TU WIEN Informatics

User Navigation Behaviour Using Simultaneous Different Locomotion Techniques in Virtual Reality

Development of an Experimental Platform

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Renate Zhang
Matrikelnummer 11921607

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Ass.PhD Hugo Brument

Wien, 2. August 2023

Renate Zhang	Hugo Brument

TU WIEN Informatics

User Navigation Behaviour Using Simultaneous Different Locomotion Techniques in Virtual Reality

Development of an Experimental Platform

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Renate Zhang
Registration Number 11921607

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Ass.PhD Hugo Brument

Vienna, 2nd August, 2023

Renate Zhang	Hugo Brument

Erklärung zur Verfassung der Arbeit

Renate Zhang

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. August 2023

Renate Zhang

Danksagung

An dieser Stelle möchte ich meinen aufrichtigen Dank an all diejenigen aussprechen, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und vor allem motiviert haben.

Zuallererst möchte ich mich bei meinem Betreuer Hugo Brument bedanken, der meine Bachelorarbeit betreut und begutachtet hat. Ohne seine hilfreichen Anleitungen und seine wertvollen Erfahrungen, sowie sein kontinuierliches und konstruktives Feedback während der gesamten Arbeit, wäre die Arbeit nicht in der Qualität, wie sie heute ist. Weiterhin möchte ich mich dafür bedanken, dass er sich die Zeit genommen hat, den ersten Testlauf für die Studie durchzuführen und mir somit wertvolle Daten zur Verfügung zu stellen.

Ich bin auch außerordentlich dankbar an meine Fakultät, insbesondere an den Forschungsbereich Virtual and Augmented Reality, der mir das gesamte technische Setup zur Verfügung gestellt hat, um meine Arbeit erfolgreich durchzuführen. Sowie gilt ein genereller Dank an die TU Wien, durch deren Schaffung einer förderlichen akademischen Umgebung ich erst das ganze Know-How akkumuliert haben konnte um zu diesen Punkt zu gelangen.

Vor allem möchte ich meinen Eltern danken, die mir das Studium überhaupt ermöglicht haben und mich nicht nur finanziell, sondern auch mental unterstützt haben. An dieser Stelle möchte ich auch einen besonderen Dank an meine Schwester aussprechen, mit der ich immer meine Sorgen teilen konnte und trotz schwierigen Phasen immer an mich geglaubt hat.

Ebenfalls möchte ich meinen Freunden meinen aufrichtigen Dank aussprechen. Ihre Ermutigung und vor allem ihr Verständnis während dieser Arbeit waren eine stützende Motivation für mich. Auch danke ich all jenen Personen, die mein Spiel immer wieder durchprobiert haben, um mir wertvolle Einblicke zu verschaffen und somit wesentlich zum Fortschritt dieser Arbeit beigetragen haben.

Dankeschön!

Renate Zhang

Wien, 21.07.2023

Acknowledgements

At this point, I would like to express my sincere thanks to all those who have supported me during the process of creating this bachelor's thesis and especially constantly motivated me.

First and foremost, I would like to thank my supervisor, Hugo Brument, whose idea sparked the bachelor thesis to begin with and who supervised my bachelor's thesis. Without his helpful guidance, valuable experience, and ongoing constructive feedback throughout the whole process, the thesis would definitely not have reached the quality it holds today. Thank you, for always having an open ear if I proposed different ideas and especially being understanding when things were going slow. Furthermore, I also want to thank him for his research when I needed literature and for taking the time to conduct the first test run for the study, helping me to gain a deeper understanding of the topic and providing me with valuable data.

I am also extremely grateful to my faculty, especially the Virtual and Augmented Reality research area, for providing me with all the necessary technical setup I needed for my work. I would also like to extend my thanks to TU Vienna for creating a supportive academic environment, which enabled me to accumulate the knowledge I needed to reach this point.

Above all, I want to thank my parents, who made my studies possible in the first place and supported me not only financially but also mentally. At this point, I also want to express special thanks to my sister, with whom I could always share my concerns no matter what and who always believed in me and my academic abilities, even during challenging times.

I would also like to extend my sincere thanks to my friends. Their encouragement and particularly their understanding during this work were a motivating force for me. Additionally, I want to thank all those who repeatedly tried out my game, providing me with valuable insights and significantly contributing to the progress of this work.

Thank you!

Renate Zhang

Wien, 21.07.2023

Kurzfassung

Diese Arbeit untersucht den Einfluss von Fortbewegungstechniken - Gehen, Steering und Teleportation - auf das Verhalten von Benutzern in einer virtuellen Umgebung. Das Ziel ist es, ein tieferes Verständnis dafür zu gewinnen, wie diese Fortbewegungsmethoden das Verhalten von Benutzern beeinflussen, wenn ihnen in einer Spielumgebung die Möglichkeit gegeben wird, zwischen ihnen zu wählen. Um dies genauer zu untersuchen wurde ein Spiel entwickelt, dessen Ziel es ist Pilze innerhalb eines festgelegten Zeitrahmens mit Hilfe dieser drei Fortbewegungstechniken zu sammeln.
In dieser Arbeit wird zuerst eine Übersicht und Evaluierung der angeführten Fortbewegungstechnik vorgestellt, gefolgt von einer detaillierten Erklärung der implementierten Spieltechniken. Im Anschluss erfolgt die Datensammlung und -analyse, welche Erkenntnisse über die Wahrnehmung und Nutzung der Fortbewegungsmethoden der Benutzer beleuchtet. Diese erste Ergebnisse der Pilot-Studie deuten darauf hin, dass Steering als bevorzugte Fortbewegungsmethode hervorsticht, begründet wegen der in Verhältnis höheren Kontrolle und Benutzerfreundlichkeit, während die Technik Gehen durch seine Immersion und natürliche Bewegungsführung präferiert wurde. Teleportation wurde in diesem Versuch am wenigsten eingesetzt, vermutlich wegen der erhöhten Disorientation nach der Teleportation.
Im Rahmen des Entwickeln, sowie in der Testphase sind bestimmte Verbesserungen festgestellt worden. So würde eine genauere Erkennung von Navigationszuständen sowie eine optimierte Grenzerkennung ein solche Verbessung darstellen. Des Weiteren könnte der Beachten von bereits plazierten Objekten für die Generierung von Szenen für eine gleichmäßigere Platzierung der Objekten führen.
Die Studie legt im Gesamten betrachtet den Grundstein für weitere Forschung über die Auswirkungen der Nutzung von Fortbewegungstechniken und bietet wertvolle Einblicke zur Verbesserung des Spiels. Zusammenfassend trägt diese Studie zu einem tieferen Verständnis von Fortbewegungsmethoden und ihren Auswirkungen auf virtuelle Erfahrungen bei. Zusätzliche Testläufe mit variierenden Parametern würden weitere Erkenntnisse über den Einfluss von Fortbewegungstechniken auf das Verhalten der Spieler liefern.

Abstract

This thesis explores the impact of having the opportunity to use different locomotion techniques - walking, steering, and teleportation - on user behavior within a virtual environment. The objective is to gain a deeper understanding of how these locomotion methods influence user behavior when provided with the option to choose between them in a game environment. To examine this further, an experimental platform was developed, where players were asked to collect virtual mushrooms within a given time frame using the three given locomotion techniques.
First, the study presents an in-depth overview and evaluation of each implemented locomotion technique, followed by a detailed explanation of the game mechanics. In the following data were gathered and the analysis of the data yielded valuable insights into user perceptions and utilization of the locomotion methods. Initial results showed steering to be the preferred locomotion method, probably due to high user control and ease of use, while walking demonstrated appeal in terms of immersion and natural movement. Teleportation was the least favored in this trial, probably due to the increased disorientation after the usage.
During the process of developing the experimental platform, as well as during the testing phase certain improvements have been detected. Such an enhancement could address challenges in accurately detecting navigation states and enhancing border detection in the freeze-turn method. Additionally, using already placed objects as an additional factor for scene generation could lead to more evenly spaced objects.
Generally speaking, the study lays the groundwork for further research on the implications of locomotion technique usage. Conclusively, this study contributes to a deeper understanding of locomotion methods and their implications for virtual reality systems. Extended user studies with varying parameters would offer more comprehensive insights into the impact of using several locomotion techniques at one on user behavior.

Contents

CHAPTER 1

Introduction

1.1 General Introduction

The simplest definition of Virtual Reality (VR) is immersing users in a computer-generated environment, a perceived reality which is however is not artificially created. A more formal definition might be a methodology for stimulating a user's perceptual system with the aim of recreating natural interaction in a synthetic environment [HMG11]. VR simulates a Virtual Environment (VE) which immerses users to the extent of the feeling of "being there" [BM07]. In a further sense, VR is defined by the illusion of participation in a synthetic environment rather than the mere external observation of such an environment. VR can involve three-dimensional, head-tracked displays and also optional hand and body tracking [Gig93]. However, Head-mounted Display (HMD) is not mandatory required as in Cave Automatic Virtual Environment (CAVE) systems, the environment is back displayed on each side of the CAVE walls, substituting head-mounted displays. In any case, the experience transforms VR into an immersive, multi-sensory experience [HMG11].

The term VR can be characterized based on three properties: presence, interactivity, and immersion [WP02]. The experience of being physically somewhere other than where one is is commonly interpreted as presence [SVS05]. Interaction, which influences presence, refers to the degree to which users may engage with their virtual environment in real-time [Ste00]. Immersion can be described by technological capabilities, which can be objectively measured. An example for this could be vividness, which could be measured by the richness, resolution, or quality of the displays [SVS05]. Immersion could also be characterized by the subjective involvement of the users, such as in cognitive immersion, which users feel when they solve more complex problems [NNS16].

The possibilities of VR depend on the current state of technologies, which is constantly changing and in ongoing development. Nevertheless, researchers have identified several

requirements that virtual reality systems must possess in order to provide users' with spatial and perceptual abilities [GS98]. These characteristics essentially translate to what a user is able to perceive and do in a VR system.

Sensory feedback is one of the requirements. Our perception of physical reality is formed by the information presented to our senses, consequently, the output channels of a VR application correlate to our senses [GS98]. Currently, in virtual reality, the focus of senses can be divided to visual, sound and haptic perception, which correspond to the enabling technologies in order for users can see, hear, and sometimes also feel in a VE.

Another requirement is user input. Humans emit information and interact with their surroundings through the input channels of VR applications. Interaction mostly takes place in form of locomotion, selection and manipulation [LaV17], while communicating information occurs predominantly by voice, gestures and facial expressions. User input is typically enabled by different kinds of tracking, such as position tracking for locomotion [GS98].

Out of the many possible actions one can do in virtual reality, in this thesis the focus lies on the locomotion of the user, which is a fundamental interaction in VR. Locomotion relies on how the player can move from one place to another in a VE, which we will go into further detail in the following.

1.2 Problem Statement

In VR the user has different ways to move in the VE, one has the choice to make the use of different locomotion techniques, if they are provided by the applications. In this thesis, we focus primarily on walking, steering and teleportation and in further sense the documentation of each navigation state. The reason why we focus on these techniques is because they are the mostly implemented, and each one has different properties. Walking provides more vestibular feedback, steering can enable motion without moving in real, and teleportation allows fast motion but can disorient.

Other studies have been made, through in most of that evaluations the techniques have been assessed separately, hence giving the user no choice but to choose one of them during the same trial. Thus, it would be interesting to investigate how users behave when they have the opportunity to navigate with different locomotion techniques as the same time.

In order to study user navigation preferences and behaviors when different locomotion techniques can be used, we designed a VR experimental platform where the goal is to collect as many virtual mushrooms as possible in a certain time limit. Furthermore, the player should do so while avoiding colliding with obstacles in the VE. The number of collected mushrooms and collisions, and also the time left to collect the mushrooms can be seen on the Head-up-Display (HUD) of the headset. The player can freely choose the locomotion technique, either by walking, which translates the real-life movements of the user to the environment. Another option is to steer, in this case, the player can use a button on the controller to move forward, while the direction can be controlled

by rotation of the headset itself. The last locomotion technique in our experimental platform is teleportation, when pressing the trigger of the controller, a laser will aim at the designated destination which will transport the player to on release. Subsequently, the combination of each locomotion technique is also feasible, as you can walk while you steer, resulting in a faster walking motion.

This development of the experimental platform is the focus of this thesis, along with the preliminary features for the actual study, such as the implementation of locomotion techniques, design of the VE, design of the procedure, detection of the navigation state, and the data recording. Additionally, a pilot study is conducted, in which first data are gathered and the environment is tested. In the actual user study, which goes beyond the scope of this thesis, the number of each locomotion technique is going to be measured, and the probability of each method is calculated. The essential goal of the study is to find out whether it is possible to predict the use of each locomotion technique beforehand and also to assess the reasoning behind each choice.

CHAPTER 2

Related Work

In immersive VR applications, 3D visual display devices and 3D input devices are combined to create the illusion of the user being inside a virtual world [LaV17]. The way the player can be immersive in this virtual world and move with seemingly no borders is fundamental in establishing the illusion of being in a virtual world, thus demonstrating the importance of the implementation of various locomotion techniques.

2.1 Locomotion Techniques

Similar to LaViola, in our thesis we also classified the techniques based on their overall interaction metaphor, which is easier comprehensible from the view of the user. In literature there exists further a great multitude of different locomotion techniques, which require more or less body movements of the user. Travelling by manipulation, for instance, manipulates the vision or the entire virtual world by using metaphors based on hand manipulation [LaV17]. However, this technique and others are not the scope of our thesis and we will focus only on walking, steering and teleportation.

2.1.1 Walking

The most natural way a player could move in a VE is walking, as users do not have to think about it, enhancing the sense of presence and improving the sense of scale [LaV17]. But in most cases, it is not possible to use real walking as the only travel technique due to the restricted size of the tracked area. To solve this issue researchers have developed walking metaphors, which have been classified based on the human gait. In our thesis we will focus on the full gait, as its name suggests a full gait cycle is involved, as opposed to the partial gait, where you e.g. walk in place. Full gait techniques can be further divided into real walking, redirected walking, and scaled walking. In this thesis, we will make use of the first technique and also overt redirection for our virtual environment [LaV17].

Real Walking

Real walking is the most natural and straightforward full gait technique since it provides cues to help the user to understand the size of the environment. Furthermore, researchers have observed that real walking leads to a higher level of spatial knowledge in a more complex environment [CGBL98]. As mentioned before however, this approach cannot always be carried out due to the restrictions of the area. Naturally, the tracked area has to be bigger than the virtual environment and has to be free of obstacles. This could consequently raise issues with the choice of the used devices as a result of the necessary cabling, even though there are nowadays more and more wireless solutions for HMDs.

Real walking can be therefore used in a limited area in the majority of VR applications, but other methods are required to access other areas of the environment. However, researchers found out that when users are given the choice between walking and using a virtual locomotion technique, they rapidly learn to solely use the virtual option as it takes less effort [LaV17].

Overt Redirection

The key benefit of using overt techniques is that we can alert users when they approach the work space's limits, increasing safety compared to subtle techniques. These methods aim to "reset" the user's position (i.e., place them in the middle of the workspace) without disrupting their sense of immersion and in a manner so that the positioning of objects in the virtual environment remains the same [WNR+07].

In his study, Williams presents three strategies for "resetting" users who reach the physical boundaries of the tracking system [WNR+07].
When approaching the border, in the 'Freeze-Backup' approach the virtual world freezes in a sense that the position of the user is not updated anymore also if he moves in the real world. The player is then indicated to take some steps backwards while the virtual world remains frozen. If enough steps were taken, the system informs the user to stop and the virtual world unfreezes and the user can continue.
When using the '2:1' method, the tracked is asked to turn until they completed a full 360° turn in the virtual world. The rotational angle is multiplied by two, so that the user rotates 180° in the real world, but completes a 360° turn in the VE.

In this thesis we will use 'Freeze Turn'. Similarly to 'Freeze-Backup', in this technique the VE freezes as well, however the user is instructed to turn around if the limits of the tracking area is reached. As a result, the user is facing away from the tracking boundaries and has enough space to continue. The objective here is to persuade the user to turn in the first place. Researchers have looked into a variety of strategies for helping the user turn, including location-based visual cues [PFW09].

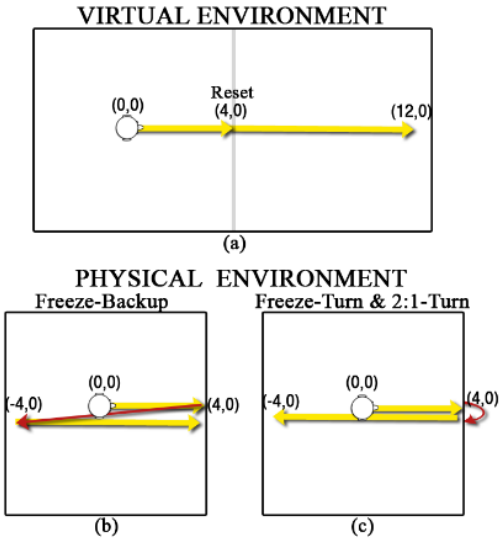


Figure 2.1: The path a.) in the virtual environment, b.) Freeze-Backup and c.) Freeze-Turn and 2:1 Turn[WNR+07]

2.1.2 Steering

Even though a lot of research has been done on natural locomotion techniques, most 3D applications adopt some form of virtual travel technique. When it comes to virtual techniques the by far most used virtual technique is steering, which describes how the user continuously controls the direction of travel. Most frequently, either spatial interactions or physical steering props (e.g. simulating a vehicle) are used to specify the movement direction. In our thesis we will focus on spatial interactions for steering [LaV17].

Spatial Steering Techniques

Through spatial steering techniques, the user can steer and control the direction of travel by adjusting the tracking device's orientation. The highest level of control for users is provided using steering techniques, which are relatively simple to learn. Spatial steering techniques can be categorised by how the direction is specified, such as gaze-directed steering, which will be implemented in our experimental platform and is also the most common steering technique [KBH01]. Simply said, using this technique allows the user to move in the direction of his gaze, which is obtained from the orientation of a head device in a tracked environment.
The direction of travel can specified by other body segments as well. It is also possible to do steering by controlling the direction with the user's torso or hand-directed steering [LaV17]. Furthermore, there is also a variety of approaches on how to update speed. Discrete control allows the user to choose from a fixed set of speeds by e.g. pressing a button while continuous control allows the user to select the navigation speed over a

continuous scale, such as using a joystick axis with a range between minimal and maximal speed [BOMA20]. Also, speed can be increased in a linear or rotational approach, the latter being mostly used for redirecting [SHCV+18].

We will implement discrete gaze-directed steering, therefore we will discuss this approach in more detail. From the perspective of the user, gaze-directed steering is quite easy to understand and intuitive. Also, the hardware requirements are fairly moderate, as only a head tracker and a button, to stop and start the motion, is needed. The issue with gaze-directed steering however is the fact that direction of gaze and travel is linked together, hence users are not able to look in one direction while going to another. Even though this appear to be a minor concern, it should be considered how often people look in a direction other in which they are moving, e.g. while walking or driving in the real world [LaV17].

2.1.3 Selection-Based Travel Metaphors

Another significant category of travel metaphors relies on the user directly choosing the destination to travel to, called selection-based travel metaphors. The user is not constantly required to consider the details of travel using these selection-based travel metaphors, which often simplify the travel process. Instead, the user sets the desired travel parameters before and then passes the control of the actual movement to the travel technique itself. Although these methods are not the most natural, they are typically relatively simple to comprehend and use [LaV17].

Ray-Casting Selection Technique

There are many ways to specify the target of travel, one of them is using a ray to select the destination. In our thesis we will use a laser beam that the user can control, the endpoint of the beam sets the target of travel. After confirming the destination the user will be transported to the selected target instantly.
However, there are some side notes regarding teleportation. A study found that teleporting a user to its target instantly decreases its spatial orientation significantly [BH99]. Hence if spatial orientation is important continuously moving the user from start to end destination should be favored. Contrarily, a continuous movement which is not under the user's control can cause cybersickness. To compromise both issues, it is often recommended to not teleport the player right away, but to move the user very fast in the virtual environment to its specified target. Using this approach gives the user enough spatial knowledge to comprehend how they are transported, but due to the short time also unlikely to make them feel sick [LaV17].
It is noted that in our thesis the user will be teleported instantly and spatial disorientation is taken into account. However, the mentioned approach above could be added as an enhancement in further developments.

2.2 Evaluation of Locomotion Techniques

In the previous sections, we went through a variety of locomotion methods in great detail. The issue is now how to assess and put them in comparison. What framework has to be created in order to evaluate different locomotion techniques? Which requirements have to be met in order to compare for example walking with steering?

In this section we will discuss what is needed in order to assess the locomotion techniques of walking, steering and teleportation in our setting, starting from the design of the environment itself to the factors we want to measure. Various locomotion techniques have obviously been evaluated before, therefore we will refer to relevant papers and summarise their findings. Based on these papers we will also make assumptions about which locomotion technique might prevail under what conditions in our context.

2.2.1 Requirements

In order to be able to conduct a study to evaluate and compare locomotion techniques certain things have to be assumed, such as requiring certain conditions in regards of the evaluation platform. Therefore, in this section we will cover what essentially is needed to evaluate locomotion techniques in general and also state what our approach is in this study.

Design of VR Environment

The virtual environment plays an important role in evaluation process as the surroundings have a significant impact on their performance. In most studies, researchers chose to embed the tasks for the users to interact in an experimental platform. Depending on the locomotion task and study focus, the design of the VR environment can vary. Typically, environments are categorized into outdoor and indoor settings [KR21], with varying levels of realism. For tasks that do not require high detail, low-poly objects are often chosen to minimize distractions and unnecessary inputs for users [CCBWS18]. On the other hand, a higher degree of realism in the environment contributes to a more immersive experience.

A common practice in virtual reality experiments is to design the platform as a maze to stress test participants and assess their orientation in complex VE [SFR$^+$10]. The size of these platforms can vary, ranging from room-sized VEs [LLS18] to almost open-world settings [CA17]. In most cases, the designed environment remains rigid and constant throughout the study. However, there is an alternative approach where the environment is generated anew after each trial, taking into account that the virtual environment can influence the user's navigation [RL06]. By creating an infinite number of virtual environments, each setting can contain varying amounts of visual information, which can influence the user's navigation behavior. Subsequently, the amount of information provided by the environment can vary from experiment to experiment, hence the outcome from the same task varies depending on the different environment. To ensure that the

environment does not provide a familiarity even after several runs and therefore giving the user an unwanted advantage, thus manipulating the results, it is an option to generate the scene from scratch every time. This approach helps maintain a level playing field and avoids confounding factors that may arise from familiarity with the environment.

Locomotion Task

In order to create a standardized and fair comparison between different locomotion techniques, researchers commonly conduct experiments involving users in navigation tasks [KR21]. These navigation tasks typically involve instructing participants to navigate through a virtual environment with the objective of locating specific items or destinations [CCBWS18]. Additionally, experiments may include scenarios where participants are initially provided with a map displaying the location of their wayfinding target. This requires participants to move through the virtual environment towards previously seen targets but with a rough estimation of their route [CTHP16]. Furthermore, users can be assigned locomotion tasks after their exploration of the virtual world. After each trial, participants could be presented with a VR pointing task to assess their mental map building. For instance, users could view pictures of the virtual world and then indicate the spot they believe the photographer was standing at the time [SSH20].

Similarly to the environment, the type of task can also modify the performance and behavior of the user. For instance, the degrees of freedom necessary for motion in the chosen locomotion techniques might influence how the player will travel in the task [WKFK18]. Moreover, the number of provided locomotion techniques itself can vary in different experiments. Previous literature often focused on testing one locomotion technique at a time and divided participants into separate test groups for each technique [LLS18]. This approach allows researchers to assess the performance and user experience of each locomotion method independently, facilitating a detailed comparison and analysis of the results.

Locomotion Measurements

By employing navigation tasks in experiments, researchers can objectively evaluate the performance and user experience associated with different locomotion techniques. However, in order to systematic evaluate locomotion strategies, it is essential to measure user performance. This is often achieved by collecting observational data during the experiment [KR21], utilizing objective metrics such as time, distance, or participant positions. Additionally, variables related to the locomotion task, such as the success rate of finding the target [CTHP16] or the number of times the target is visited [CCBWS18], are tracked and analyzed.

In the majority of studies, subjective perception of participants is assessed using questionnaires. These questionnaires inquire about various aspects, such as perceived naturalness, motion sickness, and subjective preferences regarding locomotion techniques. Additionally, they aim to gain insights into participants' gaming experience and if present

to which extent. In terms of cybersickness, there is a variety of questionnaires which focuses on evaluating its symptoms [KLA+23]. Specialized questionnaires, such as the Simulator Sickness Questionnaire (SSQ) [KLBL93], was originally designed to assess simulator sickness in aviators, but are nowadays frequently used to measure cybersickness induced by VR exposure [SSB+20]. However, it is worth noting that the SSQ might not adequately capture cybersickness symptoms experienced in VR, as a recent study indicated its limitations in evaluating VR-induced cybersickness [SB20]. To address this, researchers have developed a variant of the SSQ known as the VR Sickness Questionnaire (VRSQ). The VRSQ was designed to isolate the relevant items from the SSQ specifically related to cybersickness [KPCC18]. Nevertheless, both questionnaires examine symptoms after VR exposure and not during, thus producing results that are difficult to interpret [KLA+23] and could eventually lead to false conclusions.

2.2.2 Examples of Evaluations of VR Locomotion Techniques

There are several works that have dealt with evaluating different locomotion techniques, which we will summarise in the following.

As mentioned earlier real walking is the most natural and intuitive locomotion technique as it provides feedback that help the user understand the size of the environment and avoid getting sick, and promotes spatial understanding [LaV17]. However, **Suma, et al** [SFR+10] showed that in a complex virtual environment, such as a 3D maze that required lots of turns, walking increased simulator sickness compared to head or gaze steering. If the purpose is to make decisions based on information seen in the virtual world, then their results show that virtual travel is a suitable option in a complex landscape. Yet, actual walking has benefits over controller-based virtual travel methods for applications that call for quick, effective navigation or travel that closely reflects real-world behaviour.

According to the experiences of **LaViola, et al.** [LaV17], when users have the option to both use real walking and a virtual locomotion technique, they tend to use only the virtual one because it requires less effort. There might be the assumption that this would even be more the case in large scaled tracking areas. However, in the research of **Sayyad, Sta and Höllerer** [SSH20], which setting consists of a large physical space, the usage of walking is favoured. The study explored the impact of natural walking in large physical spaces on presence and user preference in comparison to teleportation. Results demonstrate that the majority of participants preferred walking and that teleportation causes much more self-reported simulator nausea. The data also indicate a tendency toward increasing self-reported walking presence.

With regard to performance **Ruddle and Lessels** conducted multiple studies to compare walking to other techniques for completing various tasks in a virtual environment. In one study they investigated how body-based information of movement helped participants to fulfill a navigational search task, which is finding targets hidden inside boxes in a room-sized space. In contrast to the other locomotion techniques, walking while using all of your body's information produced higher search results [RL09].

In a different trial, they verified these findings and found that walking produced more accurate search performance than head steering or joystick control with monitor [RVB11]. Finally, in a different experiment, users were instructed to travel a 24 m-long route by repeatedly following a virtual arrow in a virtual environment with corridors. They demonstrated that walking resulted in a quicker task completion time and fewer collisions than head steering [RVB13].

Furthermore, **Christou and Aristidou** [CA17] studied the ability to navigate from one point to another in a virtual environment using gaze-directed steering, pointing steering, and teleport. Similar to our locomotion task, the participants were asked the collect tokens, which as a result tested their spatial capacity. Results of the study showed that compared to teleporting, the two steering techniques caused higher degrees of cybersickness. Teleporting was more efficient in terms of travel speed. Unexpectedly, it also allowed users to finish their tasks equally as well, suggesting that user disorientation was not a significant problem. The main drawback of the teleport method was that it was more likely that not all tokens were found due to the tendency to miss detail. However, they observed that some participants compensated for this issue by making rapid yet small teleport leaps.

For our study, these findings could be applied as well. As seen in the studies of Suma [SFR+10] and Sayad, et al. [SSH20] the preference to walking depends on the type of virtual environment given. But still, walking remains the most effective locomotion technique in terms of performance [RL09], which may also be the case when users are attempting to get precisely to the targets in our locomotion task. Based on Christou and Aristidou's study [CA17] it could be suggested that users might not use steering technique due to cybersickness. Even though cybersickness is not a factor we actively assess, still, cybersickness could influence the choice of locomotion technique for our users and thus should be taken into account to some degree. Moreover, it can be also concluded that when users primarily use teleportation they might have issues finding all mushrooms, as seen by the study of Christou and Aristidou [CA17].

CHAPTER 3

Methods and Implementation

In this chapter we will go into further detail about the framework of the environment, such as which devices are used, and also the implementation and thought process of each feature. Methods such as locomotion techniques and scene generation will also be explained more specifically. In conclusion, this chapter aims to give the reader some insights regarding the technical aspects of the development.

3.1 Software and Hardware

The virtual reality setup consists of an HTC Vive HMD Figure 3.1[1] featuring a 1080x1200 pixel resolution for each eye and a 110-degree field of view. Both of the HMD's screens have a 90 Hz refresh rate. User interaction was facilitated using two single handheld controllers Figure 3.2[2], which also have a virtual representation in the virtual environment. The head and controller tracking system uses a set of base stations that were spaced roughly 2 meters apart and at the opposite ends of the tracking region during the development.

Furthermore, the VE was rendered using a Windows 10 workstation with Intel Core i5-12600K 3.7GHz and 32GB RAM with NVIDIA GeForce RTX 3060 and 12GB onboard memory. For creating the experiment setting with terrain and obstacles, the Unity3D game engine was used, while custom C# scripts to handle the game's mechanics. Additionally, the OpenXR[3] plugin by Khrones Group is used, which is a newer standard that enables applications to work across multiple devices and platforms and that different hardware is

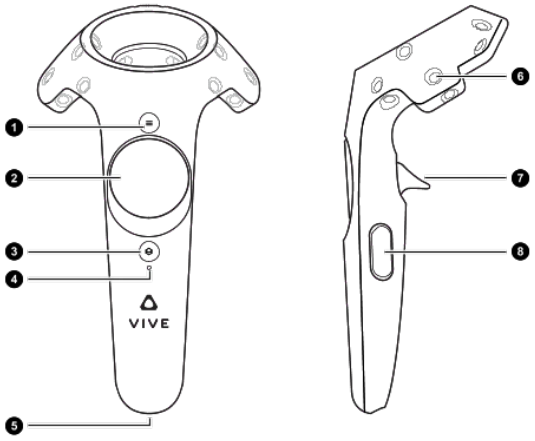
[1] https://www.vive.com/de/product/vive/, accessed April 7, 2023

[2] https://www.vive.com/us/support/vive/category_howto/about-the-controllers.html, accessed April 7, 2023

[3] https://www.khronos.org/openxr/, accessed April 7, 2023

interpreted the same way. We also used the XR Interaction Toolkit[4], which is a high-level, component-based, interaction system for creating VR and AR experiences and handles some of the locomotion techniques.

Figure 3.1: HTC Vive headset with the included two controllers and two base stations [5]



1: Menu button
2: Trackpad
3: System button
4: Status light
5: Micro-USB port
6: Tracking sensor
7: Trigger
8: Grip button

Figure 3.2: HTC vive controllers and its input buttons[6]

[4] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/index.html, accessed April 7, 2023

[5] https://www.amazon.com/HTC-VIVE-Virtual-Reality-System-pc/dp/B00VF5NT4I, accessed June 23, 2023

[6] https://www.vive.com/us/support/vive/category_howto/about-the-controllers.html, accessed June 23, 2023

3.2 Design of the Platform

In our case, a forest setting serves as the virtual environment, which includes several obstacles such as trees and rocks. The environment is entirely flat, and it should not be possible to get on top of any of the obstacles. The forest is filled with 3D low-poly objects, which all are free assets from the Unity Asset store [7,8,9]. The VE consists of a 10x10m field, which is enclosed by wood fences, bordering the walkable virtual area. When the player enters in the VE, the scene is generated anew and is filled with rocks, trees, and bushes, which serve as obstacles. The player is situated in the center of the tracking area, which is marked by a yellow square. HUD provides instructions so the player can start the experiment and gets additional information regarding the current status, such as the number of collisions, the timer, and the amount of gathered red and brown mushrooms, see Figure 3.3. The mushrooms, which are situated on top of tree stumps for easier picking, are only generated once the user is calibrated and ready.

Figure 3.3: Start screen when entering the VE, the HUD displays start instructions and status

3.3 Procedure

3.3.1 Locomotion Task

The locomotion task and objective of our experimental platform is to collect two different kinds of mushrooms, red and brown ones, as fast as possible and without running into

[7] https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153, accessed July 4, 2023

[8] https://assetstore.unity.com/packages/3d/vegetation/trees/free-trees-103208, accessed July 4, 2023

[9] https://assetstore.unity.com/packages/3d/props/exterior/low-poly-fence-pack-61661, accessed July 4, 2023

any obstacles, performed from a first-person perspective. The player's HUD displays the total amount of picked mushrooms and the number of collisions, however other than that the player does not receive any penalties when colliding. The player can slide through obstacles, which could help him save some time, but it is stated before the start of the experiment that collisions should be avoided.

The user can freely choose either between walking, steering, or teleporting to fulfill the task. It is possible to complete the task with only one locomotion technique, while this is quite simple to do so with steering or teleporting there might be difficulties with walking due to the size restrictions of the tracked area. If this case occurs the freeze-turn method is applied, in which the player has to stop the game-flow in order to turn. A note here, during this process the timer will not be stopped, which could hinder the locomotion task. In our experimental platform we make use of gaze-directed steering, thus the direction of travel is controlled by the direction of the gaze. The user utilizes the controller and by pressing a button he will move forward. In our experimental platform it is not possible to steer to other directions, e.g. left, right, through the buttons, so the player always has to look in the direction he wants to move forward to. Teleportation is also executed through the controller, if the player slightly presses the trigger a curved beam will shoot out of the controller in the virtual world, which endpoint on the ground is the designated destination. By pressing the trigger firmly the player confirms the action and teleports the player to the spot. It is not possible to teleport on or to objects, as the teleportation will not be carried out. At the very beginning, the player is asked to move to the platform's centre in order to guarantee that they always start at the same spot and do not have any unfair advantages. The location is indicated by a yellow square, and once the player has positioned themselves correctly, they can start by clicking a button that also starts the timer.

3.3.2 Implementation

The core part of the experiment is encapsulated in a *Procedure* class. In this script, the main game logic is implemented. If the player enters the VE, the scene is already generated, which is handled by its own class *Scene Generation* 3.8.

The players have to go through the *Calibration* 3.6 of their position first, only afterwards the experiment starts which consequently turns the mushrooms visible and also triggers the *Data Recording* 3.9, as seen in Figure 3.4. If the player finds mushrooms, the *Mushroom Counter* 3.5 increases respectively to the mushroom type. Also, collisions against the objects of the virtual forest are detected, which as a result also increases the *Collision Counter* 3.5. There are two conditions on which the experiment can end, either the player finds every mushroom, which results in a win, or the time runs out before that. In both cases the player is unable to pick up any mushrooms afterward anymore and cannot use any virtual locomotion techniques, additionally, the data recording stops.
Furthermore, it is also possible to restart the process after the experiment has finished which is executed by pressing the space bar. This should be a simplification for the supervisor, so for generating the same setting one has not close the application every

time. When restarting the experiment, the time, all counters, and also the corresponding scripts are reset, the player is set to active again and a new scene is generated. The whole procedure of the experiment is depicted in Figure 3.4.

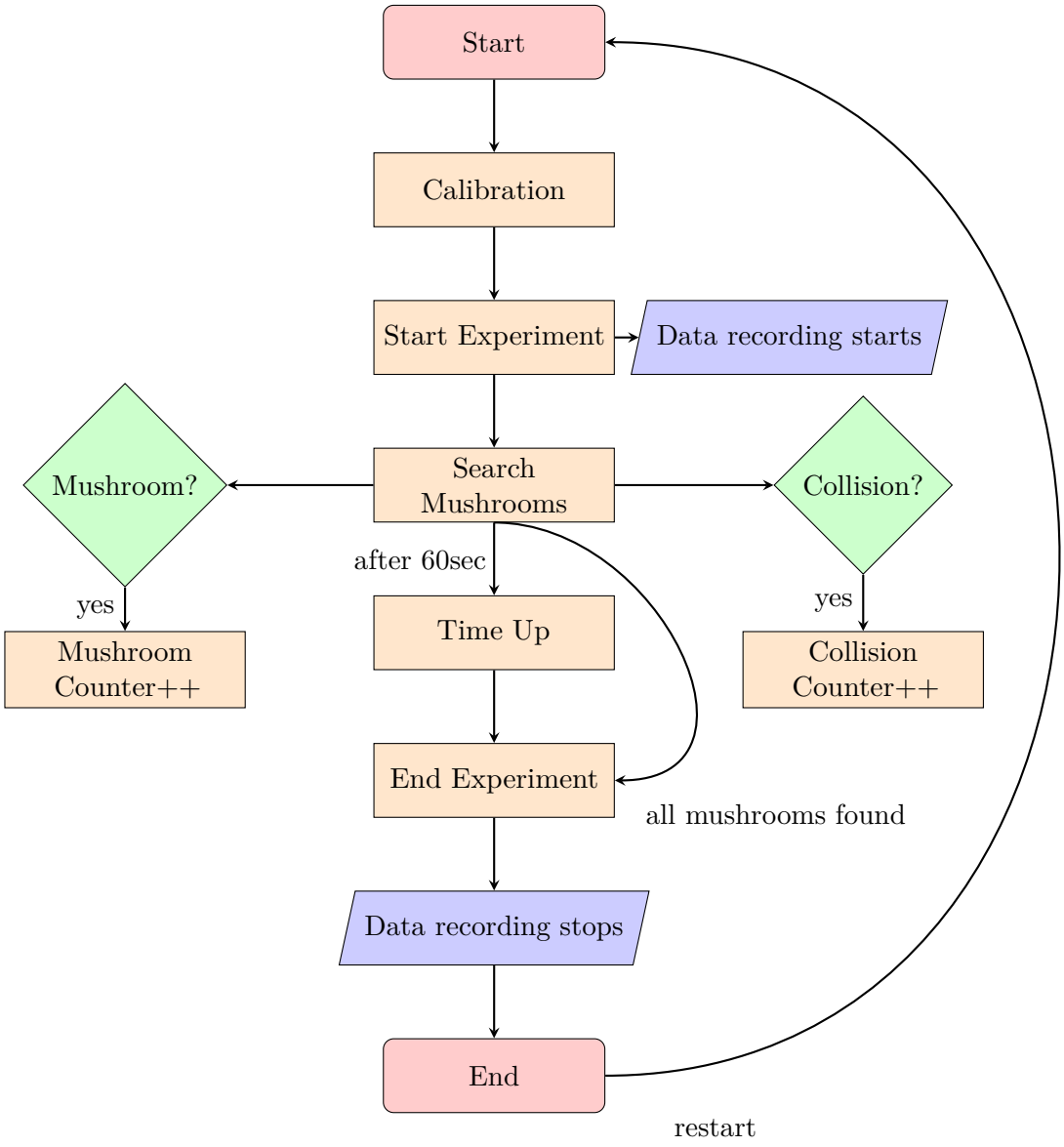


Figure 3.4: Procedure of the experiment

3.4 Interactions

All interactions of the user are encapsulated in *OpenXR* GameObject, which further holds representative GameObjects for the camera and interactive controllers, see Figure 3.5. The *Main Camera* GameObject represents the camera and handles HMD motions, such as position and orientation. Generally, for hand-based interactions we are using the *XR Controller* Component provided by the XR Interaction Toolkit, the default XR Interaction Controller preset of the asset samples has been used to configure the component. Essentially, the component interprets feature values of the controller into XR Interaction states[10], while in another Component *Input Action Manager*[11] the actual input is managed.

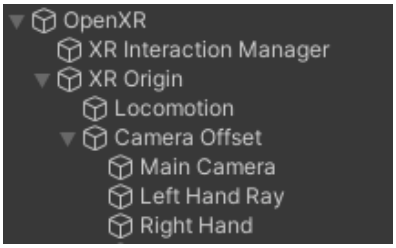


Figure 3.5: Interaction GameObjects in Unity

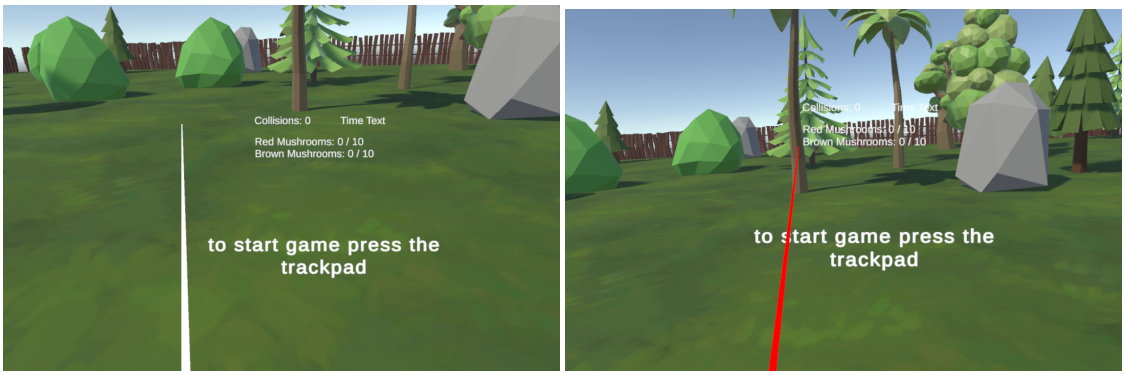
3.4.1 Locomotion Techniques

Walking refers, as the name suggests, to the player's natural walking state and is enabled by the tracking of the HTC Vive Headset. In the Unity project, the HMD is represented by the *Main Camera* GameObject, in which the *Tracked Pose Driver*[12] component of Unity is handling the tracking. This component applies the current values of a tracked device to the transform of the GameObject. In this way when wearing the headset the movement of the player will automatically be transformed into the virtual world and no additional action has to be taken.
If the user reaches the boundaries of the tracked area, the freeze-turn process activates. The player is then requested to turn to continue the experiment and has enough space to further use walking as a locomotion technique, see in section 3.4.3.

Teleportation attributes to the user's ability to change its position from one point and is carried out through the left controller, which is represented by the *Left Hand Ray* GameObject. The *XR Ray Interactor*[13] component is used for interacting with

[10] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-controller-action-based.html`, accessed April 13, 2023
[11] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/input-action-manager.html`, accessed April 13, 2023
[12] `https://docs.unity3d.com/2018.2/Documentation/ScriptReference/SpatialTracking.TrackedPoseDriver.html`, accessed May 4, 2023
[13] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-ray-interactor.html`, accessed May 4, 2023

interactable at a distance. The interaction is handled via the selection of the grip button, see Figure 3.2, on the controller with the visual indication by ray casts that update the valid targets for the interactor.
The base ground serves as our interactable and is called *Teleportation Area*[14] which is also handled by its own component provided by the XR Toolkit. Also, we utilize the *Locomotion System*[15] component, this component controls access to the XR Origin. This approach enforces that the XR Origin can be moved by just only one locomotion provider at a time, which is especially of concern if you have several components that are trying to move the XR Origin at the same.
Altogether, the left controller emits a constant ray that remains active at all times. If the intended teleportation destination is not viable, the ray will display a red hue as seen on Figure 3.6b. On the other hand, if the intended location is feasible for teleportation, the ray will appear white, see Figure 3.6a, thereby indicating that the user can activate the process by using the trigger button on the controller. It is important to note that teleportation is only possible on a flat surface and not through any objects or structures. Moreover, it should be noted that during the teleportation process, no collision or any other form of physical contact will occur.



(a) Selecting a valid destination (b) Destination not available

Figure 3.6: Teleportation ray implementation

Steering describes the gaze-directed steering motion of the player, similar to how it is frequently used in video games, the player's position follows the player's gaze (defined by the forward vector of the HMD) when the trigger is pressed. This locomotion technique is handled by the right controller, the *Right Hand* GameObject.
In the assets folder, you can find the *XRI Default Input Actions*[16], where you can locate the custom created actions for the steering, referred as body-based steering. This action

[14] /urlhttps://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/teleportation-area.html, accessed May 4, 2023
[15] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/locomotion-system.html`, accessed May 4, 2023
[16] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@1.0/manual/samples.html`, accessed May 21, 2023

is bound to the *triggerPressed* path of the HTC Vive Controller, which means when pressing the trigger of the controller the corresponding script will be activated.
The steering feature is encapsulated in the *BodyBasedSteering* script. In this script, the input is handled by a reference, which then initiates the actual steering function. In the main function the calculation for the new position is as follows:

$$Vector3\ deltaSteering = camera.forward * Vector3(1, 0, 1) \tag{3.1}$$

$$player.position\ +=\ deltaSteering * speed * deltaTime \tag{3.2}$$

The player's position will be updated according to their gaze when pressing the steering button, therefore a forward-facing vector of the player's camera is needed. This value *camera.forward* is retrieved by the *Main Camera* component which represents the HMD of the user.
To prevent the player from floating away when they look up at the sky, the player's forward-facing vector is multiplied by the second vector *Vec3(1,0,1)* in order to equalize the y-axis value. Hence, this results into the auxiliary variable *deltaSteering*, a 3D vector which consists of the forward-facing 3D vector of the HMD *camera.forward* and the equalizing 3D vector *Vec3(1,0,1)*. The coordinates are 3D vector representations of the position in the world coordinates.
The player's position is then updated by adding the deltaSteering value corresponding to the predetermined speed, which in our case has been set to 2, and time.
The script is attached to the *XR Origin* component, where the reference, the camera, the speed, and the player's component for the position is specified. The previously mentioned custom-created action is connected to this reference. This enables the connection between the component with the required inputs, the relevant controller trigger, and the script with the specified calculations.

3.4.2 Picking up Mushrooms

Interacting with the mushrooms is also done with the right controller, the *Right Hand* GameObject, via the *XR Direct Interactor* component. The mushroom is picked up when pressing the grip button, see Figure 3.2, of the controller and automatically disappears on release. Similar to steering, picking up mushrooms is also handled with a custom-created action through the *XRI Default Input Actions*.
All mushrooms are *Grab Interactables* [17], which are components that hook into in the interaction system *XRInteractionManager* to allow basic "grab" functionality. Also, the interactable has the ability to attach itself to the interactor, which is in our case the right hand, and can follow the interactor around. To make the mushroom disappear after release, the *CountMushroom* script is attached to the right-hand component, and besides keeping track of the counter, the script additionally set the interactable mushroom

[17] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-grab-interactable.html`, accessed May 21, 2023

inactive, thus making it invisible. This method is attached to *Interactor Events* [18], when the select is exited the *PickUp* method of the *CountMushroom* script is called and therefore the grabbed mushroom is set inactive. Furthermore, the *Right Hand* GameObject needs a *SphereCollider*[19] component to act as a trigger for the grab interactable. It is also noted that we are using two different kind of mushrooms, a brown and red one. However, besides the visual differentiation and having a respective counter each, the functionality remains the same for both mushrooms.

Figure 3.7: Grabbing a mushroom with the right hand interactor

3.4.3 Freeze Turn

Naturally, there will be instances when the tracked area is smaller in size than the virtual environment. If the player continues on completing the given task only through walking, we employed the freeze-turn procedure in our application. Whenever the user is facing the borders of the tracked area the environment freezes and the user is requested to do a half-turn. The turn is completed if the player is gazing at a sphere, which is situated in the center of the room, thus urging the player to make a turn. When the user is looking at the sphere the virtual environment unfreezes. It is worth highlighting, that in this scenario the virtual world did not change, hence the user can continue the experiment as usual. As mentioned before, essentially in this stop-and-go technique the player will stop its natural walking path when colliding against the borders of the tracked area to perform the necessary turning motion and can proceed afterward [LaV17].

The freeze-turn procedure is handled by the *FreezeBackCoroutine* Script. Before the first frame update the *Start()* method is executed, which retrieves the boundaries of the SteamVR workspace. It is worth noting, that in our application this is only possible

[18] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/interactor-events.html`, accessed May 21, 2023

[19] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-grab-interactable.html`, accessed May 21, 2023

Figure 3.8: Freeze turn environment

for square-shaped tracked areas so far. The *Update()* method is called once per frame after the *Start()* function and checks whether the player is getting too close to the border of the tracked area while not standing with the back to the boundaries. If this occurs, a coroutine starts and the freeze-backup procedure is triggered. When the *FreezeBack* function is invoked, the entire scene is deactivated, thus the scene remains unchanged after the turn and the user can continue on his planned course. After inactivating the scene, a sphere is created behind the user which acts as a visual cue for the player as an indication where to look. The user is then asked to perform a half-turn and look at the sphere, if the angle between the camera forward vector and the sphere is near to zero, the half-turn is accomplished. The sphere is destroyed afterwards and the scene is set active again. The scene remained the same, however, as the player turns in the real world, they also turn in the virtual environment as well. To cancel out the virtual half-turn, we rotate the player 180 degrees in the VE, putting him in the same position as before the procedure. After finishing the freeze-turn, the method can be called anew if the user is facing the boundaries again.

In the first iteration of the freeze-back implementation, the *FreezeBack* function is called in every *Update()* method anew, thus creating a new sphere in every frame. To make the procedure more efficient, for this implementation, we chose to use a coroutine[20], which allows you to distribute tasks across multiple frames. A coroutine is a method in Unity that can pause its execution and return control, but then resume where it left off on the subsequent frame. To be more specific, a coroutine is a method that has an IEnumerator return type and a yield return statement anywhere in its body. The yield return is the point at which execution is paused and resumed in the next frame. To start a coroutine you have to use the *StartCoroutine* function. In our case, to set the coroutine running at first we call *StartCoroutine("FreezeBack")*. Afterward, in the *IEnumerator FreezeBack()* the angle is checked and consequently sets a breakpoint. The procedure only continues if the angle is the required size, which is handled by our yield statement.

[20] `https://docs.unity3d.com/Manual/Coroutines.html`, accessed June 6, 2023

3.5 Count Mushroom and Collision

3.5.1 Count Collision

For detecting the collision between the player and the obstacles in the environment, we make use of the provided colliders of Unity and its corresponding *OnTriggerEnter*[21] function. When one GameObject collides with another GameObject, Unity invokes *OnTriggerEnter*. Registering a collision require both GameObjects to have a Collider component, whereas one has to enable *Collider.isTrigger* and contain a Rigidbody. It should be pointed out that if both GameObjects have *Collider.isTrigger* enabled, the collision will not be registered, as it is also the case if both GameObjects do not have a *Rigidbody* component.
The implementation is encapsulated in the *CountCollisions* script and is attached to the *Main Camera* GameObject, which represents the headset in the Unity project. When *OnTriggerEnter* is called in during the experiment, two conditions are checked to ensure that the collision counter is not increased: whether the experiment is still ongoing, as no collision should be added after the timer runs out, or if the player picked up any mushrooms, as picking up mushrooms should not penalize the player.

3.5.2 Count Mushroom

Counting the number of picked mushrooms works a little bit differently. The inner workings of counting mushrooms are encapsulated in the *CountMushroom* script and added as a component to the *Right Hand* GameObject. We utilize the functions used for picking up mushrooms, see subsection 3.4.2. Using the *XR Direct Interactor*, we added the *PickUp* function of the script to the *Select Exited* interactor events. This concludes that every time the right hand is selecting something and is exiting the select mode, which means the person is releasing the button, *PickUp* is called. Following a check to verify if the experiment is still running, all valid interactable that are selected are examined to see if they are mushrooms by comparing the tag. If the tag matches, the corresponding counter increases, and the mushroom is set to inactive to make it invisible, as previously discussed.

3.6 Calibration

Calibration is accomplished by instructing the user to stand in the center of the experimental platform, which is marked by a yellow square on the ground, see section 3.6. Calibration is required to ensure that each participant begins the experiment from the same point, giving an equal foundation for additional data processing while also eliminating any unfair advantages.

[21] `https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html`, accessed June 6, 2023

In the *Procedure* script the position of the player is verified through a simple comparison of the current position of the player and the marked spot. If they are not yet at the designated place, they are instructed to go to the yellow mark of the play area, otherwise, the trial does not start. Once they are at the correct location the player can start the experiment by pressing the trackpad of their controller. This is handled by custom input actions, similar to how it is implemented with the steering and picking up mushrooms, see subsection 3.4.1 and subsection 3.4.2 respectively. If the experiment starts the script for the locomotion techniques are set to active, only then does the player have the ability to steer and use teleportation.

Figure 3.9: Yellow square to indicate calibration spot

3.7 Navigation State

In order to gather information for data recording and subsequent analysis, we determine the navigation state by checking various conditional statements to identify specific cases. To facilitate this check, a timer is employed, which triggers the evaluation of the navigation state after a certain amount of time has elapsed. The steering and teleportation actions are identified by verifying whether the respective *InputActionReference* [22] are being pressed or not. In Table 3.1 the references are termed as steeringReference and teleportationReference respectively.

However, when it comes to walking, additional variables are required. These variables are necessary to differentiate between walking and remaining stationary (i.e., not moving at all). To make this distinction, we calculate the *differenceLocation* between the player's current location and their previous position (Equation 3.4) and compare it against a predetermined *distanceThreshold*.
The current position is retrieved from the position of the *Main Camera* GameObject, which

[22] `https://docs.unity3d.com/Packages/com.unity.inputsystem@0.1/api/UnityEngine.Experimental.Input.InputActionReference.html`

represents the HUD of the player. At the outset, the initial value of *previousLocation* is set to zero (Equation 3.3), and subsequently updated with the current location (Equation 3.5). In our case, the *distanceThreshold* is configured with 0.3. Consequently, if the difference between the previous and the current location is greater than 0.3 m in either the x or y-direction, it will be definitely classified as the walking navigation state at least.

$$Vector3\ previousLocation = Vector3.zero; \tag{3.3}$$

$$Vector3\ differenceLocation = previousLocation - camera.position; \tag{3.4}$$

$$previousLocation = cameraGameObject.position; \tag{3.5}$$

Moreover, the combination of different locomotion techniques is also considered a separate navigation state, as each technique can be executed independently, although the execution of certain combinations may not always be practical or useful. The Table 3.1 outlines the conditions for the eight different navigation states that the user can be in, which are checked in the described order from top to bottom.

Table 3.1: Navigation state conditions

State	**Condition**
`Walking && Steering && Teleportation`	`(differenceLocation.x > distanceThreshold || differenceLocation.y > distanceThreshold) && steeringReference.action.IsPressed() && teleportationReference.action.IsPressed()`
`Walking && Steering`	`(differenceLocation.x > distanceThreshold || differenceLocation.y > distanceThreshold) && steeringReference.action.IsPressed()`
`Steering && Teleportation`	`steeringReference.action.IsPressed() && teleportationReference.action.IsPressed()`
`Walking && Teleportation`	`(differenceLocation.x > distanceThreshold || differenceLocation.y > distanceThreshold) && teleportationReference.action.IsPressed()`
`Steering`	`steeringReference.action.IsPressed()`
`Teleportation`	`teleportationReference.action.IsPressed()`
`Walking`	`differenceLocation.x > distanceThreshold || differenceLocation.y > distanceThreshold`
`Nothing`	`else`

3.8 Scene Generation

The scene generation process is initiated in the *Start()* function of the *GenerateScene* script. A scene is created by producing each prefab in the amount specified. To be more

specific, the prefabs conists of prefabs for red and brown mushrooms, prefabs for each type of tree (9 in total), and prefab for logs, grass, rocks and trees.

The key function responsible for scene generation is *generatePrefab()*. Within this function, a random position and rotation are generated. Subsequently, it is checked whether there are any other objects present at this random position using *Physics.OverlapShere()*[23]. The function returns an array with all the colliders touching or inside the sphere. If the size of the array is one, which means in this sphere only lies the ground plane, then no other objects are occupying that particular spot, and a clone of the prefab is instantiated[24]. Additionally, for a better overview, each clone is set to a parent according to its prefab. In cases where the parameters are too restrictive, such as when the distance of the random position is set too small, the while loop may never end. To prevent an infinite loop, the function attempts to generate a clone 5000 times. If no successful clone is generated within this limit, the loop is broken and an error message is logged.

```
generatePrefab(numberPrefabs, parent, prefab)
int breakCounter = 0
while (numberPrefabs != 0) {
    // create random position
    float randomPosX = nextFloat(-maxDistanceX, maxDistanceY)
    float randomPoxZ = nextFloat(-maxDistanceZ, maxDistanceZ)
    Vec3 randomPos = new Vec3(randomPosX, 0, randomPoxZ)

    // create random rotation
    float randomRotY = nextFloat(0, 360)
    Vec3 randomRot = new Vec3(0, randomRotY, 0)

    // check if there are other objects in this position
    var hitColliders = Physics.OverlapSphere(randomPosition);

    breakCounter++
    if (hitColliders.Length == 1) { // only collide with plane
    var clone = Instantiate(prefab, randomPos, randomRot)
    clone.transform.SetParent(parent.transform)
    numberPrefabs--
    }

    if (breakCouter == 5000) { // do not try more than 5000 times
    Debug.Log("Something went wrong")
    break
    }
```

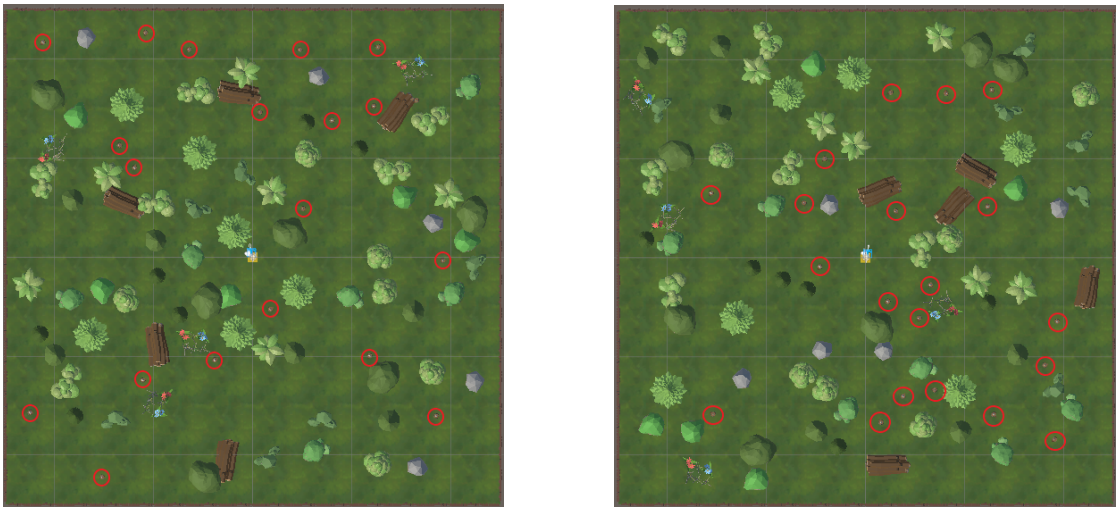
[23] https://docs.unity3d.com/ScriptReference/Physics.OverlapSphere.html, accessed July 3, 2023

[24] https://docs.unity3d.com/ScriptReference/Object.Instantiate.html, accessed July 3, 2023

}

Although the order of generation is not critical, the process begins with mushrooms, followed by trees, logs, grass, rocks, and finally bushes.

As mentioned in Figure 3.4, it is possible to regenerate a scene. The process involves first destroying the existing scene and then generating a new one. To destroy the scene, each prefab is removed by iterating through all children of a parent object and then destroyed[25] each child game object.



(a) Example scene 1 (b) Example scene 2

Figure 3.10: Random scene generation, mushrooms are circled in red

3.9 Data Recording

To conduct data analysis in the later stages of this thesis, specifically in chapter 4, it is necessary to capture and record relevant data. In this project, data is recorded by storing pertinent information in a CSV file. The functionality responsible for data recording and the detection of navigation states is encapsulated within the *Data Recording* GameObject. As seen in the procedure of the experiment Figure 3.4, data recording initiates only after the experiment has started, so after the calibration is successfully completed. In the *Update()* function *writeCSV()* is called, which verifies whether a CSV file has been created. If not, it proceeds to invoke the *createCSV()* method. This method establishes the file name based on the time of creation and writes the corresponding headers, based on the selected data, into the file. When initializing a new instance of *StreamWriter*[26]

[25] `https://docs.unity3d.com/ScriptReference/Object.Destroy.html`, accessed July 3, 2023

[26] `https://learn.microsoft.com/en-us/dotnet/api/system.io.streamwriter?view=net-7.0`, accessed June 20, 2023

	A	B	C	D	E	F	G	H
1	Frame ID	Time	Collision Co	Red Mushro	Brown Mush	RL Position X	RL Position Y	RL Position Z
2	1586	00:00:01	0	0	0	-0,1148084	1,448811	0,04211891
3	1587	00:00:02	0	0	0	-0,1147426	1,448636	0,04202247
4	1588	00:00:03	0	0	0	-0,1147089	1,448487	0,04190952
5	1589	00:00:04	0	0	0	-0,1146729	1,448338	0,04180086
6	1590	00:00:05	0	0	0	-0,1146619	1,448282	0,04167587

(a) Frame ID, time, collision counter, red and brown mushroom counter, real-life position

I	J	K	L	M	N	O
Game Position X	Game Position Y	Game Position Z	Rotation X	Rotation Y	Rotation Z	Navigation State
0	0	0	0,1169373	0,8338777	-0,1613389	N
0	0	0	0,1176886	0,8336179	-0,1617164	N
0	0	0	0,1186963	0,8331517	-0,1628485	N
0	0	0	0,1189026	0,8331509	-0,162289	N
0	0	0	0,1190004	0,8331552	-0,1622314	N

(b) In-game position, real-life and in-game rotation, navigation state

Figure 3.11: CSV File example data recording

class for the specified file, the constructor automatically creates a file. It is important to note that if the file already exists, it will be overwritten. After the generation of the file, the current time is retrieved along with additional data written in the CSV file, see Figure 3.11a and Figure 3.11b respectively. Data are recorded at every frame and include the frame counter, time, collision counter, respective mushroom counter, camera position in the real environment, camera position in the virtual environment, camera orientation (which remains consistent in both the virtual world and real life), and the navigation state as described in section 3.7.

CHAPTER 4

Data Gathering and Results

In this chapter, we did some pilot testing to test the usability of the experiment platform. In the process of gathering data, a series of trials were conducted to collect relevant information. This section provides an overview of the types of data that were gathered and analyzes the data, which subsequently formed the base for interpreting the results.

4.1 Data Gathering

The data for this pilot study consisted of one participant doing 5 repetitions of the procedure. The tracking area was 1.5x1.5 meters. Data were collected by using the implemented data recording function, as discussed before, see section 3.9. The data gathered included various aspects of the user's interactions and behaviors in the virtual environment. In order to visualize the trajectory of the player, the HMD position data recorded during the trials was utilized. This data was used to create the figures in 4.1, which represent the trajectory for both real-life and in-game movement. Both the x-axis and z-axis represent distances travelled in meters. The origin, which serves also as the starting point, is situated at coordinates (0, 0) The start and end points of the player's movement are additionally marked with the letters 'S' and 'E' respectively, providing further context to the trajectory visualisation. Additionally, the trajectory path is color-coded in both plots based on the corresponding navigation state mentioned in section 3.7 and further Table 3.1.

Furthermore, the step charts in Figure 4.2 provided visual representations of the collision and mushroom counters throughout the trial. These charts showcase the occurrences of collisions and mushroom interactions during the trials. In these graphs, the x-axis depicts the time and the y-axis the number of collected mushrooms or the counted collisions. In addition to the plots before, another relevant aspect of data analysis involved determining the percentage of each technique usage, as illustrated in Figure 4.3. This was achieved by summing up the frames in which a particular technique was utilized. The resulting

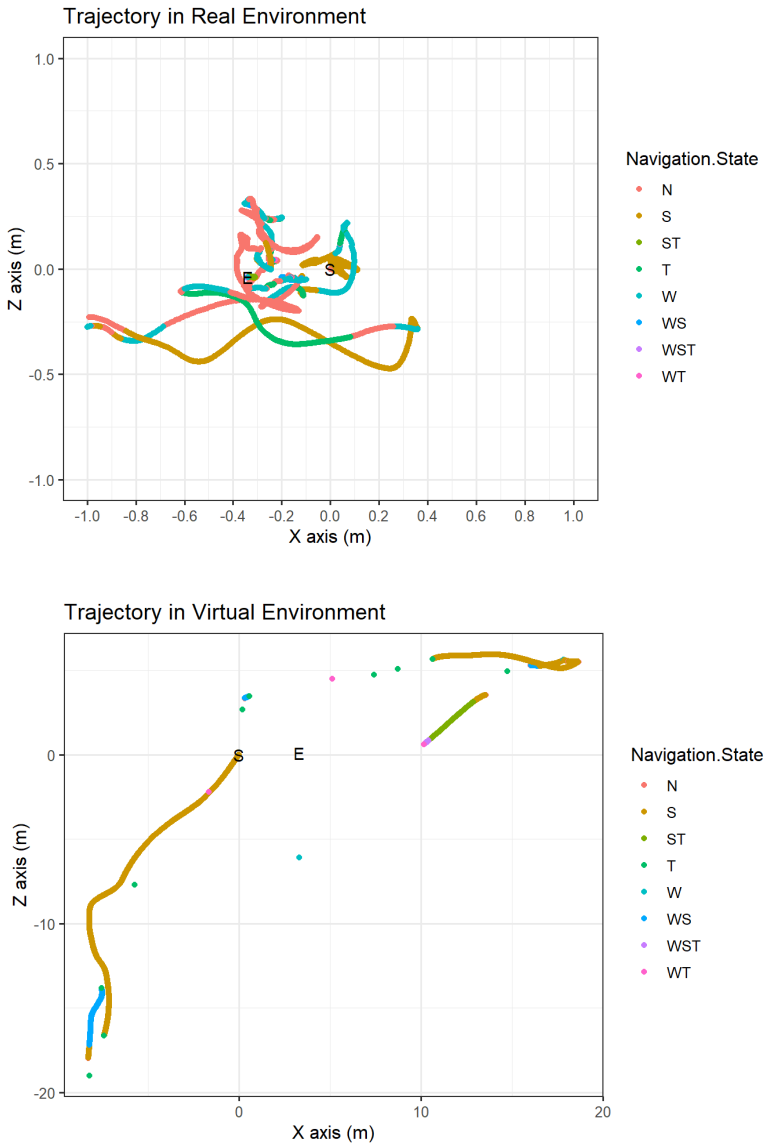


Figure 4.1: Trajectories in real and virtual environment

values in the y-axis represented the proportion of uses the technique employed during the trials, hence generating a histogram that visually depicted the distribution of technique usage across the trial.

4.2 Results and Interpretation

4.2.1 Trajectories

Based on the observations from plots 4.1, it can be inferred that the tracked space available for the trial was either limited in size or the user chose not to move extensively within the

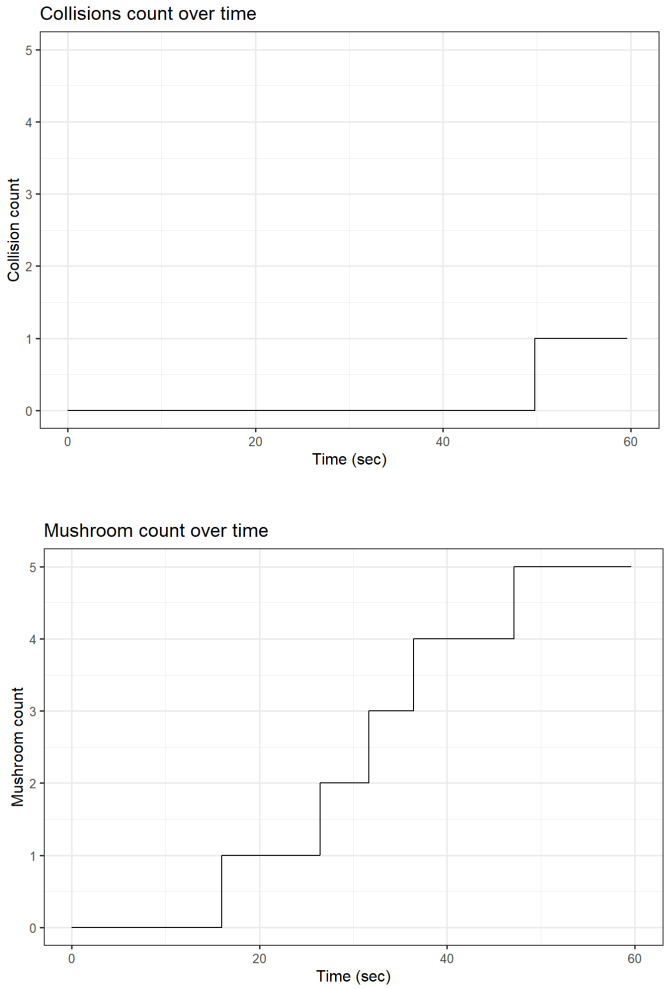


Figure 4.2: Collision and red mushroom counter over time

environment. This can be concluded as the trajectory in the real-life environment was only recorded inside a 1m x 1m periphery, which is also the cause for the different size scaling of the plot. In hindsight, when considering the trajectory of the real environment, it is intriguing to observe that many trajectory paths are classified under the "N" navigation state, which represents a state of not moving. This is noteworthy since the user should ideally not be moving during this state. This case is also attributable to the "T", the teleportation state, as the player does not need to move or else it would account as "WT", the mixed navigation state of walking and teleportation. It is possible that these occurrences could be attributed to the potential inaccuracies in the detection of movement versus no movement. For instance, if the user is walking very slowly, it might be interpreted as a state of "nothing" since the navigation state is based on a snapshot of a current time frame.

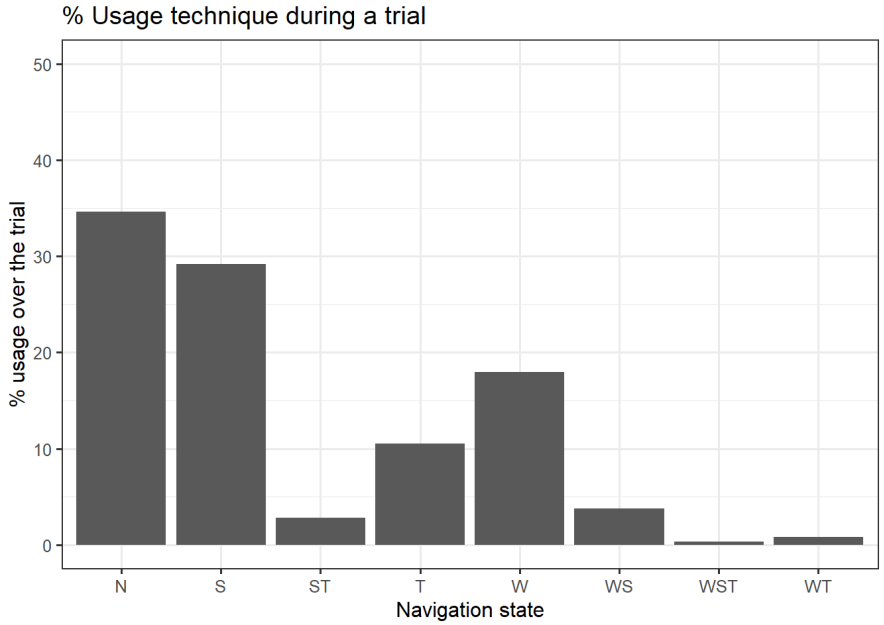


Figure 4.3: Usage technique during a trial

Furthermore, based on the trajectory of the VE, in the virtual world, the greatest distance was conquered using steering, as visually shown by the brown color coding. This is also evident in the histogram of technique usage in 4.3, which indicates that the most used locomotion method was in fact steering, with a usage of around 30%. This can be explained by the advantages of steering, as already discussed in subsection 2.1.2. Steering enables users the highest level of control for users and is in relation fairly easy to understand. Hence, the player might choose steering over options as it provides the player more speed in contrast to walking but more control and a faster sense of orientation in comparison to teleportation. Upon further analysis of the locomotion techniques, walking is in second, accounting for approximately 18% of the trial. It is preferred before teleportation, which was employed for around 10% of the trial duration. As previously explored in subsection 2.1.1, this preference for walking could be attributed to its perceived benefits in terms of immersion and natural movement. On the other hand, teleportation is perceived rather negatively, as its usage is tied with temporary disorientation, as discussed in 2.1.3. Especially in a time-bound setting, this would result in the loss of valuable time, hence the preference for other locomotion techniques.

On the other hand, the combination of multiple locomotion methods is perceived less favorably and is used less frequently compared to individual locomotion techniques. However, it is worth noting that among the mixed navigation states, the combination of walking and steering is the most commonly used. This can be explained by the fact that by combining walking and steering, the player achieves even faster movement compared to using either of these locomotion techniques alone. At the same time, the player is still

in a fairly controlled position, differentiating it from teleportation. In general, any mixed navigation state involving teleportation is practically seen as not of use and also used less frequently. This could be reasoned by the fact that teleportation 'overwrites' other techniques, limiting their effectiveness when combined with teleportation.

4.2.2 Collisions and Mushrooms

As observed in the step chart in plot 4.2, it is evident that not all mushrooms were collected during the trial. However, it is not explicitly indicated which specific mushrooms were gathered or if the chart represents the cumulative sum of all mushrooms encountered. Nevertheless, notable patterns can be observed from the data. In the initial 20 seconds of the trial, no mushrooms were gathered. This could be explained as the player possibly needs time to orient themselves and become accustomed to the locomotion techniques. During this period, the player may have been focused on exploring the virtual environment and becoming familiar with the controls rather than actively collecting mushrooms.

As time progresses between 20 and 50 seconds, the number of mushrooms gathered steadily increases. This suggests that the player became more proficient in utilizing the locomotion techniques and started actively seeking out and collecting mushrooms. However, at around the 50-second mark, a collision occurred, which appears to be the only instance throughout the trial. This collision may be attributed to the time pressure experienced by the player as the trial progressed, leading to increased stress and potentially unintentional mistakes. In further trials, several experiments with different time limits could be conducted and also in various repetitions to examine in what way the factor time influences the behavior. To continue along this line of thought, it would be of interest to investigate how a rather experienced player would approach this scenario and which locomotion techniques would be favored in this case.

Additionally, for further analysis, it would interesting to examine the usage of locomotion techniques over time and explore any potential correlations between the usage of locomotion techniques, the number of collisions, and the number of gathered mushrooms. Therefore, it would be possible to identify any patterns or trends in relation to each occurrence. This analysis could help uncover whether certain locomotion techniques are associated with a higher or lower frequency of collisions or a greater efficiency in the mushroom collection.

CHAPTER 5

Conclusion

In this thesis, we explored various aspects of our chosen locomotion techniques - walking, steering, and teleportation, and their impact on user behavior within a virtual environment. The goal of our study is to gain further understanding of these locomotion methods, particularly how the user's behavior is affected when they have the option to select among them. This is achieved by providing a experimental environment, in which the player has the task to collect as many mushrooms as possible in a given time frame with the choice to use any of the three locomotion techniques.
In theoretical hindsight, we first gave the reader an in-depth overview of each implemented locomotion technique and an evaluation of each in chapter 2. Subsequently, the experiment itself was further explained in detail in chapter 3, focusing on how each game mechanic was developed and with which intention behind it. The final section of this thesis involves the data gathering and the analysis of this data chapter 4. Through data gathering and analysis, we gained valuable insights into how different locomotion methods are perceived and utilized by users. In the pilot study, we collected interesting results of how the user perceived each locomotion technique and concluded the first implications. In these first results, it was highlighted the significance of steering as the preferred locomotion method, mainly due to its high level of user control and ease of understanding. Walking also emerged as a favored choice, supposedly due to its immersive and natural movement qualities. Teleportation however depicts the least used locomotion technique. This could be explained by its increased spatial disorientation after usage, which is a crucial factor to consider in a time-driven setting.

Due to time constraints, certain aspects of the platform could not be further improved but are still encouraged to be developed in the future. One area for improvement is enhancing the detection of the navigation state in section 3.7, differentiating between moving and nothing more accurately. Generally speaking, the detection and classification of user movement in real-time can be a challenging task, and subtle variations in movement speed or changes in the user's position could lead to misinterpretations. These inaccuracies

might result in trajectory paths being assigned to the "N" navigation state despite the user actually being in motion, albeit at a slower pace. Another limitation involves the border detection in the freeze-turn method, see subsection 3.4.3. Currently, border detection relies on comparing the player's distance and angle from the wall. This could be further improved by incorporating additional external values for more precise border detection. Furthermore, the scene generation could be improved by incorporating already placed objects. By doing so, obstacles would no longer be placed in a completely random order but instead be positioned based on available space, ensuring a more even distribution of objects within the scene. Regarding the general implementation of the locomotion techniques, it was suggested in section 2.1.3 to prevent the lost sense of spatial orientation to move the player very quickly. However, in our current implementation, the user is teleported instantly, which could pose a disadvantage in the user's flow. It is essential to handle this consideration with caution, as this implementation might not truly be considered as "teleportation" in the traditional sense. Future refinements could explore alternative teleportation methods that maintain a smoother transition and preserve the player's sense of spatial orientation.
Generally speaking, an enhanced experimental platform could benefit the research as undesired disturbances are avoided and the focus of the player and thus of the study can shift towards the evaluation of locomotion techniques.

In conclusion, this study contributes to a deeper understanding of locomotion methods and their implications for virtual reality experiences. Even though this thesis does not examine the implications of a certain usage of locomotion techniques and would go beyond the scope of this thesis, it does lay down the groundwork for a study of how locomotion techniques can impact the user. Furthermore, based on the first data gathered, certain findings can already be concluded which could serve as a first orientation on where to focus. This information can be used to improve the platform further for future testing. In the future, extended studies should be conducted, especially with changed parameters, such as with a variety of time constraints, number of mushrooms, and trials per player. As already mentioned before in chapter 4 it would be an interesting conclusion how the experience of a player impact the usage of locomotion techniques.
One key takeaway from this research is the emphasis on user-centered design when implementing locomotion systems. Considering individual user preferences and optimizing gameplay mechanics will be crucial in delivering satisfying and immersive experiences in virtual reality. Furthermore, the broader implications of this study extend beyond this specific research domain. The insights gained from this exploration can inform and influence future developments in virtual reality technology, elevating user experiences and advancing the field as a whole. This study represents a meaningful step towards a more comprehensive understanding of locomotion methods in virtual reality, with the potential to impact future design decisions, enhance user engagement, and contribute to the ongoing evolution of virtual reality technology.

List of Figures

List of Tables

Bibliography

[BH99] Doug Bowman and Larry Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. *Symposium on Interactive 3D Graphics*, 182, 09 1999.

[BM07] Doug A. Bowman and Ryan P. McMahan. Virtual reality: How much immersion is enough? 2007.

[BOMA20] Hugo Brument, Anne-Hélène Olivier, Maud Marchal, and Ferran Argelaguet. Does the control law matter? characterization and evaluation of control laws for virtual steering navigation. In *ICAT-EGVE 2020-International Conference on Artificial Reality and Telexistence & Eurographics Symposium on Virtual Environments*, pages 1–10, 2020.

[CA17] Chris G. Christou and Poppy Aristidou. Steering versus teleport locomotion for head mounted displays. volume 10325 LNCS, pages 431–446. Springer Verlag, 2017.

[CCBWS18] Noah Coomer, William Clinton, Sadler Bullard, and Betsy Williams-Sanders. Evaluating the effects of four vr locomotion methods: Joystick, arm–cycling, point–tugging, and teleporting. Association for Computing Machinery, Inc, 8 2018.

[CGBL98] Sarah Chance, Florence Gaunet, Andrew Beall, and Jack Loomis. Locomotion mode affects the updating of objects encountered during travel: The contribution of vestibular and proprioceptive inputs to path integration. *Presence*, 7:168–178, 04 1998.

[CTHP16] Chris Christou, Aimilia Tzanavari, Kyriakos Herakleous, and Charalambos Poullis. Navigation in virtual reality: Comparison of gaze-directed and pointing motion control. Institute of Electrical and Electronics Engineers Inc., 6 2016.

[Gig93] Michael A. Gigante. 1 - virtual reality: Definitions, history and applications. In R.A. Earnshaw, M.A. Gigante, and H. Jones, editors, *Virtual Reality Systems*, pages 3–14. Academic Press, Boston, 1993.

[GS98] Enrico Gobbetti and Riccardo Scateni. Virtual reality: Past, present, and future, 1998.

[HMG11] Paul Havig, John McIntire, and Eric Geiselman. Virtual reality in a cave: Limitations and the need for hmds? *Proceedings of SPIE - The International Society for Optical Engineering*, 8041, 05 2011.

[KBH01] G. Kessler, Doug Bowman, and Larry Hodges. The simple virtual environment library: An extensible framework for building ve applications. *Presence: Teleoperators and Virtual Environments*, 9, 03 2001.

[KLA+23] Panagiotis Kourtesis, Josie Linnell, Rayaan Amir, Ferran Argelaguet, and Sarah E. MacPherson. Cybersickness in virtual reality questionnaire (csq-vr): A validation and comparison against ssq and vrsq. *Virtual Worlds*, 2:16–35, 1 2023.

[KLBL93] Robert S. Kennedy, Norman E. Lane, Kevin S. Berbaum, and Michael G. Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, 1993.

[KPCC18] Hyun K. Kim, Jaehyun Park, Yeongcheol Choi, and Mungyeong Choe. Virtual reality sickness questionnaire (vrsq): Motion sickness measurement index in a virtual reality environment. *Applied Ergonomics*, 69:66–73, 2018.

[KR21] Yong Min Kim and Ilsun Rhiu. A comparative study of navigation interfaces in virtual reality environments: A mixed-method approach. *Applied Ergonomics*, 96, 10 2021.

[LaV17] Joseph LaViola. *3D User Interfaces*. 2nd edition. edition, 2017.

[LLS18] Eike Langbehn, Paul Lubos, and Frank Steinicke. Evaluation of locomotion techniques for room-scale vr: Joystick, teleportation, and redirected walking. Association for Computing Machinery, 4 2018.

[NNS16] Niels Nilsson, Rolf Nordahl, and Stefania Serafin. Immersion revisited: A review of existing definitions of immersion and their relation to different theories of presence. *Human Technology*, 12:108–134, 11 2016.

[PFW09] Tabitha C. Peck, Henry Fuchs, and Mary C. Whitton. Evaluation of reorientation techniques and distractors for walking in large virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 15:383–394, 5 2009.

[RL06] Roy A. Ruddle and Simon Lessels. For efficient navigational search, humans require full physical movement, but not a rich visual scene. *Psychological Science*, 17(6):460–465, 2006.

[RL09] Roy A. Ruddle and Simon Lessels. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction*, 16, 4 2009.

[RVB11] Roy A. Ruddle, Ekaterina Volkova, and Heinrich H. BüLthoff. Walking improves your cognitive map in environments that are large-scale and large in extent. *ACM Transactions on Computer-Human Interaction*, 18, 6 2011.

[RVB13] Roy A. Ruddle, Ekaterina Volkova, and Heinrich H. Bülthoff. Learning to walk in virtual reality. *ACM Transactions on Applied Perception*, 10, 5 2013.

[SB20] Volkan Sevinc and Mehmet Ilker Berkman. Psychometric evaluation of simulator sickness questionnaire and its variants as a measure of cybersickness in consumer virtual environments. *Applied Ergonomics*, 82:102958, 2020.

[SFR+10] Evan Suma, Samantha Finkelstein, Myra Reid, Sabarish Babu, Amy Ulinski, and Larry F. Hodges. Evaluation of the cognitive effects of travel technique in complex real and virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 16:690–702, 2010.

[SHCV+18] Patric Schmitz, Julian Hildebrandt, André Calero Valdez, Leif Kobbelt, and Martina Ziefle. You spin my head right round: Threshold of limited immersion for rotation gains in redirected walking. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 01 2018.

[SSB+20] Dimitrios Saredakis, Ancret Szpak, Brandon Birckhead, Hannah A. D. Keage, Albert Rizzo, and Tobias Loetscher. Factors associated with virtual reality sickness in head-mounted displays: A systematic review and meta-analysis. *Frontiers in Human Neuroscience*, 14, 2020.

[SSH20] Ehsan Sayyad, Misha Sra, and Tobias Hollerer. Walking and teleportation in wide-area virtual reality experiences. pages 608–617. Institute of Electrical and Electronics Engineers Inc., 11 2020.

[Ste00] Jonathan Steuer. Defining virtual reality: Dimensions determining telepresence. *Journal of Communication*, 42, 07 2000.

[SVS05] Maria Sanchez-Vives and Mel Slater. From presence to consciousness through virtual reality. *Nature reviews. Neuroscience*, 6:332–9, 05 2005.

[WKFK18] Tim Weissker, Andre Kunert, Bernd Frohlich, and Alexander Kulik. Spatial updating and simulator sickness during steering and jumping in immersive virtual environments. pages 97–104, 03 2018.

[WNR+07] Betsy Williams, Gayathri Narasimham, Bjoern Rump, Timothy P McNamara, Thomas H Carr, John Rieser, and Bobby Bodenheimer. Exploring

large virtual environments with an hmd when physical space is limited. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, pages 41–48, 2007.

[WP02] Kenneth R. Walsh and Suzanne D. Pawlowski. Virtual reality: A technology in need of is research. *Commun. Assoc. Inf. Syst.*, 8:20, 2002.