no positional tracking is supplied, the created environment shows only from a single point, and it is not possible to look around objects in a scene [26].

Gyroscopes, structured light systems, magnetometers, and accelerometers are some of the sensors that can detect and monitor eye or head motion. To minimize rendering burden, sensors are used to detect a user's gaze position and subsequently reduce rendering resolution away from the user's gaze. Image clarity is influenced not just by camera quality, but also by display resolution, optic quality, refresh rate, and FOV. The camera is also used to track motion for room-scale VR experiences, in which the user wanders about a room while experiencing VR. However, sensors are more effective for this since cameras often have a higher latency. The ability to travel freely in space as you explore VR settings is a big worry with PC–tethered VR headsets. Inside-out

and outside-in tracking are terms used in virtual reality to describe how the VR system tracks the location of the user and associated devices as they move around a space. Inside-out tracking systems, such as *Microsoft HoloLens*, employ a camera mounted on the headset to track the user's location relative to the surroundings. Outside-in systems, such as *HTC Vive*, employ sensors or cameras positioned in the room environment to determine the headset's location relative to the surroundings [19].

The *Oculus Rift* headset's initial prototype was designed and created in 2010. It featured a never-before-seen 90-degree field of view and relied on the processing capacity of a computer to provide images. *Sony* revealed *Project Morpheus*, a VR headset for the *PlayStation 4*, in 2014. The same year, *Google* released *Cardboard*, a low-cost and do-it-yourself stereoscopic viewer for smartphones depicted in Fig. 2.13, while *Samsung* announced the *Samsung Gear VR*, a headset that uses a *Samsung Galaxy* smartphone as a viewer [15].

Figure 2.13: *Google Cardboard*, from [5]

Most VR headsets offered dynamic binaural audio in 2016, and haptic interfaces were still in the works, therefore handsets were generally button-operated. *HTC* produced the *HTC VIVE SteamVR* headset, seen in Fig. 2.14, the first commercial release of a headset with sensor-based tracking that allowed users to freely walk about a place. The first generation *Oculus Rift* device was released and *Sony* introduced *PlayStation VR*, depicted in Fig. 2.15. In 2018, *Oculus* demonstrated a new headset prototype, the *Half Dome*, a varifocal headset with a FOV of 140 degrees, and released their untethered *Oculus Go* headset [15].

In 2019, *Oculus* released *Oculus Quest*, depicted in Fig. 2.16, which was a standalone headset that created a lot of interest and momentum. The following year, *Oculus*

Figure 2.14: The *HTC Vive* headset, image from [15]

Figure 2.15: The *PlayStation VR* headset, image from [41]

Quest 2 was released, however, it was renamed *Meta Quest 2* later on, and it garnered mainly positive reviews as an incremental improvement to the *Quest* and continues to sell in the millions worldwide [15]. The *Meta Quest 2* headset can be depicted in Fig. 2.17.

Figure 2.16: The *Oculus Quest* headset, image from [15]

Figure 2.17: The *Meta Quest 2* headset, image from [43]

2.5. Software Support for Virtual Reality

The *Unity*, *Unreal Engine 4 and 5* and *CRYENGINE* game engines are famously known for game development, but they can also be used to create VR experiences for a variety of other sectors [3]. For example, they may be used to develop VR solutions for automotive, transportation, manufacturing, media and entertainment, engineering and construction. *Amazon Sumerian* is a VR engine that does not require the user to understand 3D graphics or VR programming skills in order to create a VR experience. The *Google VR* developer platform offers software development kits (SDKs) for all VR platforms it supports, as well as a wide range of VR development tools. To have 3D objects in a virtual world, they must first be created. Software like *Blender*, *3ds Max*, *SketchUp Studio*, *Maya* and *Oculus Medium* allow the user to create, model and paint any 3D elements they require [6]. There are also many other SDKs that help developers by easing the creation of VR experiences, such as *SteamVR SDK*, *VRTK*, *PSVR Dev Kit*, *Oculus SDK* and *Oculus Mobile SDK*.

2.6. Virtual Reality Applications

VR is used in a variety of industries due to its flexibility in content. The most widespread commercial use of VR is immersive gaming. Users can explore video game worlds and play video games in their full FOV with a VR headset and a gaming system. With the addition of haptic gloves and other VR gear, they may even translate their physical movements into their gameplay experience for a higher degree of interactivity [8].

In medicine and healthcare, VR is an excellent tool for training doctors on new technologies and providing experience in rare surgical procedures. These VR experiences allow physicians to practice difficult procedures before entering an actual operating room. A patient's pain can be managed by using different VR visuals to help distract the patient's brain by confusing pain pathways. VR can also be used for the treatment of post-experience trauma by re-living traumatic experiences and in turn helping medical experts understand patients' conditions and devise ways to help with their problems. VR can also aid patients with autism, a disorder that hampers reasoning, interaction, and social abilities, by increasing brain activity and imaging. VR can be applied in monitoring symptoms of different social disorders. Paraplegics can utilize VR to see different environment outside their confinements without requiring them to travel. It has also been used to assist paraplegics in regaining control of their limbs [19].

Following COVID-19, many educational institutions were looking for ways to engage and immerse students in an online environment. Professors can use VR to make these experiences nearly as immersive as a traditional classroom. VR also makes it possible for students to access institutions all around the world. Employees may collaborate on their tasks while maintaining a sense of presence while using VR. Businesses can use VR to test their ideas before creating them. It is also useful in testing new designs and models [8].

Many other professions can also benefit from VR simulations for training and education. From pilot simulators to automotive diagnostic tools, many businesses train their employees this way which lets them stay out of danger while saving costs [8].

3. Virtual Reality Video Games

VR gadgets are becoming increasingly popular among gamers. Furthermore, user and player awareness of VR is gradually rising. The arrival of VR headsets on the market triggered a tremendous transformation and revolution in the gaming industry [27].

VR gaming is the application of a 3D artificial environment in computer games. It is a term used to characterize a new generation of video games that employ VR technology to provide players with a genuinely realistic, first-person perspective of game action, allowing them to alter the virtual world. Unlike traditional video games, in which the perspective is slightly above and/or behind the player's characters, VR allows player to experience the video game world through the eyes of their character. VR gaming devices and accessories, such as VR headsets, sensor-equipped gloves, hand controllers and more, allow the player to both experience and manipulate the game environment [33].

A regular keyboard and mouse, game controllers or motion capture technologies may be used as controls in VR gaming. More complex VR rooms may contain treadmill floors or other similar technologies to increase the player's sensation of freedom of movement and feelings of immersion in a virtual environment. In some VR gaming configurations, a player may be restricted to a small place around a computer, but have full range of motion within that area [55].

3.1. Examples of Virtual Reality Video Games

While there are many examples of VR video games, the following subchapters will focus on two of them. The first video game significantly increased the popularity of VR gaming. The second one includes mechanics comparable to the VR video game developed in chapter 4.

3.1.1. Beat Saber

Beat Saber, seen in Fig. 3.1 is a VR rhythm game developed and published by Czech game developer *Beat Games* [18]. The player wields a pair of glowing sabers, typically a blue one for the right hand and a red one for the left hand, but this color setup can change, using VR controllers. When the player picks a song, blocks approach the player in rhythm with the chosen song. Each block is marked either with a dot or an arrow. If the block is marked with a dot, it may be hit with the saber of the corresponding color from any direction. On the other hand, if the block is marked with an arrow, it must be cut in the indicated direction.

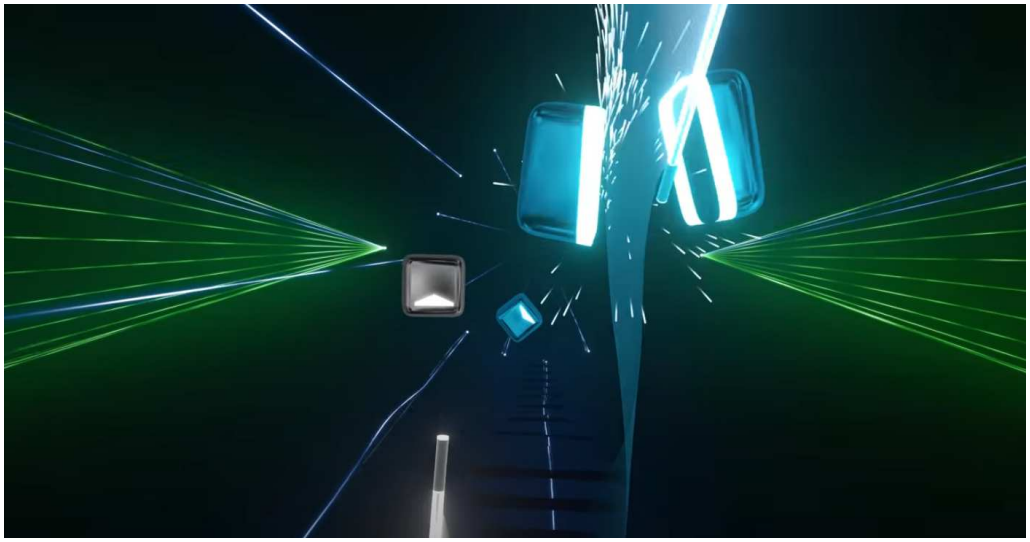

Figure 3.1: *Beat Saber*, screenshot from [40]

Since the addition of *Original Soundtrack 5*, two new mechanics were added to the game. One of those mechanics is a new block called *Chain*, which starts with a slice of a block indicating in which direction to cut it. Following this starting slice, there are a number of smaller slices which can be cut in the same direction as the leading one. The second mechanic is an arc showing a line the saber with the corresponding color has to follow. It is connected to a block and goes on until it either connects to another block or stops completely.

There are two types of obstacles which the player should avoid. The first are walls, which the player's heads should avoid. The second are bombs, which the player should not hit with the sabers. Below the path where blocks travel is a white bar that slowly fills up as the player hits notes correctly. Hitting walls, bombs or notes in incorrect positions and not hitting blocks at all decreases the white bar. If the bar becomes completely empty, the level ends.

Beat Saber contains both singleplayer and multiplayer mode, the latter of which is called party mode. Party mode includes a scoreboard with the score of all of the player's names, which are entered after each song is played. There is also a multiplayer mode which allows two to five players to play a level together, with the one with the highest score being the winner. Badges are also given based on performance. In addition, *Beat Saber* contains a level editor and a practice mode that allows the player to alter the speed the song is played at and to start playing it from any point in time.

3.1.2. Rec Room

Rec Room, depicted in Fig. 3.2, is a VR online video game developed and published by *Rec Room Inc.* [21]. It is playable on many platforms, including VR headsets. A hub room, called the *Rec Center*, vaguely resembles a lobby of a recreational center with doors that lead to various different games and rooms created by other players. All players are visible to every other player in the same room. Players can change their appearance, specifically their head, torso and hands.

Figure 3.2: *Rec Room*, screenshot from [37]

The game may be played with or without a VR headset. In VR mode, the game utilizes full 3D motion capture of the player's movement using a VR headset's motion capture technology and two hand-held motion controllers, which are necessary to pick up and manipulate, consisting of balls, weapons, construction tools and other objects. Players may explore the environment around them within the limitations of their actual floor-space by utilizing the controller buttons to teleport a short distance with minimum or no VR sickness. There also exists an option for players to move continuously rather

than teleport, however, this type of movement poses a higher risk of motion sickness.

Rec Room consists of standalone built-in multiplayer games known as *Rec Room Originals*. The game contains first-person shooters, a *Battle Royale* game, cooperative action-adventure games, a *Pictionary*-like game and numerous sports games. Players are also create many user-generated experiences, thanks to the provided in-game tools for user-generated content.

3.2. Quality of Experience of Video Games in Virtual Reality

A player's QoE is essential when playing a VR game since better QoE generally means better, more immersive gameplay. When a VR game developer discovers a component of their VR game is not meeting the demands of the majority of players, they may modify it, resulting in a more captivating, immersive and user-friendly game. Screen display latency, the level of immersion and the intuitiveness of the controls for single-player VR games are all crucial elements that players are questioned about in a QoE study. In multiplayer games, players are also questioned about the latency of their actions, the latency of other players' actions, and jitter. However, depending on the type of game, there are several more elements to consider.

In the scope of the ITU-T, recent standardization work has specified methods for assessing QoE for gaming [25]. The user's experience with gaming may change their QoE rating. With respect to test duration, short interactive tests should not be used for games that are difficult to learn and require a long time to master. Furthermore, multiple factors have been shown to impact player QoE ratings [24]. The device on which the user plays has an impact on how they assess the QoE. Better comfort typically results in a higher rating if the device must be worn. The user's QoE rating is also affected by the frame rate and display size. The way delay impacts QoE in multiplayer games varies on the type of game, whereas jitter may make the game look less smooth. The player's environment can also influence their rating. If more than one user is required for testing (e.g., multiplayer mode), the users' relationship may influences how they assess the overall QoE.

4. CATch Cafe Application Development

CATch Cafe is a VR hide-and-seek game that can be played in either singleplayer or multiplayer mode. Players find themselves located in a cafe, modeled after the popular concept of a "cat cafe". They are cafes in which cats are present. Sometimes, these cats might have been rescued, be it from the street or from the wild, but now live permanently on the premises. Other times, these cats may be from a lock shelter and learning to socialize so they could have greater chance of adoption. Sometimes, these cafes charge higher prices, entrance fees or charge by the hour in order to pay for the cats' veterinarian bills and their food [1]. In Zagreb, Croatia, there are only two "cat cafes", while in Taipei, Taiwan, where the concept of a "cat cafe" began, there are over 20 [2].

When playing CATch Cafe, the main menu appears when the player first enters the game. They have the option to play multiplayer, singleplayer, change their settings, or exit the game. If the player wishes, they can change the current locomotion type (with the following options available: teleportation and continuous movement) and whether or not sound will be heard in-game through the settings menu.

In singleplayer mode, a waitress greets the player and informs them that they must discover all cats hidden throughout the cat cafe to make up for the waiting time, since all tables are already full. If the player proceeds, nine cats will appear at random locations around the cat cafe, changing their positions each round. The player must find each one of them and bring them back to the waitress, placing them on the table in front of her. The waitress keeps note of how many cats are found. The game is won when the player finds all nine cats and brings them to the table.

Two players can play together in multiplayer mode, being able to choose between *collaborative* and *competitive* versions. If the collaborative version is selected, players must work together to find all the nine cats hidden throughout the cat cafe. The number of cats is counted collectively for both players. They win the game by bringing all nine

cats to the table.

In the competitive version, however, the two players have to individually find cats and the number of cats found is counted separately. When one of the players brings at least five cats to the table, they have won the game, as this automatically renders the other player unable to find more cats than their opponent. After winning a game, players can return to the main menu.

The motivation behind the development of CATch Cafe is to conduct a user study to determine the influence of the chosen multiplayer version on a player's experience. The two multiplayer versions are collaborative and competitive. CATch Cafe was also developed with the player's enjoyment in mind. A player can play the game alone or with others for leisure and entertainment.

4.1. Used Technologies and Tools

Unity is a cross-platform game engine developed by *Unity Technologies*. The engine is compatible with a wide range of platforms, including desktop, mobile, console and virtual reality. It can be used to create 3D or two-dimensional games, as well as interactive simulations and other experiences. Because of the versatility of content that can be created, it has also been adopted by industries outside video gaming, such as film, automotive, architecture, engineering, construction and the military [58]. The version of *Unity* used for the development of CATch Cafe is 2021.2.17f.

C# is a general-purpose, multi-paradigm programming language. It encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was designed by *Microsoft* and introduced alongside the *.NET Framework* and *Visual Code* [56]. The language is most commonly used with the *.NET Framework*, but it can also be used for the development of other applications, such as game development. In *Unity*, *C#* is used for any content-related scripting. The version of *C#* used for the development of CATch Cafe is 9.0.

The *XR Interaction Toolkit* package is a high-level, component-based interaction system developed by *Unity Technologies*. It provides a framework that makes 3D and user interface interactions available from *Unity* input events. The core of this system is a set of base *Interactor* and *Interactable* components, and an *Interaction Manager* that ties these two types of components together [16]. Interacting with the user interface and objects in the scene using VR controllers or other accessories, changing the camera's position and rotation based on how the player moves their head, and teleporting

or walking around the scene using different input devices are just some of the things that this toolkit makes easier to do in Unity. The version of the *XR Interaction Toolkit* used for the development of CATch Cafe is 2.0.1.

Photon Pun Networking 2 is a networking solution developed by *Exit Games* that simplifies the development of multiplayer games in *Unity*. To minimize latency and provide the shortest round-trip times, *Photon Pun 2* games are hosted on a globally distributed *Photon Cloud*. The architecture used is client to server [17]. The version of *Photon Pun 2* used for the development of CATch Cafe is 2.40.

Microsoft Visual Studio 2022 is an integrated development environment from *Microsoft* [57]. It contains an integrated debugger and supports numerous programming languages, with many more available via plug-ins. *Unity* is configured to work in this environment. The version of *Microsoft Visual Studio 2022* used for the development of CATch Cafe is 17.1.

The *Unity Asset Store* is a marketplace of assets that can be used in a project in *Unity* [50]. *Café Hotel* was used in designing the cat cafe that CATch Cafe takes place in [46]. The *Distant Lands Free Characters* asset pack provided the game with human characters [47]. *Low Poly Cats* were used as the cat models in CATch Cafe [48]. *PUN 2 - FREE* is used to ease the development of the multiplayer version [49]. The *VR Headset Vol - 2* asset pack was used to show the head position of the other player in the multiplayer versions [51].

4.2. Implementation and Features of CATch Cafe

4.2.1. Integration of the *XR Interaction Toolkit* into CATch Cafe

The following *YouTube* video shows in detail how to set up a *Unity* project to work with VR devices: https://www.youtube.com/watch?v=yxMzAw2Sg5w. After importing the *XR Interaction Toolkit* into the *Unity* editor, an *XR Origin* was created by right-clicking the hierarchy, choosing *XR* and then *XR Origin (Action-based)*. This creates an *XR Origin* object, depicted in Fig. 4.1, containing a *Camera Offset* in which there is a *Main Camera*, acting as the head of the player, and the *LeftHand* and *RightHand Controllers*, which follow the position and and input of the controllers the player is using.

To allow the player to move through the scene, a *Locomotion System*, also depicted in Fig. 4.1, was added by right-clicking the hierarchy, choosing *XR* and then choosing *Locomotion System (Action-based)*. The player now has red lasers that point out of

where their hands are. These lasers turn white when the player can pick up an object the laser is pointing at or teleport to a *Teleportation Area*, as seen in Fig. 4.2, where the floor is a *Teleportation Area*.

Figure 4.1: *XR Origin* object

Figure 4.2: Lasers showing what the player can interact with, the white beam is close enough to the floor to teleport to that spot, but the red one is too far away.

4.2.2. Scene Design

There are two scenes that make up CATch Cafe. The Main Menu is the scene that the player first encounters when playing the game. The Cat Cafe is loaded once the player picks either singleplayer or multiplayer mode.

Main Menu

The Main Menu, depicted in Fig. 4.3, contains the CATch Cafe title and the options to play multiplayer, singleplayer, open the setting or exit the game. Upon clicking *Play Multiplayer*, the player connects to the server and is able to choose a lobby, which is thoroughly explained in subsection 4.3.6. If the player clicks *Play Singleplayer*, the Cat Cafe scene is loaded and the game begins, as depicted in Fig. 4.4. If the player clicks the *Exit* button, they will exit the game.

Figure 4.3: The Main Menu of CATch Cafe

```
1 public void Play() {
2     SceneManager.LoadScene("Cat Cafe");
3 }
```

Figure 4.4: The *Play()* function loads the Cat Cafe

Upon clicking the *Settings* button, the settings menu opens, depicted in Fig. 4.5. In this menu, the player can change whether sound will be heard in-game by clicking the volume icon. They can also change their movement type, with the two that are available being teleportation and continuous movement. Lastly, the player can close the settings and return to the Main Menu.

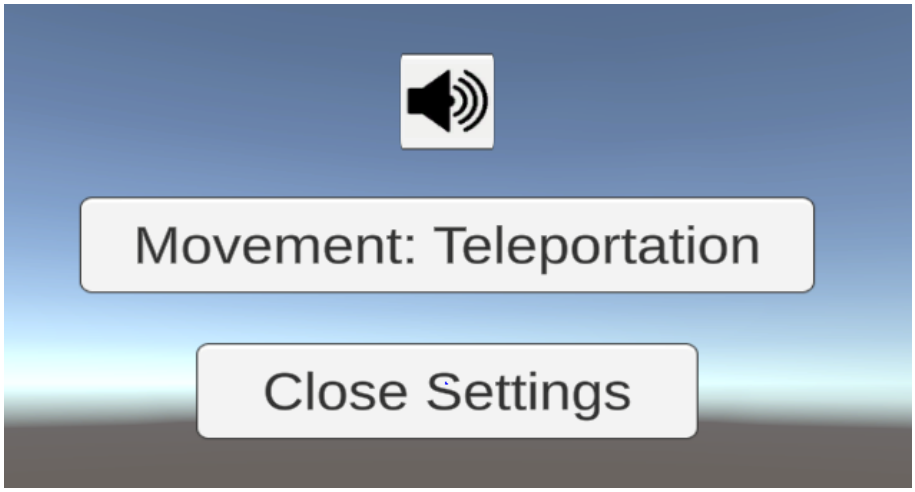

Figure 4.5: The settings menu

When the player clicks on the volume icon or the movement type, the *ChangeSound()*, seen in Fig. 4.6, and *ChangeMovement()* functions are called, respectively, which switch the current sound or movement setting.

```
1 public void ChangeSound() {
2     if (Utility.ChangeSound().Equals("soundOn")) {
3         soundOnImage.SetActive(true);
4         soundOffImage.SetActive(false);
5     } else {
6         soundOnImage.SetActive(false);
7         soundOffImage.SetActive(true);
8     }
9 }
```

Figure 4.6: The *ChangeSound()* function

In line 2, the *ChangeSound()* function from the static class *Utility* is called, which checks the current sound setting, switches it and sets the value of the current sound setting saved in *Unity's PlayerPrefs* to the new sound setting, as seen in Fig. 4.7.

```
1 public static string ChangeSound() {
2     string newSound = null;
3     if (sound.Equals("soundOn")) {
4         newSound = "soundOff";
5     } else {
6         newSound = "soundOn";
```

```
7       }
8
9       sound = newSound;
10      PlayerPrefs.SetString("sound", newSound);
11      return newSound;
12  }
```

Figure 4.7: The *ChangeSound()* function

Because this setting is saved in *Unity's PlayerPrefs*, the value can stay the same even when the player restarts the game, which is done in the *Start()* function of the script *InitializeMenu*, depicted in Fig. 4.8.

```
1   void Start() {
2       if (Utility.InitializeSound().Equals("soundOff")) {
3           soundOnImage.SetActive(false);
4           soundOffImage.SetActive(true);
5       }
6
7       if (Utility.InitializeMovement().Equals("continuous")) {
8           movementText.GetComponent<TMPro.TextMeshProUGUI>().
                text = "Movement: Continuous";
9       }
10  }
```

Figure 4.8: The *Start()* function in the *InitializeMenu* script

To initialize the sound value, the above mentioned function calls the *InitializeSound()* function, depicted in Fig. 4.9, from the static class *Utility*. The same principle explained here for setting the sound is used for checking and changing the movement type of the player.

```
1   public static string InitializeSound() {
2       if (PlayerPrefs.HasKey("sound")) {
3           sound = PlayerPrefs.GetString("sound");
4       } else {
5           PlayerPrefs.SetString("sound", "soundOn");
6       }
```

```
7       return sound;
8   }
```

Figure 4.9: The *InitializeSound()* function in the static class *Utility*

Cat Cafe

Upon entering either singleplayer or multiplayer mode, the required scripts based on the player's chosen movement type in the settings are enabled, as seen in Fig. 4.10. At the beginning of the game, the player is placed in front of one of the waitresses at the Cat Cafe, along with her dialogue box explaining that the players needs to find all cats hidden throughout the Cat Cafe, seen in Fig. 4.11 and starts her talking animation. The player is placed into the main sitting area, depicted in both Fig. 4.12 and Fig. 4.13. In front of the player is the bar, seen in Fig. 4.14, while to the right is the lounge area, shown in Fig. 4.15.

```
1   void Start() {
2       if (Utility.GetMovement().Equals("continuous")) {
3           GetComponent<TeleportationProvider>().enabled = false;
4           GetComponent<ActionBasedSnapTurnProvider>().enabled =
                false;
5           GetComponent<ActionBasedContinuousMoveProvider>().
                enabled = true;
6           GetComponent<ActionBasedContinuousTurnProvider>().
                enabled = true;
7       }
8   }
```

Figure 4.10: The *Start()* function in the script *InitializeGame*

The furniture, walls, ceiling and floor were taken from the *Café Hotel* asset pack [46]. The workers and customers are from the *Distant Lands Free Characters* asset pack [47]. The animations used for their movement were downloaded from the *Mixamo* webpage, changing the *Rig's Animation Type* to *Humanoid* and dragging the *mixamo.com Animation Clip* into the *Animator* component of the workers or customers [36].

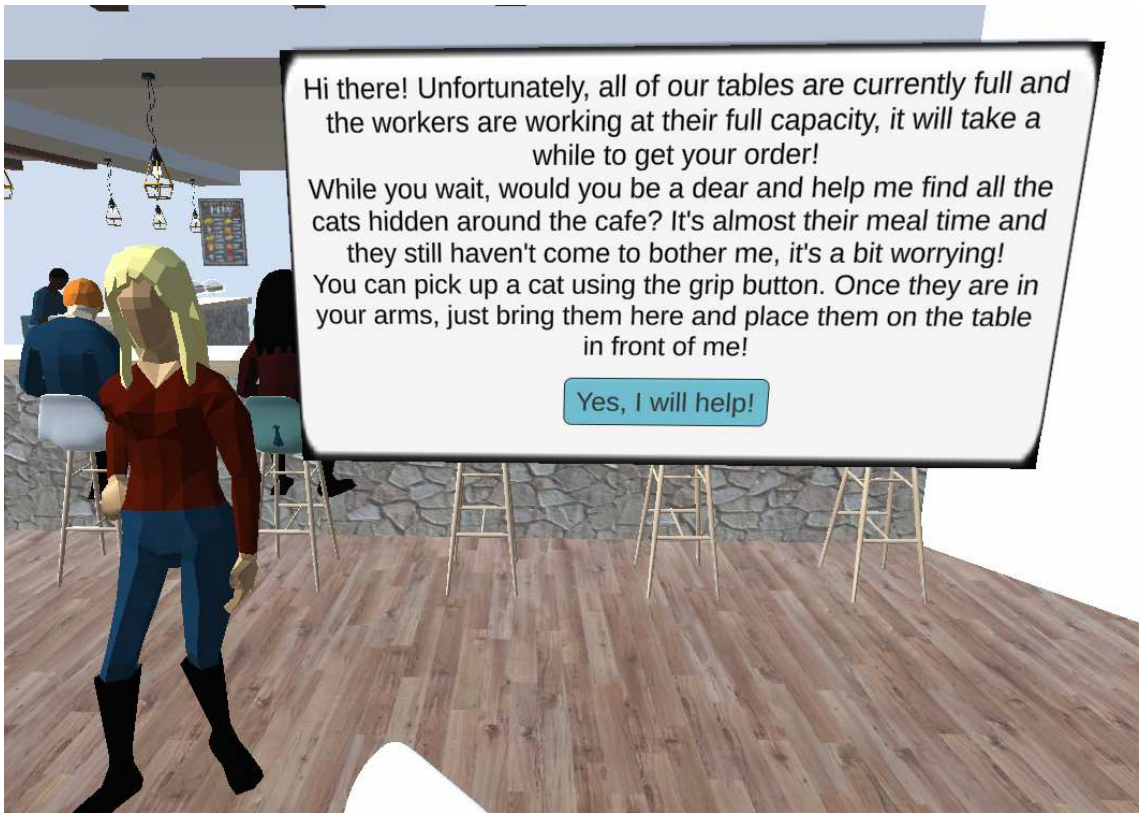

Figure 4.11: The main waitress and her dialogue box

Figure 4.12: The first part of the main sitting area of the Cat Cafe

Figure 4.13: The second part of the main sitting area of the Cat Cafe

Figure 4.14: The bar of the Cat Cafe

Figure 4.15: The lounge area of the Cat Cafe

The floor contains the *Teleportation Area* script on it. This allows the player to teleport at any position on the floor. Because of the blueprint of the Cat Cafe, seen in Fig. 4.16, parts of the floor, which are outlined in red, go beyond the playable area. To restrain the player from teleporting into those parts of the Cat Cafe, the size of the *Box Collider* component of the floor has been reduced and changed to fit only the area the player is supposed to be able to move around in.

Figure 4.16: The blueprint of the Cat Cafe, the parts outlined in red are beyond the playable area, but still contain the floor

4.2.3. Cat Animations, Box Colliders and Sounds

All cats are given an *XR Grab Interactable* script, allowing the player to grab them. In the *Attach Transform* field of the script, an empty *GameObject*, which is a child of the cat, is referenced in order to move the position of the cat around the hand once it is grabbed. Animations of the cats are also part of the *Low Poly Cats* asset pack [48].

Each cat additionally has an *Animator* script, in which their starting animation is referenced. When cats are grabbed by the player, their grabbed animation is played. After cats are let go, their idle animation is played until they are picked up again. This was done by adding events to interactions in a cat's *XR Grab Interactable* script. In the *Select Entered* interactable event, the *OnGrabAnimation()* function is called, while the *OnLetGoAnimation()*, shown in Fig. 4.17, function is called when the *Select Exited* interactable event occurs.

```
1 public void OnLetGoAnimation() {
2     catAnimator.runtimeAnimatorController =
          letGoAnimatorController;
3     catAnimator.Play("Entry");
4 }
```

Figure 4.17: The *OnLetGoAnimation()* function

There are a total of eleven materials that come with the asset pack. Even though every individual part of the body, such as the left paw or the mouth, can have their color changed, only the original eleven were used in the making of CATch Cafe. Because every animation changes the position of the cat, specific *Box Colliders* have been made for each animation in order to portray the cat's shape as closely as possible. In Fig. 4.18, *Box Colliders* for the sitting animation are shown.

To help the player find cats more easily, the nine cats around the Cat Cafe produce meowing sounds. This is done by giving each cat an *Audio Source* component. By setting its *Spatial Blend* value to 1, the meowing sounds 3D, or, in other words, the player can tell from which direction the meowing is coming from. The *3D Sound Settings* for the meowing sounds are shown in Fig. 4.19.

In the *Update()* function of the script *Meowing* time is counted by adding the value of *Time.deltaTime* to the time that has already passed. If the sum of the time passed is bigger or the same as the time between each meowing sound, the sum is subtracted by that time, a random cat is chosen from the nine cats hidden around the Cat Cafe and

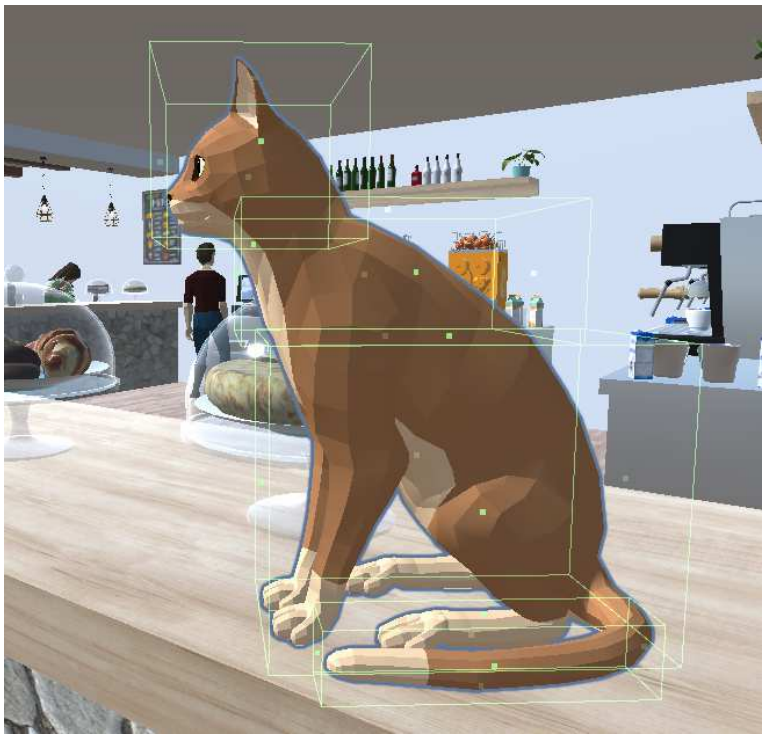

Figure 4.18: *Box Colliders* for a cat with the sitting animation

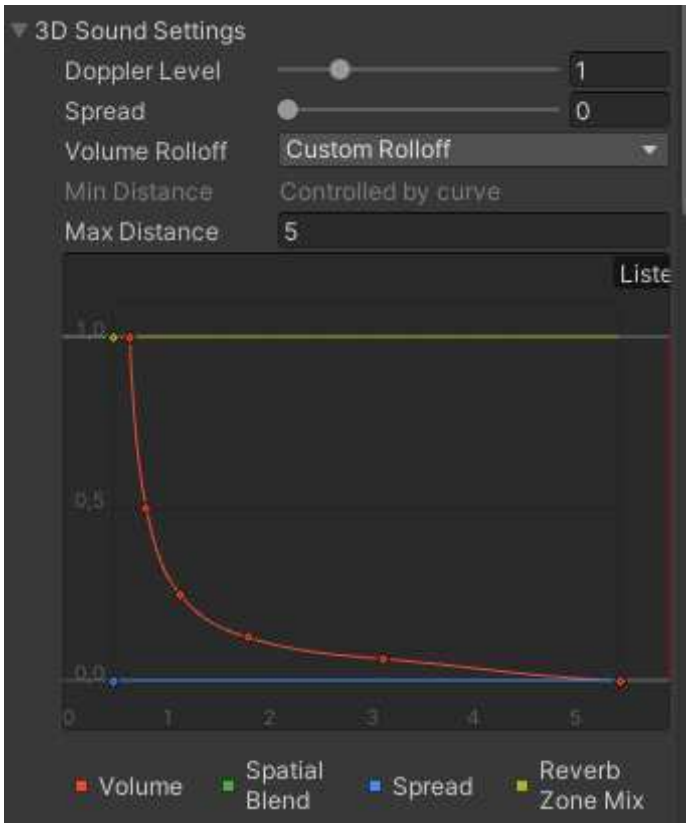

Figure 4.19: The *3D Sound Settings* for the cat meowing sounds

a randomly chosen meowing audio clip is played from the position of the chosen cat. If the sound has been turned off in the settings, none of the cats will produce sounds. This implementation is seen in Fig. 4.20.

```
1  void Update() {
2
3      if (!userTable.GetComponent<UserTableScript>().
           IsGameStarted()) {
4          return;
5      }
6
7      if (userTable.GetComponent<UserTableScript>().IsVictory()
           || PlayerPrefs.GetString("sound").Equals("soundOff")) {
8          return;
9      }
10
11     timer += Time.deltaTime;
12     double secondsBetweenMeows = startingSecondsBetweenMeows +
           Math.Pow(userTable.GetComponent<UserTableScript>().
           GetPoints(), 2) / 2.0;
13
14     if (timer >= secondsBetweenMeows) {
15
16         timer -= secondsBetweenMeows;
17
18         GameObject randomCat = userTable.GetComponent<
               UserTableScript>().GetRandomCat();
19
20         int next = r.Next(0, audioClips.Count);
21         randomCat.GetComponent<AudioSource>().clip =
               audioClips[next];
22         randomCat.GetComponent<AudioSource>().Play();
23     }
24 }
```

Figure 4.20: The *Update()* function of the script *Meowing*

4.2.4. Random Cat Initialization

There are a total of 125 cat hiding places around the Cat Cafe, but only nine are shown each round. Upon clicking the *Yes, I will help!* button shown in the dialogue box, the *SingleplayerInit()* function, seen in Fig. 4.21, will be called, which will make nine cats around the Cat Cafe visible and ready for the player to find.

```
1 public void SingleplayerInit() {
2     InitializeCats();
3     gameStarted = true;
4     welcomeSingleplayer.SetActive(false);
5     pointsCoop.SetActive(true);
6 }
```

Figure 4.21: The *SingleplayerInit()* function

The *InitializeCats()* function, called in line 2, is depicted in Fig. 4.22. This function randomly generates different numbers between 1 and the total number of available cats, which is 125. It has to generate the number of cats that is set, which is nine in the final build of CATch Cafe. After generating a number, it finds the cat with the corresponding number in its name using the function *FindChildFromParent()*, seen in Fig. 4.23, from the static class *Utility*. Upon finding that cat, the function sets that cat as active and puts it in a list of all active cats. The lines depicted with three dots are used for multiplayer synchronization and will be thoroughly explained in subsection 4.3.6.

```
1  public void InitializeCats() {
2          ...
3          List<int> randomNumbers = new List<int>();
4          GameObject catParent = GameObject.Find("Cats");
5
6          while (randomNumbers.Count < numberOfCats) {
7              int rInt = r.Next(1, rangeOfCats);
8              if (!randomNumbers.Contains(rInt)) {
9                  randomNumbers.Add(rInt);
10
11                 string catName = "Cat" + rInt;
12                 GameObject cat = Utility.FindChildFromParent(
                       catParent, catName);
13                 cat.SetActive(true);
```

```
                catObjects.Add(cat);
                ...
            }
        }
        ...
        guideAnimator.runtimeAnimatorController =
            idleAnimatorController;
    }
```

Figure 4.22: The *InitializeCats()* function

```
public static GameObject FindChildFromParent(GameObject parent
    , string name) {
    Transform[] trs = parent.GetComponentsInChildren<Transform
        >(true);

    foreach (Transform t in trs) {
        if (t.name.Equals(name)) {
            return t.gameObject;
        }
    }

    return null;
}
```

Figure 4.23: The *FindChildFromParent()* function from the static class *Utility*

4.2.5. Point Counting

When the nine random cats are initialized, the dialogue box of the main waitress changes to show the number of cats found by the player, depicted in Fig. 4.24, or, in other words, how many points the player has gotten. Additionally, the main waitress now changes to her idle animation.

When the player drops a cat onto the table in front of the waitress, the *CatCollision(GameObject collisionObject)* function is called. This happens since the *OnCollisionEnter(Collision collision)* function only calls this function if a cat collided, which

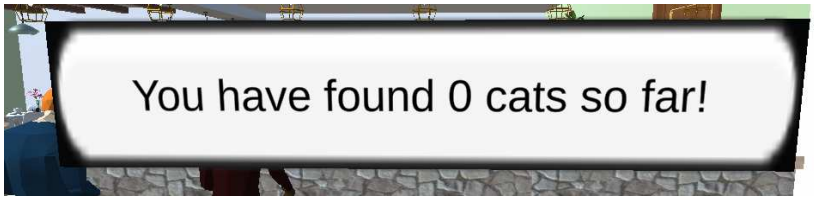

Figure 4.24: The dialogue box of the main waitress changes to count the number of cats found after they have been initialized.

is checked by making sure the name of the object that collided with the table starts with *Cat*. Another thing that is checked is that this cat has not already been counted as a point, as sometimes this function can be called multiple times before the execution of the code in the function has finished. The *CatCollision(GameObject collisionObject)* function, depicted in Fig. 4.25, does the following:

- adds the name of the cat to the list of cats counted as points,
- adds one point to the player's point count,
- calls the *UpdatePoints()* function which updates the text in the dialogue box of the waitress to show the correct number of points,
- saves the current colors of the body and the eyes of the cat dropped on the table,
- makes the cat that was dropped on the table invisible,
- finds the next cat that will be displayed on the table in front of the waitress by using the number of points,
- changes the colors of the found cat to the ones that were saved,
- changes the scale of the found cat to match the one that was dropped on the table,
- makes the found cat visible,
- removes the cat dropped on the table from the list of cats, so it can not be chosen to meow and
- if the player has not found all the cats yet, sets the value of *gotPoint* to *true*, which is checked in the *Update()* function and will make the main waitress perform a dance for five seconds, before going back to her idle animation.

```
1 // some code has been edited out, as it is used for
2 // multiplayer, which will be explained in subsection 4.3.6.
3 private void CatCollision(GameObject collisionObject) {
4     string name = collisionObject.name;
```

```
5       cats.Add(name);
6       if (points < numberOfCats) {
7           points++;
8           UpdatePoints();
9       }
10      GameObject body = Utility.FindChildFromParent(
            collisionObject, "Cat.L");
11      Material bodyMaterial = body.GetComponent<Renderer>().
            material;
12      GameObject leftEye = Utility.FindChildFromParent(
            collisionObject, "Cat.L_Eye.L");
13      Material leftEyeMaterial = leftEye.GetComponent<Renderer
            >().material;
14      GameObject rightEye = Utility.FindChildFromParent(
            collisionObject, "Cat.L_Eye.R");
15      Material rightEyeMaterial = rightEye.GetComponent<Renderer
            >().material;
16
17      collisionObject.SetActive(false);
18
19      GameObject finalCatParent = GameObject.Find("
            CatsFinalPosition");
20      string finalCatName = "CatFinal" + points;
21      GameObject finalCat = Utility.FindChildFromParent(
            finalCatParent, finalCatName);
22
23      GameObject finalBody = Utility.FindChildFromParent(
            finalCat, "Cat.L");
24      finalBody.GetComponent<Renderer>().material = bodyMaterial
            ;
25      GameObject finalLeftEye = Utility.FindChildFromParent(
            finalCat, "Cat.L_Eye.L");
26      finalLeftEye.GetComponent<Renderer>().material =
            leftEyeMaterial;
27      GameObject finalRightEye = Utility.FindChildFromParent(
            finalCat, "Cat.L_Eye.R");
28      finalRightEye.GetComponent<Renderer>().material =
            rightEyeMaterial;
```

```
29
30      finalCat.transform.localScale = collisionObject.transform.
            localScale;
31
32      finalCat.SetActive(true);
33
34      catObjects.Remove(collisionObject);
35
36      if (!IsVictory()) {
37          guide.transform.rotation = guideRotation;
38          guideAnimator.runtimeAnimatorController =
                pointAnimatorController;
39          gotPoint = true;
40          pointTimer = 0;
41      }
42  }
```

Figure 4.25: The *CatCollision(GameObject collisionObject)* function

Additionally, at the end of the *OnCollisionEnter(Collision collision)* function, the *CheckVictory()* function is called. This function checks whether the player found all nine cats in the Cat Cafe. If they have, the animation of the main waitress is changed to celebrate the victory of the player and her dialogue box now thanks the player for finding all of the cats, as depicted in Fig. 4.26.

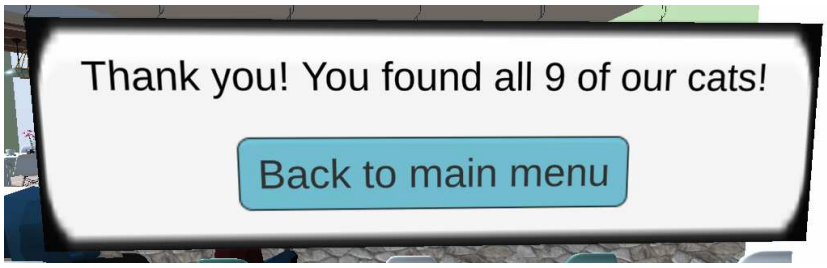

Figure 4.26: The dialogue box of the main waitress changes to thank the player after they have found all nine cats hidden around the Cat Cafe

From this dialogue box, the player can click *Back to main menu* to return to the Main Menu of the game.

4.2.6. Multiplayer Implementation

Photon Pun 2 was used to implement the collaborative and competitive versions of multiplayer mode in CATch Cafe [17]. The following *YouTube* tutorial and the two parts that come after it go show how to make a multiplayer VR game in *Unity* by using various functions from *Photon Pun 2*: https://www.youtube.com/watch?v=KHWuTBmT1oI. Additionally, some functions that were defined and described in the subsections that came before will be described here again since they were expanded to support multiplayer mode.

Connecting to a Server

Before connecting to a server, a new app was made in the *Photon Dashboard*: https://dashboard.photonengine.com/en-US/. This app was named CATch Cafe and this app's *App ID* will be used to allow players to connect to the same server. In *Unity*, by going to *Window*, then *Photon Unity Network* and finally clicking *Highlight Server Settings*, the current *Photon Server Settings* are shown in the inspector. The *App ID* given in the *Photon Dashboard* was copied into the *App Id PUN* field in the *Photon Server Settings*.

The script used to connect a player to the server is the *Network Manager* script. In the Main Menu, when the player clicks the *Play Multiplayer* button, the *ConnectToServer()* function is called, which calls the *PhotonNetwork.ConnectUsingSettings()* function and hides the Main Menu. Once the player has been connected to the server, the callback function *OnConnectedToMaster()* is called, which calls the *PhotonNetwork.JoinLobby()* function. After the player joins the lobby, all available lobbies are shown, those being the collaborative and competitive versions, as seen in Fig. 4.28.

Joining a Room

Inside of the *Unity* inspector, the *Network Manager* script is given two *Default Rooms*. One is named *Coop* with a scene index of 0. The other is named *Vs* with a scene index of 1. Both *Default Rooms* have their maximum player number set to 2. The *Default Rooms* structure was defined in the *Network Manager* script, as seen in Fig. 4.27, and the value of the competitive room can be seen in Fig. 4.29.

```
1 public class DefaultRoom {
2     public string Name;
3     public int sceneIndex;
4     public int maxPlayer;
```

```
5 }
```

Figure 4.27: The *Default Room* structure in the *Network Manager* script

Figure 4.28: The two lobbies in multiplayer mode, collaborative and competitive

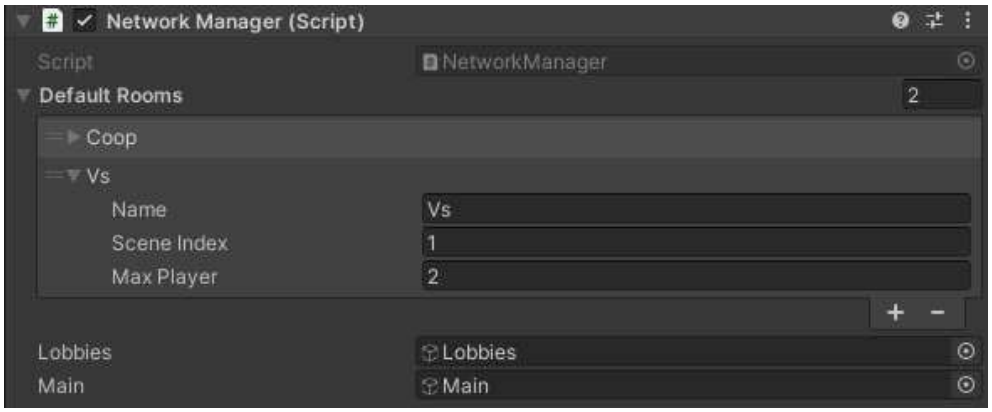

Figure 4.29: The competitive room values in the *Unity* inspector

When the player clicks either one of the rooms, the *InitializeRoom(int defaultRoomIndex)* function, depicted in Fig. 4.30, is called. This function picks the correct room using the given index and chooses the required scene with the room's scene index. The function then either creates the room, if one with the same name and properties did not exist up to this point, or joins the room if one already exists, with a maximum of two players being able to join the room at the same time.

```
public void InitializeRoom(int defaultRoomIndex) {
    DefaultRoom roomSettings = defaultRooms[defaultRoomIndex];

    PhotonNetwork.LoadLevel(roomSettings.sceneIndex);

    RoomOptions roomOptions = new RoomOptions();
    roomOptions.MaxPlayers = (byte) roomSettings.maxPlayer;
    roomOptions.IsVisible = true;
    roomOptions.IsOpen = true;

    PhotonNetwork.JoinOrCreateRoom(roomSettings.Name,
       roomOptions, TypedLobby.Default);
}
```

Figure 4.30: The *InitializeRoom(int defaultRoomIndex)* function

Player Presence

By following the above given *YouTube* tutorial to create a multiplayer VR game, hands were added to the player model. They change when the player presses the grip and trigger buttons. A *Network Player* prefab was made representing another player's presence in the same lobby and it is depicted in Fig. 4.31. It consists of a headset that follows the player's head and two hands that follow the player's controllers.

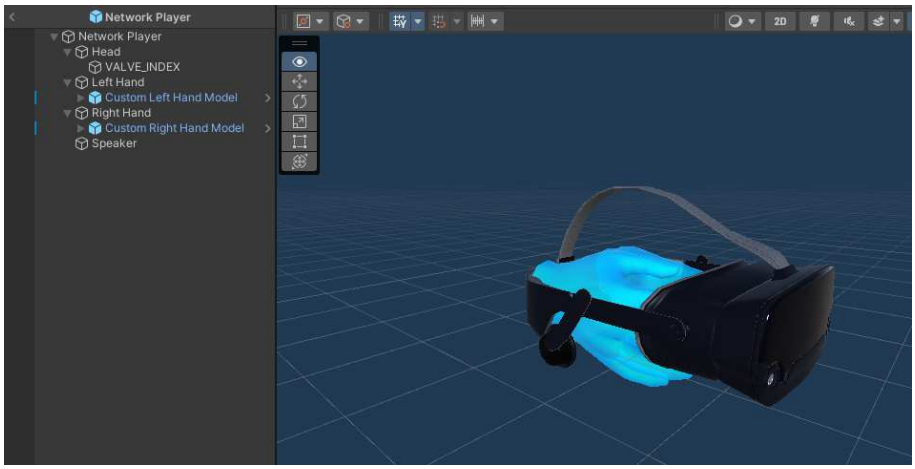

Figure 4.31: The prefab that will represent the other player in the room

The *Network Player* script, which is put on the *Network Player* prefab, maps the position and rotation of the headset and hands to the player's head and controllers. However, since the player does not need to see their own headset and already sees their own hands, the *Network Player* prefab that is used to show the player to other players in the same lobby is made invisible to the player who's presence it is.

Since the hands have animations depending on which buttons the player is pressing, the hand models have been given the *Photon Animator View* script, which synchronizes their animations for all players in the room. Because the hands and head also move and rotate according to where the players hold their controllers and move their head, the *Photon Transform View* script has been applied to them, with their position and rotation being synchronized. Finally, the *Photon View* script has been added to the parent of the *Network Player* prefab to synchronize all children of the prefab for the two players that will be in the same room. Each time a new player joins the lobby, they instantiate a prefab of the *Network Player* that acts as their player presence in the room. An example of how the current players sees the other player in the room is depicted in Fig. 4.32.

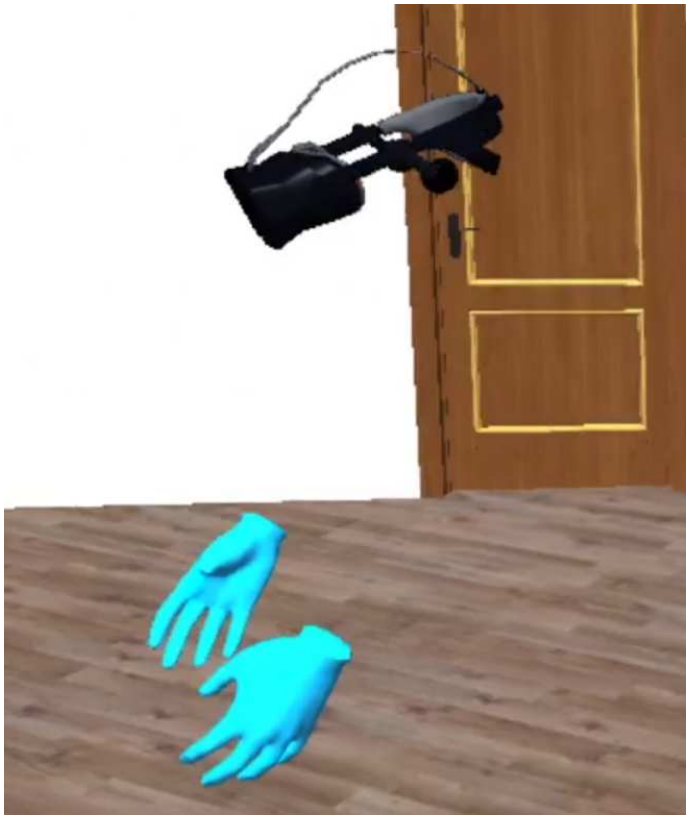

Figure 4.32: How the current player sees the other player

Random Cat Initialization in Multiplayer Mode

In order for all players to see how other players move and grab the cats around the Cat Cafe, the *Photon View* and *Photon Rigidbody View* scripts need to be added to each cat object. For the *Photon View* object, the *Ownership Transfer* value has to be set to *Takeover* to allow players to move cats around the Cat Cafe and have other players also see those cats move. Additionally, in the *Photon Rigidbody View* script, teleporting if distance of object is greater than 3, velocity synchronization and angular velocity synchronization have been enabled for better clarity of cat grabbing by players.

In order for other players to see when the current player picks up a cat, the *XR Grab Interactable* script that was added to each cat has to be changed. The reason is because only the player who initialized the room has ownership of the *PhotonView* components on all of the cat objects. If the other player moves any of the cats, the player who initialized the room will not see it. So, we put the *XR Grab Network Interactable* script, seen in Fig. 4.33, on each cat. This new script extends the *XR Grab Interactable* script and overrides the *OnSelectEntering(XRBaseInteractor Interactor)* to request ownership of the *PhotonView* of the cat object and allow the other player to see how the current player is moving and grabbing cats.

```
1  public class XRGrabNetworkInteractable : XRGrabInteractable {
2      private PhotonView photonView;
3
4      void Start() {
5          photonView = GetComponent<PhotonView>();
6      }
7
8      protected override void OnSelectEntering(XRBaseInteractor
           interactor) {
9          photonView.RequestOwnership();
10         base.OnSelectEntering(interactor);
11     }
12 }
```

Figure 4.33: The *XR Grab Network Interactable* script

Point Counting in Multiplayer Mode

In both multiplayer versions, the first player to click the *Yes, I/we will help!* button uses the *Custom Properties* of the room to set a variable letting the other player know they are ready to start the game. This is done by first fetching them using *Hashtable hash = PhotonNetwork.CurrentRoom.CustomProperties*. Next, a new entry is added to the hashtable using *hash.Add("firstPlayer", "ready")*. Finally, the updated hashtable is set as the new *Custom Properties* of the room with the line *PhotonNetwork.CurrentRoom.SetCustomProperties(hash)*. This allows the other player to read that value from the *Custom Properties* of the room once they start the game as well. While the first player waits for the second player to click the button as well, the text shown in Fig. 4.34 indicates that the other player is not ready yet.

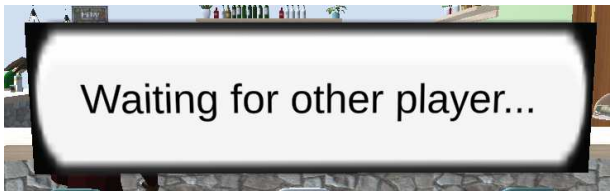

Figure 4.34: Text indicating the other player is not ready to start the game yet

For the player who clicked the button second, the *InitializeCats()* function, seen in Fig. 4.22, is called, with a few additions, visible in Fig. 4.35. The points of the players are added to the *Custom Properties* of the room, as well as the name of each cat initialized. The *PhotonNetwork.IsConnected()* function returns *true* if the player joined a server, or, in other words, if they are playing multiplayer mode.

```
1  public void InitializeCats() {
2      Hashtable hash = null;
3
4      if (PhotonNetwork.IsConnected) {
5          hash = PhotonNetwork.CurrentRoom.CustomProperties;
6          hash.Add("firstPlayerPoints", 0);
7          hash.Add("secondPlayerPoint", 0);
8          hash.Add("points", 0);
9      }
10     ...
11     while (randomNumbers.Count < numberOfCats) {
12         int rInt = r.Next(1, rangeOfCats);
13         if (!randomNumbers.Contains(rInt)) {
14             ...
```

```
15              catObjects.Add(cat);
16              if (PhotonNetwork.IsConnected) {
17                  catNames.Add(catName);
18                  hash.Add(catName, "HashCat" + catObjects.Count
                        );
19              }
20          }
21      }
22
23      if (PhotonNetwork.IsConnected) {
24          PhotonNetwork.CurrentRoom.SetCustomProperties(hash);
25      }
26      ...
27  }
```

Figure 4.35: The updates in the *InitializeCats()* function

The first player that clicked the button to start the game can now read from the *Custom Properties* of the room that the other player is ready as well. At that point, the *ShowCatsForOtherPlayer()* function, shown in Fig. 4.36, is called. It checks the names of all cats the other player has initialized and makes them visible as well.

```
1   public void ShowCatsForOtherPlayer() {
2       var hash = PhotonNetwork.CurrentRoom.CustomProperties;
3       GameObject catParent = GameObject.Find("Cats");
4
5       foreach (String catName in hash.Keys) {
6           if (!catName.StartsWith("Cat")) {
7               continue;
8           }
9           GameObject cat = Utility.FindChildFromParent(catParent
                , catName);
10          cat.SetActive(true);
11
12          catObjects.Add(cat);
13          catNames.Add(catName);
14      }
15
```

```
16      guideAnimator.runtimeAnimatorController =
            idleAnimatorController;
17  }
```

Figure 4.36: The *ShowCatsForOtherPlayer()* function

To be able to tell which player put a cat on the table in front of the waitress, in the *OnGrabAnimation()* function, which is called each time a player picks up a cat, the actor number of the local player is added to the *Custom Properties* of the room. This number changes in the *Custom Properties* of the room each time a different player than the last picks up one of the cats. Later, in the *CatCollision()* function, shown in Fig. 4.25, this value is checked for the multiplayer version. A point is then added to the corresponding player's current points.

The synchronization of cat positions may not always be completely correct over the network, causing one player to experience a collision of a cat and the table when the other player might not. Because of this, the *Update()* function continuously checks whether the number of points the players have stored locally is the same as the number of points stored in the *Custom Properties* of the room. If the number of points is different, the code depicted in Fig. 4.37 is executed. It finds which cats are no longer saved in *Custom Properties* of the room but still exist as in the list of cats and runs through all the code it would if a collision of the cat and the table in front of the waitress happened.

```
1   foreach (String catName in catNames) {
2       if (!hash.ContainsKey(catName)) {
3           GameObject cats = GameObject.Find("Cats");
4           CatCollision(Utility.FindChildFromParent(cats, catName
                ));
5
6           if (PhotonNetwork.CurrentRoom.Name.Equals("Coop")) {
7               UpdatePoints();
8           } else {
9               UpdateBothPlayerPoints();
10          }
11
12          CheckVictory();
13      }
```

```
14 }
```

Figure 4.37: Updating points if the number of locally stored points and point stored in the *Custom Properties* of the room differs

For the collaborative version, the point and victory texts stay the same as in singleplayer mode, as seen in Fig. 4.24 and Fig. 4.26. To win the collaborative version, players have to find all nine cats hidden around the Cat Cafe. For the competitive version, points are counted separately for each player. An example of this is shown in Fig. 4.38. The first player that finds five cats wins the game, since the other player can not find more cats than that player. The victory text if the current player wins the game is depicted in Fig. 4.39. After the game is over, players can return back to the Main Menu.

Figure 4.38: Point counting in the competitive version

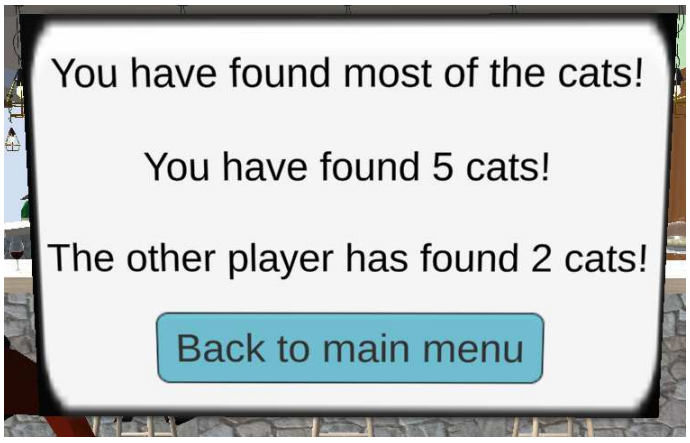

Figure 4.39: Victory text in the competitive version

4.3. Limitations

During the development of CATch Cafe, some problems were come across that were not able to be fixed for the duration of the development. None of these problems are game breaking, but their nature would lower the QoE for players.

4.3.1. Player Limit

CATch Cafe currently supports four simultaneous multiplayer players, two in the collaborative version and two in the competitive version. This is due to the fact that rooms do not generate dynamically. Rather, participants can enter one of two rooms, one for the collaborative version and one for the competitive version, which can both hold up to two players. Because this game was developed primarily to assess the QoE in multiplayer mode, which was always done in a laboratory with a single pair of individuals, this issue does not represent a risk such as losing the player base.

4.3.2. Grabbing Cats from Other Players

When one player is holding a cat and the other player snatches the cat from the first player's hand, the first player still sees the cat in their own hands. Once they release the cat, it teleports to the place where the cat actually is at that moment. Two potential solutions were explored when attempting to resolve this issue.

The first was that the other player unabled to snatch the cat from the first player at all. This would require removing or disabling the *XR Grab Network Interactable* script from the cat once it is picked up and re-adding or enabling it when the cat is dropped, however this proved unsuccessful. The second idea was to drop the cat from the first player's hand after the cat was taken from them by the other player. This likewise did not provide positive outcomes, as the cat could not be dropped from the first player's hand.

Because of this issue, the first player, who was carrying a cat, may have the cat snatched from them by the other player without even noticing. The second player could then drop the cat on the table in front of the waitress, which would count as a point for that second player, as they were the last player to hold the cat and to be recorded in the *Custom Properties* of the room. If the first player then dropped the same cat on the table, unbeknownst to the fact that the other player has previously received for that cat, they would expect to get a point. However, the code that checks whether the points stored locally matched those stored in the *Custom Properties* of the

room will award a point to the other player at that moment, as it may finally disable the cat the first player was holding.

4.4. Further Expansion of CATch Cafe Features

The current version of CATch Cafe is complete and fully playable. However, further expansion could raise the overall QoE of the VR game. Two ideas will be explored that could expand the original idea of CATch Cafe.

4.4.1. Multiple Rooms

Keeping the issue of limited rooms in mind, creating extra rooms would increase the number of people who could play the multiplayer version at the same time, improving the QoE for players who would otherwise be unable to play. Furthermore, allowing users to construct their own rooms would allow them to invite other players to join the same room as them. Finally, password-protecting particular rooms would allow players to play exclusively with those they want and invite.

4.4.2. Voice Chat

When testing the game, two players were always in the same real-world room, allowing them to communicate with one another. However, if two players wanted to play the game but were not nearby, they would be unable to communicate. Adding voice chat to the game would assist with this since they would be able to converse without the use of other programs.

5. User Study

5.1. Study Motivation and Methodology

The motivation for conducting a user study was to investigate the impact of the chosen version of multiplayer mode on the overall user experience. The user study involved 8 participants aged from 20 to 22, with the average age being 21. 5 of the participants were men and 3 of them were women. The testing procedure, including the beginning explanations, lasted around 45 minutes. The questionnaire used in the user study is given in Appendix A.

The participants always came as a pair, so a total of 4 pairs of participants were involved in the study. Each participants first briefly played singleplayer mode with the beginning explanations of the game and the controls. Then, they filled out the general questions in the questionnaire. Next, each of them played one full round of singleplayer mode, which allowed them to fill out the questions regarding singleplayer mode. Next, the pair played the collaborative version and filled out the questions for that version. Then, they played the competitive version and filled out the questions regarding that version. Finally, participants filled out the last part of the questionnaire in which they compared the two multiplayer version they had previously played.

5.2. The Results

The average time needed to beat the singleplayer version for all participants was 5 minutes and 16 seconds. The average time needed for the participants to find collectively all nine cats in the collaborative version was 3 minutes and 48 seconds. The average time for one player to find five cats and win the game in the competitive version was 1 minute and 14 seconds.

As seen in Fig. 5.1, the level of gaming experience among the participants greatly varies. As depicted in Fig. 5.2, half of the participants never had any experience

with VR technology before and most of them said their overall level of experience is considerably low. All players stated the controls for the game were very intuitive, as depicted in Fig. 5.3.

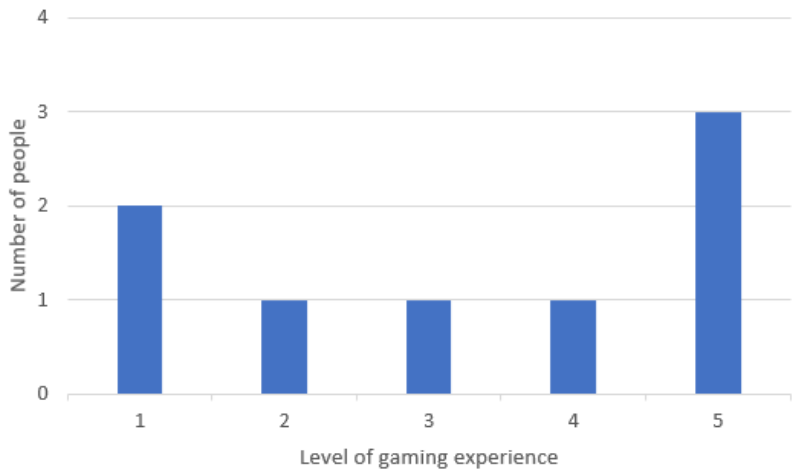

Figure 5.1: Level of gaming experience among the participants (1 Beginner — 5 Experienced)

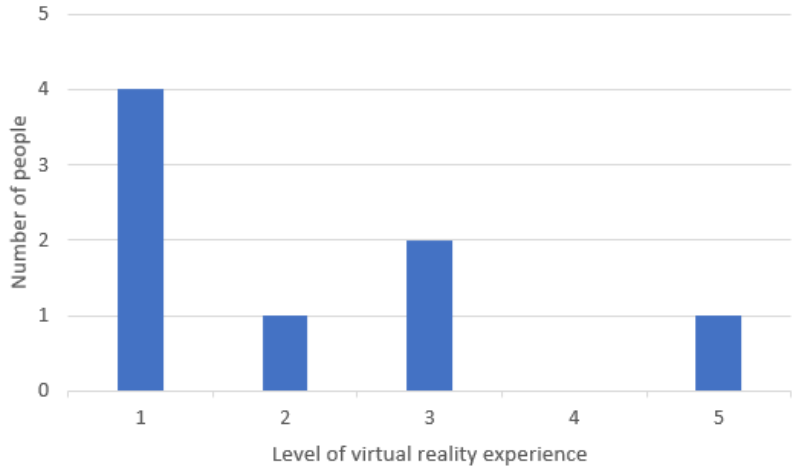

Figure 5.2: Level of virtual reality experience among the participants (1 Beginner — 5 Experienced)

When asked about their proneness to nausea, most participants stated they are not very prone to nausea, as seen in Fig. 5.4. Along with that, 5 participants did not feel any discomfort after playing all of the rounds, but the other 3 stated they felt a little nauseated at the end, as depicted in Fig. 5.5.

The overall QoE was lower for singleplayer mode than for the multiplayer modes, but participants rated the overall QoE for the two multiplayer modes the same, as depicted in Fig. 5.6.

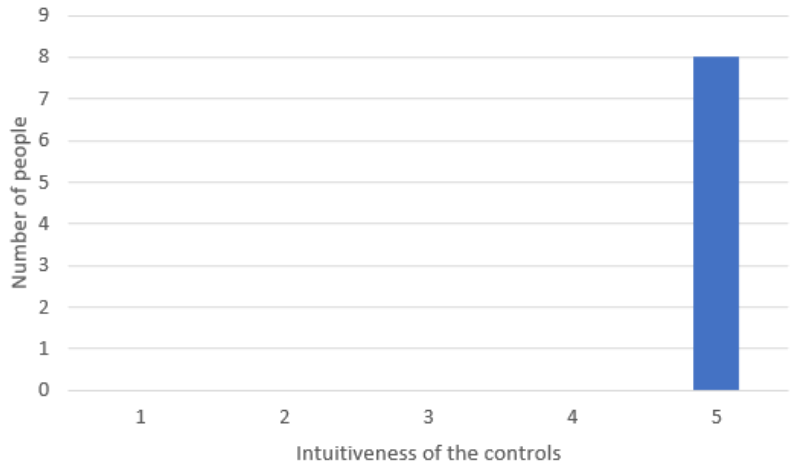

Figure 5.3: Level of intuitiveness of the controls among the participants (1 Very counter intuitive - 5 Very intuitive)

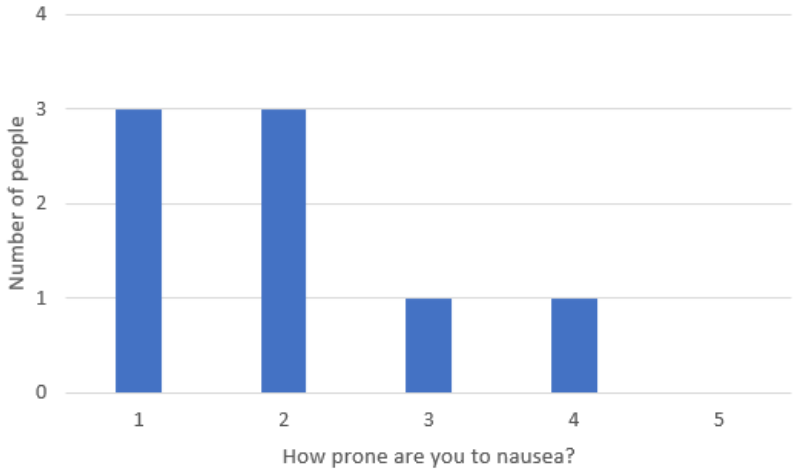

Figure 5.4: Proneness to nausea among the participants (1 Not prone at all - 5 Very prone)

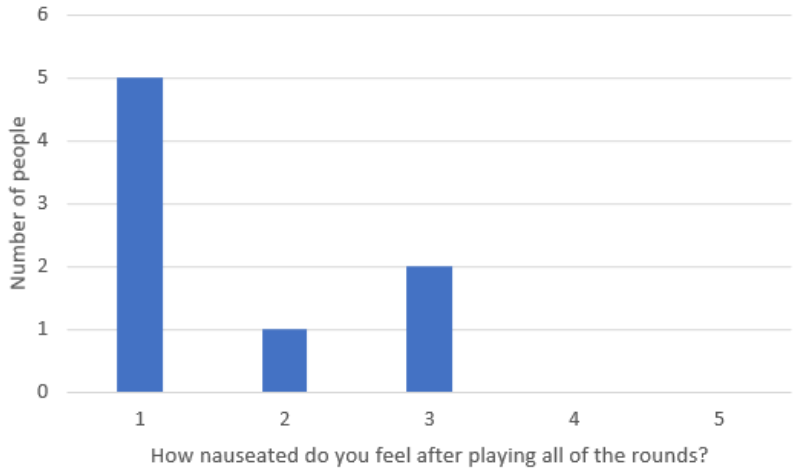

Figure 5.5: Level of nausea among the participants after playing all rounds (1 I don't feel nauseated at all - 5 Very nauseated)

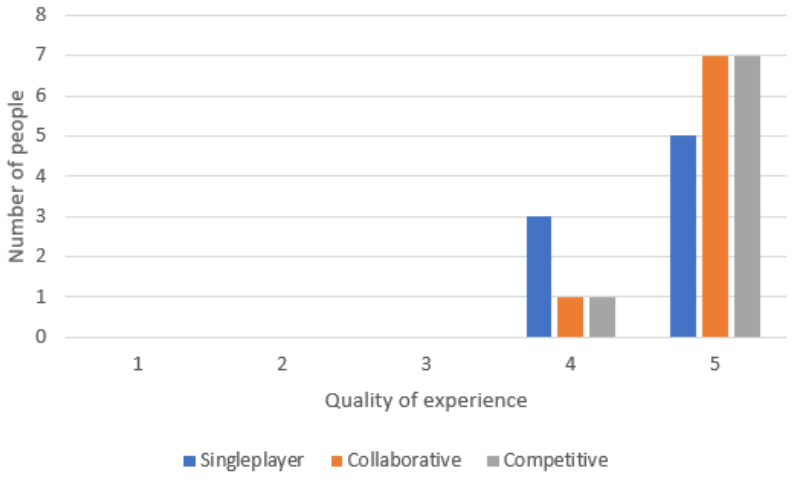

Figure 5.6: The overall QoE for all rounds (1 Very bad - 5 Excellent)

In the collaborative version, all participants enjoyed playing with the other player, as seen in Fig. 5.7. 5 of the participants also stated that this version made their view on the game a bit better with respect to the singleplayer version, as depicted in Fig. 5.8, while 2 stated their view on the game was much better. 1 participant said their view did not change after playing the collaborative version.

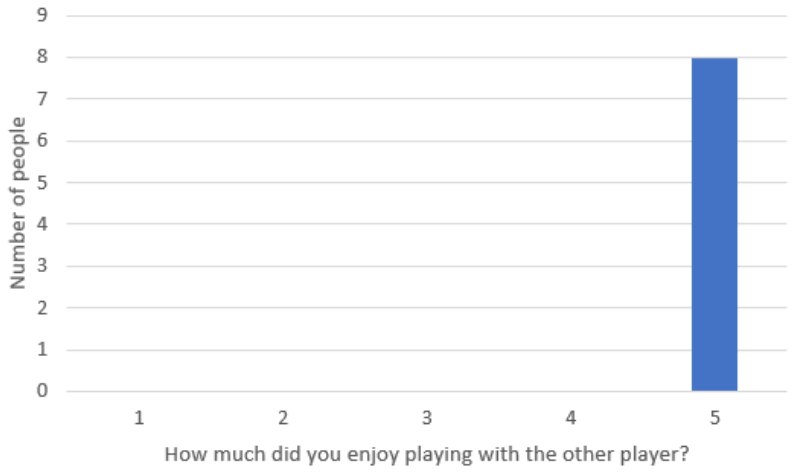

Figure 5.7: The level of enjoyment when playing with the other player in the collaborative version (1 I didn't enjoy at all - 5 I enjoyed it a lot)

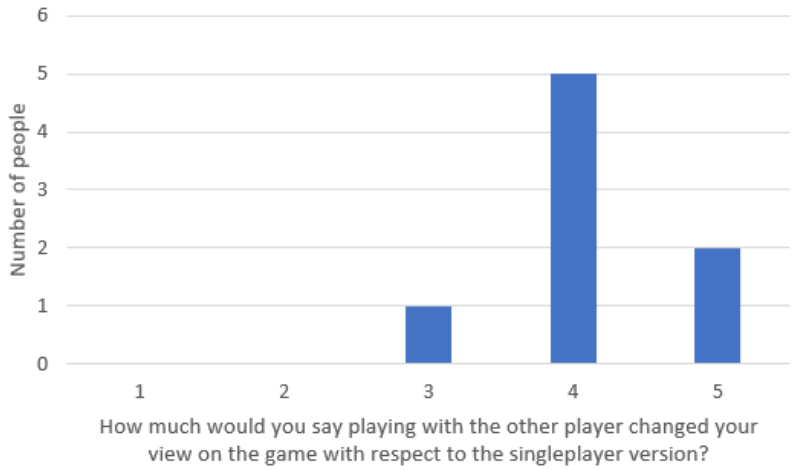

Figure 5.8: The level at which playing the collaborative version changed the participant's view on the game in respect to the singleplayer version (1 It worsened it a lot - 5 It improved it a lot)

In the competitive version, 5 participants enjoyed playing with the other player, as seen in Fig. 5.9, while 2 of them enjoyed it a bit and 1 participant stated they neither enjoyed it nor hated it. 5 of the participants also stated that this version made their view on the game a bit better in respect to the singleplayer version, as depicted in Fig.

5.10, while 2 stated their view on the game was much better. 1 participant said their view did not change after playing the competitive version.

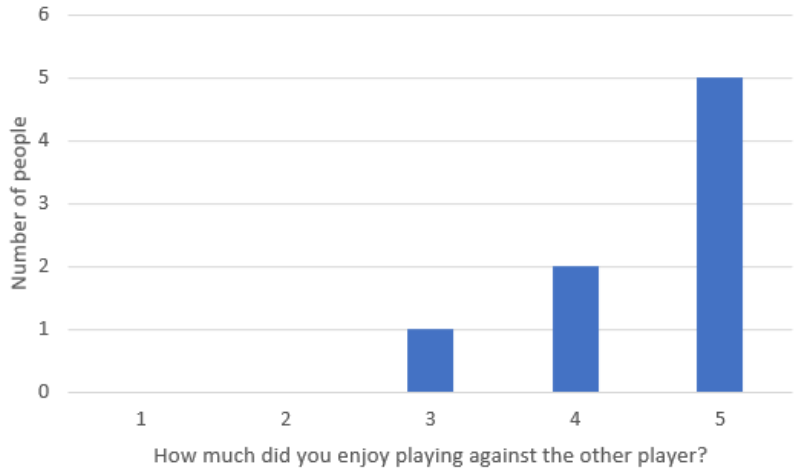

Figure 5.9: The level of enjoyment when playing against the other player in the competitive version (1 I didn't enjoy at all - 5 I enjoyed it a lot)

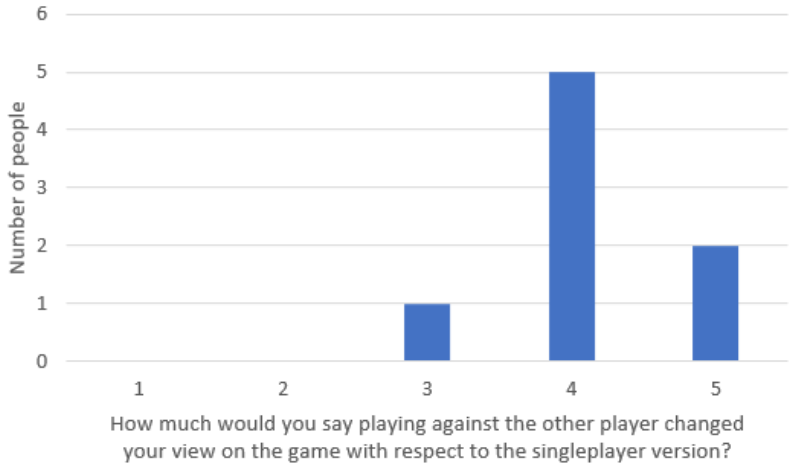

Figure 5.10: The level at which playing the competitive version changed the participant's view on the game in respect to the singleplayer version (1 It worsened it a lot - 5 It improved it a lot)

When asked to compare the two multiplayer version, half of the participants stated they did not enjoy one or the other more, as depicted in Fig. 5.11. 1 participant enjoyed the competitive version a bit more, while 2 of them enjoyed the collaborative version a bit more. 1 participant enjoyed the collaborative version significantly more than the competitive one.

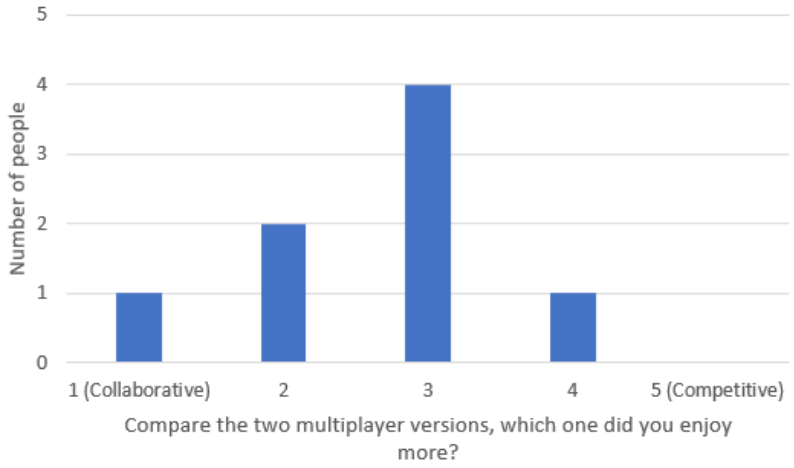

Figure 5.11: The level at which participants enjoyed one multiplayer version over the other (1 I enjoyed cooperative much more - 5 I enjoyed competitive much more)

5.3. Analysis of the User Study

The average time needed to complete a round in multiplayer mode was lower than in singleplayer mode. This was expected, as it should take more people less time to find the same amount of cats. The surprising data here is that the time needed for players to complete the collaborative version is almost three times longer than the time needed for one of the players to complete the competitive version. Arguably, since the winner of the competitive version needed to find only five cats instead of all nine, this could explain why the time is so much shorter. However, in total, in 3 of the 4 rounds of the competitive version assessed in the user study all nine cats were found by the two players. In 1 round where this was not the case, eight cats were found. This shows that even though players still found most, if not all, of the cats in competitive mode, the time was still much shorter than in the collaborative version, where all of the cats needed to be found. An explanation of this would be that the participants felt a need to win against the other player and therefore performed better at finding the cats.

Participants who rated their level of gaming experience higher than that of their opponent were usually the ones who won the competitive version. The only exception is one in which both participants rated their level of gaming experience equally. However, the same pattern could not be noticed with the level of VR experience. In 4 pairs of participants, the player with more experience in VR won the game in 2 of those pairs, the player with less experience won the game in 1 of them and in the last pair neither of the participants had any experience with VR. The most probable reason for this is that there were not enough participants tested in the user study.

2 of the 8 participants, who were the only ones that rated their proneness to nausea at 3 or above, also stated they felt a bit nauseated after the user study was over. Additionally, neither of them have had any experience with VR before playing CATch Cafe, so this may also play a factor in their discomfort, as they are not used to wearing VR headset and being inside of a virtual environment. 3 participants rated their level of virtual reality experience at 3 or above and none of them experienced nausea at any point during the game. However, all of them rated their proneness to nausea at level 1 or 2, so more participants are needed to see whether virtual reality experience really lowers feelings of nausea while wearing a VR headset and playing a game. One participant who rated their proneness to nausea as level 1 experienced slight nausea after completing the user study, but they also stated they have no experience in virtual reality. Therefore, this feeling of nausea could come from the inexperience with being in virtual worlds.

Since all participants rated the overall QoE for the multiplayer mode higher or the same as for the singleplayer mode, this shows that CATch Cafe is more enjoyable when played with other players. All players enjoyed playing with the other player in the collaborative version and most preferred it over playing alone in singleplayer mode. The one player who did not had no preference over one or the other. The reason why the collaborative version might be more enjoyable over singleplayer mode could be because people can play with their friends and they can work together to find all of the cats. In all pairs that were tested, participants communicated and tried to work out the best way to find all the cats, for example one player would search one part of the Cat Cafe, while the other player would search the second part.

While all participants enjoyed playing with the other player, when playing against them, 5 participants said they still enjoyed themselves, while 2 participants enjoyed it, but not fully, and 1 participant neither enjoyed it nor hated it. Even though players enjoyed playing with the other player more than playing against them, the rating of all participants of how playing against the other player changed their view on the game with respect to the singleplayer version is the same as when asking how playing with the other player changed their view. Not all participants answered the same, but all ratings have the same amount of answers.

When comparing whether the user won the competitive round with the enjoyment of playing against the other player, no pattern can be seen, as there are not enough participants to make a clear distinction. The same goes when comparing this to how the participant's view changed in respect to the singleplayer version when looking at both the collaborative and competitive versions.

Overall, participants enjoyed the collaborative version more than the competitive one. However, players who lost the round in the competitive version generally enjoyed the collaborative version more than the ones who won. The average score of the participants who lost was 2.25, while this score was 3 for those that won. This is probably because those that won the game enjoyed the round more through their victory, while those who lost felt the opposite effect. Participants commented that the collaborative version was more enjoyable because they could communicate with the other player and achieve great teamwork. Those that lost the competitive version stated they liked the collaborative one more because they did not enjoy losing. Another comment was that while in the collaborative version the ending was always known, which is that the players will eventually win, in the competitive version, players did not know who would win in advance, which made it more exciting.

6. Conclusion

VR technologies are continually being enhanced in order to provide a better experience for every single user. They function by obscuring the user's vision of the actual world, typically using headsets, and immersing the user in a totally virtual environment created with specialized VR software. Despite the fact that VR may be used to replicate numerous hazardous circumstances, one of its most prevalent applications today is in the gaming industry. There are several factors that might degrade a player's QoE when playing VR games, including screen display latency, immersion level, and control intuitiveness. In multiplayer games, action latency and jitter can also be one of those factors.

The VR multiplayer game CATch Cafe was created in this thesis to investigate the impact of collaborative and competitive versions on the QoE of the user. The singleplayer mode of the game requires finding hidden cats throughout a cat cafe and bringing them all to a table. Once the player has found all of the cats, the game is ended and may be restarted. Every time the game is restarted, the cats hide in different locations around the cat cafe, increasing replayability. Furthermore, the game has a multiplayer option that may be played collaboratively or competitively. In the collaborative version, players must discover all of the cats together, whilst in the competitive version, the person who finds the most cats wins.

A user study was conducted that investigated the impact of the chosen multiplayer version on the user experience. The results show that, while the game is great in singleplayer mode, it is far more enjoyable in multiplayer mode. When working together or against each other, players needed substantially less time to find all of the cats than when playing alone. Overall, participants preferred the collaborative version over the competitive one.

BIBLIOGRAPHY

[1] What is a cat cafe?, . URL https://www.theneighborscat.com/blog/2017/11/20/what-is-a-cat-cafe. last accessed on 07/06/2022.

[2] All countries, . URL https://www.theneighborscat.com/all-countries. last accessed on 07/06/2022.

[3] Unity. URL https://unity.com/. last accessed on 07/06/2022.

[4] Basu A. A brief chronology of virtual reality. 2019.

[5] Buczkowski A. Google cardboard – paper virtual reality set now supports street view. 13 October 2015.

[6] Davies A. 10 great tools for vr development.

[7] Kovačević A. How does virtual reality (vr) technology work? 2021.

[8] Adorama. How virtual reality works. 19 June 2018.

[9] Onkar P. S. Arya A. K. Presenting tangible heritage throughvirtual reality in education contexts. 2017.

[10] Gleb B. Vr vs ar vs mr: Differences and real-life applications. 4 January 2020.

[11] bestware. Manus prime ii. URL https://bestware.com/en/manus-prime-two.html. last accessed on 17/05/2022.

[12] Coiffet P. Burdea G. C. *La réalité virtuelle*. Hermès, 1993. ISBN 9782866013868. URL https://books.google.hr/books?id=sPj6PAAACAAJ.

[13] Coiffet P. Burdea G. C. *Virtual reality technology*. John Wiley & Sons, 2003.

[14] Petrov C. 45 virtual reality statistics that will rock the market in 2022. 3 June 2022.

[15] Barnard D. *History of VR - Timeline of Events and Tech Development*. URL https://virtualspeech.com/blog/history-of-vr.

[16] Unity Documentation. Xr interaction toolkit. URL https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html. last accessed on 02/06/2022.

[17] Photon Engine. Pun. URL https://www.photonengine.com/pun. last accessed on 02/06/2022.

[18] Beat Games. Beat saber. URL https://beatsaber.com/. last accessed on 31/05/2022.

[19] Software Testing Help. What is virtual reality and how does it work, 2022. URL https://www.softwaretestinghelp.com/what-is-virtual-reality/. last accessed on 28/05/2022.

[20] Microsoft HoloLens. Microsoft hololens: Holostudio. URL https://www.youtube.com/watch?v=BRIJG0x_We8. last accessed on 16/05/2022.

[21] Rec Room Inc. Rec room. URL https://recroom.com/. last accessed on 31/05/2022.

[22] The Franklin Institute. What's the difference between ar, vr, and mr? URL https://www.fi.edu/difference-between-ar-vr-and-mr. last accessed on 12/04/2022.

[23] Pillai J. S. Ismail I. Virtual reality: Introduction. URL https://www.dsource.in/course/virtual-reality-introduction/evolution-vr/telesphere-mask. last accessed on 30/03/2022.

[24] ITU-T Recomm. G.1032. Influence Factors on Gaming Quality of Experience. *Int. Telecomm. Union-Telecomm. Standard. Sect.*, 2017.

[25] ITU-T Recomm. P.809. Subjective evaluation Methods for Gaming Quality. *Int'l Telecomm, Union-Telecomm. Standard. Sect.*, 2018.

[26] Bardi J. What is virtual reality: Definitions, devices, and examples. 29 March 2022.

[27] Gupta J. How virtual reality is transforming the gaming industry. 8 April 2021.

[28] Turi J. The sights and scents of the sensorama simulator. 2014.

[29] Willis J. Librivox: Pygmalion's spectacles by stanley g. weinbaum. 2021.

[30] Feyerabend L. Vr's motion sickness problem — and how to hack it (with the help of your cat). 4 September 2017.

[31] Rosenberg L. Vr vs. ar vs. mr vs. xr: What's the difference? 18 January 2022.

[32] Tremosa L. Beyond ar vs. vr: What is the difference between ar vs. mr vs. vr vs. xr? February 2022.

[33] Lenovo. What is virtual reality gaming. URL https://www.lenovo.com/us/en/faqs/gaming/what-is-virtual-reality-gaming/. last accessed on 31/05/2022.

[34] Merriam-Webster. Virtual reality. URL https://www.merriam-webster.com/dictionary/virtual%20reality. last accessed on 29/03/2022.

[35] Kishino F. Milgram P. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[36] Mixamo. Animations. URL https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack. last accessed on 03/06/2022.

[37] Nathie. Rec room vr on the oculus quest! URL https://www.youtube.com/watch?v=nK1WOUiwiF4. last accessed on 31/05/2022.

[38] Inc. Niantic. Pokémon go. URL https://pokemongolive.com/en/. last accessed on 16/05/2022.

[39] House of Play. Virtual reality – virtuix omni. URL https://www.houseofplay.com/store/virtuix-omni. last accessed on 17/05/2022.

[40] Beat Saber Official. Electronic mixtape | release trailer. URL https://www.youtube.com/watch?v=ja67hoanefY. last accessed on 31/05/2022.

[41] PlayStation. URL https://www.playstation.com/en-us/ps-vr/. last accessed on 17/05/2022.

[42] Carter R. How does virtual reality work? 14 June 2021.

[43] Skarredghost. Everything we know about the new oculus quest (...and oculus go?). 26 July 2020.

[44] Arm Blueprint Staff. xr, ar, vr, mr: What's the difference in reality? 1 April 2022.

[45] GameSpot Staff. Sony will 'probably reject' playstation vr games that run below 60fps. 17 March 2016.

[46] Unity Asset Store. Café hotel, . URL https://assetstore.unity.com/packages/3d/props/interior/caf-hotel-162023. last accessed on 02/06/2022.

[47] Unity Asset Store. Distant lands free characters, . URL https://assetstore.unity.com/packages/3d/characters/distant-lands-free-characters-178123. last accessed on 02/06/2022.

[48] Unity Asset Store. Low poly cats, . URL https://assetstore.unity.com/packages/3d/characters/animals/mammals/low-poly-cats-185996. last accessed on 02/06/2022.

[49] Unity Asset Store. Pun 2 - free, . URL https://assetstore.unity.com/packages/tools/network/pun-2-free-119922. last accessed on 02/06/2022.

[50] Unity Asset Store, . URL https://assetstore.unity.com/. last accessed on 02/06/2022.

[51] Unity Asset Store. Vr headset vol - 2, . URL https://assetstore.unity.com/packages/3d/props/vr-headset-vol-2-161106. last accessed on 02/06/2022.

[52] Elara Systems. Vr, ar, mr: What are the difference between vr and ar and mr? URL https://elarasystems.com/vr-ar-mr-difference/. last accessed on 12/04/2022.

[53] English T. Vr headsets work through a combination of different tracking technologies. 9 May 2020.

[54] Aniwaa team. The ultimate vr, ar, mr guide. 6 August 2021.

[55] TechTarget. virtual reality gaming (vr gaming). URL https://www.techtarget.com/whatis/definition/virtual-reality-gaming-VR-gaming#:~:text=Virtual%20reality%20gaming%20is%20the,artificial%20environment%20to%20computer%20games. last accessed on 31/05/2022.

[56] Wikipedia. C sharp (programming language), . URL https://en.wikipedia.org/wiki/C_Sharp_(programming_language). last accessed on 02/06/2022.

[57] Wikipedia. Microsoft visual studio, . URL https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. last accessed on 02/06/2022.

[58] Wikipedia. Unity (game engine), . URL https://en.wikipedia.org/wiki/Unity_(game_engine). last accessed on 02/06/2022.

[59] Gibson I. Zheng J. M., Chan K. W. Virtual reality. *Ieee Potentials*, 17(2):20–23, 1998.

List of Figures

ABBREVIATIONS

AR *Augmented Reality*

DoF *Degrees of Freedom*

XR *Extended Reality*

FOV *Field of View*

FPV *First-Person View*

FPS *Frames per Second*

GPU *Graphics Processing Unit*

HMD *Head Mounted Display*

MR *Mixed Reality*

PC *Personal Computer*

QoE *Quality of Experience*

SDK *Software Development Kit*

3D *Three-Dimensional*

VR *Virtual Reality*

Appendix A
User Study

User study for Bachelor Thesis "Development of a Multiplayer Hide-and-Seek Virtual Reality Game"

The information and responses gathered by this user data will be used solely for research reasons as part of the Bachelor Thesis "Development of a Multiplayer Hide-and-Seek Virtual Reality Game." Because all data will be evaluated collectively, your personal information will be kept anonymous.

General questions

Name and surname: _____

Age in years: _____

Gender:

☐ Woman

☐ Man

☐ Transgender woman

☐ Transgender man

☐ Prefer not to answer

☐ Other: _____

Level of gaming experience: **(Beginner) 1 — 5 (Experienced)**

Level of virtual reality experience: **(Beginner) 1 — 5 (Experienced)**

How prone are you to nausea? **(Not prone at all) 1 — 5 (Very prone)**

Singleplayer version

In the singleplayer version, you will be playing a match alone. After playing, fill out this section of the user study. The examiner will tell you the duration of the round.

How would you rate the overall Quality of Experience of the singleplayer version? **(Very bad) 1 — 5 (Excellent)**

Duration of the round: _____

How intuitive were the controls for the game? **(Very counter intuitive) 1 — 5 (Very intuitive)**

Collaborative version

In the collaborative version, you will be playing a match along with the other player. After playing, fill out this section of the user study. The examiner will tell you the duration of the round.

How would you rate the overall Quality of Experience of the collaborative version? **(Very bad) 1 — 5 (Excellent)**

Duration of the round: _____

How much did you enjoy playing with the other player? **(I did not enjoy at all) 1 — 5 (I enjoyed it a lot)**

How much would you say playing with the other player changed your view on the game with respect to the singleplayer version? **(It worsened it a lot) 1 — 5 (It improved it a lot)**

Competitive version

In the competitive version, you will be playing a match against the other player. After playing, fill out this section of the user study. The examiner will tell you the duration of the round.

How would you rate the overall Quality of Experience of the competitive version? **(Very bad) 1 — 5 (Excellent)**

Did you win the round?

☐ Yes

☐ No

Duration of the round: _____

How much did you enjoy playing against the other player? **(I did not enjoy at all) 1 — 5 (I enjoyed it a lot)**

How much would you say playing against the other player changed your view on the game with respect to the singleplayer version? **(It worsened it a lot) 1 — 5 (It improved it a lot)**

Questions after all the rounds

Fill out this part of the user study after you have played the singleplayer, collaborative and competitive rounds.

How nauseated do you feel after playing all of the rounds? **(I do not feel nauseated at all) 1 — 5 (Very nauseated)**

Compare the two multiplayer versions, which one did you enjoy more? **(I enjoyed collaborative much more) 1 — 5 (I enjoyed competitive much more)**

If you enjoyed one more than the other, why do you think that is?:

_____

The end of the user study

Thank you for taking part of this user study!

Development of a Multiplayer Hide-and-Seek Virtual Reality Game

Abstract

This thesis delves into the concept and history of virtual reality. Some of the key aspects of virtual reality are discussed, as well as the most prevalent uses of this technology, such as virtual reality video games. The quality of experience in multiplayer virtual reality games is briefly explained. The *Unity* game engine is used to create CATch Cafe, a multiplayer virtual reality game. The *XR Interaction Toolkit* is utilized to allow players to interact with the game through the use of virtual reality equipment. Finally, *Photon Pun 2* is employed in the creation of two multiplayer versions of the game, a collaborative and a competitive one. A user study is conducted to evaluate the quality of experience of the multiplayer component in the developed game.

Keywords: VR, Unity, Photon Pun 2, XR Interaction Toolkit, multiplayer games, video games, quality of experience

Razvoj višekorisničke igre skrivača uporabom tehnologije virtualne stvarnosti

Sažetak

Ovaj završni rad bavi se konceptom i poviješću virtualne stvarnosti. Raspravlja se o nekim od ključnih karakteristika virtualne stvarnosti, kao i o najčešćim uporabama ove tehnologije, kao što su video igre u virtualnoj stvarnosti. Kratko je objašnjena iskustvena kvaliteta u višekorisničkim video igrama u virtualnoj stvarnosti. Razvojna okolina za igre *Unity* koristi se za stvaranje CATch Cafe, višekorisničke video igre u virtualnoj stvarnosti. *XR Interaction Toolkit* koristi se kako bi se igračima omogućila interakcija s igrom korištenjem opreme za virtualnu stvarnost. Konačno, *Photon Pun 2* koristi se u stvaranju dvije višekorisničke verzije video igre, koje su kolaborativna i kompetitivna. Provodi se korisnička studija kako bi se ocijenila iskustvena kvaliteta višekorisničke komponente u razvijenoj video igri.

Ključne riječi: VR, Unity, Photon Pun 2, XR Interaction Toolkit, višekorisničke igre, video igre, iskustvena kvaliteta