Master's Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

VR games for lying patients

Bc. Tereza Langová

Supervisor: Ing. David Sedláček, Ph.D.
Field of study: Open Informatics
Subfield: Computer Graphics
January 2025

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Langová Tereza**	Personal ID number:	**484890**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Graphics and Interaction**		
Study program:	**Open Informatics**		
Specialisation:	**Computer Graphics**		

II. Master’s thesis details

Master’s thesis title in English:

VR games for lying patients

Master’s thesis title in Czech:

VR hry pro ležící pacienty

Guidelines:

1) Design a VR gaming experience for relaxation and gentle exercise for lying patients (discuss/test different lying/reclining positions). Assume elderly patients or patients with a spinal cord injury, non-violent VR environment.
2) Implement a prototype to test possible interactions with lying patients, test with users.
3) According to the design (1) and the results of the prototype (2), design a VR application with at least three different environments and main interaction mechanics. Then design and implement a mobile application to control the environment and connect it to the VR application. Expect to use the Meta Quest VR platform.
4) Implement the connection of your app to a portal for storing activities in VR [4] and adapt the visualisations on this portal to your needs.
5) Follow the UCD methodology for design and implementation. Test the application with at least five users from the target group.

Bibliography / sources:

1] Jason Jerald. 2015. The VR Book: Human-Centered Design for Virtual Reality. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA.
2] Steven M. LaValle - Virtual Reality, Cambridge University Press 2016
3] T. Lowdermilk, User-Centered Design, O’Reilly Media, 2013
4] Leoš Řeháček, Aplikace pro sběr a analýzu dat z VR tréninkových aplikací. DP ČVUT FEL, 2024.

Name and workplace of master’s thesis supervisor:

Ing. David Sedláček, Ph.D. Department of Computer Graphics and Interaction FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **16.02.2024** Deadline for master's thesis submission: **07.01.2025**

Assignment valid until: **21.09.2025**

Ing. David Sedláček, Ph.D.	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I am grateful to Ing. David Sedláček, Ph.D. for his guidance, willingness to help and for providing me with contacts to the institutions mentioned in the thesis. I would also like to thank all of the employees of the institutions, who helped me understand their clients' needs and limitations and aided me while I tested the application. The thesis could not be completed without the willingness to participate in the testing and patience of the clients in the institutions, so I owe them a big thank you. Last but not least, I am grateful to my partner, family, and friends who supported me when I needed it.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 6 January 2025

Abstract

This thesis focuses on the design of a virtual reality (VR) application that is suitable for entertainment and gentle exercise for lying patients. It aims to first explore the possible target user groups, and based on the analysis, to design suitable game experiences, their controls, and mechanics for the selected target group. Subsequently, a prototype of the proposed application will be implemented. Further, according to the UCD methodology, the prototype has to be tested, appropriate changes suggested and implemented and this process iterated several times. The resulting application and the included game experiences should be suitable for elderly people, people with spinal cord injuries, or otherwise ill (and therefore bedbound) patients.

Throughout the design and implementation process great emphasis is placed on the limitations of the target group. Many of these limitations are physical, such as reduced sensitivity in the hands and fingers, weak hand grip and overall weakened arm, or inability to turn the head and body due to the lying position, etc. The users must be able to control the application comfortably regardless of these limitations.

The resulting app allows the player to perform upright redirection so they see themselves as if standing and supports gesture detection instead of VR controllers. An associated mobile device application allows the assistant to help the VR player with navigation in the menus, upright redirection, setting custom gestures, and more. Three games can be played in the application – Tangram puzzle, Shooting Gallery, and an Endless Runner game. They are designed to appeal to a wide spectrum of users with different kinds of physical limitations.

The application was tested twice throughout its development and changed according to the results of these testing sessions, then tested for the last time to verify it fulfills the given requirements. According to the final testing the application is usable and entertains the users.

In the future, some minor changes could be made according to the final testing of the application. The app could also be expanded to include other gaming experiences, and more Tangram and Shooting Gallery levels could be added. A tutorial could be created that would help both mobile device and VR users understand the controls and goals of the games and the whole application in detail.

Keywords: VR, virtual reality, VR application, VR games, lying patients, bed bound, bedridden, bed confined

Supervisor: Ing. David Sedláček, Ph.D.

Abstrakt

Tato práce se zaměřuje na návrh aplikace virtuální reality (VR), která je vhodná pro zábavu a mírné cvičení pro ležící pacienty. Cílem této práce je nejprve prozkoumat možné cílové skupiny uživatelů a na základě analýzy navrhnout vhodné herní zážitky, jejich ovládání a mechaniky pro vybranou cílovou skupinu. Následně bude realizován prototyp navržené aplikace. Dále je třeba v souladu s metodikou UCD prototyp otestovat, navrhnout vhodné změny a tento proces několikrát iterovat. Výsledná aplikace a obsažené herní zážitky by měly být vhodné pro starší osoby, osoby s poraněním míchy nebo jinak nemocné (a tudíž ležící) pacienty.

Během celého procesu navrhování a realizace byl kladen velký důraz na omezení, která cílová skupina má. Mnohá z těchto omezení jsou fyzického rázu, například snížená citlivost rukou a prstů, slabý úchop rukou a celkově oslabená paže nebo nemožnost otáčet hlavou a tělem kvůli poloze vleže atd. Uživatelé musí být schopni aplikaci pohodlně ovládat bez ohledu na svá omezení.

Výsledná aplikace umožňuje hráči provádět rekalibraci, která mu umožňuje sebe a své okolí vnímat, jako by stál, a podporuje detekci gest namísto VR ovladačů. Přidružená mobilní aplikace umožňuje asistentovi pomáhat hráči VR s navigací v nabídkách, rekalibrací, nastavením vlastních gest a podobně. V aplikaci lze hrát tři hry – Tangramovou skládačku, pouťovou střelnici či nekonečnou skákačku. Ty jsou navrženy tak, aby zaujaly široké spektrum uživatelů s různými fyzickými omezeními.

Aplikace byla testována dvakrát v průběhu vývoje, měněna na základě výsledků těchto testování, a poté bylo provedeno závěrečné testování, které mělo za cíl ověřit, zda aplikace splňuje stanovené požadavky. Závěrečné testování ukázalo, že je aplikace použitelná a zábavná.

V budoucnu by mohla být aplikace ještě mírně pozměněna na základě výsledků závěrečného testování. Také by mohla být rozšířena o další herní zážitky, či úrovně do Tangramové skládačky či střelnice. Mohla by navíc být vytvořena výuková sekvence, která by mohla pomoci jak asistentovi používajícímu mobilní aplikaci tak uživateli ve VR porozumět více do hloubky ovládání a cílům her a celé aplikace.

Klíčová slova: VR, virtuální realita, VR aplikace, VR hry, ležící pacienti, upoutaní na lůžko

Překlad názvu: VR hry pro ležící pacienty

Contents

Figures

Tables

Chapter 1

Introduction

Despite the recent increase in the popularity of virtual reality, there are not many games people can play while lying down. It would be beneficial for older people, people who have a spinal cord injury or are bedbound due to a disease, to have a way of entertaining themselves and incorporating a little movement into their daily lives.

As a result, this thesis should help older and/or injured people by providing a VR application that is easy and intuitive to use and functions well in any position between sitting and lying. The resulting VR application should consist of multiple smaller games so that each person can choose the ones that they find the most engaging and fun to play. It should also motivate people to move their arms by providing interesting interaction with objects in the games while respecting the limitations that players may have in terms of mobility and strength. The VR application should also have a mobile device application associated with it, as it will allow other people (for example, the workers in facilities) to help the player inside VR to set up everything that is needed for them to comfortably play the games.

The motivation for this work is so that people can enjoy their free time when they have no option of going out or spending time with various activities like they used to do, whether bedridden forever or only for some time. The goal of this thesis is therefore to perform an analysis of the target users and other VR applications that have a similar goal as this one, then design the application, the gaming experiences in it, and the controls. Then a prototype is to be implemented and tested, and the application is to be reworked and tested as many times as will be needed for the users to be able to use it intuitively, easily, and without any frustration.

The first part of this thesis focuses on the theoretical background, the next chapter on analysis. The third chapter focuses on the design of the application, and the fourth chapter on implementation. The different iterations of testing are described in the fifth chapter, whereas the last part is focused on the resulting application and possible future improvements.

Chapter 2

Theoretical background

This chapter aims to define the terms, their properties, and principles by compiling information from the cited sources. All this information was necessary for us to know to be able to appropriately design and implement the application. After reading this chapter, the reader should be able to understand the terms used in the rest of the paper.

2.1 Virtual reality

The goal of virtual reality (VR) is to induce some desired behavior in response to the designed experience. The experience uses artificial sensory stimulation (usually targeting more than one sense) to make the person unaware [1] of the fact that the virtual world is not natural and to make the person feel a sense of presence in the virtual world. VR can thus provide many different experiences that can serve many purposes including video games and immersive cinema for entertainment, simulations to gain empathy for someone else's situation, and applications for education, virtual prototyping, and health care. This section is dedicated to mentioning the most important parts of the design process for VR.

The illusion of presence [2] in VR has four fundamental components - the stable place illusion, the illusion of self-embodiment, the illusion of physical interaction, and the illusion of social communication. Despite one may think the illusion will get better as the world and its characters get more realistic, it is better to avoid attempting to create a realistic character because of the Uncanny Valley. This effect of disgust instead of increased immersion takes place when the visuals approach reality but do not achieve it, as the characters come off as creepy to the users.

VR experiences can be very compelling. However, numerous potential issues may arise from not undertanding perception, users or interaction and design principles. The users might get frustrated or even sick. To avoid the possible side effects of using VR such as nausea, disorientation, headache, and eyestrain, it is crucial to avoid the visual-vestibular conflict [3], which occurs when the user's actual motion and motion perceived in a head-mounted display (HMD) differ (see Section 2.1.1). In addition, when designing a VR experience, the designers have to take into account how humans visually perceive [1, 2] the world. For instance, incorrect depth perception cues might lead to the user assuming the scale of objects incorrectly or the perception of motion might be lost when the frame rate is too low.

Audio is another powerful component of VR because it supports and enhances the visual input from the experience and can increase the player's sense of presence. Additionally, it can help the users feel particular emotions [2] or provide another layer of information (given that people can estimate the location of a sound source just by hearing it) and cue their visual attention or make them more aware of their surroundings.

2.1.1 Visual-vestibular conflict

Conflicting inputs from visual and vestibular[3, 4] receptors — responsible for sensing the user's perceived motion and actual motion, respectively, can result in motion sickness (also called cyber sickness [5]), nausea, disorientation, or instability. It is important to keep visual-vestibular conflict in mind while designing and implementing VR applications and to actively try to avoid all the side effects that would eventually frustrate users, make them sick, and discourage them from ever using the application again.

Various properties can cause visual-vestibular conflict such as a wide field of view, the latency of the system, incorrect calibration or tracking, or low frame rate. Furthermore, people can get motion sickness from vection – experiencing motion in VR while being stationary in the real world, most likely from angular velocity or linear acceleration. Models for prediction of motion sickness [6] exist that can help to avoid making users sick.

Motion sickness may be reduced when the user is supplied with some fixed points [2] that could be identified as a so-called rest frame. Having a fixed rest frame should help the user maintain eye stability and thus reduce the amount of unnatural eye movement (which causes motion sickness) that is required to keep the scene's projection on the retina stable. Ratcheting – instantaneous discrete turns with a fixed angle – can also help reduce sickness. The sickness of users can be measured using the Kennedy Simulator Sickness Questionnaire (SSQ), which has become a standard in this area.

2.1.2 Interaction

When it comes to interaction, it is vital to design the interaction mechanisms in a way that they are intuitive, easy to learn, do not require too much attention from the user or cause them discomfort, and the tasks for which they are designed can be performed effectively. Applications need to be discoverable and allow people to explore how they work. This is especially important in VR applications, where the user cannot simply read a manual while using them.

Users should be able to discover [2] how the system works thanks to correctly implemented affordances, signifiers and constraints, feedback and understandable mappings. The affordances should clearly define possible actions and define means of interaction that users can easily perform. Signifiers should clearly communicate the purpose and behavior of the affordance, as well as its constraints, even before the user begins an interaction. Feedback communicating the results of user's actions is essential, but should be used carefully as too much of it can overwhelm the user. The mapping is a relationship between the control and its results. For instance, writing by hand on paper can be remapped to writing on a keyboard when using a computer. The motions required to control the VR application should not cause user physical fatigue such as Gorilla arm fatigue, which results from having to hold one's hands up for longer periods.

Interaction can be either direct, indirect, or semi-direct. When directly interacting with an object, it seems to the person as if the object is an extension of their body. Indirect interaction requires the person to convert between input and output and thus requires more cognition. As an example, searching on the Internet requires the user to think of a query, type it, and after the search is complete, interpret the result.

It is intuitive for people to interact with the object in the world using both hands, and the same applies in VR [2], but it is important to design the interaction appropriately and not to force two-handed interaction where it does not belong. The two-handed interaction can be symmetrical, where both hands perform identical actions, or asymmetric, which is more common in the real world. The interactions requiring both hands should let the hands work together fluidly while switching between symmetric and asymmetric interaction depending on the task.

The interactions can have a different level of fidelity [7]. This means they have a different degree of correspondence between the physical actions used to perform actions in the real world and those used to perform actions in the virtual world. For realistic interactions the users do not have to learn much, because they already know how to perform those actions from the real-world.

Non-realistic interactions might on the other hand increase performance, increase enjoyment and cause less fatigue. Magical interactions that lay somewhere in the middle of the spectrum require users to perform natural movements, but the performed actions are enhanced and more powerful, such as grabbing an object from a distance.

Sensory Substitution

Most VR experiences are not able to provide haptic feedback [2] such as stopping the user's hand when it comes to contact with an object in the virtual world. To replace the unavailable sensory cue, there are different ways of performing sensory substitution. With ghosting, an object such as a hand can be rendered twice, once to represent the user's physical hand, the second to represent virtual physically simulated hand that may have collided with an object in the scene and thus stopped moving. Highlighting by changing the visuals of the object is used to represent collision with an object and signifies that a person can interact with it. This information can also be represented using an audio cue or a rumble (a form of haptic feedback where the input device vibrates).

2.2 Upright redirection

The main problem with playing games while lying down [8] is that all the player can see is a ceiling or the sky. As this thesis aims to make games playable for people who are bedbound, a possible solution is to change the angle at which the player views the world. We can perform so-called upright redirection – rotate the player in the virtual world – to help them perceive the interactions and environment as if they were standing (see Figure 2.1). This section will briefly go through the results of some studies that monitored how the player perceives such change.

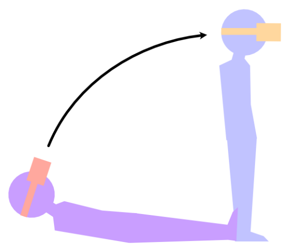


Figure 2.1: Upright redirection: the process of transforming a lying user in the virtual world so that they feel like standing.

In the first study [9] on the effect of reclining angle on the perception of horizontal plane for HMD users four degrees of recline were studied (between upright position at 0° and laying at 90°). Participants wearing a head-mounted display (HMD) were asked to locate the horizontal plane. The results of this study show that the subjectively perceived horizontal plane is closely related to the recline of the upper body.

The second study [10] on exploring the effect of sensory conflict due to upright redirection while using VR in reclining & lying positions tested different recline angles (5 angles between 0 ° and 90 °) and the perception of the user of their own body and movement. The results show that most people are well aware that their body is not in a standing position even though their virtual body is. The majority found the 45° reclining angle the most uncomfortable, and more than half of the participants found the lying position (90° recline) the most comfortable among the reclined ones.

There are many locomotion methods for VR, but some of them are more suitable for when VR is being used lying down. A recent study [11] on exploring locomotion methods with upright redirected views for VR users in reclining & lying positions describes the challenges of locomotion in reclining position and proposes some locomotion methods that are suitable. It describes gesture interaction methods where arms, head, or legs are used to perform gestures that serve as metaphors for locomotion. The results of this study show that most people prefer to tap their feet to walk and rotate on office chairs to rotate in virtual reality. These locomotion techniques are also strongly impacted by the reclining angle and, interestingly, the results from the above study repeat, as the 45 ° angle was considered the worst angle and 90 ° the most comfortable, although it is the most different from the standing position. Many people also reported that the chosen locomotion techniques were easier and less fatiguing to perform in a laying position than when standing upright.

2.3 User-centered design

User-centered design (UCD) [12] is a methodology used by developers and designers to ensure they're creating products that meet the needs of users.

"Design [13] should:

- *Make it easy to determine what actions are possible at any moment (make use of constraints).*
- *Make things visible, including the conceptual model of the system, the alternative actions, and the results of actions.*

- *Make it easy to evaluate the current state of the system.*

- *Follow natural mappings between intentions and the required actions; between actions and the resulting effect; and between the information that is visible and the interpretation of the system state."*

The design process of UCD is iterative and has four phases [14]. In the first phase, designers try to understand the needs of users and analyze the context of use of the product. The user requirements can then be specified, followed by the creation of a design solution. Subsequently, the design is evaluated and tested. These phases are iterated (see Figure 2.2) until the results satisfy the requirements.

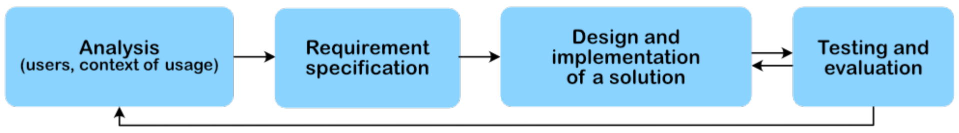

Analysis
(users, context of usage)
Requirement specification
Design and implementation of a solution
Testing and evaluation

Figure 2.2: UCD process diagram

Usability testing [15, 16] is performed to improve usability by involving real users in the testing, giving them tasks to accomplish, and then allowing the testers to observe the actions of the participants. The testers are then able to analyze the data and propose changes according to the results of the tesing. Ideally, the number of users for qualitative testing should be 5 according to Figure 2.3 that shows a graph of the ratio [17] of usability problems found to the number of test users studied.

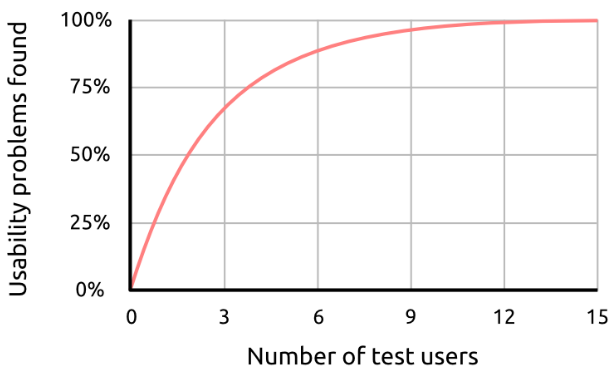


Figure 2.3: Ratio of usability problems found to the number of test users.

2.4 Hand gesture recognition

Hand gestures [18] are part of body language. They can be discerned by looking at the shape formed by the hand and its fingers. There are static hand gestures that describe a stable shape of the hand (like the thumbs-up gesture), and dynamic ones that are created by performing a sequence of hand movements (for example waving).

Gesture recognition [2] is a process of detecting specific gestures that can be interpreted as commands. The bare-hand input devices work by detecting hands by sensors and display them in the virtual world using a skeletal model and are ideal to minimize fatigue in arms, which are thus not encumbered by additional devices such as hand-held controllers. However, this form of input also has its disadvantages, such as not having a sense of touch or problems with the consistency of gesture recognition for a wide range of users.

Vision-based [19] gesture recognition is a type of bare hands gesture recognition in which the hands are recorded using cameras. There are two main cathegories of approaches to vision-based gesture recognition. The 3D hand model-based approach uses a 3D hand model and tries to estimate the parameters of the model by comparing its projection to 2D to a 2D input image. Appearance-based approaches use image features extracted from the input and model the visual appearance of a hand using the image features.

2.5 Networking

The resulting VR application should have a mobile device application associated with it to help the user in VR with whatever they might need help with and for the assistant to be able to see what the user in VR is doing and how. To have the two devices connected, client-server communication needs to be established. Because server communication is not the main goal of this thesis, an existing library is chosen to help with the networking. Some of the most popular networking solutions that exist for Unity are listed below:

- **Mirror[1]:** Mirror is a free open-source library made for Unity networking. It provides high-level functionalities like spawning objects on the server, remote procedure calls (RPCs) from either client-side (Commands) or server-side (ClienRpc), synchronization of variables and GameObjects, and more. There are also many library extensions for Mirror.

[1]Mirror: github.com/MirrorNetworking/Mirror#made-with-mirror

- **Netcode for GameObjects[2]:** Unity has a web page dedicated to networking in their game engine and Netcode for GameObjects (NGO) is the first library proposed there. NGO is a high-level networking library, that uses client and server remote procedure calls to provide client-server and server-client synchronization, and enables users to send GameObjects and other data across the network.

- **Unity Transport[3]:** Unity also provides a library for networking called Unity Transport. It is a low-level library, that provides a connection-based abstraction layer over UDP sockets. It is used to manage connections to servers and send and receive data and doesn't provide any higher-level functionalities like automatic synchronization of objects.

A high-level library is preferred as it provides more functionalities. Because there seemed to be no great differences between Mirror and NGO, Mirror was chosen based on having some previous experience with it.

[2]NGO: docs-multiplayer.unity3d.com/netcode/current/about/
[3]Unity Transport: docs-multiplayer.unity3d.com/transport/current/about/

Chapter 3

Analysis

The goal of this chapter is an analysis of the target users and other VR applications with a similar goal. It is important to understand the target user group, their motivations, limitations, and frustrations to be able to design an application that will help users reach their goals without problems. Analyzing different VR applications that also focus on enabling bedbound or otherwise disabled people, will help with the design of controls and gaming experiences for the target users, whether the resulting application will be inspired by the choices, or decide they are not useful or well executed.

3.1 Finding target users

The target user group of the resulting application is quite large and diverse. People can be bedbound due to a permanent or temporary injury/illness. Many factors, including being elderly, can cause that. Injuries and illnesses can have different impacts on the mobility or mental ability of a person. It was important to visit different institutions and gather information about what their clients enjoy and what they can and cannot do because of their limitations, with the main goal being to find out how the application will be controlled and what gaming experiences it will contain. To not violate anyone's privacy, no names of people will be mentioned in the thesis.

3.1.1 Diaconia Lifetool

The first institution visited was Diaconia Lifetool[1], which provides long-term care to the elderly. The diaconia employees talked about how they use VR

[1]Webpage of Diakonia Lifetool: lifetool.diakonie.cz

to help their clients revisit memories and thus make communication with their families easier and more meaningful. They do this by presenting a set of photos from their life in a VR gallery, and they expect these to invoke questions from family members and joyful memories of the clients.

The diaconia employees said that rehabilitation and improvement in movement or mental abilities are not their goal. They want to make the last years of the lives of their clients as pleasant as possible. Instead of complex interaction with the environment and gamification of tasks, the employees thought a calm, familiar environment with an emphasis put on impacting different senses would be a better fit for the clients. As an example of the scene that could be created, the employees suggested an animated realistic Czech summer forest, filled with different kinds of local plants, animals, and trees. Their goal remains the same as with the other VR experiences – to make clients remember some sceneries and stories and communicate those to others. According to them, the elderly they care for do not like to interact with artificially-looking environments or to play games.

3.1.2 Home for the elderly Nová Slunečnice

At the Nová Slunečnice home for the elderly[2], they already use VR applications Beat Saber, Wander, and Kaleido. Beat Saber is a game full of quick hand movements and thus can be played only by the most lively clients. In Wander, clients can visit different places of the world while seated. Kaleido is an app developed specifically for the elderly and provides an experience similar to Wander with a focus on cognitive abilities, but the employee we spoke with told us that the videos in the application are blurry and that their clients frequently experience motion sickness as a result of using it. In addition to the fact that none of the applications are designed to be used while lying down, the sounds in the applications are too quiet for the elderly to hear, and in some cases, it is harder for them to experience things in the intended way because of dementia or a similar illness.

According to the employee, the elderly in the home have problems using VR controllers, because their fingers can be numb or they do not have enough strength overall to hold them and move their hands for longer periods. The clients are also expected to be slow in performing tasks. Some of the favorite topics of the elderly are cooking, animals and nature in general, Czech environments. Men also like cars, airplanes, and fishing. The main purpose of the application should be entertainment.

[2]Webpage of the Home for the elderly Nová Slunečnice: novaslunecnice.cz

3.1.3 Center Paraple

The last institution visited was the Center Paraple[3]. They already use VR to motivate patients and improve their motor skills and physical condition. They use applications like Beat Saber, Rocket Tennis, and HOLOFIT - there are multiple sports suitable for the clients like handbike riding, ski running, and rowing. While a client uses the VR headset, the employees watch a stream of their activity on a device like a tablet or a TV.

The employee mentioned that the people in the center have problems with grabbing things, so they cannot hold the VR controllers on their own. To be able to use the applications, they usually strap the VR controllers to clients' hands. Also, not all people in the center have a mobile chest or waist, so they have to be stabilized in some way in a wheelchair while using the VR applications because they need to be seated. The mobility in the arms of the patients is usually good, and they can also move their head. The majority of people in the center are younger men (not elderly); therefore, their favorite themes are cars, sports, and competing.

3.2 Existing applications

To be able to design the application, it is important to see how similarly focused games are designed and to learn from their successes and mistakes. Many VR games can be played while sitting, for example, Half-Life: Alyx, I Expect You To Die, The Room VR, etc. Unfortunately, playing the games sitting down is not always possible for older or injured people, who can be bedbound. When trying to play VR games while lying down, people can encounter many problems including not seeing the important things (usually players can see just the sky or the ceiling of a room), not being able to interact with the game correctly or not being able to reach objects because of their positioning.

There are not many applications that focus on people who are neither standing nor sitting. For a short while, Oculus released a Laying Down Mode, that allows one to play games and generally control the VR while lying down, but they have taken the feature down according to a thread [20] on Meta Community Forum.

There is information on the internet forums [21] about people trying to find games to play while lying down, and some of the responders mention a

[3]Webpage of the Center Paraple: parable.cz

game called *Demeo*[4]. It is a co-op tabletop VR game where the developers have implemented a complex way of navigating across the map (see Figure 3.1 and 3.2) using VR controllers. The player can even tilt the whole game. This is why it does not matter if the player is lying down while playing[5]. In one of the recent updates players also got an option to play the game without controllers.

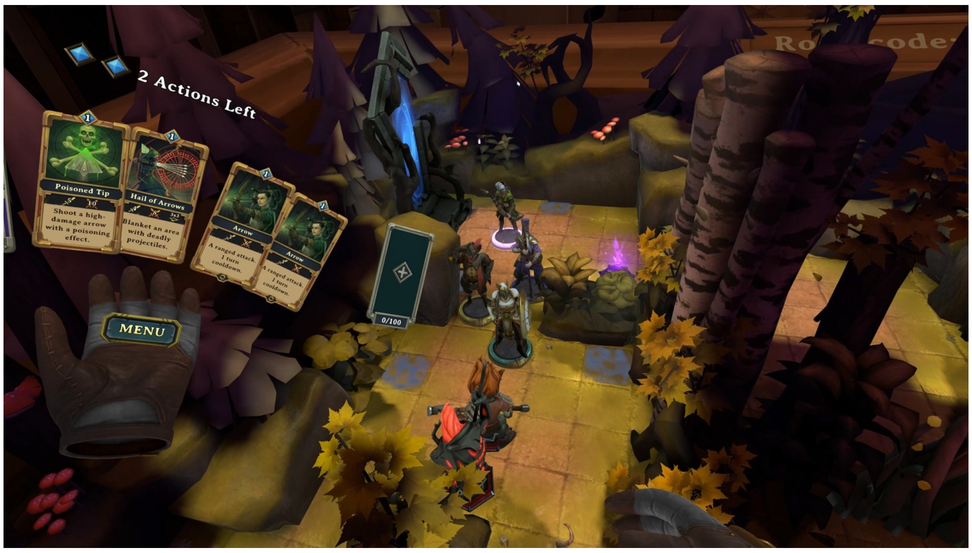


Figure 3.1: Demeo: The player is zoomed in to see a particular location of the board.

[4]Demeo page: www.resolutiongames.com/demeo
[5]Demeo review:
mixed-news.com/en/demeo-review-vr-co-op-role-playing-game/#A_VR_game_with_ambition

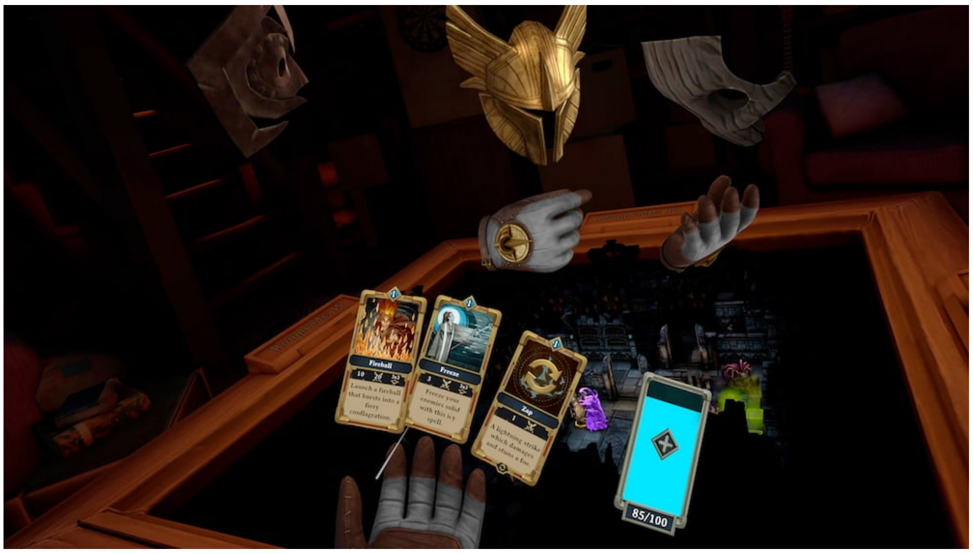


Figure 3.2: Demeo: The player is zoomed out to see the whole board.

Chapter 4

Design

This chapter describes the design of the application controls, the design of the gaming experiences, and proposes the data to be collected by the VR logger application [22] by Leoš Řeháček.

4.1 Controls

This section explains the design of controls of the application with a focus on usability. The design of this application strongly relies on the information gained during the analysis part of this thesis, mainly the information concerning the abilities, limitations, and motivations of the target users.

4.1.1 Bare hands control

For the target user group to be able to interact with the environment in virtual reality, it was decided to use gesture detection instead of VR controllers. This decision was made because many people from the target group have numb fingers and would not feel the buttons they should be using, or they do not have enough strength in their arms to lift the VR controllers' weight. The decision to use gesture detection brings up other questions that have to be answered such as: what gestures are the target users able to perform or whether or not these gestures are intuitive. There is no way of finding the answers to these questions beforehand; this is why regular testing of this application is crucial to the process of development.

Some gestures must be chosen for the first version of the application, although they may not be perfect. With the assumption that it should be easier for the users to control the whole set of fingers on the hand as opposed to individual ones, the first design has two gestures implemented, rock and paper (closed fist and open one with straightened fingers; see Figure 4.1). To avoid confusion of the users when told to make the gesture, two versions of each gesture were added (hopefully covering most of the interpretations) and it was decided that the gesture detection would have a greater tolerance regarding the difference between the current gesture and the target one. The implementation details of gesture detection can be found in Section 5.2.

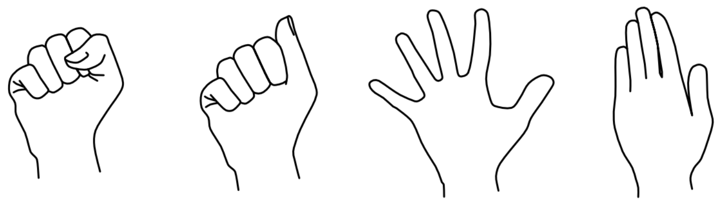


Figure 4.1: First two gestures (rock, paper) and their two variations

4.1.2 Mobile device application

The VR application should have a mobile device application linked to it for various reasons. Although it is not known how many functionalities the mobile device application should have and all of its possible usages will be discovered later in the development process, some are known at the beginning of the development process.

The first reason to have the mobile device application is for the employee of the institution who provides VR experiences to the clients to be able to see what the client does in the VR application. Multiple solutions for this kind of streaming already exist, but they do not provide any control over the VR application for the person watching the stream. The mobile device application needs to give the assistant (employee) control over the VR application, because the person in VR may not be able to perform all of the actions needed for starting a game experience or setting up the environment (for example recalibration for upright redirection) either because of physical constraints or simply not being familiar with technologies. In this way, the person in VR can simply focus on the game they chose and play it because the assistant can help them with anything else they might need.

4.2 Gaming experiences

The application must include various gaming experiences (at least two or three) so that each person can choose a game that suits them. Some of the games should focus more on the elderly, some should be designed for younger players, some should contain action, and some should be relaxing. It is also important to create experiences that are targeted at both men and women. This section will describe the possibilities in designing activities that will be implemented later on, their characteristics, possible controls, advantages, and disadvantages. The first three gaming experiences described are the ones that were chosen to be implemented, and in the section following them the activities that were not chosen in the end but were considered are summarized.

4.2.1 Tangram

A Tangram is a rearrangement puzzle that consists of seven shapes: two large triangles, one medium triangle, two small triangles, a square, and a parallelogram. The player is given the shapes and an outline (or silhouette), and the goal of the game is to place the shapes inside of the outline so that they fill it without overlapping or laying outside of it.

Because Tangram is a puzzle game without any pressure to solve it quickly, it is a relaxing game. The game has no particular theme that would make it considerably more suited for male or female audiences, or younger or older audiences, so it is very versatile in terms of its target audience.

Because there are more than 6000 shapes that can be composed of the seven pieces and each of them has only one correct solution, it would be beneficial if the Tangram game had many different levels (meaning shapes) the person could play. This is why the first thing a player would encounter after selecting the Tangram game is a menu with a wide range of different puzzles. The player would then place the shapes in the outline and when they place everything correctly and finish the level, they could choose another level.

The process of solving one puzzle could work as follows. There is a board in front of the player and a silhouette is displayed on it. There are 7 Tangram pieces randomly placed and rotated on the board, outside of the silhouette (for an example, see Figure 4.2). The player grabs a piece (rock gesture), then moves their hand where they want to place the piece, rotates the hand (still holding the piece, so the piece rotates, too), and then lets go of the piece (releases the hand so the rock gesture is not detected anymore). To aid the player with the controls, the rotation could 'snap' to multiples of some given value (for example, of 45°), the pieces could be grabbed from a distance (the hand could cast a ray on the objects) and put down from distance, too - it would be helpful to display a duplicate (ghost) of the piece on the board so the player knows where they are placing the piece.

There are two ways in which the evaluation of the completeness of the puzzle could work. The first would immediately snap the piece in its place when it is placed correctly. This approach can be easily abused and is therefore not ideal from the game design point of view because the player can try a few placements and rotations randomly and know that they succeeded in the guessing immediately. The second approach would be to snap all the pieces in their place only after the whole puzzle is correctly solved. The disadvantage of this approach is that players might try to accurately place the pieces and eventually get frustrated. Testing of the game is needed to show which of these approaches is better.

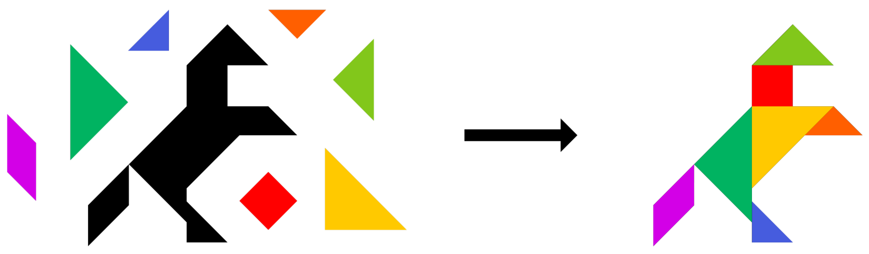


Figure 4.2: Example of the Tangram puzzle in the shape of a horse at the beginning of the level (left) and after its completion (right).

4.2.2 Shooting gallery

Shooting from an air rifle is known to many Czechs - they are either taught by their grandparents or parents, or they have tried shooting roses in a Shooting Gallery. This activity can be considered more for the male audience and could be relaxing or include a lot of action due to limited time and score-based gameplay.

The game will be set in a carnival Shooting Gallery with many targets to shoot at, some of which may be large, some small, some static, and some moving. There are several ways in which the shooting could work. An air rifle is usually held with both hands, so it might limit some people who may have problems with one of their hands. The best solution for this game experience might be to have more guns to choose from, for example, a pistol (one-handed) and a rifle (two-handed). The guns should also have some aiming visualization – a ray cast from the gun's barrel might suffice in indicating where the gun will shoot. The shooting could be either automatic with a given period in between individual shots (like in Rogue Ascent VR [1]), or manually controlled

[1] www.meta.com/experiences/rogue-ascent-vr/5395586720470296/

by the player using gestures. As it is interesting to see the player's accuracy in shooting, gesture-controlled shooting will be implemented.

The game will contain two game modes: one relaxing, where the goal is to eventually shoot all of the targets or reach a given score, and the quick action mode, where the player gets a given amount of score points for each target shot and the goal is to reach the highest score possible in a limited time.

4.2.3 Endless runner

The goal of the Endless Runner game is for the player to reach the highest score possible. To do that they would have to perform gestures to avoid hitting obstacles with the game's character. The character would be able to run, jump, and crouch and hand gestures would control these actions. The player would get score points for the distance the character ran and there could also be bonus points that could be picked up while avoiding obstacles so that the player is positively motivated to perform the gesture.

It is important to provide a fixed rest frame so the users do not perceive themselves as moving which may cause motion sickness. Thus the Endless Runner will have a room containing a belt on which the character runs and obstacles are placed on. The character should run in the opposite direction of the belt at the same speed so that it stays in one place as a result, because the laying patients may not be able to turn their heads to see the character if it was elsewhere.

4.2.4 Other possible gaming experiences

This section summarizes the gaming experiences that will not be implemented and describes their benefits and limitations to show why the activities above were chosen instead.

Many activities that were considered would require including lots of content in the game because the player has to perform repetitive tasks and thus the game would be unable to keep them engaged and interested for longer periods. For example, fishing games usually provide large amounts of content: many different kinds of fish can be found in diverse environments, like in Real VR Fishing [2]. Similarly, a cooking or a rhythmic hand-gesture game would require many recipes (and ingredients) or music pieces, respectively. As the goal of this application is to contain more than one game, it may be impossible to provide enough variety in this type of games without negatively

[2]Real VR Fishing: store.steampowered.com/app/1266470/Real_VR_Fishing/

impacting the other game experiences. This is why even though the themes are popular among the target users, these games will not be included in the application.

As sports were also mentioned among popular topics among the target users, many of them were considered, including archery. Archery would have to be changed greatly in comparison to reality because of the lying position and the inability to move arms behind one's body. Also, there was already an option to choose another shooting game (with a rifle) which is less limited in terms of movement. Hand cycling, cross-country skiing, or rowing are all sport activities already performed at Center Paraple as the application Holofit[3] they purchased contains them. Besides being already implemented, they usually require moving the user in the virtual world (so they do not get bored), which might cause motion sickness to the users, which is not suitable. In conclusion, there exist many sports that could be created in VR, but most of them are already implemented for VR or they are not suitable for being played lying down.

4.3 Storing activity data

One of the thesis requirements was to log and store activity data from the VR application using the application developed by Leoš Řeháček [22]. This section describes the data that would be interesting to store for future analysis. The application can log the data in three different ways – as events or custom record data, where both belong to one record with a specific timestamp, or as custom data of the whole activity. It also allows recording the position and rotation of the player's head and hands. The Table 4.1 shows description of the data, and the planned way of sending the data to the server that stores it.

The activity data is sent each time a scene changes or a game is reset, so that the information about activity/gameplay length can be read from the automatically saved value. Unfortunately, it is impossible not to display some of the custom (activity) data depending on the game because the data are stored in a JSON format common for the whole application. The implementation details of the logging can be found in Chapter 5.6.

[3]Holofit: www.holodia.com/

Data description	Way of logging
head rotation	automatic with each record
left-/right-hand position	automatic with each record
number of upright redirections performed	custom data
upright redirection with information about the rotation angle with respect to the previous rotation	event
Tangram: piece grabbed	event
Tangram: piece dropped	event
Tangram: level completed	event
Tangram: level name	custom data
Tangram: level completion time	event
Tangram: level fastest completion time	custom data
* hit an obstacle in Endless Runner tutorial section	event
Endless Runner: score	custom data
Endless Runner: number of bonus points collected	custom data
Endless Runner: bonus point collection accuracy (collected/spawned)	custom data
* Endless Runner mode (with/without obstacles)	custom data
Endless Runner: jump begin / jump end	event
Endless Runner: crouch begin / crouch end	event
Endless Runner: obstacle hit	event
Shooting Gallery: score	custom data
Shooting Gallery: time	custom data
Shooting Gallery: best time / highscore	custom data
* Shooting Gallery mode (slow/normal/fast timed/untimed)	custom data
Shooting Gallery: accuracy (shots fired / targets shot)	custom data
Shooting Gallery: average score per target	custom data
Shooting Gallery: gun type (rifle / rifle on stand)	custom data
Shooting Gallery: target shot with information about how many points it was worth	event

Table 4.1: Table describing data that would be interesting to log together with the planned way of logging it. Data marked with a star '*' were added to this list after some feature was added to the application after testing.

Chapter 5

Implementation

This chapter describes the individual parts of the applications that were implemented. Because the application needs to be tested frequently and the design and implementation might change, each section will mention any modifications that were not in the prototype together with referencing the testing that caused the changes. The application is implemented in Unity version 2022.3.10f1.

5.1 Upright redirection

The upright redirection can be invoked either by the player in VR by performing a given gesture (the default setting for it is both hands doing the paper gesture) or by the mobile device user by pressing a button.

There is a special object in each of the game scenes that saves the target rotation and position of the player for the particular scene. The first thing computed when performing upright redirection is the rotation, which is done by taking the current transform of the VR player camera and subtracting that from the target rotation. We get the difference in rotation between these two objects and can rotate the VR origin around the camera position using these angles. Then we do the same thing with the position of the two. It is crucial to have the operations in this order because the rotation affects the position of the player's head, and this must be corrected by the position computation. See Figure 5.1 for visual explanation of upright redirection. Upright redirection implementation might look as follows:

```
rotDiff = targetTransform.rotation - camera.rotation;
originVR.RotateAroundCameraPosition(camera.right,   rotDiff.x);
originVR.RotateAroundCameraPosition(camera.up,      rotDiff.y);
```

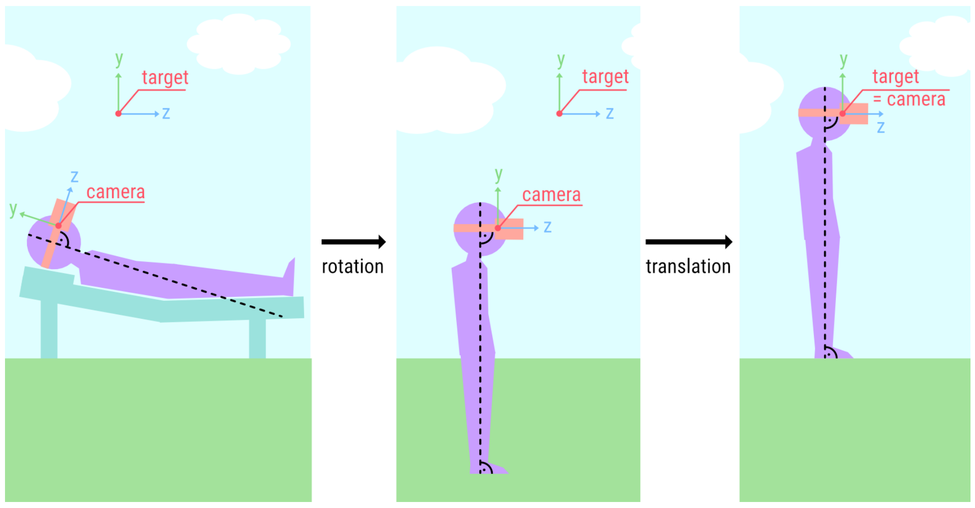


Figure 5.1: Upright redirection implementation visualization – shows the process of rotating the player so they see the virtual world as if standing upright, and moving them to a predefined ideal place in the scene.

```
originVR.RotateAroundCameraPosition(camera.forward, rotDiff.z);

posDiff = targetTransform.position - camera.position;
originVR.position += posDiff;
```

The target transform position might not be ideal for everyone - the player might, for example, want to be closer or further to the interactable objects in the game. Thus, another functionality was added to the mobile device version of the application that simply moves the player using an offset from the target transform. This offset is then saved, so there is no need for repeated recalibration after changing scenes.

5.2 Gesture detection

The gesture detection is performed with each call of the `Update` function. The default gestures mentioned in the design chapter are stored in a list inside a scriptable object (SO). Each gesture has a list of bone transforms relative to the root of the object, a gesture type (for example a rock), and information about whether it is on the left or right hand. The detection has a threshold and computes the difference between the current bone transforms on the hand and the ones saved in the gesture. If the difference is lower than the threshold the gesture is detected.

Following this paragraph is a pseudocode for the function that is used to recognize the gestures on both hands.

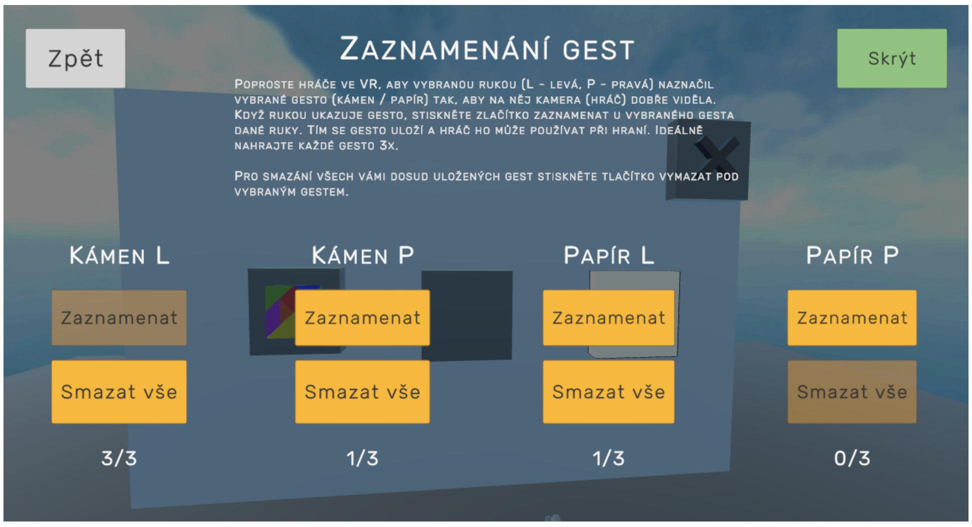
```
RecognizeGesture(currentGesture) {
    differenceSumMin = thresholdGesture;
    currentGesture = null;

    foreach (gesture in existingGestures) {
        differenceSum = 0;
        notRecognized = false;
        hand = skeleton;

        root = hand.Bones[0].transform;
        for (int i = 1; i < hand.Bones.Count; i++) {
            thisBone = hand.Bones[i].transform.position;

            //bone transform relative to the root of the hand
            currTransform = root.InverseTransform(thisBone);

            difference = Vector3.Distance(
                currTransform,
                gesture.transforms[i]
            );

            if (difference > thresholdBone) {
                //one bone is too different
                notRecognized = true;
                break;
            }
            differenceSum += difference;
        }

        if (!notRecognized) {
            if (differenceSum < differenceSumMin) {
                //hand gesture is recognized and saved
                differenceSumMin = differenceSum;
                currentGesture = gesture;
            }
        }
    }
}
```

The code above contains the following variables:

- *thresholdBone*: maximum difference in transform for one bone (compared to the existing recognized gesture's bone)

- *thresholdGesture*: maximum difference in transforms for the whole hand skeleton
- *existingGestures*: list of all saved gestures which contain a list of transforms of bones of a hand
- *skeleton*: contains a list of transforms of bones of left/right hand
- *currentGesture*: output parameters saving the current gesture on left/right hand, may be null

After the prototype testing, a need for player-added gestures was noticed. To read more about the prototype testing and its results, see Chapter 6.1. As a result, the ability to add and remove custom gestures using the mobile device menu in cooperation with the VR player was added. The player in VR needs to perform the selected gesture and while they are performing it, the assistant saves it by using respective buttons. You can see the section of the application for managing custom gestures in Figure 5.2.


Zaznamenání gest

Poproste hráče ve VR, aby vybranou rukou (L - levá, P - pravá) naznačil vybrané gesto (kámen / papír) tak, aby na něj kamera (hráč) dobře viděla. Když rukou ukazuje gesto, stiskněte zlačítko zaznamenat u vybraného gesta dané ruky. Tím se gesto uloží a hráč ho může používat při hraní. Ideálně nahrajte každé gesto 3x.

Pro smazání všech vámi dosud uložených gest stiskněte tlačítko vymazat pod vybraným gestem.

Figure 5.2: Menu for managing gestures - sorted by type of gesture and left or right hand.

While the gesture is detected and some action is supposed to happen after some duration of the gesture being performed, there is audio and visual feedback implemented. Both of the feedback actions are performed at the same time to ensure the player knows something is happening. You can see how the visual feedback (of gradually filling the hands with color) works on custom-set gestures in Figure 5.3. The hand detection in Meta Quest devices may sometimes not work ideally and the gestures are thus not detected properly. This happens especially when there is excessive light (for example when the person lying down has a ceiling light on) or on the other hand not enough of it. When the player performs a gesture that is supposed to be held for a while to perform some action, it causes the progress to sometimes reset, even though that was not the intention of the player and they do not change the gesture they are performing. Thus a time period was implemented where if the gesture is the correct one after the period passes, the gesture stays detected even though the gesture was not detected in a few frames.

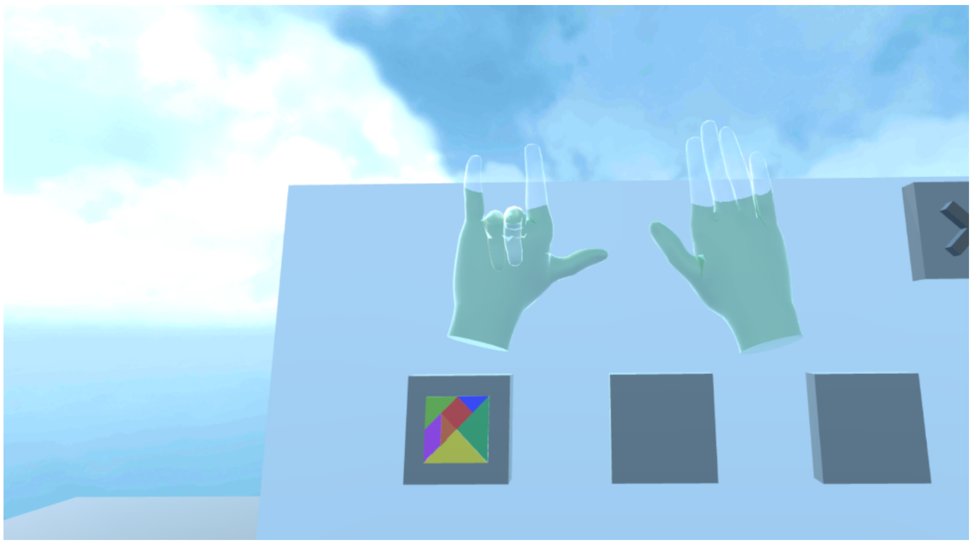


Figure 5.3: Visual feedback given on custom gesture for upright redirection.

5.3 Gaming experiences

This section is dedicated to explaining how the mechanics of different gaming experiences are implemented.

However, prior to mentioning specific implementation of each game, a common feature will be mentioned. After the second testing, short audio instructions were added to the games, that are played to the user after they choose a game.

5.3.1 Tangram

The Tangram implementation is divided into Tangram pieces, Tangram slots (placements where the pieces belong), and a correct placement resolver.

There are two ways of grabbing the Tangram piece - direct and from a distance, both are based on the implementation of interaction between `Interactor` and `Interactable` components (for detailed explanation see

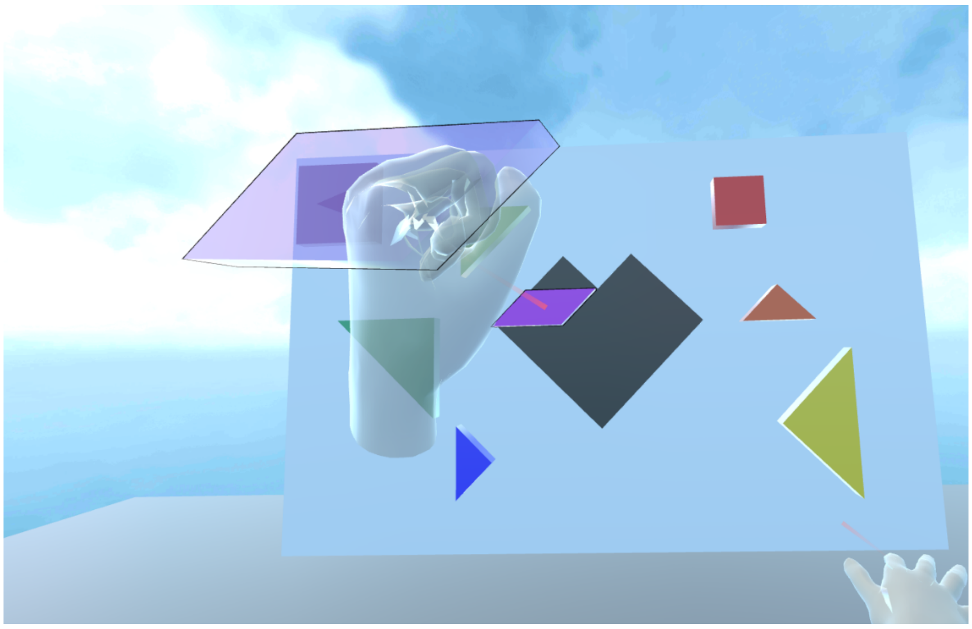


Figure 5.4: Holding the piece far from the board will result in a ghost piece appearing at the end of the casted ray.

Chapter 5.4). The pieces can be grabbed using a rock gesture and put down by releasing a rock gesture in both of the situations. When grabbing the piece directly, the user should position their hand on the piece and grab it. When the user places his hands far from the board, the hands start casting rays back onto the board. To grab a piece from a distance, the player should position their hand so that the ray points to a piece (the piece will highlight and show an outline) and then grab it. If the hand is far from the board when holding the piece, a 'ghost' piece will appear having the position and rotation of the piece as it would have if it was dropped at the moment (see Figure 5.4). The ghost piece is simply positioned as the grabbed piece and snapped onto the board by changing its Z-coordinate, and its rotation is similar to the grabbed piece but displayed already snapped to the multiples of 45 degrees. The held piece is made partially transparent so that the player can always see the board even when having their hand right in front of their face. The ghost pieces are created and updated in the `Update` function of the `TangramPiece` class.

The prototype testing (read Chapter 6.1 for more details) showed that the VR player needs to pick up, rotate, and drop the piece multiple times to be able to rotate it to the desired angle very different from the starting one. Multiple ways of rotation mapping were studied but none of them seemed to suit this particular case. This is why it was decided that the rotation of the piece would be simply computed as the hand rotation multiplied by a value greater than one. This mapping of the rotation should help reach

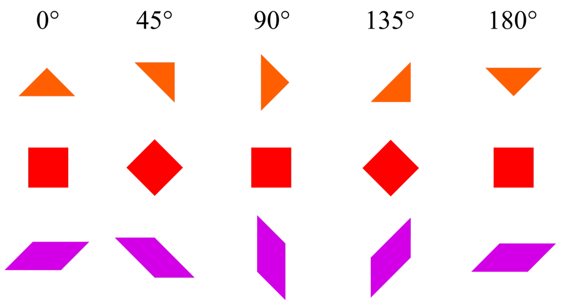


Figure 5.5: Symmetry angle of Tangram pieces is an angle that produces the same shape after being used for rotation. For a square, the symmetry angle is 90°, for a parallelogram it is 180°, and for triangles 360°.

the desired rotation only by grabbing and rotating the piece once while remaining intuitive. The rotations are performed in the `Update` function and implemented as follows:

```
rot = (rot.x, rot.y, (rot.z - rotPrev) * multiplier + rotPrev);
rotPrev = rot.z;
```

Each time a piece is dropped back onto the board, the correct placement resolver begins to work. Each Tangram slot has an assigned piece type (large triangle, rectangle, etc.) that belongs in it. Each type of piece has its angle of symmetry - when the player rotates the piece and the rotation angle is equal to the angle of symmetry, the piece will visually look the same as it did before the rotation was performed (see Figure 5.5). The placement is correct when the type of the piece corresponds to the type in the slot, the distance of the piece to the slot is lower than a certain threshold and the rotation of the piece is the same (with symmetry in mind).

When all the pieces are correctly placed, they snap into the exact accurate positions and a special effect and audio recording are played to give feedback to the player. The movement of the pieces into their accurate positions is animated.

In the final version of the application that was created after the second testing, there is also an option that allows the user to snap each individual piece in its place after it is placed correctly and disables the interactions with it to help the user focus more on the puzzle and less on the accuracy of the placement of pieces and to also stop them from grabbing the already correctly placed pieces, which used to be an issue. The snapping can be turned on and off, because some users might like the challenge of not knowing if they have placed the pieces correctly or not until they finish the puzzle.

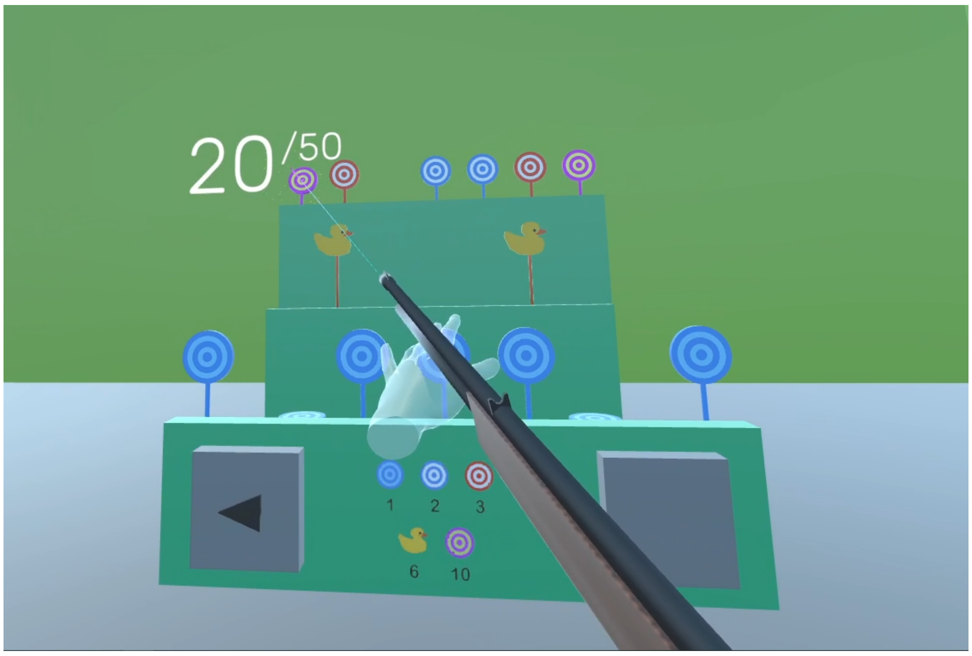


Figure 5.6: Screenshot from the application where the player is aiming at a target in the distance.

5.3.2 Shooting gallery

The Shooting Gallery game can be played in two modes: timed and untimed. After selecting one of the modes, the player can choose a gun and play. The implementation of the timed mode counts the player's score while waiting for the time to run out. The second and more relaxed mode checks whether the player has already reached a given score, no matter how quickly. Both modes track time and score, but the untimed mode does not show the timer in game.

After the second testing, three difficulty levels were also added to each mode that the player has to choose from. The difficulty sets the speed of moving duck targets, so each user can find the speed they find challenging but not overly difficult. An in-game screenshot that captures a player who aims at a target at a distance can be seen in Figure 5.6.

Gun types

There are three guns to choose from in the game: a rifle on a stand located near the trigger, a rifle on a stand located closer to the muzzle, and a rifle. Because most of their functionality is shared, they all inherit from a class `Gun`. The only differences between the gun types are their visualization and their sounds. The player chooses a gun at the beginning by grabbing it using the rock gesture. Then the game starts, and the player can shoot by performing the rock gesture with any of their hands, no matter the gun they chose. The shooting interaction is carried out using the component `Interactor` for the gun and `Interactable` for the targets. The freedom of choice of the hand that performs gestures should enable people who can use both hands to aim with one and shoot with the other hand and thus potentially increase accuracy, and enable people who can only use one hand to play the game too.

The rifles on a stand are visualized and aimed using only one of the player's hands, specifically the hand further from the player's body. The rifle, on the other hand, is "held" in both hands, and the hand closer to the player's body is always the one that holds the trigger part of the rifle. The positions of both hands are then subtracted and form a vector that affects where the gun points in terms of rotation. The implementation of the rifle visualization is done as follows:

```
protected override void VisualizeGun()
{
    posHandL = handL.Interactor.RaycastStartPt.position;
    posHandR = handR.Interactor.RaycastStartPt.position;

    //player is facing Z+
    //the hand closer to their body is holding the rifle
    bool holdingHandisL = posHandL.z < posHandR.z;

    posBackHand = holdingHandisL ? posHandL : posHandR;
    posFrontHand = holdingHandisL ? posHandR : posHandL;

    newHoldingHand = holdingHandisL ? handL : handR;

    if (HoldingHand != newHoldingHand) //swap holding hand
    {
        HoldingHand = newHoldingHand;
        transform.SetParent(holdingHand.Interactor.RaycastStartPt);
        transform.localPosition = Vector3.zero; //translate
    }

    transform.right = posFrontHand - posBackHand; //rotate
}
```

As previously mentioned, the hand further from the player's body will always control the one-handed gun movement. The rifle on stand works similarly to the two-handed rifle but substitutes the position of the hand closer to the player by a static point in the scene, enabling the player to use only one hand to aim.

All the guns cast rays to help the player see what they are aiming at. This is done by placing two empty objects on the gun prefab and then casting a ray in the direction of their difference, starting at the bullet hole.

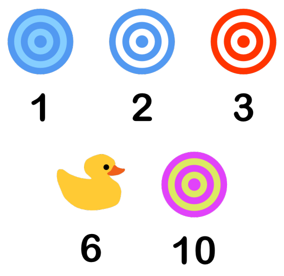


Figure 5.7: The user will get a different amount of points for each kind of target when they shoot it. This image is displayed in the game for the player to see it when playing.

Targets

Each target has an amount of points the player will get after shooting it (see Figure 5.7). Before the game begins, the targets are hidden behind shelves. After it starts, the targets will randomly choose a time from the set values and stand up when the time runs out. There is also an option to enable the target to fall down again without being shot, after an amount of time from a given range. If the target is shot, it falls and informs the game manager about the points the player got.

The duck targets move up and down and left to right at a given speed. The `ShelfOfTargetsSideAnim` may contain multiple of them and animate them according to its dimensions. When the targets reach the right end of the shelf, they will fall down and start moving to the left end of the shelf, where they will appear again. While moving left to right, they will also move up and down in between given Y positions.

5.3.3 Endless runner

When the player enters the Endless Runner scene, they have to perform a gesture (rock or paper) to start the game. The character then begins to run, and obstacles spawn randomly. The obstacle prefabs also contain collectible screws that the player can pick up to increase their score for each collected one. The obstacles spawn on a given place in the scene with given intervals in between them, and then move to the left with a given speed that is gradually increased with the player's current score. The character animation speed also depends on the speed of the game. The speed is computed using an `AnimationCurve` as follows:

```
animCurveVal = animCurve.Evaluate(score / maxSpeedScore);
speedMultiplier = 1 + (animCurveVal * (maxSpeedMultiplier - 1));
speed = defaultMovementSpeed * speedMultiplier;
```

The player has to perform the rock gesture to get the character to crouch and the paper gesture to jump, as can be seen in Figure 5.8. To stop crouching or jumping, the player simply needs to stop performing the gesture. The jump is dependent on

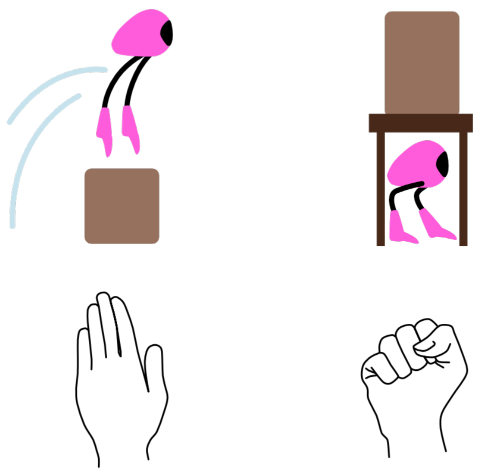


Figure 5.8: Character controls in Endless Runner game: paper to jump over a low obstacle, rock to crouch under a high obstacle.

an animation curve, which also ensures that the player cannot jump infinitely, but the character will eventually fall and begin running again.

After an obstacle is overcome by the player, it eventually reaches a position in the scene where it will be destroyed. If the player fails to perform the gestures correctly, the character will collide with an obstacle, and the game will end. The player is provided with three buttons: back to menu button, game restart button, and control button. The third button was added to the game after testing the implemented controls. The game used to let the player use both hands to control the character, but because it was common for the player to accidentally perform a gesture with a hand they were not using to control the character, that had to be removed. Instead, the player chooses a hand they will use to control the character, and they can switch hands when they want by pressing the respective button. The hand that controls the character changes color to help distinguish it. See Figure 5.9 capturing a player crouching the character to go under an obstacle.

After the second testing, the three buttons that could be previously accessed while the character was alive were hidden, leaving only the back to menu button, because the users kept accidentally pressing them even though their placement was meant not to interfere with the game. A second game mode was also added, which does not contain obstacles, but only collectible screws, and is time-limited. New prefabs without obstacles were thus created and are spawned in the mode and when the time runs out, the game ends.

The next feature that was added after the second testing was that when playing the mode with obstacles, there is a score threshold, and until the user reaches it, the robot is enclosed in a protective shield and cannot die. Each time the robot collides with an obstacle, their score is set to 0, but the game does not end. To support players and help them learn, audio cues were also added that say either “Rock” or “Paper” at a correct time for the user to listen and make the required gesture.

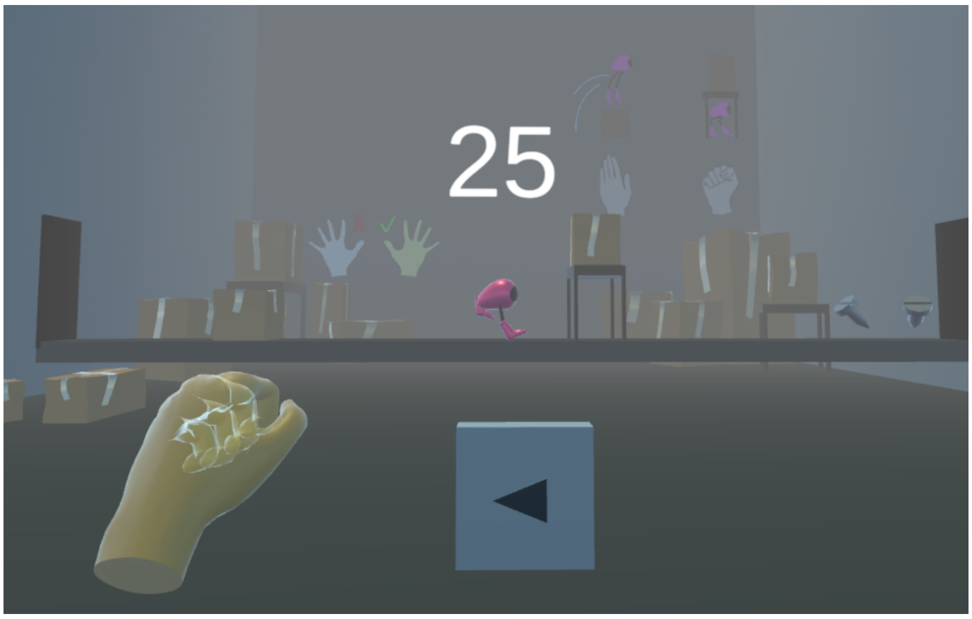


Figure 5.9: Player crouching the character using the rock gesture to go under an obstacle.

5.4 Interaction

To create an interaction for different objects in VR, classes `Interactable` and `Iteractor` were created. `Interactable` is a component that is used on the objects that the player can interact with (like a button, a grabbable object or a shooting target), `Interactor` is a component for objects that can interact with interactable objects (for example, a hand or a gun).

5.4.1 Interactor

The `Interactor` has three main functionalities:

- It saves a reference of two `Interactables`. The first interactable saved is the one that the `Interactor` is in physical collision with, the second one is in a ray cast invoked (distant) collision with it.
- It notifies the saved `Interactables` of the current gesture and collision.
- It can visualize a ray that serves for interaction with objects from a distance.

The `Interactor` has three `Func` delegates storing references to methods that set how the ray will be cast. For example, the `Gun` sets the following values to them, which allows aiming at targets in the Shooting Gallery as it relies solely on the transform of the gun:

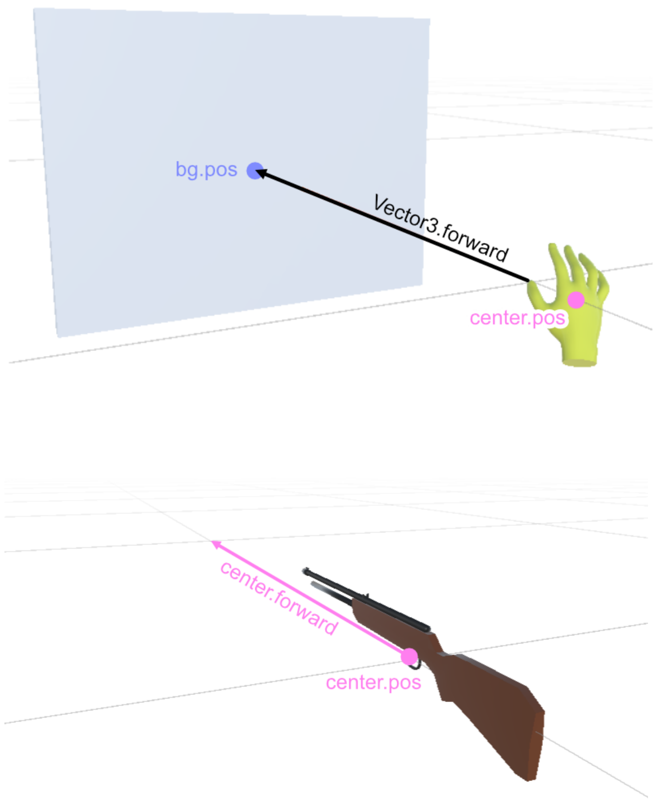


Figure 5.10: Illustration of the variables that exist in the computation of ray positions and directions for hand interactor and gun interactor respectively.

```
interactor.GetRayEndpoint =
    (center, bg) => center.pos + center.forward * rayLength;
interactor.GetRayStartpoint = (center) => center.pos;
interactor.GetRayDirection = (center) => center.forward;
```

In contrast, the `Interactor` instance in the `HandManager` is set to cast perpendicularly to the background (game boards and UI), so it uses the world's rotation in the computation. The implementation is structured in the following manner:

```
interactor.GetRayEndpoint =
    (center, bg) => new Vector3(center.pos.x, center.pos.y, bg.pos.z);
interactor.GetRayStartpoint = (center) => center.pos - Vector3.forward;
interactor.GetRayDirection = (center) => Vector3.forward;
```

See Figure 5.10 for a visual representation of the variables used in both computations.

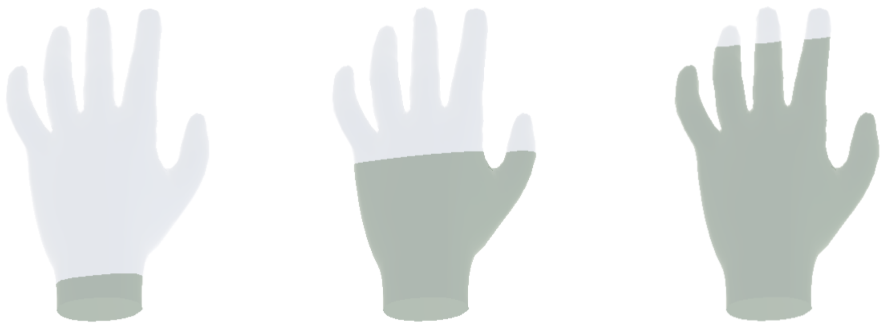


Figure 5.11: Gesture feedback given by progressively filling the hand performing a gesture with color.

5.4.2 Interactable

The `Interactable` serves the following purposes:

- From the `Interactor` it receives user-performed gesture information containing the type of gesture, whether it had begun or ended, the type of collision, and the instance of `Interactor` that called this function.
- It invokes different events depending on gesture or collision information. These events can be subscribed to by the other components of the `Interactable` object and thus can be reacted to. For example, a piece from the Tangram game will be grabbed by `OnRaycastRockEnter` (and dropped by `OnRaycastRockExit`) being invoked.

5.4.3 Custom buttons in VR

The UI elements in VR are implemented as mechanical buttons instead of the classic flat and digital ones. This decision was made primarily because the physical 3D representation mimics the real world more and may help the interaction be intuitive even for older people.

The buttons can be pressed in two ways, directly and from a distance. To directly push the button, the VR player needs to place their hand on the button (it does not matter what gesture the hand performs). The button will then highlight, outline, and also move further from the board it is on - this is the opposite of what happens in the real world, but this decision was made so that the button does not escape from the reach of the player's hand after being pressed. To press a button from a distance, the player needs to position the ray their hand casts so that it points at the button and then perform the grabbing gesture (rock).

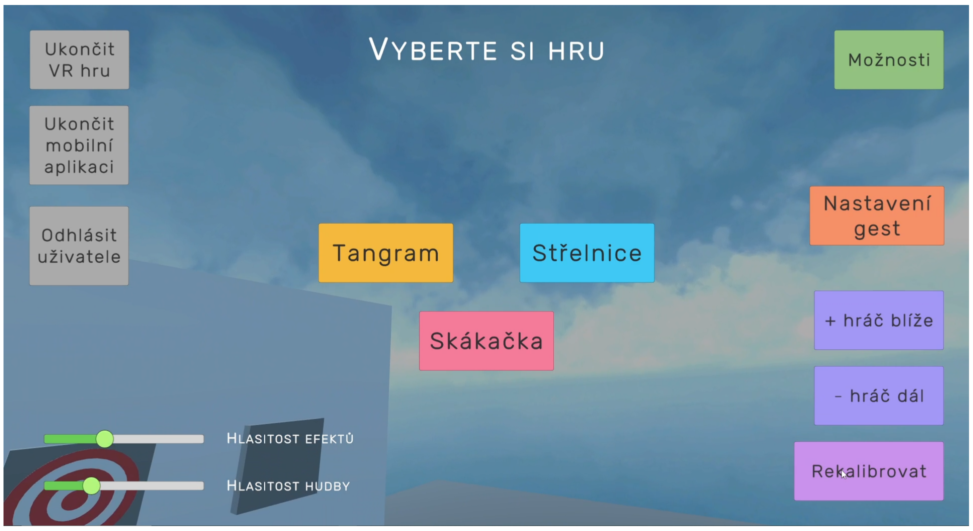


Figure 5.12: Main menu with expanded UI in the mobile device application while the game is running.

To avoid accidental pushing of buttons, each button performs the given action only after it has been pressed for a given duration. The progress of the button press uses audio and visual feedback to inform the player they are pressing the button - this way they should have enough time to move their hands from the button if they did not intend to press it. Visual progress is indicated by the progressive filling of the hand that pushes the button with a color as can be seen in Figure 5.11.

5.4.4 Mobile Device User Interface

In the mobile device application, the user interface (UI) is hidden from the start so that the assistant can see what the VR player is doing. To see all of the possible actions, the assistant needs to press the only button that is displayed in the upper right corner. The UI will overlay the game scene with a semi-transparent background and buttons. The background is not opaque to allow the assistant to see what happens in VR at least partially (see Figure 5.12). Each menu has different possible actions such as selecting a game, or selecting a game mode or a level. Many of these actions are available across all menus. The assistant always has the option to help the VR player recalibrate in the scene, move them forward or backward from the default position, or quit the mobile device application, quit the VR application, and hide the mobile device UI.

User Selection Menu

The User Selection Menu is the first screen shown to the assistant after the mobile device application is connected to the VR application. It allows them to manage users saved on the device and select one that represents the person who uses the VR

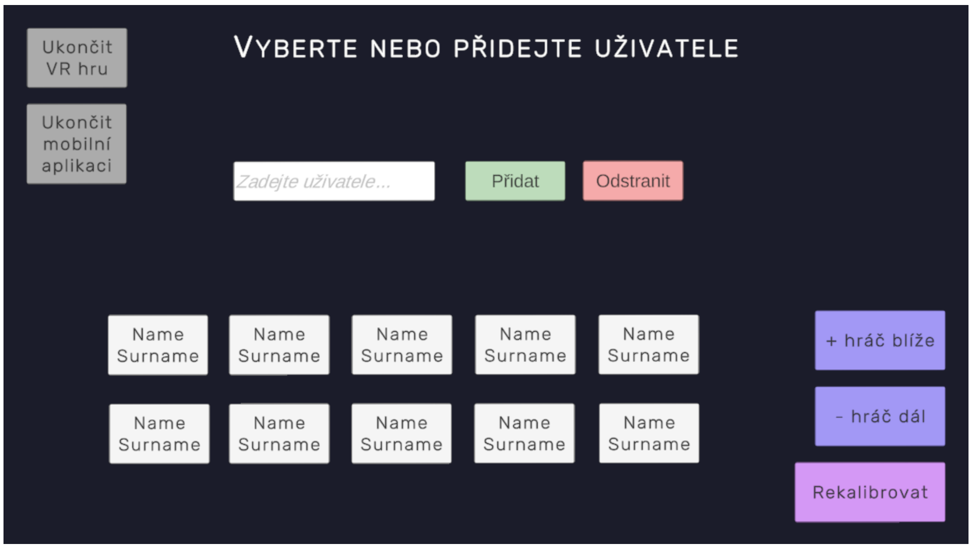


Figure 5.13: User Selection menu UI in the mobile device application - screenshot from the Unity Editor.

application at the moment. There can be up to 10 users on one device, as can be seen in Figure 5.13. If there are less than 10 users on the device, the buttons that do not represent any users are hidden. The local user is added or removed using the nickname they are assigned in the dashboard web application. The application does not allow for the creation of new users in the dashboard, nor their modification or deletion.

Main Menu

After selecting a user, the player and assistant appear in the Main Menu, which can be seen in Figure 5.14. Here, the assistant can help the VR user set custom gestures by opening the Gesture Settings Menu, setting sound and music volume, and most importantly selecting the game the VR user wishes to play. There is also an option to log out the current VR user.

Gesture Settings Menu

The mobile device user can help the VR user manage the custom gestures for the application in the gesture settings menu. There is an option to deactivate inputs from one of the hands if the VR user has a limb that does not work properly – it might trigger gesture detection and cause unwanted actions to happen in the VR application (as was discovered during the second testing session). The menu contains instructions on how to add a new custom gesture or how to remove all existing custom gestures. The default gestures (rock and paper variations) cannot be deleted. Figure 5.15 shows the layout of this menu.

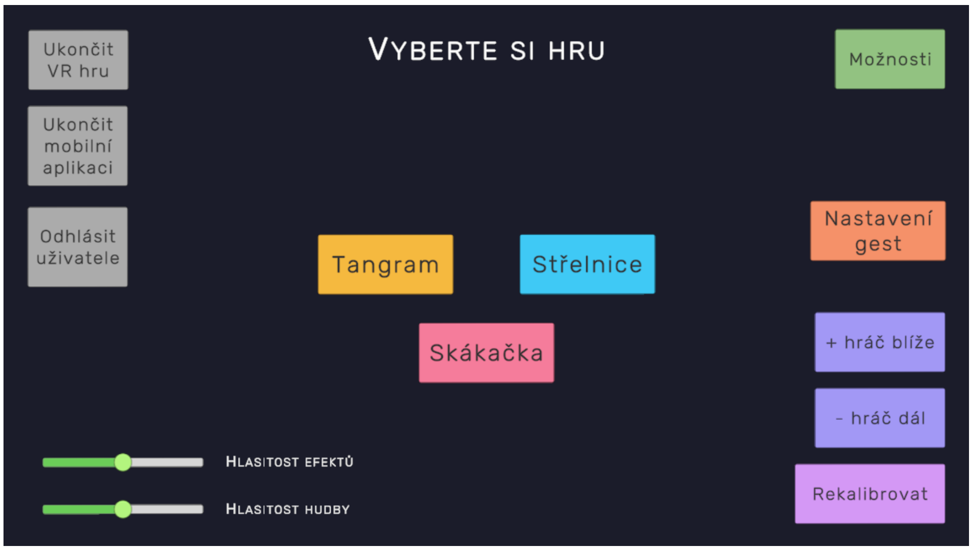


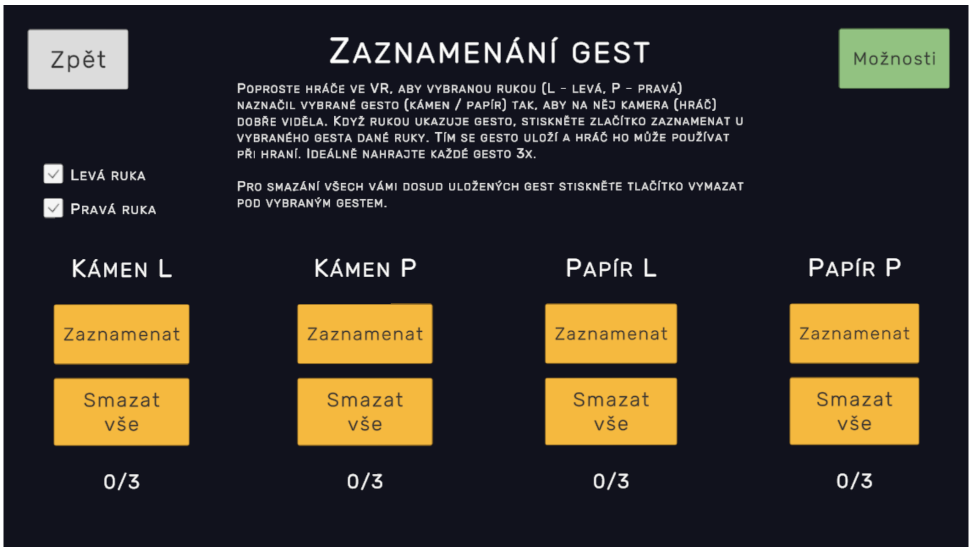
Figure 5.14: Main menu UI in the mobile device application - screenshot from the Unity Editor.

5.5 Mirror networking

This section will briefly describe the networking behind VR and mobile device applications that allows their synchronization and various functionalities.

The whole server runs on the local network. The host, which advertises itself on the server, can be discovered by a client. A custom network manager had to be implemented due to the need for two types of players - the VR and the mobile device player. The first connection on the server is the host, the network VR player, and the following ones are clients who are assigned a client player prefab. The client player has no functionality and serves only as a representation of the client's connection. The network VR player gets a different camera type for different types of connections (the host camera is for the VR player, and the client camera is for the mobile device player). The host device will turn its host camera on after authorization, and the client will be left with the client camera that has the host camera as a parent object (so it copies its transform). The `NetworkTransform` (together with `NetworkIdentity`) component is used to automatically synchronize the transformations of objects on the network.

The same scenes are used for VR and mobile device applications but each has access to different functionalities: the server allows the player to control the VR player and interact with the scene, and the client can use mobile device UI to switch scenes, recalibrate the VR player, etc. To provide the mobile device player with control over the VR application, the buttons on the mobile device UI call functions that have an attribute `[Command]`. Because the transformations of objects on the client might be delayed, it is better to allow only the host (VR player) to invoke events related to the hand movement and gestures and then call the respective functions with an attribute `[ClientRpc]`, which will propagate the changes to the client side.



Poproste hráče ve VR, aby vybranou rukou (L - levá, P - pravá) naznačil vybrané gesto (kámen / papír) tak, aby na něj kamera (hráč) dobře viděla. Když rukou ukazuje gesto, stiskněte zlačítko zaznamenat u vybraného gesta dané ruky. Tím se gesto uloží a hráč ho může používat při hraní. Ideálně nahrajte každé gesto 3x.

Pro smazání všech vámi dosud uložených gest stiskněte tlačítko vymazat pod vybraným gestem.

Figure 5.15: Gesture Settings menu UI in the mobile device application - screenshot from the Unity Editor.

5.6 Activity data logging

This section describes the implementation of logging that uses the library developed by Leoš Řeháček [22].

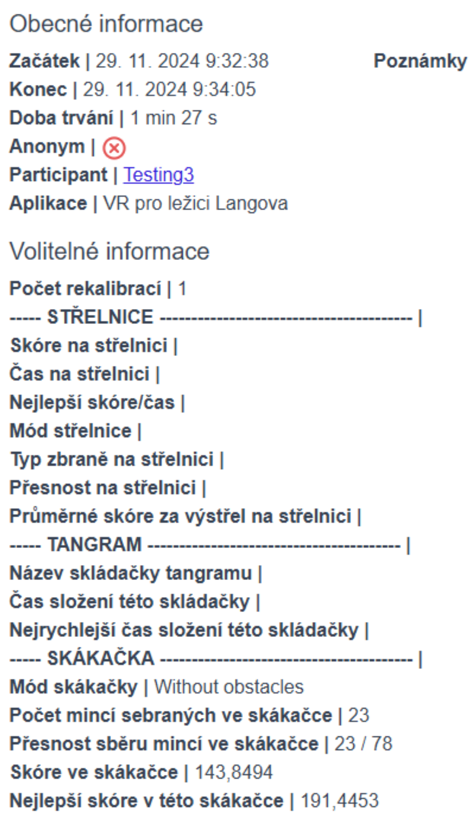
5.6.1 Logger Communication Provider class

`LoggerCommunicationProvider` is a singleton that communicates with the class `VrLogger` from the library and manages what data is logged, when the logging begins and ends, and more. Each function also checks whether it was called by the server (VR application), otherwise it performs no actions.

The `StartLogging` function is called when a user is chosen, a scene is changed, or when some game starts. It initializes the `VrLogger` and calls its logging function. The `StopLogging` function is called when a scene changes, a game starts, the application is stopped, or the current user is logged out. This means that a new record is sent to the server multiple times while the user uses the application.

This class also contains a function `FindParticipantIDByNickname` that invokes one of two `UnityActions` that are passed to it depending on whether the user was found using the given nickname.

A simple function for adding a variable with a given value to the custom data is implemented. This function is used frequently, as it allows to set values to the variables like score or accuracy reached in a game, which are then sent to the server with each record and displayed. See an example of custom data collected in the Endless Runner game during the final testing in Figure 5.16.



Obecné informace

Začátek | 29. 11. 2024 9:32:38 **Poznámky**
Konec | 29. 11. 2024 9:34:05
Doba trvání | 1 min 27 s
Anonym | ⊗
Participant | Testing3
Aplikace | VR pro ležici Langova

Volitelné informace

Počet rekalibrací | 1
----- STŘELNICE -- |
Skóre na střelnici |
Čas na střelnici |
Nejlepší skóre/čas |
Mód střelnice |
Typ zbraně na střelnici |
Přesnost na střelnici |
Průměrné skóre za výstřel na střelnici |
----- TANGRAM -- |
Název skládačky tangramu |
Čas složení této skládačky |
Nejrychlejší čas složení této skládačky |
----- SKÁKAČKA -- |
Mód skákačky | Without obstacles
Počet mincí sebraných ve skákačce | 23
Přesnost sběru mincí ve skákačce | 23 / 78
Skóre ve skákačce | 143,8494
Nejlepší skóre v této skákačce | 191,4453

Figure 5.16: Custom data example for the Endless Runner game. All variables except the ones relevant to the game remain unset.

The `LoggerCommunicationProvider` also contains a function for recording events and forwards the name of the event to the `VrLogger`. An example of events logged in the Shooting Gallery game during the final testing can be seen in Figure 5.17.

The library's `VrLogger` inherits from `LoggerBase`. The `LoggerBase`'s code had to be slightly changed, because while the application runs, the user's hands might sometimes be undetected and appear later. The `LoggerBase` did not account for this, as it set the variables `_isLeftHandNotNull` and `_isRightHandNotNull` only at start. Following changes (described in the comments) were made to the code of `LoggerBase`:

```
// these variables were made properties
// so that their values are set to match the current state of hands
private bool _isLeftHandNotNull { get { return leftHand != null; } }
private bool _isRightHandNotNull { get { return rightHand != null; } }
public void Start() {
    _isHeadNotNull = head != null;
    // this section was commented out because it was no longer needed
    /*_isLeftHandNotNull = leftHand != null;
    _isRightHandNotNull = rightHand != null;*/
    _isLeftEyeNotNull = leftEye != null;
    _isRightEyeNotNull = rightEye != null;
}
```

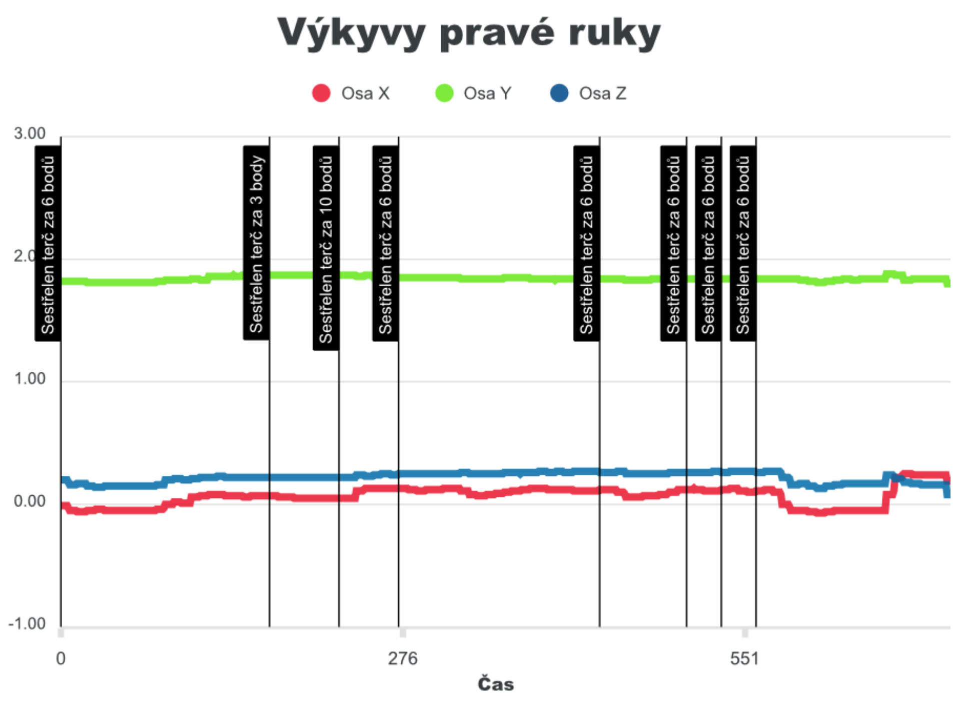


Figure 5.17: An example of events that occurred during the Shooting Gallery game that the VR Dashboard displays into a graph containing right-hand position fluctuations.

5.6.2 Additional processing of logged data

All the logged data may be additionally processed in order to see information about the player's behavior, their improvements in the various games, and more.

As an example, the logged data from the Shooting Gallery game was downloaded from the VRDashboard. Seeing a heatmap of angles where the player aimed could be interesting, but this way of processing the data from the hand positions can be used only when the user shoots using the two-handed rifle. This heatmap could help distinguish the places that the player likes to aim (and shoot) at and the ones that they avoid. For example, in the third testing (see Section 6.3 for more information), one of the participants shot with the two-handed rifle. Information about the angles at which he shot can be seen in Figure 5.18 displayed as a heatmap. The heatmap shows that the participant preferred to shoot the targets with the rifle rotated more to the left side, most likely because they held the rifle with their right hand closer to their body. Reference heatmaps can be found in Appendix E.2 together with information about where the player aimed and which hand they used to shoot. These reference heatmaps can be used to analyze the user-generated heatmap data, as they show how the heatmap looks if the player aims at the extremities of the Shooting Gallery.

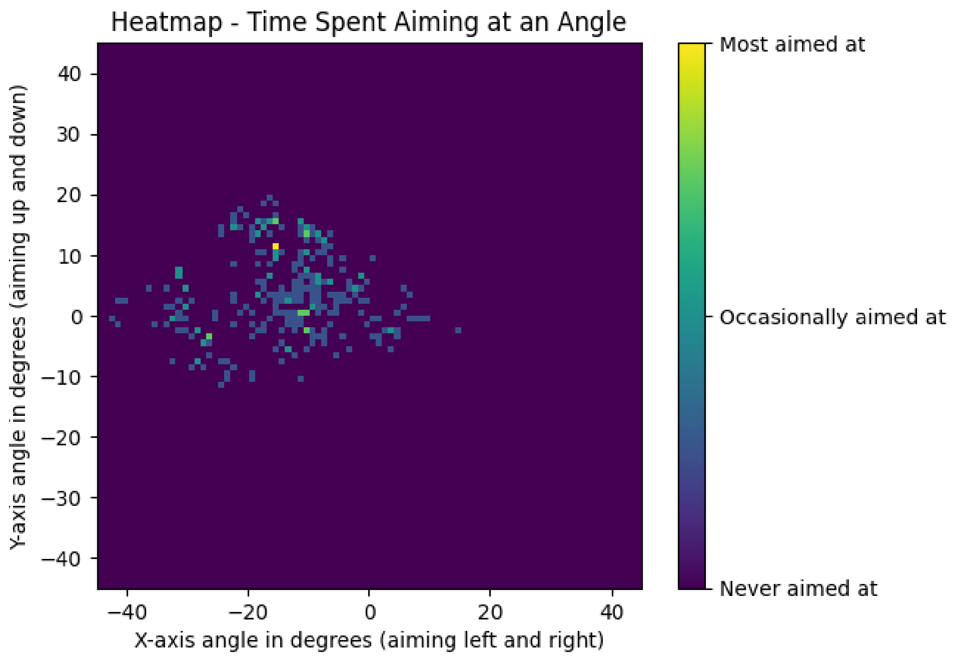


Figure 5.18: Information about time spent aiming at a particular angle displayed in a heatmap. The data was collected during the third testing.

The Python script for extracting the aiming angle data from `.json` files and displaying them using a heatmap can be found in the files handed in together with the thesis. The list of all attached files can be found in Appendix D.

More information could be extracted from the data without difficulty, such as the improvement of a user in terms of score in the games or the time spent solving a Tangram puzzle. The data could also show how many gestures of each kind the user made during the gameplay or what range of arm motion the user used during the games.

5.7 Tangram Piece Editor Actions

To make the creation of new Tangram puzzles easier, multiple features were implemented, most of which use a package called `EasyButtons`[1] that can display a button in Unity Editor that will call an implemented function.

The creation of Tangram prefabs manually without any automation takes a long time. The correct placements for the Tangram pieces must be manually found, then a Tangram slot must be created and manually set the same transform as the piece. After that the piece must be placed elsewhere on the board, and set a rotation which must be different from the correct on only by a multiple of 45 degrees.

[1]Easy buttons: github.com/madsbangh/EasyButtons.git#upm

The first function that was implemented was one that rotates the Tangram piece by 45 degrees when the function's button is pressed. This way, the person who creates the prefab can simply press the button a few times to achieve a different rotation than the one that is correct for the piece.

To be able to create the Tangram slot more easily, a function was implemented that creates the slot's GameObject, automatically copies the transform of the Tangram piece that is correctly placed at that time, and assigns the components that are needed to the slot object.

Chapter 6

Testing

This chapter describes in detail each iteration of testing, the environment, the process, discovered problems, and possible solutions to those problems.

6.1 Prototype testing

The first prototype testing was conducted in a home for the elderly. The state of the application at the time of the testing was as follows:

- The main menu in VR only had the Tangram game button; there was also a button for quitting the game.
- The mobile device main menu UI contained recalibration and forward/backward shift of the player, a button with the Tangram game, and two quit buttons - for the mobile device and VR application. The menu could also be expanded or hidden.
- The application had no audio feedback, and there was no visual feedback of hands being filled with color.
- There were two versions of rock and paper gestures; no custom gestures could be set. The threshold for recognizing gestures was relatively high to compensate.
- The Tangram pieces rotated the same amount as the player's hand that held them.
- Interaction (pushing buttons, grabbing pieces) worked directly and from a distance.
- There were two Tangram puzzles implemented - horse and heart shape.

Before the testing itself, the VR application and the mobile device application were connected to the local Wi-fi, and the connection was checked. The connection did not work properly, supposedly because the network settings prohibited the devices from connecting. The employee there had fortunately offered a portable router where the connection was successfully established.

To introduce the project to the employee, the mobile device application was shown to them and then they were instructed to try the VR application. The first informal testing thus began, and there were no serious problems. They understood what the goal of the Tangram game was and found the controls to be intuitive. The employee sometimes grabbed a piece unintentionally and thought that the buttons were supposed to be pressed with an index finger and only for a short time. Otherwise, they were able to complete both puzzles with occasional clues regarding the correct placements of the pieces.

The first client of the home for the elderly that helped test the application had dementia. At first, they seemed to understand what the goal of the testing and Tangram game was but then stopped being able to follow instructions that were given to them (even from the employee) and eventually got tired and frustrated. They never dropped a Tangram piece they held in their hand, even after multiple attempts to explain it to them, and their whole body was clenched. After having recognized the client was not feeling comfortable, the testing was ended.

The application was then given to another employee. They explained their discomfort with the rotation that needs to be performed multiple times to reach greater angles. They sometimes accidentally grabbed a Tangram piece when relaxing their hand and tried to press buttons using only the index finger for a short period of time.

The next two clients had problems with their hands. One of them had a hand that could not be detected by the VR application at all, and the other had an immobile left arm, decreased mobility in the right shoulder, and all fingers immobile except for the thumb and index finger on the right hand. Unfortunately, the application did not recognize any attempts of the client to perform gestures, and the testing was thus ended early.

The last client had one blind eye. They said they had problems in the past using VR applications because of the lack of depth perception. When testing the application, the client kept accidentally grabbing the pieces when they relaxed their hand and sometimes failed to grab the pieces when they meant to. They did not mention any problems regarding their blindness and were able to successfully complete the puzzles in a relatively short time.

In conclusion, all of the problems that were encountered were:

1. **Gesture problems:** The VR players had problems with not grabbing the piece when intending to, or accidentally grabbing it when relaxing a hand.
2. **Lack of feedback:** The buttons have to be pressed for a duration of time and some gestures need to be held for multiple seconds, but the game does not indicate this to the player.
3. **Tangram piece rotation:** The rotation of the Tangram piece copies the rotation of the hand, but that means that the player has to sometimes pick up a piece, rotate and drop it, pick it up again, and repeat this process multiple times to reach the desired rotation of the piece.

4. **Tangram game:** The goals and controls of the Tangram game may be hard to comprehend (for example, for people with dementia).

As the application must be intuitive, easily controlled, and entertaining for many people with different physical and mental abilities, some solutions need to be proposed and implemented to fix the encountered problems. The solutions proposed in the next list are numbered according to the numbering in the list of problems they are supposed to fix.

1. **Custom gestures and lower tolerance in gesture detection:** Clients with decreased finger mobility might benefit from being able to set custom gestures that they could use to interact with the application. As a result, the tolerance threshold in gesture detection could be lower, which should stop the players from accidentally grabbing Tangram pieces.
2. **Addition of feedback:** Audio and visual feedback that would be played while a button is pressed or a gesture (which has to be held for a duration of time) is being performed should help players understand that they have to continue doing what they currently are to perform the desired action.
3. **Modification of the Tangram piece rotation angle:** To avoid forcing the players to perform rotation in multiple steps, the angle of rotation of the hand could be multiplied and applied to the rotation of the Tangram piece held.
4. **Simple practice levels:** To practice grabbing, rotating and dropping the Tangram pieces, some very simple levels could be introduced, where the correct placement of the pieces should be clear from the start. The simplicity of the level should help the player focus on their actions and instructions given by an assistant. An example of an easier level can be seen in Figure 6.1

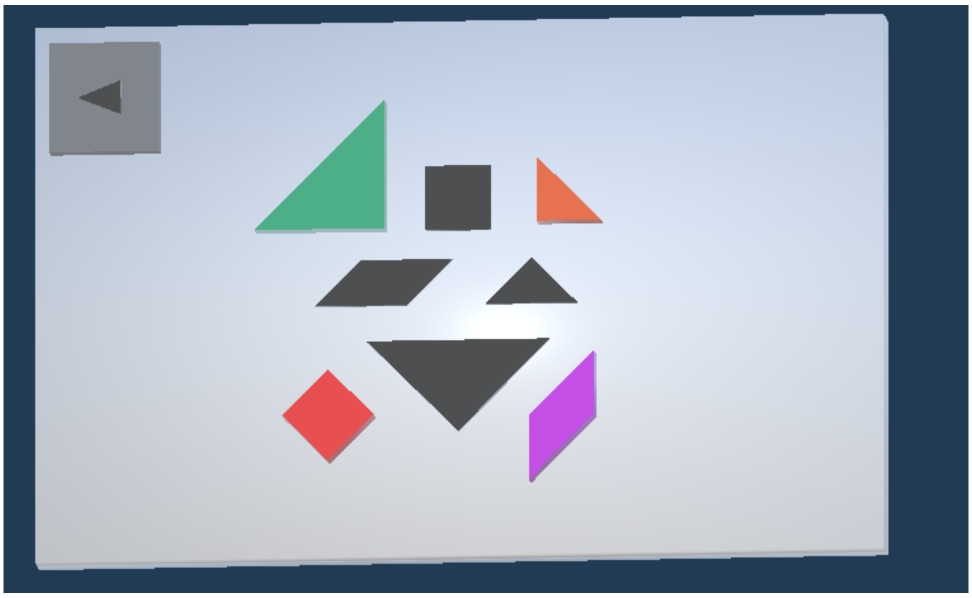


Figure 6.1: Simple practice level for people with limited mental abilities.

6.2 Testing of the games

The second testing took place in the same home for the elderly as in the previous testing. The state of the application at the time of the testing was as follows:

- The Endless Runner could be lost immediately at the first obstacle if the person did not perform the gesture correctly.
- The Endless Runner had only one mode – with obstacles.
- There were three buttons in the Endless Runner (switch hands, quit game, reset game), even during the game.
- The Shooting Gallery had only two modes – timed and untimed.
- The Shooting Gallery could be quit by shooting once while aiming at the quit button.
- There was no option to disable input (gestures) from a hand.
- There was no snapping to the position of the Tangram pieces, only rotation.
- The goal shapes in the Tangram puzzle were filled in with a dark color.

The testing process was similar to the first prototype testing. Patients were informed about the purpose of the test, instructed on how the application works, and then observed while using the application. At the end of each test, the participant was asked a few questions that focused on the experience of controlling the application and the level of enjoyment of each game.

The first patient was an 82-year-old man with a painful shoulder. He found the Tangram controls intuitive, but had trouble distinguishing the pieces and the goal shape when playing the simple practice level (shown in Figure 6.1). He really enjoyed the Shooting Gallery game and made many positive comments such as: *„That would be something for our boys!"* and *„I would bankrupt the fair worker!"*. He tried both rifle and rifle on stand, but did not enjoy using the two-handed rifle as it is harder to control and his shoulder hurt. The Endless Runner was too fast for him, even though he seemed to understand the controls. His favorite game was the Shooting Gallery.

The second patient was a 68-year-old man with a paralyzed left arm. He also had problems distinguishing the Tangram pieces from the silhouette on the board. In the Shooting Gallery, his hand frequently triggered the shot as it was paralyzed and its fingers were closed in a fist. It took him a while to figure out how to aim with the gun, but even though he was not very successful at hitting the targets, he wanted to continue playing and get better. Once, he accidentally hit the quit button, so he had to replay the game after that. In the Endless Runner, it was very obvious that he liked to collect the screws and he even mentioned it verbally many times. As a result, this patient frequently ignored the obstacles and lost the game relatively early. He also accidentally pressed the menu buttons. They were placed in a reachable distance below eye level, but high enough for a person who relaxes their hands near their hips to not press them. The patient seemed to want to see his hands while playing, and thus placed the hands at the same height as the buttons. His favorite game was the Shooting Gallery and he said that nothing in the games is hard to understand, it just takes a while to get used to.

An 82-year-old woman without serious physical limitations was the next participant. Like the previous participants, she had also struggled to distinguish the Tangram pieces and silhouette. In addition to that, she kept accidentally grabbing the pieces she had already placed correctly. In the Shooting Gallery, she struggled with how the aiming worked initially but figured it out after a while. In the Endless Runner, she accidentally reset the game by pressing one of the buttons but she excelled at the game otherwise and got one of the highest scores among the patients. When asked about her favorite game, she said that she liked all of them.

The next patient was a 92-year-old woman with limited mobility of the arm and shoulder. The first problem encountered was that this patient kept her head tilted back and could not properly reach the pieces in Tangram. Although there were several attempts to correct her posture, the attempts were unsuccessful (whether it was because the patient did not understand the instructions or wanted to stay in that particular position). Although she had to engage her core muscles to play, the patient still managed to complete the Tangram puzzle and said she enjoyed it. In the Shooting Gallery, she had trouble comprehending how to aim the gun at the targets, and she was eventually stopped and given the last game. She found that the Endless Runner was hard to understand and did not hold the gestures long enough to overcome the obstacles. She also accidentally pressed the buttons in the Endless Runner, just like the other patients. Her favorite game was Tangram and she did not like the Shooting Gallery at all.

The last participant was an 81-year-old woman in a wheelchair. In Tangram, she had problems distinguishing the goal shape and pieces, too, and she sometimes grabbed a piece accidentally. However, she placed the pieces accurately and finished the puzzle relatively quickly. In the Shooting Gallery, she had to be told to release the rock gesture when she did not want to shoot, after that she understood the game and even aimed at the static targets very accurately. She found that the duck targets were moving too quickly. She enjoyed playing the Endless Runner and was successful in getting high scores.

In conclusion, the testing helped discover new problems that can be fixed to improve the application and showed that each game had at least one person who liked it the best. Listed here are the main problems that should be fixed in the next iteration:

1. **Tangram piece accidental grabbing:** Users kept accidentally grabbing the pieces they had already correctly placed.
2. **Tangram pieces and goal shape indistinguishable:** The majority of participants had problems distinguishing between the Tangram pieces and the goal shape in the puzzle level presented to them.
3. **Quick moving targets:** The duck targets in the Shooting Gallery move too quickly for some people.
4. **Paralyzed hand triggering gestures:** A patient with a hand paralyzed in such a way that it held a rock gesture all the time could not stop shooting in the Shooting Gallery, because the application kept recognizing the gesture.

5. **Accidental game quitting:** Participants kept accidentally quitting games (Shooting Gallery and Endless Runner) by pressing the buttons without intending to.
6. **Tempting screws:** A patient enjoyed the collection of screws in the Endless Runner so much that they ignored the obstacles and always lost the game early because of that.

The design and implementation had to be changed to fit the limitations, needs, and motivations of the patients. Thus, the following changes were proposed and implemented into the application after the second testing:

1. **Tangram piece snapping option:** An option was added to the game to turn the Tangram snapping on / off. If the setting is on, the pieces snap immediately if they are placed correctly and cannot be moved afterward.
2. **Tangram goal shape to outline:** The goal shape that used to be represented by a filled-in shape is now displayed only using an outline to inform the player that it cannot be grabbed.
3. **Moving targets speed option:** Three difficulty options were added for both modes of Shooting Gallery. The difference in these difficulty levels is in how fast the duck targets move to provide different users with a suitable challenge.
4. **Option to disable input from the hands:** An option was added that allows to disable input from the left or right hand. It can be found in the gesture settings menu.
5. **Quitting a game:** To quit a Shooting Gallery game, the player now has to hold the gesture for a given period of time, so they can not quit accidentally. The button for quitting the game was removed from the in-game menu in the Endless Runner, so it can only be pressed when the robot is not running.
6. **Screw-collecting mode:** A mode was added to the Endless Runner game where there are no obstacles and only screws are spawned. The goal of the new mode is to collect as many screws as the user can before time runs out. **Invulnerability period:** The robot is now invulnerable in the mode with obstacles until the player reaches a given score without colliding with any obstacles. The score is set to 0 every time the player hits an obstacle, but the game does not end. This is visualized to the player using a shield bubble that is displayed around the robot.

6.3 Final testing

The last testing at the home for the elderly had the goal of verifying that all changes made according to the second testing have helped the users with controlling the application or improved their enjoyment. The testing process was the same as for the previous testing sessions.

The first patient was an 86-year-old woman in a wheelchair with a tremor that affected her head movement and her fingers were not able to fully extend (which was not apparent at first). Otherwise she did not suffer from any noticable physical or mental illnesses.

She understood Tangram's goal very well and completed the horse puzzle with a few hints. However, she had trouble grabbing and putting down the pieces as she often put her hand behind the game board and did not see how her hand and the piece were positioned. When moved virtually further from the board and instructed that she could grab the pieces from a distance, she tensed and tried to reach the pieces physically again, as she did not prefer the interaction from a distance. At first, the feature that snaps the pieces in place after they are correctly placed was turned off, but the patient commented that they were not able to place the piece precisely. The feature was then turned on for them, which they enjoyed.

She chose the two-handed rifle in the Shooting Gallery. She understood the controls well from the start. Despite the head tremor, she was successful in shooting the targets. The arm she had to have stretched in front of her to aim the gun got tired after a few minutes of shooting, so she did not complete the goal of reaching 50 points.

After she agreed to play the Endless Runner, she was instructed that she did not have to hold her hand in front of herself to control the robot, and she rested her hand on her lap. She tried the screw-collecting mode at first, then tried the mode with obstacles. She did not clearly hear the audio instructions of which gesture she should perform that were played before each obstacle, but this was not clear until the end of the testing. She also had problems performing the default paper gesture as her fingers were not able to straighten fully, but she preferred to end the testing here rather than have the gestures recalibrated.

She claimed that she liked all the games that she played and said that she had the most problems when playing the Tangram, as it was the first game she tried. She found the Shooting Gallery easier to control. In the Endless Runner, she confessed that she did not understand the goal at first, but then it got better. She preferred the screw-collecting mode over the one with obstacles.

The second participant was a 90-year-old man. He had no mental limitations, and the only noticeable physical limitation was that his right hand was not as strong as his left. After being introduced to the tagram game, he seemed to find the controls intuitive and finished the horse puzzle without any help. At first, the snapping of pieces was turned off, but the patient enjoyed the game more after it was turned on.

When asked if he wanted to use a one-handed or a two-handed weapon in the Shooting Gallery, he wanted to try the one-handed weapon but quickly changed his mind after trying it out. He preferred the two-handed weapon and, as a former soldier, he performed very well in the Shooting Gallery.

The patient said he enjoyed the Shooting Gallery game the most. He preferred the screw-collecting mode in the Endless Runner over the one with obstacles.

In summary, no serious problems were discovered during the final testing, but the minor ones are listed below:

1. **Tangram pieces can be held behind the board:** The first participant put her hand behind the Tangram board and did not see how she had the piece positioned.
2. **Snapping is preferred in Tangram:** Both participants complained about not being able to place the Tangram pieces precisely. They seemed relieved when the Tangram piece-snapping was turned on (and the default setting is with snapping turned off).
3. **Unclear instructions in Endless Runner:** Participants who are not used to playing computer games do not find it intuitive that their actions affect and control a character.
4. **Custom gestures were not set at the beginning of the testing:** As the problem with the gestures not being recognized emerged during the gameplay and the participants did not want to be interrupted, correct custom gestures were not set throughout the whole testing.

To solve the issues mentioned above, the following solutions could be implemented:

1. **Tangram pieces ghosting:** To visualize the pieces even though they are behind the board, ghosting could be used in the same way as it is when manipulating with them from a distance.
2. **Snapping turned on as default option:** Simply changing the default setting for Tangram piece snapping could improve the gaming experience.
3. **Changed instructions in Endless Runner:** Instructions focusing on clearly explaining the relationship between the player's gestures and the robot's actions could be added.
4. **Custom gesture setting forced at first playthrough:** When a new user is added to the device and starts the application, they would be forced to set at least one custom gesture for each gesture type on each hand.

Chapter 7

Conclusion

The goal of this thesis was to analyze the target users, their limitations and motivations, and then design and implement a VR application with suitable gaming experiences and controls. In addition, a mobile device application was to be implemented through which an assistant could help the VR player perform tasks. The prototype applications were to be tested with a group of target users and appropriate changes and fixes were to be designed and implemented.

An interview with workers from three institutions provided insight into the lives of bedbound people. The Tangram puzzle game was chosen as the first gaming experience. It was designed and implemented into the application including two complex puzzles and one simpler for practicing the controls. The second gaming experience added to the application was a Shooting Gallery that provides a timed and untimed mode, both with three difficulty options that vary the speed of moving targets. The Shooting Gallery also allows the user to choose from three different guns, two of which can be controlled using only one hand. The third game is an Endless Runner, where the user controls the character's jumping or crouching using paper or rock gestures, respectively. There is a mode with obstacles and a timed screw-collecting mode.

The resulting VR application features upright redirection that helps the player see the world and interact with it as if they were standing. The interaction in the VR application works without VR controllers as it uses gesture detection. The gestures can be set by the player to fit their needs and limitations using the associated mobile device application, which also provides means to navigate the VR player through menus and help with recalibration.

The application logs data about the transformation of the VR user's head and hands, and also additional information about the actions the user performs in the games. This data can be analyzed to see the progress of each patient in playing the games.

The application was tested twice during development, then once at the end. The results of the first two testing sessions led to the design and implementation changing, in order to create an application and games better suited for the target user group. The third and final testing confirmed that the application is entertaining and motivates the users to move their arms and hands. Appendix E.1 lists videos showing the whole application in the state it was in during the third testing both from the VR user's and mobile device user's perspective.

To further expand the application, more levels in the existing games or new gaming experiences could be added in the future. A more detailed tutorial could also be created to further explain the games and controls.

Part of this work was previously published [23] and presented at the 15th International Conference on Disability, Virtual Reality, and Associated Technologies (ICDVRAT), held September 3 to 6, 2024 in Prague, Czech Republic.

Bibliography

[1] Steven M LaValle. *Virtual reality.* Cambridge university press, 2023.

[2] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality.* Association for Computing Machinery and Morgan & Claypool, 2015. ISBN: 9781970001129.

[3] Jaekyung Kim et al. "Virtual Reality Sickness Predictor: Analysis of visual-vestibular conflict and VR contents". In: *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX).* 2018, pp. 1–6. DOI: 10.1109/QoMEX.2018.8463413.

[4] Hironori Akiduki et al. "Visual-vestibular conflict induced by virtual reality in humans". In: *Neuroscience Letters* 340.3 (2003), pp. 197–200. ISSN: 0304-3940. DOI: https://doi.org/10.1016/S0304-3940(03)00098-3. URL: https://www.sciencedirect.com/science/article/pii/S0304394003000983.

[5] J.A. Jacko. *Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition.* Human Factors and Ergonomics. CRC Press, 2012. ISBN: 9781439829448.

[6] Jaekyung Kim et al. "Virtual Reality Sickness Predictor: Analysis of visual-vestibular conflict and VR contents". In: *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX).* 2018, pp. 1–6. DOI: 10.1109/QoMEX.2018.8463413.

[7] Doug A. Bowman, Ryan P. McMahan, and Eric D. Ragan. "Questioning Naturalism in 3D User Interfaces". In: *Communications of the ACM* 55.9 (2012). DOI: 10.1145/2330667.2330687. URL: https://doi.org/10.1145/2330667.2330687.

[8] Thomas van Gemert et al. "Towards a Bedder Future: A Study of Using Virtual Reality while Lying Down". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.* 2023, pp. 1–18.

[9] Hideki Kawai, Hiroki Hara, and Yasuyuki Yanagida. “Effect of Reclining Angle on the Perception of Horizontal Plane for HMD Users”. In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2018, pp. 599–600. DOI: 10.1109/VR.2018.8446090.

[10] Tianren Luo et al. “Exploring Sensory Conflict Effect Due to Upright Redirection While Using VR in Reclining & Lying Positions”. In: Oct. 2022. DOI: 10.1145/3526113.3545692.

[11] Tianren Luo et al. “Exploring Locomotion Methods with Upright Redirected Views for VR Users in Reclining & Lying Positions”. In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. UIST ’23. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400701320. DOI: 10.1145/3586183.3606714. URL: https://doi.org/10.1145/3586183.3606714.

[12] Travis Lowdermilk. *User-centered design: a developer’s guide to building user-friendly applications.* " O’Reilly Media, Inc.", 2013.

[13] Donald A. Norman. *The psychology of everyday things.* Basic Books, 1988. ISBN: 978-0-465-06710-7.

[14] *What is User Centered Design (UCD)?* en. https://www.interaction-design.org/literature/topics/user-centered-design. Accessed: 2024-3-11. June 2016.

[15] James R. Lewis. “Usability Testing”. In: *Handbook of Human Factors and Ergonomics.* John Wiley Sons, Ltd, 2012. Chap. 46, pp. 1267–1312. ISBN: 9781118131350. DOI: https://doi.org/10.1002/9781118131350.ch46. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118131350.ch46. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118131350.ch46.

[16] Joseph F Dumas and Janice C Redish. *A practical guide to usability testing.* Greenwood Publishing Group Inc., 1993.

[17] Jakob Nielsen. *Why you only need to test with 5 users.* Accessed: 2024-7-11. Mar. 2000. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[18] Munir Oudah, Ali Al-Naji, and Javaan Chahl. “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques”. In: *Journal of Imaging* 6.8 (2020). ISSN: 2313-433X. DOI: 10.3390/jimaging6080073. URL: https://www.mdpi.com/2313-433X/6/8/73.

[19] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. “Vision based hand gesture recognition”. In: *International Journal of Computer and Information Engineering* 3.1 (2009), pp. 186–191.

[20] *What happened to “Laying Down Mode”?* Accessed: 2024-7-11. Aug. 2023. URL: https://communityforums.atmeta.com/t5/Get-Help/What-happened-to-quot-Laying-Down-Mode-quot/td-p/1072419.

[21] *Serious Question: Any VR games that you can play in bed seated or lying down?* Accessed: 2024-7-11. Nov. 2023. URL: `https://www.reddit.com/r/OculusQuest/comments/17lctx7/serious_question_any_vr_games_that_you_can_play/`.

[22] Leoš Řeháček. “Aplikace pro sběr a analýzu dat z vr tréninkových aplikací”. MA thesis. Czech Technical University in Prague, 2024.

[23] Tereza Langová, and David Sedláček. “Virtual Reality Games for Lying Patients”. English. In: *Proceedings of the 15th International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT 2024)*. Ed. by Iveta Fajnerová et al. 2024, pp. 147–149. ISBN: 978-80-87142-63-9.

Appendix A

Set up manual

This appendix contains a brief manual on how to properly set up the devices so that the application works correctly.

First, follow the instructions listed below:

1. Enable developer mode on both mobile device and Meta Quest headset.
2. Install the VR application on the Meta Quest headset.
3. Enable hand tracking in the device settings of the Meta Quest headset.
4. Check if hand tracking works correctly by putting away the VR controllers and looking at your bare hands. Virtual hands should appear in the headset that copy your hand movements.
5. Install the mobile application on the mobile device.
6. Connect both devices to the same network.
7. Start the application on both devices.

After performing the steps above, the mobile application should display a screen with a single button *"Connect to the VR"*. Press the button. The application should connect the devices and the mobile application should show a User Selection Menu (see Chapter 5.4.4). If the devices do not connect, it is possible the network you are connected to has security precautions that do not allow two devices to connect to each other. In this case, try to connect to a different network.

Appendix B

List of used packages

The Unity project of the application contains the following packages:

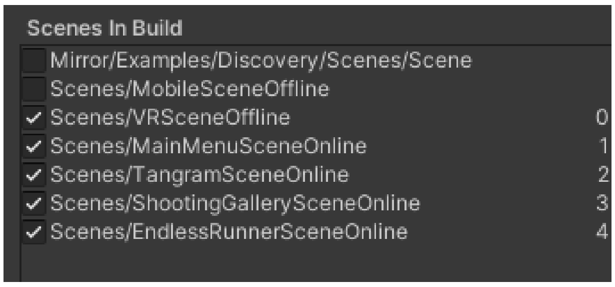
- **VR dashboard logger**[1] **(Version 1.0.8)**: Used to log VR data to a server.
- **Mirror**[2] **(Version 81.4.0)**: Used for networking.
- **Easy Buttons**[3] **(Version 1.4.0)**: Used for creating buttons for the Unity editor that serve for easier creation of Tangram levels.
- **XR Hands**[4] **(Version 2.5.0.)**: Used to visualize VR hands.
- **Oculus XR Plugin**[5]: Provides display and input support for Oculus devices.

[1] VR dashboard logger: gitlab.fel.cvut.cz/rehacleo/vr_dashboard_logger.git
[2] Mirror: mirror-networking.gitbook.io/docs
[3] Easy Buttons: github.com/madsbangh/EasyButtons.git#upm
[4] XR Hands: docs.unity3d.com/Packages/com.unity.xr.hands@1.3/manual/index.html
[5] Oculus XR: docs.unity3d.com/Packages/com.unity.xr.oculus@4.0/manual/index.html

Appendix C

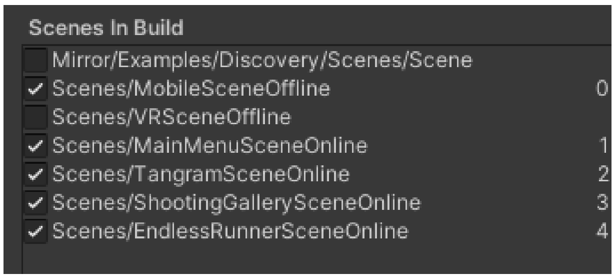
Building the application

The difference between building the application for mobile device and the VR headset is in including different "offline" scenes into the build using Unity Editor's Build Settings. Figure C.1 shows how the settings should look for each of them. To build the VR application, "VRSceneOffline" must be included in the build, and "MobileSceneOffline" must be excluded. These should be checked in the opposite way when building for mobile devices.



```
Scenes In Build
☐ Mirror/Examples/Discovery/Scenes/Scene
☐ Scenes/MobileSceneOffline
✓ Scenes/VRSceneOffline                      0
✓ Scenes/MainMenuSceneOnline                 1
✓ Scenes/TangramSceneOnline                  2
✓ Scenes/ShootingGallerySceneOnline          3
✓ Scenes/EndlessRunnerSceneOnline            4
```

(a) : Scenes included for VR device build.



```
Scenes In Build
☐ Mirror/Examples/Discovery/Scenes/Scene
✓ Scenes/MobileSceneOffline                  0
☐ Scenes/VRSceneOffline
✓ Scenes/MainMenuSceneOnline                 1
✓ Scenes/TangramSceneOnline                  2
✓ Scenes/ShootingGallerySceneOnline          3
✓ Scenes/EndlessRunnerSceneOnline            4
```

(b) : Scenes included for mobile device build.

Figure C.1: Build Settings - scenes to be included to build for different devices.

Appendix D

List of attached files

- **AimingHeatmap.py** – Python code processing logged .json data from Shooting Gallery to display an aiming heatmap.
- **GitlabRepo.txt** – Text file with a link to a Gitlab repository with the whole Unity project.
- **imgs.zip** – 6 required screenshots of the application.
- **Mobile.zip** – Contains a build for the mobile device.
- **VR.zip** – Contains a build for the Meta Quest 2 headset.

Appendix E

Videos and screenshots

This appendix contains additional images and links to videos showing the application.

E.1 Videos

The first video can be found at `youtu.be/09lbT0ASmtQ` and shows the whole application controlled independently by the VR player. At the beginning, they recalibrate in the scene so that it fits their current comfortable position and rotation. Then they select a user account that belongs to them, followed by starting the Tangram puzzle game. They turn on auto-snapping and start the heart puzzle, which they eventually complete. They start the simple shape puzzle to show the hinting button next. After completing the Tangram puzzle, they start the Shooting Gallery at the highest difficulty without a timer. They choose a two-handed rifle and shoot until they reach the given score. After winning, they restart the game and show the one-handed rifle, also. In the following footage, the player starts the Endless Runner with obstacles, shows the controls, and loses the game shortly after reaching the score that makes the shield disappear. By going back to the menu, they are then able to start the screw-collecting mode. After the time runs out, they go back to the main menu, log-out the user, and quit the application.

The video at `youtu.be/rH0DpJ1uXhE` shows the entire application. In the top left corner, there is a smaller screen showing what the mobile device application user sees, and the larger video shows what the VR player sees. At first, adding, removing and choosing a user is shown. Then the mobile device player recalibrates the VR player. The setting of a custom rock gesture in the shape of a bunny follows. This gesture is used to push some buttons and is then removed. The video also shows that the player can be moved closer or further from the board in front of them by pressing a button in the mobile device application. After the mobile player selects the Tangram game, they turn auto-snapping of the pieces on and off a few times. The puzzle in the shape of a horse is chosen by the mobile player and then completed by the VR user. After the puzzle, the mobile device player navigates back to the main menu and selects the Shooting Gallery game with the untimed mode of medium difficulty. They also choose a two-handed gun for the VR player. The VR player then reaches the score of 50 points and finishes the game. Next, the mobile device player quits the Shooting Gallery and starts the Endless Runner with obstacles. The mobile device player then changes the hand that will control the character a few times. The VR player then starts the game and shows the use of the bubble shield as they bump into an obstacle. As they are protected, the game continues, only the score is lowered to zero. The next time a character touches an obstacle is after they have reached the score of 80, which means the shield is no longer there to protect the VR player from losing. The game ends.

E.2 References for the heatmaps in Shooting Gallery

These additional heatmaps are created from the logged data that was downloaded from the VR Dashboard. These images can be used to analyze the heatmap data gathered when using the application with patients, as they show how the heatmap looks if the player aims at the extremities of the Shooting Gallery.

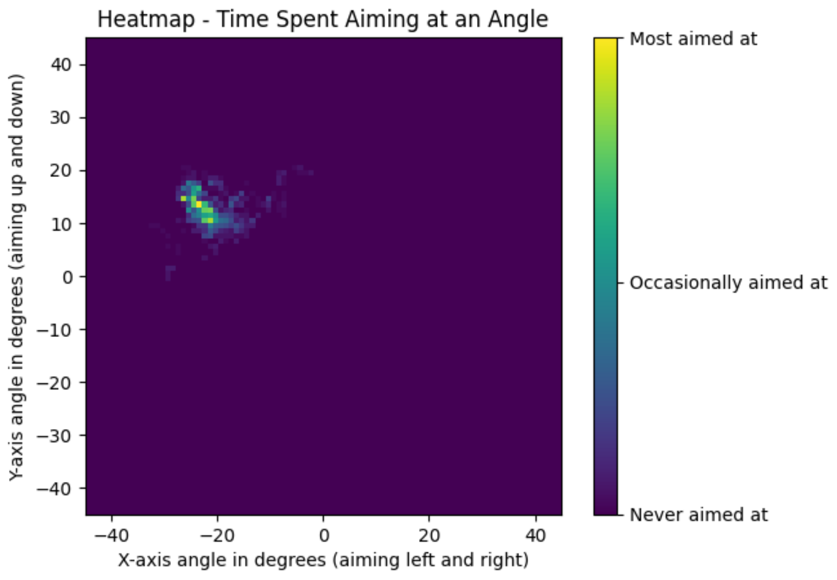


Figure E.1: Heatmap from data collected while shooting at upper left targets in the Shooting Gallery.

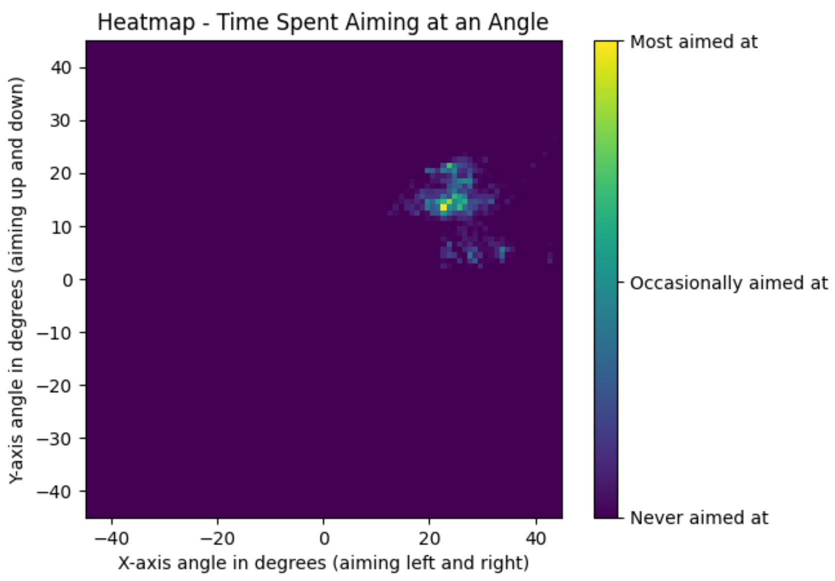


Figure E.2: Heatmap from data collected while shooting at upper right targets in the Shooting Gallery.

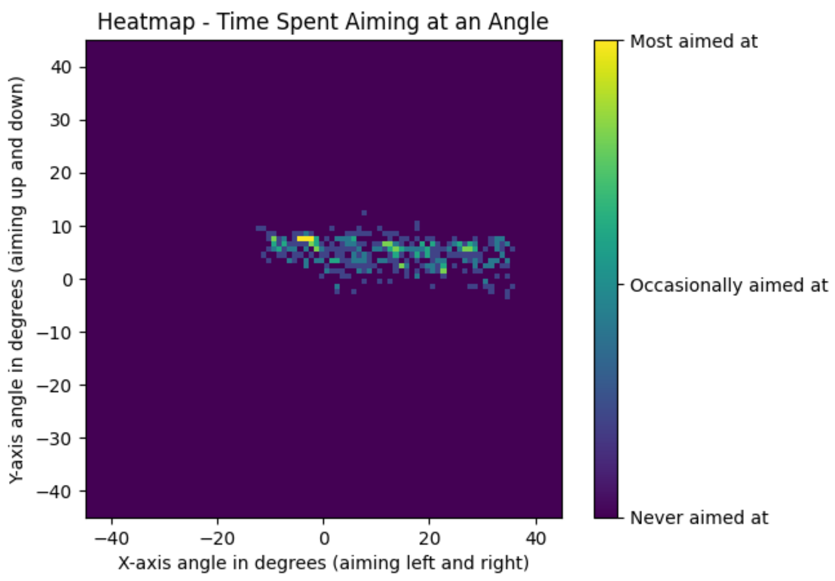


Figure E.3: Heatmap from data collected while shooting at the lowest targets in the Shooting Gallery while having the right hand further from the body to aim.