# Exploring Feasibility Of Common Virtual Reality Training Scenario Interactions Utilising Oculus Quest's Hand-Tracking

Master's Thesis

Group 201033

Aalborg University
Department of Architecture, Design and Media Technology

AALBORG UNIVERSITY
STUDENT REPORT

**Department of Architecture, Design and Media Technology**
Aalborg University
http://www.aau.dk

**Title:**
Exploring Feasibility Of Common Virtual Reality Training Scenario Interactions Utilising Oculus Quest's Hand-Tracking

**Theme:**
MED10 Master's Thesis

**Project Period:**
Spring Semester 2020

**Project Group:**
201033

**Participant(s):**
Christoffer Sand Kirk
Hassan Hameed
Simon Fjordvald

**Supervisor(s):**
Martin Kraus

**Copies:** 1

**Page Numbers:** 109

**Date of Completion:**
May 28, 2020

**Abstract:**

This paper seeks to explore the potential and limitations of camera-based hand-tracking in virtual reality on the Oculus Quest. Through three individual studies regarding *pointing behaviour*, *hand- vs. controller-tracking interactions comparison*, and *hand-tracking stability*, insights has been gathered on a broad scale to base the presumptions made. The general conclusion throughout this paper is that hand-tracking has too many flaws in its current state. It is too difficult to use, resulting in a slower and more error-prone task completion. Despite worse performance and tracking flaws, hand-tracking is found suitable for directly pushing virtual buttons which imitates exactly how the interaction would happen in real life. Secondly, the default hand-tracking pointing implementation is found to be a suitable alternative as it opens for interaction with both hands equivalently to controllers, however, it is identified as a slower input method that feels unnatural to use.

**AALBORG UNIVERSITET**
STUDENTERRAPPORT

**Department of Architecture, Design and Media Technology**
Aalborg Universitet
http://www.aau.dk

**Titel:**
Exploring Feasibility Of Common Virtual Reality Training Scenario Interactions Utilising Oculus Quest's Hand-Tracking

**Tema:**
MED10 Kandidatspeciale

**Projektperiode:**
Forårssemesteret 2020

**Projektgruppe:**
201033

**Deltager(e):**
Christoffer Sand Kirk
Hassan Hameed
Simon Fjordvald

**Vejleder(e):**
Martin Kraus

**Oplagstal:** 1

**Sidetal:** 109

**Afleveringsdato:**
28. maj 2020

**Abstract:**

This paper seeks to explore the potential and limitations of camera-based hand-tracking in virtual reality on the Oculus Quest. Through three individual studies regarding *pointing behaviour*, *hand-tracking vs. controller tracking comparison*, and *hand-tracking stability*, insights has been gathered on a broad scale to base the presumptions made. The general conclusion throughout this paper is that hand-tracking has too many flaws in its current state. It is too difficult to use, resulting in a slower and more error-prone task completion. Despite worse performance and tracking flaws, hand-tracking is found suitable for directly pushing virtual buttons which imitates exactly how the interaction would happen in real life. Secondly, the default hand-tracking pointing implementation is found to be a suitable alternative, as it opens for interaction with both hands equivalently to controllers, however, it is identified as a slower input method that feels unnatural to use.

# Contents

# Chapter 1

# Introduction

The first examples of hand-tracking can be dated back to when the Sayre Glove was developed in 1977 [27]. The glove used bendable tubes along with each finger and each tube would have a light source at the end that would decrease the amount of light passing through when the tubes were bent, thus allowing detection of the amount of finger bending. The Sayre Glove is known as an active data glove, contrary to a passive data glove, that solely detects fingers based on a camera and a sensor-less glove with colour markers. It was not until the early 1980s that camera-based tracking was developed for hand-tracking, although the technique required a sensor-free glove with colour markers for the camera to properly detect each finger [27]. Since then, there has been a continual increase in the level of image recognition processes, in terms of recognition accuracy which has led to the use of an optical image recognition system. Hand-tracking shows a lot of promise and potential for frictionless Virtual Reality (VR) experiences which could benefit large industries in implementing improved VR training simulations. Another example is the rehabilitation of stroke patients, who are unable to hold physical controllers in their hands [15].

One of the most recent methods for hand-tracking was introduced to the Oculus Quest back in December 2019 [13], which welcomed a new form of interaction in VR that solely depended on tracking the user's hands, thus removing the need for controllers [32]. The Quest utilises four wide-angle cameras mounted on the front of the head-mounted device to capture, recognise and track the user's hands [8].

The company Unity Studios has functioned as a collaborative partner throughout this project. Unity Studios works with developing VR applications, amongst other XR technologies, and were interested in exploring the potential of hand-tracking technology and if it could be beneficially implemented into their applications, specifically focused on the VR training scenarios. The collaborative work has helped guide this paper with a professional context and create reasoning for choices throughout the paper.

This paper was initially planned to explore how to efficiently utilise hand-tracking as a substitute for controllers and how using hands as an input device compares to using controllers. During the exploration of this, there was a shift in focus towards tracking-stability, investigating the influence of tracking technology. This happened as a result of the collected data from study of this paper, which was used to guide the evolution and focus towards the third study on tracking stability.

The general approach of this paper is to develop prototypes for evaluations to obtain quantitative data for statistical analysis. Qualitative data will also be collected on a minor scale to function as an explanatory tool to account for tendencies in the data.

# Chapter 2

# Background Research

This chapter focuses on relevant background research related to the project, including sections on ergonomics, ray cast pointing followed by natural pointing behaviour. Finally, general tendencies of previous studies investigating hand-tracking compared to controller-tracking interactions are synthesised.

## 2.1 Ergonomics and Interaction Space

The following section contains an analysis of ergonomic considerations in regards to hand-based interactions, as the design of interactable elements in the virtual scene has direct implications on the muscular strain. With controllers, the user can manipulate the controller to perform the desired interaction, such as pointing in different directions, without physically translating the hand a large distance as is the case for pointing at objects with fingers. Furthermore, hand-based interaction affords touch input, where the physical arm movement towards touching a virtual element is dependent on the placement of the objects in the virtual world. Pointing is another aspect with ergonomic implications, as people tend to carry over their natural pointing behaviour with a stretched arm [14] towards the intended object [5], which imposes a muscular strain on participants [10, 24]. Extended periods of arms reaching into strenuous positions can lead to the "gorilla arm" effect as it is termed, meaning the user experiences fatigue or their arm starts to hurt [10]. Ray cast pointing in VR is often a continuous operation, rather than a singular instance for guiding attention towards an object [14, 18], which can quickly leave users fatigued. Therefore, ergonomic considerations in the design process are important to consider.

### 2.1.1 Direct Object Interaction

When it comes to direct object interaction without ray cast assisted pointing at virtual objects in VR environments, vertical and horizontal placement of the virtual objects are both influential on the required arm movements. In terms of vertical placement of virtual objects in the scene, the most frequently interacted objects should be placed between 0°-(-15°) degrees to avoid musculoskeletal strain [24], where 0° denotes the horizontal level from a user's eyes. Similarly, participants in the study reported that they would place objects between (-0.5°)-(-6.5°) degrees, meaning objects are desired to be placed just below the horizontal level for interaction. When it comes to object placement, Hincapie-Ramos et al. recommends designing placements that support interactions with a center+bent+inwards arm placement [10], as seen in the last panel of figure (2.1). The preceding three-panel pictures illustrate the effects of target placement on arm and hand placement.

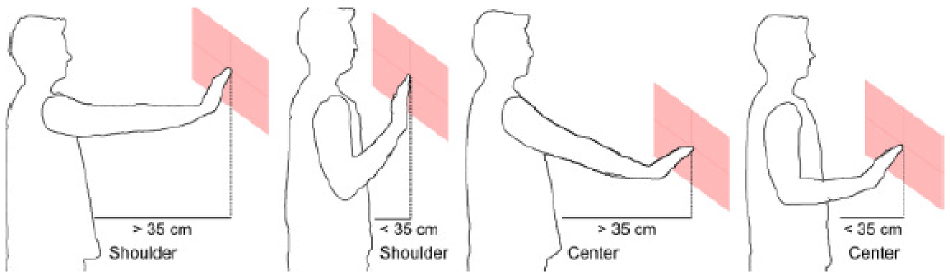


**Figure 2.1:** Target location effects on physical arm placement, with the last panel illustrating recommended target location for supporting ergonomic interactions [10].

While distance to object would depend on each user's arm length, their recommendation states placement <35cm from the user's abdomen. Oculus provide similar recommendations of virtual scene design, as presented in figure (2.2) [8], where they define primary, secondary and tertiary zones of interaction.

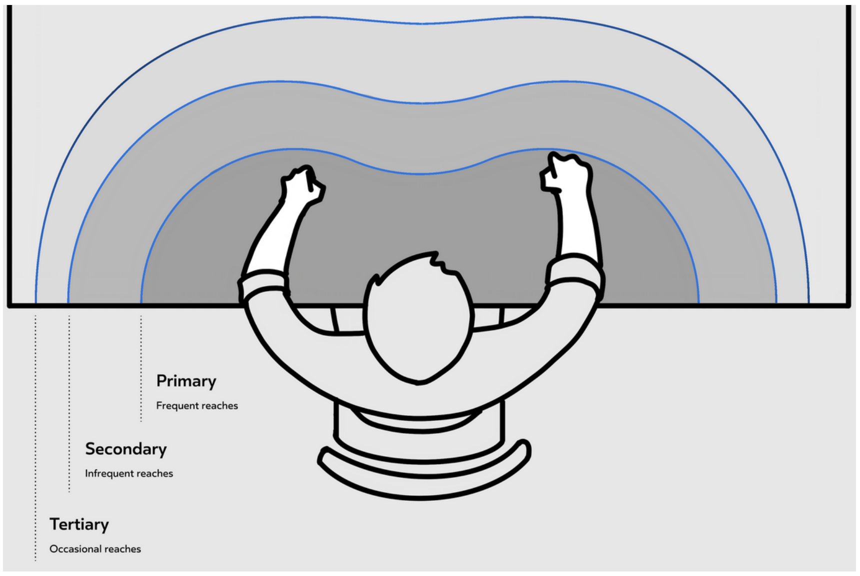


**Figure 2.2:** Primary, secondary and tertiary interaction zones as recommended by Oculus [8].

The primary zone supports an ergonomic position for prolonged or frequent interactions, whereas the secondary and tertiary interactions ought to be reserved for shorter sporadic interactions. Hincapie-Ramos et al. took their study a little further by investigating musculoskeletal strain in a UI setup, finding interactions occurring in the lower-left corner of a 25cm x 25cm grid are less fatigue-inducing than interactions in the upper right corner as illustrated in figure (2.3) [10].

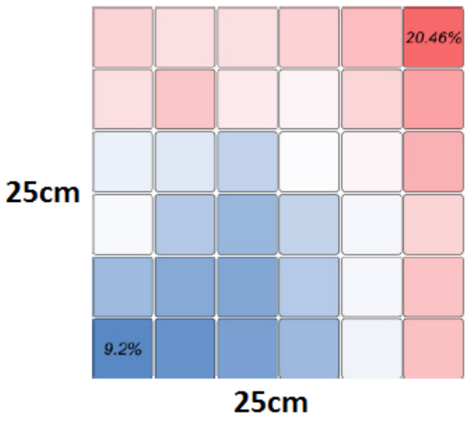


**Figure 2.3:** Consumed Endurance heatmap [10] (25cm x 25cm grid) of target location (imagine a UI placed in front of the user in VR at the center+bent+inwards hand placement illustrated in figure 2.1) where blue indicates least strenuous location and red indicates the most strenuous location to reach.

## 2.2 Ray Casting with Oculus Quest's Hand-Tracking

Ray cast pointing is an action that can occur continuously over the course of engagement with the virtual environment and therefore serves as an interaction with the requirements of living up to the ergonomic recommendations. The default pointing method implemented by Oculus for the Oculus Quest hand-tracking feature, is implemented in a manner that supports the ergonomic recommendations outlined in section (2.1), but this implementation does not come without trade-offs.

The pointing implementation works by anchoring the ray cast to two points on the user's body, namely the controlling anchor on the knuckle and the stabilising anchor on the spine as illustrated in figure (2.4) [8].

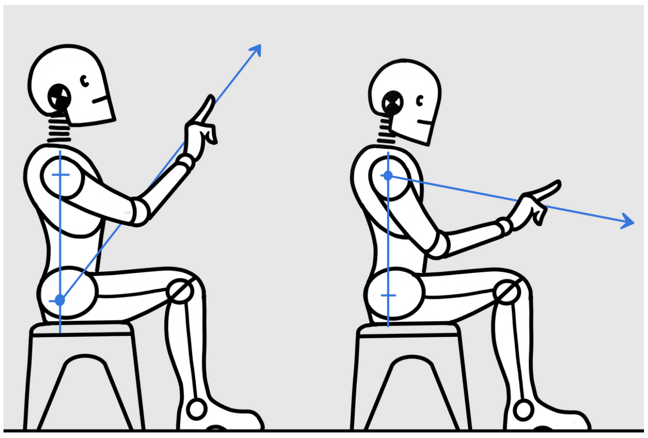


**Figure 2.4:** Oculus default ray cast pointing implementation for hand-tracking, where stabilising anchor point on the spine dynamically moves up and down, based on vertical gaze angle [8].

The stabilisation anchor point is likely obtained through estimation from the user's height, which can be extracted from head-tracking data by calculating the difference between the guardian floor and the y-axis position of the headset. The head-gaze angle of the user controls stabilisation point position on the spine, meaning if head-gaze is directed upwards from the horizontal level, the stabilising anchor point dynamically moves towards the hip for instance [8]. By dynamically shifting the stabilising anchor point, an ergonomic arm position can be maintained throughout use with the application. Supporting an ergonomic pointing position is particularly important considering some users tend to extend their natural pointing behaviour into the VR world at first [5], meaning they extend their arms into stretched out positions for extended periods of time that they would not do in the real world, quickly leaving the user fatigued.

The implementation does not, however, adhere to natural pointing behaviour.

When it comes to pointing out a singular specific object in a context, pointing distinctly takes the form of index finger pointer with hand palm down [14], as illustrated in figure (2.5).

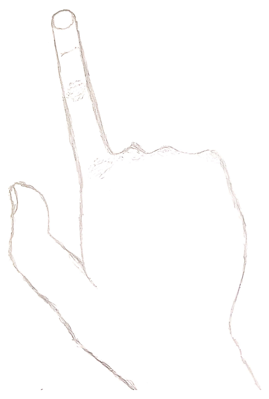


**Figure 2.5:** Index finger pointing with hand palm down, distinctly used by people to individuate a specific object in a context.

An implementation of this natural pointing method is vulnerable to natural finger tremors, which gets amplified over distance due to the permanently visible ray cast [8]. Stabilisation of the ray is achievable by anchoring the stabilisation anchor at the knuckle and anchoring the controlling anchor to the index fingertip [25], as seen in figure (2.6).

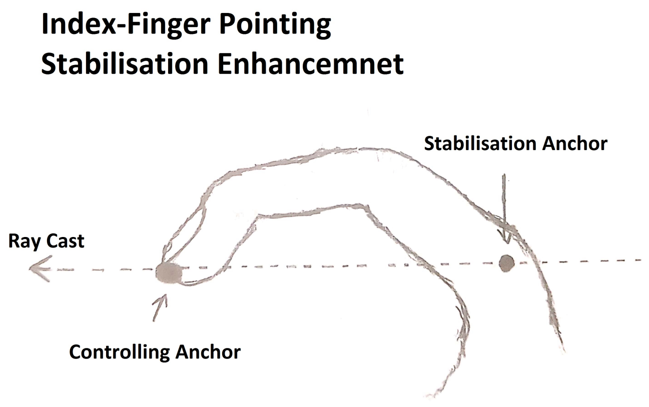


**Figure 2.6:** Method for stabilising ray cast implementation adhering to natural index finger pointing.

However, pinch selection accuracy would be affected by finger movement, which highlights that the Oculus pointing method opens for selections with the pointing hand, similarly to what is possible with a mouse or controller, as the knuckle controlling anchor point is unaffected by finger movements occurring during selection.

## 2.3 Natural Pointing

Early internal tests of pointing implementations made it immediately clear to the research team that pointing with the Oculus implementation did not feel natural to use, however, natural tremors were also a distracting factor with an implementation that adhered to the index finger pointing method that is more natural. Juggling trade offs between pointing implementations begged the question of what exactly is meant by natural pointing behaviour, which the following section examines.

Speech, physical arm movement and cognitive inference of intended target all mix together in the process of natural pointing [18]. Typically, pointing occurs when more than one person is present, where a speaker is intending to guide a recipient's attention towards an object, location or person of interest [14]. Pointing can take different shapes depending on context of a conversation, but for this project, we are only interested in the process of individuation of a singular object in a context, as is a common task in VR experiences, which distinctly takes the shape of index finger pointer with hand palm down as illustrated in figure (2.5), presented in the previous section. In a solo VR experience, speech does not typically accompany the process of pointing, which is commonly employed through ray casting, meaning visualisation of a virtual "laser" ray shooting out from a pre-defined origin point on the user's hand. While speech recognition systems are being investigated [9, 6, 12], they are not usually found in today's mainstream VR applications beyond a select few select examples. But what does this missing piece of natural pointing in VR mean for users?

Answering that question requires splitting the pointing process up into its individual parts, namely the physical act of pointing and the speech we use to guide attention towards an object. In an identification game experiment by Lücking et al., they find that the physical act of pointing cannot be considered a direct reference towards a singular object, but should instead be thought of as a cone (presented in figure (2.7)) we shoot in a general direction to establish context [18].

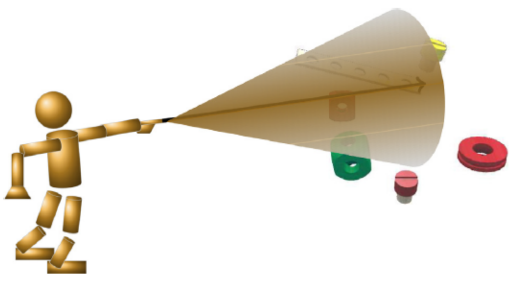
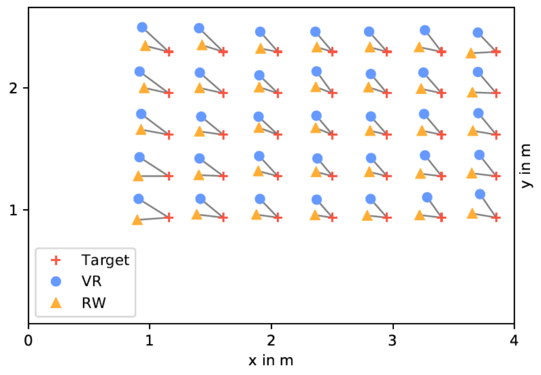


**Figure 2.7:** Pointing considered as a cone for establishing context, which gets increasingly difficult to disambiguate over distance [18].

In their experiment, one participant acts as the description giver, meaning the person pointing towards objects (24 in 8x4 grid) on a table (140cm x 70cm), whereas the object identifier attempts to identify the specific object pointed at. Without speech allowed from the participants (22 identification games) over 1408 trials of data in a between-subjects design experiment, identification error rate starts to rise substantially at >0.5m, from 5% error rate at 0.68m to 42.5% at 1.22m. In the second group with speech allowed, identification error rate remains at 0.02% across all distances, but the difference shows itself in the number of words used to identify the object. At 0.5m, participants (description giver) used 3 words to describe the object, which increases to 6 words per object at >1m. Not only can physical pointing not be considered a direct reference towards a specific object, complexity of speech increases in tandem with the fuzziness of pointing accuracy [18]. Similar findings on pointing inaccuracy were documented by Mayer et al., who found that with no visual guidance, such as ray casting, participants will systematically overshoot their pointing, resulting in 10-60cm offset from an intended target point (2m distance between participant and target), dependent on origin of the ray on the participants arm [21, 25]. These offsets matched those found by Lücking, as shown in figures (2.8) and (2.9), which supports the idea that the physical act of pointing is not an accurate process without speech as a guiding mechanism.



(b) Index finger ray cast (IFRC)

**Figure 2.8:** Systematic offset error in pointing (27cm-29cm). Ray cast origin from the right hand's index fingertip, without visual feedback of the pointing ray cast [21]. Intended target is marked with a red cross, VR data visualised in blue and real world data visualised in light orange.

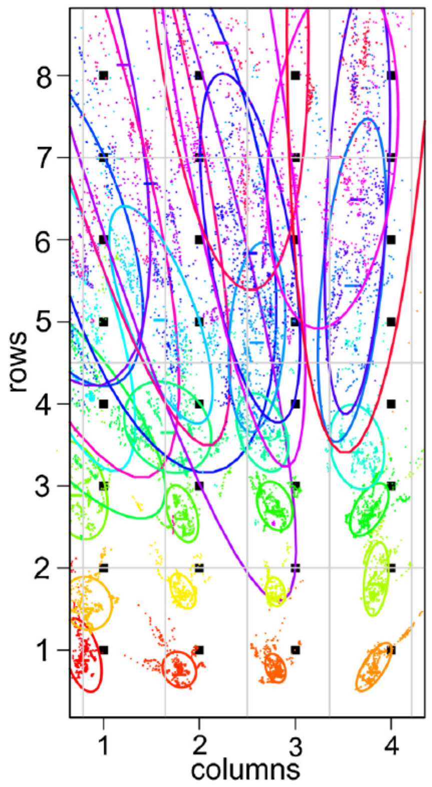


**Figure 2.9:** Systematic offset errors in the pointing task from the Description Giver's perspective. Table is sized 140cm x 70cm with objects placed in a 8x4 grid. At distal areas (>0.5m, row 3+), precision of pointing ray falls off dramatically and turns into a cone. Center of drawn circles marks the average offset error of where the pointing ray cast would land on the table if visualised. Black boxes marks the center of the real world object being pointed at [18].

It is, however, important to note that visual feedback in pointing, as is employed by ray casting, nearly eliminates precision issues from 7cm offset to 1.1cm offset [21], and therefore can be considered a simple guiding mechanism in VR. It remains unknown how this mechanism changes users' perception of pointing accuracy.

Simple linear ray cast works well for simple VR scenes with scattered objects and invariant depth placement. However, more complex scenes with occluding objects or densely packed objects complicate matters with ambiguous situations, which is one reason for research to be focused on enhancing the shape and behaviour of the ray cast. Objective measures such as *completion time* and *error rate* are at the core of such enhancement method experiments, as the benchmark to beat is considered traditional linear ray casting [22, 29, 3]. Bendcast is one such enhancement method, where the ray cast vector determines which target is closest based on distance [17]. This can be determined either at the end of the vector or at any given point along the ray cast vector. By bending the ray cast into a circular arc, it locks onto the closest target as visualised in figure (2.10).

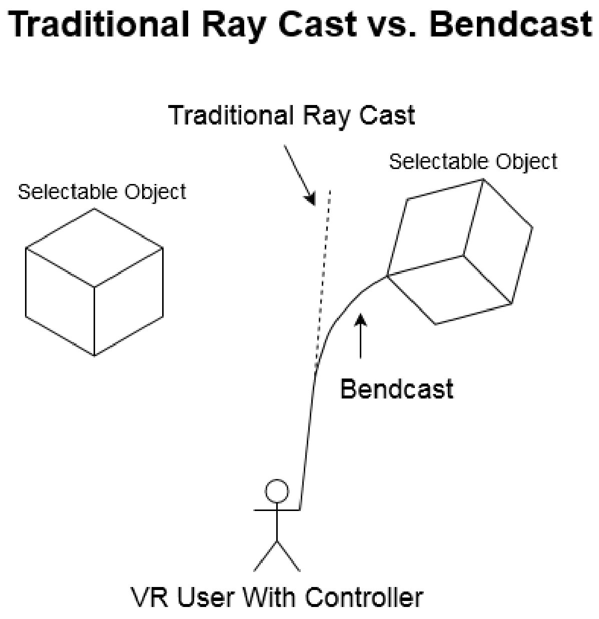


**Figure 2.10:** Traditional ray cast visualisation and behaviour compared to Bendcast. Traditonal ray cast is linear, Bendcast locks to the nearest object.

In collaborative tasks, or re-targeting, the movement of the circular arced ray is valuable in providing immediate feedback, either for inferring intention of movement from a collaborative partner [29], or how close one is to re-target another object.

Bendcast does not lend itself well to complex scenes with clustered objects [3, 17], due to smaller, if any, areas of free movement between targets. The snapping threshold essentially makes the target bigger than its visualisation, which affects user behaviour in the pointing process. Larger targets require less precision, which means users do not require fine-tuning their selection to be precise, which is a slower process [22]. Selection time improves, but the relaxed state of mind consequently produces more errors as the user does not take specialised care in avoiding errors [3, 22], which can be considered more critical due to selection of an unintended object. In a non-clustered environment, a Fitt's law task with targets located at 5m, 10m and 15m, selection time for Bendcast (named "Snap-To" here) was significantly faster than traditional ray cast as shown in table (2.1), summarising data of the non-clustered condition tested by Moore et al. [22].

**Table 2.1:** Selection time per object in a non-clustered environment, compared between traditional linear ray casting and Bendcast/Snap-To method, showing significant selection time advantages for the Snap-To method, especially at further distances (>10m) where selection time was 2-2.5 times faster [22].

| Distance | Time(s)/Object Traditional | Time(s)/Object Snap-to |
|---|---|---|
| 5m | 1.136s | 0.712s |
| 10m | 1.476s | 0.788s |
| 15m | 2.030s | 0.835s |

However, as stated before, the trade-off is a higher error rate for Bendcast/Snap-To (Traditional: 0.80% vs. 3.80% Snap-To), meaning 4.5 times the amount of errors occur during all distance conditions when the Snap-To method is used. In a similar study, Cashion et al. compared Bendcast against Expand (two-step process of selecting an area with a cone-shaped pointer into an expanded view, followed by a precise selection of an intended object in the expanded view) and their auto-selection algorithm, which chooses the most suitable selection method on a frame-to-frame basis, based on the density of objects. Selection time per object is significantly faster, but more error-prone as shown in the table (2.2) [3].

**Table 2.2:** Selection time per object and Error rate %, compared between Bendcast, Expand and the researchers' developed Auto-Selection algorithm [3].

| Method | Time(s)/Object | Error (%) |
|---|---|---|
| Bendcast | 1.44s | 0.42% |
| Expand | 2.86s | 0.08% |
| Auto-Select | 2.38s | 0.17% |

Even though these studies are based on different contexts (Cashion used PS3 controller and 3D environment, Moore used Wii-Remote controller in a VR environment), similar findings can be found. To our knowledge, similar experiments have not been conducted with bare hand pointing, which we know from the literature is both a slower and more error-prone input method compared to using controllers, as described in section (2.4), although it is focused more on direct object interaction.

## 2.4 Controller-Based Interaction vs. Hand-Based Interaction

Changing the input method for a technological system immediately begs the question of why one would use the newer method (hand-tracking) over the older one (controller-tracking). This section is primarily focused on direct object interaction but may contain examples also relevant to ray cast pointing. Hand-tracking has the advantage of providing a more frictionless VR experience compared to controller tracking, as no peripheral device is required for engaging with the digital environment [2], since the tracked hands acts as the controllers for interaction. Secondly, hand tracking ought to be easier to learn and use as design can build on natural human behaviour [35]. A notion that is validated by users approaching grasping real and virtual objects equivalently [4], indicating interaction intuitiveness afforded by hand-tracking. Finger-tracking, which hand-tracking provides, brings the promise of fine motor control interaction possibilities, because of more degrees of freedom (each joint in hand-tracking adds to degrees of freedom, which is usually controlled by trigger/button activated animations with controllers). Possibilities one could imagine would aid in matching a VR training scenario to the real-world task, as is a common use of VR technology in business contexts.

Hygiene challenges in medical care present an option where mid-air interactions could provide a direct benefit. For instance, a standardised procedure in surgery involves one doctor examining pictures on a computer in a separate room, who then relays instructions to the doctor performing the surgery [16]. Kim Yonjae et al. describes that contactless hand tracking for surgical robot control is an attractive alternative because it can be executed with a minimal footprint at the patient's bedside without impairing sterility while eliminating current disassociation between the surgeon and patient [16]. By enabling mid-air interactions with digital displays, relay of information could be eliminated, which can be prone to miscommunication or lacking important details. Rehabilitation of stroke patients present another option for hand-tracking to provide value, as some of these patients may not have the physical strength or motor control, to use a handheld controller [15].

The potential for hand-tracking is there, however, the lack of haptic feedback in grasping a virtual object presents a fundamental challenge towards the naturalness of the VR experience. While mid-air haptics is a research area being explored through ultrasonic sensors [20], it is currently necessary to augment the system with a third party device, which would increase the friction of use similarly to controllers. For this reason, we do not explore this research area further, as it would eliminate one of the primary purposes for hand-tracking. At a fundamental level, the lack of haptic and force feedback in the process of picking up an object goes against natural human behaviour. Picking an object out of thin air exposes the complex coordination of mental and physical processes taking place in what may seem like a simple task in the real world. While the approach to grasping an object is equivalent between the real world and virtual, differentiation between the two environments shows itself in the contact points on the surface of the virtual object. Without a physical medium to naturally stop movement into the object, users in virtual overshoots their finger placements [4]. Five participants averaged 1.05cm (SD: 0.53cm) between the index finger and the thumb, comparatively to the 2.5cm ideal distance matching the virtual object's visual width. A shortcoming showing that the lack of haptic feedback makes people uncertain of how much force to apply in their grasping behaviour. One could also imagine that without haptic feedback, interacting with an object outside peripheral vision would be next to impossible, as there would be no confirmation that the object is touched.

The pinch gesture visualised in figure (2.11) may not be considered a natural gesture in the sense that humans do not use the gesture to select something in the real world, but it provides haptic feedback that would be lacking when attempting to pick up a virtual object in a natural manner.

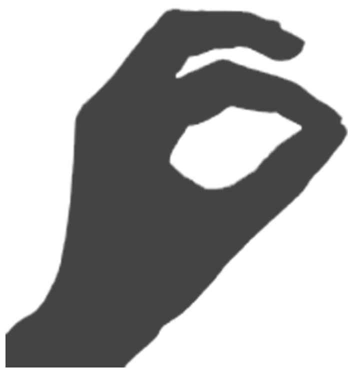


**Figure 2.11:** Pinching gesture used to select and object in VR environments.

For this reason, the pinching gesture is recommended by Oculus as the method of object selection [8], both for direct object interactions and ray cast pointing selections. Secondly, since it is not a naturally occurring gesture, it can be recognised as a distinct interaction by the tracking system. Designing distinct interactions can be a challenge in gesture recognition systems, but is important for avoiding recognition of unintended behaviour that can lead to interaction errors [35].

Technological constraints in state-of-the-art within hand-tracking presents more limitations that can hinder the applicability of hand-tracking. The leap-motion sensor used in research studies previously [2, 7, 5, 23] is based on infrared imaging sensors combined with artificial intelligence software predictions of finger joint location [26]. Oculus Quest's hand-tracking feature is based on two frontal camera recordings augmented by computer vision software prediction of hand location [8]. Commonly for both systems, depth understanding is not provided through hardware, but rather a software predictions which can be faulty if assumptions are violated. Occlusion for instance, as illustrated in figure (2.12), is an issue that is difficult for either system to handle which consequently limits the design space to two separate hands with limited ability to work together.

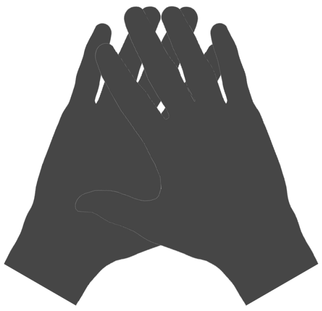


**Figure 2.12:** One hand occluding the other, where the optical tracking system cannot disambiguate between the two.

Hand position is also limited to positions where all fingertips are visible to the cameras, as the system can otherwise lose tracking due to missing crucial information in prediction. As shown in figure (2.13), when pinching with the hand pointing forwards, the two middle fingers would potentially be occluded from the camera view, which can introduce tracking losses.

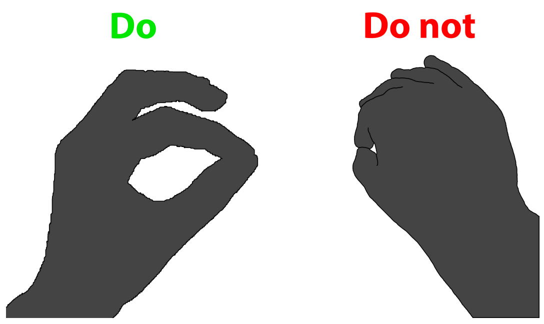


**Figure 2.13:** Acceptable pinching state with all fingers visible to the cameras, compared with a non-acceptable pinching state occluding the two middle fingers from cameras.

Depth sensing cameras that are starting to appear on the market, both as standalone and in smartphones [33], may potentially advance tracking abilities, however, implementing this emergent technology requires a hardware upgrade.

With the stated advantages and disadvantages in mind, the second element of comparison involves performance (time completion and error rate) and user experience differences. Since hand-tracking on the Oculus Quest is only four months old at the time of research for this project, we turn to literature using custom-made hand-tracking implementations and studies utilising leap motion. The following sections focus on tasks requiring direct object interaction.

### 2.4.1 Task Time Completion

Previous studies investigating task time completion between hand-tracking and controller-based interactions typically recruited between 8-30 participants, with an overall average between 27-33.62 [2, 7, 5, 23]. Pre-defined interaction tasks in VR (re-position objects, grab and release, direct object interaction) has been a common experimental approach [2, 7, 5], however, Olbrich et al. conducted their comparative study in a VR training scenario for handling an emergency situation on a space station [23]. Looking at the data presented in table (2.3), the designed hand interactions result in slower completion time in nearly all tasks explored with a factor of 1.1-2.4 times the controller interaction benchmark.

**Table 2.3:** Overview of studies comparing task time completion between hand-tracking interactions and controller-tracking interactions, showing tendency for hand interactions as a slower input method.

| Study | Task | Hand Time (s) | Controller Time (s) |
|---|---|---|---|
| Caggianese 18 [2] | 1.) Re-position objects (5) (Horizontal plane) | 79s | 42s |
| | 2.) Re-position objects (4) (Vertical stacking) | 55s | 23s |
| | 3.) 3D Sequence matching (6) (Sel./Pos./+Rotation) | 113s | 50s |
| Gusai 17 [7] | 1.) Dock objects (12) (Shape match + re-position) | 161.1s | 68.5s |
| Figueiredo 18 [5] | 1.) Text input (Direct touch) | 2.19s/Input | 2.19s/Input |
| | 2.) 1D Slider (Direct touch) | 5.64s/Input | 4.45s/Input |
| | 3.) 2D slider (Direct touch) | 3.98s/Input | 4.09s/Input |
| | 4.) Grab and Release (Re-position objects) | 3.31s/Input | 3.28s/Input |
| Olbrich 18 [23] | 1.) Training Scenario (6 tasks) (Selection and Touch) | 155s | 140s |

For controller interactions, the user would select an object by pressing and holding a button on the controller, while hovering the virtual controller representation inside the virtual object to be selected [2]. The controller interaction design was also used by Gusai et al. [7], Olbrich et al. [23] and partially by Figuiredo (task 4) [5]. Hand-tracking interactions were more varied, where users would close their hands in a fist inside the virtual object to make selections [2], directly touch the virtual object [5, 23], or grabbing and releasing the object in a natural manner [7, 5]. The direct touch tasks (1-3) by Figuiredo et al. followed controller interaction design similar to the hand interaction, as only direct touch with the tip of a HTC Vive controller was necessary to interact with the virtual object [5].

Comparison is difficult due to differences that arise between interaction methods. For instance, the time discrepancy reported by Gusai et al. (161.1s vs. 68.5s) can be a result of natural grabbing being subject to the uncertainty of virtual hand placement to grab the object [4], unlike the controller interaction which is a binary state that ought to be easier to understand [7]. Their data also stands in contrast to the grab and release task (task 4) of Figuiredo et al. (3.31s/input vs. 3.28s/input) finding no significant differences in time completion [5], suggesting interaction design specifics can be a major factor on reported task time completion differences. It is uncertain which specific differences result in this deviation, but releasing grabbed virtual objects in a seamless manner has previously been reported as a difficult obstacle to overcome in implementation [11]. In the training scenario tested by Olbrich et al. only the overall completion time is reported [23], obscuring exactly which of the six interaction tasks contributed to slower completion time. Summarising the data, hand interactions appear to be slower than established controller interaction paradigms, though the differences may be influenced by interaction design decisions.

The influence of interaction design on results brings the question of which approach to take in experimental design, either mimicking interactions or differentiating interaction to device advantages. By mimicking interactions between hand-tracking and controller-tracking, the research team can obtain better comparative results but their applicability to the real world can fall if the designed interaction for one device would not be used. Conversely, accepting interaction differences between hands and controllers in design can lessen the comparative understanding obtained from results, but may give better applicability of results to the real world. Context of study is influential in this decision for the research team, for instance if the goal is to expand the objective understanding of interaction methods, mimicking interactions provides potential discovery of new interaction methods. If the goal is to understand interaction design for real-world applications, such as training scenarios or games, the team may adopt a design approach of accepting the differences.

### 2.4.2 Error Rate

When it comes to error rate comparison, the data reported in table (2.4) signifies hand interactions as more error-prone than controllers. Participants in the Gusai et al. study were tasked with re-positioning virtual objects into matching empty shapes, meaning they were "docking" the virtual objects. The score started at 12, with a deduction for each position error made by the participant, resulting in re-positioning by the facilitator. Without a definition of how many points are deducted per error, it is unclear how many errors occur exactly, but it is clear that controller interactions are near flawless whereas precision falls with hands [7]. The error rate reported by Speicher et al. signifies the number of edited characters in the participants' task of replicating a string, by directly touching virtual keyboard buttons [30]. Similarly in the text entry task (task 1) of Figuiredo et al., the error rates constitute the number of times the virtual delete button was pressed. As for task 2 and 3 with sliders, errors were counted for each time the slider was released outside the intended target indicated by the task. For task 2 the slider represented a volume picker of 0-100, whereas the 2D slider in task 3 mimicked a colour picker.

Errors in the grab and release task (task 4) were counted for each time the object was released and for each time the object was released over an unintended target [5].

**Table 2.4:** Overview of studies comparing error rates between hand-tracking interactions and controller-tracking interactions, indicating hand interactions as a less precise and more difficult input method.

| Study | Task | Hand Error rate | Controller Error rate |
|---|---|---|---|
| Gusai 17 [7] | 1.) Dock objects (12) (Shape match + re-position) (Start score = 12, deductible per interaction error) | 10.3/12 | 11.9/12 |
| Speicher 18 [30] | 1.) Text Entry (Direct touch) | 7.57% | 1.94% |
| Figueiredo 18 [5] | 1.) Text Entry (Direct touch) | 3.22% | 1.88% |
| | 2.) 1D slider (Direct touch) | 17.57% | 12.37% |
| | 3.) 2D slider (Direct touch) | 24.53% | 13.48% |
| | 4.) Grab and Release (Re-position objects) | 5.28% | 1.50% |

In summary, it appears to be more difficult to achieve the same level of precision as controller interactions as well as keeping an object grabbed in your hand, at least for natural grasping interaction as designed in the grab and release task.

### 2.4.3 User Experience

As the previous two sections exemplify, hand-tracking interactions are objectively slower and more error-prone than the controller interaction benchmark. A similar story follows in regards to the user experience, as exemplified in table (2.5), where hand interactions score equivalently or worse in all measurements.

**Table 2.5:** Overview of studies investigating various user experience measurements between hand-tracking and controller-tracking interactions.

| Study | UX Measure | Hand | Controller |
|---|---|---|---|
| Caggianese 19 [2] | Perceived difficulty (Very easy: 1 - 7: Very difficult) | 2.69 | 1.13 |
| Gusai 17 [7] | Preference (Range: 0-100% of respondents) | 13% | 73% |
| Olbrich 18 [23] | Satisfaction (With Time-Completion) (Range: 0-100% of respondents) | 32 % | 82% |
| Reski 19 [28] | NASA TLX (Mental Workload) (Range: 0-100, Lower = better) | 50/100 | 41/100 |
| Figueiredo 18 [5] | System Usability (SUS) (Range: 0-100, Higher = better) | 86.28 | 86.80 |
| | Fatigue (Range: 0-3.5, Lower = better) | 2.28 | 2.25 |
| Speicher 18 [30] | User Experience (UEQ) (Range: -3 - 3, Higher = better) | 0.55 | 0.56 |

Hand-tracking interactions are perceived as more difficult to perform (2.69 vs 1.13, 1-7 Likert scale) [2], are less preferable as an input method (13% vs. 73%) [7] and considered less satisfactory when it comes to task completion time (32% vs. 82%) [23], suggesting task-time completion could be an influential parameter in the user's experience. Using hand interactions also induces a higher experienced workload compared to controllers (50/100 vs. 41/100, as measured by the NASA TLX Questionnaire where lower score = lower workload experienced) [28]. The direct object interaction tasks explored by Figuiredo et al. were not found to be less useful or more fatigue inducing comparatively to controllers [5], neither were user experience scores found significantly different in a text entry task [30]. It should be noted that Speicher et al. tested direct object interaction with controllers in the form of inverting the handheld Vive-controller, simulating a stylus input device. By taking controller pointing into account, which can be considered a more frequent interaction method employed by applications, the user experience difference shows itself at 0.55 for hands versus 1.17 for controller pointing, on a range of -3 (very bad) to +3 (excellent) [30]. While controller pointing is not the focus of this section, this detail provides an idea that familiarity of interaction may play a factor in the user experience.

There is an example where qualitative data from informal interviews would suggest some users adopting hand-tracking interactions despite their flaws, at least for near field direct object interactions (see primary-tertiary zone figure (2.2))[5]. Near field direct object interactions also contribute to learning complex machinery assembly procedures [37], however, the study did not take a comparative approach, meaning they lacked a ground truth benchmark.

To summarise the data presented in table (2.5), there are no cases where hand-tracking seem to provide a better experience that would advocate for its adoption. Evident by various user experience measures in six different studies paints a picture of hand-tracking as a less useful input method compared to the established controller-based interaction mappings. At best a similar user experience can be expected [5, 30], but more often one would expect a worse experience [2, 7, 23, 28, 30].

### 2.4.4 Hands vs. Controllers Summary

Summarising the data in the previous three subsections, we arrive at the conclusion of hand-tracking interactions being slower, more error-prone, and a consequently worse user experience. Since a worse experience can be expected in most cases, with the best case scenario equivalent to controller-tracking experiences, it would be easy to conclude that hand-tracking interactions ought not be explored further. However, in most cases the experiments took the approach of an objective study on individual hand interactions, detached from real world scenarios [2, 7, 5, 30]. When it comes to VR training scenarios, Olbrich et al. did conduct their experiment in a complete training scenario, but they did not split the data analysis into its individual tasks [23], meaning there is a lacking understanding of which exact interaction task out of the six task steps (including re-positioning objects and direct object interactions) contributed to a less satisfactory time completion. A gap is therefore identified in the literature, which the second study of this paper will seek to answer, specifically combining the singular hand interaction focus with contextualised understanding of applicability to training scenarios.

# Chapter 3

# Problem Formulation

Based on the background research, it was decided to have an overall problem formulation revolving around the general usability of hand-tracking. Throughout the paper we want to explore hand-tracking in different situations, and learn more about what and where specifically hand-tracking proves usable. This created the problem formulation:

*Can Oculus Quest's hand-tracking in its current state successfully function as a suitable alternative to the controllers in any aspect of VR applications?*

For each of the studies throughout the paper we worked with minor questions to guide the evaluations. Inspiration for the first study revolves around Oculus Quest's default hand-tracking pointing implementation. The input method implementation opens for selection with the pointing hand, but it does not feel natural to use. Focus shifted towards enhancement of the pointing implementation, namely the Bendcast method [17], which has previously shown to be faster, but more error-prone [22, 3], though in contexts unrelated to hand-tracking. Previously hand-tracking has shown to be a slower input method compared to controllers [2, 7, 5, 23], which brings us to the research question stated as follows:

- *In the context of hand-tracking pointing, how do Bendcast behaviour and visualisation variables affect task completion time, error rate and preference?*

For the second study, inspiration is taken from the knowledge that hand-tracking is considered a slower and more error-prone input method [2, 7, 5, 23, 30]. However, the literature finds a lack of focus on real-world VR use cases, such as VR training scenarios. A study that investigated a VR training scenario context did not establish an understanding of which singular interaction performed by participants lead to slower completion time [23]. With a focus on singular interactions, the research question is stated as follows:

- *In the context of VR training scenarios, are there singular hand-tracking direct object interactions that perform better, or are found to be suitable alternatives to equivalent controller-tracking interactions?*

The third study was based on insights from the previous analysis, that prompted us to focus on the tracking stability of hand-tracking with the Oculus Quest. This introduced the following research questions:

- *In the current state of Oculus Quest's hand-tracking, how often is tracking lost during the interaction?*
- ***Sub-Question:*** *Are there interactions where hand-tracking is found as a preferable alternative to controller-tracking?*

# Chapter 4

# Study One: Linearity and Snapping Ray Cast Behaviour - Design & Implementation

The following chapter starts by establishing motivation for the study along with the research question it seeks to answer, followed by design decisions taken to accomplish that goal. The remaining sections present how the prototype was implemented, with in-application screenshots and code snippet details.

## 4.1 Motivation

In the early exploration of development possibilities with Oculus Quest's hand-tracking, the research team identified the default pointing method implementation as unnatural to use. As described earlier in section (2.2), the developers have traded natural perception for an increased number of interaction possibilities, considering the input method opens for selection with the same hand as one is pointing with. After researching and understanding the problems solved with the default pointing implementation, the focus shifted towards behaviour and appearance of the ray cast.

The hand-tracking demo released by Oculus in late December 2019, includes hand-tracking pointing in the Oculus Quest menus, taking the visual form seen in figure (4.1).

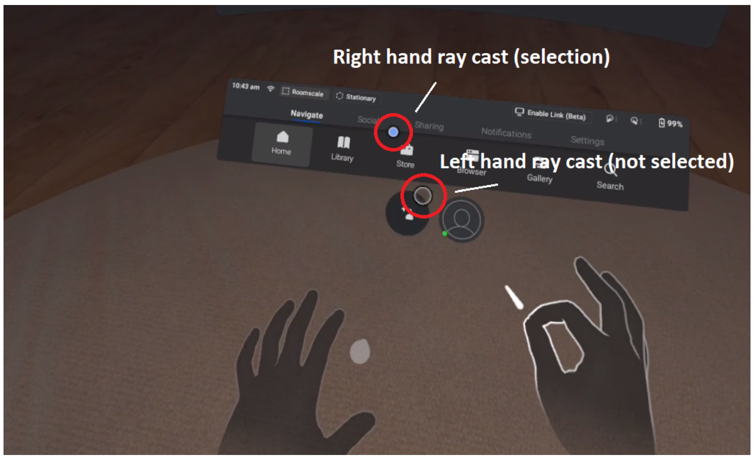

Right hand ray cast (selection)
Left hand ray cast (not selected)

**Figure 4.1:** Ray cast implementation for hand-tracking in the Oculus Quest menus, where ray cast endpoint is visualised as a dot.

Besides feeling unnatural to use, the visual representation of ray casting lacked a clear connection between hand and end-point. Often the research team members would find it difficult to identify which hand controlled the corresponding dot, which results in confusion.

Attention was directed towards the permanently visible Bendcast/Snap-To ray cast method, which locks onto a target further away than the target visualisation, while simultaneously bending the ray cast into a circular arc [17]. Bendcast visualisation and behaviour is controlled by two variables, namely *linearity* of the ray cast and *snapping behaviour*, meaning threshold distance to target before snapping. Previously, this method has shown to be a faster, but more error-prone enhancement method, in controller-held contexts (PS3 and Wii Remote in VR) [22, 3]. However, hand-tracking has shown to be a slower input method in a variety of task contexts [2, 7, 5, 23], although those studies focused on direct object interaction. Using Oculus Quest's default hand-tracking pointing implementation with Bendcast enhancement, the research question for this study is stated as follows:

- *In the context of hand-tracking pointing, how does Bendcast behaviour and visualisation variables affect task completion time, error rate and preference?*

To answer this question, a VR application prototype, utilising the Oculus Quest hand-tracking, was designed and implemented for evaluation purposes.

## 4.2 Design

Unlike study two and three described later in this paper, this study could be conducted alongside the participants in the real world. For this reason, implementation did not have a requirement of focusing on guiding the participant through the application exactly, as guidance would be handled by the facilitator during the experiment. A choice that also allowed for proceeding with a controlled randomised order, to avoid order effects (learning) from affecting the results. For similar reasons, a simplistic environment design was chosen to avoid environmental factors.

Regarding task design, it should be simple for participants to execute, so they do not spend the entire experience learning the interaction task. Avoiding learning effects provided the reason for including a familiarisation scene, where participants could get comfortable with the interaction before testing. By designing the task as too simplistic the data analysis could run into issues of ceiling effects because the participants would complete the test perfectly, which is particularly relevant with error rate as a dependent variable. For this reason, the task needed to incorporate ways of making errors.

A permanently visible ray cast was decided to be implemented, as the research team expected similar experiences of confusion with the default dot ray cast representation to arise with test participants.

## 4.3 Unity Game Implementation

This prototype was created using Unity version 2019.3.0f6, which at the time was the newest official release. It was chosen since this version of Unity officially supports the Universal Render Pipeline, which allows the use of single-pass rendering optimised for VR [34], combined with more control over general performance of graphics [31]. Universal render pipeline replaced the light-weight render pipeline in Unity version 2019.3 but has the same functionality.

### 4.3.1 Oculus SDK

Since the prototype was developed for the Oculus Quest, we decided to use Oculus Integration version 15 (released 21/04/2020) for implementing Oculus SDK features in our project. This SDK was the newest at the time of implementation, including all the tools necessary for hand-tracking.

### 4.3.2 Hand-Tracking Implementation

To implement hand-tracking in the prototype, we used several essential Oculus built-in prefabs. The first one was the OVRCameraRig, which replaces the regular Unity camera with a VR camera. This is necessary for displaying the scene correctly in VR, as it uses two cameras - one for each eye - to display the scene. Furthermore we used the OVRHandPrefab, one for each hand, as a child object to the OVRCameraRig. These prefabs handle functionality for the hands, such as enabling physics and detailed colliders for the hands, scaling of the hands to match the user's actual hand size, and easy changing of the hands' visual appearance, all by simple toggle switching and drag and drop.

Another prefab called HandsManager was used to visualise both hands in the VR space. The HandsManager allows for visualising either the mesh or the skeleton of the hands. We used this prefab to render and visualise the hands as standard meshes.

The last prefab used for the hands was called InteractableToolsSDK. This prefab was used to implement and visualise pointing rays from each hand. Rays are visualised from an origin position between the index finger and thumb, which is the default location of rays when using implementation by the Oculus SDK for hand-tracking.

## 4.4 Scene Implementation

For the evaluation we used three scenes, a familiarisation scene made for the participants to get familiar with the interactions, a test scene for testing and collecting data, and a lobby scene for the participants to chose any of the two former mentioned scenes with different conditions. All scenes had a similar layout and design.

### 4.4.1 Lobby Scene

The lobby scene presented in figure (4.2) was used to enter the training scene and the test scenes. Four square buttons were placed on the wall in front of the user, one button for each test scene. The different buttons indicate different conditions for each scene, which changes how the ray works. Lastly, a button was placed far left of the user, which led to the familiarisation scene.

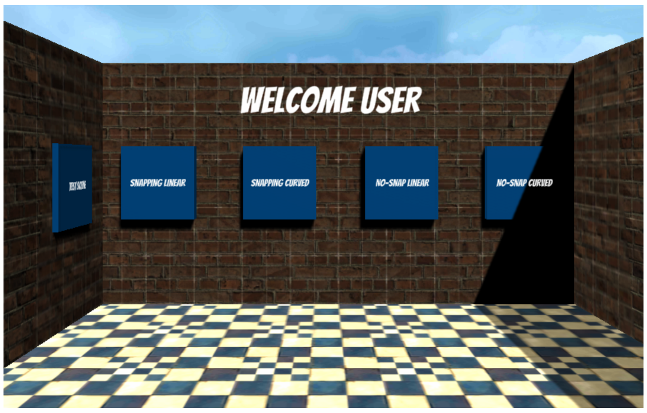


**Figure 4.2:** Screenshot of the lobby scene, with interactable buttons to enter a scene with the corresponding condition applied.

### 4.4.2 Testing Scene

When the participant first enters the test scene, they are shown a single button in front of them, which starts the evaluation upon selection. Once the test is started, 24 boxes appears in front of them (12 green boxes and 12 red boxes), as shown in figure (4.3). Each green box appears with a number 1 to 12. The participant's task is to target and select the 12 green boxes in the order of the corresponding number that appears on the box, starting from 1 and ending on 12.

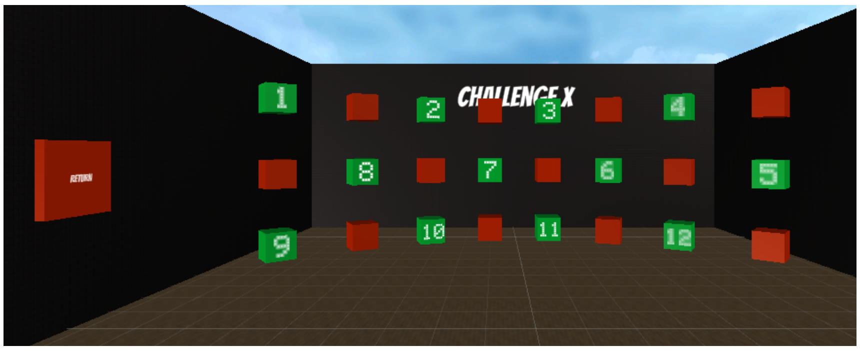


**Figure 4.3:** Screenshot of the testing scene, where green marks an intended box to select and red signifies an unintended box for selection.

If the participant selects a green box in the correct numbered order, the box disappears and a bell sound will play to indicate that the action was correct. If the participant selects a wrong box, a buzzer sound will play to indicate that the action was wrong, but the box will remain active. We evaluated four conditions and used the same scene in every one of them. We did this through a *DontDestroyOnLoad()* function on a game object with a static class to pass variables. This means that the object would be active in the lobby scene where the conditions would be set based on which button was selected, and then these variables would be passed forward upon entering the test scene.

The boxes are placed in a 3x8 grid, with an inter-distance of 1.15m per row and 1.15m per column. Each square box has a width and a height of 0.8m. Participants were standing 6m-7m from the boxes at all times, as the boxes were placed in a curved surface formation at different heights. The curved formation was created in order to surround the user and avoid depth perspective differentiating target size, meaning this formation partially eliminates differences in target acquisition difficulty between targets.

### 4.4.3 Familiarisation Scene

The familiarisation scene seen in figure (4.4) was set up almost identical to the test scene, with the exception that boxes would re-spawn 0.2 seconds after being selected (objects disappear upon selection). All boxes had an identical colour to signify that they all did the same when selected. When the user had familiarised themselves with the interaction, they could return to the lobby by selecting a back button placed on the left wall.

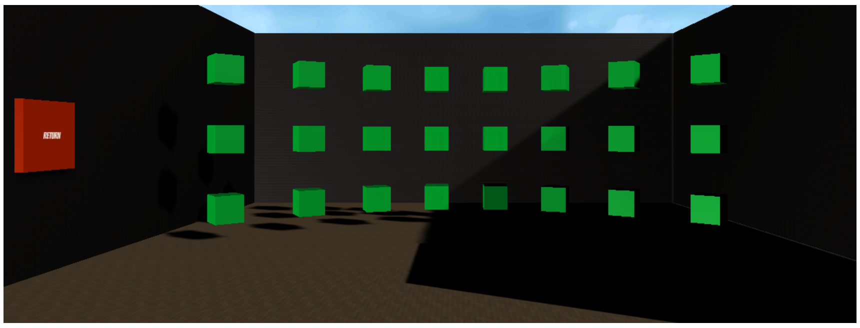


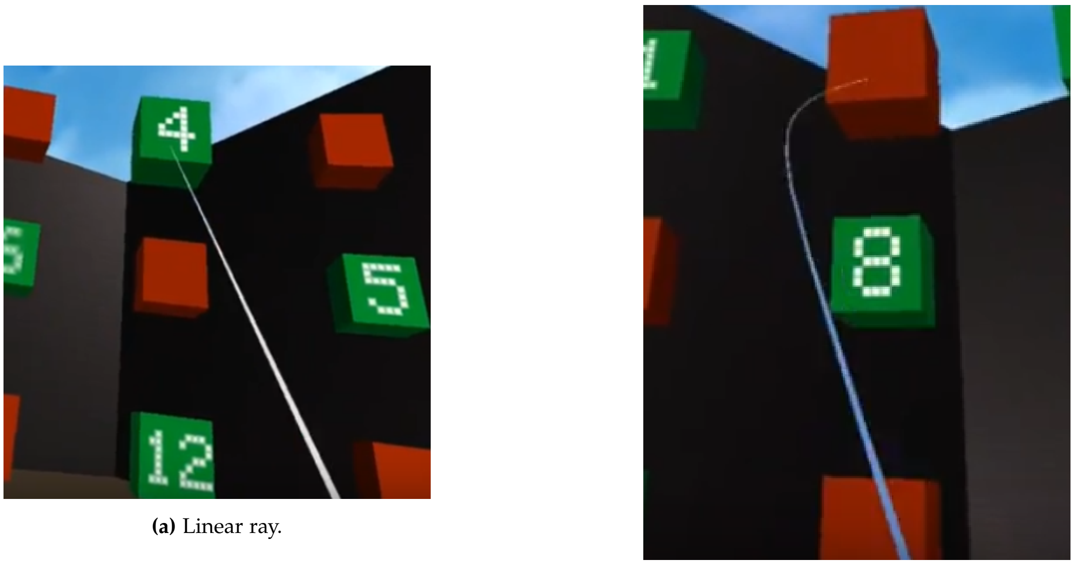
**Figure 4.4:** Screenshot of the familiarisation scene with selectable cubes.

## 4.5 Interaction Implementation

The interaction was done entirely through pinches with ray casting as a selection tool. To interact with an object, the user would point at it with the ray cast, and then pinch their index finger and thumb to interact with objects in the application. The Oculus SDK includes a function *(GetFingerIsPinching())* which is used to calculate the distance between any one finger and the thumb. This function returns a value between 0 to 1, to indicate whether the fingers are far apart or if they are touching, respectively. We created a boolean value to turn true whenever the *GetFingerIsPinching()* method returns a value of 1, which in turn adds 1 to the number of pinches made.

### 4.5.1 Ray Cast Implementation

Ray cast implementation was achieved through the Oculus SDK, using the already mentioned InteractableToolsSDK, which included a script to generate these rays from the users hands. The script already had a default curvature set in the code, so we manually tweaked the code in order to change the curvature. We used a singleton reference to get the information about the amount of curvature and were able the change it through a variable stored in the singleton. The default curvature is determined by a Bezier curve and visualised through a line renderer. The difference in linearity between the two test levels in this study are visualised in figure (4.5).



**(a)** Linear ray.

**(b)** Curved ray.

**Figure 4.5:** Linearity of ray, illustrating the differences in visual representation between the two tested levels.

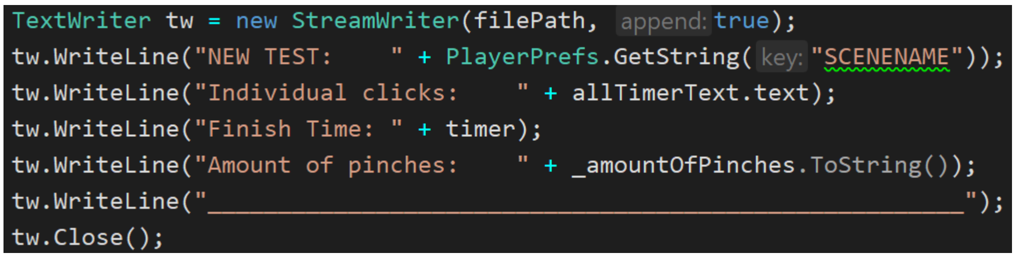
### 4.5.2 Snapping

The snapping functionality was created by the Oculus SDK, where we added a variable to control the magnitude of the snapping, in a similar way as the ray casting curvature. The snapping level is based on the distance from the edges of the objects and is controlled to snap to the closest option of selectable targets.

Distance between the ray and the object for snapping is determined in the script via a method called *FindInteractableViaConeTest()*, which is made by Oculus. In the method we changed a single value to determine the distance required between the ray and the object to snap. When snapping is on (heavy snapping), the ray needs to be 1m from the object's edge to snap. When snapping is off (minor snapping), the ray still snaps to the object, but does so minimally with a required distance of maximum of 0.4 meters between the object's edge and the ray cast.

## 4.6 Data Logging Implementation

For data logging we implemented a feature in the application that created a text file and wrote directly to it. In this file we wrote individual information from the entries in the evaluation, to collect and store data as presented in the code snippet of figure (4.6).



```
TextWriter tw = new StreamWriter(filePath, append: true);
tw.WriteLine("NEW TEST:    " + PlayerPrefs.GetString( key: "SCENENAME"));
tw.WriteLine("Individual clicks:    " + allTimerText.text);
tw.WriteLine("Finish Time: " + timer);
tw.WriteLine("Amount of pinches:    " + _amountOfPinches.ToString());
tw.WriteLine("_____________________________________________________");
tw.Close();
```

**Figure 4.6:** Snippet of the code that stores the captured data in a text file.

The data logged was **completion time**, **total pinches** and **correct box pinch/wrong box pinch, wrong pinch** (leading to no interaction).

- Completion time was calculated trough a float variable which would be increased every frame with Time.deltaTime. It calculates the time between every frame in an *Update()* method which makes the amount of frames irrelevant. The timer would start when participants pressed the start button to indicate that they were ready, ending when the last correct selection was made.

- To detect the total amount of pinches we created a Boolean to be true when the pinch threshold was activated, and false when it was deactivated. This meant that the participant had to release a pinch before the next could be detected. The total amount of pinches would be increased by one whenever this was detected.

- To detect correct box pinches/wrong box pinches/wrong pinches, we created a list which would have an entry added every time a selection was made. On figure (4.6) this list is represented in the third line with the *allTimerText* variable, which stores these throughout the test. A list entry would consist of a timestamp for the selection, along with a letter to represent one of the three categories. This allowed us both to collect information about the amount of times it happened, but also to gain further knowledge about where specifically it happened.

# Chapter 5

# Study One: Linearity and Snapping Ray Cast Behaviour - Evaluation

In this study, we sought to understand how pointing enhanced with Bendcast, based on Oculus Quest's hand-tracking, affects dependent variables such as task completion time, error rate, and preference. Regarding ray casting behaviour, two independent variables were explored, namely *linearity* of the ray cast itself and *snapping behaviour* of the ray upon entering proximity to targets. Each independent variable was tested with two-levels (*Linearity*: Linear and Curved, *Snapping behaviour*: Minor Snapping and Heavy Snapping), resulting in a total of four conditions as follows:

- Heavy Snapping + Linear (S+L).
- Heavy Snapping + Curved (S+C).
- Minor Snapping + Linear (NS+L).
- Minor Snapping + Curved (NS+C).

## 5.1 Measurements

Before engaging with the testing environment, demographics data was collected regarding gender, age, previous VR- and Hand-tracking experience as well as study faculty their education belonged to.

While participants were carrying out the experiment, the system logged their interaction data behind the scenes. Time (seconds) was logged from when the participant started the experiment by pressing the start button on the wall until the last box had been selected. For each box that was selected, a time-stamp was logged

to ensure selection time between boxes could be calculated when processing the results.
Regarding errors, participants can make the following errors:

- Select the wrong (red) box.
- Select without completing interaction (empty pinch).
- Select the wrong instance in the numbered sequence.

For this reason, for each box the system also logged whether it was a correct box in the numbered sequence or a wrongful selection, meaning a red box. Each pinch that occurred over the duration of the test was logged, both pinch selections that were correct and pinches that did not lead to an interaction.

Preference was collected in a ranking task post-testing of the four conditions. Participants were asked to rank the four conditions from 1-4, where 1 indicates the most preferable. The ranking task was completed in a forced-choice questionnaire manner, mixed with physical interaction of placing their rankings (see figure 5.1) in accordance to the recommendations by Anne Marie Kanstrup's description of the visual tangible artefacts (VTA) method. She explains that a VTA is used as a physical tool to support the cooperation of users and their ability to express themselves [1]. Participants were asked if they could provide a reason for their rankings afterward.

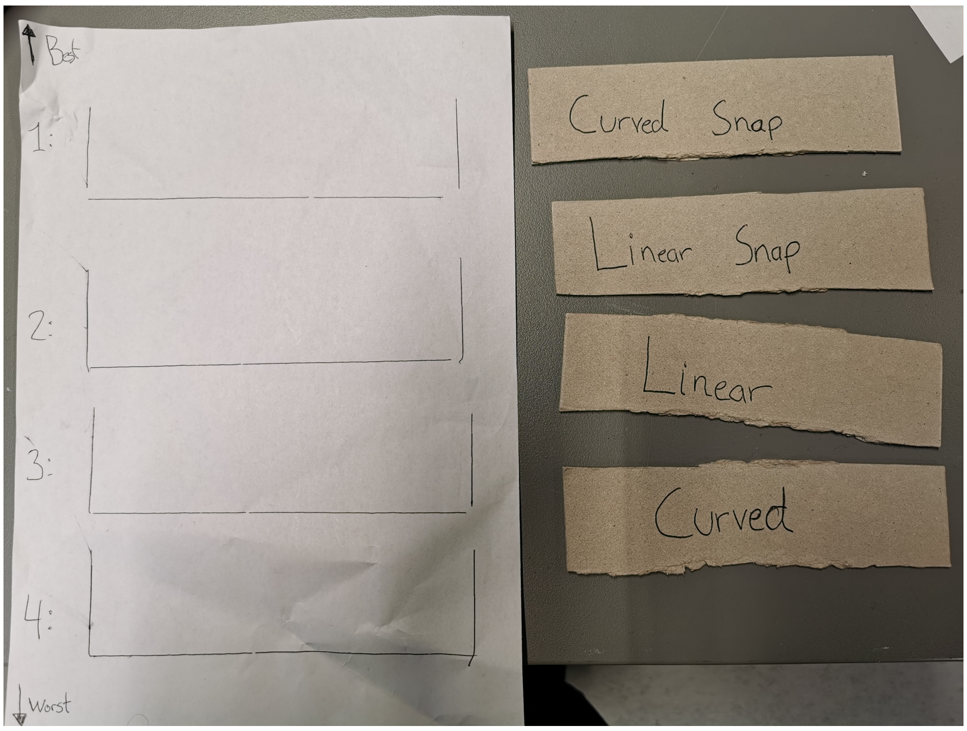


**Figure 5.1:** Ranking method where participants physically place cardboard cutouts in their preferred order.

Besides these measures, the researchers took notes from observations during testing alongside notes from an open post-test question on whether they had comments regarding their experience.

## 5.2 Setup

Testing took place over two days with 12 participants each day and in two different locations. For both locations, the participant and the two researchers were the only people present during the testing procedure and the participants were not physically limited in either location. Therefore we do not see differing locations as a noisy factor since the test takes place in a VR environment. The facilitator could guide the participants through the test by watching their progress through the screen capture streamed from the Oculus Quest headset to the laptop.

## 5.3 Participants

We recruited 24 (13m, 11f) volunteers at Aalborg University for testing, completing a full study design of four conditions. All participants were university students ranging between 21-28 years (Mean = 23.33, SD = 1.9). These participants were split between faculties as follows: Engineering & Science (5), IT & Design (7) Medicine (1), and Humanities (11).

Of the 24 participants, 12 had no experience with VR, 10 had tried it a few times and 2 had tried it several times. In general, they were all inexperienced users, with little to no knowledge about VR. Two of the 24 participants answered "yes" to having tried hand-tracking before, but we believe that they did not know the exact meaning of the question when they answered it, as they further expressed themselves as if they had not tried this kind of hand-tracking before. Therefore, our group can generally be described as first-time users of this technology, with no prior assumptions or opinions.

## 5.4 Procedure

The same procedure was carried out for each participant, structured as follows:

- Recruit test person with a short introduction to study.
- Test person fills out a demographic questionnaire and consents to data collection.
- Detailed introduction to study and how interactions work.
- Enter familiarisation scene (2-3 min).
- Short description of what will happen when they start the actual test.
- Complete conditions in randomized controller order.
- Rank conditions by preference.
- Final comments.

Since we expected most participants to be new to hand-tracking interaction, we created a familiarisation scene where the participants could get used to pointing and selecting with the pinch gesture, as described in section (4.4.3).

Each participant completed all four conditions in a controlled randomised order. To ensure the controlled randomised order was followed, the facilitator guided the participants into the correct testing scenes, each corresponding to one of the four tested conditions.

## 5.5 Results

In summary, the results reported in table (5.1) indicate more critical errors occur at the heavier snapping level (S+L and S+C), whilst providing faster task completion time. Less uneventful interactions also occur as indicated by the "Avg. Wrong Pinches" column, meaning pinch selections that lead to no selection. Preference, as indicated by the last two columns, indicates linearity of the ray had a higher influence on their choice of ranking than snapping level, with the curved condition ranking higher. Unequal ranking within the two linearity (linear/curved) conditions could, however, indicate that the optimal snapping level lies in an interval between the minor and heavy snapping level conditions tested.

**Table 5.1:** Summary of test results on the three dependent variables, *error rate* (Columns "Wrong Boxes", "Critical Error Rate (%)" and "Avg. Wrong Pinches"), *task completion time* (Columns "Total Time 1-12 (s)" and "Avg. Box Selection Time (s)") and *preference* (Columns "Weighted Rank Total" and "Rank").

| | **Error Rate** | | | **Time Completion** | | **Preference** | |
|---|---|---|---|---|---|---|---|
| Condition | Wrong Boxes | Critical Error Rate (%) | Avg. Wrong Pinches | Total Time 1-12 (s) | Avg. Box Selection Time (s) | Weighted Rank Total | Rank |
| S+L | 14 | 4.86% | 2.92 | 21.69s | 1.97s | 53 | 4 |
| S+C | 6 | 2.08% | 2.75 | 21.90s | 1.99s | 63 | 2 |
| NS+L | 1 | 0.35% | 6.75 | 24.75s | 2.25s | 57 | 3 |
| NS+C | 2 | 0.69% | 6.38 | 22.97s | 2.09s | 67 | 1 |

## 5.6 Analysis

### 5.6.1 Task Completion Time

Given that the experiment was conducted with four conditions, a one-way repeated measures ANOVA test seemed to fit the criteria. There are three assumption that must be ensured to obtain a valid analysis when using ANOVA, which is listed as follows:

- Each data sample must be obtained independently.
- The data must be normally distributed. This can be visualised with a normal Q-Q plot and validated with the Shapiro-Wilk test.
- Variance must be homogeneous, meaning no statistical differences must be present in regards to variance between groups. Homogenity is tested with Levene's Test.

The first analysis pass revealed no statistical differences between any of the tested groups (p = 0.275), however, validation of test assumption revealed three outliers, as shown by the three numbered data points in figure (5.2).

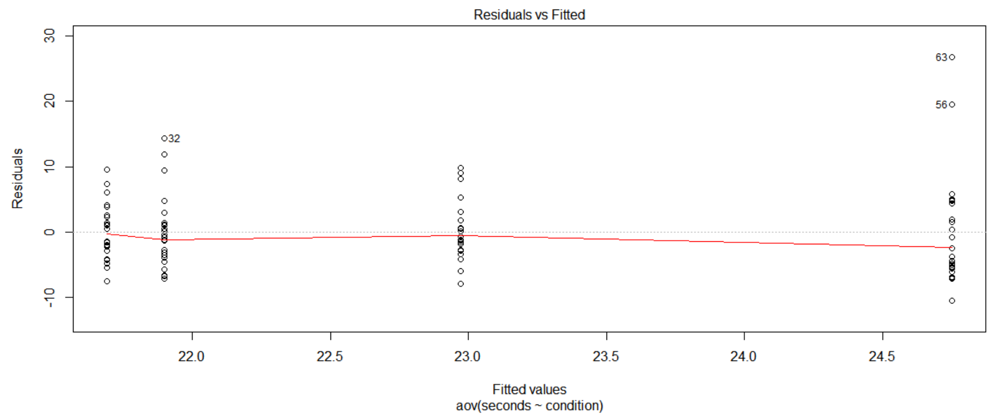


**Figure 5.2:** Residuals vs. fitted plot, showing indication of variance homogeneity and outlier detection.

While homogeneity of variance was existent as found by Levene's test (p = 0.1224), which is above the significance level of 0.05, further analysis into normal distribution made it clear that the data was not normally distributed. As shown in the normal Q-Q plot in figure (5.3), the outliers stray far away from fitting the linear line, suggesting normal distribution is not present in the data. A follow-up Shapiro-Wilk test only confirmed the suspicion with a p-value below 0.05, as shown in figure (5.4).

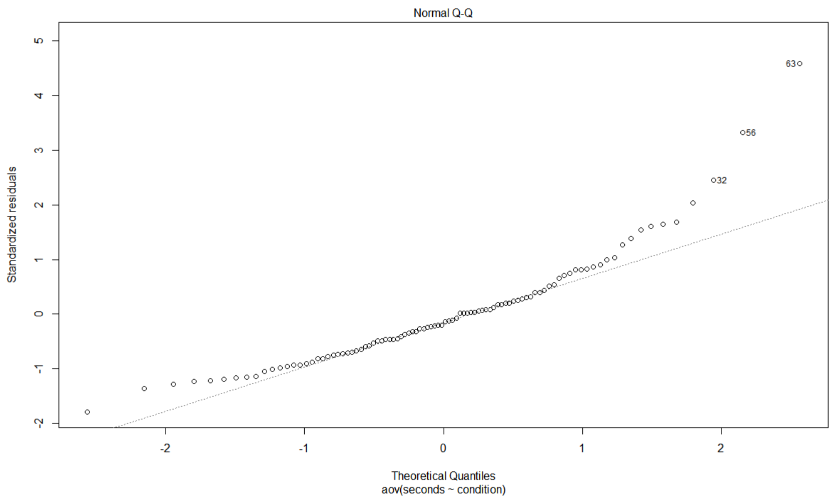
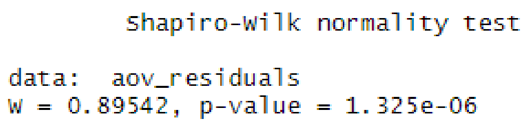


**Figure 5.3:** Normal Q-Q plot, indicating the data is not normally distributed as some of the samples do not fit the straight line.



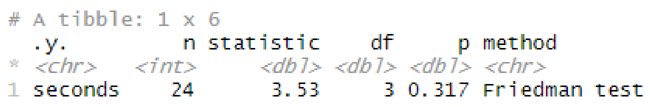
```
                    Shapiro-Wilk normality test

data:  aov_residuals
W = 0.89542, p-value = 1.325e-06
```

**Figure 5.4:** Shapiro-Wilk normality test, showing the data is not normally distributed with p < 0.05.

Based on these findings, the three detected outliers were deleted and the analysis re-run. Two outliers belonged to condition NS+L, the last one belonged to condition S+C. After rerunning the analysis, a similar conclusion was reached as the data was not found to be normally distributed (Shapiro-Wilk p = 0.01164). Therefore the ANOVA test was not found suitable for the data.

Instead, the non-parametric Friedman's test was run on the whole data-set, which did not show any significant differences between the four tested conditions, as seen by the summary in figure (5.5).



```
# A tibble: 1 x 6
  .y.         n statistic    df     p method
* <chr>   <int>     <dbl> <dbl> <dbl> <chr>
1 seconds    24      3.53     3 0.317 Friedman test
```

**Figure 5.5:** Friedmann test summary, showing p > 0.05, meaning no significant differences between task completion times were found.

Therefore, it can not be concluded that the tested conditions influence task-completion time in any significant ways.

### 5.6.2 Error - Wrong Pinches and Wrong Box Selections

Regarding the number of wrong pinches, meaning pinch selections that lead to no interaction in the scene, the summary states 2.75-2.92 (1.23 pinch selections/box) of such pinches would occur for heavy snapping conditions (S+L and S+C). For the minor snapping conditions (NS+L and NS+C), this number increases to 6.38-6.75 (1.55 pinch selections/box). Running Friedman's test (p = 1.448047e-07) on the data followed by a Bonferroni corrected Wilcoxon Signed-Rank test shows significant differences between minor/heavy snapping conditions, but none were found between the two minor snapping conditions, or two heavy snapping conditions, as shown in the summary figure (5.6).

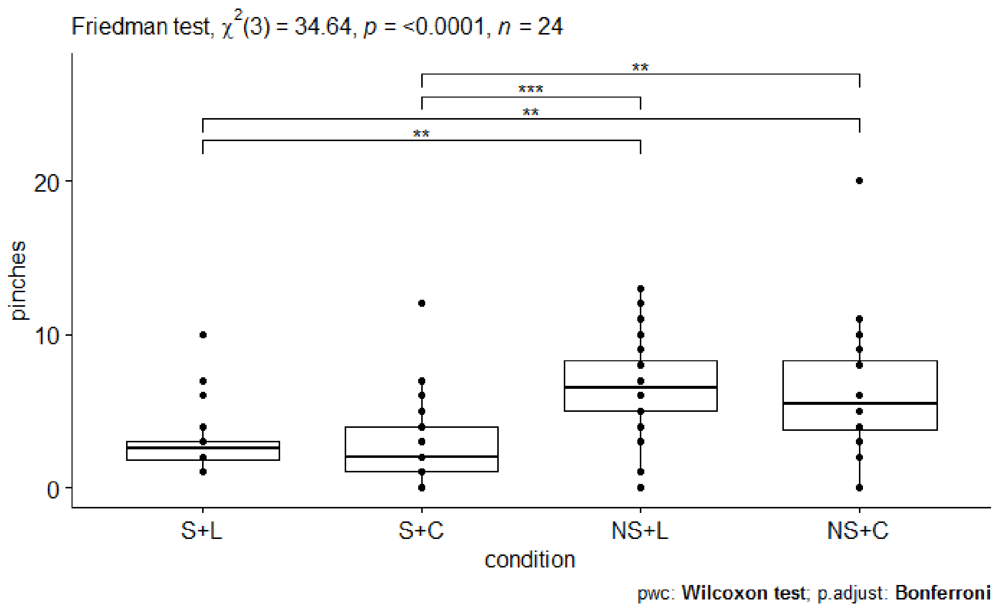


**Figure 5.6:** Test summary, showing significant differences between heavy/minor snapping conditions in regards to wrong pinches.

Regarding the number of wrong box selections, the opposite story follows. Heavy snap + linear ray (S+L) condition produced 14 errors, meaning an error rate of 4.86% (14/288 total selections). The error rate falls to 6 errors (2.08%) for heavy snap + curved (S+C) condition, but both are considerably higher than the minor snapping conditions (minor snap + linear ray (NS+L) = 1 error, 0.35%) and (minor snap + curved ray (NS+C) = 2 errors, 0.69%).

More pinch selections are required to achieve the interaction intent with minor snapping levels, but errors produced heavy snapping conditions are considered more critical, as selections would complete an unintended interaction.

### 5.6.3 Preference

Table (5.2) lists the frequency at which participants ranked the conditions in regards to preference. To produce a more meaningful way of looking at this data, the rankings are weighted, summarised into a single score and visualised.

| Condition | Rank 1 Frequency | Rank 2 Frequency | Rank 3 Frequency | Rank 4 Frequency |
|---|---|---|---|---|
| **S+L** | 5 | 5 | 4 | 10 |
| **NS+L** | 5 | 6 | 6 | 7 |
| **S+C** | 7 | 4 | 10 | 3 |
| **NS+C** | 7 | 9 | 4 | 4 |

**Table 5.2:** Frequency of rankings per condition, as voted by the participants.

Weighted rankings are calculated by assigning a weighted ranking score to each of the rankings. Rank 1 gives 4 "ranking points", rank 2 gives 3 etc. For example, looking at S+L (heavy snapping + linear ray), rank 1 is weighted at 20 points (4*5) whereas rank 3 is weighted at 8 (4*2) points. Maximum obtainable score is 92, if all participants rated one condition as the most preferable and another one the least preferable. The least preferable would obtain 24 points at minimum in that case. After weighting calculations, the result produces table (5.3), which is visualised in figure (5.7).

| Condition | Weighted Total Rank<br>Max = 92<br>Min = 24 | Rank |
|---|---|---|
| **S+L** | 53 | 4 |
| **NS+L** | 57 | 3 |
| **S+C** | 63 | 2 |
| **NS+C** | 67 | 1 |

**Table 5.3:** Weighted total rank based on frequencies in table (5.2), alongside resulting ranking table of the tested conditions.

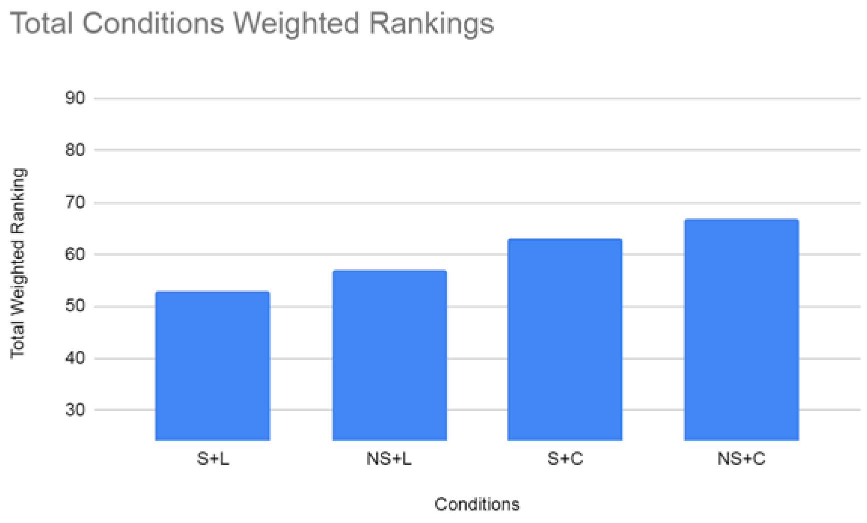


**Figure 5.7:** Weighted rankings by conditions visualised to get a clearer view on the preference differences between conditions.

Clear separation between linear and curved conditions is visible, with overall preference for the curved ray method. Considering none of the rankings are equal in score, it could suggest that an ideal snapping level value lies somewhere between the two snapping levels tested.

## 5.7 Discussion

Finding no significant differences in task completion time stands in contrast to previous studies which has indicated that Bendcast should be a faster interaction method compared to traditional linear ray casting [22, 3]. The underlying pointing implementation differs between this study (hands) and theirs (PS3 controller [3] and Wii Remote controller in VR [22]), which could suggest that Oculus Quest's default pointing method is more influential than the enhancement method. Finding no significant differences in completion time for this study could further suggest that hand-tracking interaction raises the floor, or lower-limit, meaning completion time is inhibited by the input method and not the enhancement method. This would fit with the knowledge that hand-tracking interactions have previously shown to be a slower input method compared to controllers [2, 7, 5, 23]. It is important to note, however, that the emulated traditional ray casting method (NS+L condition) is still subject to minor snapping, meaning it can be considered a tweaked Bendcast method rather than a traditional ray cast method. For future studies, it would be relevant to examine this relationship between input method and enhancement method effects in completion time.

The amount of wrong boxes increases with heavier snapping (ray snaps onto the middle of the target when the ray is 1m away from the target) while the amount of wrong pinches is less. This can be described due to the fact that mistakes often were made when the participant pinched their fingers for selection, where the slight physical finger movement caused by natural finger tremors pushed the ray to another target. With heavy snapping, the ray had a chance of targeting another unintended box, whereas with minor snapping (ray snaps onto the middle of the target when the ray is 0.4m away from the target) the ray would simply get unattached from the current box and hit nothing instead.

When considering this issue further, it becomes clear that hitting wrong boxes is a much bigger issue than making a pinch that hits nothing, which lets us believe that heavy snapping is not ideal to use in situations where making a mistake is critical. This could lead to some unfortunate errors if the results should be understood in the context of user interfaces, where a wrong selection could send the user into a different menu than intended. While not tested in this study, Moore et al. solved this issue by introducing a backlog of 0.2 seconds, where the pointing target is logged for each frame. At the time of selection, the target with most entries gets selected [22]. This would be relevant for future studies to employ, as it is an enhancement that affects the underlying pointing implementation without interfering with other enhancement techniques.

Another thing we can see is, that the amount of wrong boxes is less with a heavy snapping + curved (S+C, 6 wrong boxes) compared to heavy snapping + linear (S+L, 14 wrong boxes). This difference can be described due to feedback of the curved ray while snapped, as the curve would continuously react in size and direction based on the hand movement even when snapped. This made it easier for participants to know when the ray was about to get unattached from the box, and therefore they would be less likely to make mistakes with a curved snap than linear snap. One of the participants said *"I liked the feedback from the curved ray when I was moving away from something, but I would prefer that it did not get anymore curved than it already is"*, highlighting one reason for why the curved condition was preferred. Similar statements for liking the curved feedback was given by five other participants, meaning it was more than just a one-off sample.

This study did not consider targets at varying depth levels, meaning the results cannot be extrapolated to contexts different from 2D interactions. At varying depth levels, occlusion would be an issue that does not seem likely to be solved by snapping levels. If an object is partially occluded by another object, it would be considerably more difficult to hit the back target than the front target. Solutions would need to be implemented to solve this problem if snapping is to be used in complex 3D contexts. Secondly, since the snapping level was dependent on distance to the target, the snapping level is dependent on the distance between objects which another variable to consider in the scene design.

## 5.8 Conclusion

To conclude upon the research question, it was decided to split the conclusion into two parts; one for *linearity* and another for *snapping*.

### 5.8.1 Curved versus Linear Ray

The results show a clear tendency towards curved being preferred. The two highest-scoring rays in preference were both curved and the general comments by the participants who chose curved showed that the feedback of the curve was the key in choosing this. There were still several participants who choose linear as their preferred, and therefore it might be worth considering to reduce the size of the curve a bit to find a middle-way which is more generally appealing. On another note, since no significant differences were identified in regards to completion, decision Bendcast loses one of its advantages over traditional linear ray cast. However, one should be cautious of this conclusion, as the emulated linear ray still snapped to the target, although in minor fashion.

### 5.8.2 Effect of Snapping

Based on preference, there was a slight tendency towards snapping not being preferred. When the participants commented on this, it was mentioned several times that reducing the snapping magnitude could improve the quality of the snapping, which leads us to consider that the relatively ambiguous results could prove that finding a middle point between the two levels would be a good idea.

The data shows that snapping has a lot more errors in choosing the wrong box. We consider these errors critical compared to a pinch that hits nothing, as choosing the wrong target leads to more critical mistakes. This effect is important to remember when considering the amount of snapping for an application, as the right amount can depend on the context and severity of miss-clicking a box. Snapping did prove to yield quicker completion times, though not significantly, which adds relevance to considering snapping in the right context; if something has to be done quickly, errors are not critical or for simplistic scenes where errors are not likely to occur due to sparsity in object density.

# Chapter 6

# Study Two: Hands vs. Controller Interactions - Design & Implementation

This chapter discusses the design and implementation of the hand-tracking vs. controller-tracking prototype. The motivation for the study is established at first, containing the specification of the question this study seeks to answer. A description of relevant design choices for the interaction tasks follows, which further leads into sections on implementation. Functions used to implement the code and relevant source code will be displayed and explained.

## 6.1 Motivation

The motivation for this study was guided by input from the collaborative partner (Unity Studios) and the gap identified in the literature on hand-tracking vs. controller-tracking interactions. The previous study on ray cast pointing was initially meant as a preliminary study but changed identity to a complete study after deliberating on the initial 12 results from day one that were found lacking in providing a meaningful result. Continuing down the track of pointing, the research team identified that proceeding would require diving into the underlying pointing implementation. Such an approach was considered less relevant to the collaborative partner, who was more interested in direct object interaction in VR training scenarios. Combined with our wish to explore direct object interactions, the focus shifted away from pointing in this study. The new focus is based on the potential hand-tracking opens for fine-motor control interactions, which should match the VR training task closer to the real-world task. To understand hand-tracking interactions' potential for these scenarios, a benchmark input method was needed for comparison, which is predominantly considered tracked controllers.

Previous studies on the topic of hand-tracking vs. controller-tracking interactions paint a picture of hand-tracking resulting in a slower, more error-prone, and worse user experience. However, often studies have taken the approach of investigating singular interactions without a real-world context [2, 5, 7, 30], or analysed the data as an overall average of completion time in a designed training scenario [23]. Therefore it is identified that there is a lacking understanding of which exact interactions contribute to slower interaction times, errors, and user experience when training scenarios are considered as a contextual element. Secondly, context effects on preferential choice do not appear to have been reported in the previous studies reported in this project. Thirdly, interactions designed with Oculus Quest's hand-tracking has not been formally compared to controller-tracking in previous literature.

With a focus on singular interactions, the research question is stated as follows:

- *In the context of VR training scenarios, are there singular hand-tracking direct object interactions that perform better, or are found to be suitable alternatives to equivalent controller-tracking interactions?*

To answer this question, a prototype with three singular interaction tasks was designed and implemented with the purpose of conducting remote testing, due to the Covid-19 situation during spring of 2020 when the project was conducted.

## 6.2 Design

Inspiration for the interactions came from our collaborators at Unity Studios. We asked them to send us a list of possible interactions that they could see use in a VR training simulation context, where we chose the following three;

- Pushing buttons using direct object interaction (*Button-pushing*).
- Grabbing an object and placing it somewhere (*Grab and release*).
- Grabbing an object and rotating it (*Rotation*).

Due to the Covid-19 situation, it was not an option to physically attend the evaluation and supervise participants. Therefore it was necessary to conduct the evaluation online, which was managed by publishing the application on a community driven VR application platform called SideQuestVR. SideQuestVR allows anyone with an Oculus Quest to sideload applications from the SideQuestVR store to their device. For this reason, the data collection shifted to collection through Unity Analytics, instead of locally on the device. For similar reasons, it was not an option for the research team to replicate a real-world training scenario, as it would have required observation of the process occurring in the real world. Testing with a real-world training scenario replicated in VR would have been the better option to explore for understanding the contextual relation, but was not an option for the project.

Since it was known at the time of design and implementation that the study was to be conducted remotely, decisions were taken to create a more engaging experience for the participants, to increase the chances of getting more data entries. Gamification of the three interaction tasks seemed an obvious choice for engaging the participating during testing, taking the approach of designing around the specific interaction in question.

Remote testing required guiding participants inside the application, which could be achieved by placing text and graphical demonstrations of interaction inside the application. Video call with the participant during testing was considered as an option for guidance, but not found a reliant tool, as it was expected some participants would not want to go through the effort.

## 6.3 Unity Game Implementation

In this section we reference the prototype as a single prototype comparing hand-tracking and controller-tracking, but technically we split this prototype up into two separate applications. This was necessary given an issue was encountered with getting both hand-tracking and controller-tracking to work in the same project. We never concluded on the real reason behind this issue but decided to move on and develop two prototypes with each of their tracking modes, instead of one with both tracking modes. Unity and Oculus SDK versions used were the same as reported for the first prototype in section (4.3).

### 6.3.1 Hand-Tracking

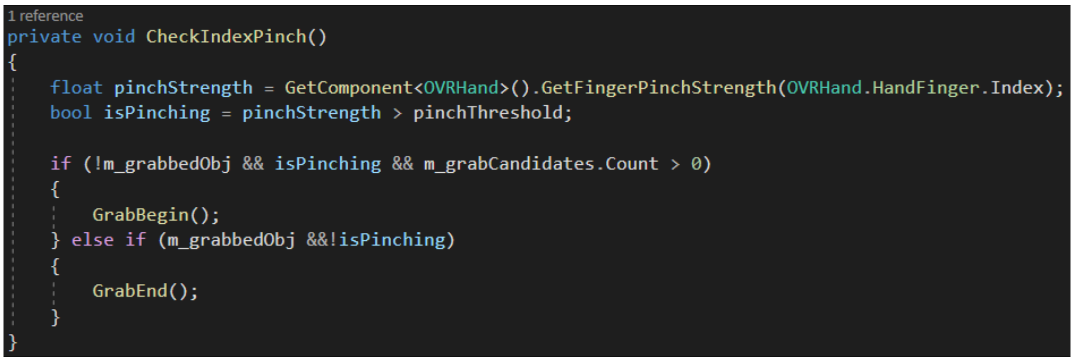
Hand-tracking implementation is similar to the first prototype in terms of how it uses Oculus prefabs to set up the overall scene functionality (section (4.3.1)). One difference is that this prototype has implemented physics for some interactions (*button-pushing*) and a trigger system for other interactions (*grasp and release* & *rotation*). The trigger system was implemented by placing a sphere collider on the hands, set as a trigger. When a grabbable object collided with the sphere collider and a pinch was detected, the user would grab the object.

### 6.3.2 Controller-Tracking

The controller-tracking for this prototype was implemented to render the hands instead of the controllers in the VR space. This was done since we wanted the controller interactions to visually imitate the hand interactions in the VR space. Implementing controller-tracking was done effortlessly since the default Oculus implementation only requires drag and drops of a couple of prefabs.

### 6.3.3 Pinching

In order to grab an object with a pinch, the participants needs to pinch inside the object. As mentioned in chapter (4.5), the Oculus SDK includes a function that returns a value from 0 to 1 to indicate whether any finger and the thumb is pinching. This can be seen on the code snippet in figure (6.1), which shows how this data was retrieves from the Oculus SDK and further used.



```
1 reference
private void CheckIndexPinch()
{
    float pinchStrength = GetComponent<OVRHand>().GetFingerPinchStrength(OVRHand.HandFinger.Index);
    bool isPinching = pinchStrength > pinchThreshold;

    if (!m_grabbedObj && isPinching && m_grabCandidates.Count > 0)
    {
        GrabBegin();
    } else if (m_grabbedObj &&!isPinching)
    {
        GrabEnd();
    }
}
```

**Figure 6.1:** Code snippet showing how the *GrabBegin()* function was called in order to allow for grabbing an object with a pinch.

In order to implement pinching with grabbing we elaborated the functionality with a threshold that decides when a pinch will be detected. Our pinch threshold were set to 0.7 based on the trial and error method we conducted, and when true it activates grabbing if an available grab candidate is detected.

## 6.4 Scene Implementation

In order to separate the interactions into singular focus, the application was built with multiple scenes, where pressing virtual buttons would transport the user into the intended scenes. The application had a familiarisation scene, a lobby scene, and three different singular interaction scenes.

### 6.4.1 Familiarisation Scene

When the application was opened, the participant would start in the familiarisation scene as seen in figure (6.2). In this scene they would experience the three different interactions and have a chance of trying them out before doing the actual test. All three interactions were included here, and worked in the same way as the test scenes. The familiarisation scene did not have a time limit.

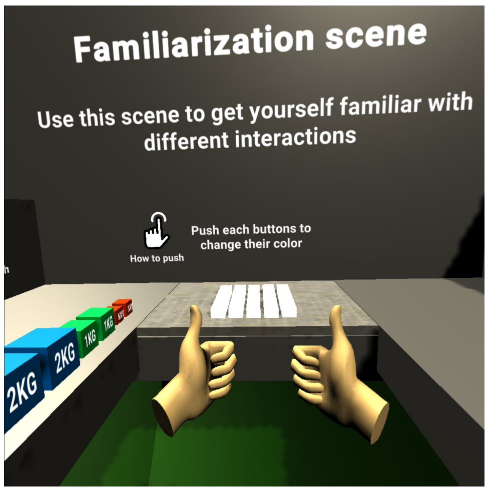

Familiarization scene
Use this scene to get yourself familiar with different interactions
How to push
Push each buttons to change their color
2KG
2KG
1KG
1KG

**Figure 6.2:** Familiarisation scene for the hand-tracking application, containing test objects equivalent to those found in the interaction task testing scenes.

### 6.4.2 Lobby scene

The lobby scene presented in figure (6.3) was a simple room, containing a table with three buttons on top of it. Each button would lead to another scene, containing one of the interaction tasks. The text was displayed on top of each button to provide the user with information as to which scene each button would place them in.

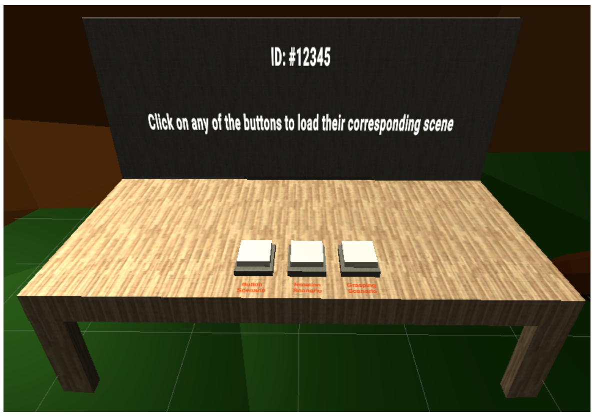

ID: #12345
Click on any of the buttons to load their corresponding scene

**Figure 6.3:** Lobby room, containing three buttons for transporting the user to each of the respective scenes containing the corresponding interaction task.

## 6.5 Interaction Task Scenes Implementation

There was a total of three interaction task scenes where each test scene was designed with the same layout, but a slight difference in materials used. The scenes all had one table in the middle of the room, which the user would start in front of. The tables had a backboard storing general information about the scene and the status of the user's progress in the scene. A button was placed on the table, which participants used to start the test, submit an entry for an individual trial, finish the test, and return to the lobby scene. The three interaction scenes included tasks for 1.) *button-pushing*, 2.) *grab and release* and 3.) *rotation* of virtual objects. All interactions were put into a gamification context, where the user had to complete a task for each interaction three times.

### 6.5.1 Button Pushing

For *button-pushing*, physics was used where the participant had to interact with the button as if it was a real button. With hands this was done through touch, while the controller selection would occur by pressing the side-trigger button when in the collision range of the button. For this interaction task, a 5x5 matrix of buttons was created as shown in figure (6.4). Each button would change colour between black and white when pressed. The idea of this interaction was to replicate the presented pattern by pushing buttons. All the buttons would start out as white, where the user had to colour the black pattern. The buttons were implemented through a spring-joint setup with a connection to an invisible object floating above each button. When a button was pushed down, the connected spring-joint would pull it back up.

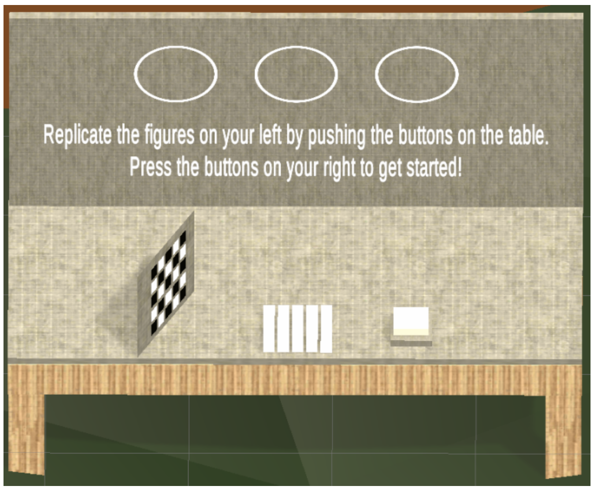
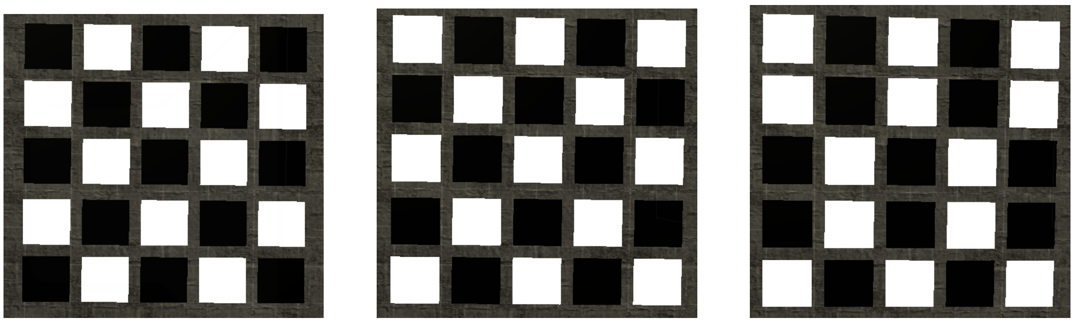

Replicate the figures on your left by pushing the buttons on the table.
Press the buttons on your right to get started!

**Figure 6.4:** Button pushing scene, with the replication pattern to the left and the interactable buttons on the table in a 5x5 grid.

The *button-pushing* task was implemented with the use of Unity's physics system and a spring system. By setting up the spring-joint and using colliders to constrain the movement of the button, it was ensured that it would always stay in place and bounce back up after being pushed down. The participant had to replicate the three patterns in succession, as illustrated in figure (6.5).



**(a)** First replication pattern. **(b)** Second replication pattern. **(c)** Third replication pattern.

**Figure 6.5:** Replication patterns participants went through in succession during the *button-pushing* task.

### 6.5.2 Grab and Release

The *grab and release* interaction task presented in figure (6.6) was created around an authentic balance scale with squares as weights. The weights had a range between 0.5-2kg and could be picked up and placed on the two platforms of the scale. The participant completed the task three times with a requirement of balancing 4.5kg, 5.0kg, and 5.5kg respectively, meaning a minimum of 6, 6, and 8 interactions per respective task repetition.

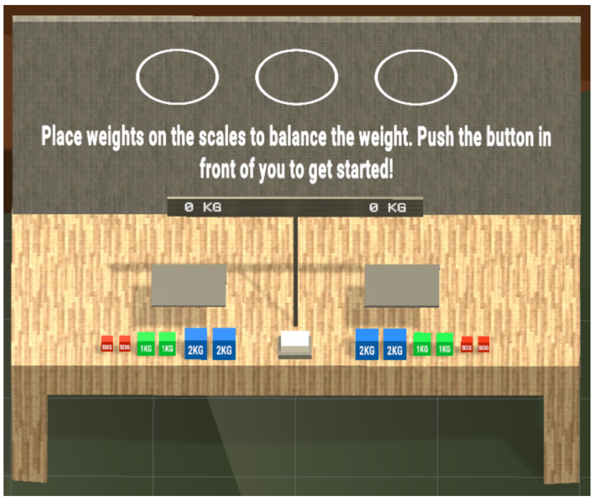

Place weights on the scales to balance the weight. Push the button in front of you to get started!
0 KG
0 KG
1KG
1KG
2KG
2KG
2KG
2KG
1KG
1KG

**Figure 6.6:** *Grab and release* interaction scene, where participants pick up cubes to balance the scale.

The balance scale was made through a spring-joint system, using the real mass of the weights to balance the scales. Each of the grabbable weights contained a script from the Oculus SDK called OVRGrabable. This script easily allowed interactions with the objects, with a button-press on the controller or by pinching with hand-tracking to grab the objects and position them around the scene. Physics were not used for this interaction.

### 6.5.3 Rotating Ojects

For the *rotation* interaction, the user had to match the rotation of an objects floating in the air. As presented in figure (6.7), they would see two dice cubes in front of them where they had to match the rotation of the green cube to the blue cube. The task was completed three times at three different rotations, namely (**1:** x = 50, y = 180, z = 180), (**2:** x = 0, y = 100, z = 0) and (**3:** x = 40, y = 180, z = 60). A minimum of one selection is required to complete each rotation, but participants are expected to re-select the object multiple times to achieve the precision required for this task.

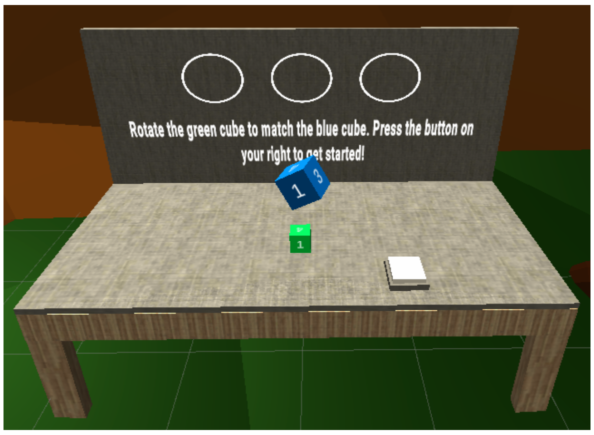
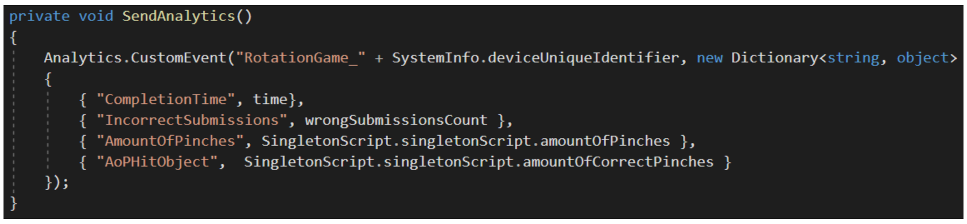

Rotate the green cube to match the blue cube. Press the button on your right to get started!

**Figure 6.7:** Screenshot of the rotation scene, where participants would pinch and rotate the green cube to match the blue cube's rotation.

It was decided to use an object without gravity for this task. This way it could be positioned in the air in front of the user, without it falling down and losing the wanted rotation. Colliders were set as triggers on the controllers/hands, so they would be able to go through the object and grab it (pinching with hands, trigger button press for controllers). When the object was grabbed, it would follow the change in rotation of the hand/controller. The object would not snap to the same rotation as the hand/controller when picked up, but rather follow the change in angles since it was picked up. To accept a rotation as accurate, a threshold was created with a magnitude of 30 degrees.

## 6.6 Data Logging Implementation

For data logging we used Unity Analytics. We created individual custom events for each scene, which would be sent once upon finishing each of the test scenes. Custom events, as presented in figure (6.8), containing information regarding completion time, the total amount of pinches and the number of pinches that hit an object for intended interaction. This meant that each participant would send a total of six events, three for each application, which would hold all the data we needed. As we needed some way of identifying people between the two applications, we stored unique device info through SystemInfo.deviceUniqueIdentifier. We furthermore provided each participant with a randomly generated ID, which they had to remember and type into the questionnaires. This was used to connect the survey entries to the data stored through Unity Analytics.



```
private void SendAnalytics()
{
    Analytics.CustomEvent("RotationGame_" + SystemInfo.deviceUniqueIdentifier, new Dictionary<string, object>
    {
        { "CompletionTime", time},
        { "IncorrectSubmissions", wrongSubmissionsCount },
        { "AmountOfPinches", SingletonScript.singletonScript.amountOfPinches },
        { "AoPHitObject",  SingletonScript.singletonScript.amountOfCorrectPinches }
    });
}
```

**Figure 6.8:** Code snippet that stores the analytics for each scene containing variables for completion time, total amount of pinches and pinches that hit an object.

# Chapter 7

# Study Two: Hands vs. Controller Interactions - Evaluation

The following study was conducted with the purpose of investigating whether there are singular VR interactions that are more preferable with Oculus Quest's hand-tracking than their controller-tracking, alongside an exploration of whether application context (games vs. training scenario) affects the preference decision. Dependent variables include time completion, pinch interactions, preference, and select presence questions between hand- and controller-tracking. In three different interaction tasks (*button-pushing*, *grab and release* and *rotation*), participants went through two applications - one for hand-tracking and one for controller-tracking - followed by answering questionnaires regarding their experience.

The experiment was conducted remotely, meaning the researchers were not present as the participant went through the procedure with their privately owned Oculus Quest headset. An instruction sheet was provided for the participant alongside options for contacting the researchers if they encountered issues or had questions about their experience. See appendix section (A.1) for the instructions sheet.

## 7.1 Measurements

Overall task-completion time was logged and saved behind the scenes as the participant proceeded with the experiment. Task completion time was initiated from the moment the participant started the task by pushing the button on the table the first time until the third repetition of that task was completed. The same would occur behind the scenes in the other two tasks.

As for pinch interactions, things got more complicated due to the experiment setup. Initially, the system was set up to record the total number of pinches over the course of the task and pinches that resulted in an interaction. The difference between the two numbers would be used as the error rate, representing the number of interactions that did not land in an intended interaction. While logging of pinches resulting in an interaction worked in internal tests by the research team, data was not collected correctly as the program went live for participants. Therefore, error rates are not a measurement in this study but are instead replaced by an overall average difference in the total number of pinches/controller trigger activations during the tasks.

Preference was measured with three questions posed to the participant post-test. One overall question on their immediate interaction method preference (Hands or Controllers) followed by two preference questions placing the user in a game- and training scenario respectively, before asking which interaction they would prefer in those cases. Similar questions were presented to participants in all three tasks, with slight modifications to them so they matched the interaction.

Finally, we chose six select questions from the Witmer & Singer Presence Questionnaire [36], regarding the naturalness of their interaction experience, adjustment to the interaction method, and whether the interaction method interfered with task completion. The interference question was modified by combining the essence of three questions into one. The six questions presented to participants were as follows [36]:

- How natural did your interactions with the environment seem?
- How much did your experiences in the virtual environment seem consistent with your real-world experience?
- How well could you move or manipulate objects in the virtual environment?
- How quickly did you adjust to the interaction experience?
- How proficient in moving and interacting did you feel at the end of the experience?
- How much did the interaction method interfere with the performance of assigned tasks or other activities?

These questions were asked twice, once in the context of each condition. Questions were ranked on a 1-7 scale, from very bad to very good, with the wording matched to the question-wording. The full Witmer & Singer Presence Questionnaire is included in Appendix section (A.3). See Appendix section (A.2) for the full length of questionnaires used in this study.

## 7.2 Setup

Due to the Covid-19 situation, testing was done online over a duration of several weeks. The application was released on the platform SideQuest to make it available for online testing and the potential of gaining additional test participants.

## 7.3 Procedure

Since the researchers were not present during the testing procedure, an instructions document was formulated with the necessary information enabling the participant to conduct the experiment in their own time. The document contained information regarding testing context, direct links to the two applications, procedure steps, and links to the questionnaires on demographics, preference, and presence. The full document presented to participants as a view-only link can be seen in the appendix section (A.1). Participants were instructed to go through both applications first, followed by answering the five questionnaires post-test. Inside the applications, participants could make their own choice of order in which they entered the interaction task scenes.

## 7.4 Results

Data collection did not work correctly for participants trying the hand-tracking application, which resulted in missing data. Some participants only had data for completion time, whereas pinch interaction data was missing. Other participants had pinch interaction data but lacked the completion time data. Data from controller interactions were mostly complete from 50 participants, but hand-tracking data from 100 participants were subject to the problem described. The few cases (5-10, dependent on task) where matching was possible, the data was still fractured, resulting in comparisons of 3 data points vs. 10 data points for instance. Paired data was therefore not an option and it was instead decided to take the approach of independent sample comparisons, which raised the number of data points to a range of 9-42, dependent on measurement. It is important to state that each of the subsequent result sections is treated as mostly independent from each other, as data did not come from exactly the same participants.

### 7.4.1 Time-Completion

Before conducting any statistical analysis, the data was analysed for outliers. By plotting the time completion data in a boxplot for each condition and task combination, outliers were identified and discarded. For instance, in figure (7.1), a boxplot of the controller condition in the *button-pushing* task is visualised, where the three grey dots at the top are identified as outliers. The same approach was taken for the other five condition and task combinations. Eight data points (3 in controller, *button-pushing* task. 2 in controller, *grasping* task. 3 in hands, *rotation* task) were discarded from this method.

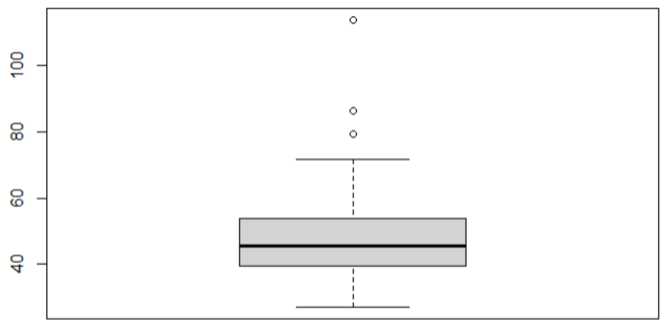


**Figure 7.1:** Boxplot visualisation of time completion data for the controller condition in the *button-pushing* task, where three data points are identified as outliers.

On average, completion time with Oculus Quest's hand-tracking was 2.18-2.39 times slower than controller-tracking interactions, as presented in figure (7.2).

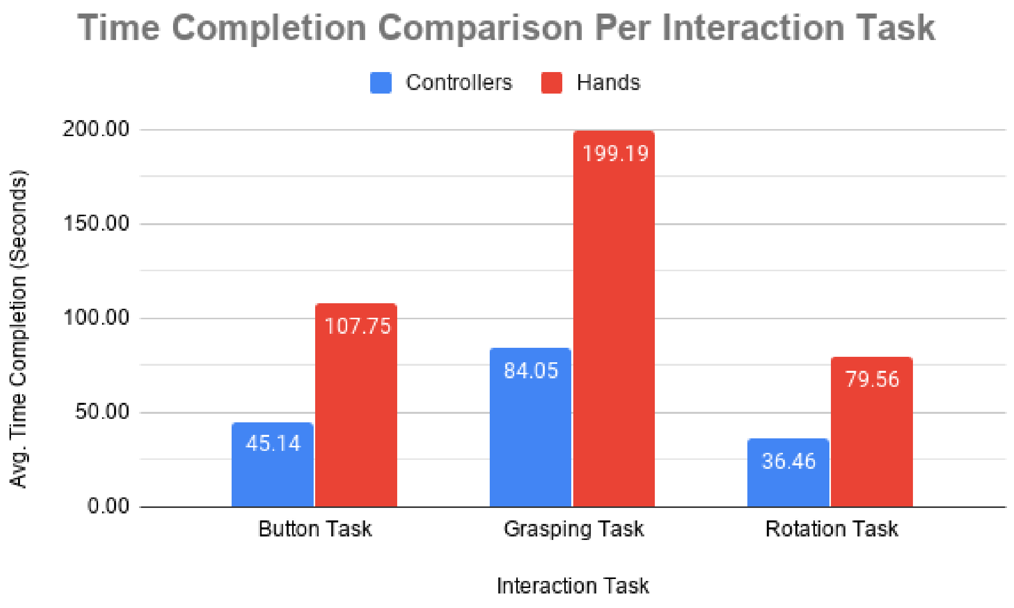


**Figure 7.2:** Time completion comparison per interaction task, showing hand-tracking completion times considerably slower on average.

More detailed information on top of the average completion time value is presented in table (7.1), containing standard deviation, confidence value and number of data entries included.

**Table 7.1:** Number of data entries, average, standard deviation and confidence value for the time completion data per interaction task.

| | **Button Task** | | **Grasping Task** | | **Rotation Task** | |
|---|---|---|---|---|---|---|
| | Controller | Hands | Controller | Hands | Controller | Hands |
| Entries | 36 | 42 | 39 | 23 | 39 | 26 |
| Avg. | 45.14s | 107.75s | 84.05s | 199.19s | 36.46s | 79.56s |
| SD | 9.97 | 38.70 | 34.72 | 72.49 | 14.58 | 32.94 |
| Conf. Val. | 3.26 | 11.70 | 10.90 | 29.63 | 4.58 | 12.66 |

An independent samples t-test seemed to fit as the choice of statistical test on this comparison study within each interaction task, but two requirements must be met:

- The underlying data must be normally distributed for both conditions.
- Variance must be homogeneous between both conditions.

Testing for normal distribution of the data was done by visualising the data in a histogram and normal Q-Q plot, followed by the Shapiro-Wilk test. The Shapiro-Wilk test statistic output for the hand-tracking condition in figure (7.3) identifies the hand-tracking data as not normally distributed with a *p-value = 0.02*, which is below the 0.05 significance level. The same approach was taken for the other two tasks, finding a similar result of one condition not being normally distributed.

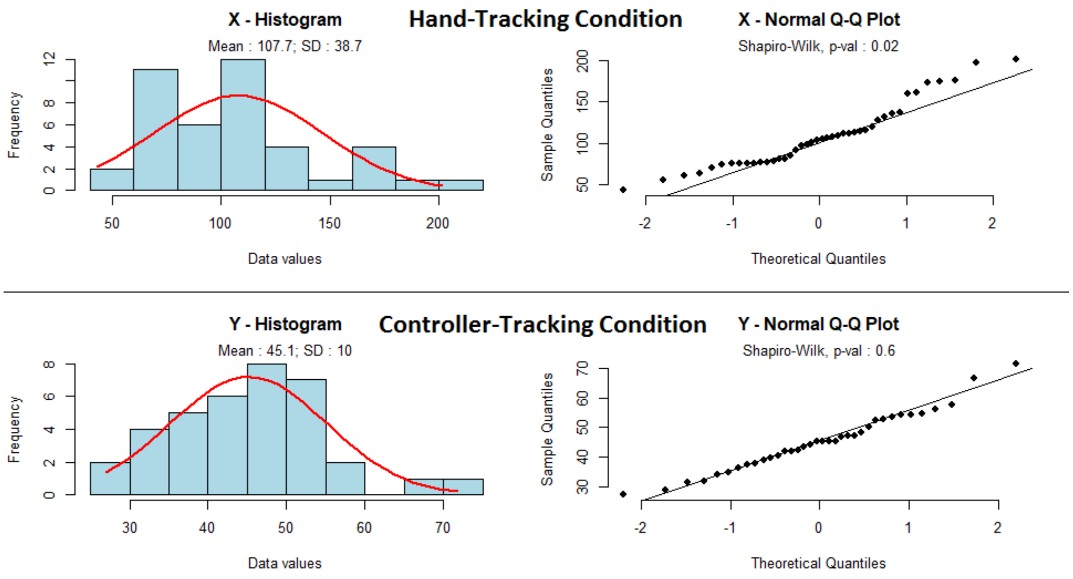


**Figure 7.3:** Histogram, Normal Q-Q plot and Shapiro-Wilk test statistic of time-completion data for the *button-pushing* task, showing that the hand-tracking data is not normally distributed.

The independent samples t-test was therefore found unfit to run on the time-completion data. Instead, the non-parametric Wilcoxon's test was run on the data, finding significant differences between hands and controllers in all tasks (Button-pushing task *p-value = 3.548e-13*, Grasping task *p-value = 0.01368*, Rotation task *p-value = 6.044e-10*). As expected by the comparison plots earlier, it can be concluded that the three tested hand-tracking interactions are significantly slower than equivalent controller-tracking interactions.

### 7.4.2 Interaction - Pinch/Controller Trigger Activations

Outlier detection took the same approach as described in section (7.4.1) on time-completion, resulting in six discarded outliers. Regarding total number of interactions, participants on average used more interactions to complete the three task repetitions, as shown in figure (7.4). More data details on number of data entries, average, standard and confidence value is reported in table (7.2).

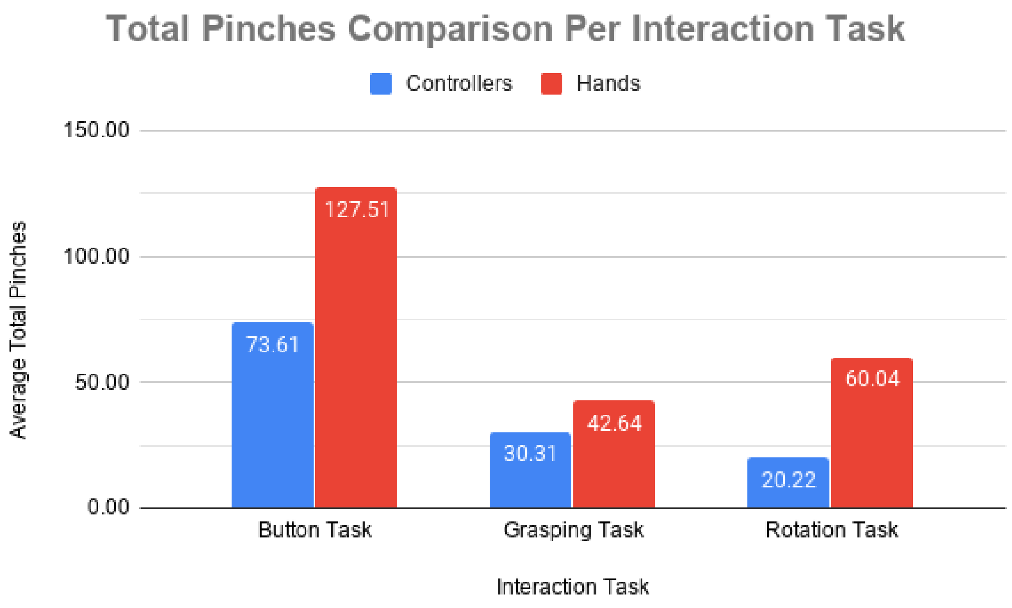


**Figure 7.4:** Average total amount of pinches performed to complete the three interaction tasks.

**Table 7.2:** Number of data entries, average, standard deviation and confidence value for the pinching/trigger activations data per interaction task.

| | **Button Task** | | **Grasping Task** | | **Rotation Task** | |
|---|---|---|---|---|---|---|
| | Controller | Hands | Controller | Hands | Controller | Hands |
| Entries | 36 | 39 | 13 | 11 | 9 | 26 |
| Avg. | 73.61 | 127.51 | 30.31 | 42.64 | 20.22 | 60.04 |
| SD | 20.80 | 50.03 | 7.54 | 12.76 | 14.46 | 30.07 |
| Conf. Val. | 3.26 | 11.70 | 10.90 | 29.63 | 4.58 | 12.66 |

This result makes sense, considering that participants spent more than twice the amount of time completing the tasks with hand-tracking. As mentioned previously, data collection did not report correctly on the number of hit objects for the controller condition, meaning an error rate could not be calculated, which could have produced a more meaningful result. Interestingly, it took three times as many pinch interactions on average for the rotation task, which can be explained by forward-rotation as a problematic hand position. Forward rotation occludes the two middle fingers, meaning the system loses confidence, resulting in deactivating the hand visualisation, as will be further elaborated on in section (8.3).

Testing for normal distribution took the same approach as in section (7.4.1) on time-completion. All conditions and task combinations were found to be normally distributed with p-values > 0.05. The second step for using the independent sample t-test involved checking for homogeneity of variance between both conditions. For the *button-pushing* task, the variance was not found homogeneous, meaning the Welch Two Sample t-test was used to obtain the test statistic *(t(51) = 6.175, p-value = 1.046e-07)*. For the *grasping* task, variance was found to be homogeneous, meaning the standard independent two-sample t-test was used to obtain the test statistic *(t(22) = 2.936, p-value = .00765)*. Welch t-test run on the *rotation* data resulted in the test statistic *(t(29) = 5.2271, p-value = 1.344e-05)*.

This result may be indicative of more errors (as in interaction attempts leading to no intended interaction) occurring with hands to complete a task, but may simply just be indicative of more time requiring more interactions. Without detailed information on the interactions, it can only be concluded that on average, significantly more interactions occur with the hand-tracking implementation, but the nature of those interactions cannot be established beyond speculation.

### 7.4.3 Questionnaires - Demographics, Presence and Preference

For the questionnaires participants were asked to fill out afterward, 10 entries were obtained. Two entries were deleted to comply with GDPR rules as they were under the age of independent consent (16+). Of the remaining 8 entries, participants were in the age range of 17-31 (Average = 25.25, SD = 4.68), all male. These participants were moderately experienced with hand-tracking (4 had tried it a few times before, 2 several times and 2 were frequent users), professional VR enthusiasts (2 used VR frequently, 6 worked with VR) and had higher education of university bachelor (3) and master degrees (4), with one participant not disclosing their educational level.

The six select presence questions (described in measurements section (7.1)) presented to the eight participants, after testing both applications, produced the results shown in figure (7.5).

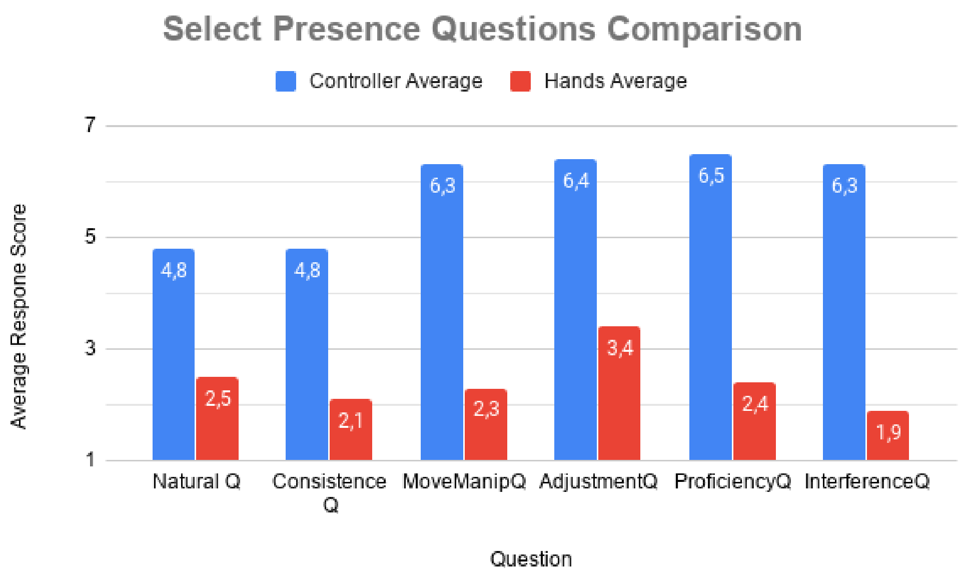


**Figure 7.5:** Comparison of the average score given to the six select presence questions, as answered by eight participants. A score of 1 indicates it was very bad, whereas a score of 7 indicates it was very good.

The contrast is clear, with controller interactions scoring considerably better than hand interactions for all six questions. Of particular note, in the last question on the input method's interference with completing the task, hand-tracking input was found highly interference inducing. Even questions about the natural perception of the input method and consistency between the real and virtual world representation found hands to score considerably worse. In these two questions, one could imagine hand-tracking scoring higher, since it is a direct replication of body movements rather than simulated/trigger activated animations of the virtual hand representation applied to the controllers.

The presence results gave reason for a detailed analysis of the comments provided in three open questions for each task. For each task, participants were asked the three following questions, which were optional to provide input for:

- Can you imagine contexts in which you would prefer using controllers for <insert task>? (Optional)
- Can you imagine contexts in which you would prefer using hands for <insert task>? (Optional)
- Do you have any other comments?

Out of a maximum possible of 72 comments, 40 entries were provided. To group responses together we went through a statistical coding process. One of the researchers read through the 40 entries and gave it categories matching the theme(s) of the comment. A second research member cross-checked the result by providing their own categorisation on instances where an agreement was not reached. Agreement was settled in a joint discussion. Comments could have more than one category applied. As presented in table (7.3), comments were generally regarding the quality of the tracking implementation.

**Table 7.3:** Overview of comment themes in the open comment questions as provided by participants.

| Comment Category | Total Responses |
|---|---:|
| Tracking | 28/40 |
| Precision | 6/40 |
| Realism/Natural/Immersion | 6/40 |
| Near-field Interaction | 2/40 |

Controllers were preferred because it *"does not lose tracking like hand tracking does"*, with similar sentiments shared by other participants in this question. As for preferring hands, comments regarding tracking took the approach of *"if the hand tracking worked well, I would probably prefer it when rotating something"* and *"Depends on the quality of the tracking. If the quality of the tracking was the same for hands and controllers, I would probably prefer hands in any context for pushing buttons"*. With this approach, participants gave a dependency for preferring hands that ties to the hand-tracking technology more so than the interaction they tried. Without being prompted by the question to provide such details, participants still mentioned it in their answer, which can indicate that the tracking quality was a heavy focus for the participant's experience. These results motivated investigating quality of the hand-tracking implementation (see chapter (8)), to obtain exact numbers of how often tracking is lost during usage.

As for the binary preference questions, participants were asked to provide their preference without context, training scenario context, and game context. As seen in figure (7.6), participants preferred controllers, with only the *button-pushing* task suggesting hand-tracking interactions having a potential. While the training scenario context in the *button-pushing* task did change participant answers to a 50/50 split, with only eight data entries, this result is not considered reliable. Game context results were similar, but not reported here as they are not considered the focus of this study.

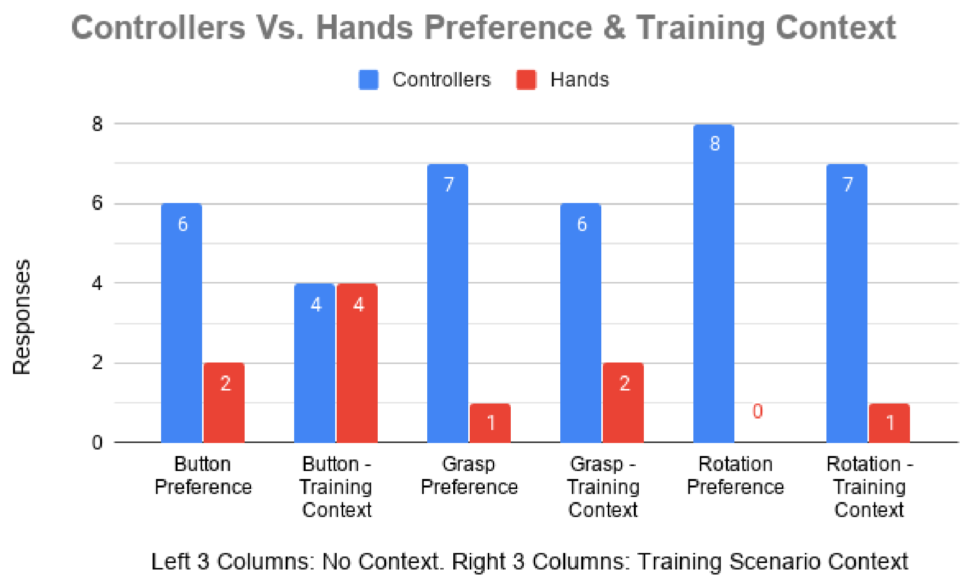


**Figure 7.6:** Preference comparison per interaction task, with no context preference question reported in the first two columns, followed by the preference question, with training scenario context established first, in the two subsequent columns for each interaction task.

## 7.5 Discussion

The study presented here is limited by the independent data treatment enforced by the lack of data, in the sense that the experiment design choice could not be adhered to during analysis. A deeper understanding of the relationship between variables was not an option, resulting in statistical analysis that only concludes general averages. Each subsection of results only confidently concludes a general tendency for the participant data entries for that specific variable. With a variance in participants included for each condition and task combination, comparison runs into sampling issues. Relating the number of pinches/interactions to time completion makes conceptual sense, but during comparison, uncertainty arises when it cannot be established that the data samples come from an equivalent population, resulting in an uncertain conclusion on those variables.

Further study limitations emerged with the remote study configuration in terms of data validity. Without the research team being present during testing, it was not an option to fix errors that would occur during run time, nor was it possible to observe errors unknown to the research team that could explain why some data entries were subject to outliers. Therefore, the study relied on participants figuring out the procedure and interactions on their own, which is subject to human error.

While guidance was implemented in the testing application, it cannot be expected that it covers all problems encountered by participants. Contact information was given to participants, but it requires effort from the participant to reach out. Furthermore, environmental factors could not be controlled, as each participant tried the application in their own surroundings. This is particularly relevant for the hand-tracking technology, which is reliant on lighting conditions, requiring a well-lit room to function [8]. The experience could vary between participants, where one participant may find themselves losing tracking more often than another participant.

Considering the data that emerged from open comments, there appears to be a fundamental problem with the underlying tracking technology that needs to be handled. With a provided dependency that hand tracking interactions would be preferred in the case that tracking stability worked equivalently to controllers, the interaction form itself has become secondary in the preferred choice. This is in line with impressions of the research team over the course of development iterations, where tracking issues were clear. With these considerations in mind, it would not be unreasonable to consider this study as an investigation of two different tracking implementations, rather than a comparative view of interaction tasks.

Regarding the designed interactions, inside the *button-pushing* task participants were asked to press a trigger button on the controller to press the virtual button. This stands in contrast to the hand interactions, where the user merely had to push the button by moving their hand and fingers. Therefore, the two interactions can, in retrospect, not be considered equivalent. Future studies with interaction tasks in focus, including a *button-pushing* task, should consider dropping the trigger activation for controllers to create a better comparative understanding. To press a button in the scene, the participant would have to hold down a button on the controller which would animate the hand to point a finger.

## 7.6 Conclusion

After analysing the results, the study can no longer be considered a comparison between equivalent interaction tasks, but should instead be thought of as a comparison between two tracking implementations (hand-tracking vs. controller-tracking). With that in mind, the current hand-tracking implementation on the Oculus Quest averages as a significantly slower input method requiring more interactions to complete similar tasks. Hand-tracking stability is suggested to be a fundamental technological hurdle to overcome before real comparisons can be made between similar interactions. A fundamental challenge that appears to affect presence and preference results, creating stark contrasts where hand-tracking interaction score significantly worse on all investigated measures. However, the validity of the results falls off, due to the remote testing setup introducing sampling- and environmental issues that could not be controlled. Nevertheless, the *button-pushing* interaction is suggested to be the only one of the three tested interactions with potential for use.

# Chapter 8

# Study Three: Hand-Tracking Stability - Design & Implementation

This chapter describes the third and last evaluation of the paper. The prototype was designed to evaluate the loss of tracking for the hands, as the previous study indicated this was an issue in the current state of hand tracking. The interaction tasks from study two carried over into the prototype built for this study, digging into hand-tracking stability alongside exploration of whether the participants felt they would prefer controllers after executing the three interaction tasks with hands.

## 8.1 Motivation

Motivation for this study was based on the results of the previous evaluation. The qualitative data, extracted from the open comment entries, showed a high tendency towards participants identifying tracking issues as the main reason behind the lack of quality in hand-tracking. Several participants mentioned that they did not believe hand-tracking could compete with hands until this was improved. Therefore we thought it would be interesting to research this further, and learn more about exactly how the tracking performs in its current state. This lead to study design with a focus on gathering data about the loss of tracking while performing the three different interaction tasks explored in study two. This lead to the following two research questions:

- *In the current state of Oculus Quest's hand-tracking, how often is tracking lost during the interaction?*
- ***Sub-Question:*** *Are there interactions where hand-tracking is found as a preferable alternative to controller-tracking?*

## 8.2 Study Design

The design of this prototype was based upon the previous application, as we wanted to continue using the different interactions made, to ensure the results of this study would not be subject to different interaction. There were a few changes to the design based on experience from the first study, but a lot of what worked well was brought into this design. A key difference was that it was decided to collect every piece of data through the application. Therefore application consisted of two scenes; one for retrieving demographic data and one for performing the test.

As it was known the study would be performed online, the prototype was designed very strictly in regards to allowing the participant to make any decisions. The participant would only have one option available at all times, and the amount of time they were allowed to be specific places would be fixed. It was also ensured that the participant could only make one entry per device in the study, to prevent them from sending corrupt data or making multiple entries.

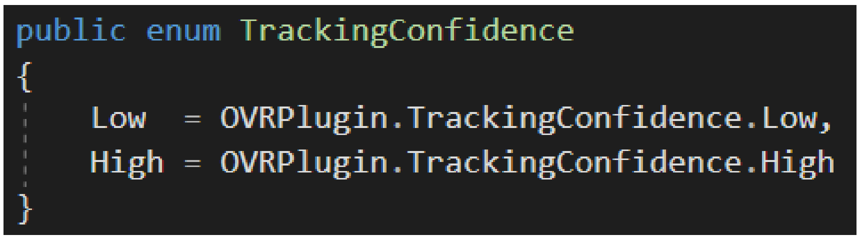
Unlike study two where the application was split into different scenes, the prototype built for this study took place in the same scene but split into multiple interaction zones that handled activation of relevant parameters (physics) for the interaction tasks. Initially the idea was to allow participants to roam freely in the application, but further deliberation on study design lead to constriction of participants, to ensure equivalence in test conditions. A time-limit per interaction task was introduced, to ensure the comparison of data entries would be valid. Ensuring a controlled randomised order was not an implementation option, but instead a decision was taken to randomise the order of interaction tasks, to partially handle order effect influences on the data.

## 8.3 Tracking Stability

When working with hand-tracking, there are times when the cameras do not recognise the position of the hands. There are two different variables used by the Oculus SDK to detect this: 1.) confidence level and 2.) lost tracking. For each interaction there was a total of four integers to count each time confidence and tracking was lost, meaning two counters for both the left and right hand.

### 8.3.1 Loss of Confidence

Confidence level of the hands is based on a variable of how sure the algorithm is that it can see that hands correctly. This is determined by an enum of two confidence levels; low and high, as presented in figure (8.1).



```
public enum TrackingConfidence
{
    Low  = OVRPlugin.TrackingConfidence.Low,
    High = OVRPlugin.TrackingConfidence.High
}
```

**Figure 8.1:** Code snippet of the tracking confidence enum.

High confidence means that the hands will be rendered and is visible for the participant, whereas low confidence means the visual hand representation disappears. When losing confidence, any virtual objects currently grabbed will not be dropped, but the visual representation of the hand will be set to invisible. As per internal testing with the prototype, confidence loss is expected to occur under the following conditions:

- Moving too fast - either with hands or shaking the headset too much.
- Poor lighting conditions - hand-tracking is recommended for well-lit rooms to improve tracking conditions [8].
- Moving one hand towards the other, making it difficult for the computer vision algorithm to differentiate the hands.
- Occluding fingers from the field of view of the cameras.

Specifically for *Grasping* and *Rotation* tasks:

- Forward rotation - occluding the two middle fingers from the camera view.

The main issue with confidence tracking was that it was not possible to adjust the confidence decision or get an idea of how confidence is decided by the system. Despite the lack of knowing the exact confidence level required for the device, internal trial and error testing deemed the counter for lost confidence as reliable.

### 8.3.2 Loss of Tracking

The other variable to determine tracking stability is the loss of tracking, which can be perceived as when the camera cannot see the hand. As per internal testing, this happens under two conditions:

- When the hands are placed outside the tracking volume of the cameras.
- Occluding a hand entirely with another hand.

Both tracking and confidence were measured for each hand. One issue in measuring the loss of tracking is, that close to every time loss of tracking would happen, it would also trigger the loss of confidence briefly before losing tracking. This was something worth noting when analysing the data of these two variables.

### 8.3.3 Pinching Objects and Pushing Buttons

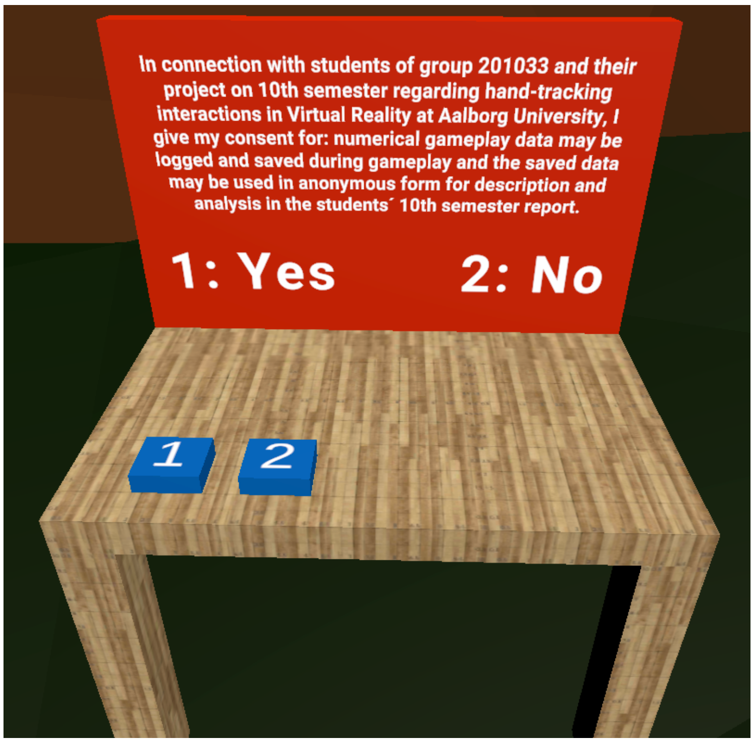
In two of the stations (*Rotation Interaction* and *Grasping Interaction*) we counted each time a participant would make a wrong and a correct pinch. A wrong pinch meant that the participants pinched outside of a grabbable object, meaning they did not grab anything, whereas a correct pinch means that the participants pinched inside of a grabbable object. For the third station (*Buttons Interaction*) we tracked the number of button presses, which would count up every time the participants pressed a button.

## 8.4 Scene Implementation

For this application there was a demographic scene and a test scene. The participant would start in the demographic scene and complete all of the questions before being able to continue to the test scene.

### 8.4.1 Demographics Scene

Upon starting the application, the participant would be placed in the demographic scene. In this scene the participant would press buttons to answer questions about themselves, and give consent, allowing us to use their data for our study. The participant is placed in front of a table, with a board behind the table displaying the text or questions, as seen in figure (8.2).


In connection with students of group 201033 and their project on 10th semester regarding hand-tracking interactions in Virtual Reality at Aalborg University, I give my consent for: numerical gameplay data may be logged and saved during gameplay and the saved data may be used in anonymous form for description and analysis in the students´ 10th semester report.

1: Yes 2: No

**Figure 8.2:** A screenshot of the consent text inside the demographics scene.

Buttons appear on the table according to how many options the question has (e.g. showing two buttons when asked which hand is their dominant hand).

It was chosen to let the participant input their answers with big buttons since we assumed big buttons in a hand-tracking application would easily signify to the participant to simply push down on the button, thus hopefully removing any confusion as to how the participant interacts and responds to the questionnaire. We chose big buttons in order to minimise the risk of the participant pressing a button by mistake.

### 8.4.2 Test Scene

Upon finishing the demographics scene, the participant would be sent to the test scene. This scene consists of three stations, one for each interaction, and a station in the middle where the participant would be introduced to the test scene. A single button would appear in front of the participant as shown in figure (8.3), which transports them to a station at random when pressed.

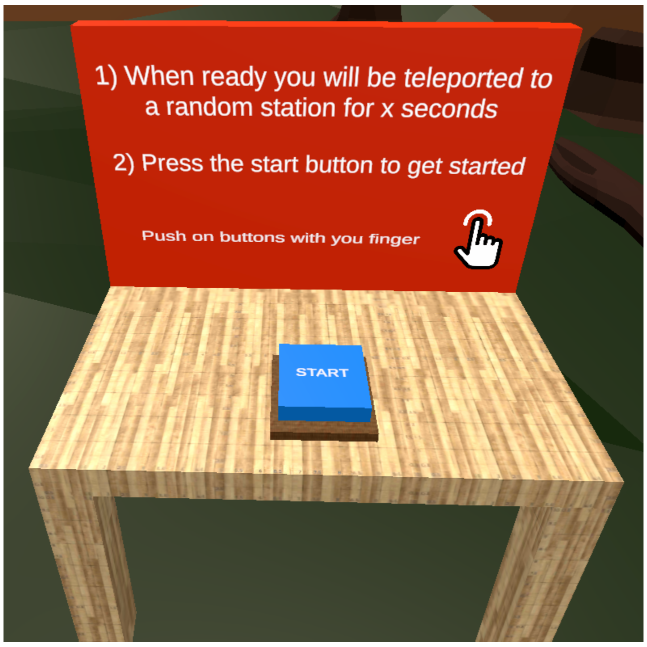

1) When ready you will be teleported to a random station for x seconds
2) Press the start button to get started
Push on buttons with you finger
START

**Figure 8.3:** Screenshot of the middle station in the test scene, which transport the participant to the interaction tasks upon interaction with the button.

The participant would be placed at a station for 90 seconds until being transported back to the middle. After completing each station, a yes/no question would prompt in front of the participant regarding whether they preferred using controllers for the experience. While participants would not get to complete the experience with controllers, it was expected that participants would be experienced with using controllers and therefore able to make a valid decision.

## 8.5 Interaction Implementation

The interactions were implemented almost identically to the second evaluation. We did some minor adjustments to each interaction so they could be included in the same scene, but generally it was closely related to their previous task. The main difference between the interactions in this prototype and the previous prototype was that it was not required for the participant to complete any specific task in order to finish. The participant could freely play around with each interaction as they pleased. The three interaction stations were *Button Interaction*, *Grasping Interaction* and *Rotation Interaction*.

### 8.5.1 Pushing Buttons

For the *Button Interaction* seen in figure (8.4), the size of the button matrix was reduced to 4x3, from 5x5 for the previous prototype. We also increased the size of the individual buttons hoping to make the buttons easier to hit when pushing. The buttons would switch between being black and white when pressed, just as they did in the second prototype, but for this prototype we added a musical loop to each button. This way the participant could occupy themselves during their time at the station by playing around with different sounds.

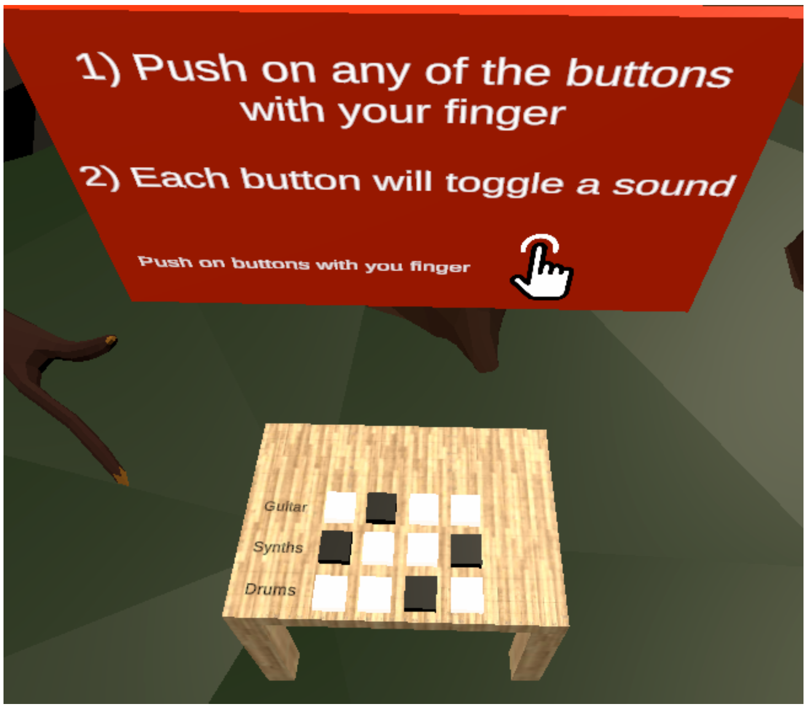

1) Push on any of the buttons with your finger
2) Each button will toggle a sound
Push on buttons with you finger
Guitar
Synths
Drums

**Figure 8.4:** Button pushing stations for the third prototype, where interaction with the buttons initiated musical loops.

### 8.5.2 Grasp and Release

For the *Grasping Interaction*, seen in figure (8.5), another row of weights was added, in order to give the participant more weights to grasp and place on the scales. A button in the middle of the station would reset the positions of all the weights, in case the participants should place all the weights out of reach for the participant. Besides the removal of the submission check for task completion, the weight functioned the same way as in study two.

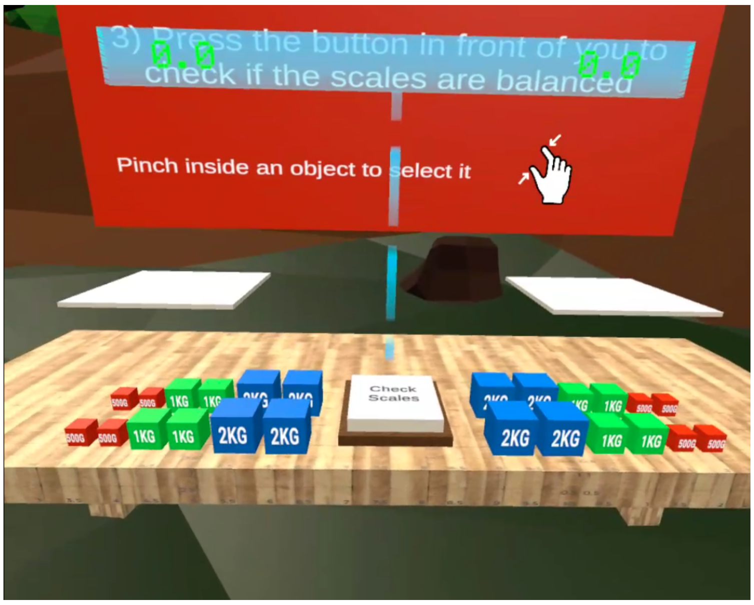

3) Press the button in front of you to check if the scales are balanced
0.0
0.0
Pinch inside an object to select it
Check Scales
500G
1KG
2KG

**Figure 8.5:** Screenshot of the grasp and release station in the third prototype.

### 8.5.3 Rotating Objects

For the *Rotation Interaction* we added an additional four objects, amounting to five objects that the participant could rotate, instead of just having one object as we did in the previous prototype. Each object was of a unique shape, and behind each of them, a holographic copy of them was present as shown in figure (8.6).

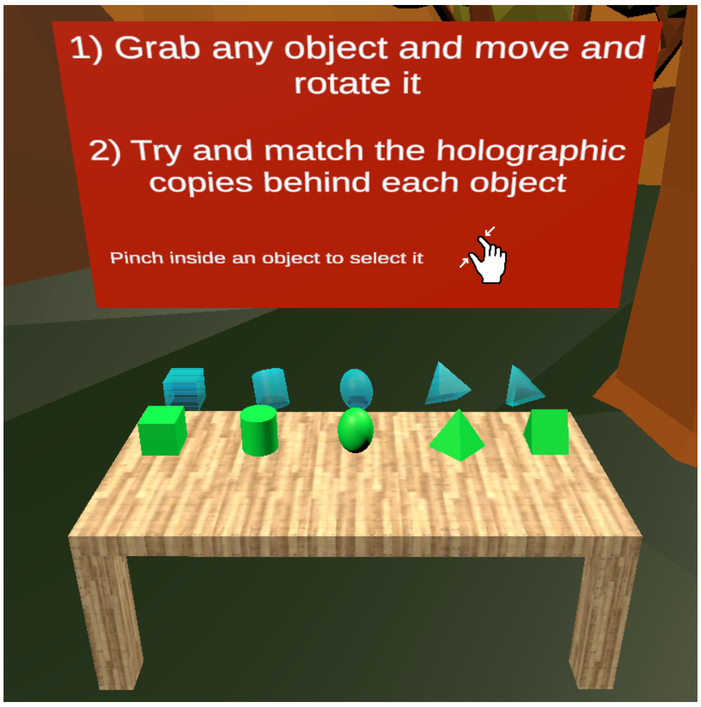

1) Grab any object and move and rotate it
2) Try and match the holographic copies behind each object
Pinch inside an object to select it

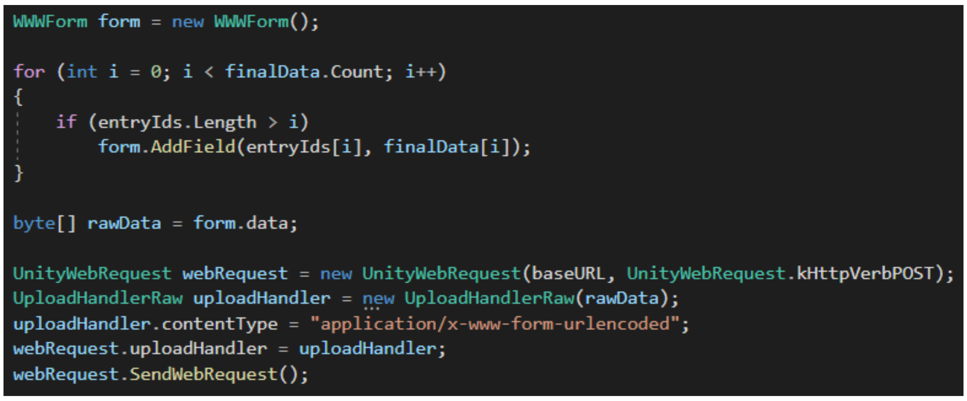
**Figure 8.6:** Rotation station with five interactable objects and five holographic objects with slight changes in rotation.

The objects were not rotated from the beginning but the holographic copies were. We asked the participants to try and match the objects with their holographic counterparts, but as mentioned before, the participants were not required to do this. This was solely in hopes for the participants to be able to pass the time while at this station.

## 8.6 Data Logging Implementation

For data logging it was decided to use a google drive implementation to send answers to a questionnaire form, which could then be transformed into a spreadsheet using Google Sheets. This was done since we only had one application the participants used, thus we did not need to connect data from several places to each other.

Data logging was done through a web request. We had gotten an entry ID for each of the fields in the questionnaire and connected it to the data of our code, so that we could send all of the data at once, upon the participant completing the application. In order to store the data and save it to Google Forms, we used an open-source project from GitHub [19]. The code for sending the data to our Google Forms page can be seen in figure (8.7). In short, the code simply takes in a list of data, iterates through the list one by one, in chronological order in relation to the Google Form page we created and adds each data entry to our Google Form page.



```
WWWForm form = new WWWForm();

for (int i = 0; i < finalData.Count; i++)
{
    if (entryIds.Length > i)
        form.AddField(entryIds[i], finalData[i]);
}

byte[] rawData = form.data;

UnityWebRequest webRequest = new UnityWebRequest(baseURL, UnityWebRequest.kHttpVerbPOST);
UploadHandlerRaw uploadHandler = new UploadHandlerRaw(rawData);
uploadHandler.contentType = "application/x-www-form-urlencoded";
webRequest.uploadHandler = uploadHandler;
webRequest.SendWebRequest();
```

**Figure 8.7:** Code snippet showing how the data is processed and sent to a Google Forms page.

# Chapter 9

# Study Three: Hand-Tracking Stability - Evaluation

In this study we sought to obtain a better understanding of the tracking stability of the Oculus Quest when tracking hands. For this experiment, four dependent variables are measured; lost tracking, lost confidence, amount of pinches, and participants' preference between hand-tracking and controllers.

## 9.1 Measurements

Before each participant began the experiment, they were required to fill out a questionnaire about their general demographic data. The questionnaire was integrated into the VR space as described in section (8.4.1) and involved six questions. Firstly the participant was asked for their general consent, followed by questions about their age, gender and dominant hand. Participants were also required to input their previous experience with hand-tracking in VR and if they had a background in IT. Data was collected anonymously, with no method for the research team to trace the data entries to a specific person.

In the experiment, measurements revolved around relevant data for tracking. While the participants were doing different interactions, data would be logged for both hands regarding loss of tracking and confidence per interaction zone. After each interaction task, participants would be asked *"Would you prefer using controllers over hand-tracking for the task you just did?" (Yes/No)*.

## 9.2 Setup

Due to the Covid-19 situation, testing was done online over a duration of five days. The application was released on SideQuest as our previous application, as it had gained a lot of downloads (700+) which provided a reason to believe that a high number of participants could be recruited. A description with information about the study was attached to the application description on SideQuest, which allowed anyone to be slightly informed about the purpose of the application. Unfortunately online remote testing did not allow us to supervise the tests, as it would be tested by random people around the world.

The participant was guided through the application by text and constraints, which was put in place to ensure the participant would complete the study in the desired manner. Furthermore all the data was sent collectively upon finishing the application, to ensure that all relevant data had been recorded before it was sent, avoiding a fractured data set as was the case from study two.

## 9.3 Participants

There was a total of 34 participants in this evaluation, 30 male, 1 female, and 3 others. The age of the participant varied from 11-90. (Average: 24.62, SD: 16.92). Of the participants, 6 were left-handed and 28 was right-handed (Left Hand: 17.6%, Right Hand: 82.4%). When asked about their experience with hand tracking, 2 (5.9%) had none, 8 (23.5%) had little while 13 (38.2%) had some and 11 (32.6%) had a lot, which shows that the participants, in general, were experienced users of hand tracking, while a few were still novice users. The participants were asked if they had a background in IT, which resulted in 9 (26.5%) answering yes and 25 (73.5%) answering no.

## 9.4 Procedure

Since the researchers were not present during the testing procedure, the application was made with all the necessary information and selections, enabling the participant to conduct the experiment in their own setting. The participant would begin the test in a locked environment, where they had to fill out demographic questions about themselves. Upon completing every question, they would be allowed to load into the real test. In this part, the participant would be randomly teleported around to the different interaction stations, and stay there for a fixed time duration (90 seconds). The participant could try out this application as many times as they wanted, but it would only collect data the first time, to ensure the equivalence of data conditions between entries.

## 9.5 Results and Analysis

In this section the results of the study will be presented. Four subsections describes the results regarding tracking stability, interaction details, controller preference and preference in relation to hand-tracking experience. The analysis primarily consists of descriptive statistics.

### 9.5.1 Tracking Stability

As mentioned previously for tracking, two different kinds of loss were identified; loss of tracking and loss of confidence. To better compare the data, it was decided to examine the data to see how often it happened over time and not just a total value.

Taking both hands into consideration, the data presented in table (9.1) for lost tracking and lost confidence, shows an average loss of tracking to be 13.48 times per minute over the course of the entire application, which on average is once per 4.45 seconds. There is an average loss in confidence 14.43 times a minute, which on average is once every 4.16 seconds. Average per minute from the start position was computed based on the average time all participants spent in the zone (26.22s), whereas each interaction was based on the knowledge that they spent 90s in those zones. Separating the data into left- and right-handed was an option, since the participant information was collected in the demographics scene, but found unfeasible for statistical analysis due to discrepancy in number of data entries (left-handed people: 6, right-handed people: 28).

**Table 9.1:** Lost tracking and lost confidence data averaged over a minute for each station, with the total value representing over the course of engagement with the application.

| Interaction | Lost Track. Avg/Min | Lost Conf. Avg/Min |
|---|---|---|
| Start Position | 16.56 | 12.38 |
| Button Position | 13.43 | 18.73 |
| Grasp Position | 12.94 | 13.63 |
| Rotate Position | 11 | 13 |
| **Total** | 13.48 | 14.43 |

When viewing the individual interactions, the data shows for both versions of tracking loss that pushing buttons had the biggest fallout of tracking. Especially the confidence level is remarkably larger for pushing buttons and shows that this happened on average 18.73 times every minute, which is once every 3.2 seconds. In comparison, the loss of confidence for the rotation interaction is 13 times every minute, which is once every 4.6 seconds. This shows a 35.9% difference between confidence loss in these two interactions. The loss of tracking also shows a noticeable difference, where pushing buttons have a 20% higher loss of tracking than rotating objects.

The loss of confidence is considered the most relevant of these variables, as this is what is most likely to happen during an interaction. This distinction is based on the knowledge that lost tracking is identified in situations where the hands is outside of the tracking volume, or camera view, and therefore unlikely to be part of the interaction, but merely describing human behaviour. When tracking is lost, loss of confidence also counts up just before it. With this knowledge in mind, the lost confidence values presented in table (9.1) were calculated by subtracting the lost tracking value from the total lost confidence value. The value that remains, and reported here, is a deemed a better indicator of tracking issues that only occurs during interaction with the virtual objects.

### 9.5.2 Pinching and Button-Pushing Interactions

To gain insight into the participants' engagement patterns throughout the application, it was chosen to log the number of interactions as well. For *grasping* and *rotating* it was logged how many correct and wrong pinches were made, where wrong pinches were defined as pinches that did not hit anything. For *button-pushing* it was logged how many times it happened in total, as there were no correct/false buttons presses to differentiate between.

Some of the data measured had noticeable errors, and upon trying to recreate them it was discovered that when holding both hands on top of each other and slightly moving them, it was possible to make the tracking confused. Furthermore, measurements of pinching could be corrupted when the participant was pinching and holding an object with one hand and then tried to pinch the same object with the other hand simultaneously. This would make the counter for pinches count up once every frame, quickly logging a lot of false entries. When analysing the data, this was taken into consideration and several values were removed from the data set as they were deemed to be severe outliers.

**Table 9.2:** Pinching and button press data over the course of 90 seconds participants were engaged in the experience.

| | **Button Presses** | **Grasping Task Pinch** | | **Rotation Task Pinch** | |
|---|---|---|---|---|---|
| | Amount | Correct | Wrong | Correct | Wrong |
| Entries | 33 | 29 | 31 | 18 | 31 |
| Average | 60.79 | 32.07 | 29.87 | 33.17 | 24.65 |
| Std. Deviation | 19.14 | 13.96 | 22.09 | 16.11 | 13.96 |

In the table (9.2) these data are presented, and when comparing *grasping* and *rotation* it shows that these data are very similar. In total, *pushing buttons* averaged 60.79 activations, *grasping* averaged 61.94 interactions (correct + wrong values) and *rotation* averaged 57.8 interactions. This shows an even spread of interactions throughout the applications and with the 90 seconds spent at each location, the participants have on average performed around 0.67 interactions per second. Therefore it is considered that participants were actively engaged with interactions during testing. The large standard deviation values can be considered a reflection of variant human behaviour, where some people would be highly active whereas others took a slower approach.

### 9.5.3 Controller Preference Comparison

After each interaction, the participants were asked if they preferred using controllers (it was assumed that anyone participating would own an Oculus Quest and therefore at least have some experience with controllers). This data was collected as a follow-up to the previous study, where controllers and hand-tracking were directly compared, but lacked data entries for a confident conclusion.

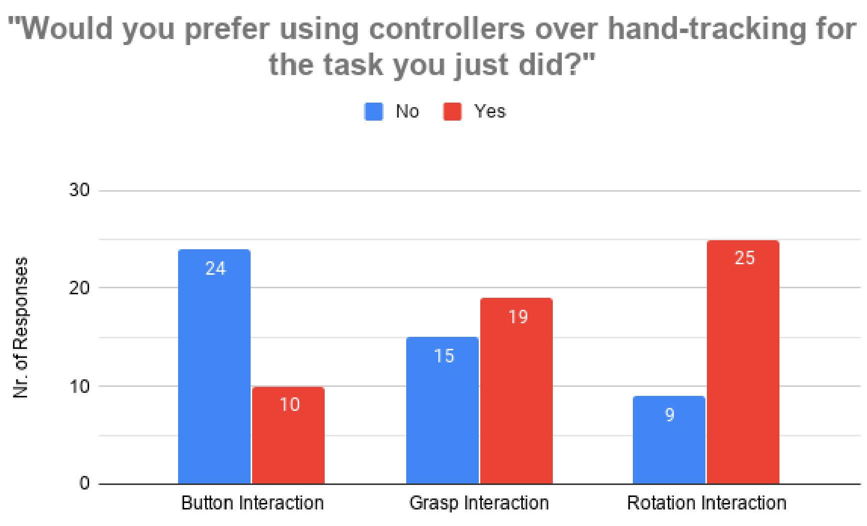


**Figure 9.1:** Participant responses to preference question posed to them after each interaction scene.

The graph presented in figure (9.1) shows a tendency that participants preferred using **controllers** over hands mainly at the *Rotation Interaction* (25/34 = 73.5%) and the *Grasp Interaction* (19/34 = 56%), whereas 24/34 (70.5%) of participants preferred **hands** over controllers for the *Button Interaction*. An interesting note to take from this data is that it is similar to our previous study (chapter 7) in regard to which input method participants preferred. The data implicate that button-pushing, in general, seems to be the interaction most preferred by participants to interact with using their hands, making button pushing the interaction with the most potential out of the three interactions.

### 9.5.4 Hand-Tracking Experience and Preference

We asked all participants to input their previous experience with hand-tracking in order to identify relation between experience level and interaction method preference. Only two participants had no experience at all with hand-tracking while the remaining 32 had little to a lot of experience with hand-tracking. Grouping the participants' experience level with preference resulted in the data table (9.3) seen below.

**Table 9.3:** Table showing participants answer to the question (*"Would you prefer using controllers over hand-tracking for the task you just did?"*) in accordance with their hand-tracking experience level.

| | **Button Task** | | **Grasping Task** | | **Rotation Task** | | **Total** | |
|---|---|---|---|---|---|---|---|---|
| Experience | Yes | No | Yes | No | Yes | No | Yes | No |
| None | 0 | 2 | 1 | 1 | 1 | 1 | 2 | 4 |
| Little | 2 | 6 | 4 | 4 | 4 | 4 | 10 | 14 |
| Some | 3 | 10 | 6 | 7 | 9 | 4 | 18 | 21 |
| A lot | 5 | 6 | 8 | 3 | 10 | 1 | 23 | 10 |

Looking at table (9.3) there is a tendency for participants to be more inclined to prefer controllers over hands, the more experience they have with hand-tracking. In total, all participants with a lot of hand-tracking experience (11 participants) preferred to have used controllers in 23/33 (70%) of cases, when looking at all three interactions combined. For none, little and some experienced participants, we see a contrary result when looking at their overall data, with these participants slightly preferring using hands. For none experience participants (2 participants), 2/6 (33%) total responses were given for controllers. As for little experienced participants (8 participants), 10/24 (41%) total responses preferred controllers. In the some experienced group (13 participants), 18/39 (46 %) responses took preference for controllers. As hand-tracking level increases, so does preference for using controllers.

Looking at experienced participants ("A lot" row) data for the individual interactions, we see that 10/33 total responses preferring hands, 6 of those responses was given in the *Button Interaction*, 3 were given in the *Grasping Interaction* and only 1 response was given in the *Rotation Interaction*. These results show that although experienced participants are less inclined to prefer hands, there still was a similar tendency for participants to prefer hands over controller when looking at *Button Interaction*.

## 9.6 Discussion

Prior to conducting the study, it was discussed which interaction was expected to yield the highest amount of tracking problems, and the result of that discussion was that it would be grasping or rotation, as those required the hands to move in a somewhat complicated pattern, compared to pushing buttons. When the data was collected, it was surprising to see that this was not the case at all, and pushing buttons actually had more tracking problems than any other interaction. The reason for this is hard for us to know for sure, as we, unfortunately, did not have the opportunity of supervising the participants while they performed the evaluation. To gain a deeper insight into this, it could have been worth distinguishing between what hand was used to do the interactions. This could have given some context to these numbers, as we would have been better able to identify if some of the tracking loss happened due to an unused hand in the periphery of the cameras.

This also brings forth another discussion point; what exactly do the tracking loss and confidence value mean? This is not defined by Oculus and was therefore something we had to test internally, to gain knowledge of these variables and their importance. The loss of tracking happens as a result of the hand not being perceived in any way by the cameras, which would often be a result of the participant moving them out of the tracking space. This happens quite often with natural movement, and it is therefore expected this variable will count up. In comparison the loss of confidence happens when the cameras perceive the hands but are uncertain about their position or rotation. This means that the confidence level is what causes the loss of tracking which is visible during interactions.

Dialling back to the surprisingly high number of tracking issues with button-pushing, there are a variety of issues that could have lead to those numbers. Since the buttons were horizontally placed, forward rotation of the hand, occluding the front fingers, could have been an issue. Placing buttons in a vertical position, replicating buttons on a wall or machine, could alleviate this issue as more tracking information is available to the camera. The participant could also have moved around more in this scene, since it was playing music upon interaction. Or, potentially the participants only used one hand for pushing the buttons, which left the unused arm in a position swinging in and out of tracking view. However, without exact specification of which hand was used for interaction, or video recordings, the reasoning cannot be established concretely.

Participants showed a high preference in using their hands for pushing buttons, as this was the only interaction where the majority of participants did not prefer using controllers. These results are a bit ambiguous, as it shows that the interaction where tracking performed worse was also the interaction participants most often preferred. This was something to consider, as the vast majority of the participant from the second evaluation had mentioned the tracking problems as the main reason for why hand-tracking was not as good as controllers.

This leads us to a different hypothesis towards the interactions or has at least created a new variable to the equation, which is immersion. Of the three interactions tested, two of them use pinches to replicate the act of grabbing, whereas pushing buttons is an actual representation of the real world. It is likely that when people have to add new layers to an interaction that does not feel real, it creates a disconnect between the real feeling of doing it, thus removing some immersion from the interaction.

## 9.7 Conclusion

This study investigated Oculus Quest's hand-tracking in three direct object interactions, namely *button-pushing*, *grasping* and *rotation*. On average, Oculus Quest's hand-tracking runs into losing tracking confidence 14.43 times/minute for users who are actively engaged in interacting with virtual objects at 0.67 interactions per second. Confidence loss results in the virtual hand representation being disabled, although grabbed objects can still be interacted with. Despite tracking issues, 24/34 (70.5%) of users would prefer using hand-tracking over controller-tracking for *button-pushing* interactions, indicating this interaction form holds potential as an alternative interaction method, unlike the manipulation interactions (*grasping* and *rotation*). A result that still holds for participants with a lot of hand-tracking experience, who had a higher tendency for preferring controllers (23/33 = 70.5%) over the tested hand interactions, comparatively to lesser experienced participants (none: 2/6 = 33%, little: 10/24 = 41%, some: 18/39 = 46%).

# Chapter 10

# Discussion

In this chapter general discussion points are discussed, including sections on issues arising with remote studies, novelty effect and implementation of hand-tracking experiences.

## 10.1 Remote Studies

Most of this paper is based on remote studies, where the researches have not been able to attend and view the tests. This causes a lot of potential external variables to have a chance of occurring, with very little chance of identifying and handling them. This results in the data, in general, being less trustworthy, as people can either directly or indirectly influence the data badly by purposely trying to break out of the conditions or mistakenly trying to do the wrong thing due to lack of guidance.

An example of the lack of interference from our part is noticeable in the questionnaire from our third evaluation. One participant had put 90 years old as their age, which is significantly higher than the next highest age of 67 of all our participants. This entry skews the average age and standard deviation a bit but it is impossible for us to tell whether that was a correct age or just someone putting in the maximum age for fun. This issue could have been avoided to a certain degree by implementing various validation techniques. We could have randomised the answers, or included a question that asked them if they answered truthfully to the previous questions. In general though, this is an issue that is very hard to control when doing remote studies.

Another negative aspect of doing remote studies was the need for having an online database to save and retrieve our entries. When doing local studies we only relied on a single device to keep track of our data which is easily implemented by saving data entries to a notepad file on the device. By having to conduct our studies remotely, we were forced to put more effort and time into how to save and process our data than we otherwise would have done, if we had conducted solely local studies.

The enforced remote study situation also limited methodological approach that could be undertaken with the project. Study two and three focused on context of VR training scenarios, which could have been better explored if the research team were able to replicate a real world training scenario at a workplace.

## 10.2 Novelty Effect

Another interesting topic regarding hand-tracking is the novelty effect. Hand-tracking is a relatively new technology on the regular consumer spectrum, which a lot of people can see potential in using for many scenarios. Being able to see a 3D model of your hand in real-time while wearing a VR headset is quite amazing for a lot of people, and this could have been a potential cause of some people preferring hands over controllers. We did experience that the majority still preferred controllers, and therefore it is possible to wonder if the novelty effect did play a role.

When digging a bit deeper into this, some signs of the novelty effect does show up. In the third study where participants were asked about controller vs. hand-tracking preference, the data showed that people with a lot of experience with hand-tracking were most likely to prefer controllers. This could arguably be the result of people with experience not being influenced by the novelty effect anymore and therefore being more influenced by the flaws of hand-tracking. If circumstances allowed us to test locally and under more controlled environments, we could have an equal number of participants with and without hand-tracking experience in order to properly evaluate how big of an impact the novelty effect has on users' general perception of hand-tracking.

## 10.3 Implementing Hand-Tracking

The Oculus implementation of hand-tracking proved rather difficult to use for development, which had a large impact on how this paper evolved. It was initially planned to work on a much broader scale in terms of developing functionality, but ended up being more about handling all the potential errors of using this tracking. A simple example is when the participant is holding an object and the cameras lose tracking of the hand - what happens to the object? And how is this played out in interactions with consistent movement when the hand suddenly is perceived incorrectly? There is a lot of consideration and work which had to go towards handling these scenarios.

Another large issue in the current implementation is the positions of hands that simply can not be tracked. If hands are put into positions where your eyes cant perceive your fingers, then the cameras in that position most likely can not do it either. In reality you can still move your fingers or hold on to stuff though, but this is not an option with hand-tracking. This basically means that certain interactions are close to impossible to perform, as you will have to move your hand into some of the patterns which will not get recognised.

In regard to developing with the Oculus SDK, there was a lack of transparency towards variables and methods, which created a barrier of getting into developing with it. To correctly implement it we were forced to use forum posts and videos to collectively fix all the mistakes in the standard implementation. Further when developing, we had to spend a large amount of time working through their standard implementation to add simple functionality and variable for changing it at runtime. This took a lot of focus away from spending time working on the evaluations and general development, which we believe could have helped improve our own implementations.

## 10.4 Hand-Tracking Technology

A conclusion that is brought forward throughout our second and third studies is that hand-tracking on the Oculus Quest is not working well enough to be usable yet. In that regard it could have been interesting to further research other available implementations for tracking hands in VR (Leap Motion infrared imaging sensor, digitally enhanced gloves), and learn about their performance. This could provide a deeper insight into the future of this kind of tracking and what to expect from it if the performance increases. One of the main concerns with using the cameras on the headset as a tracking device is the field of view. This is currently a limitation for doing a lot of interactions, as everything has to be done within the relatively small tracking zone.

# Chapter 11

# Conclusion

Through three developed hand-tracking prototypes, this project investigated the default hand-tracking pointing implementation of the Oculus Quest, hand-tracking vs. controller tracking in direct object interactions and hand-tracking stability.

Regarding pointing with Oculus Quest's hand-tracking, enhanced by Bendcast and a permanently visible ray, does not result in the usual gain of improved time-completion, but does keep a higher error rate as expected. Therefore it is suggested that the hand-tracking input method inhibits time-completion, by raising the lower limit that can be achieved. Based on 24 participants, a curved ray cast visualisation combined with snapping to the target is preferable to a linear ray, as it provides continuous feedback in the re-targeting process. The default pointing method by Oculus does feel unnatural to use, but opens for interaction possibilities with both hands as one would expect from a mouse, or controllers, signifying the trade-off related to this implementation.

Hand-tracking comparatively to controller-tracking interactions are slower and more error-prone in similar tasks. Comments would suggest that a direct comparison of interactions would require equivalence in tracking stability between the two input methods, which the third study concluded is not the case. As hand-tracking loses confidence 14.43 times/minute during active interaction with virtual objects (0.67 interactions per second), stability is not considered equivalent to controller-tracking. A discrepancy that should be minimised before confident conclusions can be reached on interaction comparison, rather than tracking technology comparisons.

Despite tracking flaws, the *button-pushing* task imitating real world button pressing was found preferable with hand-tracking by 70.5% of 34 participant. The direct object manipulation tasks, *grasping* and *rotation*, did not find equivalent success, concluding that they are not suitable alternatives to controller based interaction.

The enforced situation (due to the Covid-19 crisis) of conducting the last two studies remotely introduced issues of data validation, environmental factors, sampling issues and limitation of methodological approach. Nevertheless, we reach the conclusion that in the current state of Oculus Quest's hand-tracking, two aspects are found to be suitable alternatives to controller-tracking. 1.) The default pointing implementation enhanced by a permanently visible ray cast and Bendcast, with awareness of the trade-offs involved in the underlying pointing implementation. 2.) Directly touching virtual buttons imitating the physical behaviour of real world buttons.

# Bibliography

[1] Anne-Marie Kanstrup & Bertelsen. *User Innovation Management*. Aalborg University Press, 2011.

[2] Giuseppe Caggianese, Luigi Gallo, and Pietro Neroni. “The Vive controllers vs. Leap motion for interactions in virtual environments: A comparative evaluation”. In: *International Conference on Intelligent Interactive Multimedia Systems and Services*. Springer. 2018, pp. 24–33.

[3] Jeffrey Cashion, Chadwick Wingrave, and Joseph J LaViola. “Optimal 3D selection technique assignment using real-time contextual analysis”. In: *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE. 2013, pp. 107–110.

[4] Manuela Chessa et al. “Grasping objects in immersive Virtual Reality”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 1749–1754.

[5] Lucas Figueiredo et al. “A comparative evaluation of direct hand and wand interactions on consumer devices”. In: *Computers & Graphics* 77 (2018), pp. 108–121.

[6] John Galvez. *IBM and Ubisoft® Partner to Bring Voice Command with Watson to Virtual Reality in Star Trek™: Bridge Crew*. 2017. URL: `https://uk.newsroom.ibm.com/2017-05-11-IBM-and-Ubisoft-R-Partner-to-Bring-Voice-Command-with-Watson-to-Virtual-Reality-in-Star-Trek-TM-Bridge-Crew`.

[7] Elisa Gusai et al. “Interaction in an immersive collaborative virtual reality environment: a comparison between leap motion and HTC controllers”. In: *International Conference on Image Analysis and Processing*. Springer. 2017, pp. 290–300.

[8] *Hand Tracking - Best Practices*. 2020. URL: `https://developer.oculus.com/design/hands-design-bp/`.

[9] Z. He et al. “A speech recognition-based interaction approach applying to immersive virtual maintenance simulation”. In: *2017 Second International Conference on Reliability Systems Engineering (ICRSE)*. 2017, pp. 1–5.

[10] Juan David Hincapié-Ramos et al. "Consumed endurance: a metric to quantify arm fatigue of mid-air interactions". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2014, pp. 1063–1072.

[11] Markus Höll et al. "Efficient physics-based implementation for realistic hand-object interaction in virtual reality". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2018, pp. 175–182.

[12] Ibm. *IBM/vr-speech-sandbox-vive*. 2019. URL: `https://github.com/IBM/vr-speech-sandbox-vive`.

[13] *Introducing Hand Tracking on Oculus Quest-Bringing Your Real Hands into VR*. 2019. URL: `https://www.oculus.com/blog/introducing-hand-tracking-on-oculus-quest-bringing-your-real-hands-into-vr/`.

[14] Adam Kendon and Laura Versante. "Pointing by hand in "Neapolitan"". In: *Pointing*. Psychology Press, 2003, pp. 117–146.

[15] Maryam Khademi et al. "Free-hand interaction with leap motion controller for stroke rehabilitation". In: *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. 2014, pp. 1663–1668.

[16] Yonjae Kim et al. "Kinect technology for hand tracking control of surgical robots: technical and surgical skill comparison to current robotic masters". In: *Surgical Endoscopy* 28.6 (2014), 1993–2000. DOI: `10.1007/s00464-013-3383-8`. URL: `https://link.springer.com/article/10.1007/s00464-013-3383-8`.

[17] Joseph J LaViola Jr et al. *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.

[18] Andy Lücking, Thies Pfeiffer, and Hannes Rieser. "Pointing and reference reconsidered". In: *Journal of Pragmatics* 77 (2015), pp. 56–79.

[19] Luzan. *luzan/Unity-Google-Spreadsheet*. 2017. URL: `https://github.com/luzan/Unity-Google-Spreadsheet`.

[20] Jonatan Martinez et al. "Touchless haptic feedback for supernatural vr experiences". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2018, pp. 629–630.

[21] Sven Mayer et al. "The Effect of Offset Correction and Cursor on Mid-Air Pointing in Real and Virtual Environments". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–13.

[22] Alec G Moore et al. "VOTE: A ray-casting study of vote-oriented technique enhancements". In: *International Journal of Human-Computer Studies* 120 (2018), pp. 36–48.

[23] Manuel Olbrich et al. "Virtual reality based space operations–a study of ESA's potential for VR based training and simulation". In: *International Conference on Virtual, Augmented and Mixed Reality*. Springer. 2018, pp. 438–451.

[24] Sai Akhil Penumudi et al. "The effects of target location on musculoskeletal load, task performance, and subjective discomfort during virtual reality interactions". In: *Applied Ergonomics* 84 (2020), p. 103010.

[25] Katrin Plaumann et al. "Towards accurate cursorless pointing: the effects of ocular dominance and handedness". In: *Personal and Ubiquitous Computing* 22.4 (2018), pp. 633–646.

[26] Daniel Plemmons and Paul Mandel. *Designing Intuitive Applications*. URL: `https://developer-archive.leapmotion.com/articles/designing-intuitive-applications`.

[27] Prashan Premaratne. *Human computer interaction using hand gestures*. Springer Science & Business Media, 2014.

[28] Nico Reski and Aris Alissandrakis. "Open data exploration in virtual reality: a comparative study of input technology". In: *Virtual Reality* (2019), pp. 1–22.

[29] Kai Riege et al. "The bent pick ray: An extended pointing technique for multi-user interaction". In: *3D User Interfaces (3DUI'06)*. IEEE. 2006, pp. 62–65.

[30] Marco Speicher et al. "Selection-based text entry in virtual reality". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–13.

[31] *The Lightweight Render Pipeline: Optimizing Real Time Performance*. 2019. URL: `https://blogs.unity3d.com/2018/02/21/the-lightweight-render-pipeline-optimizing-real-time-performance/`.

[32] *Thumbs Up: Hand Tracking Available on Oculus Quest This Week*. 2019. URL: `https://www.oculus.com/blog/thumbs-up-hand-tracking-now-available-on-oculus-quest/?locale=en_US`.

[33] Maggie Tillmann. *What is a ToF camera? The Time-of-flight sensor explained*. 2020. URL: `https://www.pocket-lint.com/phones/news/147024-what-is-a-time-of-flight-camera-and-which-phones-have-it`.

[34] *Using Single Pass Stereo Rendering and Stereo Instancing*. URL: `https://developer.oculus.com/documentation/unity/unity-single-pass/`.

[35] Daniel Wigdor and Dennis Wixon. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.

[36] Bob G Witmer and Michael J Singer. "Measuring presence in virtual environments: A presence questionnaire". In: *Presence* 7.3 (1998), pp. 225–240.

[37] Yuan Xie, Yuzhe Zhang, and Yiyu Cai. "Virtual Reality Engine Disassembly Simulation with Natural Hand-Based Interaction". In: *VR, Simulations and Serious Games for Education*. Springer, 2019, pp. 121–128.

# Appendix A

# Evaluation Materials

## A.1 Instructions Sheet - Hands vs. Controllers Study.

**Context**
We are studying the comparison between using controllers and hand tracking while trying to evaluate in which situations one is preferred.

We have developed two applications to evaluate this. One for hand tracking and one for controllers. Most of the data will be automatically recorded through the application, but we have a few short questionnaires after trying the applications.

**It is important to us that you only play each scenario through once!**
*If you only have time to play through 1 scenario, that is ok too! We would just appreciate it if you completed that 1 scenario in both the hand-tracking and controller-tracking applications.*

**Procedure**
Open the first application, "HandTracking".
Link: https://sidequestvr.com/#/app/742

- Use the first scene to get familiar with using hand-tracking in different task scenarios.
- You will be given a unique ID, **remember this** as you will be asked to type it into the questionnaire .
- Play through each scenario once.

Open the second application, "ControllerTracking"
Link: https://sidequestvr.com/#/app/741

- Use the first scene to get familiar with using controllers.

- Play through each scenario once.

Fill out consent form and demographics info: ->link
https://www.survey-xact.dk/LinkCollector?key=32MGNQY2SP9P
Fill out button-questionnaire: -> link
https://www.survey-xact.dk/LinkCollector?key=DG2VQRWJL13P
Fill out rotation-questionnaire: -> link
https://www.survey-xact.dk/LinkCollector?key=MD4VNRY2LN12
Fill out grasp/release-questionnaire: -> link
https://www.survey-xact.dk/LinkCollector?key=6C2YQMYPU131
Fill out miscellaneous questionnaire: -> link
https://www.survey-xact.dk/LinkCollector?key=TQHTA4QPJ231

Finished! Thanks a lot!
**Contact**
Discord server: https://discord.gg/gSQTSkm
Mail: sfjord14@student.aau.dk

## A.2 Questionnaires for Hands vs. Controllers Study

### A.2.1 Demographics

**Consent form for usage of numerical data**

In connection with students Christoffer Sand Kirk, Hassan Hameed, Simon Fjordvald and their project on 10th semester regarding hand-tracking interactions in Virtual Reality at Aalborg University, I give my consent that this session may be used as part of the students' education based on the following agreements and specifications (see question 2 & 3).

I give my consent for:
Numerical data from the gameplay session may be logged and saved (Yes/No)

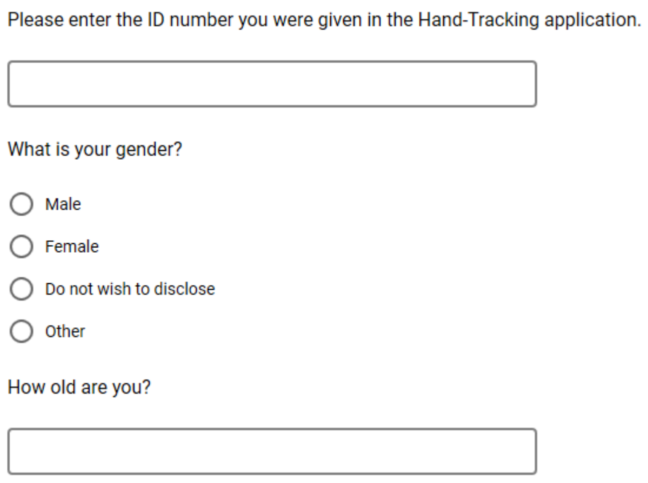
I give my consent for:
Written description and analysis of the material be used for the students' 10th semester project report in anonymised form (Yes/No)

Premise for agreeing to this consent form is that all materials will be stored safely and classified in regards to the demands from Datatilsynet. The materials will be stored until the project exam has been passed (June 2020 or August 2020), where it will be deleted afterwards. Everyone who has access to see the material are sworn to secrecy and must handle the data under regulations of confidentiality.

Withdrawal of consent is possible at any time, whereafter numerical data will be deleted.

If you have any questions regarding consent, or would like your data to be deleted, you can contact the facilitator/students on the following mail.
Simon Fjordvald: sfjord14@student.aau.dk

Figures (A.1, A.2) show the remaining demographics questions that were presented to participants.

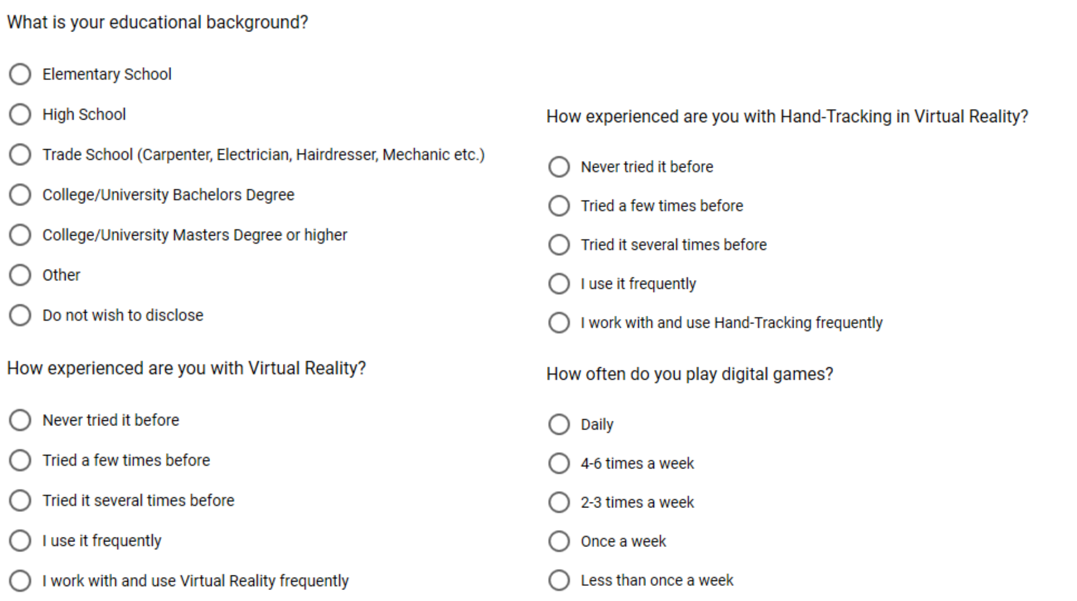

Please enter the ID number you were given in the Hand-Tracking application.

What is your gender?

- Male
- Female
- Do not wish to disclose
- Other

How old are you?

**Figure A.1:** Unique ID number given in the application - to cross-reference with the logged game play data during analysis. Secondly, foundational demographics questions on gender and age.



What is your educational background?

- Elementary School
- High School
- Trade School (Carpenter, Electrician, Hairdresser, Mechanic etc.)
- College/University Bachelors Degree
- College/University Masters Degree or higher
- Other
- Do not wish to disclose

How experienced are you with Virtual Reality?

- Never tried it before
- Tried a few times before
- Tried it several times before
- I use it frequently
- I work with and use Virtual Reality frequently

How experienced are you with Hand-Tracking in Virtual Reality?

- Never tried it before
- Tried a few times before
- Tried it several times before
- I use it frequently
- I work with and use Hand-Tracking frequently

How often do you play digital games?

- Daily
- 4-6 times a week
- 2-3 times a week
- Once a week
- Less than once a week

**Figure A.2:** Educational background as well as VR, Hand-Tracking and digital game experience, which affects task completion time.

### A.2.2 Grasp and Release Task Questionnaire

Please enter the ID number you were given in the hand tracking application.
<- Insert number into text entry field ->

**Question 1.**
Which method of interaction did you prefer for completing the Grasp and Release Task?

- Hands
- Controllers

**Question 2.**
Imagine you are in a game where you need to place items on the correct platform. Which method of interaction would you prefer in this scenario?

- Hands
- Controllers

**Question 3.**
Imagine you are in a training scenario where you have to pick up an item and move it to your workstation. Which method of interaction would you prefer in this scenario?

- Hands
- Controllers

**Question 4.**
Can you imagine contexts in which you would you prefer using controllers for grasping and releasing items? (Optional)

**Question 5.**
Can you imagine contexts in which you would prefer using controllers for grasping and releasing items? (Optional)

**Question 6.**
Do you have any other comments?

### A.2.3 Rotation Task Questionnaire

Please enter the ID number you were given in the hand tracking application.
<- Insert number into text entry field ->

**Question 1.**
Which method of interaction did you prefer for completing the Rotation task?

- Hands
- Controllers

**Question 2.**
Imagine you are in a game where you need to rotate dials to steer an object. Which method of interaction would you prefer in this scenario?

- Hands
- Controllers

**Question 3.**
Imagine you are in a training scenario where you have to rotate dials on a machine to adjust the temperature. Which method of interaction would you prefer in this scenario?

- Hands
- Controllers

**Question 4.**
Can you imagine contexts in which you would prefer using controllers for rotating objects? (Optional)

**Question 5.**
Can you imagine contexts in which you would prefer hands for rotating objects? (Optional)

**Question 6.**
Do you have any other comments?

### A.2.4 Button-pushing Task Questionnaire

Please enter the ID number you were given in the hand tracking application.
<- Insert number into text entry field ->

**Question 1.**
Which method of interaction did you prefer for completing the button-pushing task?

- Hands
- Controllers

**Question 2.**
Imagine you are in a game where you need to enter a four-digit code to open the door in front of you. Which method of interaction would you prefer in this scenario?

- Hands
- Controllers

**Question 3.**
Imagine you are in a training scenario where you have to push four buttons in sequence to start an industrial machine. Which method of interaction would you prefer in this scenario?

- Hands
- Controllers

**Question 4.**
Can you imagine contexts in which you would you prefer using controllers for pushing buttons? (Optional)

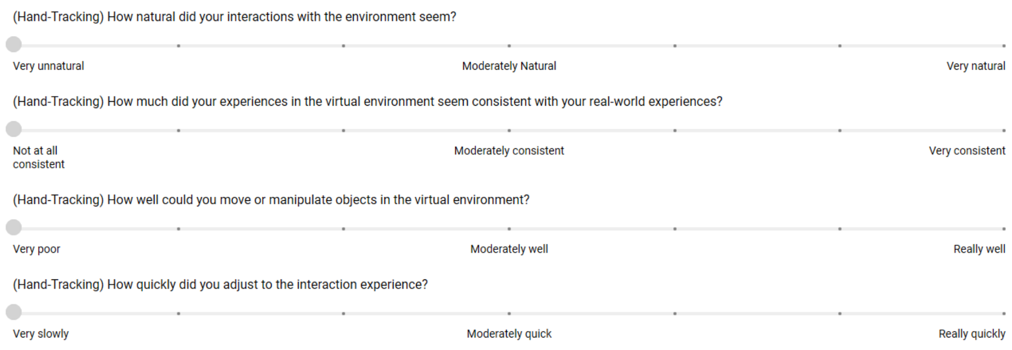
**Question 5.**
Can you imagine contexts in which you would you prefer using hands for pushing buttons? (Optional)

**Question 6.**
Do you have any other comments?

### A.2.5 Presence Questionnaire

Please enter the ID number you were given in the hand tracking application.
<- Insert number into text entry field ->

Figures (A.3, A.4 & A.5) shows the presence questions as they were presented to users post-test.

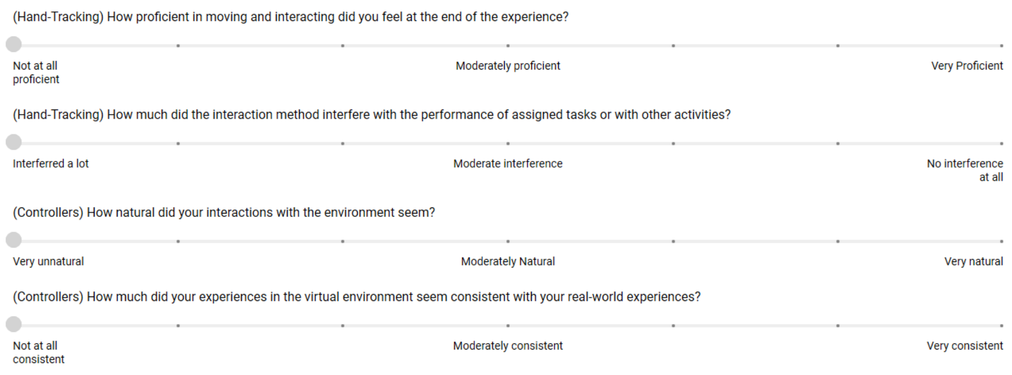

(Hand-Tracking) How natural did your interactions with the environment seem?
Very unnatural — Moderately Natural — Very natural

(Hand-Tracking) How much did your experiences in the virtual environment seem consistent with your real-world experiences?
Not at all consistent — Moderately consistent — Very consistent

(Hand-Tracking) How well could you move or manipulate objects in the virtual environment?
Very poor — Moderately well — Really well

(Hand-Tracking) How quickly did you adjust to the interaction experience?
Very slowly — Moderately quick — Really quickly

**Figure A.3:** Question 1-4 of the shortened presence questionnaire.

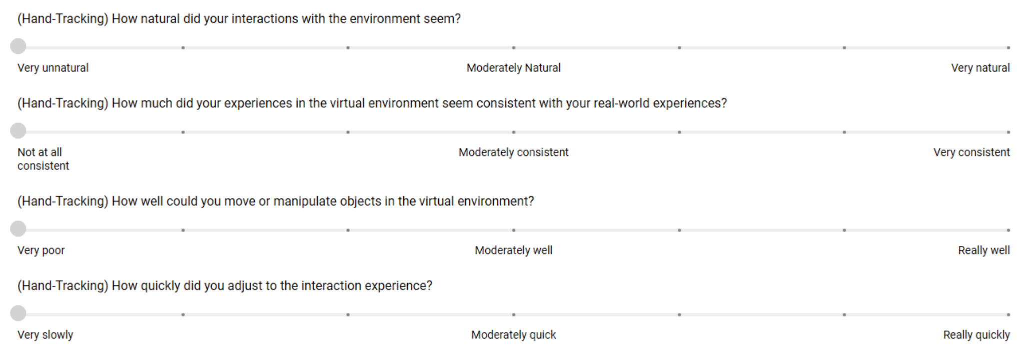

(Hand-Tracking) How proficient in moving and interacting did you feel at the end of the experience?
Not at all proficient — Moderately proficient — Very Proficient

(Hand-Tracking) How much did the interaction method interfere with the performance of assigned tasks or with other activities?
Interferred a lot — Moderate interference — No interference at all

(Controllers) How natural did your interactions with the environment seem?
Very unnatural — Moderately Natural — Very natural

(Controllers) How much did your experiences in the virtual environment seem consistent with your real-world experiences?
Not at all consistent — Moderately consistent — Very consistent

**Figure A.4:** Questions 5-8 of the shortened presence questionnaire.



(Hand-Tracking) How natural did your interactions with the environment seem?

Very unnatural · Moderately Natural · Very natural

(Hand-Tracking) How much did your experiences in the virtual environment seem consistent with your real-world experiences?

Not at all consistent · Moderately consistent · Very consistent

(Hand-Tracking) How well could you move or manipulate objects in the virtual environment?

Very poor · Moderately well · Really well

(Hand-Tracking) How quickly did you adjust to the interaction experience?

Very slowly · Moderately quick · Really quickly

**Figure A.5:** Question 9-12 of the shortened presence questionnaire.

## A.3 Witmer & Singer Presence Questionnaire

The full Witmer & Singer Questionniaire, from which we selected six questions in study two. Questions range on a likert scale of 1-7, with 1 indicating the worst experience and 7 indicating an excellent experience [36]. "Selected" text in bold after questions specify which questions were selected for testing.

- How much were you able to control events?
- How responsive was the environment to actions that you initiated (or performed)?
- How natural did your interactions with the environment seem? **(Selected)**
- How completely were *all* of your senses engaged?
- How much did the visual aspects of the environment involve you?
- How much did the auditory aspect of the environment involve you?
- How natural was the mechanism which controlled movement through the environment?
- How aware were you of events occurring in the real world around you?
- How aware were you of your display and control devices?
- How compelling was your sense of objects moving through space?
- How inconsistent or disconnected was the information coming from your various senses?
- How much did your experiences in the virtual environment seem consistent with your real-world experiences? **(Selected)**

- Were you able to anticipate what would happen next in response to the actions that you performed?
- How completely were you able to actively survey or search the environment using vision?
- How well could you identify sounds?
- How well could you localise sounds?
- How well could you actively survey or search the virtual environment using touch?
- How compelling was your sense of moving around inside the virtual environment?
- How closely were you able to examine objects?
- How well could you move or manipulate objects in the virtual environment? **(Selected)**
- How involved were you in the virtual environment experience?
- How distracting was the control mechanism? **(Partially selected, combined with interference question 4 items below this one.)**
- How much delay did you experience between your actions and expect outcomes?
- How quickly did you adjust to the virtual environment experience? **(Selected)**
- How proficient in moving and interacting with the virtual environment did you feel at the end of the experience? **(Selected)**
- How much did the visual display quality interfere or distract you from the performing assigned tasks or required activities? **(Partially selected, combined with question on control mechanism 4 items above this one.)**
- How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities? **(Partially selected, combined with previous question.)**
- Did you learn new techniques that enabled you to improve your performance?
- Were you involved in the experimental task to the extend that you lost track of time?