RWTH Aachen University
Virtual Reality & Immersive Visualization Group
and
Academic and Research Department Engineering Hydrology

Utilizing Alternative Tracking Methods for Intuitive Controllerless Navigation in Virtual Reality Environments

Master Thesis

presented by

Ján Jeremy Tugsbayar, B.Sc.

1st Examiner: Prof. Dr. Torsten Kuhlen

2nd Examiner: Prof. Dr.-Ing. Heribert Nacken

Advisor: Patrick Querl, M.Sc.

The present work was submitted to the Virtual Reality & Immersive Visualization Group

Aachen, June 4, 2025

Abstract

This thesis investigates whether hand tracking based alternatives to traditional VR controllers can support a complete virtual reality experience, and whether adding gaze tracking enhances those alternatives. Using the XR Interaction Toolkit and XR Hands, we design two new controller-less interaction modes that combine already established concepts from research. The first "hands only" mode, utilizes hand gestures for both interaction and locomotion including hand orientation based steering. The second mode "hands with gaze", combines those same gestures for locomotion but utilizes gaze based steering and replaces long range hand interaction with a gaze based interaction. In both modes, the pinch gestures replace controller buttons for selecting, grabbing, and UI interactions.

We implement these modes in Unity and develop a VR application that also supports a standard controller based interaction mode as baseline. To evaluate feasibility and usability, we conduct a within-subjects user study, in which participants complete five tasks, ranging from spatial navigation to UI interaction using our two new interaction modes and with controllers as a baseline. Performance metrics are logged and participants subsequently fill out User Experience Questionnaire (UEQ) and Presence Questionnaire (PQ) surveys. Finally, semi-structured interviews is performed to gain further insight into the participant's impressions.

Results consistently show that controller based interaction remains the fastest and most reliable across all tasks. Hands only interaction yields slower performance but is generally well received, especially for close range manipulation, gaze based interaction introduces additional errors and user discomfort. Questionnaire scores mirror these findings: controllers score highest on learnability, dependability, efficiency, and presence, while hands only scores are more variable but acceptable. Hands with gaze elicits the lowest and most inconsistent ratings with wide variance.

Interviews reveal that hand and gaze steering struggles with fine adjustments at narrow passages and that hand steering can cause cyber-sickness when movement direction conflicts with expected gaze direction. Participants also emphasize the challenges of gesture learnability and its consistency of detection. While some appreciate the novelty of hand tracking and its potential in low-interaction scenarios, most find controllers more consistent.

In conclusion, this work demonstrates that hand tracking based VR interaction is feasible for simple tasks but currently cannot fully replace controllers for complex or high-precision applications. The additional complexity introduced by gaze interaction outweighs its benefits. We recommend that developers explore hybrid approaches, hand only interaction with optional gaze steering for steering.

Contents

Chapter 1

Introduction

Virtual Reality (VR) as technology has seen considerable growth over the last decades. Initially, VR was relative niche technology, which was inaccessible to the wide population, due to its cost and limited application. Lately it has become widespread with many different use cases. VR and its sister technology *Augmented Reality* (AR) has a wide range of deployment ranging from military training [TAA+24] to surgical medical training [GSS+21]. Moreover, it's easily accessible nowadays to the wide public as a form of entertainment. This is due to its affordable prices and the ability to run standalone without additional hardware. With the affordable prices of the current generation VR technology, its application in general education has been increasing as highlighted by Asitah et al. in [AML+24]. Furthermore it has been applied in context of higher education as highlighted by the systematic reviews by Bicalho et al. [BPdLM23] and Radianti et al. [RMFW20], because it offers a potential huge improvement for students, as they can visualize and understand complicated concepts taught in higher education or train their skills in a virtual environment [SKS23]. As an example of it being used in higher education, it can be deployed in the context of civil engineering, in construction of flood prevention units [QCB+24].

Therefore this thesis serves as a contribution to research of how VR can deployed in educational or academic setting, in which we want to determine whether alternative tracking methods, such as hand tracking, can be utilized as an intuitive replacement for the traditional controllers that are used for VR.

1.1 VR Terminology

Virtual reality can come in different forms, for example as cave automatic virtual environment (CAVE) [MSB+14], where projectors render a virtual environment on walls. Various techniques can be applied to increase the immersion, for example one approach is to introduce a treadmill to simulate real walking [WLC+23]. These highly immersive setups are too costly and require too much space for mass adaption in an educational setting. Therefore for a educational setting, small commercially available VR devices are typically employed. As we use such devices for the thesis, we now describe the relevant terminology and the current state of VR from the perspective of using such commercial devices, for example the Meta Quest product lineup that we employed.

A VR device consists of a *Head Mounted Display* (HMD) that projects a *virtual environment* on the display for the user to see. Current generations of commercially available VR devices allow the user to move freely within a user-defined space, also known as the *play-space*. As an input device, traditionally hand held *controllers* are used. For the move-

ment within the defined space to work, the position of the HMD and controllers needs to be tracked and updated in real time. The Meta Quest series utilizes *inside-out* tracking, which employs an *inertial motion unit* (IMU) that measures and calculates the inertia caused by movement and optical tracking of the environment to determine the current location. The benefit of inside-out tracking is that everything is contained within the HMD and the controllers, therefore no additional hardware is required for tracking. An alternative to inside-out tracking is the *outside-in* tracking used by other competitors, for example the HTC Vive Pro or Valve Index. Outside-in tracking utilizes fixed-position external devices as *lighthouses* that emit infrared light, which is detected by light sensors on the devices to determine the current position of devices. The downside of outside-in tracking is the requirement of external hardware, which needs to be set up any time VR is to be used.

Using the above-mentioned tracking technologies, the position of the user is tracked and updated in real time within the virtual environment. Since our movement is limited by the size of the play-space, there needs to be another option for moving in VR when the virtual environment is bigger than our play-space. Movement in VR is commonly referred as *locomotion*, to generalize the term, locomotion is the ability to move within the virtual environment. Various locomotion methods exist, such as *continuous movement*, which attempts to emulate smooth continuous movement that we perform in reality. There also exist more abstract locomotion methods, for example *teleportation* or the usage of portals.

In order to make the experience more interactive, we also need a way to interact with objects in the virtual environment. Some basic VR *interaction* is, for example, the ability to poke or grab virtual objects.

The conventional input method for locomotion and interaction is controllers, they are hand held interface devices that commonly feature joysticks and buttons. These can be used to perform locomotion and to interact with interactable objects in VR.

Lastly, audio is important to provide an authentic and immersive experience. In order to mimic real acoustics, spatial audio is required [MSU+25] to provide the best experience and it is handled by the speakers on the HMD.

Often an avatar is used as a virtual representation of the user. The quality and the capabilities of avatars can be quite varied, ranging from photorealistic avatars based on the real appearance of the user [CCB+24] to anime inspired [BKSS20].

1.2 Motivation

While the controllers are a simple and effective means of locomotion and interaction, a *hands-free* VR might be required in some VR scenarios. For example, Hombeck et. al in [HVH+23] highlights the need for a hands-free VR in a medical scenario, in which the user must have free hands to use and operate medical instruments and they attempt to solve this by using leaning based movement and compare that to speech based movement. They conclude that voice based movement is better suited, as leaning has a higher risk of cyber-sickness. *Cyber-sickness* is a phenomena when using virtual reality technology that is inherent to the natural functioning of the human body and mismatched perceptions, there is non consensus on the solution of it [DNN14], though there are several ways to help alleviate such as dynamically reducing the field of view (FOV) [ZMF+24]. The possibility of cyber-sickness must be taken into account when designing a VR application. The benefit of being hands-free is enabling the user to perform other actions using their hands. Additionally, using the hands-free approach enables easy introduction of the tangible VR concept [BB21], in which we have real props that are duplicated in VR by the use of a tracking technology.

As a use case, ordnance disposal training can be performed using tangible replica of an explosive [RR22] or training of police officer with tangible tactical equipment [MUNR23].

Our next motivational aspect is immersion and presence, as they are important aspects of a VR experience [WS98]. Immersion is the capability of the VR device to provide as authentic an experience as possible, which blurs the line between reality and virtual reality. Presence is the feeling of being part of the world, which can be achieved by providing the ability to fully engage with the virtual environment. However, using the standard controllers can have a negative impact on the user's immersion and presence, as there is a constant feeling and awareness of holding the controllers in their hands, and current research suggests that hand tracking with virtual hands is more immersive than controllers [SKT+17, HPM21]. In order to provide a better immersive experience [Sla09], we attempt to remove the need for controllers and rely on hand tracking technology to provide virtual hands to the user. In fact the CAMIL framework for learning using VR [MP21] indicates that increasing the presence and the agency, i.e the feeling of being in control and being able to interact to generate actions, has a positive effect on the person's learning interest.

There is also the aspect of cost. VR devices often come as a package with included controllers, however, it is also often possible to procure the HMD alone. By saving on the cost of controllers, more HMDs can be afforded to increase the capacity of users. Moreover, controllers introduce another point of hardware failure as they feature mechanical moving parts, which experience wear and tear.

Lastly, using VR in an educational setting could feature a high number of students participating who have no experience in VR. Hand tracking based interaction could be perceived as more natural and intuitive, thus we also explore whether traditional controllers or hand tracking based locomotion and interaction may be more natural compared to traditional controllers for first time users.

Towards this end, we propose a system which uses hand tracking, gestures, and head tracking as a full replacement for interaction and locomotion in VR.

1.3 Proposal

We propose two VR locomotion and interaction modes. The first one utilizes only hand tracking (shortened to *hands* or *hands only*). The second method combines head tracking, i.e. the gaze of the user, with hand tracking (shortened to *hands with gaze*). When designing our solution, we followed the principle of keeping the same capabilities for each mode, and the controllers served as the baseline, hence our proposal serves as a full replacement for the traditional VR controllers. The proposed modes cover both locomotion and interaction, however, they do not provide any haptic feedback nor affect audio. Hence, our main research question is determining whether our proposed controller-less locomotion and interaction solution is feasible in the context of an educational setting, in which there might be many novices or first time users. Therefore, the goal of our proposal is to be as intuitive as possible.

In our proposal, both modes use hand gestures to initiate and stop locomotion. However, they differ in their steering. The hands mode utilizes the direction of the hand to determine the movement direction, whereas the hands with gaze use steering based on the user's gaze, i.e. the user moves in the direction of their gaze.

In the proposal, the interaction also attempts to replicate the capabilities of the controllers. The controllers are able to provide long range interaction, using a ray generated from the controllers with which we can aim at objects at a distance and click a button on the controllers to interact. They are also capable of providing close range interaction

of poking and grabbing using collision mechanics, that is, when the controllers virtually collide with the interactable object and we click the button. Our hands only mode provides the same capabilities by generating a ray from the hands for long range interaction, afterwards, to interact the user must perform a pinch gesture. The hands only mode also provides the same collision based close range interaction as the controllers: when the hands intersect with an object, we can again perform a pinch gesture to interact with them. The hands with gaze mode replaces the ray from the hands with a ray from the user's head, specifically their point of view. Using their gaze, users can aim at objects from a distance and then perform a pinch gesture using their hands to interact with them. For close range interaction, the same mechanic is used as with hands only. We are essentially replacing the controller button click with a pinch gesture using hand tracking and replacing aiming with controllers with either hands or gaze.

The reasoning for creating two modes, with inclusion of gaze being the differentiating factor, is that head tracking is viewed as the most natural aspect of VR [Sla09]. It is head tracking that makes a difference between just a monitor being attached to the head and VR. Determining whether integrating head tracking in a sensible way into hand tracking based locomotion and interaction forms our second research question.

To evaluate our proposal, we implement it using the Unity game engine and perform a user study. We chose Unity as it provides an extensive packaged called XR, that includes all the tools to create a VR application and because of previous experience developing in Unity. In the user study that we perform, participants need to perform five simple tasks in VR. The tasks are performed three times: once with the controllers as a baseline, then with our proposed two locomotion and interaction modes. The order of tasks, the order of interaction modes, and the tasks themselves are randomized. During the study we measure certain values, such as time required or amount of mistakes performed. These metrics are used as quantitative data to be analyzed. After performing these tasks, the participants answer a questionnaire in which they rate their experiences as another form of quantitative data. Lastly, we also perform an interview to gain personal qualitative feedback from participants.

1.4 Related Work

None of the concepts we are using are novel, they are all well established topics in the field of VR. We now present a selection of related work on the individual concepts that we utilize, which served as the basis of our work.

Gaze steering is a method that has been used in VR since its inception [BKH97, QT18]. Interacting with gaze is also a commonly researched topic [PLLB17, LHM22]. So much so that basic gaze interaction is built into the Unity game engine, which we utilize to implement and test our proposal. While our gaze aiming and pinching to interact is meant to mirror the Meta Quest VR device product series, the concept has been explored before [PMMG17, KML+25, BHN21].

Neither is gesture based locomotion a new thing. Zhang et al. [ZCP+17] propose a gesture based locomotion, however, their implementation requires a separate hand tracking device, as HMDs with built-in hand tracking were not common in 2017. Moreover, it only provides locomotion and does not support gesture based interaction. A more modern proposal by Bonino et al. [BTG+24] uses an HMD with built-in hand tracking. They compare what they defined as semi-natural technique, which utilizes a pointing gesture for locomotion a with a fully natural technique that emulates walking by leaning with a focus on navigating through an environment with elevation changes. Results show, that

the semi-natural technique, which is viewed as less natural, is more comfortable to the users.

Gesture based interaction has been explored in the context of educational by Maarof et. al in [MMN+21]. They implemented gesture based interaction for desktop virtual reality using and they highlight that careful and proper design is required to implement it in a practically usable way. Khundam et al. compare the difference between controller and hand tracking medical training of intubation in [KVP+21], however they do not find any significant difference. Similarly Buń et.al [BHK22] implemented hand tracking based pinch gesture selection similar to ours in an mixed reality for educational applications. Their limited testing during the COVID pandemic gave positive results in its application in teaching manual skills. Cardoso et al. [Car16] in have a very similar locomotion to our proposal, they use gestures to initiate and to stop locomotion. They also employ hand steering, though instead of the orientation of the entire hand, they use the angle of the index finger. Additionally they also employ a gaze steering together with the gesture locomotion. They compare these two locomotion methods with controllers, which is a similar approach to our user study, though only the locomotion aspect. The results show that game-pad was the best locomotion mode with gaze being second and hand steering being the worst perceived one.

Minnekanti et al. propose a VR based classroom for lectures and exams. [MBB24], though they use traditional controllers as an input method and provide no evaluation, as it is a work in progress with future extensions planned.

To the best of our knowledge, there does not exist work that provides a full package of hand tracking and head tracking based interaction and locomotion, with the goal of fully replacing controllers in the context of educational setting, and this thesis attempts to fill this research gap.

1.5 Organization of the Thesis

In the next Chapter 2 we give a high level overview of our proposed solution to controller-less VR, specifically the two new additions that we had to design and integrate into existing VR development frameworks. Next, Chapter 3 covers the implementation aspect of our proposal and how we utilize the XR package to implement our proposed solution. Chapter 4 covers the design and implementation of the user study we performed and how we gather data in order to later analyze it to answer our two research questions. The results of the user study are analyzed and evaluated in Chapter 5. Lastly, a conclusion follows with our findings in Chapter 6.

Chapter 2

Design

Our proposed controller-less VR locomotion and interaction is heavily based on already existing implementations for ease of development and multi-platform support, which will be presented in Chapter 3. This chapter serves to give a high-level overview of our custom additions, which are gesture-based locomotion and gaze interaction that utilize the hand tracking and gesture-detection systems of the HMD.

2.1 Gaze Interaction

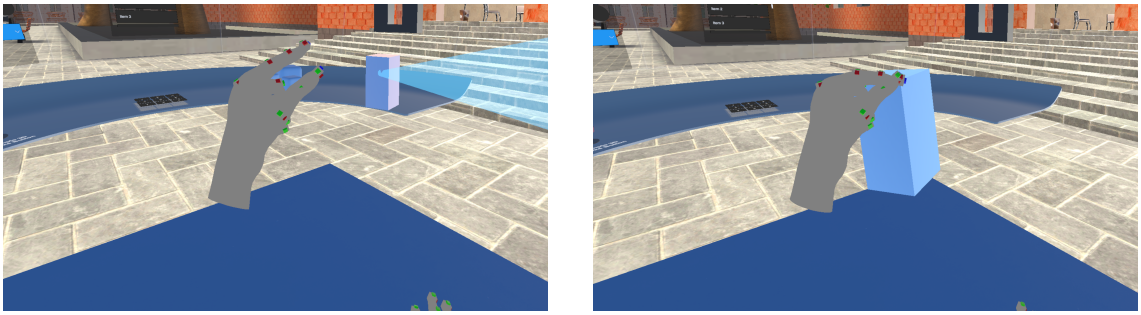


Figure 2.1: Demonstration of custom gaze interactor pinching to grab. Left side shows the ray hitting a valid object. Right side shows the object attached to our pinching fingers after the pinch is performed.

We have designed a new gaze interaction functionality which enables the user to pick up objects and to interact with UI elements. It is based on an existing implementation that uses a delay to activate interaction, our modifications and additions replace this by providing the ability to perform a pinch gesture to grab and interact with objects. It combines gaze tracking as an aiming method with gestures detected by hand tracking. To interact, the user must aim using their gaze, upon aiming at a valid interactable object, a ray appears to visually confirm that they are indeed aiming at an interactable object.

The hand tracking and gesture-detection system uses the HMD's cameras to track the position of the hands and each finger's curl. By detecting the curl of each finger, it is capable of detecting different hand gestures.

For grabbable objects, when the user performs a pinch gesture, the object gets immediately attached to the pinching fingers. The grabbed object remains in the hand as long as the pinching gesture is held. After the object has been grabbed and is held in our pinch,

the ray is deactivated. Releasing the pinch drops the grabbed object. The functioning of grabbing objects with gaze is visible in Fig. 2.1.

UI interaction functions the same: we aim at a UI element and pinch to activate it. This works well for click-based UI elements, such as buttons. However, for scrolling, we needed to create a way to scroll using gaze. To scroll, the user needs to aim at a scrollable UI, then perform a pinch and hold the pinch. After a set amount of delay, we enter scrolling mode and we stay in scrolling mode as long as the pinch gesture is held. In this mode, the user is able to move their head up and down to scroll in the corresponding direction. Moving the head up scrolls upwards and moving the head down scrolls downwards. Using a UI slider functions the same way, except that the user needs to rotate their heads to move the slider to the left or the right.

When designing how to perform gaze interaction, we contemplated how to grab objects using gaze. Our initial idea was to grab the object by keeping it hovering in front of the user, though this idea causes obstructed vision. Another idea was to attach the grabbed object to the user's body, for example the chest, however, we do not have an avatar system and the object would be simply floating below the user's head. In the end, we decided to attach the object to the user's hand, which allows the user to handle and keep holding the object in the same manner as with real hands and is analogous to existing hand based interaction implementations. The interaction with UI by pinching is purposely inspired and designed to mimic the hand based UI interaction of the Meta Quest VR product series.

Initially, we had the ray active and visible all the time, however, that was found disturbing and distracting. Hence, we changed it to be active only when aiming at a valid object. For the case when there might be many objects in range, we have a set maximal allowed range for gaze interaction. The origin of the ray is the user's point of view, in VR view, it is visible as a ray coming from between the eyes.

Details about how exactly it works under the hood are presented in Section 3.2.

2.2 Gesture Locomotion

Using the gesture detection system, we created a full locomotion system, which features forwards and backwards continuous movement with steering. Details about the implementation are presented in Section 3.4.

2.2.1 Forwards and Backwards Locomotion

To move forwards with a hand gesture, we initially chose the pointing hand gesture. It requires the index finger to be straight, while the rest of the fingers are fully curled. However, we soon realized that this gesture is not suitable. One of the simplest interactions in VR is poking, and the most common way of poking is using the tip of the index finger, since this is the most natural way for humans to poke objects, as visible in Fig. 4.2. We used a set of existing predefined gestures and the pointing gesture specifically requires all the other fingers to be fully curled, with only the index finger being straight and not curled. During internal testing, we discovered that some people poke with their other fingers curled. This was accidentally detected by the system and activated forwards locomotion. To avoid accidental locomotion, we had to specifically choose gestures that are not used in VR interaction nor in *gesticulation*. While also taking into consideration their social meaning, as specific gestures can have differing meanings in various cultures [Nor10].

While many VR novices do not fully gesticulate or use body language, seasoned VR users, who especially use VR for social interactions [AML+20], use gesticulation and body

Figure 2.2: Locomotion forward using the open-palm gesture with a gesture debugging UI showing the different predefined hand gestures.

language extensively. Provided that they have some sort of finger tracking and full body tracking. Furthermore, when faced with full hand tracking, they may be even more inclined to use gesticulation in a social interaction. Therefore, our initially naive choice of gestures for locomotion became less trivial. After testing internally what gestures would never or hardly be ever used for other purposes, we chose to use the *palm-up* gesture for forwards movement and the *thumbs-down* gesture for backwards movement. The specific palm-up gesture can be viewed in Fig. 2.2. The palm-up gesture does indeed have use cases in social interaction, such as when expecting to be passed over an item or begging, however, these circumstances are not so common in VR as there are usually no physical items to interact with. Therefore, we felt safe using this gesture for forwards movement.

Thumbs-up is a commonly used gesture in social interactions to provide approval, however, thumbs-down is less commonly used and has negative connotations. This is intuitive for backwards movement, as backwards movement is usually required when we are unable to move forwards or to adjust our position in the environment when we overshot our destination.

Lastly, in order to alleviate cyber-sickness, we added a FOV-reducing feature, which is a proven technique to help with cyber-sickness [ZMF+24] by using a black vignette around the user's vision.

2.2.2 Steering

For steering when using gestures, we have created two different methods:

- Hand steering:
  - While holding the corresponding locomotion hand gesture, users can steer by aiming their hand.
  - The orientation of the virtual hand is used as a source of direction for steering.

  - The chosen forwards gesture of palm-up provides a good steering functionality of aiming with the hand and the fingers in the direction of movement.

- Gaze steering:
  - Instead of using the hand's rotation, we use the user's gaze to steer.
  - When moving using the gestures, the user is able to look to the sides to move in their gaze's direction.
  - The direction in which the user's head is looking at, i.e. their gaze is used as the source direction of steering.

Hand Steering

The hand steering uses only hands while omitting any gaze input. However, this method can clash with the gesture detection. When rotating the hand to the left or right, while keeping the palm-up, it may occur that the gesture detection is no longer able to recognize the gesture if the hand is rotated excessively, for example, if one attempts to rotate the hand inwards to strafe to the side. To clarify inwards, let us use the left hand, for example: if the left hand is rotated clockwise in the palm-up position to a degree higher than 45°, the gesture detection fails. Rotating outwards, e.g. the left hand counter-clockwise, is not as problematic, as the nature of human joints does not allow a big rotation outwards. Moreover this same limitation of the joint rotation does not allow big steering inputs when attempting to steer in the outward direction of each hand. To solve this we designed a mechanic of hand switchover, which will be covered in Section 2.2.3. The thumbs-down gesture is substantially harder to detect when the hand is rotated compared to palm-up rotation, however, we do not foresee backwards movement to be extensively used for long-distance locomotion, it is mostly used for small corrections of positions. For longer-range backwards locomotion, it is more natural to turn around and move forward in order to be able to navigate. This assumption was later confirmed in our user study.

Gaze Steering

As an alternative to hand steering and its limitation, we have created a second steering mode called gaze steering. The user still has to perform the gesture, however, we bypass the problem of hand steering by utilizing the user's gaze direction. This effectively decouples movement and steering.

Initially, we had a third steering mode, called locked gaze steering. This steering mode also used the gaze as a source of direction, however, after the locomotion had started, the movement direction was locked and could not be adjusted until the locomotion had ended. We felt that this was too restrictive and decided to omit it in our user study, however, the functionality remains in the implementation.

2.2.3 Double Hands

An additional feature of our gesture locomotion is the ability to use either hand to initiate locomotion and to perform a switchover of hands.

Handedness

It accommodates both left handed and right handed people, as it is reasonable to assume that most people will use their dominant hand for single hand gestures.

Hands Free Interaction

It leaves the other hand free for interactions. When using hand tracking both for locomotion and interaction, it may be necessary to hold an object and move at the same time. The user can use their preferred hand to grab an object and use their other free hand for locomotion. It also enables using tangible VR with real props that are tracked and represented virtually.

Hands Switchover

When using a single hand for locomotion, it is possible for the user to switch over hands by putting their other hand into the same locomotion gesture, afterwards ceasing to perform the gesture with their original hand, the locomotion will not stop and the steering input in case of hand steering is switched to the new hand. This can be used to alleviate problems of keeping their hand in a specific position for longer periods of time. Moreover, it provides better steering when using hand steering, since outward hand rotation is limited, this enables the user to switch to the other hand on the fly to provide better steering. It is, of course, still possible to stop locomotion and switch hands. However, on the fly switching can provide a more fluid movement without interruptions.

Enabling a Sprint Functionality

Sprinting is performed when both hands perform the same locomotion gesture. When using a single hand to initiate locomotion, the user may put their other hand into the locomotion gesture to initiate a sprint. Or they could simultaneously put both hands into the locomotion gesture to sprint. By ceasing to perform the hand gesture with whichever hand, the sprint is deactivated, but locomotion still remains active with the regular movement speed. Sprinting applies a multiplier to the movement speed, we have used a multiplier of 2x, hence it allows the user to save time when moving between locations. However, we omit this functionality in our user study, as some participants may utilize the sprint excessively while others may not, hence leading to skewed time results in our user study. Additionally, we wanted to reduce the number of gestures required, as learning and using gestures is not as straightforward as one would assume [Nor10]. Hence, not having a separate gesture for speed, but simply using both hands, reduces the number of gestures required. One might argue, what if we need the other hand when sprinting? This can be either solved by good design of the virtual environment, in which the user does not need to carry objects over longer distances, as the sprint is most commonly used for traveling between longer distance destinations. However, another possibility is using a gesture based teleportation technique.

2.2.4 Wrist Teleportation

To have locomotion using hands that is equivalent to using controllers, we have also implemented a wrist teleportation technique. It functions by having a button on the bottom side of the wrist of the user's virtual hand. Pressing the button once activates a teleportation ray, similar to the controller's teleportation when holding the right joystick forward. When the ray is active, the user is able to aim using their hand, and, afterwards, pressing the button for the second time teleports the user to their aimed destination. If the button is not pressed for the second time within a set amount of time, the ray deactivates.

The idea was inspired by previous internal work done at the supervising institute, however, the idea of using hand tracking and wrists for teleportation is not a novel idea and there exist several papers about this concept [CUP+22], [KJKa23].

Locomotion through teleportation is commonly used for faster movement than continuous movement or to avoid cyber-sickness. During internal testing, we noticed that users have a personal preference for using mostly teleportation or just continuous movement. Having both continuous movement and teleportation available in the user study could lead to skewed time results, as some participants will prefer to teleport, while others may not. Thus, we have decided not to include the wrist teleportation technique, forcing all the participants to use the continuous movement of the gesture locomotion or the controllers, both of which have the same movement speed. Nevertheless, the functionality still remains in the code and could be used for further research.

Chapter 3

Implementation of Our Proposal

After explaining our design choices of our VR interaction & locomotion and the user study, this chapter provides insight into our implementation.

To implement our vision of controller-less VR, we have used the Unity game engine and its XR Interaction Toolkit[1] and XR Hands[2] packages. These packages provide all the basic functionalities required to develop a VR application, including basic prefabs for locomotion, hand tracking, and gesture detection.

We deliberately chose to use as many existing prefabs and assets provided by XR as possible, to achieve a wide range of compatibility on devices. For instance, our hand tracking and gesture tracking is simply the starter asset prefab provided by the XR package. The same applies for locomotion with the controllers, hand visualization, etc. Although we do not cover every minute detail of the code, as that would be too exhaustive, we present the interesting core aspects. We only cover our custom code and omit explaining code provided by XR, however, then give an overview of the base locomotion and interaction provided by XR. More in-depth information about the entire XR Interaction Toolkit and XR Hands packages can be found in their respective manuals.

All of the development was done in Unity 6 (version 6000.0.32f1) using the XR Interaction Toolkit (version 3.0.7) and XR Hands (version 1.5.0). Meta Quest 3 with the included Quest Touch Plus controllers was used for both the development, testing, and the user study. Source code is available on the RWTH GitLab for any signed-in users[3].

The next sections provide an overview of the relevant functionalities of XR that we use and present the logic of our custom code.

3.1 Interaction

Interaction in XR is handled by *interactors*. These interactors provide different methods of interaction and they function with the concept of “hover” and “select.” The interactors constantly detect and maintain a list of all interactable targets it can “see”, these can be set to the "hovered" state if the conditions of the interactor are fulfilled. If a subsequent input action occurs, e.g. a controller trigger press, it then attempts to “select” that same target, which activates the object’s functionality. For example, when the button is in the "hover" state, it can become highlighted to provide visual feedback to the user. When the user attempts to interact with it by pressing the controller’s trigger, the button will enter

[1] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.2/manual/index.html
[2] https://docs.unity3d.com/Packages/com.unity.xr.hands@1.4/manual/index.html
[3] https://git.rwth-aachen.de/jan-tugsbayar/master-thesis

the "selected" state and the button's `OnClick()` event will be fired.

In order to be able to interact with an object, it requires an interactable component, the components that we use are `XR Simple Interactable` or `XR Grab Interactable`. All interactable components are derived from `XRBaseInteractable`. Similarly, all XR interactors are derived from `XRBaseInteractor`. This provides seamless implementation of various possible interactions using different interactors.

3.1.1 XR Interactors

XR supplies a family of *Interactor* components for different methods of interactions. We utilize three of the included interactors in our VR design and we provide a high level overview of their functionalities.

Poke Interactor

`Poke Interactor` provides the ability to "poke" objects as the simplest form of interaction. It uses an attach transform, which serves as the point of interaction, the default setting for the attach is the index finger's fingertip, and we left it so, as it is the most natural way to poke objects. When this point of interaction is near the vicinity of an interactable object, it will enter its hover state. Afterwards, if the point collides deeper into the object, a selection will be performed.

Near–Far Interactor

The `Near-Far Interactor` provides both long range and close range grabbing of objects and UI interaction. The long range function works by utilizing a `Curve Interaction Caster`, which casts a curved ray to detect valid interactable objects. The close range function is performed by `Sphere Interaction Caster`, which casts a sphere around the interactor that detects valid objects. When the sphere or curve caster hits a valid grabbable object, the user can then perform a select. The interactor can be used in both controllers and hand tracking. When using the controllers, the grab button is used to grab objects, while the trigger is used for UI interaction. In case of using hands, both UI interaction and grabbing are done by performing a pinch gesture using the index finger and the thumb.

To assist with aiming, the near–far interactor provides a visual ray originating from the user's hand. When grabbing an object, the object stays attached to this ray as long as the pinch pose is held. Moreover, with `Interaction Attach Controller` it is possible to pull closer or push away grabbed objects by performing high-velocity pulling or pushing movements with the hand or the controller.

Direct Interactor

`Direct Interactor` provides simple grabbing of objects when direct contact with the object is made, similar to the Near–Far Interactor's close range grabbing. It can be attached to both the controllers and the hands. It functions by casting a sphere collider around it, similar to the near–far interactor. When this sphere collides with an object that has the `XR Grab Interactable` component and the user performs a grab, the object will be attached and held by the interactor. In case of controllers, the grab is performed by pressing the grab button. When using hand tracking, the grab is performed by pinching the index finger and thumb similar to the near–far interactor's close range grab. In other words, the direct interactor implements the close range grabbing function of the near–far interactor. The

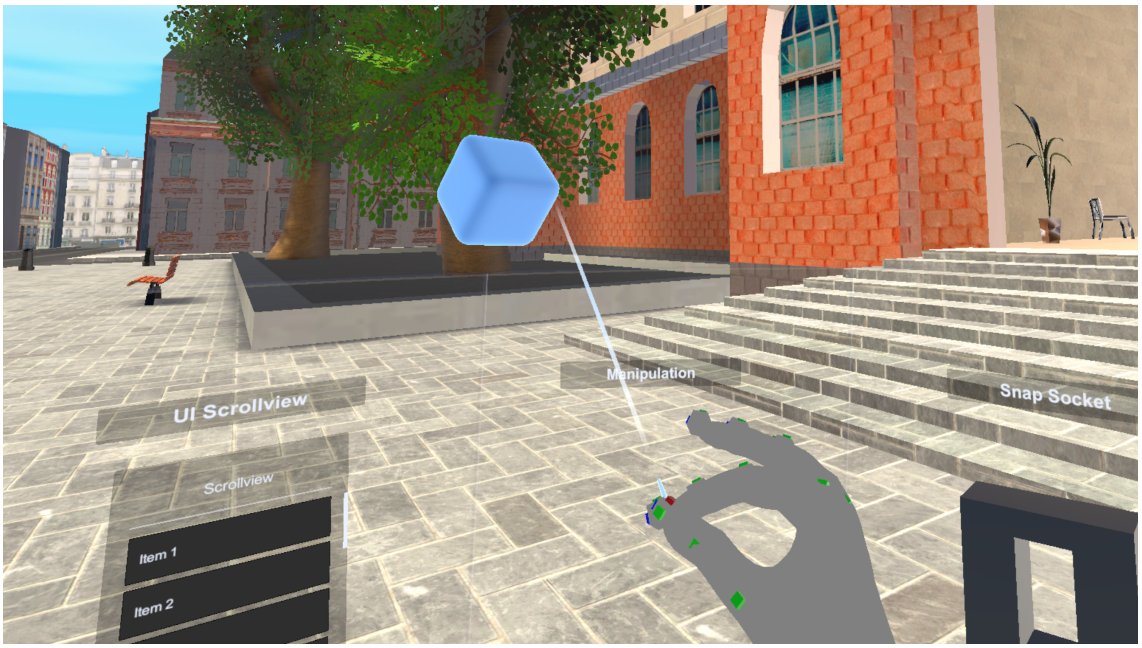


Figure 3.1: Near–Far Interactor's long range grab in hands mode using the ray and pinching.

direct interactor also supports two handed grabbing, handing off items between hands, and scaling of objects if it is enabled.

3.2 Custom Gaze Interactor Implementation

To incorporate gaze tracking in a meaningful way, we decided to create a new gaze interactor. Our gaze interaction is implemented as `CustomGazeInteractor` and is based on XR's existing `gaze interactor`[4].

The included gaze interactor performs a ray-cast to scan for interactable objects within a ray-cast range that have `XR Simple Interactable` or `XR Grab Interactable` components attached with `Allow Gaze Interaction` enabled. When aiming with gaze at such an object, it activates XR's hover function and after a set amount of delay, it can be selected. This allows for interacting with UI elements and with grabbable objects. We initially planned to use this included gaze interactor as it is. However, when testing out this functionality, we felt that it was inadequate, as the forced delay causes unnecessary wait and frustration until selection occurs. Moreover setting the delay too small caused accidental interaction when looking at objects. Additionally interacting with objects, with no involvement of hands felt unnatural, as the vast majority of interactions with the environment is performed using our limbs. Since we want to combine hand tracking with gaze tracking, we ultimately decided to create our own implementation of a gaze interactor that integrates hand gestures similar to the Meta Quest's UI interaction in its menu.

Our custom gaze interactor functions similar to the base gaze interactor in detecting interactable objects using a ray-cast. However, instead of using a delay to activate select, we use the hand gesture of pinching to perform selection. When aiming at simple interactable

[4]https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.2/manual/xr-gaze-interactor.html

UI elements, such as a button or a toggle, pinching performs immediate selection of the UI element. However, more advanced UI elements, such as a slider or scrolling, need to be handled differently. When aiming at a scrollable element or a slider, the user needs to perform the pinch gesture and keep holding it. After a set delay of 0.7 seconds, the custom gaze interactor enters sliding mode. While still pinching, the user is then able to move their head, i.e. their gaze, up or down in case of scrolling or to the sides when interacting with a slider. As long as they keep pinching, their head movement will adjust the scroll and slider. To exit slider mode, the user simply ceases the pinch gesture.

To achieve this behavior, we have modified and extended the existing gaze interactor. Our modifications include how "select" is activated by detecting a pinch and how it is exited. We also handle scrolling and slider interaction using gaze and pinch. To provide an overview of the logic of the custom gaze interactor, we provide a code snippet of the `ProcessInteractor` function in Listing 1. The `ProcessInteractor` is called by XR to update the Interactor after interaction events occur. In order to introduce pinch gesture selection, we override the base function.

The detection of pinching when using Meta Quest 3 is done via `metaGestureDetectors`. We iterate over the two `metaGestureDetectors` (one for each hand) and look for the `bool isIndexPinching` flag, as visible in lines 4 to 10 in the code snippet Listing 1. If it is true, a pinch was detected. If the interactable element we are aiming at is a grabbable object or a UI button (or a toggle), then the `PerformGestureSelect()` function is called on line 17. That function mimics the base gaze interactor for UI elements, but for grabbable objects, it attaches the object to the user's hand by simply adjusting the object's transform and forcing it to follow our hand as long as the pinch is still held. When the pinch gesture is no longer detected, indicated by `!anyIndexPinching` and we have a currently active select object on line 21, we exit the select by calling `SelectExit` on line 23.

Slider interaction is also initiated by `PerformGestureSelect()`, though it calls the `IEnumerator HandleSliderInteractionUsingYaw(Slider slider)` coroutine, which adjusts the slider's value based on the player camera's yaw angle. When a pinch is no longer detected, we stop the coroutine on line 26.

The player's point of view camera's current yaw is saved as `currentYaw = Camera.main.transform.eulerAngles.y`, any changes to the yaw angle are calculated as a delta: `float deltaYaw = Mathf.DeltaAngle(InitialYaw, currentYaw)`. Then the slider's new value can be calculated using the delta value as: `float newValue = InitialSliderValue + deltaYaw * SliderSensitivity`. In order to not force the user to make giant head rotations, we use `SliderSensitivity` as a multiplier to the slider value adjustments.

The scroll interaction functions analogously, the difference being in place of using the y-axis angle as in sliders, we use the x-axis angle `Camera.main.transform.eulerAngles.x`. To exit scrolling, the user simply ceases to perform the pinch gesture, this is detected on line 21 and if the scrolling coroutine is active, it is stopped on line 26. Commonly, scrolling is performed vertically, while sliders are horizontal. In theory, it is possible to use them at any angle, though our implementation is limited to only using them in their common orientation.

To summarize, the custom gaze interactor constantly scans for interactable objects using a ray-cast. It can differentiate between grabbable objects using `XR Grab Interactable`, simple interactable objects such as doors in the house task using `XR Simple Interactable`, simple UI elements such as buttons and toggles, and lastly advanced UI elements such as scrolling and sliders. For simple UI elements and `XR Simple Interactable`, the base XR code is called to handle it. For the rest, additional new code is required to attach grabbed objects to the hand and to handle scroll and sliders.

Listing 1 Custom Gaze Interactor's Process Interactor Snippet.

```
public override void ProcessInteractor(XRInteractionUpdateOrder.UpdatePhase updatePhase{
    base.ProcessInteractor(updatePhase),
    if (updatePhase == XRInteractionUpdateOrder.UpdatePhase.Dynamic){
        bool anyIndexPinching = false,
        foreach (var detector in metaGestureDetectors){
            if (detector != null && detector.isIndexPinching){
                anyIndexPinching = true,
                break,
            }
        }
        //If slider mode is active, a new pinch exits slider mode regardless of current UI hit.
        if (IsSliderAdjusting && anyIndexPinching && !WasAnyIndexPinching){
            EndSliderInteraction(),
        }
        //Otherwise, if not in slider mode, a new pinch triggers the normal selection routine.
        else if (anyIndexPinching && !WasAnyIndexPinching){
            PerformGestureSelect(),
        }

        //Exit select when not pinching anymore.
        if (!anyIndexPinching && base.isSelectActive && !IsSliderAdjusting && !isScrollingUI){
            if (interactionManager != null && firstInteractableSelected != null){
                interactionManager.SelectExit(this, firstInteractableSelected),

                if (scrollCoroutine != null){
                    StopCoroutine(scrollCoroutine),
                    scrollCoroutine = null,
                }
                isScrollingUI = false,
            }
        }
        WasAnyIndexPinching = anyIndexPinching,
    }
    //If an object is currently selected, disable the line visuals.
    if (isSelectActive){
        if (InteractorLineVisual != null && InteractorLineVisual.enabled){
            InteractorLineVisual.enabled = false,
        }
        if (InteractorLineRenderer != null && InteractorLineRenderer.enabled){
            InteractorLineRenderer.enabled = false,
        }
    }
}
```

3.3 Base Locomotion

Locomotion in XR is handled by *Locomotion Providers* and the `Locomotion Mediator`, which serves as mediator between multiple locomotion providers. Continuous locomotion uses the `Continuous Move Provider`, which enables continuous movement when using the controllers. By moving the joystick on the left controller, the user is able to move in the joystick's direction. Turning is achieved by moving the right controller's joystick to the sides. It can be either continuous, which is provided by `Continuous Turn Provider`, or as snap turns at predefined angles by the `Snap Turn Provider`.

Another common locomotion method in VR is teleportation, this is performed by the `Teleportation Provider`. Teleportation is performed by moving the right controller's joystick forward, which activates a teleportation ray originating from the right controller's visualization. The user is able to aim with a ray to choose their desired destination. When the joystick is released back into its neutral position, teleportation occurs.

Our goal is to emulate these functionalities of the baseline locomotion of controllers using hand tracking and gaze tracking. Hence, we designed gesture locomotion and wrist teleportation.

3.4 Gesture Locomotion Implementation

Initially, gesture locomotion was implemented as a standalone script called `GestureMoveProvider`, that inherited from `ContinuousMoveProvider`. It was responsible only for gesture locomotion, joystick movement was handled by `DynamicMoveProvider`, which also inherits `ContinuousMoveProvider` and is part of the XR sample assets. During implementation, we encountered a problem of movement speeds between joystick and gesture locomotion differing, we initially thought that was because of conflicting scripts inheriting and accessing the same parent script. To alleviate this issue, we combined all the locomotion functionalities into a single object and script called `FullMoveProvider`. This script handles keyboard locomotion, joystick locomotion, and gesture locomotion with several steering inputs as described in Section 2.2.2. Albeit this did not solve the problem of differing movement speeds, which was later identified as multiple update calls within a single frame. However the result of having a single script handling all types of locomotion, makes it is easier to integrate it into other projects.

To explain the flow and logic of `FullMoveProvider`, we provide two code snippets. The first one is Listing 2, which shows our Unity update function. As `FullMoveProvider` inherits from `ContinuousMoveProvider`, we shadow the base script's Update similar to the gaze interactor, in order to introduce our logic. We use control variables `bool isJoystickActive`, `bool isKeyboardActive`, and `bool isGestureActive` to control the logic of our script. When the gesture detection system recognizes our forward or backwards movement gesture, the variables `VRMovementSpeed` and `VRBackwardMovementSpeed` are incremented respectively, this sets `bool isGestureActive` to true, indicating we are using gesture locomotion. The joystick and keyboard variables are set to true whenever a joystick or keyboard input is detected.

To determine the direction of movement, XR uses a `Vector3`, a 3-dimensional vector like Unity's 3D coordinate system. We initially set our movement direction variable `combinedTranslation` as a zero `Vector3`. When joystick input is detected, we compute the direction based on the `joystickInput` via the `ComputeDesiredMove(joystickInput)` and save it into our `combinedTranslation`, the same is done if keyboard input is detected using the `ComputeKeyboardMove()` function. In case of a gesture being active, we compute the

Listing 2 FullMoveProvider's Update function.

```
protected new void Update(){
    //Get input from joystick, keyboard, and gestures. ReadInput() isn't
    ↪  available in this context, so we calculate joystick input manually.
    Vector2 joystickInput = leftHandMoveInput.ReadValue() +
    ↪  rightHandMoveInput.ReadValue(),
    bool isJoystickActive = joystickInput != Vector2.zero,
    bool isKeyboardActive = Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.A) ||
                            Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.D),
    bool isGestureActive = (VRMovementSpeed >= 1 || VRBackwardMovementSpeed >=
    ↪  1),

    //Movement vector.
    Vector3 combinedTranslation = Vector3.zero,

    //If no keyboard or gesture input is active, then joystick movement.
    if (!isKeyboardActive && !isGestureActive && isJoystickActive){
        //combinedTranslation += base.ComputeDesiredMove(joystickInput),
        combinedTranslation += ComputeDesiredMove(joystickInput),

        //ComputeDesiredMove
    }

    //Keyboard movement if active.
    if (isKeyboardActive){
        combinedTranslation += ComputeKeyboardMove(),
    }

    //Gesture movement if active.
    if (isGestureActive){
        // Pass Vector2.zero to trigger the gesture branch.
        combinedTranslation += ComputeDesiredMove(Vector2.zero),
    }

    //Apply movement (once per frame)
    if (combinedTranslation != Vector3.zero){
        MoveRig(combinedTranslation),
    }
    else{
        TryEndLocomotion(),
    }
}
```

direction again using `ComputeDesiredMove(Vector2.zero)`, however with a zero vector as parameter. The function is able to detect when a gesture locomotion is used and ignores the Vector2.zero, it calculates and returns a new vector based on the steering mode's source object and will be explained in the next snippet Listing 2. Lastly, once the direction of locomotion is calculated, the player object is translated, i.e. moved, using the calculated vector via XR's `MoveRig(combinedTranslation)`.

The next snippet Listing 3 features the `ComputeDesiredMove()` function that calculates

the direction of movement. We omit the branch for joystick movement, as that is simply XR sample asset's joystick movement and it returns the base function's execution.

The gesture branch features control variables `bool isForwarding`, `bool isBackwarding` to determine whether we are moving forward or backwards. Another variable `bool isHandBackwarding` is used for backwards locomotion when using hand steering. At first, we determine the direction of movement as 2-dimensional `Vector2 direction2D` using the `DetermineDirection2D(input, isForwarding, isBackwarding)` function on line 15, which simply returns a `Vector2(0f, 1f)` for forwards movement and `Vector2(0f, -1f)` for backwards movement. Afterwards, it gets converted into a `Vector3 baseMove` via calling `base.ComputeDesiredMove(direction2D)`.

In case we are using gaze steering with gesture locomotion, the player object's head transform is set as the player's camera and adjusts the `m_CombinedTransform`, which is an internal transform variable used by the joysticks on line 31. Effectively, this replaces the joystick input by the player's gaze.

When using hand steering, the `HandSteering` function is called, which uses the player's hand object's transform as a source of direction. The transform is a `Vector3`, however, its y-axis has a value of 0 and is normalized, as we do not allow flying or other types of vertical movement.

The `FullMoveProvider` also contains logic to determine which hand initiated the gesture locomotion. In case both hands are used and one of them ceases performing the gesture, it is able to set the other hand as the initiating hand. Additionally, using both hands will set the aforementioned `VRMovementSpeed` and `VRBackwardMovementSpeed` values to 2, which is used to perform a sprint. The sprint can be applied as a multiplier to the `moveSpeed` variable on line 20 in Listing 3. However, this function was disabled in the user study, similarly, the `LockedDirectionSteering` is also disabled.

As presented, the gesture locomotion attempts to use as many base XR functions as possible. It uses the base code for joystick input and simple keystroke detection to implement keyboard movement. The gesture detection system invokes functions that track which hand has initiated the gesture, how many hands are used, and in which direction the user wants to move. Using this system, it calculates the direction of movement either using the transform of the hands for hand steering or the transform of the user's head for gaze steering.

3.5 Interaction Modes

Since the main goal of this thesis is to evaluate the feasibility of VR interaction and locomotion without controllers, we defined three *interaction modes* by combining the existing XR interactors and locomotion with our own custom gaze interactor and the gesture locomotion. These interaction modes are called: `controller`, `hands`, and `hands with gaze`. An overview of each interaction mode's components can be viewed in Table 3.1.

To address the second research goal of this thesis, whether adding gaze tracking is beneficial and how it could be implemented to supplement hand tracking we created two different interaction modes without controllers. The hands mode has no gaze tracking incorporated, while hands with gaze incorporates gaze tracking in two manners: namely as an interaction method using the custom gaze interactor and as a steering input for locomotion.

When creating these interaction modes, we followed the principle of keeping the same capabilities in each mode. The controller mode serves as a baseline for this purpose. Its near–far interactor provides the ability to interact both from a distance and directly in close

	Controllers	Hands	Hands with gaze
Poke interactor	✓	✓	✓
Direct interactor			✓
Near–Far interactor	✓	✓	
Custom Gaze interactor			✓
Locomotion	Controllers	Gesture	Gesture
Steering	Controllers	Hand steering	Gaze steering

Table 3.1: Interaction modes with their corresponding active interactors and locomotion method.

range. To retain these abilities, the hands with gaze utilizes the custom gaze interactor as a substitute for long range interaction, meanwhile, the direct interactor substitutes as the close range interactor. The hands mode utilizes the same interactors as controllers, although adapted for hand tracking, so it has the same capabilities as the controller.

This principle is applied to locomotion as well: gesture locomotion provides both forward and backwards movement just as the controllers. Following this principle, each interaction mode provides long range interaction, close range interaction, poking, locomotion in both directions, and steering. The only feature of the controllers not present in the other interaction modes is the ability to do a 180° turn. We found this a niche feature that even experienced VR users were not aware of, hence, we decided not to duplicate this feature using gestures. However, adding such a feature would be trivial and should be considered in the future.

To summarize the design of VR mechanics, we have created two additional interaction modes by creating a custom gaze interactor and gesture based locomotion. These two interaction modes are called hands and hands with gaze. One of them integrates gaze tracking while the other does not. They can be viewed as equivalent to the classical controllers. We use these interaction modes to answer our research question of determining the feasibility of VR without controllers.

Listing 3 Snippet of ComputeDesiredMove(Vector2 input)

```
protected override Vector3 ComputeDesiredMove(Vector2 input){

    bool isJoystickActive = input != Vector2.zero,
    //Joystick branch
    if (isJoystickActive){
    .
    return base.ComputeDesiredMove(input),
    }
    //Gesture branch (when joystick input is zero):
    else{
        bool isForwarding = (VRMovementSpeed > 0) && (VRBackwardMovementSpeed < 1),
        bool isBackwarding = (VRBackwardMovementSpeed > 0) && (VRMovementSpeed < 1),
        bool isHandBackwarding = (VRBackwardMovementSpeed > 0) && backwardsGesturesEnabled && (VRMovementSpeed < 1),

        Vector2 direction2D = DetermineDirection2D(input, isForwarding, isBackwarding),

        float originalSpeed = moveSpeed,
        Vector3 baseMove = base.ComputeDesiredMove(direction2D),

        moveSpeed = originalSpeed,

        if (gazeSteeringEnabled){
            ResetLocks(),
            //Update the forward source to follow the head (gaze)
            if (m_HeadTransform == null){
                var xrOrigin = mediator.xrOrigin,
                if (xrOrigin != null && xrOrigin.Camera != null)
                    m_HeadTransform = xrOrigin.Camera.transform,
            }
            if (m_HeadTransform != null){

                m_CombinedTransform.SetPositionAndRotation(m_HeadTransform.position, m_HeadTransform.rotation),
            }
            return baseMove,
        }
        else if (handSteeringEnabled && RightHandAiming != null && LeftHandAiming != null){
            ResetLocks(),
            return HandSteering(baseMove, isHandBackwarding),
        }
        else{
            return LockedDirectionSteering(baseMove, isForwarding, isBackwarding),
        }
    }
}
```

Chapter 4

User Study

To evaluate our proposed interaction modes, we performed a user study. This chapter describes how we designed our user study and how it is used to determine the feasibility of our proposed implementation. We also give some insight into the implementation aspects of the user study.

The study consists of three parts: *VR session with tasks*, *questionnaire* and an *interview*. For the VR session we created a user-study VR application for the participants to take part in. It consists of five simple tasks, each covering a specific aspect of VR interaction and locomotion. Participants performed these tasks in three separate *rounds*, each round used a different *interaction mode* as explained in Section 3.5. The order of the interaction modes and tasks is *randomized*, the tasks themselves feature a *randomization mechanic* to ensure that it does not exactly match the previous round. Each task in the VR application included *metrics*, which automatically measured certain values, for example the time required to perform the task. This data forms the basis of our *quantitative analysis*. However doing the same tasks thrice can lead to a learning effect despite their randomization mechanics. How we interpret and analyze the data while taking this into account is explained in Section 5.2.

The *questionnaire* was another form of quantitative metric and it was carried out using Google Forms, in which the participants had to rate their VR experience. Lastly we performed an optional semi-structured recorded *interview*, with the participants to gain more personal feedback as a basis for the qualitative analysis. As the interview is recorded and the questionnaire asks for some personal information to form demographical data, all participants signed a written consent to collect and use their data for the purpose of the study with GDPR compliance upon arrival. Afterwards the participants were given an explanation, that the study consists of three parts. We will now cover each of the described study parts in more detail.

We conducted an initial pilot phase of the study with staff members of the LFI institute, who were unaware of the project. Subsequently, we recruited volunteers from the wider RWTH community by circulating flyers and inviting friends and acquaintances. Interested individuals scheduled appointments to participate. The participants performed a VR session with our user study VR application. For this purpose we used the Meta Quest 3, as all the development and testing was performed on it, moreover it is an affordable device that features hand tracking, which suits the end goal application of our VR interaction implementation in an educational setting, which requires a high amount of affordable VR devices.

4.1 Initialization

Throughout the VR session, participants were overseen by a study supervisor. The supervisors were instructed to avoid disclosing the purpose of the study and engage in off-topic conversations, however if participants encountered difficulty, hints and explanations were allowed.

Upon donning the VR HMD and starting the application, the participants could see a UI with some basic instructions and another UI stating the current interaction mode. The instructions stated that a locomotion tutorial video will be played when they are ready. The tutorial video can be started by the supervisor by pressing a key on the keyboard. After watching the locomotion tutorial video, the participants had some time to familiarize themselves with the locomotion method. Afterwards they would proceed to the next area, which included a demonstration table containing buttons, sliders, toggles, scrollable panels, poke objects, and sockets taken from the XR Starter Assets. Here another tutorial would be played about how to interact with objects and UI elements in the current interaction mode. As with locomotion, the participants now had some time to try out the interactions on the demonstration table.

Figure 4.1: From left to right: Current tracking mode, instructions, tutorial video.

The tutorial videos were in in-application footage demonstrating the locomotion and interaction with accompanying AI generated English voice-over. AI voice was used to ensure a clear pronunciation comparable to a native speaker. English was chosen as the language for the entire study, in order to accommodate as many people as possible, even those that did not speak German.

After completing a round with all five tasks, participants were teleported back to the start location, the next interaction mode was automatically activated, and the cycle repeated until all three modes were performed. At the end of the third round, a clear message was shown on screen to the participants that the study is finished, afterwards they could remove the HMD and proceed to fill the questionnaire, with a possible break if required.

4.2 Tasks

For tasks we provide a high level overview of the tasks and highlight some of their implementation aspects, including the metrics measured by the tasks. Each task had accompanying instructions and a "Start" button next to it. Pressing the start button started the task and the timer metrics of the task. Upon finishing a task, a message was shown to the user with the text "Task finished!" while a positive feedback audio was played to clearly indicate that the task was finished. The tasks were spread out in the virtual environment, which was a virtual recreation of the Templergraben street in Aachen. As the task order is randomized and we do not want to confuse the participant, only the currently active task is visible at any given moment, the rest of the tasks are invisible.

4.2.1 Simon Says Task

SimonSays implements a memory-game inspired by the electronic version of the children's game Simon Says. Unlike the original game, we have implemented it as a 3×3 num-pad grid of Unity interactable buttons, which can be seen in Fig. 4.2. Participants can press the buttons using one of their current interactors available to them in the given round. As a reminder, the table of available interactors in each round can be viewed in Table 3.1. There are nine buttons in total and each button has an internal index assigned ranging from 0 to 8 and the buttons have the ability to be highlighted by changing its color. The goal of the game is to input a button sequence, which will be shown to the participant by the buttons being highlighted. There are multiple rounds of the game, with each round adding a new button to the previous sequence. The length of the sequence in the first round is two buttons. The game is started when the participant presses a "Start" button. Afterwards there is a small delay of 1.5 seconds, to give participant the time to focus and be ready to notice and memorize the sequence. Between each round there is also a delay of 1.5 seconds.

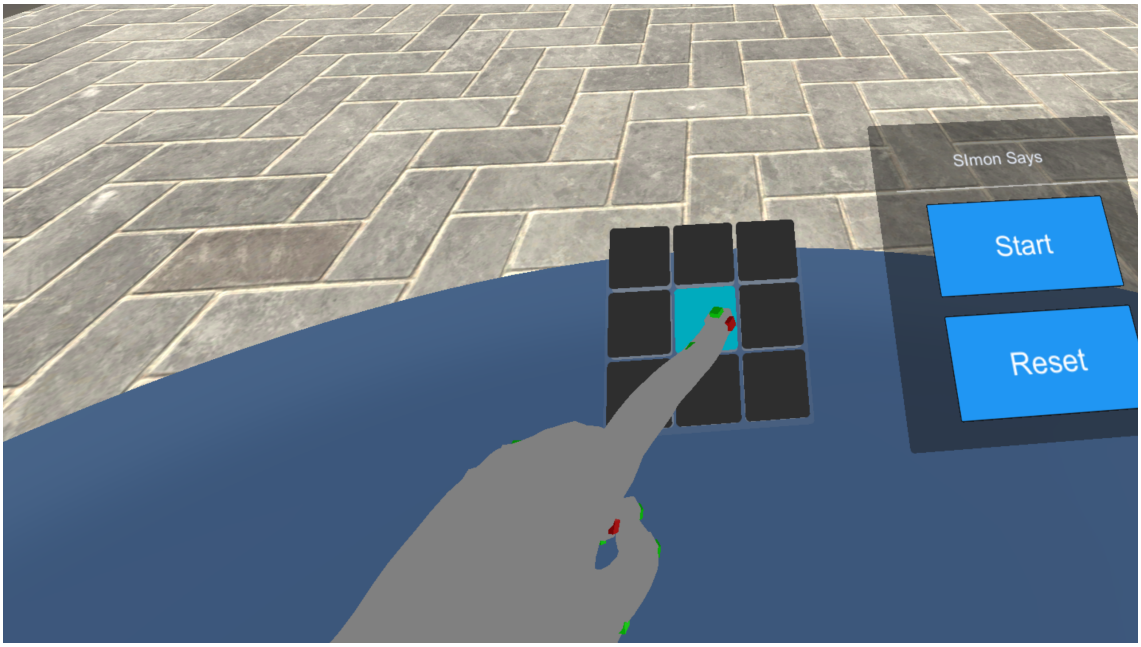


Figure 4.2: Simon Says buttons.

In early internal testing the grid buttons were labeled with numeral labels (1 to 9)

and the game consisted of 5 rounds, with the initial sequence being 2 long as above, so the total sequence length was 6. However, testing proved that using the numeral labels allowed the users to simply memorize the number sequence, which made the task too trivial. We therefore removed all numeral labels, which forces the participants to solely rely on memorizing the buttons based on spatial location or other memorization tricks. This change revealed that memorizing and reproducing a sequence of 6 over an unlabeled 3×3 grid, with possible repeats, proved excessively difficult. This resulted in frequent failures and restarts. To strike a balance between challenge and feasibility, we reduced the maximum sequence length to 4, yielding three rounds of increasing difficulty. Moreover, initially the button sizes were bigger, this proved to be too easy to accurately press, hence the buttons were made smaller and closer together with minimal empty space between them, as visible in Fig. 4.2.

While the study is not purposely designed to be stressful, having to memorize the sequence when playing a simple game can be stressful, but principally it induces an active cognitive load on the participants, to see whether it has an affect on the participant's performance. The task is designed to impose a moderate cognitive load by forcing the participants to continuously memorize and recall a lengthening sequence, while simultaneously requiring fine motor skills and precision to interact with buttons, which can be widely employed for UI interaction or other types of interaction in VR. The buttons were closely spaced, hence requiring finesse and accuracy to operate them. By comparing performance across the different interaction modes, we aim to determine the effect of each interaction mode on the user's motor dexterity and finesse.

We require some method to measure the performance of the participant. For that purpose each task has metrics, which are automatically measured by the user study VR application. The resulting metric of each task is saved inside the application and at the end of each round, they are exported as a `.csv` file. Afterwards they are reset for the next round. SimonSays has the following metrics:

- *totalTime* (T_{total}): elapsed time from the initial start to task completion, including enforced delays.
- *forcedDelayTime* (T_{delay}): total of the inter-round delays, $T_{\text{delay}} = (\text{round count}) \times (\text{roundDelay})$.
- *netTime* (T_{net}): participant response time excluding enforced 2 second delays between rounds, $T_{\text{net}} = T_{\text{total}} - T_{\text{delay}}$.
- *userClicks* and *userWrongClicks*: total button presses and those that mismatched the expected sequence.
- *averageInterClickTime* ($\overline{\Delta t}$): mean interval between successive clicks,

  $$\overline{\Delta t} = \frac{\sum_{i=1}^{N_{\text{userClicks}}} \Delta t_i}{N_{\text{userClicks}}},$$

  where Δt_i is the time between click $i-1$ and click i.
- *restarts*: number of restarts caused by failures to correctly input the sequence.

Implementation Aspects of Simon Says

Upon pressing a button, it invokes the `OnPlayerPress(int buttonIndex)` function in `SimonSaysManager`, which enables the script to detect which button is being pressed.

The logical procedure of the game is as follows:

1. The script generates an initial random sequence of length $L_0 = 2$ and saves it into `List<int> buttonSequence`. Randomization is performed simply by adding numbers from the interval $(0, 9)$ using Unity's `Random.Range`. Note that `Random.Range` with integers is max exclusive.

2. The `IEnumerator ShowSequenceRoutine()` routine is started. After a grace period of 2 seconds, each button with the matching index is highlighted for 0.6 seconds, with a pause between each highlight of 0.2 seconds. During the highlight routine button inputs are ignored.

3. After the highlight routine is finished, participants' button presses are detected via `OnPlayerPress(int buttonIndex)`. The parameter `buttonIndex` is the button's index. The function compares the button's index with the value at the current position in `List<int> buttonSequence`. The position within the list is controlled using the `playerInputIndex` variable, which is initialized as 0 and is incremented after each correct button press.

4. If the entire sequence is reproduced correctly, a positive audio feedback is played and the script advances to the next round by adding a new random number at the end of the `List<int> buttonSequence`. Then the newly expanded sequence is highlighted. The number of rounds is 3, hence the last round's sequence length is 4.

5. On an incorrect button press, all the buttons enter a short blinking routine, where each button blinks 3 times for 0.15 seconds with a 0.15 second gap between each blink. Additionally a negative feedback audio is played, and the state of the game is reset and automatically restarts with a new `List<int> buttonSequence` sequence to be generated.

4.2.2 Movement Task

The *Movement Task* involves movement over a set of circular "plates" in the virtual environment, which can be viewed in Fig. 4.3. We have a set of five circular plates in total. Each plate can be highlighted by turning bright yellow and a white chevron appears above it. The task of the participant is to step onto the highlighted plate, until they have stepped on each plate. Only a single plate can be highlighted at a time and we do not allow repetition of plates. The five plates were spread out over a small area, as visible in Fig. 4.3.

This movement task is designed to measure locomotion accuracy and steering capabilities under the different locomotion modes. For this purpose, we calculate the in-game distance traveled by the participant and save it as `float userDistanceTraveled`.

The function `ComputeFullOptimalPath()` computes the minimal "edge-to-edge" distance the participant would need to travel to visit all N circles in the prescribed order, starting from the initial position. Mathematically it can be defined as: Let

$$\{ c_0, c_1, \ldots, c_{N-1} \}$$

be the sequence of circle indices, and for each circle c_i let

$$C_i \in \mathbb{R}^3 \quad \text{and} \quad r_i > 0$$

denote its center and radius. Then denote the player's start position with $P_{\text{start}} \in \mathbb{R}^3$, where $\mathbb{R}^3$ represents the 3-dimensional coordinate system in Unity. The start position

Figure 4.3: Movement task plates.

represents the location when the participant pressed the "Start" button of the task. Then the optimal path D_{opt} is

$$D_{\text{opt}} = d_{start} + \sum_{i=0}^{N-2} d_i,$$

where

$$d_{start} = \max\bigl(0, \| P_{\text{start}} - C_0 \| - r_0\bigr)$$

is the distance from the start point to the edge of the first circle, and for each consecutive pair of circles

$$d_i = \max\bigl(0, \| C_i - C_{i+1} \| - (r_i + r_{i+1})\bigr), \quad i = 0, \ldots, N-2,$$

is the non-negative straight-line distance between their perimeters.

By comparing actual travel paths against the optimal straight-line distances and recording time, we can assess how each input method affects participants' ability to move and turn precisely towards a location precisely in VR. Initially stepping on plates was implemented as a collider, however, for the purposes of the calculations, it was reimplemented by calculating the distance of the participant to the center of the circular plate and subtracting the radius of the plate. With that, the full metrics of the movement task are:

- *totalTime* (T_{total}): elapsed time from Start to task completion.
- *averageArrivalTime* ($\bar{t}_{\text{arr}}$): mean time between successive correct plate visits,

$$\bar{t}_{\text{arr}} = \frac{1}{N} \sum_{k=1}^{N} t_k,$$

  where $t_k = \Delta t$ recorded when stepping on plate k, and $N = |\texttt{intermediateArrivalTimes}|$.

- *userDistanceTraveled* (D_{user}): cumulative path length sampled each frame,

$$D_{\text{user}} = \sum_{f=1}^{F} \left\| \mathbf{p}_f - \mathbf{p}_{f-1} \right\|,$$

  where $\mathbf{p}_f$ is the player position at frame f, and F is total frames during the task.

- *optimalPath* (D_{opt}): sum of straight-line distances from the start position to each circle edge and between successive circle edges,

$$D_{\text{opt}} = d_{-1} + \sum_{i=0}^{N-2} d_i,$$

- *userWrongPlates* (C_{wrong}): total count of incorrect plate steps.

Implementation Aspects of the Movement Task

Stepping on a plate invokes `HandleCircleStepped(int circleIndex)`, where the parameter `circleIndex` is the plate's index number. Upon pressing the "Start" button, the logical procedure of the task is:

1. A random permutation of the N plates is generated (here $N = 5$) and stored in `List<int> circleSequence`. As we cannot have repetitions, we simply create a list with the numbers 0 to 4 using a for loop and perform **Fisher–Yates** [FY48] shuffle on the list.

2. The plate with the same index as the first number in `List<int> circleSequence` is then highlighted.

3. The participant must move and step onto the highlighted plate, which invokes `HandleCircleStepped(int circleIndex)`.

   - If they step on the correct plate, it stops being highlighted, a positive audio feedback is played, and the next plate is highlighted.
   - If they step on the wrong plate, a negative feedback audio is played, and all plates blink 3 times by being changed into their highlighted color. Each blink is 0.3 seconds long and there is a gap of 0.3 seconds between each blink. This is to ensure that the participant is aware of their mistake. There is not a restart mechanism like in SimonSays, as we are not attempting to induce a cognitive load of memorizing a sequence, we simply want to measure the accuracy of the movement

4. The task is completed once the participant has correctly visited all plates in their highlighted order.

4.2.3 Shapes Task

The *Shape Task* implements a shape matching puzzle in which the participant must locate and insert objects into sockets with the same shape. There are 6 uniquely shaped empty sockets on a table. The socket mechanic allows objects to be placed inside them by putting the object inside. In our case we allow only objects with the same shape to be placed into

the sockets. Upon inserting the correct object, it stays inside the socket in a predefined transform and rotation. For each socket shape, we have 3 corresponding objects with the same shape but of different sizes: small, medium and large. The medium sized object is always the correct size and is called the `PuzzleObject`. The incorrectly sized objects are called `fakeObjects`.

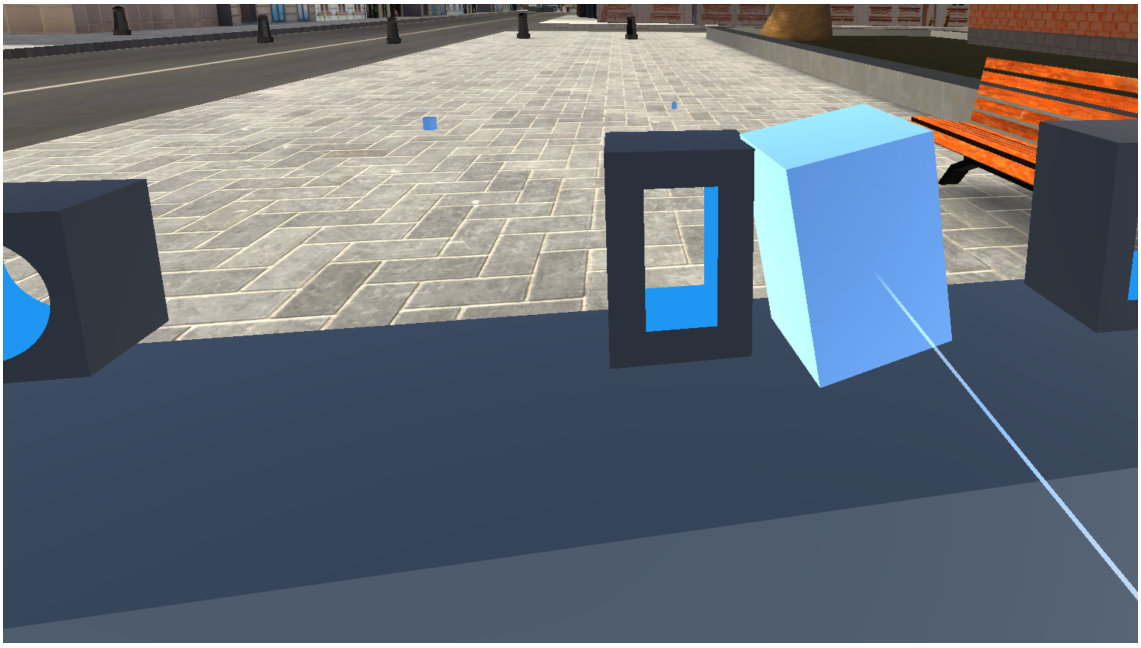


Figure 4.4: Rectangular cuboid socket with a large fakeObject of the same shape. Other objects are visible in the background.

This task evaluates the integration of grabbing dexterity and locomotion. Participants must be able to both navigate and accurately manipulate small objects, distinguishing correct from incorrect size variants. To evaluate the performance of the participants we use the following metrics:

- *totalTime* (T_{total}): elapsed time from start until all sockets are filled.
- *totalGrabs* (G_{total}): number of object grabs performed.
- *wrongAttemptsCount* (A_{wrong}): number of insertions into incorrect sockets.
- *avgFillTime* ($\bar{t}_{\text{fill}}$): mean time to place each correct piece:
  - $$\bar{t}_{\text{fill}} = \frac{1}{N}\sum_{i=1}^{N} f_i, \quad f_i = t_i - t_{i-1}, \quad N = \left|\texttt{socketFillTimes}\right|.$$
  - t_i is the absolute timestamp (in seconds) when the i-th socket was successfully filled (recorded in code by logging 'Time.time').
  - f_i is the interval (in seconds) between the $(i-1)$-th and i-th fills, i.e. the difference $t_i - t_{i-1}$.
  - N is the total number of fills (length of `socketFillTimes`), here $N = 4$.

4.2.4 Implementation Aspects of the Shapes Task

1. 4 out of the 6 shapes are chosen randomly.
2. After the shapes are chosen, the corresponding correct `PuzzleObject` and the 2 `fakeObjects` are randomly distributed near the table using a set amount of fixed spawn points. The rest of the objects are left inactive.
3. The participant may grab any object and attempt to place it into a socket.
4. If a piece does not match the socket's required shape, a negative feedback audio is played, and the piece remains unsocketed to indicate that the object does not fit into the socket.
5. When a correct piece is placed, a positive audio is played.
6. The task completes once all four sockets are correctly filled.

The `ShapeManager` implements two stages of randomization: (1) selection of which sockets will be active, and (2) random placement of the corresponding objects `PuzzleObject` and the `fakeObjects` in the vicinity of the table. The process is as follows:

1. `List<int> GenerateUniqueCombinationOf4()`
   - After pressing "Start", 4 out of the 6 shape-sockets on the table are chosen at random.
   - We have six possible sockets (indices $0 \dots 5$) but only four will be used each round.
   - Internally, we maintain a `usedCombinations` HashSet of Strings of the previously seen 4-tuples.
   - To pick a new combination:
     (a) Create a list $\{0, 1, 2, 3, 4, 5\}$.
     (b) Perform an in-place **Fisher–Yates** shuffle.
     (c) Take the first four elements of the shuffled list, sort them ascending to form combination.
     (d) If this sorted 4-tuple already exists in `usedCombinations`, repeat the above steps (rejection sampling) until a novel combination is found.
     (e) Add the new combination to `usedCombinations`.
2. `void RandomizePositionsForActiveObjects(List<Transform> objectsList, List<Transform> spawnPoints)`
   - Collect the subset P of active `puzzleObjects`.
   - Ensure at least $|P|$ spawn points are available.
   - Perform an in-place **Fisher–Yates shuffle** on the `spawnPoints` list itself.
   - Assign the k-th active object in $\mathcal{O}$ to the k-th spawn point and set their position and rotation.
   - Lastly the unused shape sockets and their corresponding `PuzzleObject` and the `fakeObjects` are set to inactive.

By using this randomization we avoid having the exact same combination of shapes each round. As for the spawn points of the task objects, there are 17 spawn points for `PuzzleObject` and 28 spawn points for `fakeObjects`, this practically ensures no repetition of the exact same spawn point combination, considering we only have 3 rounds.

4.2.5 UI Task

UI Task is a simple information UI with several topics and several sub-topics for each topic. It is rendered as a flat 2D UI inside the VR environment as seen in Fig. 4.5. There is an additional UI which has a question and a dropdown menu with several options. The UI's dimension and page content is deliberately chosen so that the text does not entirely fit into the view, thus forcing the participant to scroll. The participant has to navigate through the information UI to find the answer to the question. The procedure of the UI task is:

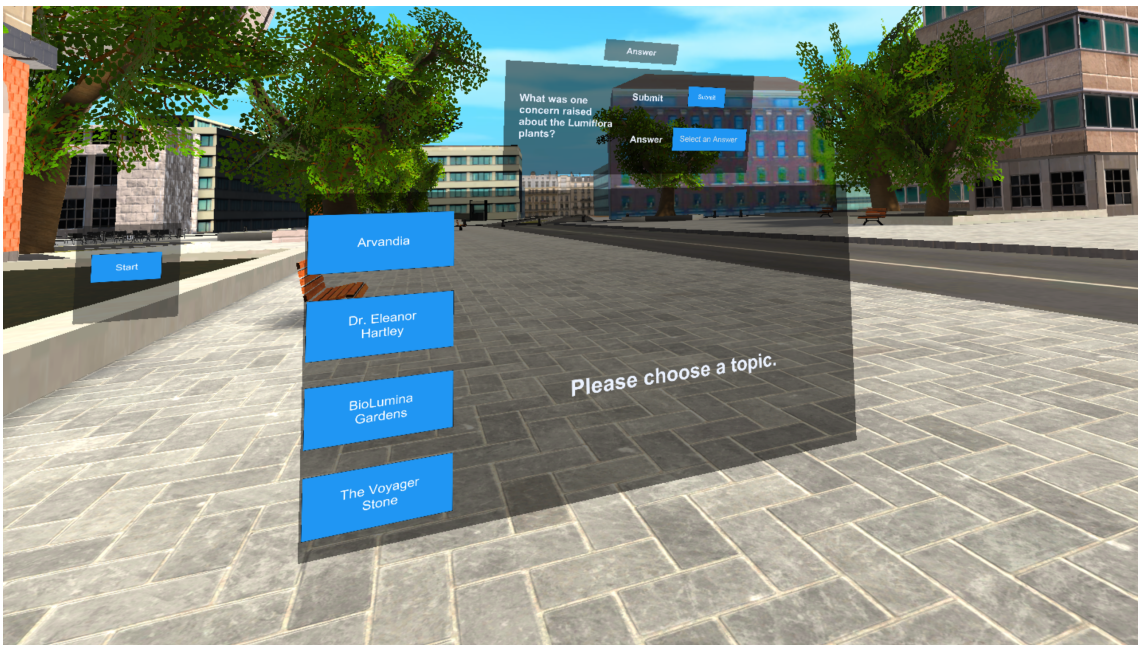


Figure 4.5: Information UI with 4 topics and the question UI above it.

1. The information panel and question prompt appear after pressing "Start".
2. The top level view shows four topic buttons and a UI with the prompt: "Please choose a topic."
3. Clicking a topic activates that topic's main page and its three subtopic buttons, and increments the page view and button click counters.
4. Clicking a subtopic hides the main topic page, displays the sub-topic page and again updates metrics. There is a back button, which returns to the previous page, it does not go up a level. If the participant viewed multiple sub-topics, the back button goes to the previous shown sub-topic.
5. When the participant has found the answer, they select their answer from a three option dropdown menu and have to click the "Submit" button.

6. A correct answer stops the timer, plays a completion sound, hides all UI, and signals task completion. An incorrect answer plays an error sound and briefly shows a "Wrong Answer" overlay before allowing retry.

Initially the topics were chosen based on real events or personalities, for example famous physicists, mathematical discoveries etc. Moreover the participant had to also input a number, in most cases a year in addition to selecting an option from the dropdown menu. The number input was implemented as a 3x3 num-pad, the same way as in SimonSays. However this caused the problem of fore-knowledge. Although inputting the specific year made the chances slim, it was still possible that the participant would already know the answer to the question and might not require to navigate through the UI to find the answer. This led us to create made-up AI generated stories and information using ChatGPT 4.5 model. Using this method ensures that there is no chance of the participant already knowing the answer, thus it forces them to navigate through the UI. As the problem with fore-knowledge is solved, we felt it was unnecessary to test the same button interaction again as in SimonSays. Therefore this part with entering a number was removed from this task.

This task evaluates the participant's ability to navigate a hierarchical 2D menu, scroll through content, and use a dropdown menu. Unlike the other tasks, it measures pure UI navigation and selection performance, including scrolling, page transitions, and menu interaction in VR. For that purpose we use the following metrics:

- *totalTime* (T_{total}): elapsed time from Start until a correct answer is submitted.
- *totalPageCount* (P): number of distinct pages or panels viewed (topic mains and subpages).
- *buttonClickCount* (C): total number of button presses (including topic, subtopic, Back, and Submit).

Implementation Aspects of the UI Task

The UI task simply generates a random order at first start up by creating a list `questionOrder = new List<int>  1, 2, 3` and randomizing its order. Each number represents a specific question. Clicking the UI buttons by the participant simply hides and activates the relevant UI elements. In code, the answer to question 1 is the first option in the dropdown menu. Answer to question 2 is the second option in the dropdown menu and so on. Each button click increments the button click counter.

4.2.6 House Task

The last task is the *House Task*, which functions as a "hat hunt" game in a three floor house, in which the participant must locate and collect three colored hats (red, green, blue). There is always one hat per floor and the location is randomized in each round with set amount of spawn points per each floor.

The procedure of the hat hunt is as follows:

1. Initially the participant must be teleported to the location of the house by pressing the first "Start" button located in the area where the rest of the tasks are.
2. Upon teleporting, there were instructions of the tasks and another "Start" button. When the participant is ready, they press this "Start" and the timers begins.

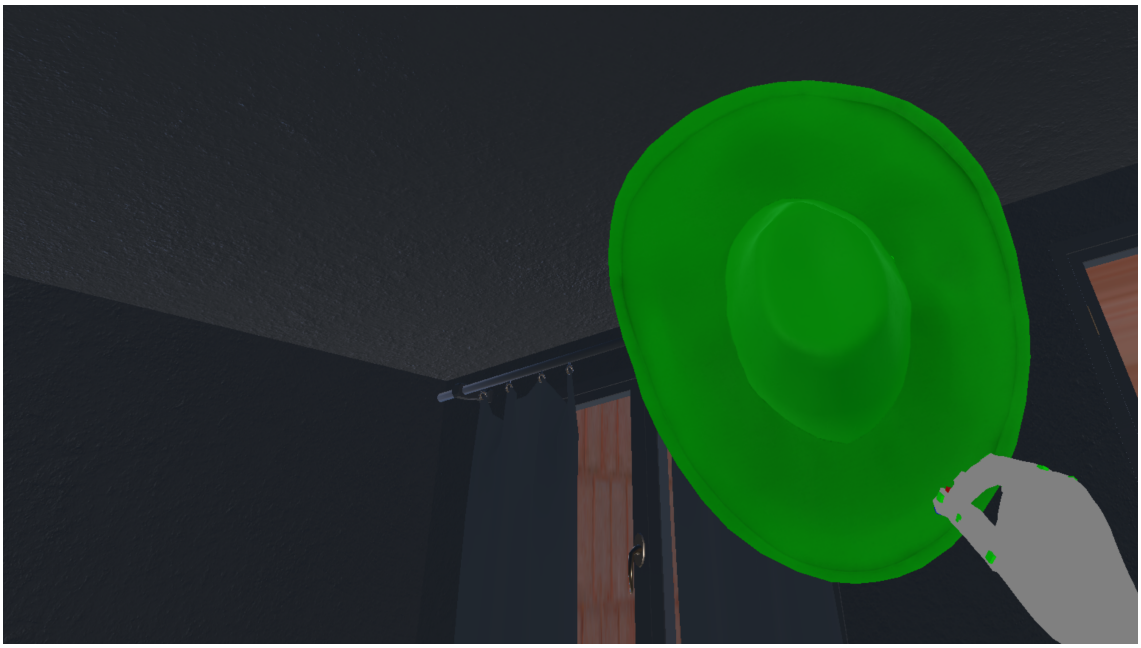


Figure 4.6: Picking up a green hat in the house using hands.

3. Three hats are randomly assigned to spawn points, one per floor. They are simply lying on the ground in the rooms, they are not hidden behind nor inside of furniture.

4. The participant navigates through doors (each door interaction increments a counter) and searches for the hat.

5. To collect a hat, the participant places it on their avatar's head, this triggers audio feedback, hides the hat, records the inter-pickup interval, and updates the average pickup time. The same socket mechanic is used as in the Shape Task, however the sockets are invisible in this case.

6. After all three hats are collected, the hat-hunt is completed and the participant is teleported back to the main location.

The house was taken from a different project, which was a Game Development Lab subject at RWTH, in which we took part. Originally the house was designed for a first-person-view horror game with low lighting that required the usage of a flashlight to be able to see and navigate. Initially we wanted to keep this functionality, as this would force the participants to use one hand for a flashlight and the other hand for locomotion and interactions. However, we have decided against it, as it creates a possible stressful environment for the participants, which could have a negative impact on their abilities.

The house was purposefully designed to be complicated and unintuitive, so the participant will have a hard time memorizing the layout. The layout itself was not changed from the original house used in the game, although some furniture and game related objects were removed to make more space for the hats.

This task combines spatial navigation in a complicated VR environment, in which the participant must be aware of collisions with the furniture, furthermore it includes door interactions and object manipulation in VR. By forcing the user to search through the environment for the hats, it tests the participants navigational skills, locomotion, steering and interaction with objects when using different locomotion methods.

One interesting effect of having to place the hat on top of their head is the problem of hand tracking via the HMD's cameras. The participants had to figure out that placing their hands above their heads caused the hand tracking to lose sight of the cameras. However Meta's hand tracking system does have a grace period of a few seconds, during which it most likely interpolates the position of the hands. During internal testing, we discovered that we must quickly place the hat above the head before the hand tracking is lost. This caused us to enlarge the invisible head socket's hitbox, which then could detect the hat when the hat was slightly above the eye level.

- *totalTime* (T_{total}): elapsed time from the task start until all the hats are placed on the head.
- *userDistanceTraveled* (D_{user}): cumulative path length sampled each frame,

$$D = \sum_{f=1}^{F} \left\| \mathbf{p}_f - \mathbf{p}_{f-1} \right\|,$$

  where $\mathbf{p}_f$ is the player position at frame f, and F is total frames during the task.
- *avgPickUpTime* ($\bar{t}_{\text{pickup}}$): average interval between successive hat pickups:

$$\bar{t}_{\text{pickup}} = \frac{1}{H-1} \sum_{i=2}^{H} (t_i - t_{i-1}), \qquad H = 3,$$

  where t_i is the timestamp of the i-th hat collection and H is the number of hats.
- *averageSpeed* (v_{avg}): mean movement speed during the task,

$$v_{\text{avg}} = \frac{D}{T_{\text{total}}}\,.$$

- *doorClicks* (C_{doors}): total number of door opening interactions performed.

Implementation Aspects of the House Task

The randomization is done using again `Fisher-Yates shuffle`. There are 14 possible spawn points for the hats. 6 possible spawn points for the ground floor, 4 spawn points for the first floor and 4 spawn points for the second floor. The randomization is done in such a way that the exact same combination of spawn points does not occur in the 3 rounds performed by the participant and functions as follows:

- `void RandomizeHatPositions (List<Transform> objectsList, List<Transform> spawnPoints)`
  1. This function assigns each of the three hats to a random spawn point on its designated floor, while avoiding repetition of the exact same assignment in consecutive rounds.
  2. A list `newIndices` is constructed by sampling:
     - One index from [0, 5] (first floor), one from [6, 9] (second), and one from [10, 13] (third).

3. The sampling is repeated—up to a maximum of 10 attempts—until `newIndices` differs from the `previousAssignmentIndices` saved from the prior call. This rejection-sampling step guarantees that participants never encounter the exact same hat distribution twice in a row.
4. Once a unique triple is obtained, each hat's transform position is moved to the corresponding spawn point.
5. Finally, their initial positions and rotations are recorded, in case a hat is lost due to clipping through objects or other unexpected physics interaction.

4.2.7 Overall Time Metrics

In addition to task specific metrics, the user study VR application also has time metrics for the entire duration of the study for each person. These are:

- **Familiarization Time** (T_{fam_i}): duration from the start of the round i until the participant pressed "Ready." For the initial round it starts with the start-up of the application.
- **Task Time** (T_{task_i}): total time required to complete round i's task.
- **Total Study Time** (T_{study}): duration of the entire study since the start-up of the application until all 3 rounds are finished.

$$T_{\text{study}} = \sum_{i=1}^{3} T_{\text{fam}_\text{i}} + T_{\text{task}_\text{i}}$$

- **Adjusted Study Time**: $T_{\text{study}} - T_{\text{fam}} - \sum_{\text{round}} T_{\text{task}}$ to isolate pure task performance.

4.3 Questionnaire

We also employed questionnaires to gain feedback from participants. Immediately after the VR session, participants were asked to fill an online questionnaire. In the first section of the questionnaire, the participants provided general demographical information such as age, gender, education, previous experience with VR. Afterwards they filled two research questionnaires that are validated and have a widespread use. These are the *User Experience Questionnaire (UEQ)* [LHS08] and *Presence Questionnaire (PQ)* [WS98]. The participants had to fill UEQ and PQ for each of the three rounds they had experienced.

We considered including additional questionnaires such as the *NASA TLX* [HS88] and *Sickness Questionnaire (SSQ)* [KLBM93]. However, based on internal testing, we deemed that the entire study was already long, which was later confirmed in our user study with participants. The entire study including the questionnaires and an interview could take up to 90 minutes in outlying cases. Filling both UEQ and PQ three times takes a considerable amount of time. Though this caused the problem of repeated measures, which we tackle in Chapter 5.

NASA TLX is designed to measure the cognitive workload. Hence it is not entirely applicable to our user study, as the tasks are not designed to induce a heavy cognitive workload, with the only possible exception being the SimonSays task. Though there exists an adjusted version designed for VR applications proposed by Vidal-Balea et al. in [VBFLFC24], which we briefly considered but decided to not include it out of time reasons.

Regarding the SSQ, *cyber-sickness* is an inherent problem of the current generation of VR technology and is not fully avoidable with no consensus of a solution [DNN14]. As with any questionnaire it also has its limitations as shown by Bimberg et al. in [BWK20]. They even go as far as recommending to use it only when genuinely necessary to answer the research question. Thus we decided to stick with UEQ and PQ. Though we did notice an interesting possible correlation between the locomotion modes and cyber-sickness, which will be covered in Chapter 5 and necessitate more research into it.

4.3.1 User Experience Questionnaire

User Experience Questionnaire (UEQ) was originally created to measure the user experience of interactive products, though it has also found application in VR/AR, for example in [SW25, WFK24]. UEQ consists of a 7-point Likert scale, with opposing adjectives at each end. The middle point's value represents a neutral value. Different items in the questionnaire belong to a broader *scale*, which covers different aspects of the user's experience. As it was not originally developed for VR, some of its scales do not fit our user study, thus we have decided to use the modular approach of UEQ+ [SST21] to create a custom questionnaire covering only certain aspects. Our choice of UEQ+ items and their corresponding scales can be viewed in Table 4.1.

Table 4.1: User-Experience Questionnaire (UEQ) items used and their scale category.

#	Negative pole	Positive pole	Scale
1	Not understandable	Understandable	**Perspicuity**
2	Difficult to learn	Easy to learn	**Perspicuity**
3	Complicated	Easy	**Perspicuity**
4	Unpredictable	Predictable	**Dependability**
5	Does not meet expectations	Meets expectations	**Dependability**
6	Impractical	Practical	**Efficiency**
7	Inefficient	Efficient	**Efficiency**
8	Confusing	Clear	**Perspicuity**
9	Annoying	Enjoyable	**Attractiveness**
10	Unpleasant	Pleasant	**Attractiveness**

We have chosen the scales *Attractiveness*, *Efficiency*, *Dependability* and *Perspicuity* as most important to determine the user's experience using the three different locomotion and interaction modes. One might argue that scales such as *Usefulness* or *Intuitive Use* would be also appropriate, however our second questionnaire is more focused on on those aspects. All of the scales represent the user's subjective impressions. Perspicuity, and its alternative name learnability, represents the user's perception of how easy it is to understand and learn using the interaction mode. This is especially important for inexperienced users. Though it is also vital for experienced users, as hand tracking-based interaction and locomotion is not commonly used, so it shows even for experienced users how well they can learn the new interaction modes. Dependability is the impression of how well the interaction mode responds to the user's commands, and how consistent and predictable it is. Efficiency represents how much effort is required to perform a task, i.e whether there are unnecessary steps. Lastly Attractiveness is the general feeling and impression of the interaction mode, or more simply put whether the users liked it or not.

The full scales in UEQ+ consists of many more items, however we decided to drop some

of the items for the scales Attractiveness, Efficiency and Dependability, as the items were not applicable to our use case. This can have an impact on the accuracy of the scales as shown in [SST21], however since UEQ+ is only one of the four different evaluations data we collect, we deem it acceptable. Another reason why we dropped some of the items in a scale is to reduce the duration of the entire study. Inviting participants for a 90 minute study would lead to less participants compared to a 60 minute study. Moreover filling the same UEQ+ questionnaire three times can lead to a sense of frustration by the participant, which can be dangerous and lead to inaccurate answers.

4.3.2 Presence Questionnaire

The UEQ questionnaire focuses on the interaction and locomotion mechanics itself, however we are also interested in how the different modes affected the participant's ability to perform the five tasks. Moreover immersion and intuitiveness are an important factor of a good VR experience. Hence we required a method to rate the effect of the three different modes on the participant's presence and immersion and their abilities to do the tasks. For this purpose we employed the Presence Questionnaire (PQ) developed by Witmer and Singer (1998) [WS98].

Table 4.2: Presence Questionnaire items retained for this study, with their associated Factor.

#	**PQ Items**	**Factor**
1	How responsive was the environment to actions that you initiated (or performed)?	Involvement
2	How natural did your interactions with the environment seem?	Involvement
3	How much did your experiences in the virtual environment seem consistent with your real-world experiences?	Involvement
4	How natural was the mechanism that controlled movement through the environment?	Involvement
5	How compelling was your sense of moving around inside the virtual environment?	Involvement
6	How well could you move or manipulate objects in the virtual environment?	Involvement
7	How involved were you in the virtual-environment experience?	Involvement
8	How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?	Adaptation-Immersion
9	How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?	Adaptation-Immersion
10	How easily did you adjust to the control devices used to interact with the virtual environment?	Adaptation-Immersion
11	How much did the control devices interfere with the performance of assigned tasks or with other activities?	Interface-Quality

PQ has been widely utilized in VR research, ranging from applications in commercial training research [TBL11] to psychology research about sexuality [RRG+02]. Presence is

defined as the feeling of *"being there"* in a VR environment and the ability to fully engage with it [SW97]. However it is important to distinguish the difference between immersion and presence, as immersion is the ability of the VR device to disconnect us from the real world and being able to stimulate all of our senses in a fully virtual environment [SW97]. Both of them are pivotal aspects of VR, however in our case we deem presence the more important factor.

It was highly approved by the users in a study performed by Degenhard et al. [DARR25], moreover it reflected the user's experience accurately. PQ items are also more task oriented, this suits our research question, as we want to discern how each interaction mode affects the user's performance of our tasks. However Degenhard et al. also warn that despite questionnaires being commonly used in VR research, they are not universally applicable and may not be able to measure every aspect of the user's experience. This is evident in the items of PQ, as many of them are not applicable to our VR user study, hence similar to UEQ we have decided to use only relevant items of the version 3.0 of the questionnaire [WJS05], these can be viewed in Table 4.2.

Participants rated each item in the table on a 7-point Likert scale with a negative adjective being 1 and a positive adjective being 7 (e.g. from "Not at all" to "Completely" or "Very poor" to "Very good"). Each question also belongs to a factor. In total there are 6 factors in the full questionnaire, as with the UEQ, many of the questions in the original are not applicable or relevant for our use case. For example some are about visual or audio aspects, which we are not interested in. Because our study compares three interaction modes: controllers, hands, and hands with gaze, PQ's emphasis on aspects such as naturalness and control, how responsive the world was, and how distracting the input devices have an effect on the presence of the participants. This property makes it especially well suited to detect differences in presence arising from replacing controllers with hand tracking and gaze interaction.

Based on our question choices, the PQ scales that we have are: *Involvement*, *Adaptation/Immersion* and *Interface Quality*. Involvement represents how involved the user is, i.e their focus of mental energy and cognitive skills at the current activity. It can also be interpreted that natural experience and interaction capture the attention of the user, letting them focus and perform what they want to do. Adaptation/Immersion is the state of being immersed and stimulated by the virtual environment and is closely linked to Involvement. Lastly Interface Quality indicates the quality of the control devices, which would be controllers or hand tracking technology in our case.

4.4 Interview

Lastly there was an optional recorded interview, which formed our qualitative data. The recordings were later re-listened and analyzed to extract the answers of the participants. Although it was optional, we found that most people were keen to describe their experiences and share their opinions about the VR application. The interview was semi-structured, as there were three subject areas. However if the participant shared an interesting piece of information, we encouraged them with follow up questions to delve deeper into the topic to gain more feedback. This was more prominent with participants that were more experienced with VR or were software developers themselves, as they understood more of VR mechanics and were quite interested in our implementation. In case the statement of the participants was without a reasoning or justification, we always followed up with a why to discern the reasoning behind their answer. The questions of the interview are:

1. Firstly we asked for advantages or disadvantages of each interaction mode. If the participant was not able to provide a clear answer, we instead asked what they personally liked or disliked in each interaction mode.
   Purpose: Elicit a high level comparison of what participants found useful or problematic in each round.

2. Next we asked whether the participant thinks VR without controllers is feasible. We explored the aspects of immersion and intuitiveness of using hand tracking versus controllers. If the participant was more knowledgeable about VR, we also asked for their opinions about the limitations and problems of hand tracking. Additionally whether it is technologically limited or by its use case implementation.
   Purpose: Since one goal is to employ controller-less interaction and locomotion in an educational setting, where many users will have minimal or no initial VR experience, we explore whether hands only or hands with gaze feels more intuitive, immersive, and practical in a learning context. For experienced users or developers, we also probe deeper into current hand tracking limitations, whether we are technology or application limited.

3. Afterwards we asked for their opinion about the benefits of adding gaze tracking. Whether it was helpful or not and which interaction mode without controllers they preferred.
   Purpose: Directly address our secondary research question: does adding gaze really enhance controller-free VR?

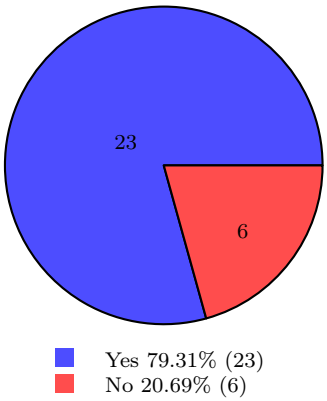
4. Lastly as an ending question we asked if they had any suggestions on how to improve the current implementation or any other remarks they would like to share.
   Purpose: Give participants a final open ended opportunity to share any ideas or feedback, that we may have missed in the previous areas.

Chapter 5

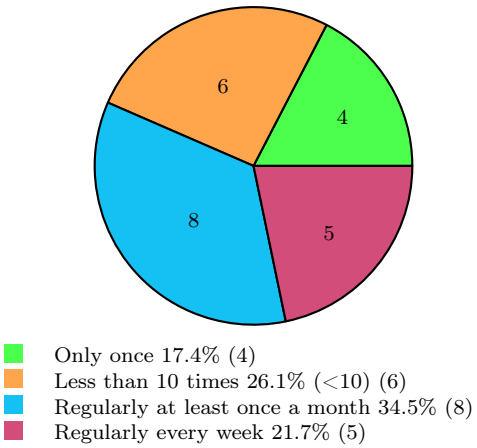
Evaluation

This chapter provides our analysis of our findings. Firstly we go over the demographics of our participants. Next we provide an overview of the results of our in-application metrics and the questionnaire. Then we analyze them to find common patterns and relationships. Afterwards we present our findings from the interviews that we have performed. Lastly we perform a retro-inspection of our design and highlight the flaws and provide potential improvements for future work.

5.1 Demographics



(a) Have you experienced VR before?



(b) How often do you use VR?

Figure 5.1: Survey results with absolute counts inside each slice and legends below.

First we present the demographic data of the participants, this data was collected in the initial part of our questionnaire. We asked the participants for several demographic data points, such as age, gender, education, and experience with computer games, with the goal of finding out whether these factors have an influence on the participants' results.

In total we had 30 participants for the user study. 10 participants were members of the LFI institute at RWTH. The rest of the 20 participants were recruited using flyers placed at the RWTH campuses. One participant could not fill out the questionnaire, due to technical issues, hence the demographic data represents 29 out of the 30 participants. Participation was voluntarily and the user study could be paused or aborted at any time.

The average age of the participants was 28.41, with the median being 28. The lowest age was 20 and the highest was 64. The 25th percentile of age is 24 and the 75th percentile of age is 29. Not including the outlier highest age of 64 reduces the average to 27.14 but has minimal impact on the median and quartiles (roughly 0.5 years difference).

We had 23 participants who identify as male and 6 participants who identify as female, which results in a 79.3% male to female ratio.

Out of the 29 participants, 20 were from Germany, 3 were from Slovakia, and the rest of the participants each came from different countries, though the majority of these countries are European.

16 participants have finished a form of higher-level education, while the rest have either finished or were in process of doing an apprenticeship or have so far finished only secondary education.

Only 3 participants have never played computer games, of the 26 who either currently regularly play or used to regularly play computer games in the past, 25 regularly play or used to play first person view computer games. We asked this information to ascertain, if experience with computer games provides a benefit for VR, as computer games often require 3D navigation and problem solving.

We asked for the experience of participants with VR, which can be visually viewed in Fig. 5.1. Out of the 29 participants, 23 have experienced VR before, moreover, 13 of them use VR regularly, which we define as regularly at least once a month or regularly every week. This leads to a demographic basis that is quite experienced with VR, which is unusual. We later analyze whether experienced participants perform better than inexperienced participants. Interestingly, 11 (52.17%) of the participants who experienced VR before have tried out hand tracking, all of these instances were with the Meta Quest series of VR devices.

Lastly, the questionnaire data had to be normalized. For example, participants had a free input field for describing what hand tracking device they had used, the answers were various forms of "Meta Quest 3," such as "quest 3 hand," "quest 3 UI," etc. Such answers were normalized simply to "Meta Quest 3." The same applies for their current occupation question. Inputs as "Master Student" were normalized to just as student.

5.2 Task Metrics Analysis

We now analyze the metric data generated by our VR application, these metrics measured various performance and speed factors of the participants. Unfortunately, some of the metric data is not usable were not including in the calculations. For example if someone had to take a pause during a task, all the metrics for the tasks were also marked as unusable. This was done in order to not skew results. The main reason for unusable data is cybersickness experienced by participants, which required a pause in the VR session. We did not include a pause function, hence the timer metrics were left running, this is unfortunately an oversight in our implementation of the user study. Many other metrics require the timer to calculate their values, so taking a break invalidates several other metrics associated with that task. Another reason for excluding some metric, was one participant's limited English skills, which made the UI task data unusable, as it involves reading a lot of text.

A implementation bug that we noticed too late was a broken timer responsible for measuring the familiarization time of participants in each round. This bug was caught after the first 8 participants, for this reason, the `familiarizationTime` and `totalFamiliarizationTime` data for the first 8 participants is also unusable.

All the data that we have and have used is available on RWTH GitLab[1], both the raw data and the normalized and the adjusted data is included. The unusable data of metrics is replaced with an X, however the familiarization data of the first 8 participants is not marked as X, since that can be easily filtered out based on the date of the participants' testing.

5.2.1 Descriptive Analysis

We first start with descriptive statistical analysis of our data. Table 5.1 shows a selection of the most relevant metrics with calculated mean, standard deviation, median, minimum, and maximum values. The full version of the table with all the metrics and additional information such as variance and quartiles can be viewed in Appendix A.

Normal Distribution and Pruning

We tested our data to see whether it is normally distributed, as many analytical tools require the data to have a normal distribution. We used the Shapiro-Wilk [SW65] test with the commonly used $\alpha = 0.05$ to determine whether our metrics are normally distributed, however, only 30 are considered to be normally distributed, whereas the remaining 54 do not follow a normal distribution. Some metrics are expected to be not normal, for example simple counters like number of wrong attempts or number of clicks. Though it surprised us that many timer metrics, such as `HouseTaskManager.totalTime` or `SimonSaysManager.totalTime`, are non-normally distributed across all three interaction modes. Most important, the `taskTime` for each interaction mode is considered to be normally distributed.

Looking at the data, it became obvious that we have many outliers that caused our data to be not normally distributed. To reduce the number of outliers, we decided to apply Tukey's IQR [Hoa03] to reduce the number of outliers. With the commonly used IQR multiplier of 1.5, we could prune 42 outliers, which would increase the number of normally distributed metrics from 30 to 40, while reducing the number of non-normally distributed from 54 to 44.

However, after deliberating over the possibility of pruning, we have decided not to perform it. As these outliers are of interest to us, they signify cases of participants struggling with a task, which results in an outlier with a higher value. Another possibility for outliers is statistical flukes, caused by our randomization mechanics in tasks and having only 30 participants. These outliers can be mostly seen as max values in Table 5.1 that are way above the mean, including the standard deviation. For example, the house task's `totalTime` maximal values are significantly higher. The statistical flukes can be seen as small minimal values, again with the house task, we see minimum values that are below the mean. This was caused by participants being lucky with the randomized position of hats, as it happened more than once that some participants were able to find them immediately by checking a single room, whereas the majority of the rest had to search multiple rooms to find the hats. These statistical flukes and non-normal distribution could have been improved if we had more participants, ideally we would have hoped to have about 50 participants, with the current 30 being the minimum. However, due to time constraints and a mediocre show-up of participants compared to previous studies, we have no other choice than to just work with the data available.

[1] https://git.rwth-aachen.de/jan-tugsbayar/master-thesis

Therefore, we have to take into consideration that not all the data follows a normal distribution when applying non descriptive analytical tools.

Table 5.1: Mode Comparison Summary per Task

Task	Mode	Mean ± SD	Med	Min	Max
HouseTaskManager					
averageSpeed	Controllers	1.78 ± 0.55	1.79	0.61	3.06
	Hands	1.20 ± 0.28	1.21	0.54	1.75
	Hands + G.	1.17 ± 0.32	1.10	0.60	2.16
totalTime (min)	Controllers	3.38 ± 2.61	2.67	1.14	12.74
	Hands	3.58 ± 1.18	3.37	1.49	6.23
	Hands + G.	3.36 ± 1.87	3.02	0.85	9.50
MoveTaskManager					
averageArrivalTime (sec)	Controllers	5.53 ± 1.02	5.40	3.53	8.43
	Hands	5.94 ± 1.31	5.85	3.73	8.74
	Hands + G.	6.75 ± 2.75	6.07	3.86	17.59
totalTime (sec)	Controllers	28.81 ± 5.38	29.26	17.64	42.18
	Hands	31.80 ± 7.40	31.59	18.68	51.47
	Hands + G.	34.49 ± 12.51	30.97	21.53	70.80
userDistanceTraveled	Controllers	57.34 ± 9.84	55.97	40.17	80.20
	Hands	55.15 ± 8.44	53.66	41.88	72.85
	Hands + G.	55.77 ± 8.03	56.17	40.44	76.09
ShapeManager					
avgFillTime (sec)	Controllers	21.55 ± 8.39	19.25	8.61	39.24
	Hands	31.83 ± 11.78	29.11	8.25	64.36
	Hands + G.	41.85 ± 21.68	36.62	16.39	102.60
totalGrabs	Controllers	7.17 ± 4.11	5.50	4.00	23.00
	Hands	8.87 ± 3.76	8.50	4.00	18.00
	Hands + G.	11.90 ± 7.26	9.00	4.00	35.00
totalTime (min)	Controllers	1.45 ± 0.56	1.35	0.57	2.62
	Hands	2.24 ± 0.72	2.00	1.28	4.29
	Hands + G.	3.02 ± 1.59	2.65	1.12	8.55
SimonSaysManager					
averageInterClickTime (sec)	Controllers	1.31 ± 0.78	1.05	0.58	4.47
	Hands	2.83 ± 1.41	2.30	0.89	6.67
	Hands + G.	2.00 ± 0.74	1.85	0.96	3.90
totalTime (sec)	Controllers	28.96 ± 15.42	23.45	18.69	93.96
	Hands	68.04 ± 57.27	53.68	21.43	333.89
	Hands + G.	42.70 ± 22.26	33.39	21.99	107.90
userClicks	Controllers	9.90 ± 2.19	9.00	9.00	18.00
	Hands	14.43 ± 8.38	12.50	9.00	52.00
	Hands + G.	11.77 ± 5.13	9.00	9.00	28.00
userWrongClicks	Controllers	0.40 ± 1.04	0.00	0.00	4.00
	Hands	1.47 ± 2.24	1.00	0.00	11.00
	Hands + G.	0.73 ± 1.20	0.00	0.00	5.00
UITaskManager					

Task	Mode	Mean ± SD	Med	Min	Max
totalTime (min)	Controllers	1.08 ± 0.61	0.96	0.36	2.78
	Hands	2.01 ± 1.32	1.51	0.33	4.61
	Hands + G.	1.80 ± 0.99	1.62	0.48	4.14

Findings

The table shows a clear trend: controllers are the fastest, respectively most efficient, interaction mode. Out of the 30 participants, 20 had the fastest `taskTime` when using controllers. This is most evident in the `taskTime` in Fig. 5.2. The figure shows the mean of all participant's taskTimes and `familiarizationTime` for all three rounds. It also shows the Q_1, Q_3, median, minimum and maximum values. Interestingly, we do not see big differences in the familiarization time of each interaction mode. However we need to take into consideration that `familiarizationTime` is represents 22 out of 30 total participants. Looking at the raw data, using only code based randomization we have almost perfect distribution of first round interaction modes: 10 participants began with controllers, 11 with hands, and 9 with hands and gaze, an almost perfect distribution of first round interaction modes. We thought that interaction modes without controllers might require longer familiarization times, however on average it does not seem so. Even looking at the maximum values, they are 11.78 for controllers, 9.31 for hand and 9.89 for hands with gaze, all the values are in minutes. This includes the time spent on watching the tutorial videos. The tutorial videos for controllers is 87 seconds long, hands is 90 seconds and hands with gaze is 78 seconds long. Thus on average participants spent 3.57 minutes familiarizing with controllers, 4.1 minutes with hands and 3.59 minutes for hands with gaze. This follows the same trend as in the figure, which includes the tutorial video time.

The mean task times for hands and hands with gaze are very close, even the median value is smaller. The same applies to the quartiles. Though in the maximum values, there was an outlier, that required almost 30 minutes for the gaze mode. This was the participant of older age and had limited English skills. Despite the problems with language and unfamiliarity with the technology, they preserved and finished the study with almost 30 minutes for their first round, which was hands with gaze. For the next round of controllers, they improved to 16 minutes and for the last round of hands required only 9 minutes demonstrating drastic improvements. They were unable to fill the questionnaire at the end of the study, due to technical issues. The full statistics of all the tasks and overall results can be viewed in Appendix A. We now go over each task and present findings from the statistics.

House Task An interesting observation is the average speed of the participants in the house task. Controllers have the highest average speed, as they provided more accurate and fluid locomotion. The supervisors noticed that many participants had problems entering rooms through the doorway when using gesture locomotion, which results in a lower average speed. With the `totalTime` we see big differences between the minimum, maximum, and mean times, this is the result of the randomization mechanic for the position of the hats, however the mean time is roughly the same between all modes.

Movement Task The purpose of the movement task is to determine how accurately the participants are able to steer and move to the designated plate. In regard to `totalTime` and `averageArrivalTime` (average time required to reach a plate), we do not see any

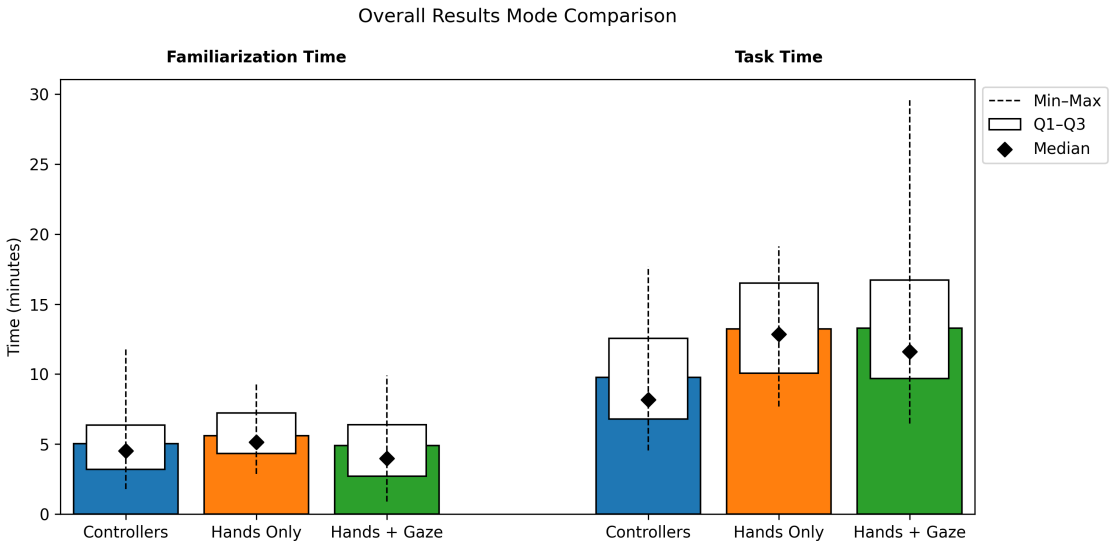


Figure 5.2: Mean familiarizationTime and taskTime of each interaction mode with ± SD represented as the black vertical line.

big differences. To determine the accuracy of the participants, we need to look at their distance traveled compared to the optimal path, as explained in Section 4.2.2. Table 5.2 summarizes the values, the full values of each participant can be viewed in Appendix C.

Table 5.2: MoveTaskManager: Difference Between User Distance Traveled and Optimal Path

Mode	Mean ± SD	Min	Max	Mean Opt. Path	Mean Traveled
Controllers	9.29 ± 5.75	3.71	24.98	48.05	57.34
Hands	7.83 ± 2.34	4.21	14.71	47.32	55.15
Hands + Gaze	8.81 ± 5.27	3.85	29.09	46.96	55.77

To give context to the data, we also provide the mean of the optimal paths and the mean distance traveled by the participants. We see in the minimal values and standard deviation that some participants are able to move very accurately, almost close to the optimal calculated path. In percentages, we see that when using controllers we have a mean 19.53% increase over the optimal distance, with hands 16.49%, and with hands with gaze 19.35%. Surprisingly, hand steering has the lowest mean and maximum values and the lowest mean increase over the optimal distance. Hand steering was not well received by the participants, as will be shown in the questionnaire and interviews. Though we hypothesize that steering with the hand was the most accurate, as holding the hand still is easier to perform compared to keeping the head still or holding the joystick tilted straight forward. As we did not foresee this situation, no focus was placed on observing the steadiness of hands vs. the head. Nonetheless, it was mentioned in the interviews that some participants found it hard to keep looking in a straight direction, as any distraction naturally causes head movement.

Shapes Task In the Shapes Task, the average fill time of each socket is higher when using gaze, the number of grabs and total time required is also higher. This is the result

of participants not being able to accurately aim and pick up items or by dropping their held puzzle pieces. From observations, many participants had a hard time aiming with the gaze interactor to pick up the puzzle objects, as the ray was not always at eye level. Additionally, participants picked up objects from a distance using their gaze interactor and pinch, however, as they did not have to aim using their hands as in the hands mode, they often moved their hands into such a position that the HMD's camera lost track of them and the pinch gesture. This caused many participants to inadvertently drop their held puzzle objects when turning and moving back to the sockets. Consequently, hands with gaze is the worst performing mode for the Shapes Task.

Simon Says Task A reverse trend is visible in Simon Says: participants were quicker using the gaze interactor than their hands, especially visible in the average time between button clicks and the total time required. We noticed that the participants just kept their pinching hand in front of the HMD and quickly aimed with their gaze on the correct button and pinched. Whereas in the hands mode, the majority chose to poke instead of using the near-far interactor, therefore they had to move their hand accurately on top of the button and then poke to click the button. There was also a high number of wrong clicks when using hands, this was caused by an oversight in our hands mode, which will be covered in more depth in Section 5.5.2. But to summarize, the near-far interactor ray was always active and it was highlighting other buttons when the participants wanted to use the poke interactor. This caused confusion and false inputs for the Simon Says Task. The results might have been different if this issue had been rectified before testing began, but as it stands, hands with gaze is the better of the two controller-less modes in Simon Says.

UI Task For the UI Task, we foresaw the scrolling with gaze being the worst, however, the hands interaction mode was the longest in actuality. This was the result of participants preferring to use their hands and poke the interface as if it were a large touchscreen or smart board. To perform touch interactions, they had to move closer to the UI, but they were not able to read the text properly from close range. Hence, participants moved back and forth when scrolling or generally interacting with the UI.

In this task we had to remove the metrics for one aforementioned participant with limited English skill and the task requires a lot of reading. Despite the removal of this participant, we see relatively high maximum values compared to minimal values. This was the result of many participants taking a systematic approach to read through everything before looking at the possible answers, whereas some participants first looked at the answers, which served as a hint where to look for the answer.

5.2.2 Learning Effect

We next have to consider the fact that, despite randomization, the participants are able to learn what to expect and what to perform in each task in the subsequent rounds, which results in a learning effect.

A simple observation, which confirms the learning effect, is that 21 participants' first round was the slowest, out of the total 30. We also tested if it is the case that we have a monotonic decrease in `taskTime` over the rounds, this was, however, inconclusive and a naive approach, as only 14 participants showed a monotonic decrease in `taskTime`.

To further investigate the learning effect and its implications, we applied repeated measures ANOVA (RM ANOVA)[Gir92]. We performed the RM ANOVA on `taskTime`, as that metric is normally distributed. While normal distribution is not a strict requirement

for RM ANOVA, having it normally distributed gives better results. The results of our analysis can be seen in Table 5.3, we omit the explanation of the methods we used and simply present the results.

Table 5.3: Learning Effect Analysis Across Rounds

(a) RM ANOVA (`taskTime` by Round)

	F Value	Pr > F
round	31.3134	0.0000

(b) Residuals Normality (Shapiro–Wilk)

W	p
0.979	0.201

(c) Post-hoc Paired t-tests (Bonferroni)

Comparison	t	raw p	bonf p
Round 1 vs 2	4.31	0.0002	0.0006
Round 1 vs 3	7.04	0.0000	0.0000
Round 2 vs 3	4.20	0.0003	0.0008

A one-way repeated measures ANOVA on task completion time revealed a significant effect of the rounds performed by the participants: $F(2, 58) = 23.668,\ p < .0001$. The results indicate that participants got faster over successive repetitions. The null hypothesis is rejected and there is significant variance and a learning effect present in regard to participants' round `taskTime`.

A key assumption for ANOVA is that the residuals, i.e. outliers, are normally distributed. To confirm this, we performed another Shapiro-Wilk normality test with $\alpha = 0.05$ on the residuals and the results are: $W = 0.977$, $p = 0.112$, which confirms that the residuals are indeed normally distributed, thereby validating the results of ANOVA.

While ANOVA simply tells us that there are differences between rounds, but not which rounds differ, we followed up with Bonferroni post hoc paired t-tests [Dun61] using the commonly accepted $\alpha = 0.05$. These showed significant reductions in mean `taskTime` between all pairwise rounds, with all adjusted Bonferroni p values being < 0.05. This confirms that, in general, participants did get better with each subsequent round.

Combining this result with the fact that 21 participants were slowest in the first round and that 14 participants had a monotonic decrease in their round `taskTime` values, we can conclude that there is a strong learning effect in play here. This has the following implication: it indicates that with multiple exposures and sessions with different interaction modes, the participants are able to adapt to the new interaction modes, which improves their `taskTime` in subsequent rounds. There is an another possible implication, as the participants learn and become faster, the metric data for subsequent rounds become less accurate. The distribution of initial interaction modes was almost evenly distributed, with 10, 9, 11, the order of tasks is randomized and so is each task itself. Despite that, we must acknowledge the possibility, that using the metric of subsequent rounds to calculate for descriptive analysis of the entire study can skew the results. Though looking at the results, it is still clear that controllers were the fastest and the other two hand tracking based ones were slower, but comparable to each other.

5.2.3 Experienced vs. Inexperienced

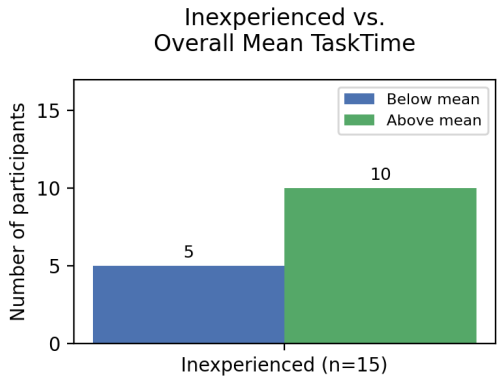
We wanted to determine how big of a role previous experience plays in our user study. We asked the participants whether they have experienced VR before by answering the following questions:

1. Have you experienced VR before?
    - No.
    - Yes.
2. How often do you use VR? (Only applicable if answered "Yes." to question 1.)
    - I only used it once.
    - I only used it a few times (<10).
    - I use it regularly, at least once a month.
    - I use it regularly almost every week.
3. Have you ever used VR with hand tracking before? (Only applicable if answered "Yes." to question 1.)
    - Yes.
    - No.
4. Do you regularly play computer games?
    - No, I have never played computer games.
    - No, but I used to regularly play computer games.
    - Yes.

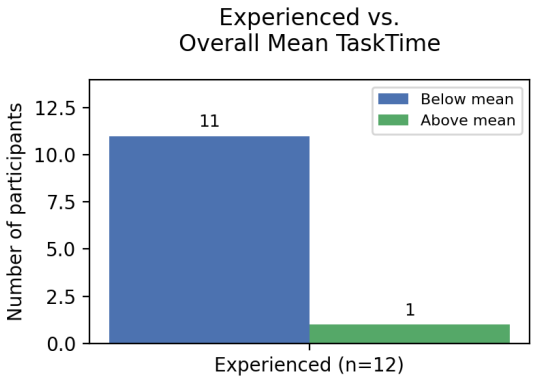
The following observations can be visually seen in Fig. 5.3. We label a participant as inexperienced if they never used VR before, tried it only once, or tried it a few times (<10). Otherwise they are labeled as experienced, as they use it regularly or have used it more than 10 times. We compared the sum of `taskTime`s of inexperienced participants to the mean of the sum of `taskTime`s of everyone, practically calculating how much time they required for the tasks in the entire study. A participant's `taskTime` was not included in the calculation if some of their `taskTime`s were marked as unusable. We had 15 inexperienced participants in total and only 5 of them were below the mean, meaning they were faster than average. Out of 12 experienced VR users, 11 of them were below the mean, which signifies that previous experience was of great benefit to them.

We further wanted to discern whether having previous experience with hand tracking was beneficial. There were 10 participants who had such experience and 8 of them were below the mean—again indicating that previous experience gives an advantage.

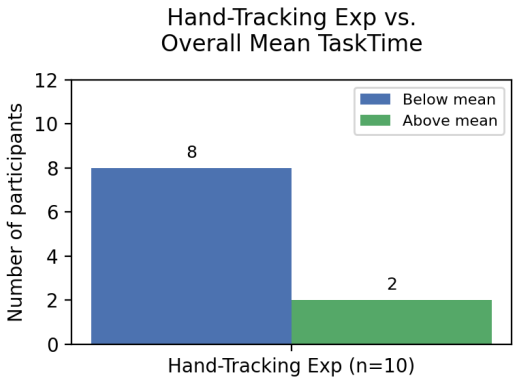
Lastly, we were also curious whether experience with computer games is beneficial to the participants. We label someone as a gamer, i.e. experienced with computer games, if they answered "Yes." or "No, but I used to regularly play computer games." to the fourth question. There were 25 participants considered experienced and 16 of them were below the mean. We also checked whether their `taskTime` with controllers was below the mean `taskTime` of controllers, and again 16 out of the 25 were below the mean. While we did not delve deeper and our sample size of 30 is relatively low, it is reasonable to say that experience with computer games is beneficial when using VR, as computer games commonly require players to navigate in a 3D environment and often require spatial problem solving skills.



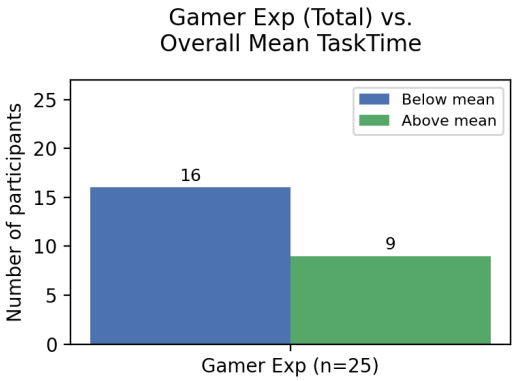
(a) Inexperienced vs. overall mean



(b) Experienced vs. overall mean



(c) Hand-tracking exp. vs. overall mean



(d) Gamer (total) vs. overall mean

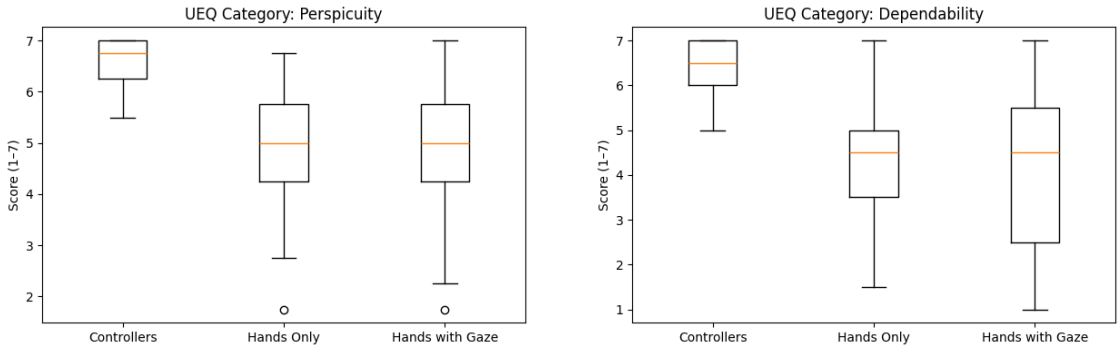
Figure 5.3: Comparison of participant subgroups (Inexperienced, Experienced, Hand-tracking experience, Gamer total) against the overall mean total taskTime.

5.3 Questionnaire Analysis

After presenting the results of the metrics, we now analyze and present the results of our two scientific questionnaires from the study questionnaire.

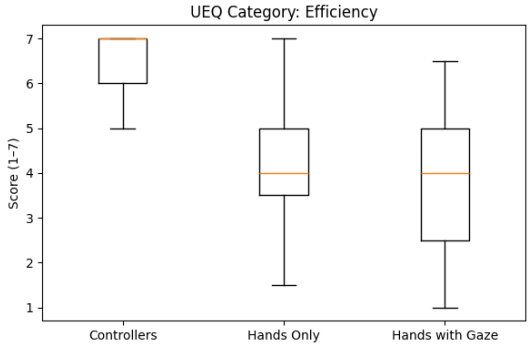
5.3.1 User Experience Questionnaire

In order to give an overview of the results of UEQ, we present each UEQ category in our questionnaire as a box plot in Fig. 5.4. The statistics of each individual question can be viewed in Appendix C. The yellow line represents the median, the top of the box represents Q_3 (75th percentile), and the bottom of the box represents Q_1 (25th percentile). The whiskers represent the mean minimum and maximum values, as each category contains several questions. Circles represent extreme cases of outliers, which are outside of the IQR (interquartile range $1.5 \times (Q_3 - Q_1)$). As a reminder, perspicuity presents the user's subjective impression of how easy it is to learn and use the interaction mode. Dependability is the user's subjective impression of how predictable and consistent the interaction mode is, and how much they feel in control. Efficiency is the subjective impression that they can achieve their goal with minimal effort. Lastly, attractiveness is their overall impression, i.e. whether they liked it or not.

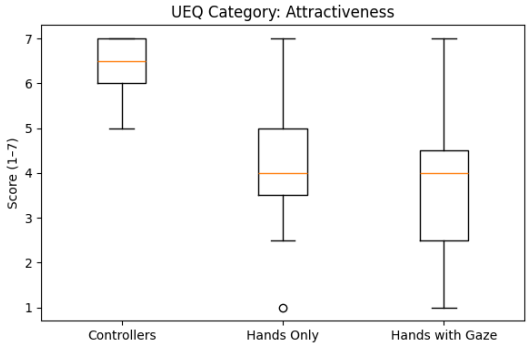


(a) Perspicuity

(b) Dependability



(c) Efficiency



(d) Attractiveness

Figure 5.4: Box-plots of UEQ category scores (Perspicuity, Dependability, Efficiency, Attractiveness) for each interaction mode (Controllers, Hands Only, Hands with Gaze).

The box plots show a clear trend, similar to the metrics: Controllers rank the highest in all categories. The participants rate the controllers as easy to learn and understand, as predictable and consistent, efficient, and generally well perceived. The average minimum score is 5, whereas the median is consistently 6.5 or higher.

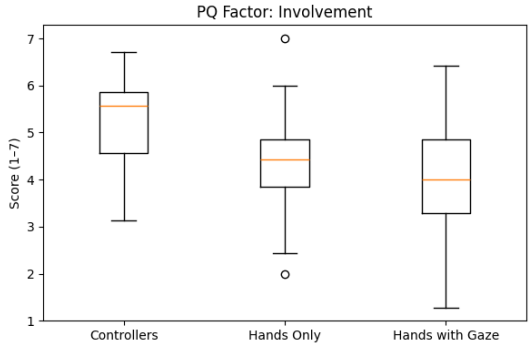
The hands only interaction mode shows a broader range of values, however, the Q_1 and Q_3 are consistently in the middle between 3 and 6. While the controllers are consistent in general, we see a lot of variance and outliers in hands only, as the mean maximum was 7 in all categories and the mean minimums can be as low as 1.

Hands with gaze has similar tendencies, though the dispersion is even greater. Evidently certain people liked it across all categories, as visible in the mean maximum values, while some really disliked it with the mean minimum values of 3 categories being only 1, with perspicuity's minimum being slightly above at 1.75. This indicates that opinions on hands with gaze are split and highly variable: the median being around the middle, but the Q_1 being below the neutral point 4 in all categories except for perspicuity, and the Q_3 being above the neutral point.

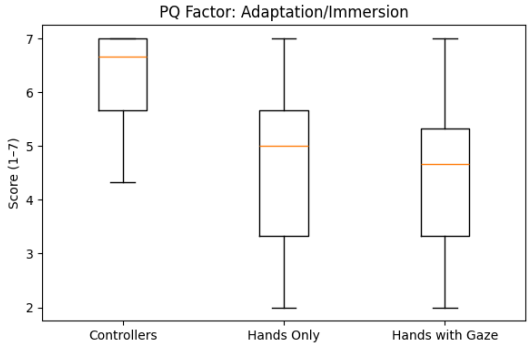
Based on the results, the controllers provide the most consistent experience that most participants agree on. Perceptions of hands and hands with gaze are more varied with many outliers and split opinions, with hands with gaze suffering to a higher degree than just hands. Though in general, hands only ranks slightly better with less variance. This implies that adding gaze tracking is not beneficial in general, though the questionnaire cannot specify whether it was the gaze interaction or gesture locomotion with gaze steering.

5.3.2 Presence Questionnaire

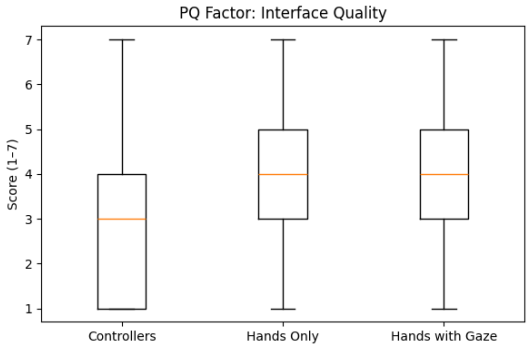
Our second questionnaire, the Presence Questionnaire, has three factors: Involvement, Adaptation/Immersion, and Interface Quality. The first two factors focus on the immersion and how involved—that is, how focused—the participants were, and the last category represents the quality of the interface devices, in our case the controllers and the gesture hand tracking implementation. The statistics of each question can be viewed individually in Appendix D, the statistics of the factors are presented in Fig. 5.5.



(a) Involvement



(b) Adaptation / Immersion



(c) Interface Quality

Figure 5.5: Box-plots of PQ factor scores (Involvement, Adaptation/Immersion, Interface Quality) for each interaction mode (Controllers, Hands Only, Hands with Gaze).

To keep the questionnaire consistent, the negative adjective and value was kept on the left side of the questionnaire. The Interface Quality factor's question was: "How much did the control devices interfere with the performance of assigned tasks or with other activities?" and its negative value: "Not at all" is perceived as a positive answer, hence the lower the value in Interface Quality, the better is the quality. Despite it being clearly stated in the questionnaire, we suspect that participants misunderstood it when facing the question for the first time, which was about the controllers. This leads to a mean maximum score of 7 for the controllers, indicating that controllers were highly interfering, whereas the mean, median, and quartiles are in the expected range.

The general trend of PQ is the same as with UEQ, with controllers ranking higher than the other two modes. We expected that controllers might rank lower, as holding controllers can be a negative point regarding immersion, though it seems the usability consistency of controllers outweighs the unnatural feeling of holding and using controllers compared to hand tracking. Though there is still a noticeable shift when using controllers in the aspect

of involvement, the box representing the quartiles and the median is lower, the mean maximum value is also not 7 or almost 7 as in UEQ. In terms of adaptation/immersion, controllers still generally rank really high, but the minimums are around 4.33 now.

Hands and hands with gaze again show a wide dispersion, with minimums and maximums in adaptation/immersion and interface quality being at 1 and 7 respectively. Interestingly, involvement is more consistent, with the quartiles and medians around the neutral point of 4, whereas in adaptation/immersion the spread is much wider in both directions. In terms of interface quality, they both have the same values.

The implications of the PQ questionnaire are generally the same as with UEQ. Hands and hands with gaze are worse, with hands being more consistent and hands with gaze reception being more spread out.

5.4 Interview Findings

Out of the 30 participants, 17 of them agreed to do an interview. To analyze the interviews we re-listened to them and extracted relevant information to find repeating patterns. We also noted down some less common, nonetheless interesting points that did not occur to us during designing and planning. Most participants were able to give clear answers, while some gave more vague answers or were contradicting themselves. The purpose of the interview is to address questions that the UEQ and PQ questionnaires are not able to, e.g. why is hands with gaze perceived so badly, is it the gaze steering or the gaze interaction. We now present our findings, categorized into locomotion, interaction, and recommendations for first time users, though we only present the clearly expressed answers.

5.4.1 Locomotion

Ten participants stated that gaze steering is more natural and preferred it to hand steering, on the other hand, only three participants preferred hand steering in general over gaze steering. Interestingly, cyber-sickness was experienced by five participants, but only when using hand steering. The cyber-sickness manifested when navigating through the house. Our hypothesis for this is that the mismatch of movement direction compared to the participants' gaze vision caused cyber-sickness when moving inside an enclosed space, which gives the illusion of faster movement because of the high speed of the rendered pixel movement.

Another common problem that we noticed, and six participants pointed out, is the high movement speed causing overshooting of the destination. One participant explicitly stated that controllers are good for precise movements, whereas four people implied this by stating that hand gesture locomotion is imprecise.

We conclude that gaze steering was better received than hand steering, however, it still has its flaws. Many participants got stuck in doorways. The reasoning is that the VR FOV—even with modern generations of HMD—is still small compared to natural human vision, hence participants tend to rotate their head when just about to enter the room to look around, which causes a steering input and they get stuck at the edge of the doorway. Additionally, when we move in reality, our entire body turns instead of just the head. While we noticed that many participants indeed use this whole body rotation for gaze steering for larger steering inputs, small steering inputs were done by the head. Afterwards, if they wanted to keep moving in that direction or look around, they had to rotate their body as well to restore the full head rotation arc. This fact was noticed by the supervisors, though only one participant explicitly expressed this as problematic. It was stated by

three participants that the ideal solution would be a combination of our two controller-less modes by utilizing gaze steering but keeping the interaction of hands only—in other words, removing the gaze interactor.

5.4.2 Interaction

Pure hands interaction was preferred by nine participants, and only two participants preferred the gaze interaction in general. There was quite an amount of negative feedback about the gaze interaction, eight participants expressed general dissatisfaction, stating that the ray was disturbing and not centered at eye level, which caused it to be inaccurate. Moreover, being forced to look down at objects felt uncomfortable for two participants, and five participants mentioned the gaze scrolling being unintuitive or "flipped." On the other hand, five participants admitted that being able to pick up objects from a distance using gaze and have them appear in their hands was a useful feature.

An interesting observation that did not occur to us, but was mentioned by two participants, is that having to use their gaze to interact caused a loss of focus when reading or scrolling. Any future work or implementation of gaze interaction should take this into account, as losing focus can lead to losing the train of thought and disruptions.

From the interviews regarding interaction, we can conclude that hand interaction was more favored by the participants.

5.4.3 Gestures

During testing it occurred multiple times that the hands of the supervisor were detected by the HMD, causing false input for the participant. Afterwards, we made sure that the supervisor stands far enough for it to not be a problem. Two participants mentioned this as a potential problem in the classroom use case, where multiple people would be using VR.

One participant brought up the concern of cultural meaning attached to different gestures, moreover, a different participant mentioned that common gesticulation performed during conversations might trigger accidental input, which is a concern we were aware of.

Regarding learning gestures, it was mentioned four times that learning specific gestures is hard. Additionally, seven participants stated that the gestures are too specific and their detection is inconsistent. We asked participants who were experienced with computer technology in general whether they thought that this is a limitation of the tracking technology or its use case implementation. The responses were mixed, some stating that the technology itself is not mature enough, while others said that bad implementation itself is the problem.

5.4.4 For First Time Users

We asked the participants whether they would recommend controllers or hand tracking based interaction and locomotion for first time VR users, specifically in an educational setting. Their responses were mixed: four participants stated that hands would be more intuitive in general, while two expressed that this would be the case only if first time users are not experienced with game console controllers.

On the other hand, five participants expressed that controllers are generally better suited for first time users. Only a single participant pointed out that learning controllers can be just as hard as gestures, especially for inexperienced users. The reasoning for

recommending controllers is the problem with gestures being hard to learn or their detection being inconsistent, as explained in Section 5.4.3.

An interesting notion mentioned by two participants is the fact that hand tracking attempts to emulate real life interaction with poking and pinching, which limits them. Meanwhile, controllers are not limited by such a notion and serve as an abstraction to real hand interaction, which leads to more creative abilities that are not possible in reality. On the contrary, a single participant expressed a counter notion that controllers have a limited set of input buttons, while hand tracking has a high number of gestures and gesture combinations that could be utilized.

5.4.5 Feasibility of VR Without Controllers

We asked the participants for their opinion on how feasible VR without controllers is. Five participants clearly stated that it is feasible, while two clearly stated that for now we might be limited by the technology, as their experience with hand tracking was inconsistent and unsatisfactory in the current state. The rest contemplated that it depends on the use case. It is feasible when there is not much interaction involved—for example, a tour or a visual explanation would be a good use case. However, whenever there is heavy interaction or locomotion involved, it falls short compared to the controllers.

An interesting observation provided by two participants is that hand tracking commonly attempts to emulate real life interaction with hands, whereas controllers serve as an abstract interface device and are not limited by this fact. We simply accept that controllers are not our hands and our inputs are not natural. This enables controllers to provide more unnatural capabilities, such as picking up objects from a distance, which we do not deem as weird. However, when using hand tracking, it attempts to emulate the interactions performed with our real hands while also attempting to emulate the capabilities of the controllers, such as picking up objects from afar using the near-far interactor or gaze interaction with pinch. This emulation is perceived as strange, as it breaks the natural feeling of hands. On the other hand, interaction with hands is still limited to the natural motion and limits of human hands, such as using gestures to initiate locomotion or to pick up objects. Seemingly a contradiction, nevertheless it is an interesting thought and design aspect that should be considered. Should we keep the hands as real as possible? In that case, the hands are severely limited compared to abstract controllers. If we attempt to emulate the controllers using hands, it is viewed as unnatural, and the range of gestures that needs to be learned causes a steep learning curve.

5.5 Limitations and Potential Improvements

After the user study had started and while observing the participants, we noticed some limitations of our user study and potential improvements. Since we could not modify the user study once it had started, we had to continue the study with these limitations present. The interviews highlighted even more possible problems and many participants provided good feedback on how to improve them. We now present these mentioned limitations and put forward possible improvements for future development.

5.5.1 Locomotion

Movement Speed

The main gripe with locomotion was the movement speed when using hand gesture locomotion. When moving between tasks that were on the other side of the designated area, the participants felt it was too slow and would have preferred some speed adjustment. This is the result of our deliberate decision to remove the sprint functionality to achieve consistent metric results. However, on the other hand, six out of 16 interviewees stated that the movement speed was too high, which caused them to overshoot their destination. Specifically they wanted to come to a UI element, but they either got too close to read it or they went through the UI element and ended up behind it. The supervisors also observed this problem even by people who did not explicitly mention it during the interview, so the actual count is higher. When using controllers, this issue is not present, as analogue joysticks allow participants to move at lower speeds by not tilting the joystick to the maximum value.

During internal testing we reduced the original movement speed of 3 (Unity units per second) to 2.5 hoping to alleviate this problem. Though the team that performed the testing consisted of experienced VR developers with many hours of experience, hence we did not encounter this problem. A simple solution to address this issue would be to lower the base movement speed even more and introduce the sprint functionality. A more elegant solution suggested by two participants would be the integration of an acceleration—inertia system. The current implementation immediately sets the user's speed to the set movement speed, however, with acceleration or inertia, the movement speed would gradually increase over a set amount of time until it reaches the maximal value. This enables the users to make small positional adjustments more easily. A similar mechanic could be implemented for ceasing locomotion as well, though the deceleration gradient would be much steeper compared to acceleration.

Another solution to high movement speed in regard to hand gesture movement would be to emulate the analogue movement speed values of the joysticks in a "quasi" way. As our forward movement is a palm-up hand gesture, in which the fingers need to be straight, it could be done in a quasi continuous manner where the current degree of finger curl maps to a certain movement speed. Alternatively, it can be done in a discrete manner: an interval of $(0\%, 33\%)$ finger curl corresponds to maximal movement speed, $[33\%, 66\%)$ curl corresponds to 50% movement speed, and $[66\%, 100\%]$ curl corresponds to 33% speed. This would also address the issue of gestures being too specific and hard to detect, as the current implementation expects fully straight fingers (with a small margin of error) to initiate locomotion.

Another improvement provided by the participants is to reduce backward movement speed. Currently both directions offer the same movement speed, however, backward movement is mostly used only for corrections when overshooting or colliding with a wall. Having backward movement at full speed causes participants to overshoot their destination twice: first in the forward direction, and then in the backward direction.

Hand and Gaze Steering

As described in our interviews, the participants found it hard to steer precisely in certain situations, such as entering rooms through a doorway. Additionally, as described in Section 2.2.2, hand steering is limited by the natural motion limit of the human wrist, specifically when performing outward rotation. Moreover, moving straight forward with

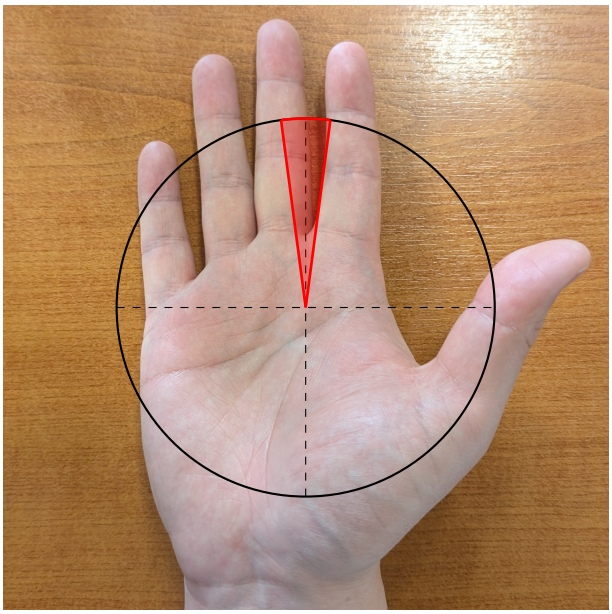


Figure 5.6: Visualization of the dead angle when using hand steering. Any rotation of the hand within the dead angle will be ignored.

hand steering is harder to achieve, as the most natural position when performing the palm-up gesture has a small hand rotation inward. A suggestion from one of the interviewees was to introduce a sensitivity to hand steering, i.e. a multiplication factor. Introducing a sensitivity multiplier could alleviate the limited hand rotation. To address straight forward locomotion, a dead angle can be utilized. For example, once locomotion is initiated, a hand rotation within a 15° angle would not lead to any steering input, hence the movement direction will stay the same until a larger rotation is performed. With a dead angle, there is some leeway to move the hand, so the user does not have to keep the hand as steady as possible when attempting to move straight forward. Nevertheless, based on participant feedback, hand steering was the least preferred steering method. A potential future work would be testing out these improvements to discern whether hand steering can be improved and used in practice, or simply discarded.

Regarding gaze steering, a similar dead angle can be introduced to enable moving in a straight line, as it is impossible to avoid minuscule head movements that could be detected by the HMD and cause miniature steering inputs.

5.5.2 Interaction

Interference of Interactors

A limitation that was overlooked by us is the fact that the near-far interactor is always active in the hands only interaction mode. This caused confusion and interference to the participants, especially when performing the Simon Says Task with a num-pad containing small buttons when they desired to use the poke interactor. The near-far interactor's ray would hit neighboring buttons and cause them to be highlighted, leading to confusion and sometimes false input. There exists a component specifically designed for Meta hand tracking, which disables the near-far interactor's ray when a poke gesture is performed, i.e. a straight index finger with the rest of the fingers curled. The component then re-enables the ray when the poking gesture is no longer performed. Unfortunately, this

feature was turned off for testing purposes and was never reactivated until we noticed the issues experienced by the participants during the study. At that point, we did not want to change the specifications of the user study, hence we could not activate the feature.

Gaze Interactor

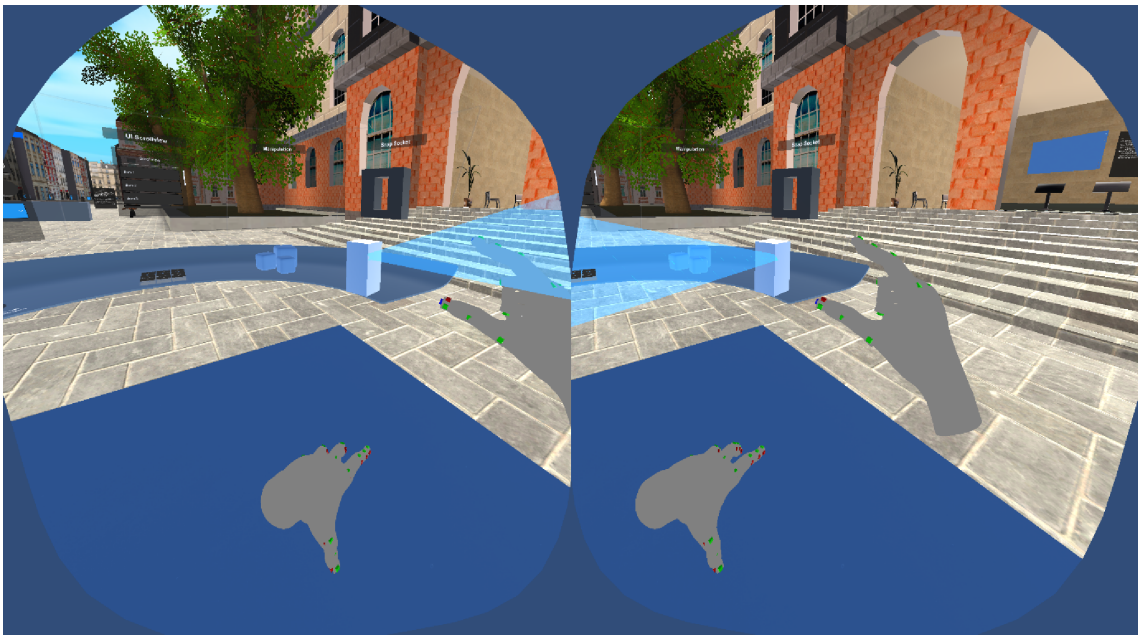


Figure 5.7: Both-eye view of the gaze interactor's ray.

The aiming ray's source is currently the main camera, i.e. the user's point of view. However, VR rendering produces two visuals, one for each eye. This caused the ray to be generated from between the eyes, as visible in Fig. 5.7. Moreover, due to facial differences and the interface gasket of the HMD, some participants saw the ray rendered above or below their vision level. The ray interactor was perceived as disturbing by eight participants. Several participants suggested an alternative aiming method: instead of rendering an aiming ray from between the eyes, render a dot or a crosshair at the ray's hit location. This way it does not interfere with vision and is easier to aim with gaze.

A possible design flaw that we noticed only during the user study, and that was mentioned by participants, is that the gaze scrolling is flipped compared to contemporary implementations of scrolling. To scroll down, participants had to look down, which is unintuitive, as on touchscreens and touch-pads one swipes up to scroll down. On the other hand, scrolling using a mouse scroll wheel is done in the opposite direction. During internal testing we did not perceive this as an issue, due to gaze scrolling not being directly comparable to touch scrolling, though five of 30 participants explicitly mentioned it.

5.5.3 User Study and Questionnaire Limitations

An oversight in our VR application was not including a pause function, in case a participant required a break due to cyber-sickness or other technical issues. This caused some of our metric data to be invalidated and could not be used for evaluation.

With the purpose of having as relevant questions in the UEQ and PQ questionnaires, we left out many questions from the original full questionnaires. Having to answer questions

that are only vaguely related can be frustrating and can lead to indecisive or inaccurate answers. On the other hand, this decision caused the Dependability, Efficiency, and Attractiveness scales in UEQ to have only two questions each. Similarly, in the PQ questionnaire, the only question in the Interface Quality factor is "How much did the control devices interfere with the performance of assigned tasks or with other activities?" while in the full questionnaire it has three associated questions. However, we believe that having relevant and accurate questions and answers is of more value than following the questionnaire's guideline perfectly. In addition, several papers criticize that the questions are not always suitable, as explained in Section 4.3, hence we deem this choice to be justified.

Chapter 6

Conclusion

6.1 Summary

The goal of this thesis was to determine whether hand tracking based alternatives to controllers are feasible for a comprehensive VR experience. Additionally, we wanted to determine if adding gaze tracking into the hand tracking is beneficial or not. To this end, we proposed two new interaction modes, called hands and hands with gaze. The former utilizes only hand tracking and the latter combines gaze tracking with hand tracking.

Our proposed interaction modes are designed to be a full replacement for the traditional controllers, encompassing both locomotion and interaction in VR. Locomotion in both modes is performed by hand gestures, they differ in the steering method. Hands use the direction and rotation of the hand as a steering input. In the hands with gaze mode, we use the well-known gaze steering, i.e., we move in the direction of our gaze. Interaction in hands is performed using hand gestures, specifically by pinching the index finger and thumb. It provides both long range interaction using a ray appearing out of the hands and close range interaction by using colliders on the hand and pinching These capabilities are analogous to the controllers, in order to serve as a replacement. In an attempt to incorporate gaze into interaction, we have designed a gaze interactor for the hands with gaze mode as a replacement for the long range interaction. The gaze interactor functions so: we can aim using our gaze at interactable objects and then perform a pinch with a hand to interact. As for close range interaction in hands with gaze, we use the same collider and pinching method as in hands only. The hand tracking technology utilizes the cameras of the Meta Quest series of VR HMDs.

In order to evaluate our proposal, we have designed and conducted a user study. Participants were invited and they had to perform three rounds of tasks, each round had a different interaction mode in a randomized order. In addition to our two proposed controller-less interaction modes, we also included controllers as a baseline to compare with. In each round, the participants performed the same five tasks, the order of the tasks was always randomized and each task had a randomization mechanic, so that each round was never the same as the previous one. The tasks were designed to explore specific aspects of locomotion, interaction, or both. Our user study application had metrics incorporated into it, which measured various performance statistics, such as time required, mistakes made, or number of interactions performed. These metrics served as our first method of analysis to compare the interaction modes.

After performing the tasks in VR, the participants filled out a questionnaire. It was composed of general questions regarding their experience and demographical data, afterwards, they had to fill out two scientific questionnaires: User Experience Questionnaire

(UEQ) and Presence Questionnaire (PQ), in which they had to rate their experiences on a scale. The results of the questionnaires served as our second method of analysis.

As questionnaires and metrics cannot cover every aspect or discover important observations or relationships, we also asked the participants to partake in a recorded interview. The interview was done in a semi-structured way, with predefined question areas, however, it was done more as an open conversation, in which the participants were guided to certain aspects of their experience and expressed their personal opinions. These interviews were the last method of our analysis in answering our research questions.

6.2 Results

Unexpectedly, the metrics showed the controllers to be the best mode, with the best metric values. The majority of participants performed best with controllers and worse with the other two modes, 20 out of the 30 participants were fastest with the controllers. Overall, we did not find a significant difference between hands and hands with gaze, there was less than 1 minute difference in the mean taskTimes between them, though there is roughly 4 minute difference between controllers and the other two modes. In the tasks themselves the metric values followed the same trend with little difference between hands and hand with gaze.

Based on the questionnaires, controllers ranked again as the best. Hands and hands with gaze were rated worse, however, here we saw a big variance: some people expressed strong dislike while others found it acceptable. The discrepancies and dispersion of hands with gaze were worse in almost every aspect compared to hands.

Lastly, the interviews confirmed that gaze interaction was heavily disliked, whereas gaze steering was viewed as favorable. Many participants were able to provide improvement ideas or gave interesting observations and points that did not occur to us while designing our proposal and user study. For example that movement speed control or that hand steering would be the culprit for cyber-sickness as it creates a big discrepancy between vision and movement.

Discussion

Based on all of these results and observations during the user study and feedback provided by the participants, we conclude that in general, hand tracking based VR is feasible, but how practical and successful it is depends on the application. It is not suitable for VR applications with heavy interaction required, as it was shown to be relatively inefficient compared to traditional controllers, as it was more time consuming. The tracking technology can also be inconsistent with our used HMDs.

Our opinion on hand tracking based gesture locomotion is tentative: our current implementation of our proposal is not suitable for general use, however, a newer iteration integrating the suggestions provided by the participants and our observations would make gesture locomotion feasible. The main improvement would be some mechanic to control the speed, as currently we have a constant speed, which is either active at full speed or not. This causes people to overshoot the destination or be unable to make small adjustments.

Another thing to consider is the gesture choice for both locomotion and interaction, as some specific gestures are simply unnatural and uncomfortable to hold for a longer period of time. Furthermore, humans naturally use gesticulation when talking and interacting with others, this can lead to inadvertent gesture detection when not desired. Lastly, one must consider the cultural aspect of gestures having different meanings in different cultures.

Lastly, the most important lesson we have learned is that controllers function so well because they are an abstract interface device, they are not supposed to be natural. As an abstraction, they are able to provide many abilities that are not possible in reality. With our current proliferation of technology, everyone has to interact with interface devices on a daily routine. Hand tracking, on the other hand, attempts to emulate the capabilities of the human hand and how we use them. Designing a system that is able to emulate every possible usage of the hands is not feasible, hence we are currently limited to gestures and hand position and rotation as sources of input. If we want hand tracking based interaction and locomotion to serve as a full replacement for controllers, then it should have the same capabilities as the controllers. But implementing all of the abstract functionalities of the controllers using only gestures, hand position, and rotation goes against the grain of how we use our hands in reality. Therefore we recommend for future research about replacing controllers with hand tracking and gestures, to attempt to model interactions as naturally as possible to real hand interactions. Specifically this would lead to ditching of long range interaction and keeping only the ability to poke and grab objects directly upon contact.

While many participants had a positive general impression of hand tracking, most likely because it's a novelty, their responses to our controller-less interaction modes were negative compared to the controllers, especially the hands with gaze mode, which confirms our recommendation from above. Therefore, our conclusion to our secondary question of: whether gaze tracking is beneficial in general is no. This is because of the unnaturalness of the gaze interactor, though gaze steering was in fact perceived positively.

The tasks in our study were designed so that specific aspects of locomotion and interaction could be measured, their content was general, comprising UI interaction, grabbing, simple door interaction, and locomotion. With these tasks, our study clearly showed that controllers were the superior mode. Similar research yields varying results. Kim [Kim22] implements hand tracking interaction and compares it to controllers, they conclude that each mode is suitable for different types of interactions—hand tracking for grabbing and controllers for browsing. Similarly, Johnson et al. [JFP+23] compare hand tracking based interaction to controllers, their findings match ours in that hand tracking is perceived as worse than controllers. Interestingly, other research projects find no significant differences between controllers and hand tracking in the context of medical procedure training [CNFS21, KVP+21].

Regarding hand tracking gesture based locomotion, Zhao et al. [ZAXL22] compare game-pad controllers to gesture based locomotion and find no significant differences. Cardoso et al. [Car16] perform a comparison between controllers, gesture based locomotion with hand steering (similar to ours), and gesture based locomotion with gaze steering. Their findings match ours: hand steering was cumbersome to use, gaze steering was better, and the game-pad was the best locomotion method. As mentioned before, to the best of our knowledge, there does not exist a work that comprehensively attempts to replace controllers for both interaction and locomotion, hence the presented works cover either locomotion or interaction only.

Further Research

Based on our research, gaze based interaction was perceived negatively, further research should be conducted on how to improve it so that it is more natural and useful. In our implementation, gaze aiming was represented by a line, so participants had to carefully and accurately aim with their gaze. This could be changed to a cone, though that can cause ambiguity in selection if multiple interactable objects are close to each other. An

improvement suggested by the participants is to replace the rendered line with a dot or a crosshair at the target hit location, which seems promising in eliminating the current discomfort experienced by our participants. Another commonly research avenue is using eye tracking, replacing our gaze tracking based interaction with an eye tracking based one would alleviate the problem of precise aiming with the gaze, though it just replaces it precise aiming with the eyes and the associated eye strain. Though the downside of eye tracking capable HMD is their high cost. Since the tasks in our user study were more general, further research should attempt to utilize a next iteration of hand tracking and possibly gaze tracking based interaction and locomotion in a proper teaching application.

Bibliography

[AML+20] Sahar Aseeri, Sebastian Marin, Richard N. Landers, Victoria Interrante, and Evan S. Rosenberg. Embodied realistic avatar system with body motions and facial expressions for communication in virtual reality applications. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 580–581, 2020.

[AML+24] Nur Asitah, Arie Widya Murni, Wahyu Maulida Lestari, Nurul Aini, and Hikmah Luqiyah Kartikasari. Virtual reality in inclusive basic education: A systematic review of roles and application for future education directions. In *2024 International Conference on ICT for Smart Society (ICISS)*, pages 1–6, 2024.

[BB21] Evren Bozgeyikli and Lal Lila Bozgeyikli. Evaluating object manipulation interaction techniques in mixed reality: Tangible user interfaces and gesture. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 778–787, 2021.

[BHK22] Paweł Buń, Jozef Husár, and Jakub Kaščak. Hand tracking in extended reality educational applications. In Justyna Trojanowska, Agnieszka Kujawińska, Jose Machado, and Ivan Pavlenko, editors, *Advances in Manufacturing III*, pages 317–325, Cham, 2022. Springer International Publishing.

[BHN21] Reigo Ban, Yutaro Hirao, and Takuji Narumi. Determining the target point of the mid-air pinch gesture. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 492–493, 2021.

[BKH97] D.A. Bowman, D. Koller, and L.F. Hodges. Travel in immersive virtual environments: an evaluation of viewpoint motion control techniques. In *Proceedings of IEEE 1997 Annual International Symposium on Virtual Reality*, pages 45–52, 1997.

[BKSS20] Liudmila Bredikhina, Takayuki Kameoka, Shogo Shimbo, and Akihiko Shirai. Avatar driven vr society trends in japan. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 497–503, 2020.

[BPdLM23] Daniela Rocha Bicalho, João Manuel Nunes Piedade, and João Filipe de Lacerda Matos. The use of immersive virtual reality in educational practices in higher education: A systematic review. In *2023 International Symposium on Computers in Education (SIIE)*, pages 1–5, 2023.

[BTG+24] Brigida Bonino, Elia Moscoso Thompson, Franca Giannini, Marina Monti, and Katia Lupinetti. Toward intuitive locomotion techniques in vr: Fully natural and semi-natural interaction. In *2024 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, pages 359–364, 2024.

[BWK20] Pauline Bimberg, Tim Weissker, and Alexander Kulik. On the usage of the simulator sickness questionnaire for virtual reality research. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 464–467, 2020.

[Car16] Jorge C. S. Cardoso. Comparison of gesture, gamepad, and gaze-based locomotion for vr worlds. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, VRST '16, page 319–320, New York, NY, USA, 2016. Association for Computing Machinery.

[CCB+24] Raymond Chandra, Koen Castermans, Djamel Berkaoui, Patrick Querl, and Nacken Heribert. Creating realistic human avatars for social virtual environments using photographic inputs. *Journal of Information Systems and Informatics*, 6(3):2110–2129, Sep. 2024.

[CNFS21] Khundam Chaowanan, Sukkriang Naparat, Noël Frédéric, and Huh Sun. No difference in learning outcomes and usability between using controllers and hand tracking during a virtual reality endotracheal intubation training for medical students in thailand. *jeehp*, 18(0):22–0, 2021.

[CUP+22] Sohan Chowdhury, A K M Amanat Ullah, Nathan Bruce Pelmore, Pourang Irani, and Khalad Hasan. Wriarm: Leveraging wrist movement to design wrist+arm based teleportation in vr. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 317–325, 2022.

[DARR25] Annalisa Degenhard, Ali Askari, Michael Rietzler, and Enrico Rukzio. When do we feel present in a virtual reality? towards sensitivity and user acceptance of presence questionnaires. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, page 1–29. ACM, April 2025.

[DNN14] Simon Davis, Keith Nesbitt, and Eugene Nalivaiko. A systematic review of cybersickness. In *Proceedings of the 2014 Conference on Interactive Entertainment*, IE2014, page 1–9, New York, NY, USA, 2014. Association for Computing Machinery.

[Dun61] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[FY48] Ronald A. Fisher and Frank Yates. *Statistical Tables for Biological, Agricultural and Medical Research*. 3rd edition, 1948. [Original 1938].

[Gir92] E.R. Girden. *ANOVA: Repeated Measures*. Number no. 84 in ANOVA: Repeated Measures. SAGE Publications, 1992.

[GSS+21] Armin Gruenewald, Ricardo Schmidt, Lukas Sayn, Christian Gießer, Tanja Joan Eiler, Vanessa Schmuecker, Veit Braun, and Rainer Brueck.

Virtual reality training application to prepare medical student's for their first operating room experience. In *2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 201–204, 2021.

[Hoa03] David Hoaglin. John w. tukey and data analysis. *Statistical Science*, 18, 08 2003.

[HPM21] Asim Hameed, Andrew Perkis, and Sebastian Möller. Evaluating hand-tracking interaction for performing motor-tasks in vr learning environments. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 219–224, 2021.

[HS88] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Peter A. Hancock and Najmedin Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183. North-Holland, 1988.

[HVH+23] Jan Hombeck, Henrik Voigt, Timo Heggemann, Rabi R. Datta, and Kai Lawonn. Tell me where to go: Voice-controlled hands-free locomotion for virtual reality systems. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 123–134, 2023.

[JFP+23] Cheryl I. Johnson, Nicholas W. Fraulini, Eric K. Peterson, Jacob Entinger, and Daphne E. Whitmer. Exploring hand tracking and controller-based interactions in a vr object manipulation task. In *HCI International 2023 – Late Breaking Papers: 25th International Conference on Human-Computer Interaction, HCII 2023, Copenhagen, Denmark, July 23–28, 2023, Proceedings, Part V*, page 64–81, Berlin, Heidelberg, 2023. Springer-Verlag.

[Kim22] Aelee Kim. A comparative study of the user experience of controller and hand-tracking interactions in a virtual environment. In *2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 744–748, 2022.

[KJKa23] Jinwook Kim, Hyunyoung Jang, Dooyoung Kim, and Jeongmi Lee and. Exploration of the virtual reality teleportation methods using hand-tracking, eye-tracking, and eeg. *International Journal of Human–Computer Interaction*, 39(20):4112–4125, 2023.

[KLBM93] Robert S. Kennedy, Norman E. Lane, Kevin S. Berbaum, and Lilienthal Mg. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3:203–220, 1993.

[KML+25] Donggyu Kim, Viduranga Munasinghe, Tae-Ho Lee, Hyun-Jun Jin, Tae Sung Kim, Jin-Sung Kim, and Hyuk-Jae Lee. Enhancing ar/vr experiences with a pinch-gesture multi-layer virtual keyboard system. In *2025 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–3, 2025.

[KVP+21] Chaowanan Khundam, Varunyu Vorachart, Patibut Preeyawongsakul, Witthaya Hosap, and Frédéric Noël. A comparative study of interaction time and

usability of using controllers and hand tracking in virtual reality training. *Informatics*, 8(3):60, 2021.

[LHM22] Geonsun Lee, Jennifer Healey, and Dinesh Manocha. Vrdoc: Gaze-based interactions for vr reading experience. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 787–796, 2022.

[LHS08] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. In Andreas Holzinger, editor, *HCI and Usability for Education and Work*, pages 63–76, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[MBB24] Jyothinadh Minnekanti, Natasha Kholgade Banerjee, and Sean Banerjee. Classesinvr: An immersive vr-based classroom. In *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 310–314, 2024.

[MMN+21] Faizah Maarof, Hizmawati Madzin, Noris Mohd Norowi, Puteri Suhaiza Sulaiman, and Mas Nida Md Khambari. Designing touchless hand gestures interactions on desktop virtual reality (dvr) in classroom: Understanding issues through thematic analysis. In *2021 7th International HCI and UX Conference in Indonesia (CHIuXiD)*, volume 1, pages 6–11, 2021.

[MP21] Guido Makransky and Gustav Petersen. The cognitive affective model of immersive learning (camil): A theoretical research-based model of learning in immersive virtual reality. *Educational Psychology Review*, 33, 09 2021.

[MSB+14] Siddhesh Manjrekar, Shubhrika Sandilya, Deesha Bhosale, Sravanthi Kanchi, Adwait Pitkar, and Mayur Gondhalekar. Cave: An emerging immersive technology – a review. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 131–136, 2014.

[MSU+25] Ifigeneia Mavridou, Ellen Seiss, Giuseppe Ugazio, Mark Harpster, Phillip Brown, Sophia Cox, Christine Erie, David Lopez Jr, Ryan Copt, Charles Nduka, James Hughes, Joseph Butera, and Daniel N Weiss. "did you hear that?": Investigating localized spatial audio effects on enhancing immersive and affective user experiences in virtual reality. In *2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 1276–1277, 2025.

[MUNR23] Markus Murtinger, Jakob C Uhl, Quynh Nguyen, and Georg Regal. Tangible tactical belt: Haptic realism for virtual reality police training. In *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, pages 599–604, 2023.

[Nor10] Donald A. Norman. Natural user interfaces are not natural. *Interactions*, 17(3):6–10, May 2010.

[PLLB17] Thammathip Piumsomboon, Gun Lee, Robert W. Lindeman, and Mark Billinghurst. Exploring natural eye-gaze-based interaction for immersive virtual reality. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 36–39, 2017.

[PMMG17] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. Gaze + pinch interaction in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction*, SUI '17, page 99–108, New York, NY, USA, 2017. Association for Computing Machinery.

[QCB+24] Patrick Querl, Raymond Leonardo Chandra, Djamel Berkaoui, Koen Castermans, and Heribert Nacken. Does self-paced learning in mobile flood protection unit construction in virtual reality have advantages over traditional measures? *Frontiers in Virtual Reality*, Volume 5 - 2024, 2024.

[QT18] Yuan Yuan Qian and Robert J. Teather. Look to go: An empirical evaluation of eye-based travel in virtual reality. In *Proceedings of the 2018 ACM Symposium on Spatial User Interaction*, SUI '18, page 130–140, New York, NY, USA, 2018. Association for Computing Machinery.

[RMFW20] Jaziar Radianti, Tim A. Majchrzak, Jennifer Fromm, and Isabell Wohlgenannt. A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & Education*, 147, 04 2020.

[RR22] Maximilian Rettinger and Gerhard Rigoll. Defuse the training of risky tasks: Collaborative training in xr. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 695–701, 2022.

[RRG+02] Patrice Renaud, Joanne-Lucine Rouleau, Luc Granger, Ian Barsetti, and Stéphane Bouchard. Measuring sexual preferences in virtual reality: A pilot study. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 5:1–9, 03 2002.

[SKS23] Monu Sharma, Pradeep Kumar, and Deepak Kumar Singh. The role of virtual reality in education: A comprehensive review of research and application. In *2023 1st DMIHER International Conference on Artificial Intelligence in Education and Industry 4.0 (IDICAIEI)*, volume 1, pages 1–6, 2023.

[SKT+17] Valentin Schwind, Pascal Knierim, Cagri Tasci, Patrick Franczak, Nico Haas, and Niels Henze. "these are not my hands!": Effect of gender on the perception of avatar hands in virtual reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1577–1582, New York, NY, USA, 2017. Association for Computing Machinery.

[Sla09] Mel Slater. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 364:3549–57, 12 2009.

[SST21] Martin Schrepp, Heike Sandkühler, and Jörg Thomaschewski. How to create short forms of ueq+ based questionnaires? Mensch und Computer 2021 - Workshopband, 2021.

[SW65] S. S. SHAPIRO and M. B. WILK. An analysis of variance test for normality (complete samples)†. *Biometrika*, 52(3-4):591–611, 12 1965.

[SW97] Mel Slater and Sylvia Wilbur. A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments. *Presence: Teleoperators and Virtual Environments*, 6(6):603–616, 12 1997.

[SW25] Adrian Sarbach and Thierry Robin Weber. Next-generation navigation: Evaluating the impact of augmented reality on situation awareness in general aviation cockpits. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery.

[TAA+24] Vishesh Tanwar, Vatsala Anand, Priyanshi Aggarwal, Mukesh Kumar, and Gotte Ranjith Kumar. Revolutionizing military training: A comprehensive review of tactical and technical training through augmented reality, virtual reality, and haptics. In *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, pages 1–5, 2024.

[TBL11] J. Tichon and Robin Burgess-Limerick. A review of virtual reality as a medium for safety related training in mining. *Journal of Health & Safety Research & Practice*, 3:33–40, 01 2011.

[VBFLFC24] Aida Vidal-Balea, Paula Fraga-Lamas, and Tiago M. Fernández-Caramés. Advancing nasa-tlx: Automatic user interaction analysis for workload evaluation in xr scenarios. In *2024 IEEE Gaming, Entertainment, and Media Conference (GEM)*, pages 1–6, 2024.

[WFK24] Tim Weissker, Matthis Franzgrote, and Torsten Kuhlen. Try this for size: Multi-scale teleportation in immersive virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2298–2308, 2024.

[WJS05] Bob G. Witmer, Christian J. Jerome, and Michael J. Singer. The factor structure of the presence questionnaire. *Presence: Teleoper. Virtual Environ.*, 14(3):298–312, June 2005.

[WLC+23] Ziyao Wang, Chiyi Liu, Jialiang Chen, Yao Yao, Dazheng Fang, Zhiyi Shi, Rui Yan, Yiye Wang, KanJian Zhang, Hai Wang, and Haikun Wei. Strolling in room-scale vr: Hex-core-mk1 omnidirectional treadmill. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5538–5555, 2023.

[WS98] Bob G. Witmer and Michael J. Singer. Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoper. Virtual Environ.*, 7(3):225–240, June 1998.

[ZAXL22] Jingbo Zhao, Ruize An, Ruolin Xu, and Banghao Lin. Comparing hand gestures and a gamepad interface for locomotion in virtual environments. *International Journal of Human-Computer Studies*, 166:102868, 2022.

[ZCP+17] Fan Zhang, Shaowei Chu, Ruifang Pan, Naye Ji, and Lian Xi. Double hand-gesture interaction for walk-through in vr environment. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 539–544, 2017.

[ZMF+24] Daniel Zielasko, Alexander Meißner, Sebastian Freitag, Benjamin Weyers, and Torsten W. Kuhlen. Dynamic field of view reduction related to subjective sickness measures in an hmd-based data analysis task, 2024.

Appendix A

A.1 Descriptive Statistics of Metrics

Table A.1: Mode Comparison Summary (Fixed Order)

Metric	Mode	Mean	Std	Var	Med	Min	Q_1	Q_3	Max
HouseTaskManager.averageSpeed	Controllers	1.78	0.55	0.31	1.79	0.61	1.51	2.14	3.06
	Hands	1.2	0.28	0.08	1.21	0.54	1.02	1.33	1.75
	Gaze	1.17	0.32	0.10	1.10	0.60	1.03	1.39	2.16
HouseTaskManager.avgPickUpTime (min)	Controllers	1.01	0.64	0.40	0.88	0.32	0.50	1.34	3.02
	Hands	1.13	0.41	0.17	1.10	0.54	0.82	1.28	2.43
	Gaze	1.02	0.49	0.24	0.94	0.29	0.82	1.16	3.10
HouseTaskManager.doorClicks	Controllers	33.11	14.88	221.51	35.00	5.00	23.00	43.50	59.00
	Hands	31.86	18.68	348.87	27.50	11.00	16.00	41.50	82.00
	Gaze	32.31	14.19	201.44	31.00	4.00	24.00	41.00	60.00
HouseTaskManager.totalTime (min)	Controllers	3.38	2.61	6.79	2.67	1.14	1.80	3.86	12.74
	Hands	3.58	1.18	1.38	3.37	1.49	2.72	4.64	6.23
	Gaze	3.36	1.87	3.49	3.02	0.85	2.09	3.76	9.50

Metric	Mode	Mean	Std	Var	Med	Min	Q_1	Q_3	Max
HouseTaskManager.userDistanceTraveled	Controllers	305.40	143.74	20661.84	258.53	124.85	208.15	350.93	776.05
	Hands	258.92	74.71	5582.31	239.41	151.33	209.16	306.42	442.46
	Gaze	215.69	87.96	7736.66	190.54	110.68	156.34	245.15	477.87
MoveTaskManager.averageArrivalTime (sec)	Controllers	5.53	1.02	1.05	5.40	3.53	4.85	6.08	8.43
	Hands	5.94	1.31	1.72	5.85	3.73	4.97	6.65	8.74
	Gaze	6.75	2.75	7.58	6.07	3.86	5.27	7.31	17.59
MoveTaskManager.optimalPath	Controllers	48.05	7.27	52.91	49.22	34.71	41.98	52.89	62.19
	Hands	47.32	6.86	47.03	46.72	34.84	41.18	50.96	60.40
	Gaze	46.96	6.50	42.22	46.74	31.22	43.82	51.65	59.02
MoveTaskManager.totalTime (sec)	Controllers	28.81	5.38	28.92	29.26	17.64	25.39	31.30	42.18
	Hands	31.80	7.40	54.70	31.59	18.68	26.71	35.97	51.47
	Gaze	34.49	12.51	156.39	30.97	21.53	26.37	37.16	70.80
MoveTaskManager.userDistanceTraveled	Controllers	57.34	9.84	96.83	55.97	40.17	51.27	62.28	80.20
	Hands	55.15	8.44	71.19	53.66	41.88	48.40	60.78	72.85
	Gaze	55.77	8.03	64.41	56.17	40.44	50.37	59.66	76.09
MoveTaskManager.userWrongPlates	Controllers	0.37	0.67	0.45	0.00	0.00	0.00	0.75	2.00
	Hands	0.67	1.12	1.26	0.00	0.00	0.00	1.00	5.00
	Gaze	0.57	1.19	1.43	0.00	0.00	0.00	1.00	5.00
ShapeManager.avgFillTime (sec)	Controllers	21.55	8.39	70.38	19.25	8.61	16.66	26.41	39.24
	Hands	31.83	11.78	138.69	29.11	8.25	26.45	35.03	64.36
	Gaze	41.85	21.68	470.12	36.62	16.39	27.45	47.13	102.60
ShapeManager.totalGrabs	Controllers	7.17	4.11	16.90	5.50	4.00	4.00	9.00	23.00
	Hands	8.87	3.76	14.12	8.50	4.00	6.00	11.00	18.00
	Gaze	11.90	7.26	52.71	9.00	4.00	8.00	14.50	35.00
ShapeManager.totalTime (min)	Controllers	1.45	0.56	0.31	1.35	0.57	1.12	1.76	2.62
	Hands	2.24	0.72	0.52	2.00	1.28	1.78	2.49	4.29

Metric	Mode	Mean	Std	Var	Med	Min	Q_1	Q_3	Max
	Gaze	3.02	1.59	2.54	2.65	1.12	2.10	3.21	8.55
	Controllers	1.40	1.87	3.49	1.00	0.00	0.00	2.00	7.00
ShapeManager.wrongAttemptsCount	Hands	0.90	0.99	0.99	1.00	0.00	0.00	1.00	3.00
	Gaze	1.00	0.95	0.90	1.00	0.00	0.00	2.00	3.00
	Controllers	1.31	0.78	0.60	1.05	0.58	0.86	1.42	4.47
SimonSaysManager.averageInterClickTime (sec)	Hands	2.83	1.41	1.98	2.30	0.89	2.09	3.77	6.67
	Gaze	2.00	0.74	0.55	1.85	0.96	1.44	2.25	3.90
	Controllers	6.75	1.92	3.69	6.00	6.00	6.00	6.00	13.50
SimonSaysManager.forcedDelayTime (sec)	Hands	10.00	6.08	36.98	8.25	6.00	6.00	12.00	37.50
	Gaze	7.90	3.20	10.23	6.00	6.00	6.00	7.50	16.50
	Controllers	22.21	13.83	191.14	17.45	12.69	15.21	20.78	81.96
SimonSaysManager.netTime (sec)	Hands	58.04	51.56	2658.12	45.16	15.43	31.22	65.38	296.39
	Gaze	34.79	19.33	373.62	26.63	15.99	22.51	41.67	92.90
	Controllers	9.00	0.00	0.00	9.00	9.00	9.00	9.00	9.00
SimonSaysManager.optimalClicks	Hands	9.00	0.00	0.00	9.00	9.00	9.00	9.00	9.00
	Gaze	9.00	0.00	0.00	9.00	9.00	9.00	9.00	9.00
	Controllers	28.96	15.42	237.71	23.45	18.69	21.21	27.17	93.96
SimonSaysManager.totalTime (sec)	Hands	68.04	57.27	3279.48	53.68	21.43	37.84	72.37	333.89
	Gaze	42.70	22.26	495.66	33.39	21.99	28.51	48.05	107.90
	Controllers	9.90	2.19	4.78	9.00	9.00	9.00	9.00	18.00
SimonSaysManager.userClicks	Hands	14.43	8.38	70.19	12.50	9.00	9.00	17.00	52.00
	Gaze	11.77	5.13	26.32	9.00	9.00	9.00	11.00	28.00
	Controllers	0.40	1.04	1.08	0.00	0.00	0.00	0.00	4.00
SimonSaysManager.userWrongClicks	Hands	1.47	2.24	5.02	1.00	0.00	0.00	2.00	11.00
	Gaze	0.73	1.20	1.44	0.00	0.00	0.00	1.00	5.00
	Controllers	4.17	3.13	9.79	3.00	2.00	3.00	4.00	14.00
UITaskManager.buttonClickCount									

Metric	Mode	Mean	Std	Var	Med	Min	Q_1	Q_3	Max
	Hands	5.69	4.78	22.86	4.00	2.00	2.00	7.00	19.00
	Gaze	3.45	1.84	3.40	3.00	2.00	2.00	4.00	11.00
	Controllers	4.07	2.83	8.00	3.00	2.00	3.00	4.00	13.00
UITaskManager.totalPageCount	Hands	5.48	4.55	20.69	4.00	2.00	2.00	7.00	19.00
	Gaze	3.38	1.68	2.82	3.00	2.00	2.00	4.00	10.00
	Controllers	1.08	0.61	0.37	0.96	0.36	0.73	1.20	2.78
UITaskManager.totalTime (min)	Hands	2.01	1.32	1.73	1.51	0.33	1.12	3.09	4.61
	Gaze	1.80	0.99	0.97	1.62	0.48	1.17	2.43	4.14
	Controllers	5.02	2.60	6.74	4.51	1.79	3.18	6.35	11.78
familiarizationTime (min)	Hands	5.60	1.96	3.83	5.13	2.86	4.32	7.23	9.31
	Gaze	4.89	2.86	8.21	3.98	0.88	2.70	6.38	9.89
	Controllers	9.77	3.96	15.70	8.18	4.53	6.78	12.54	17.70
taskTime (min)	Hands	13.24	3.61	13.02	12.58	7.67	10.07	16.49	19.12
	Gaze	13.29	5.56	32.03	11.60	6.46	9.68	16.71	29.58

Appendix B

B.1 Movement Task Distance Travelled vs. Optimal Path

Table B.1: Controllers: MoveTaskManager Distance vs Optimal Path

Participant	Traveled	Optimal	Difference
20250328_113633	51.21	43.69	7.52
20250328_142225	49.71	41.58	8.13
20250331_105813	74.35	49.37	24.98
20250402_131425	58.64	53.45	5.19
20250402_145153	54.17	49.39	4.78
20250403_114842	52.23	38.35	13.88
20250403_135837	63.74	49.06	14.68
20250407_153940	80.20	59.02	21.18
20250410_123129	59.73	53.31	6.42
20250411_104531	54.18	50.07	4.11
20250418_165413	41.50	36.11	5.39
20250420_170640	73.62	53.63	19.99
20250425_133613	55.83	50.25	5.58
20250428_130631	44.64	38.88	5.76
20250429_115054	50.52	46.81	3.71
20250502_092821	40.17	34.71	5.46
20250505_131033	60.91	41.04	19.87
20250506_102752	64.69	57.63	7.06
20250507_103230	58.04	47.52	10.52
20250507_130204	75.37	62.19	13.18
20250509_130146	46.11	40.46	5.65
20250512_104249	59.04	51.61	7.43
20250512_123340	54.24	43.57	10.67
20250512_142810	51.43	46.95	4.48
20250514_104926	62.74	57.73	5.01
20250519_125715	65.91	59.64	6.27
20250520_090351	56.00	43.17	12.83
20250521_090644	58.44	51.17	7.27
20250521_141325	46.78	40.84	5.94
20250521_160458	55.93	50.31	5.62

Summary: Mean = 9.29, Min = 3.71, Max = 24.98; Variance = 33.11, Std dev. = 5.75, Mean % increase over optimal 19.53%

Table B.2: Hands: MoveTaskManager Distance vs Optimal Path

Participant	Traveled	Optimal	Difference
20250328_113633	48.45	41.12	7.33
20250328_142225	72.85	58.63	14.22
20250331_105813	72.35	57.64	14.71
20250402_131425	46.23	40.60	5.63
20250402_145153	50.27	44.04	6.23
20250403_114842	59.74	48.97	10.77
20250403_135837	45.00	37.46	7.54
20250407_153940	63.00	54.37	8.63
20250410_123129	66.49	59.30	7.19
20250411_104531	51.51	42.99	8.52
20250418_165413	58.95	50.06	8.89
20250420_170640	58.08	50.98	7.10
20250425_133613	51.89	46.67	5.22
20250428_130631	61.08	53.16	7.92
20250429_115054	48.38	41.08	7.30
20250502_092821	68.35	60.40	7.95
20250505_131033	53.85	46.77	7.08
20250506_102752	42.19	37.22	4.97
20250507_103230	47.44	40.92	6.52
20250507_130204	59.88	50.91	8.97
20250509_130146	45.25	41.04	4.21
20250512_104249	52.75	46.43	6.32
20250512_123340	41.88	34.84	7.04
20250512_142810	62.22	54.46	7.76
20250514_104926	47.72	41.35	6.37
20250519_125715	56.87	49.25	7.62
20250520_090351	53.48	47.00	6.48
20250521_090644	52.84	45.06	7.78
20250521_141325	61.60	50.36	11.24
20250521_160458	53.94	46.51	7.43

Summary: Mean = 7.83, Min = 4.21, Max = 14.71; Variance = 5.46, Std dev. = 2.34, Mean % increase over optimal 16.49%

Table B.3: Gaze: MoveTaskManager Distance vs Optimal Path

Participant	Traveled	Optimal	Difference
20250328_113633	59.58	54.07	5.51
20250328_142225	50.30	31.22	19.08
20250331_105813	58.93	41.86	17.07
20250402_131425	50.57	45.40	5.17
20250402_145153	52.41	43.95	8.46
20250403_114842	56.23	51.03	5.20
20250403_135837	44.42	36.83	7.59
20250407_153940	63.56	51.86	11.70
20250410_123129	59.68	50.05	9.63
20250411_104531	61.61	52.91	8.70
20250418_165413	57.22	50.07	7.15
20250420_170640	54.49	46.48	8.01
20250425_133613	51.94	46.17	5.77
20250428_130631	52.72	45.59	7.13
20250429_115054	60.65	54.51	6.14
20250502_092821	46.98	40.77	6.21
20250505_131033	76.09	47.00	29.09
20250506_102752	48.45	43.77	4.68
20250507_103230	43.39	39.54	3.85
20250507_130204	40.44	34.86	5.58
20250509_130146	50.03	44.80	5.23
20250512_104249	59.45	52.97	6.48
20250512_123340	69.03	59.02	10.01
20250512_142810	69.64	54.81	14.83
20250514_104926	56.86	49.45	7.41
20250519_125715	56.11	49.83	6.28
20250520_090351	61.34	54.64	6.70
20250521_090644	58.89	50.56	8.33
20250521_141325	56.12	44.24	11.88
20250521_160458	45.99	40.68	5.31

Summary: Mean = 8.81, Min = 3.85, Max = 29.09; Variance = 27.75, Std dev. = 5.27, Mean % increase over optimal 19.35

Appendix C

C.1 UEQ Statistics

Table C.1: Detailed UEQ Question Statistics by Interaction Mode

Note: Each UEQ item has the format "Using <*mode*> was . . . ", e.g. "Using controllers only was. . . ".

Mode	Q #	**Adjective Pair**	**Mean**	**Median**	**Std**	**Variance**	Q_1	Q_3	**Min**	**Max**
Controllers	1	Not understandable ⟷ Understandable	6.72	7.00	0.45	0.21	6.00	7.00	6.00	7.00
	2	Difficult to learn ⟷ Easy to learn	6.66	7.00	0.67	0.45	7.00	7.00	5.00	7.00
	3	Complicated ⟷ Easy	6.55	7.00	0.78	0.61	6.00	7.00	5.00	7.00
	4	Unpredictable ⟷ Predictable	6.28	6.00	0.88	0.78	6.00	7.00	4.00	7.00
	5	Does not meet expectations ⟷ Meets expectations	6.48	7.00	0.63	0.40	6.00	7.00	5.00	7.00

Continued on next page

Table C.1 — continued from previous page

Mode	Q #	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
	6	Impractical ⟷ Practical	6.48	7.00	0.78	0.62	6.00	7.00	5.00	7.00
	7	Inefficient ⟷ Efficient	6.45	7.00	0.74	0.54	6.00	7.00	5.00	7.00
	8	Confusing ⟷ Clear	6.55	7.00	0.69	0.47	6.00	7.00	4.00	7.00
	9	Annoying ⟷ Enjoyable	6.17	6.00	0.85	0.72	6.00	7.00	4.00	7.00
	10	Unpleasant ⟷ Pleasant	6.52	7.00	0.57	0.33	6.00	7.00	5.00	7.00
Controllers → UEQ Category Scores										
Controllers	Perspicuity	(Q 1, 2, 3, 8)	6.62	6.75	0.47	0.22	6.25	7.00	5.50	7.00
	Dependability	(Q 4, 5)	6.38	6.50	0.61	0.37	6.00	7.00	5.00	7.00
	Efficiency	(Q 6, 7)	6.47	7.00	0.63	0.39	6.00	7.00	5.00	7.00
	Attractiveness	(Q 9, 10)	6.34	6.50	0.58	0.34	6.00	7.00	5.00	7.00
Hands Only	1	Not understandable ⟷ Understandable	5.41	5.00	1.32	1.75	5.00	7.00	2.00	7.00
	2	Difficult to learn ⟷ Easy to learn	5.14	5.00	1.48	2.19	4.00	6.00	2.00	7.00
	3	Complicated ⟷ Easy	4.66	5.00	1.34	1.81	4.00	5.00	1.00	7.00
	4	Unpredictable ⟷ Predictable	4.48	4.00	1.62	2.62	4.00	6.00	1.00	7.00
	5	Does not meet expectations ⟷ Meets expectations	4.62	5.00	1.37	1.89	4.00	5.00	2.00	7.00

Continued on next page

Table C.1 — continued from previous page

Mode	Q #	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
	6	Impractical ⟷ Practical	4.21	4.00	1.40	1.96	3.00	5.00	2.00	7.00
	7	Inefficient ⟷ Efficient	3.97	4.00	1.64	2.68	3.00	5.00	1.00	7.00
	8	Confusing ⟷ Clear	4.76	5.00	1.75	3.05	4.00	6.00	1.00	7.00
	9	Annoying ⟷ Enjoyable	4.17	4.00	1.58	2.50	4.00	5.00	1.00	7.00
	10	Unpleasant ⟷ Pleasant	4.48	4.00	1.30	1.69	4.00	5.00	1.00	7.00
Hands Only → UEQ Category Scores										
Hands Only	Perspicuity	(Q 1, 2, 3, 8)	4.99	5.00	1.26	1.59	4.25	5.75	1.75	6.75
	Dependability	(Q 4, 5)	4.55	4.50	1.38	1.92	3.50	5.00	1.50	7.00
	Efficiency	(Q 6, 7)	4.09	4.00	1.44	2.07	3.50	5.00	1.50	7.00
	Attractiveness	(Q 9, 10)	4.33	4.00	1.30	1.70	3.50	5.00	1.00	7.00
Hands + Gaze	1	Not understandable ⟷ Understandable	5.45	5.00	1.50	2.26	5.00	7.00	2.00	7.00
	2	Difficult to learn ⟷ Easy to learn	4.66	5.00	1.65	2.73	3.00	6.00	1.00	7.00
	3	Complicated ⟷ Easy	4.66	5.00	1.67	2.81	4.00	6.00	1.00	7.00
	4	Unpredictable ⟷ Predictable	4.48	5.00	1.81	3.26	3.00	6.00	1.00	7.00
	5	Does not meet expectations ⟷ Meets expectations	4.21	5.00	1.76	3.10	3.00	5.00	1.00	7.00

Continued on next page

Table C.1 — continued from previous page

Mode	Q #	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
	6	Impractical ⟷ Practical	3.79	4.00	1.74	3.03	2.00	5.00	1.00	6.00
	7	Inefficient ⟷ Efficient	3.83	4.00	1.81	3.29	2.00	5.00	1.00	7.00
	8	Confusing ⟷ Clear	4.07	4.00	1.58	2.50	3.00	5.00	1.00	7.00
	9	Annoying ⟷ Enjoyable	3.66	4.00	1.56	2.45	2.00	5.00	1.00	7.00
	10	Unpleasant ⟷ Pleasant	3.72	4.00	1.58	2.49	3.00	5.00	1.00	7.00
Hands + Gaze → UEQ Category Scores										
Hands + Gaze	Perspicuity	(Q 1, 2, 3, 8)	4.71	5.00	1.37	1.89	4.25	5.75	1.75	7.00
	Dependability	(Q 4, 5)	4.34	4.50	1.70	2.88	2.50	5.50	1.00	7.00
	Efficiency	(Q 6, 7)	3.81	4.00	1.69	2.85	2.50	5.00	1.00	6.50
	Attractiveness	(Q 9, 10)	3.69	4.00	1.51	2.28	2.50	4.50	1.00	7.00

Appendix D

D.1 PQ Statistics

Table D.1: Detailed PQ Question Statistics by Interaction Mode

Note: Each PQ item was asked separately for Controllers, Hands Only, and Hands + Gaze.

Mode	$Q\#$	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
Controllers	1	Not responsive ⟷ Completely responsive	5.93	6.00	1.28	1.64	5.00	7.00	3.00	7.00
	2	Extremely artificial ⟷ Completely natural	5.24	6.00	1.60	2.55	4.00	6.00	2.00	7.00
	3	Not consistent ⟷ Very consistent	4.31	4.00	1.63	2.65	3.00	6.00	2.00	7.00
	4	Extremely artificial ⟷ Completely natural	4.62	5.00	1.66	2.74	3.00	6.00	2.00	7.00
	5	Not compelling ⟷ Very compelling	5.34	6.00	1.45	2.09	5.00	6.00	1.00	7.00
	6	Not at all ⟷ Extensively	5.93	6.00	1.19	1.42	6.00	7.00	3.00	7.00

Continued on next page

Table D.1 — continued from previous page

Mode	$Q\#$	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
	7	Not involved ⟷ Completely engrossed	5.66	6.00	1.01	1.02	5.00	6.00	3.00	7.00
	8	Not proficient ⟷ Very proficient	6.03	6.00	1.09	1.18	5.00	7.00	3.00	7.00
	9	Not at all ⟷ Completely	6.59	7.00	0.63	0.39	6.00	7.00	5.00	7.00
	10	Not at all ⟷ Completely	6.41	7.00	0.82	0.68	6.00	7.00	4.00	7.00
	11	Not at all ⟷ Completely	2.90	3.00	1.80	3.24	1.00	4.00	1.00	7.00
Controllers → PQ Factor Scores										
Controllers	Involvement	(Q 1 . . . 7)	5.29	5.57	1.02	1.04	4.57	5.86	3.14	6.71
	Adapt./Immer.	(Q 8 . . . 10)	6.34	6.67	0.72	0.52	5.67	7.00	4.33	7.00
	Interface Quality	(Q 11)	2.90	3.00	1.80	3.24	1.00	4.00	1.00	7.00
Hands Only	1	Not responsive ⟷ Completely responsive	4.48	5.00	1.27	1.62	3.00	5.00	2.00	7.00
	2	Extremely artificial ⟷ Completely natural	4.69	5.00	1.34	1.79	4.00	6.00	2.00	7.00
	3	Not consistent ⟷ Very consistent	4.28	4.00	1.46	2.14	3.00	5.00	1.00	7.00
	4	Extremely artificial ⟷ Completely natural	3.72	4.00	1.46	2.14	3.00	5.00	1.00	7.00
	5	Not compelling ⟷ Very compelling	3.79	4.00	1.82	3.31	2.00	5.00	1.00	7.00
	6	Not at all ⟷ Extensively	4.62	5.00	1.27	1.60	4.00	6.00	2.00	7.00
	7	Not involved ⟷ Completely engrossed	4.72	5.00	1.60	2.56	4.00	6.00	1.00	7.00
	8	Not proficient ⟷ Very proficient	4.45	5.00	1.48	2.18	3.00	6.00	2.00	7.00

Continued on next page

Table D.1 — continued from previous page

Mode	$Q\#$	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
	9	Not at all ⟷ Completely	4.69	5.00	1.65	2.72	3.00	6.00	1.00	7.00
	10	Not at all ⟷ Completely	4.55	5.00	1.62	2.61	3.00	6.00	2.00	7.00
	11	Not at all ⟷ Completely	4.10	4.00	1.54	2.38	3.00	5.00	1.00	7.00
Hands Only → PQ Factor Scores										
Hands Only	Involvement	(Q 1 . . . 7)	4.33	4.43	1.11	1.23	3.86	4.86	2.00	7.00
	Adapt./Immer.	(Q 8 . . . 10)	4.56	5.00	1.41	1.99	3.33	5.67	2.00	7.00
	Interface Quality	(Q 11)	4.10	4.00	1.54	2.38	3.00	5.00	1.00	7.00
Hands + Gaze	1	Not responsive ⟷ Completely responsive	4.34	5.00	1.34	1.81	4.00	5.00	2.00	7.00
	2	Extremely artificial ⟷ Completely natural	3.86	4.00	1.88	3.55	2.00	5.00	1.00	7.00
	3	Not consistent ⟷ Very consistent	3.76	4.00	1.57	2.48	2.00	5.00	1.00	7.00
	4	Extremely artificial ⟷ Completely natural	3.86	4.00	1.48	2.19	3.00	5.00	1.00	7.00
	5	Not compelling ⟷ Very compelling	4.14	4.00	1.71	2.91	3.00	5.00	1.00	7.00
	6	Not at all ⟷ Extensively	4.07	4.00	1.46	2.14	3.00	5.00	2.00	7.00
	7	Not involved ⟷ Completely engrossed	4.41	5.00	1.50	2.25	4.00	5.00	1.00	7.00
	8	Not proficient ⟷ Very proficient	4.62	5.00	1.37	1.89	4.00	6.00	2.00	7.00
	9	Not at all ⟷ Completely	4.34	4.00	1.63	2.66	3.00	5.00	1.00	7.00
	10	Not at all ⟷ Completely	4.41	5.00	1.35	1.82	3.00	5.00	2.00	7.00
	11	Not at all ⟷ Completely	4.03	4.00	1.57	2.46	3.00	5.00	1.00	7.00

Continued on next page

Table D.1 — continued from previous page

Mode	$Q\#$	Adjective Pair	Mean	Median	Std	Variance	Q_1	Q_3	Min	Max
Hands + Gaze → PQ Factor Scores										
Hands + Gaze	Involvement	(Q 1 … 7)	4.06	4.00	1.27	1.60	3.29	4.86	1.29	6.43
	Adapt./Immer.	(Q 8 … 10)	4.46	4.67	1.37	1.87	3.33	5.33	2.00	7.00
	Interface Quality	(Q 11)	4.03	4.00	1.57	2.46	3.00	5.00	1.00	7.00